AMERICAN UNIVERSITY OF BEIRUT

ON ENLARGED KRYLOV SUBSPACE CONJUGATE GRADIENT METHOD: MSDO-CG

by

HIBA KHALED EL OWAYED

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Mathematics
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
August 2022

AMERICAN UNIVERSITY OF BEIRUT

ON ENLARGED KRYLOV SUBSPACE CONJUGATE GRADIENT METHOD: MSDO-CG

by

HIBA KHALED EL OWAYED

Approved by:

Dr. Sophie Moufawad, Assistant Professor Department of Mathematics	Advisor
Dr. Sabine El Khoury, Associate Professor Department of Mathematics	Member of Committee
Dr. Nabil Nassif, Professor Department of Mathematics	Member of Committee

Date of thesis defense: August 10, 2022

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

Student Name: El Owayed Hiba Khaled
Last First Middle

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of my thesis; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes:

- [x] As of the date of submission
- [ ] One year from the date of submission of my thesis.
- [ ] Two years from the date of submission of my thesis.
- [ ] Three years from the date of submission of my thesis.

August 17, 2022

Signature Date

Acknowledgements

First and foremost, I have to thank my thesis advisor, Dr. Sophie Moufawad. Without her assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. I would like to thank you very much and I'm very grateful for your support and understanding throughout all this year.

I would like to thank Dr.Nabil Nassif and Dr.Sabine El Khoury for being my committee members. Dr. Nassif was not only one of my committee members, he had his office door always to me whenever I needed guidance in my academics and my future career plan and I'm forever grateful for the wisdom he shared with me.

I must express my very profound gratitude to my friends , my family and to my boyfriend for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you!

Finally, my biggest thank you is to myself, my body and my mental health that suffered a lot during my master's journey and thesis writing with everything going on in Lebanon, yet they still fought to the last breath to reach another step in our dreams. I'm proud of myself!

The Thank you's are not over until I thank the most essential element in this project which is my laptop that bore all the long working hours and much longer testing hours.

An Abstract of the Thesis of

Hiba Khaled El Owayed for Master of Science
Major:Mathematics

Title: On Enlarged Krylov Subspace Conjugate Gradient Method: MSDO-CG

Solving systems of linear equations of the form $Ax = b$ has always been a challenge for scientists because the input matrix A is very large and sparse. These systems are derived mainly from the discretization of Partial Differential Equations (PDE) which are crucial and essential in most scientific fields, that's why they are usually solved using Krylov subspace methods such as : Conjugate Gradient (CG), Generalized Minimal Residual (GMRES), Bi-Conjugate Gradient (Bi-CG) and Bi-Conjugate Gradient Stabilized (Bi-CGStab). Even though these methods are efficient, they are ruled by Blas1 and Blas2 operations that are communication-bound when parallelized. To reduce communication, a new approach was to enlarge the Krylov subspace per iteration by a maximum of t vectors based on the domain decomposition of the graph of A. The enlarged Krylov subspace being a superset of the Krylov subspace will allow us to search for the solution of the system $Ax = b$ in it. Several variants of enlarged CG were introduced along with their s-step versions, and it is shown that an approximation to x is obtained in less iterations as compared to classical CG. But increasing t also means increasing the memory requirements and the possibility of some of the basis vectors becoming linearly dependent. Thus, t has to be relatively small, but not too small so that the number of iterations is reduced. In this thesis, we are mainly studying the possibility of flexibly varying the number of vectors added per iteration to the enlarged Krylov subspace, and its effect on the convergence of the enlarged CG methods : MSDO-CG and Modified MSDO-CG.

Table of Contents

Tables

Chapter 1

Introduction

In mathematics, a lot of problems solving relies on solving a linear system of the form $Ax = b$ where A is an $n \times n$ matrix and b an $n \times 1$ vector. Many methods were introduced to solve this kind of linear system and they can be categorized into two categories: Direct methods and Iterative methods.

Direct methods are methods that solves the linear system $Ax = b$ with a finite number of steps or operations and we end up with the exact solution of the system in exact precision or $\mathbb{R}$. LU decomposition, Cholesky decomposition and QR decomposition are famous direct methods for solving such system with A being dense or sparse . These methods works very well with small matrices and they are very straight forward. However, for large sparse matrices, they are no longer adequate because the obtained decomposition matrices will become denser than the input matrix .

A good alternative for direct methods are the iterative methods that compute a sequence of approximate solutions of the sytem $Ax = b$ by starting from an initial guess. These methods are commonly used with sparse large systems which may arise from discretizing partial differential equations because the direct methods are prohibitive in terms of memory when it comes to solving these systems. The Krylov subspace methods are among the most popular and practical iterative methods nowadays. These iterative methods aim to solve systems of linear equations $Ax = b$ by finding a sequence of vectors $x_1, x_2, \ldots, x_k$ from the corresponding spaces:

$$x_0 + \mathcal{K}_i(A, r_0), \quad i = 1, \ldots, k$$

where

$$\mathcal{K}_i(A, r_0) = span\{r_0, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\}$$

x_0 is the initial guess and r_0 is the initial residual.

In this thesis, we start by Chapter 2 where we introduce the Krylov subspace and its properties. Then, we discuss the Krylov subspace methods: Conjugate Gradient (CG), Generalized Minimum Residual (GMRES) and the Bi-Conjugate Gradient method (Bi-CG) along with their algorithms.
In Chapter 3, we present a new approach of enlarging the Krylov subspace by a maximum of t vectors per iteration as introduced in [1] and [2], then we introduce the A-orthonormalization process which will be used in the discussion of the enlarged Krylov subspace CG methods: SRE-CG [2], MSDO-CG [2](which is based on MSD-CG [3]) and its variant Modified MSDO-CG [4].
In Chapter 4, we introduce a flexible version of MSDO-CG and Modified MSDO-CG to reduce memory storage and hopefully reduce time till convergence. Then, the flexible versions are tested for different inputs and will be compared to the original versions.
In Chapter 5, we conclude the promising behavior of the flexible versions in terms of runtime and number of iterations.

Chapter 2

Krylov Subspace Methods

The Krylov Subspace Methods are named after the applied mathematician and naval engineer Aleksey Krylov which was introduced in his paper in 1931. These methods use a sequence of vectors to minimize the errors and get an approximate solution of the system $Ax = b$.

In this chapter, we define the Krylov subspaces and list some of their properties in section 2.1. In section 2.2, we present the projections methods and show that they are ruled by two conditions: the Subspace condition and the Petrov-Galerkin method. Section 2.3, 2.4 and 2.5 are about discussing some the projection methods, namely: the Conjugate Gradiant method (CG), the generalized minimal residual method (GMRES), the Bi-Conjugate method (Bi-CG) and the Bi-Conjugate stabilized method.

2.1 Krylov subspace and its properties

A Krylov subspace of order i is generated by a $n \times n$ matrix A and a $n \times 1$ vector f and it is spanned by the vectors of the Krylov sequence:

$$\mathcal{K}_i(A, f) = span\{f, Af, A^2f, \dots, A^{i-1}f\} \tag{2.1}$$

This subspace satisfies two main properties :

- $\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \cdots \subseteq \mathcal{K}_{k_{max}}$
- A$\mathcal{K}_k \subseteq \mathcal{K}_{k+1}$

where $\mathcal{K}_i$ is of dimension at most i and k_{max} is the grade of the Krylov subspace which we define below.

Definition 2.1.1. *The grade of a krylov subspace $\mathcal{K}_i$(A,f) noted μ is a positive integer which refers to the dimension of the largest subspace generated by A and f.*

Lemma 2.1.2. *Let μ be the grade. Then $\mathcal{K}_\mu$ is invariant under A and $\mathcal{K}_m = \mathcal{K}_\mu$ for all $m \geqslant \mu$.*

2.2 Krylov subspaces methods

The Krylov subspace methods are polynomial iterative methods that seek a sequence of vectors : $x_1, x_2, \ldots, x_k$ from the corresponding spaces :

$$x_0 + \mathcal{K}_i(A, r_0), i = 1, 2, \ldots, k \tag{2.2}$$

that should approximate the solution of our linear system $Ax = b$, where:

- x_0 is the initial iterate
- $r_0 = b - Ax_0$ is the initial residual
- $\mathcal{K}_i$(A,r_0) is the Krylov subspace of order i generated by A

Once the residual vector $r_i = b - Ax_i$ is small enough, an approximated solution is achieved.
Notation: $\mathcal{K}_i(A, r_0) \equiv \mathcal{K}_i$

2.2.1 *Krylov projection methods*

The Krylov projection methods compute a sequence of approximate solutions x_k of our liner system $Ax = b$, from an affine subspace $x_0 + \mathcal{K}_k$ (k=1,2,...). This sequence is obtained by imposing the Petrov-Galerkin condition on the k^{th} residual $r_k = b - Ax_k$ which is:

$$r_k \perp \mathcal{L}_k$$

where $\mathcal{L}_k$ is a well defined subspace in $\mathbb{R}^n$ or $\mathbb{C}^n$. $\mathcal{L}_k$ can be the same as the Krylov subspace $\mathcal{K}_k$ or can be different than it, the choice of the subspace $\mathcal{L}_k$ leads to different methods. Thus, the different Krylov projection methods are ruled by two main conditions :

1. The subspace condition : $x_k \in x_0 + \mathcal{K}_k$
2. The Petrov-Galerkin condition : $r_k \perp \mathcal{L}_k$

The Conjugate Gradiant method (CG) , the Generalized minimal residual (GMRES) method and the Bi-Conjugate method are indeed Krylov projection methods that we are going to discuss.

2.3 Conjugate Gradiant Method (CG)

The Conugate Gradiant method introduced by Hestenes and Stiefel [5] in 1952 is an iterative Krylov projection method based on taking $\mathcal{L}_k = \mathcal{K}_k$. This method is used to deal with symmetric positive definite matrices.

Definition 2.3.1. *In linear algebra, a $n \times n$ matrix A is said to be symmetric positive definite if :*

- $A^T = A$ *(Symmetric)*
- *the scalar $Z^T AZ$ is strictly positive for every non-zero column vector Z (Positive definite)*

Starting with an initial iterate x_0, the CG computes at the k^{th} iteration a new approximate solution $x_k = x_{k-1} + \alpha_k p_k$ over the corresponding space $x_0 + \mathcal{K}_k(A, r_0)$ where $p_k \in \mathcal{K}_k$ is the k^{th} search direction and α_k is the step along the search direction. The new approximate solution is obtained by minimizing $f(x) = \frac{1}{2}x^T Ax - b^T x$, because by minimizing $f(x)$ we are solving our symmetric positive definite linear system.
Since A is symmetric positive definite : $\nabla f(x) = Ax - b$, and the minimum of $f(x)$ is attained when $\nabla f(x) = 0$. Hence, the minimum of $f(x)$ occurs when $Ax = b$.
Due to the Petrov-Galerkin condition that projection methods has to abide by, the residuals must satisfy:

$$r_k^T y = 0, \forall y \in \mathcal{K}_k$$

Once we obtain x_k , either it's the exact solution of $Ax = b$ or we will need to determine a new search directory $p_{k+1} \neq 0$ to compute the new approximation $x_{k+1} = x_k + \alpha_{k+1}p_{k+1}$. This procedure will be repeated until convergence.

In the sections that follow, we discuss the properties of the residuals r_k, the search directions p_k and the convergence of the CG method.

2.3.1 *The residuals*

Proposition 2.3.2. *The residuals are orthogonal , i.e $r_i^T r_j = 0$, for $i \neq j$.*

Proof. By definition, the residual $r_k = b - Ax_k$ where $x_k \in \mathcal{K}_k$, so $r_k \in \mathcal{K}_{k+1}$. To obtain the recursion relation of r_k, we just replace x_k by its expression and we get:

$$r_k = b - Ax_k = b - A(x_{k-1} + \alpha_k p_k) = r_{k-1} - \alpha_k Ap_k$$

Moreover, we have the Petrov-Galerkin condition that $r_k \perp \mathcal{L}_k$. Therefore, $r_i^T r_j = 0$. Hence, the residuals form an orthogonal set. □

Corollary 2.3.3. *Suppose that the residuals are non zero, then* $\{r_0, r_1, \ldots, r_{k-1}\}$*forms an orthogonal basis for* $\mathcal{K}_k(A, r_0)$*.*

Proof. In proposition 2.3.2, we proved that the residuals form an orthogonal set which means the resiuals are linearly independant. And we have that

$$span\{r_0, r_1, \ldots, r_{k-1}\} \subseteq \mathcal{K}_k \tag{2.3}$$

because $r_i \in \mathcal{K}_k, \forall i \leq k-1$.
On the other hand we have:

$$dim(span\{r_0, r_1, \ldots, r_{k-1}\}) = k \leq dim(\mathcal{K}_k) \leq k \tag{2.4}$$

Then, by (2.3) and (2.4) we get that:

$$span\{r_0, r_1, \ldots, r_{k-1}\} = \mathcal{K}_k$$

□

2.3.2 *The search directions*

The search direction $p_k \in \mathcal{K}_k$ is defined according to the following recursion relation:

$$\begin{cases} p_1 = r_0 \\ p_{k+1} = r_k + \beta_{k+1} p_k \end{cases} \tag{2.5}$$

where p_1 is defined to be equal r_0 since the initial residual is equal to $-\nabla f(x_0)$. In this section, we present the main property of the search directions which is going to help us to compute α_{k+1} and β_{k+1}. But first, we define the A-conjugate concept because it's essential for describing the search directions.

Definition 2.3.4. *If A is an $n \times n$ positive definite matrix and x, $y \in \mathbb{R}^n$ then the inner product $< x, y >_A = x^T A y$ and if this inner product is equal to 0 then we say that x and y are A-conjugate.*
The corresponding norm to this definition is $||x||_A = \sqrt{x^T A x}$ is called the A-norm.

Theorem 2.3.5. *The Petrov-Galerkin condition ($r_k^T y = 0$) implies the A-orthogonality of the search directions i.e $p_k^T A p_i = 0, \forall i < k-1$.*

Proof. For $i < k-1$, we have by definition:

$$\begin{aligned} & p_k = r_{k-1} + \beta_k p_{k-1} \\ & \Longrightarrow p_k^T = r_{k_1}^T + \beta_k p_{k-1}^T \\ & \Longrightarrow p_k^T A p_i = r_{k-1}^T A p_i + \beta_k p_{k-1}^T A p_i \end{aligned}$$

by Petrov-Galerkin condition, we have $r_{k-1}^T A p_i = 0$.
Moreover, $r_{k-1}^T A p_i = r_{k-2}^T A p_i - \alpha_{k-1} p_{k-1}^T A p_i = 0$ with $r_{k-2}^T A p_i = 0$ because $i \leq k-2$. Thus, $p_{k-1}^T A p_i = 0$. Hence, $p_k^T A p_i = 0, \forall i < k-1$. □

2.3.3 *The time step* α *and* β

Our next goal is getting the expression for α_{k+1} and β_{k+1} :

- From the A-orthogonality of search directions, we have:

$$p_{k+1}^T A p_k = r_k^T A p_k + \beta_{k+1} p_k^T A p_k = 0$$
$$\implies \beta_{k+1} = -\frac{r_k^T A p_k}{p_k^T A p_k}$$

- By the Petrov-Galerkin condition, we have $r_k \perp \mathcal{K}_k$ and we notice that $p_k \in \mathcal{K}_k$. Hence we obtain $p_k^T r_k = 0$, we will use this information to compute α_{k+1} :

$$\begin{aligned}
&p_{k+1}^T r_{k+1} = 0\\
&p_{k+1}^T (b - A x_{k+1}) = 0\\
&p_{k+1}^T (b - A(x_k + \alpha_{k+1} p_{k+1})) = 0\\
&p_{k+1}^T (b - A x_k - A \alpha_{k+1} p_{k+1})) = 0\\
&p_{k+1}^T (r_k - \alpha_{k+1} A p_{k+1}) = 0\\
&p_{k+1}^T r_k - p_{k+1}^T \alpha_{k+1} A p_{k+1}) = 0\\
&\implies \alpha_{k+1} = \frac{p_{k+1}^T r_k}{p_{k+1}^T A p_{k+1}}
\end{aligned}$$

Lemma 2.3.6. *The step size* α_k *determined above gives the exact minimum of* $F(\alpha_{k+1}) = f(x_k + \alpha_{k+1} p_{k+1})$ *along the direction* p_k*, where* $f(x) = \frac{1}{2} x^T A x - x^T b$

Proof. We want to minimize $F(\alpha_{k+1})$, but first, let's compute it:

$$\begin{aligned}
F(\alpha_{k+1}) &= \frac{1}{2}(x_k + \alpha_{k+1} p_{k+1})^T A (x_k + \alpha_{k+1} p_{k+1}) - (x_k + \alpha_{k+1} p_{k+1})^T b\\
&= \frac{1}{2}[x_k^t A x_k + \alpha_{k+1} x_k^T p_{k+1} A + \alpha_{k+1} x_k p_{k+1}^T A + \alpha_{k+1}^2 p_{k+1} A p_{k+1}^T] - x_k^T b - \alpha_{k+1} p_{k+1}^T b\\
&= f(x_k) + \frac{1}{2}[\alpha_{k+1} x_k^T p_{k+1} A + \alpha_{k+1} x_k p_{k+1}^T A + \alpha_{k+1}^2 p_{k+1} A p_{k+1}^T] - \alpha_{k+1} p_{k+1}^T b\\
&= f(x_k) + \frac{1}{2}[\alpha_{k+1} x_k^T p_{k+1} A + \alpha_{k+1} x_k p_{k+1}^T A + \alpha_{k+1}^2 p_{k+1} A p_{k+1}^T] - \alpha_{k+1} p_{k+1}^T (r_k + A x_k)\\
&= f(x_k) + \frac{1}{2}[\alpha_{k+1} x_k^T p_{k+1} A - \alpha_{k+1} x_k p_{k+1}^T A] + \frac{1}{2}\alpha_{k+1}^2 p_{k+1} A p_{k+1}^T - \alpha_{k+1} p_{k+1}^T r_k\\
&= f(x_k) + \frac{1}{2}[\alpha_{k+1}^2 p_{k+1} A p_{k+1}^T] - \alpha_{k+1} p_{k+1}^T r_k \qquad \text{(because A is spd)}
\end{aligned}$$

Thus, $F'(\alpha_{k+1}) = \alpha_{k+1} p_{k+1} A p_{k+1}^T - p_{k+1}^T r_k$ and the minimum of $F(\alpha_{k+1})$ is given by $F'(\alpha_{k+1}) = 0$.

$\implies F'(\alpha_{k+1}) = \alpha_{k+1} p_{k+1} A p_{k+1}^T - p_{k+1}^T r_k = 0.$ Therefore, $\alpha_{k+1} = \dfrac{p_{k+1}^T r_k}{p_{k+1} A p_{k+1}^T}$ □

We try to **simplify** the expressions of α_{k+1} and β_{k+1} using what we have of definitions till now :
Recall that : $p_{k+1} = r_k + \beta_{k+1} p_k$ and that $r_k \perp \mathcal{K}_k$, so :

$$\alpha_{k+1} = \frac{p_{k+1}^T r_k}{p_{k+1} A p_{k+1}^T} = \frac{r_k^T r_k + \beta_{k+1} p_k^T r_k}{||p_{k+1}||_A^2} = \frac{||r_k||_2^2 + 0}{||p_{k+1}||_A^2} = \frac{||r_k||_2^2}{||p_{k+1}||_A^2}$$

Hence,

$$||p_k||_A^2 = \frac{||r_{k-1}||_2^2}{\alpha_k} \tag{2.6}$$

Now, let's move to β_{k+1} :
We will try to work on $-r_k^T A p_k$ by taking the expression of r_k and multiplying it by r_k^T :

$$\begin{aligned} & r_k = r_{k-1} - \alpha_k A p_k \\ & r_k^T r_k = r_k^T r_{k-1} - \alpha_k r_k^T A p_k \\ & r_k^T r_k = -\alpha_k r_k^T A p_k && \text{since the residuals are orthogonal} \\ & \implies -r_k^T A p_k = \frac{r_k^T r_k}{\alpha_k} = \frac{||r_k||_2^2}{\alpha_k} \end{aligned}$$

Hence, using (2.6) :

$$\beta_{k+1} = -\frac{r_k^T A p_k}{p_k^T A p_k} = \frac{\frac{||r_k||_2^2}{\alpha_k}}{||p_k||_A^2} = \frac{\frac{||r_k||_2^2}{\alpha_k}}{\frac{||r_{k-1}||_2^2}{\alpha_k}} = \frac{||r_k||_2^2}{||r_{k-1}||_2^2}$$

We present next the CG algorithm:

Algorithm 1 : CG Algorithm

Input: A, the $n \times n$ matrix; b, the $n \times 1$ right-hand side
Input: x_0, the initial guess or iterate
Input: ϵ, the stopping tolerance; k_{max}, the maximum allowed iterations
Output: x_k, the approximate solution of the system $Ax = b$

1: $r_0 = b - Ax_0$, $\rho_0 = ||r_0||_2^2$, $k = 1$
2: **while** $(\sqrt{\rho_{k-1}} > \epsilon ||b||_2$ and $k < k_{max})$ **do**
3:     **if** $(k = 1)$ **then** $p = r_0$
4:     **else** $\beta = \frac{\rho_{k-1}}{\rho_{k-2}}$ and $p = r + \beta p$
5:     **end if**
6:     $\omega = Ap$
7:     $\alpha = \frac{\rho_{k-1}}{p^t \omega}$
8:     $x = x + \alpha p$
9:     $r = r - \alpha \omega$
10:     $\rho_k = ||r||_2^2$
11:     $k = k + 1$
12: **end while**

2.3.4 *Convergence of the CG method*

In this section, we study the convergence of the CG method.

First, we start by a definition which will be useful later on.

Definition 2.3.7. *The Chebychev polynomial of the first kind of degree m is defined by :*

$$C_m(t) = cos[mcos^{-1}(t)], for -1 \leq t \leq 1 \tag{2.7}$$

and of its derivations , for $|t| \geq 1$, is :

$$C_m(t) = \frac{1}{2}[(t+\sqrt{t^2-1})^m + (t+\sqrt{t^2-1})^{-m}] \tag{2.8}$$

Now, let x_m be the approximate solution obtained at the mth step of the Conjugate Gradient algorithm, and $\tilde{x}$ the exact solution and define :

$$\eta = \frac{\lambda_{min}}{\lambda_{max} - \lambda_{min}}$$

where λ_{max} and λ_{min} are the maximum and the minimum eigenvalues of A respectively. Then,

$$||\tilde{x} - x_m||_A \leq \frac{||\tilde{x} - x_0||_A}{C_m(1+2\eta)} \tag{2.9}$$

with C_m is the Chebychev polynomial of degree m of the first kind as defined previously. Using that definition, we derive a slightly different formulation of (2.9) :

$$C_m(t) = \frac{1}{2}[(t+\sqrt{t^2-1})^m + (t+\sqrt{t^2-1})^{-m}] \geq \frac{1}{2}(t+\sqrt{t^2-1})^m$$

then,

$$C_m(1+2\eta) \geq \frac{1}{2}(1+2\eta+\sqrt{(1+2\eta)^2-1})^m \geq \frac{1}{2}(1+2\eta+2\sqrt{\eta(\eta+1)})^m$$

Now, let's look at $1+2\eta+2\sqrt{\eta(\eta+1)}$:

$$\begin{aligned}
1+2\eta+2\sqrt{\eta(\eta+1)} &= (\sqrt{\eta}+\sqrt{\eta+1})^2 \\
&= \frac{(\sqrt{\lambda_{min}}+\sqrt{\lambda_{max}})^2}{\lambda_{max}-\lambda_{min}} \\
&= \frac{\sqrt{\lambda_{min}}+\sqrt{\lambda_{max}}}{\sqrt{\lambda_{min}}-\sqrt{\lambda_{max}}} \\
&= \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}
\end{aligned}$$

where κ is the spectral condition number which is $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$.
Hence, (2.9) becomes :

$$||\tilde{x} - x_m||_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^m ||\tilde{x} - x_0||_A \tag{2.10}$$

Hence, the convergence is a function of the spectral condition number. So by the relation (2.10), the CG algorithm converges very rapidly when the condition number κ is almost one and thus, to speed-up the convergence of the CG solution to the exact solutiom we can precondition the matrix A by multiplying it by some matrix M such that the condition number of MA is almost one.
Preconditiong will be discussed in details at the end of this chapter.

2.4 Generalized Minimum Residual (GMRES)

The GMRES method was introduced by Youssed Saad and Martin H.Schultz in 1986 [6]. It's the projection method based on taking $\mathcal{L}_k = A\mathcal{K}_k$, in which $\mathcal{K}_k := \mathcal{K}_k(A, v_1)$ is the m-th Krylov subspace with $v_1 = \dfrac{r_0}{||r_0||_2}$.
Such a technique minimizes the residual norm $||b - Ax||_2$ over all vectors in $x_0 + \mathcal{K}_m$, i.e :

$$||r_k||_2 = ||b - Ax_k||_2 = min\{||b - Ax||_2, \forall x \in x_0 + \mathcal{K}_k\} \tag{2.11}$$

The minimum of the L_2 norm is zero which is equivalent to solving the system $Ax - b = 0$. This method works for any non-singular matrix and does not require it to be spd, like the Conjugate Gradiant method.
The residuals in the GMRES method do not form an orthonormal basis, hence Arnoldi's method is used to build an orthonormal basis for $\mathcal{K}_m$. This method will be discussed next in section 2.4.1 , then we briefly discuss the minimization procedure and the convergence of the method.

2.4.1 *Arnoldi's method*

Starting by a vector v_1, at each step, the Arnoldi's method multiplies the previous Arnoldi vector v_j by A and then orthonormalizes the resulting vector w_j against all previous v_j's using a standard Gram-Schmidt procedure.
Any vector $x \in x_0 + \mathcal{K}_m$ can be written as $x = x_0 + V_m y$, where y is an m-vector and V_m are the orthonormal basis vectors of the Krylov subspace $\mathcal{K}_m$ found using Arnoldi's method.
Now, let $\bar{H}_m$ be the $(m + 1) \times m$ Hessenberg matrix whose non-zero entries $h_{ij} =< w_j, v_i >$ are defined by Arnoldi-modified gram-Schmidt procedure and H_m the matrix obtained from the Hessenberg matrix by deleting its last row. So we have the following relations: (please refer to [7])

$$AV_m = V_m H_m + w_m e_m^T = V_{m+1} \bar{H}_m \tag{2.12}$$

$$V_m^T A V_m = H_m \tag{2.13}$$

2.4.2 *Minimizing the residual norm*

Denote by $J(y)$ the residual norm $||b - Ax||_2$. Since $x = x_0 + V_m y$, then:

$$J(y) = ||b - Ax||_2 = ||b - A(x_0 + V_m y)||_2 = ||b - Ax_0 + AV_m y||_2 = ||r_0 - AV_m y||_2$$

By Arnoldi's method, we have :

$$v_1 = \frac{r_0}{||r_0||_2} \quad and \quad \beta = ||r_0||_2$$

then $J(y)$ becomes:

$$J(y) = ||\beta v_1 - V_{m+1}\bar{H}_m y||_2 = ||V_{m+1}(\beta e_1 - \bar{H}_m y)||_2$$

and we have $||V_{m+1}||_2 = 1$ because the V_m's are the orthonormal basis vectors of $\mathcal{K}_m$, therefore :

$$J(y) = ||\beta e_1 - \bar{H}_m y||_2$$

The GMRES approximation is the unique vector of $x_0 + \mathcal{K}_m$ which minimizes the residual norm, i.e

$$x_m = x_0 + V_m y_m$$

where:

$$y_m = \min_y ||\beta e_1 - \bar{H}_m y||_2$$

The minimizer is inexpensive to compute as it requires the solution of $(m+1)\times m$ least square problems where m is typically small compared to the actual matrix size n.
To be able to solve the least square problems, we transform the Hessenberg matrix using plane rotations into a $n \times n$ upper triangular matrix and we denote it R_m, then we obtain the following:

$$||\beta e_1 - \bar{H}_m y||_2^2 = |\gamma_{m+1}|^2 + ||g_m - R_m y||_2^2$$

where g_m is a $m \times 1$ vector and $|\gamma_{m+1}|$ is the residual norm. The minimum is obtained when

$$||g_m - R_m y||_2^2 = 0$$

Therefore, the least square problem is converted into $m \times m$ matrix solver with R_m upper triangular which can be easily solved by backward substitution.
The GMRES process must be stopped once the residual norm $|\gamma_{m+1}|$ is small enough as shown in Algorithm 2.

Algorithm 2: GMRES

Input: A: an $n \times n$ matrix; b: $n \times 1$ right hand side vector.
Input: x_0 initial guess; tol the given tolerance.
Output: x_k an approximate solution of the system $Ax = b$.

1: $r_0 = b - Ax_0$, $\beta = ||r_0||_2$ and $v_1 = \frac{r_0}{\beta}$
2: **for** $j = 1, 2, \ldots, m$ **do**
3: $\quad w_j = Av_j$;
4: $\quad$ **for** $i = 1, dots, j$ **do**
5: $\quad\quad h_{i,j} = (w_j, v_i)$;
6: $\quad\quad w_j = w_j - h_{i,j}v_i$;
7: $\quad$ **end for**
8: $\quad h_{j+1,j} = ||w_j||_2$;
9: $\quad$ **if** $h_{j+1,j} \leq tol$ **then**
10: $\quad\quad m = j$;
11: $\quad\quad$ break (go to line 16)
12: $\quad$ **else**
13: $\quad\quad v_{j+1} = \frac{w_j}{h_{j+1,j}}$;
14: $\quad$ **end if**
15: **end for**
16: Define the $(m+1) \times m$ matrix $\bar{H}_m = \{h_{i,j}\}_{1 \leq i \leq m+1, 1 \leq j \leq m}$ and $W_m = w_{j, 1 \leq j \leq m}$;
17: Compute y_m the minimizer $||\beta e_1 - \bar{H}_m y||_2$ and $x_m = x_0 + W_m y_m$;

2.4.3 *Convergence of GMRES method*

GMRES method is known for its superlinear convergence behavior, i.e the rate of convergence seems to improve as the iterations proceed. Assuming that $||I - A||_2 \leq \beta < 1$, the following relation between the initial error and the m-th error is obtained in [8]:

$$||x_m - x^*||_2 \leq \beta^m ||x_0 - x^*||_2$$

where x^* is the exact solution, x_0 is the initial guess, and x_m is the m-th approximate solutions.

2.5 Bi-Conjugate Gradient method (Bi-CG)

The bi-Conjugate Gradient method (Bi-CG) was first introduced by Lanczos in 1952 [9] and reformulated as a Conjugate Gradient-like method by Fletcher in 1974 [10]. It is a Krylov projection method aiming to solve the linear system $Ax = b$ by introducing a shadow system $A^t\tilde{x} = \tilde{b}$ and solving the augmented system below:

$$\begin{bmatrix} A & 0 \\ 0 & A^t \end{bmatrix} \begin{bmatrix} x \\ \tilde{x} \end{bmatrix} = \begin{bmatrix} b \\ \tilde{b} \end{bmatrix} \Longleftrightarrow A'X = B'$$

where $\mathcal{L}_k = \tilde{\mathcal{K}}_k(A^t, \tilde{r}_0) = \tilde{\mathcal{K}}_k$ for the system $Ax = b$ and $\tilde{\mathcal{L}}_k = \mathcal{K}_k(A, r_0) = \mathcal{K}_k$ for the shadow system $A^t\tilde{x} = \tilde{b}$, with $\tilde{b}$ some unknown vector of size n. We let $\tilde{r}_0 = \tilde{b} - A^t\tilde{x}_0$. But since $\tilde{b}$ is unknown, $\tilde{r}_0$ is chosen to be equal to r_0. Thus, $\tilde{r}_0 = r_0 = b - Ax_0$ (just like CG)
Just like the CG method, the residuals have to abide by the Petrov-Galerkin conidition, i.e :

$$r_k \quad \perp \quad \tilde{\mathcal{K}}_k(A^t, \tilde{r}_0)$$

$$\&$$

$$\tilde{r}_k \quad \perp \quad \mathcal{K}_k(A, r_0)$$

The Bi-CG method solves both the system and the shadow system introduced, so at the k^{th} iteration we obtain two search directions p_k and the shadow one $\tilde{p}_k$, and two solutions x_k and the shadow one $\tilde{x}_k$ and two residuals r_k and the shadow one $\tilde{r}_k$. They have the following recurrence relations similar to those in the Conjugate Gradient method:

$$\begin{aligned}
x_k &= x_{k-1} + \alpha_k p_k \in \mathcal{K}_k \\
\tilde{x}_k &= \tilde{x}_{k-1} + \alpha_k \tilde{p}_k \in \tilde{\mathcal{K}}_k \\
r_k &= r_{k-1} - \alpha_k A p_k \in \mathcal{K}_{k+1} \\
\tilde{r}_k &= \tilde{r}_{k-1} - \alpha_k A^t \tilde{p}_k \in \tilde{\mathcal{K}}_{k+1} \\
p_k &= r_{k-1} + \beta_k p_{k-1} \in \mathcal{K}_k \\
\tilde{p}_k &= \tilde{r}_{k-1} + \beta_k \tilde{p}_{k-1} \in \tilde{\mathcal{K}}_k
\end{aligned}$$

Proposition 2.5.1. *Even though the Bi-CG solves two systems but they have the same step size α_k in x_k, $\tilde{x}_k$, r_k and $\tilde{r}_k$ and the same β_k in p_k and $\tilde{p}_k$.*

Proof. The Bi-CG solves the augmented system $A'X = B$.

- By the subspace condition at the k_{th} iteration, we have:

  $$X_k = X_{k-1} + \alpha_k P_k \in \mathcal{K}_k(A', R_0)$$

  which is equivalent to :

  $$\begin{pmatrix} x_k \\ \tilde{x}_k \end{pmatrix} = \begin{pmatrix} x_{k-1} \\ \tilde{x}_{k-1} \end{pmatrix} + \alpha_k \begin{pmatrix} p_k \\ \tilde{p}_k \end{pmatrix} \quad \& \quad R_0 = \begin{pmatrix} r_0 \\ \tilde{r}_0 \end{pmatrix}$$

  By splitting X_k we obtain the first two recurrence relations.

- The residual of the augmented system is:

$$
\begin{aligned}
R_k = \begin{pmatrix} r_k \\ \tilde{r}_k \end{pmatrix} &= B' - A'X_k \\
&= \begin{pmatrix} b \\ \tilde{b} \end{pmatrix} - \begin{pmatrix} A & 0 \\ 0 & A^t \end{pmatrix} \begin{pmatrix} x_k \\ \tilde{x}_k \end{pmatrix} \\
&= \begin{pmatrix} b \\ \tilde{b} \end{pmatrix} - \begin{pmatrix} A & 0 \\ 0 & A^t \end{pmatrix} \begin{pmatrix} x_{k-1} + \alpha_k p_k \\ \tilde{x}_{k-1} + \alpha_k \tilde{p}_k \end{pmatrix} \\
&= \begin{pmatrix} b \\ \tilde{b} \end{pmatrix} - \begin{pmatrix} Ax_{k-1} + \alpha_k A p_k \\ A^t \tilde{x}_{k-1} + \alpha_k A^t \tilde{p}_k \end{pmatrix} \\
&= \begin{pmatrix} r_{k-1} + \alpha_k A p_k \\ \tilde{r}_{k-1} + \alpha_k A^t \tilde{p}_k \end{pmatrix}
\end{aligned}
$$

By splitting R_k we obtain the 3^{rd} and the 4^{th} recurrence relations.

- The search direction $P_k \in \mathcal{K}_k(A', R_0)$ is chosen to be:

$$P_k = R_{k-1} + \beta_k P_{k-1}$$

which is equivalent to:

$$\begin{pmatrix} p_k \\ \tilde{p}_k \end{pmatrix} = \begin{pmatrix} r_{k-1} \\ \tilde{r}_{k-1} \end{pmatrix} + \beta_k \begin{pmatrix} p_{k-1} \\ \tilde{p}_{k-1} \end{pmatrix}$$

By splitting P_k we obtain the last two recurrence relations.
Therefore, the reason that α and β are the same for the shadow and the main vectors is that α and β are actually for the augmented system. □

Theorem 2.5.2. *The residual r_k and the shadow residual are related by the bi-orthogonality condition, i.e :*

$$(r_i)^t \tilde{r}_j = 0, \qquad for\; i \neq j \tag{2.14}$$

Proof. By the Petrov-Galerkin condition we have :

$$r_k \quad \perp \quad \tilde{\mathcal{K}}_k(A^t, \tilde{r}_0) \tag{2.15}$$

$$\&$$

$$\tilde{r}_k \quad \perp \quad \mathcal{K}_k(A, r_0) \tag{2.16}$$

By the first equation we have that $(r_k)^t \tilde{r}_j = 0$ for $j < k$. By the second equation we have that $(\tilde{r}_j)^t r_k = 0$ for $j > k$.
$\therefore (\tilde{r}_j)^t r_k = 0$ for $j \neq k$. □

Theorem 2.5.3. *The direction p_k and the shadow direction $\tilde{p}_k$ are related by the bi-conjugacy condition, i.e :*

$$(\tilde{p}_j)^t A p_k = 0 \quad k \neq j \tag{2.17}$$

Proof. By the equation (2.15) we have, for $j < k$:

$$r_{k-1} \perp \tilde{\mathcal{K}}_j(A^t, \tilde{r}_0) \quad \& \quad r_k \perp \tilde{\mathcal{K}}_j(A^t, \tilde{r}_0)$$

Thus,

$$(\tilde{v}_j)^t r_{k-1} = 0 \quad \& \quad (\tilde{v}_j)^t r_k = (\tilde{v}_j)^t r_{k-1} - \alpha_k (\tilde{v}_j)^t A p_k = 0$$

where $\tilde{v}_j \in \tilde{\mathcal{K}}_j(A^t, \tilde{r}_0)$. So $\alpha_k (\tilde{v}_j)^t A p_k = 0$. But, $\alpha_k \neq 0$, so $(\tilde{v}_j)^t A p_k = 0$. Now, let $\tilde{v}_j = \tilde{p}_j$.
$\therefore (\tilde{p}_j)^t A p_k = 0$ for $j < k$.
Similarly, using (2.16) we get $(\tilde{p}_j)^t A p_k = 0$ for $j > k$. □

Next, we find the expressions of α_k and β_k:

- At each iteration, the step α_k is chosen such that the bi-orthogonality condition (2.14) holds. Given that $r_k = r_{k-1} - \alpha_k A p_k$. We have :

  $$(\tilde{r}_{k-1})^t r_k = (\tilde{r}_{k-1})^t r_{k-1} - \alpha_k (\tilde{r}_{k-1})^t A p_k = 0$$

  Then,

  $$\alpha_k = \frac{(\tilde{r}_{k-1})^t r_{k-1}}{(\tilde{r}_{k-1})^t A p_k}$$

- For β_k, we use the bi-conjugacy condition (2.17) to find its expression. Given that $\tilde{p}_k = \tilde{r}_{k-1} + \beta_k \tilde{p}_{k-1}$. We have:

  $$(\tilde{p}_k)^t A p_{k-1} = (\tilde{r}_{k-1})^t A p_{k-1} + \beta_k (\tilde{p}_{k-1})^t A p_{k-1} = 0$$

  Then,

  $$\beta_k = -\frac{(\tilde{r}_{k-1})^t A p_{k-1}}{(\tilde{p}_{k-1})^t A p_{k-1}}$$

Algorithm 3: Bi-CG

Input: A: an $n \times n$ matrix; b: $n \times 1$ right hand side vector; x_0: initial guess; tol: the given tolerance.

Output: x_k: approximate solution of the system $Ax = b$.

```
r_0 = b - A x_0;
Choose r~_0 so that < r_0, r~_0 > ≠ 0;
Set p_1 := r_0 and p~_1 = r~_0;
for i = 1, ... till convergence do
    α_i = <r_{i-1}, r~_{i-1}> / <A p_i, p~_i>;
    x_i = x_{i-1} + α_i p_i;
    r_i = r_{i-1} - α_i A p_i   ,   r~_i = r~_{i-1} - α_i A^T p~_i;
    β_i = <r_i, r~_i> / <r_{i-1}, r~_{i-1}>;
    p_{i+1} = r_i + β_i p_i;
    p~_{i+1} = r~_i + β_i p~_i;
end for
```

2.5.1 *Convergence of the Bi-CG method*

The Bi-CG method is used to solve general systems that are not necessarily symmetric. But, in case A is spd then the Bi-CG method will be equivalent to the Conjugate Gradient method and it will arrive at the same solution with one inconvenience, is that p_k and r_k will be computed twice.
Otherwise, Bi-CG has the problem of converging irregularly often. It exhibits an unstable behavior which may slow down the speed of convergence. This is the main reason why Van Der Vorst introduced the Bi-CG Stabilized method in 1992 which is a variant of the Bi-CG method. Bi-CG Stab attempts to smoothen the erratic convergence of Bi-CG by multiplying the residual at the k-th iteration by a polynomial to minimize the norm of the residual. For more information about Bi-CG Stab please refer to [11].

2.6 Preconditioning

Krylov subspace methods are well founded theoretically, but they are all likely to suffer from slow convergence upon application because the dimension of the system we're working with is very large. That's where the idea of preconditioner was introduced.
Preconditioning is the process of transforming the original system $Ax = b$ into a new system, with the same solution and a much faster rate of convergence of the iterative method. This will make the preconditioned system easier to solve and requiring less iterations.
The first step in preconditioning is to find the preconditioning matrix M which should be cheap to construct and apply, non-singular and resembles A in some

sense.
After choosing the suitable preconditioning matrix M there's 3 ways of preconditioning:

- Left preconditioning, i.e applying the preconditioning matrix to the left :

$$M^{-1}Ax = M^{-1}b$$

- Right preconditioning, i.e applying the preconditioning matrix to the right :

$$AM^{-1}y = b \quad , \quad x \equiv M^{-1}y$$

- Mixed preconditioning, i.e multiplying from both sides, the preconditioning matrix will be of the form:

$$M = M_L M_R$$

  where M_L and M_R are triangular matrices. In this situation, the preconditioning will be split :

$$M_L^{-1}AM_R^{-1}y = M_L^{-1}b \ , \ x \equiv M_R^{-1}y$$

  In the case of symmetric matrices,the mixed preconditioning is often used to preserve the symmetry of the matrix in the linear system but it is not the only way to do it.

There are a lot of techniques to produce the preconditioning matrix M, these techniques belong to 4 essential groups:

- Preconditioning based on the splitting of the matrix A, where $A = M - N$ like Jacobi and Gauss-Seidel.
- Complete or incomplete factorization of A, like Incomplete LU factorization
- Approximation of A^{-1},i.e $M \approx A^{-1}$
- Reordering of equations or unknowns like the Domain decomposition.

For more info on preconditioning procedure please refer to [12]
In this thesis, we discuss the preconditioned Conjugate Gradient.

2.6.1 ***Preconditioned Conjugate Gradient***

As previously seen, the matrix A is symmetric positive definite and that's why we also start with an spd preconditioning matrix M.
We then compute its Cholesky factorization:

$$M = LL^T$$

which yields to the equation:

$$L_M^{-1} A L_M^{-T} y = L_M^{-1} b \quad , \quad x = L_M^{-T} y \tag{2.18}$$

the next step is to solve the system for the new matrix $B = L_M^{-1} A L_M^{-T}$ which is also spd but with a "better" condition number, hence our preconditioning purpose is achieved. After finding y, we get x by backward substitution.

If we don't want to use the mixed preconditioning technique, an alternative approach is available which is replacing the usual Euclidean inner product in the CG algorithm by the M-inner product [12]
Interesting fact is that the obtained iterates in these two techniques are identical.
Below is the Split preconditioner Conjugate Gradient algorithm.

Algorithm 4: Split Preconditioner CG

Input: A: an $n \times n$ matrix; b: $n \times 1$ right hand side vector; x_0: initial guess; tol: the given tolerance.
Input: M the preconditioner matrix
Output: x_k: approximate solution of the system $Ax = b$.

$r_0 = b - Ax_0$;
Set $\tilde{r}_0 = L^{-1} r_0$ and $p_0 = L^{-T} \tilde{r}_0$;
for $j = 0, 1, \ldots$ till convergence **do**
    $\alpha_j = \frac{<\tilde{r}_j, \tilde{r}_j>_A}{<Ap_j, p_j>_A}$;
    $x_{j+1} = x_j + \alpha_j p_j$;
    $\tilde{r}_{j+1} = \tilde{r}_j - \alpha_j L^{-1} A p_j$;
    $\beta_j = \frac{<\tilde{r}_{j+1}, \tilde{r}_{j+1}>_A}{<\tilde{r}_j, \tilde{r}_j>_A}$;
    $p_{j+1} = L^{-T} \tilde{r}_{j+1} + \beta_j p_j$;
end for

Chapter 3

Enlarged Krylov Subspace Methods

In this chapter, we introduce the new enlarged Krylov subspace which is based on domain decomposition. By enlarging the Krylov subspace by at most t vectors per iteration, we obtain the enlarged Krylov subspace methods that converges faster than the classical ones discussed before when solving the system $Ax = b$. In section 3.1 and 3.2, we introduce the Enlatged Krylov subspace and talk about its main properties and methods. Then, in section 3.3, we discuss the A-orthonormalization process which is based on the A-norm and is a vital component in the Enlarged Krylov methods and then compare it with the orthonormalization process which is based on the L2 norm. In section 3.4, we present one of the Enlarged Krylov methods which is the Short Recurrence Enlarged Conjugate gradient method (SRE-CG) introduced in [2]. In section 3.5, we present the second Enlarged Krylov subspace method Multiple Search Direction with Orthonormzlization Conjugate gradient method (MSDO-CG) introduced in [2] which is based on the MSD-CG method [3] and dicuss its modified version introduced in [4]. In the last section, section 3.6, we discuss the preconditioning process which makes the large dimensional system we're working with originally easier to solve and requiring less iterations.

3.1 The enlarged Krylov subspace

The enlarged Krylov methods consists of enlarging the original Krylov space by at most t vectors per iteration. Using a graph partitioning method, the domain of the $n \times n$ matrix A is partitioned into t distinct subdomains. If we consider the partitioning of the index domain $\delta = \{1, 2, \ldots, n\}$ into t subdomains, then $\delta = \bigcup_{i=1}^{t} \delta_i$. Then, the residual vector will also split into t vectors. We define $T_i(x)$ to be the operator that projects the $n \times 1$ vector x into the ith subdomain δ_i, so it replaces all the vector elements that are not in the ith subdomain by zero. Then, we define $T(x)$ to be an operator that transforms the $n \times 1$ vector x into t vectors of size $n \times 1$ that correspond to the projection of x onto the subdomains δ_i for $i = 1, 2, \dots, t$, i.e the output will be $(T_1(x), T_2(x), \dots, T_t(x))$. At each iteration k, the residual vector is multiplied by A. So at a step k, the new vectors are $\{A^{k-1}T_1(r_0), \dots, A^{k-1}T_t(r_0)\}$.These vectors make up the new basis vectors of the enlarged Krylov subspace $\mathcal{K}_{t,k}$, where t corresponds to the number of partitions of the matrix A and k the iteration. Such vectors along with the previous vectors span the enlarged Krylov subspace.

Definition 3.1.1. *Let*

$$\begin{aligned}\mathcal{K}_{t,k} &= span\{T_1(r_0), \dots, T_t(r_0), AT_1(r_0), AT_2(r_0), \dots, AT_t(r_0), \dots, A^{k-1}T_1(r_0), \dots, A^{k-1}T_t(r_0)\} \\ &= span\{T(r_0), AT(r_0), A^2T(r_0), \dots, A^{k-1}T(r_0)\}\end{aligned}$$

be an enlarged Krylov subspace of dimension z, $k \leq z \leq tk$, generated by the matrix A and the vector r_0, and associated to a given partition defined by δ_i, for $i = 1, 2, \dots, t$.

3.1.1 *Properties of the enlarged Krylov subspace*

Theorem 3.1.2. *The Krylov subspace $\mathcal{K}_k$ is a subset of the enlarged Krylov subspace $\mathcal{K}_{t,k}$, i.e $\mathcal{K}_k \subset \mathcal{K}_{t,k}$*

Proof. Let $y \in \mathcal{K}_k$ with $\mathcal{K}_k = span\{r_0, Ar_0, \dots, A^{k-1}r_0\}$. Then,

$$y = \sum_{j=0}^{k-1} a_j A^j r_0 = \sum_{j=0}^{k-1} a_j A^j R_0 * \mathbb{1}_t = \sum_{j=0}^{k-1}\sum_{i=1}^{t} a_j A^j T_i(r_0) \in \mathcal{K}_{t,k}$$

because we have $r_0 = R_0 * \mathbb{1}_t = [T_1(r_0), T_2(r_0), \dots, T_t(r_0)] * \mathbb{1}_t$ □

Kylov subspace methods seek a solution $x_k \in x_0 + \mathcal{K}_k$. A corollary of the previous theorem state that we can search for an approximate solution $x_k \in x_0 + \mathcal{K}_{t,k}$. So our goal is to search for the solution in the enlarged Krylov subspace and approximate it in less iterations.

Next, we'll be stating some properties of the enlarged Krylov subspace without going into their proofs. For detailed proofs, please refer to [4].

- Let k_{max} be the smallest integer such that $\mathcal{K}_{t,k_{max}} = \mathcal{K}_{t,k_{max+q}}$, for all $q > 0$. So, $\forall k < k_{max}$, the dimension of the subspaces $\mathcal{K}_{t,k}$ and $\mathcal{K}_{t,k+1}$ is strictly increasing by a number i_k and i_{k+1} respectively, with $1 \leq i_{k+1} \leq i_k \leq t$.

- By definition of the enlarged Krylov subspace

$$\mathcal{K}_{t,k+1} = \mathcal{K}_{t,k} + span\{A^kT_1(r_0), A^kT_2(r_0), \dots, A^kT_t(r_0)\}$$

Remark 3.1.3. *The sum above is not direct because the intersection is not always empty. The following property tackle this issue.*

- If $A^k T_v(r_0) \in \mathcal{K}_{t,k}, \forall 1 \le v \le t$, then we have $A^{k+q}T_i(r_0) \forall 1 \le i \le t$ and $\forall q > 0$.

- Let d_{max} be such that $\mathcal{K}_{d_{max}} = \mathcal{K}_{d_{max}+q}$ and k_{max} such that $\mathcal{K}_{t,k_{max}} = \mathcal{K}_{t,k_{max}+q}$, for $q > 0$. Then $k_{max} \le d_{max}$.
  The above property explains why we prefer having the solution in the enlarged Krylov space, because its grade is less that the Krylov subspace so it is reached faster.

- The solution of our system $Ax = b$ belongs to $x_0 + \mathcal{K}_{t,k_{max}}$, where $\mathcal{K}_{t,k_{max}+q} = \mathcal{K}_{t,k_{max}}$, for $q > 0$.

3.2 Enlarged Krylov subspace methods

We define our new enlarged Krylov projection methods based on CG by the subspace $\mathcal{K}_{t,k}$, these methods are similar to Krylov subspace methods and follow two mains conditions:

- Subspace condition:
$$x_k \in x_0 + \mathcal{K}_{t,k}$$

- Orthogonality condition:
$$r_k \perp \mathcal{K}_{t,k} \iff (r_k)^t y = 0, \forall y \in \mathcal{K}_{t,k}$$

The enlarged Krylov subspace CG methods minimize the function $f(x)$ defined in section 2.3 over the new subspace $x_0 + \mathcal{K}_{t,k}$. Recall that:

$$f(x) = \frac{1}{2} x^t A x - b^t x$$

Theorem 3.2.1. *If $r_k \perp \mathcal{K}_{t,k}$, then $f(x_k) = min\{f(x), \forall x \in x_0 + \mathcal{K}_{t,k}\}$*

Proof. By the orthogonality condition, we have $r_k \perp \mathcal{K}_{t,k}$

$$\begin{aligned} \implies (r_k)^t y = 0, \qquad \forall y \in \mathcal{K}_{t,k} \\ (b - Ax_k)^t y = 0, \forall y \in \mathcal{K}_{t,k} \\ b^t y - (x_k)^t A y = 0, \forall y \in \mathcal{K}_{t,k} \end{aligned}$$

We let $y = x_k - x_0 \in \mathcal{K}_{t,k}$

$$\begin{aligned}
&\implies (x_k)^t A(x_k - x_0) - b^t(x_k - x_0) = 0 \\
&\implies (x_k)^t Ax_k - b^t x_k = (x_k)^t Ax_0 - b^t x_0 \\
&\implies f(x_k) = \frac{1}{2}(x_k)^t Ax_k - b^t x_k = -\frac{1}{2}(x_k)^t Ax_k + (x_k)^t Ax_0 - b^t x_0
\end{aligned}$$

We still have to prove that $f(x) \geq f(x_k), \forall x \in x_0 + \mathcal{K}_{t,k}$ to reach our result.

$$\begin{aligned}
f(x) - f(x_k) &= \frac{1}{2}x^t Ax - b^t x - [-\frac{1}{2}(x_k)^t Ax_k + (x_k)^t Ax_0 - b^t x_0] \\
&= \frac{1}{2}x^t Ax - b^t z + \frac{1}{2}(x_k)^t Ax_k - (x_k)^t Ax_0, where\ z = x - x_0 \in \mathcal{K}_{t,k} \\
&= \frac{1}{2}x^t Ax - (x_k)^t Az + \frac{1}{2}(x_k)^t Ax_k - (x_k)^t Ax_0, since\ b^t z = (x_k)^t Az \\
&= \frac{1}{2}x^t Ax - (x_k)^t Ax + \frac{1}{2}(x_k)^t Ax_k \\
&= \frac{1}{2}(x - x_k)^t A(x - x_k) \geq 0, because\ A\ is\ positive\ definite.
\end{aligned}$$

□

Theorem 3.2.2. *x_k minimizes f over $x_0 + \mathcal{K}_{t,k}$ if and only if it minimizes $||x^* - x||_A$ over $x_0 + \mathcal{K}_{t,k}$, where x^* is the exact solution of the system $Ax = b$.*

Proof. Let

$$\begin{aligned}
g(x) &= ||x^* - x||_A^2 \\
&= (x^*)^t Ax^* - 2(x^*)^t Ax + x^t Ax \\
&= b^t x^* - 2b^t x^* + x^t Ax \\
&= b^t x^* + 2f(x)
\end{aligned}$$

The minimum is achieved when $g'(x) = 0$, i.e when $f'(x) = 0$. □

3.2.1 *Convergence*

The classical CG methods , converges in $\hat{L}$ iterations with $\hat{L} \leq n$, if $A \in \mathbb{R}^{n,n}$ is spd [5]. In addition, as we showed in the inequality (2.10), that the k-th error of CG is a function of the conditional number:

$$\hat{e}_k = ||x^* - \hat{x}_k||_A \leq 2\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^m ||\hat{e}_0||$$

where x^* is the exact solution of the system and $\hat{x}_k$ is the approximate solution at the k-th iteration.

By the Theorems 3.2.1 and 3.2.2, and if the k-th residual abide by the orthogonality condition, we have:

$$\begin{aligned} ||e_k||_A = ||x^* - x_k||_A &= min\{||x^* - x||_A, \forall x \in x_0 + \mathcal{K}_{t,k}\} \\ &\leq min\{||x^* - \hat{x}||_A, \forall \hat{x} \in x_0 + \mathcal{K}_k\},\ since\, \mathcal{K}_k \subset \mathcal{K}_{t,k} \\ &\leq ||\hat{e}_k||_A \end{aligned}$$

So, the new enlarged Krylov subspace CG methods will converge in L iterations such that $L \leq \hat{L} \leq n$, which makes its converges at least as fast as the classical CG method.

3.3 A-orthonormalization

Enlarged CG methods rely on the A-orthonormalization procedure, which is simply an orthonormalization using the A inner product $< .,. >_A$ instead of the L2 inner product $< .,. >$. Thus, we start by defining the orthonormalization process and the different methods used to perform it. Then, in section 3.3.1 and 3.3.2 we present and compare between the Classical Gram Schmidt orthononormalization and A-orthonormalization.

Definition 3.3.1. *Orthonormalization is the same as the orthogonalization process that finds an orthogonal basis of the span of given vectors with the addition of normalizing each vector by dividing the vector by its norm and then the resulting vectors will be unit vectors.*

We have different methods to perform orthogonalization:

- Gram-Schmidt process which uses projections
- Householder transformation which uses reflections
- Givens rotation

Orthonormalizing a tall and skinny matrix like the ones we are going to discuss later can be done using classical Gram Schmidt (CGS), modified Gram Schmidt (MGS) or a QR factorization like Householder factorization or based on Cholesky factorization.

3.3.1 *Classical Gram Schimdt (CGS)*

We will only discuss the orthonormalization using CGS process . For the other methods, please refer to [1].

Given an $n \times tk$ matrix Q_k with orthonormal column vectors, i.e $Q_k^t Q_k = I$

for all $j = 1, 2, \ldots, tk$, then the orthonormalization of the column vectors of the $n \times t$ matrix P_{k+1} against the vectors of Q_k is done by projecting the j-th column vector $P_{k+1}(:, j)$ onto all the $Q_k(:, i)$ vectors and substracting it from $P_{k+1}(; , j)$. For all $j = 1, 2, \ldots, t$, we let

$$\tilde{P}_{k+1}(; , j) = P_{k+1}(; , j) - \sum_{i=1}^{tk} (Q_k^t(; , i) P_{k+1}(; , j)) Q_k(; , i)$$

and we get then:

$$\begin{aligned}\tilde{P}_{k+1}^t(; , j) Q_k(; , c) &= P_{k+1}^t(; , j) Q_k(; , c) - \sum_{i=1}^{tk} (Q_k^t(; , i) P_{k+1}(; , j)) Q_k^t(; , i) Q_k(; , c) \\ &= P_{k+1}^t(; , j) Q_k(; , c) - (Q_k^t(; , c) P_{k+1}(; , j)) Q_k^t(; , c) Q_k(; , c) \\ &= 0\end{aligned}$$

for all $c = 1, 2, \ldots, tk$ because we have $Q_k^t Q_k = I$. Then, we normalize the vectors using the following relation:

$$\tilde{P}_{k+1}(:, j) = \frac{\tilde{P}_{k+1}(:, j)}{||\tilde{P}_{k+1}(:, j)||_2}$$

The orthonormalization of the vectors of P_{k+1} against each other is done as follows:

Algorithm 5: Orthonormalization using CGS of a tall and skinny matrix

Input: P_{k+1} the matrix to be orthonormalized
Output: P_{k+1}, the orthonormalized matrix ($P_{k+1}^t P_{k+1} = I$)

1: Let $\tilde{P}_{k+1} = P_{k+1}$
2: **for** $i = 1 : t$ **do**
3:     **for** $j = 1 : (i - 1)$ **do**
4:         $\tilde{P}_{k+1}(:, i) = \tilde{P}_{k+1}(:, i) - (P_{k+1}^t(:, j) P_{k+1}(:, i)) P_{k+1}(:, j)$
5:     **end for**
6:     $\tilde{P}_{k+1}(:, i) = \frac{\tilde{P}_{k+1}(:, i)}{||\tilde{P}_{k+1}(:, i)||_2}$
7: **end for**

3.3.2 *Classical Gram Schmidt A-orthonormalization*

The CGS A-orthonormalization has the same concept as the CGS orthonormalization but using the A-norm instead of the L2 norm and hence the projection expression of the j-th column onto the previous vectors is multiplied by A.
We present the algorithms for the A-orthonormalization of the vectors of P_{k+1} against previous vectors and against themselves using CGS.

The A-orthonormlization of P_{k+1} against the vectors of all the previous P_i's for $i < k + 1$ is represented in the following algorthim.

Algorithm 6: A-orthonormalization against previous vectors using CGS

Input: A, the $n \times n$ SPD matrix; Q_k the tk orthonormal vectors
Input: P_{k+1}, the t vectors to be A-orthonormalized against Q.
Output: $\tilde{P}_{k+1}$, the search directions A-orthonormalized against Q

1: Let $\tilde{P}_{k+1} = P_{k+1}$
2: **for** $j = 1 : t$ **do**
3:     **for** $i = 1 : tk$ **do**
4:         $\tilde{P}_{k+1}(:, j) = \tilde{P}_{k+1}(:, j) - (Q_k^t(:, i)AP_{k+1}(:, j))Q_k(:, i)$
5:     **end for**
6:     $\tilde{P}_{k+1}(:, j) = \frac{\tilde{P}_{k+1}(:,j)}{||\tilde{P}_{k+1}(:,j)||_A} = \frac{\tilde{P}_{k+1}(:,j)}{\sqrt{\tilde{P}_{k+1}^t(:,j)A\tilde{P}_{k+1}(:,j)}}$
7: **end for**

If we let $W_{k+1} = AP_{k+1}$ and $Q_k = [P_1, P_2, \ldots, P_k]$, we will be moving to a Block Classical Gram Schmidt (BGCS) version which can be used in addition to MGS and A-CholeskyBGS to A-orthonormalize P_{k+1} against previous vectors.

To A-orthonormalize a skinny $n \times t$ matrix P_{k+1} or compute its oblique QR factorization we have two classes. The first one requires the factorization of the matrix $A = B^tB$ using Cholesky or eigenvalue decomposition. The second class consists of avoiding any factorization of A, like Classical Gram Schmidt(CGS), CGS2 , Modified Gram Schmidt (MGS)
We present below the algorithm for A-orthonormalizing the vectors of P_{k+1} against each other with CGS.

Algorithm 7: A-orthonormalization against each other using CGS

Input: A, the $n \times n$ SPD matrix
Input: P_{k+1}, the search directions to be A-orthonormalized
Output: P_{k+1}, the A-orthonormalized search directions

1: Let $\tilde{P}_{k+1} = P_{k+1}$
2: **for** $i = 1 : t$ **do**
3:     **for** $j = 1 : (i - 1)$ **do**
4:         $\tilde{P}_{k+1}(:, i) = \tilde{P}_{k+1}(:, i) - (P_{k+1}^t(:, j)AP_{k+1}(:, i))P_{k+1}(:, j)$
5:     **end for**
6:     $\tilde{P}_{k+1}(:, i) = \frac{\tilde{P}_{k+1}(:,i)}{||\tilde{P}_{k+1}(:,i)||_A} = \frac{\tilde{P}_{k+1}(:,i)}{\sqrt{\tilde{P}_{k+1}^t(:,i)A\tilde{P}_{k+1}(:,i)}}$
7: **end for**

For more info about orthonormalization and A-orthonormalization, please refer to [1].

3.4 Short Recurrence Enlarged Conjugate gradient method (SRE-CG)

The SRE-CG [2] is a class of enlarged Krylov projection CG methods that aims to solve the system $Ax = b$ by having its approximate solution at the k-th iteration defined by: $x_k = x_{k-1} + Q_k\alpha_k \in x_0 + \mathcal{K}_{t,k}$, such that:

$$f(x_k) = min\{f(x), \forall x \in x_0 + \mathcal{K}_{t,k}\}$$

where $f(x) = \frac{1}{2}x^tAx - x^tb$, $Q_k\alpha_k \in \mathcal{K}_{t,k}$ and Q_k is an $n \times tk$ matrix containing the basis vectors of $\mathcal{K}_{t,k}$.
We will present three versions that have the same general derivations, but differ in the way the basis is constructed. Just like the classical CG method, our goal is to solve the system $Ax = b$ which is equivalent to minimizing $f(x)$. Since $f(x_k) = min\{f(x), \forall x \in x_0 + \mathcal{K}_{t,k}\}$, then:

$$f(x_k) = f(x_{k-1} + Q_k\alpha_k) = min\{f(x_{k-1} + Q_k\alpha), \forall\alpha \in \mathbb{R}^{tk}\}$$

At the k-th iteration, the x_k obtained either is our exact solution and we're done, or t new basis vectors and the new approximation x_{k+1} are computed. We repeat this procedure until convergence, so out next step is to find the recurrence relations of r_k and α_k.

3.4.1 *The residual* r_k

The residual is defined by $r_k = b - Ax_k$, with $x_k \in x_0 + \mathcal{K}_{t,k}$.So $r_k \in \mathcal{K}_{t,k+1}$, and we can obtain the recurrence relation of r_k by replacing x_k by its expression:

$$\begin{aligned} r_k &= b - Ax_k \\ &= b - A(x_{k-1} + Q_k\alpha_k) \\ &= r_{k-1} - AQ_k\alpha_k \end{aligned}$$

3.4.2 *Recurrence expression of* α_k

The choice of α_{k+1} at each iteration should be such that:

$$f(x_k) = min\{f(x_{k-1} + Q_k\alpha), \forall\alpha \in \mathbb{R}^{t(k+1)}\}$$

We let $F(\alpha) = f(x_{k-1} + Q_k\alpha)$, with $f(x) = \frac{1}{2}x^tAx - x^tb$. So we have:

$$\begin{aligned} F(\alpha) &= \frac{1}{2}(x_{k-1} + Q_k\alpha)^tA(x_{k-1} + Q_k\alpha) - (x_{k-1} + Q_k\alpha)^tb \\ &= f(x_{k-1}) + \frac{1}{2}[(x_{k-1})^tAQ_k\alpha + \alpha^t(Q_k)^tAx_{k-1} + \alpha^t(Q_k)^tAQ_k\alpha] - \alpha^t(Q_k)^tb \\ &= f(x_{k-1}) + \frac{1}{2}[(x_{k-1})^tAQ_k\alpha - \alpha^t(Q_k)^tAx_{k-1}] + \frac{1}{2}\alpha^t(Q_k)^tAQ_k\alpha - \alpha^t(Q_k)^tr_{k-1} \\ &= f(x_{k-1}) + \frac{1}{2}\alpha^t(Q_k)^tAQ_k\alpha - \alpha^t(Q_k)^tr_{k-1} \; \textit{because A is SPD} \end{aligned}$$

The minimum of $F(\alpha)$ is attained when $F'(\alpha) = 0$, hence:

$$F'(\alpha) = (Q_k)^t A Q_k \alpha - (Q_k)^t r_{k-1} = 0$$

Thus,

$$\alpha_k = ((Q_k)^t A Q_k)^{-1}((Q_k)^t r_{k-1})$$

Theorem 3.4.1. *With the assumption that $x_k = x_{k-1} + Q_k \alpha_k$, the orthogonality condition $r_k \perp \mathcal{K}_{t,k}$ is equivalent to having x_k as the minimum of $f(x)$ in $x_0 + \mathcal{K}_{t,k}$*

Proof. $\Longleftarrow$ As we showed above, the minimum of $F(\alpha)$ is given by $F'(\alpha) = (Q_k)^t A Q_k \alpha - (Q_k)^t r_{k-1} = 0$. And since we are given that x_k is the minimum, then $\alpha = \alpha_k$ and hence:

$$\begin{aligned} F'(\alpha) &= (Q_k)^t A Q_k \alpha_k - (Q_k)^t r_{k-1} = 0 \\ &= (Q_k)^t A Q_k \alpha_k - (Q_k)^t (r_k + A Q_k \alpha_k) = 0 \\ &= (Q_k)^t A Q_k \alpha_k - (Q_k)^t r_k - (Q_k)^t A Q_k \alpha_k = 0 \end{aligned}$$

$\therefore -(Q_k)^t r_k = 0 \Longrightarrow r_k \perp \mathcal{K}_{t,k}$

$\Longrightarrow$ We are going to use contradiction. Suppose $r_k \perp \mathcal{K}_{t,k}$ and x_k is not the minimum of $f(x)$ in $x_0 + \mathcal{K}_{t,k}$. Then $F'(\alpha_k) \neq 0$ and so $Q_k^t r_k \neq 0$ i.e $r_k \not\perp \mathcal{K}_{t,k}$. This contradicts our assumption. Thus, x_k is the minimum of $f(x)$. □

The basis vectors of $\mathcal{K}_{t,k}$ are $\{T(r_0), AT(r_0), \ldots, A^{k-1}T(r_0)\}$. We can either orthonormalize the basis vectors or A-orthonormalize it. If we orthonormalize the basis, we reach a Long recurrence enlarged CG version, where we have to solve at each iteration k, the system $\alpha_k = ((Q_k)^t A Q_k)^{-1}((Q_k)^t r_{k-1})$ of size $tk \times tk$ with Q_k the matrix containing the set of orthonormal basis vectors of $\mathcal{K}_{t,k}$ which makes this version expensive. For more details about the LRE-CG, please refer to [1]

On the other hand, if we A-orthonormalize the basis vectors then we have $Q_k^t A Q_k = I$ and hence α_k will be equal to $Q_k^t r_{k-1}$.

By the orthogonality condition, we have :

$$Q_{k-1}^t r_{k-1} = 0$$

So,

$$\begin{aligned} \alpha_k &= Q_k^t r_{k-1} \\ &= [Q_{k-1} W_k]^t r_{k-1} \\ &= [0_{t(k-1)}; W_k^t r_{k-1}] \\ &= [0_{t(k-1)}; \tilde{\alpha_k}] \end{aligned}$$

where W_k is the set of t newly computed vectors, and α_k is a $tk \times 1$ vector. Then, we obtain:

$$
\begin{aligned}
x_k &= x_{k-1} + Q_k \alpha_k \\
&= x_{k-1} + [Q_{k-1} W_k][0_{t(k-1)}; \tilde{\alpha_k}] \\
&= x_{k-1} + W_k \tilde{\alpha_k}
\end{aligned}
$$

And hence similarly, $r_k = r_{k-1} - AW_k \tilde{\alpha_k}$.
The procedure of A-orthonormalizing of W_k against $Q_{k-1} = [W_1 W_2 \dots W_{k-1}]$ goes as follows:

$$
\begin{aligned}
W_k &= AW_{k-1} - Q_{k-1} Q_{k-1}^t A(AW_{k-1}) \\
&= AW_{k-1} - \sum_{i=1}^{k-1} W_i W_i^t A(AW_{k-1}) \\
&= AW_{k-1} - W_{k-1} W_{k-1}^t A(AW_{k-1}) - W_{k-2} W_{k-2}^t A(AW_{k-1})
\end{aligned}
$$

because $(AW_i)^t (AW_{k-1}) = 0, \forall i < k-2$ as a result of the A-orthonormality of the basis vectors of $\mathcal{K}_{t,k}$. This version is called SRE-CG and it requires to store only the last $3t$ vectors, x_{k-1},r_{k-1} to define x_k and r_k.

We might face a loss of A-orthogonality between the last set of computed basis vectors and the first ones. This gave rise to SRE-CG2 where we A-orthonormalize W_k against all the basis vectors Q_{k-1}. It also requires to store the last $3t$ vectors, x_{k-1},r_{k-1} to define x_k and r_k. But, to be able to A-orthonormalize W_k against all the basis vectors, we also need to store all the tk basis vectors.

In case there is not enough memory to store all the tk basis vectors, we use a truncated version of the A-orthonormalization against previous vectors. In this truncated version, we A-orthonormalize W_k against a subset of $\{W_1, W_2, \dots, W_{k-3}\}$ along with W_{k-1} and W_{k-2}.
We give next the SRE-CG algorithm. The SRE-CG2 algorithm is the same as the SRE-CG algorithm except for line 7 where we will orthonormalize W_k against $W_i, \forall 1 \leq i \leq k-1$.

Algorithm 8: SRE-CG algorithm

Input: A, the $n \times n$ SPD matrix
Input: b, the $n \times 1$ right-hand side; x_0,the initial guess or iterate
Input: ϵ, the stopping tolerance, k_{max} the maximum allowed iterations
Output: x_k, the approximate solution of the system $Ax = b$

```
r_0 = b − Ax_0, ρ_0 = ||r_0||_2^2, k = 1
while (sqrt(ρ_{k−1}) > ε||b||_2 and k < k_max) do
    if k == 1 then
        Let W_1 = T(r_0), and A-orthonormalize its vectors
    else
        Let W_k = AW_{k−1}
        A-orthonormalize the vectors of W_k against the vectors of W_{k−1} and W_{k−2} if k > 2
        A-orthonormalize the vectors of W_k
    end if
    α̃_k = (W_k^t r_{k−1})
    x_k = x_{k−1} + W_k α̃_k
    r_k = r_{k−1} − AW_k α̃_k
    ρ_k = ||r_k||_2^2
    k = k + 1
end while
```

3.5 Multiple Search Direction with Orthogonalization Conjugate Gradient Method (MSDO-CG)

The MSD-CG method introduced by Gu et Al [3] is an enlarged Krylov subspace method that solves $Ax = b$. First, we have to partition the domain into t subdomains and then, define at each iteration k a search direction p_i^k on each of the subdomains such that $p_i^k(\delta_j) = 0, \forall j \neq i$.
We define the approximate solution at the k-th iteration to be $x_k = x_{k-1} + P_k\alpha_k$, with $P_k = [p_1^k\, p_2^k\, p_3^k \dots p_t^k]$ which is the matrix with all the t search directions at the kth iteration as its columns and α_k a vector of size t. Given an initial guess x_0, the residual is defined as $r_k = b - Ax_k$.
The approximate solution at the k-th iteration is $x_k = x_{k-1} + P_k\alpha_k$ such that:

$$f(x_k) = min\{f(x), \forall x \in \mathcal{K}_{t,k}\}$$

with P_k and α_k as defined above in MCD-CG. Our goal is to minimize $f(x)$ which is equivalent to solving $Ax = b$.
The residual is defined by $r_k = b - Ax_k$, with $x_k = x_{k-1} + P_k\alpha_k \in \mathcal{K}_{t,k}$, so we

have $r_k \in \mathcal{K}_{t,k+1}$ and :

$$\begin{aligned} r_k &= b - Ax_k \\ &= b - A(x_{k-1} + P_k\alpha_k) \\ &= b - Ax_{k-1} - AP_k\alpha_k \\ &= r_{k-1} - AP_k\alpha_k \end{aligned}$$

Remark 3.5.1. *If $r_k \perp \mathcal{K}_{t,k}$, then $(r_k)^t r_i = 0$ for all $i < k$ and $r_k \neq 0$. So, the residuals form an orthogonal set.*

At k-th step we have x_k, either we stop because it's our desired solution, or we compute t new domain search directions P_{k+1} and new approximation $x_{k+1} = x_k + P_{x+1}\alpha_{k+1}$. We repeat this procedure until convergence, Thus, the search directions are defined as:

- for $k = 1$, $p_i^1 = T_i(r_0)$
- for $k > 1$, $p_i^k = T_i(r_{k-1}) + \beta_i^k p_i^{k-1}$, for $i = 1, 2, \ldots, t$.

Where β_i^k is a scalar and T_i is an operator that projects a vector onto the subdomain δ_i. But, the P_k's are not A-orthogonal which means the orthogonality condition is not respected, the converse of this is proved next in theorem (3.5.2). Therefore, MSD-CG is not a projection method, and this what lead to the introduction of MSDO-CG.

The multiple search directions with orthogonalization CG (MSDO-CG) is an enlarged Krylov projection method similar to MSD-CG but ensure, at each iteration k, the A-orthogonalization of the search direction P_k against all previous $P_i, i = 1, 2, \ldots, k-1$.This step is crucial to respect the Petrov-Galerkin orthogonality condition $r_k \perp \mathcal{K}_{t,k}$, hence guaranteeing that this method converges at least as fast as classical CG.

Theorem 3.5.2. *If the orthogonality condition is satisfied then the block search directions are A-orthogonal, i.e:*

$$(r_k)^t y = 0,\ \forall y \in \mathcal{K}_{t,k} \Longrightarrow P_i^t A P_j = 0,\ \forall i \neq j\ \&\ i, j \leq k.$$

Proof. We have $P_i \in \mathcal{K}_{t,i}$ and $\mathcal{K}_{t,i} \subset \mathcal{K}_{t,i+1} \Longrightarrow P_i \in \mathcal{K}_{t,i+c}$ for $c \geq 0$
By the orthogonality condition, we have now: $r_{k-1}^t P_i = 0\ for\ i \leq k-1$ and :

$$\begin{aligned} r_k^t P_i &= 0 \\ (r_{k-1}^t - \alpha_k^t P_k^t A)P_i &= 0 \\ r_{k-1}^t P_i - \alpha_k^t P_k^t A P_i &= 0 \end{aligned}$$

since the first term is equal zero and by definition, $\alpha_k \neq 0$, then:

$$P_k^t A P_i = 0\ for\ i \leq k-1$$

□

So our next step is to find the recurrence expressions of P_k, α_k and β_k.

3.5.1 ***The domain search direction*** P_k

Just like MCD-CG, the domain search direction $P_k = [p_1^k\, p_2^k\, p_3^k \dots p_t^k]$, with $p_i^1 = T_i(r_0)$ and $p_i^k = T_i(r_{k-1}) + \beta_i^k p_i^{k-1} \in \mathcal{K}_{t,k}$, for $i = 1, 2, \dots, t$.
The recurrence expression of P_k is:

$$P_k = T(r_{k-1}) + P_{k-1} diag(\beta_k) \tag{3.1}$$

where $diag(\beta_k)$ is a $t \times t$ matrix with the vector β_k on the diagonal.
P_k defined above are not A-orthogonal to each other. That's why at each iteration k, the block vector P_k is A-orthonormalized against all previous P_i's and then the column vectors of P_k are A-orthonormalized against each other.This procedure is done directly once the P_k is defined, so this way we ensure that the orthogonal condition is valid.

3.5.2 ***Recurrence expression of*** α_{k+1} ***and*** β_{k+1}

The choice of α_{k+1} at each iteration should be such that:

$$f(x_{k+1}) = min\{f(x_k + P_{k+1}\alpha), \forall \alpha \in \mathbb{R}^t\}$$

We let $F(\alpha) = f(x_k + P_{k+1}\alpha)$, with $f(x) = \frac{1}{2}x^tAx - x^tb$. So we have:

$$\begin{aligned}
F(\alpha) &= \frac{1}{2}(x_k + P_{k+1}\alpha)^t A(x_k + P_{k+1}\alpha) - (x_k + P_{k+1}\alpha)^t b \\
&= \frac{1}{2}x_k^t A x_k + \frac{1}{2}[(x_k)^t A P_{k+1}\alpha + \alpha^t (P_{k+1})^t A x_k + \alpha^t (P_{k+1})^t A P_{k+1}\alpha] - \alpha^t (P_{k+1})^t b \\
&= f(x_k) + \frac{1}{2}[(x_k)^t A P_{k+1}\alpha - \alpha^t (P_{k+1})^t A x_k] + \frac{1}{2}\alpha^t (P_{k+1})^t A P_{k+1}\alpha - \alpha^t (P_{k+1})^t r_k \\
&= f(x_k) + \frac{1}{2}\alpha^t (P_{k+1})^t A P_{k+1}\alpha - \alpha^t (P_{k+1})^t r_k \; because\; A\; is\; SPD
\end{aligned}$$

The minimum of $F(\alpha)$ is attained when $F'(\alpha) = 0$, hence:

$$F'(\alpha) = (P_{k+1})^t A P_{k+1}\alpha - (P_{k+1})^t r_k = 0$$

Thus,

$$\alpha_{k+1} = ((P_{k+1})^t A P_{k+1})^{-1}((P_{k+1})^t r_k)$$

Assuming that the vectors of P_{k+1} are A-orthonormal (i.e $(P_{k+1})^t A P_{k+1} = I$)then α_{k+1} reduces to:

$$\alpha_{k+1} = (P_{k+1})^t r_k$$

As for β_{k+1}, we have:

$$\begin{aligned}
P_{k+1} &= T(r_k) + P_k diag(\beta_{k+1}) \\
\Longrightarrow P_k^t A P_{k+1} &= P_k^t A T(r_k) + P_k^t A P_k diag(\beta_{k+1})
\end{aligned}$$

Because we have $P_k^t A P_k = I$ from the A-orthonormality of the matrix P_k, then we should have $diag(\beta_{k+1}) = -P_k^t AT(r_k)$. But, the problem here is that $P_k^t AT(r_k)$ is not guaranteed to be a diagonal matrix. Hence, we chose:

$$\beta_{k+1} = -P_k^t A r_k$$

Algorithm 9: MSDO-CG algorithm

Input: A, the $n \times n$ symmetric positive definite matrix
Input:b, the $n \times 1$ right-hand side; x_0,the initial guess or iterate
Input: ϵ, the stopping tolerance, k_{max} the maximum allowed iterations
Output: x_k, the approximate solution of the system $Ax = b$

$r_0 = b - Ax_0, \rho = ||r_0||_2^2, k = 1$
Let $P_1 = T(r_0)$ and $W_1 = AP_1$
While ($\sqrt{\rho} > \epsilon ||b||_2$ and $k < k_{max}$) **do**
  if $k == 1$ **then**
    A-orthonormalize P_1 and update W_1
  else
    $\beta_k = -W_{k-1}^t r_{k-1}$
    $P_k = T(r_{k-1}) + P_{k-1} diag(\beta_k)$
    $W_k = AT(r_{k-1}) + W_{k-1} diag(\beta_k)$
    A-orthonormalize P_k against all P_i's and update W_k
    A-orthonormalize P_k and update W_k
  end if
  $\alpha_k = P_k^t r_{k-1}$
  $x_k = x_{k-1} + P_k \alpha_k$
  $r_k = r_{k-1} - W_k \alpha_k$
  $\rho = ||r_k||_2^2$
  $k = k + 1$
end while

3.5.3 *Modified MSDO-CG*

To reduce communication, the **s-step** methods were introduced as a way to restructure the Krylov methods algorithms. These methods compute s basis vectors per iteration and then use them to update the next approximate solution.
In the case of MSDO-CG, at each iteration k, t search directions are built and A-orthonormalized as mentioned before and then used to update the approximate solution. But, the construction of the search directions depends on the previously computed approximate solution which makes the process of merging s iterations of the MSDO-CG algorithm almost impossible. For that reason the modified version of MSDO-CG was introduced [4] and it works on building a modified enlarged Krylov basis instead of computing search directions.
In general, the modified enlarged Krylov subspace for a given s value is defined

as follows:

$$\begin{aligned}\bar{\mathcal{K}}_{t,k,s} = span\{&T(r_0), AT(r_0), \dots, A^{s-1}T(r_0),\\ &T(r_1), AT(r_1), \dots, A^{s-1}T(r_1),\\ &T(r_2), AT(r_2), \dots, A^{s-1}T(r_2),\\ &\vdots\\ &T(r_{k-1}), AT(r_{k-1}), \dots, A^{s-1}T(r_{k-1})\}\end{aligned}$$

The modified enlarged Krylov subspace $\bar{\mathcal{K}}_{t,k,s}$ is of dimension at most kst.
We are going to consider the case where $s = 1$, and so the modified enlarged Krylov subspace becomes:

$$\bar{\mathcal{K}}_{t,k} = span\{T(r_0), T(r_1), \dots, T(r_{k-1})\}$$

Theorem 3.5.3. *The Krylov subspace $\mathcal{K}_k$ is a subset of the modified enlarged Krylov subspace $\bar{\mathcal{K}}_{t,k,1}$, i.e $\mathcal{K}_k \subset \bar{\mathcal{K}}_{t,k,1}$*

Proof. The proof of this theorem is very similar to theorem 3.1.2 with suitable changes. □

At the kth iteration, the t vectors of $T(r_{k-1})$ are computed and stored in the $n \times t$ matrix V_k. Then, these t A-orthonormalized vectors are used to define $\tilde{\alpha}_k = V_k^t r_{k-1}$ and update $x_k = x_{k-1} + V_k\tilde{\alpha}_k$ and $r_k = r_{k-1} - AV_k\tilde{\alpha}_k$.
We present next the Modified MSDO-CG algorithm.

Algorithm 10: Modified MSDO-CG

Input: A, $n \times n$ SPD matrix; k_{max}, maximum allowed iterations
Input: b, $n \times 1$ right-hand side; x_0, initial guess; ϵ, stopping tolerance.
Output: x_k, approximate solution of the system $Ax = b$

$r_0 = b - Ax_0$, $\rho_0 = ||r_0||_2$, $\rho = \rho_0$, $k = 1$;
while ($\rho > \epsilon\rho_0$ and $k < k_{max}$) **do**
  if ($k == 1$) **then**
    A-orthonormalize $V_1 = T(r_0)$, let $Q = V_1$
  else
    A-orthonormalize $V_k = T(r_{k-1})$ against Q
    A-orthonormalize V_k, let $Q = [Q \;\; V_k]$
  end if
  $\tilde{\alpha}_k = V_k^t r_{k-1}$
  $x_k = x_{k-1} + V_k\tilde{\alpha}_k$
  $r_k = r_{k-1} - AV_k\tilde{\alpha}_k$
  $\rho = ||r_k||_2$
  $k = k + 1$
end while

Even though the s-step MSDO-CG algorithm for $s = 1$ is different than the MSDO-CG one , but they converge in the same number of iterations because they are theoretically equivalent

3.6 Preconditioning

In general, a system $Ax = b$ can be right, left or split preconditioned as we saw in section 2.6. However, in the case of conjugate gradient methods, the matrix A is SPD so the preconditioned matrix should also be SPD.
It is hard to be able to find a matrix M such that $M^{-1}A$ or AM^{-1} is SPD. That's why we go for split preconditioning assuming that $M = LL^t$, and then the obtained split preconditioned matrix $L^{-1}AL^{-t}$ is SPD.
We substitute the matrix A by the new preconditioned matrix $\hat{A} = L^{-1}AL^{-t}$ and then the preconditioned enlarged Krylov subspace corresponding to the system $\hat{A}\hat{x} = \hat{b}$ with $\hat{x} = y = L^t x$ and $\hat{b} = L^{-1}b$ becomes:

$$\mathcal{K}_{t,k} = \ span\{T(r_0), \hat{A}T(r_0), \ldots, \hat{A}^{k-1}T(r_0)\}$$

with $r_0 = L^{-1}(b - AL^{-t}y_0)$, b the $n \times 1$ vector, $y_0 = L^t x_0$ and x_0 the initial guess. So the subspace condition for the preconditioned enlarged CG methods becomes: $y_k \in y_0 + \mathcal{K}_{t,k}(\hat{A}, r_0)$, and the Petrov-Galerkin condition becomes $r_k \perp \mathcal{K}_{t,k}(\hat{A}, r_0)$ and everything discussed before follows for the new system $\hat{A}y = L^{-1}b$.
In the case of MSDO-CG, the split preconditioned MSDO-CG with CGS2+Â-CholQR Â-orthonormalization converges very well.
In case of the modified MSDO-CG, we'll use the split preconditioning i.e the system $\hat{A}\hat{x} = \hat{b}$ as defined above. Recall these relations from before that are adjusted for the new system:

$$\begin{aligned}
\hat{\alpha}_k &= \hat{V}_k^t \hat{r}_{k-1} \\
\hat{x}_k &= \hat{x}_{k-1} + \hat{V}_k \hat{\alpha}_k \\
\hat{r}_k &= \hat{r}_{k-1} - \hat{A}\hat{V}_k \hat{\alpha}_k
\end{aligned}$$

In this method, $\hat{V}_k$ is set to $[T(\hat{r}_{k-1})]$ and then Â-orthonormalized against all previous vectors. In addition, we have $\hat{V}_k^t \hat{A} \hat{V}_i^t = 0$ for $i \leq k$.
Note that we have $\hat{r}_k = \hat{b} - \hat{A}\hat{x}_k = L^{-1}b - L^{-1}AL^{-t}L^t x_k = L^{-1}(b - Ax_k) = L^{-1}r_k$.
We now derive the corresponding equations for x_k and r_k:

- $\hat{\alpha}_k = \hat{V}_k^t \hat{r}_{k-1} = \hat{V}_k^t L^{-1} r_k = (L^{-t}\hat{V}_k)^t r_k$
- $\hat{x}_k = L^t x_k = \hat{x}_{k-1} + \hat{V}_k \hat{\alpha}_k = L^t x_{k-1} + \hat{V}_k \hat{\alpha}_k \Longrightarrow x_k = x_{k-1} + (L^{-t}\hat{V}_k)\hat{\alpha}_k$
- $\hat{r}_k = L^{-1} r_k = \hat{r}_{k-1} - \hat{A}\hat{V}_k \hat{\alpha}_k = L^{-1} r_{k-1} - L^{-1}AL^{-t}\hat{V}_k \hat{\alpha}_k$
  $\Longrightarrow r_k = r_{k-1} - A(L^{-t}\hat{V}_k)\hat{\alpha}_k$

Let $V_k = L^{-t}\hat{V}_k$, then:

$$\begin{aligned}
\hat{\alpha}_k &= V_k^t r_k \\
x_k &= x_{k-1} + V_k \hat{\alpha}_k \\
r_k &= r_{k-1} - AV_k \hat{\alpha}_k
\end{aligned}$$

For the Â-orthonormalization, we require that $\hat{V}_k^t \hat{A} \hat{V}_i^t = 0$ for some values of $i \neq k$. But, we have:

$$\hat{V}_k^t \hat{A} \hat{V}_i^t = \hat{V}_k^t L^{-1} A L^{-t} \hat{V}_i^t = (L^{-t}\hat{V}_k)^t A (L^{-t}\hat{V}_i) = V_k^t A V_i$$

So, it is enough to A-orthonormalize $V_k = L^{-t}\hat{V}_k$ instead of Â-orthonormalizing $\hat{V}_k$. In our modified MSDO-CG method, we have:

$$V_k = L^{-t}[T(\hat{r}_{k-1})] = L^{-t}[T(L^{-1}r_k)]$$

This summarizes the preconditioning steps. More details are presented in [4] and [1] .

Chapter 4

Flexible MSDO-CG and Flexible Modified MSDO-CG

In this chapter, we are going to introduce a flexible version of MSDO-CG and modified MSDO-CG and test them.

4.1 MSDO-CG variants

As discussed before in section 3.5, MSDO-CG method solves the system $Ax = b$ by having its approximate solution at the kth step:

$$x_k = x_{k-1} + P_k \alpha_k$$

At each iteration, the approximate solution x_k minimizes the function $f(x)$ defined in section 2.3 over $x_0 + \mathcal{K}_{t,k}(A, r_0)$.
We have two methods that we're going to talk about next and they differ by the number of previous search directions to which the current t search directions are A-orthonormalized against.

- **MSDO-CG** that A-orthonormalize the t search directions at the kth iteration against all the previous search directions and this will require to store all the tk search direction

- **MSDO-CG(trunc)** that A-orthonormalize the t search directions at the kth iteration against the search directions computed in the last 'trunc' iterations and this will require to store at most $(trunc + 1) \times t$ search directions

The full MSDO-CG converges faster than the truncated version i.e it requires less iterations to find the solution. Yet the full MSDO-CG remains very costly memory-wise.
So our goal is to look into other variants of MSDO-CG which reduce memory requirement without affecting the number of iterations and if possible to reduce runtime.

4.2 Modified MSDO-CG variants

As discussed in section 3.5, the Modified MSDO-CG solves the system $Ax = b$ without relying on the search directions to update the approximate solution and the residual at each iteration. Instead, it works on building a modified enlarged Krylov basis:

$$\bar{\mathcal{K}}_{t,k} = span\{T(r_0), T(r_1), \ldots, T(r_{k-1})\}$$

We have two methods that we're going to talk about next and they differ by the number of previous basis vectors to which the current t basis vectors are A-orthonormalized against.

- **Modified MSDO-CG** that A-orthonormalize the t basis vectors at the kth iteration against all the previous basis vectors and this will require to store all the tk search direction

- **Modified MSDO-CG(trunc)** that A-orthonormalize the t basis vectors at the kth iteration against the basis vectors computed in the last 'trunc' iterations and this will require to store at most $(trunc+1) \times t$ basis vectors

The original Modified MSDO-CG converges faster than the truncated version i.e it requires less iterations to find the solution. Yet the classical Modified MSDO-CG remains very costly memory-wise.
So our goal is to look into other variants of MSDO-CG which reduce memory requirement without affecting the number of iterations and if possible to reduce runtime.

4.3 Flexible variants

The idea of the variants we're going to discuss for both methods is based on the observation that the norm of the residuals stagnates for partitions $t = 32, 64$ and 128. As a result, we can reduce the number of search directions or basis vectors produced into half. So, instead of producing **number of partitions** search directions or basis vectors per iteration, after some tolerance is reached, we compute half of this number.
The switch is done when the relative residual norm defined as $\frac{|||r_{k+1}||_2 - ||r_k||_2|}{||r_0||_2}$ becomes smaller than a chosen tolerance noted as switch tolerances.
Before presenting the algorithms, we are going to introduce some notations that will be used in them.

- T^t which is the operator that projects the vector over t subdomains δ_i where $\delta = \bigcup_{i=1}^{t} \delta_i$

- $T^{\frac{t}{2}}$ which is the operator that projects the vector over $\frac{t}{2}$ subdomains $\tilde{\delta_i}$ where $\delta = \bigcup_{i=1}^{\frac{t}{2}} \tilde{\delta_i}$ and $\tilde{\delta_i} = \delta_{2i-1} \bigcup \delta_{2i}$ for $i = 1, 2, \ldots, \frac{t}{2}$.

$$T^t(v) = \begin{pmatrix} * & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ * & 0 & 0 & 0 \\ 0 & * & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & * & 0 & 0 \\ \vdots & 0 & * & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & * \end{pmatrix}_{n \times t} \qquad T^{\frac{t}{2}}(v) = \begin{pmatrix} * & 0 \\ \vdots & \vdots \\ * & 0 \\ * & 0 \\ \vdots & \vdots \\ * & 0 \\ 0 & * \\ \vdots & \vdots \\ 0 & * \\ 0 & * \\ \vdots & \vdots \\ 0 & * \end{pmatrix}_{n \times \frac{t}{2}}$$

This change in the algorithm will cause reduction of the number of partitions and thus reduction of the vectors produced and stored per iteration after the switch. So if the switch is happening at the i-th iteration, the flexible MSDO-CG seeks the approximate solution at the k-th iteration with $k > i$ in

$$x_0 + \mathcal{K}_{t,i}(A, r_0) + \mathcal{K}_{\frac{t}{2},k-i}(A, r_i)$$

where :

$$\mathcal{K}_{\frac{t}{2},k-i}(A, r_i) = span\{T^{\frac{t}{2}}(r_i), AT^{\frac{t}{2}}(r_i), \ldots, A^{k-i-1}T^{\frac{t}{2}}(r_i)\}$$
$$\mathcal{K}_{t,i}(A, r_0) = span\{T^t(r_0), AT^t(r_0) \ldots, A^{i-1}T^t(r_0)\}$$

And the Modified MSDO-CG seeks the approximate solution at the k-th iteration with $k > i$ in

$$x_0 + \bar{\mathcal{K}}_{t,i}(A, r_0) + \bar{\mathcal{K}}_{\frac{t}{2},k-i}(A, r_i)$$

where:

$$\bar{\mathcal{K}}_{\frac{t}{2},k-i}(A, r_i) = span\{T^{\frac{t}{2}}(r_i), \ldots, T^{\frac{t}{2}}(r_{k-1})\}$$
$$\bar{\mathcal{K}}_{t,i}(A, r_0) = span\{T^t(r_0), \ldots, T^t(r_{i-1})\}$$

Algorithm 11: Flexible MSDO-CG algorithm

Input: A, the $n \times n$ symmetric positive definite matrix
Input:b, the $n \times 1$ right-hand side; x_0,the initial guess or iterate
Input: ϵ, the stopping tolerance, k_{max} the maximum allowed iterations, switchTol
Output: x_k, the approximate solution of the system $Ax = b$

counter = 0; tol1 = 1;
$r_0 = b - Ax_0, \rho_0 = ||r_0||_2^2, k = 1$
Let $P_1 = T^t(r_0)$ and $W_1 = AP_1$
While ($\sqrt{\rho} > \epsilon ||b||_2$ and $k < k_{max}$) **do**
  if $k == 1$ **then**
    A-orthonormalize P_1 and update W_1
  else
    if (tol1 < switchTol) and (counter== 0) **then**
      $P_k = T^{\frac{t}{2}}(r_{k-1})$;
      $W_k = AP_k$;
      counter= counter +1;
    else if (counter==0)
      $\beta_k = -W_{k-1}^t r_{k-1}$
      $P_k = T^t(r_{k-1}) + P_{k-1} diag(\beta_k)$
      $W_k = AT^t(r_{k-1}) + W_{k-1} diag(\beta_k)$
    else
      $\beta_k = -W_{k-1}^t r_{k-1}$
      $P_k = T^{\frac{t}{2}}(r_{k-1}) + P_{k-1} diag(\beta_k)$
      $W_k = AT^{\frac{t}{2}}(r_{k-1}) + W_{k-1} diag(\beta_k)$
    end if
    A-orthonormalize P_k against all P_i's and update W_k
    A-orthonormalize P_k and update W_k
  end if
  $\alpha_k = P_k^t r_{k-1}$
  $x_k = x_{k-1} + P_k \alpha_k$
  $r_k = r_{k-1} - W_k \alpha_k$
  $\rho_k = ||r_k||_2^2$
  $k = k + 1$
  tol1 $= \dfrac{|\sqrt{\rho_{k+1}} - \sqrt{\rho_k}|}{\sqrt{\rho_0}}$
end while

Algorithm 12: Flexible Modified MSDO-CG

Input: A, $n \times n$ SPD matrix; k_{max}, maximum allowed iterations
Input: b, $n \times 1$ right-hand side; x_0, initial guess; ϵ, stopping tolerance, switchTol
Output: x_k, approximate solution of the system $Ax = b$

counter= 0, tol1 = 1; $r_0 = b - Ax_0$, $\rho_0 = ||r_0||_2$, $\rho = \rho_0$, $k = 1$;
while ($\rho > \epsilon\rho_0$ and $k < k_{max}$) **do**
  if ($k == 1$) **then**
    A-orthonormalize $V_1 = T^t(r_0)$, let $Q = V_1$
  else
    if (tol1 < switchTol) and (counter== 0) **then**
      $V_k = T^{\frac{t}{2}}(r_{k-1})$;
      counter=counter +1;
    else if (counter==0)
      $V_k = T^t(r_{k-1})$
    else
      $V_k = T^{\frac{t}{2}}(r_{k-1})$
    end if
    A-orthonormalize V_k against Q
    A-orthonormalize V_k, let $Q = [Q\ \text{‘}V_k]$
  end if
  $\tilde{\alpha}_k = V_k^t r_{k-1}$
  $x_k = x_{k-1} + V_k \tilde{\alpha}_k$
  $r_k = r_{k-1} - AV_k \tilde{\alpha}_k$
  $\rho = ||r_k||_2$
  $k = k + 1$
  tol1 $= \dfrac{|\sqrt{\rho_{k+1}} - \sqrt{\rho_k}|}{\sqrt{\rho_0}}$
end while

4.4 Testing

To further investigate the consequences, we have tested this modification on large sparse matrices that were partitioned according to k-way partitioning [13], where the testing was performed sequentially and not in parallel.
The matrices referred to as NH2D and Sky3D, arise from boundary value problems of the convection diffusion equations (for a detailed description refer to [2]).
The Algorithm was implemented and tested in MATLAB R2021a on a PC with the following specifications: The operating system is Windows 10 Pro, Version 21H2 , Installed RAM is 8.00 GB and processor Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz.

We tested the methods MSDO-CG and flexible MSDO-CG. For the modified version, we tested Modified MSDO-CG and Flexible Modified MSDO-CG. The matrices were partitioned into t=2,4,8,16,32,64 and 128 partitions with a stopping tolerance 10^{-8}. The initial guess x_0 is chosen to be 0, the exact solution x is set as random vector using Matlab's rand function and $b = A * x$.

		CG	MSDO-CG		Modified MSDO-CG	
	t	k	k	time	k	time
NH2D	2	256	256	2.8668	256	2.8348
	4		206	3.8747	206	3.8553
	8		169	5.6934	169	5.9825
	16		139	10.5404	139	10.2358
	32		107	18.9626	107	18.8869
	64		77	31.0054	77	31.8225
	128		54	56.8148	54	54.5988
SKY3D	2	900	647	14.0199	647	13.8869
	4		426	12.4039	426	12.0988
	8		232	8.3176	233	7.7433
	16		133	7.1996	133	8.1222
	32		79	8.6446	79	9.2962
	64		50	12.5128	50	13.0755
	128		34	20.4538	34	18.527

Table 4.1: Comparison of the number of iteration k and time needed till convergence in the MSDO-CG and Modified MSDO-CG for matrices NH2D and Sky3D with number of partitions t =2,4,8,16, 32, 64 and 128

As expected from the previous discussion, MSDO-CG method and the Modified MSDO-CG method require less iterations to converge than the classical CG method.

		MSDO-CG		Flexible MSDO-CG 10^{-3}			10^{-5}			10^{-7}		
	t	k	time	k	sw	time	k	sw	time	k	sw	time
NH2D	2	256	2.866	267	14	1.210	257	58	1.247	257	143	1.970
	4	206	3.874	254	14	3.137	229	58	3.419	208	163	3.885
	8	169	5.693	204	14	4.176	183	69	4.812	170	145	5.607
	16	139	10.540	165	14	7.119	152	56	8.008	139	123	9.820
	32	107	18.962	135	14	11.654	117	57	14.391	107	99	17.742
	64	77	31.005	100	16	22.037	93	27	22.412	77	74	31.385
	128	54	56.814	72	13	38.956	54	47	49.750	54	53	55.541
SKY3D	2	647	14.019	751	18	6.243	744	35	6.239	656	505	12.149
	4	426	12.403	643	23	14.814	627	59	15.477	435	389	12.814
	8	232	8.317	412	17	13.008	383	50	12.863	338	98	11.860
	16	133	7.199	219	16	8.677	215	23	8.893	133	131	7.420
	32	79	8.644	120	14	7.883	79	69	8.149	79	77	8.966
	64	50	12.512	68	11	9.073	50	42	10.283	50	48	13.627
	128	34	20.453	41	11	12.802	37	18	15.042	34	33	18.994

Table 4.2: Comparison of the number of iteration k and time needed till convergence in the original MSDO-CG and flexible MSDO-CG version for matrices NH2D and Sky3D with number of partitions t =2,4,8,16, 32, 64 and 128 and three switchTol $10^{-3}, 10^{-5}$ and 10^{-7} .The switch iteration (sw) is reported for flexible MSDO-CG

Remark 4.4.1. *In table 4.2, the switch iteration (sw) is the iteration when the relative residual becomes less than the switchTol.*

We can see that when switching early on for switchTol $= 10^{-3}$ the number of iterations increased for both matrices NH2D and Sky3D compared to the original MSDO-CG, flexible MSDO-CG (switchTol= 10^{-5}) and flexible MSDO-CG (switchTol= 10^{-7}). As for convergence time, in both matrices and all partitions, the runtime of flexible MSDO-CG (switchTol= 10^{-3}) was significantly less than MSDO-CG except for Sky3D(4,8,16) .

When switching for switchTol $= 10^{-5}$, the number of iterations for the matrix NH2D increased in partitions 2,4,8,16, 32 and 64 but stayed the same for t=128. In Sky3D, the number of iterations remained almost the same for partitions 32,64 and 128 and increased for t=2,4,8 and 16. The time of convergence in both matrices and all partitions was less than the time of convergence of MSDO-CG except for Sky3D(4,8,16)

As for switchTol= 10^{-7}, in NH2D ,the number of iterations were almost the same as MSDO-CG.In SKY3D it was almost the same as MSDO-CG in partitions t=2 and 4,increased at t=8 and was exactly the same at t=16,32,64 and 128. This is because we were so close to the stopping tolerance 10^{-8} and so the needed enlarged Krylov subspace was built already. As for the time of convergence, it was very close to the time taken by MSDO-CG.

The increase in the number of iterations in flexible MSDO-CG for the switchTols 10^{-3} and 10^{-5} was expected as the dimension of the enlarged Krylov subspace is smaller compared to the original Krylov subspace, this is due to building half the number of basis vectors after the switch. Thus, finding the solution will require more iterations.The decrease in time isn't a surprise either and it is due to the reduction of the number of the basis vectors built.

Based on the results presented in table 4.2, the best switchTol would be 10^{-5}. Although it needed more iterations as compared to flexible MSDO-CG with switchTol = 10^{-7}, but taking into consideration the three variables: iterations till convergence, runtime and memory storage, it is better than switchTol= 10^{-3} and 10^{-7}.

		Modified MSDO-CG		Flexible Modified MSDO-CG 10^{-3}			10^{-5}			10^{-7}		
	t	k	time	k	sw	time	k	sw	time	k	sw	time
NH2D	2	256	2.883	267	14	1.028	257	58	1.272	257	143	1.941
	4	206	4.005	254	14	3.092	229	58	3.466	208	163	3.931
	8	169	5.730	204	14	4.310	183	69	4.805	170	145	5.803
	16	139	9.488	165	14	6.436	152	56	8.073	139	123	9.667
	32	107	17.531	135	14	11.607	117	57	14.179	107	99	16.944
	64	77	29.547	100	16	19.363	93	27	21.168	77	74	28.108
	128	54	49.509	72	13	34.106	54	47	43.396	54	53	48.653
SKY3D	2	647	13.903	751	18	6.212	744	35	6.161	670	460	11.589
	4	426	11.261	645	23	15.051	622	59	15.483	428	411	12.801
	8	233	9.254	414	17	12.832	383	50	12.856	233	230	8.997
	16	133	7.959	219	16	8.994	215	23	8.992	133	131	7.503
	32	79	8.408	120	14	8.037	79	69	7.967	79	77	9.256
	64	50	12.091	68	11	7.988	50	42	9.709	50	48	11.293
	128	34	18.527	41	11	11.537	37	18	13.696	34	33	18.154

Table 4.3: Comparison of the number of iteration k and time needed till convergence in the original Modified MSDO-CG and flexible Modified MSDO-CG version for matrices NH2D and Sky3D with number of partitions t =2,4,8,16, 32, 64 and 128 and three switchTol $10^{-3}, 10^{-5}$ and 10^{-7} .The switch iteration (sw) is reported for flexible Modified MSDO-CG

Remark 4.4.2. *In table 4.3, the switch iteration (sw) is the iteration when the relative residual becomes less than the switchTol.*

We can see that when switching early on for switchTol $= 10^{-3}$ the number of iterations increased for both matrices NH2D and Sky3D compared to the original Modified MSDO-CG, flexible Modified MSDO-CG (switchTol$= 10^{-5}$) and flexible Modified MSDO-CG (switchTol$= 10^{-7}$). As for convergence time, in both matrices and all partitions, the runtime of flexible MSDO-CG (switchTol$= 10^{-3}$) was significantly less than MSDO-CG except for the matrix Sky3D at the partitions t=4,8 and 16 .

When switching for switchTol $= 10^{-5}$, the number of iterations for the matrix NH2D increased in partitions 2,4,8,16, 32 and 64 but stayed the same for t=128. In Sky3D, the number of iterations remained almost the same for partitions 32,64 and 128 and increased for t=2,4,8 and 16. The time of convergence in both matrices and all partitions was less than the time of convergence of Modified MSDO-CG except for Sky3D(4,8,16)

As for switchTol$= 10^{-7}$, in NH2D ,the number of iterations were almost the same as Modified MSDO-CG.In SKY3D it was almost the same as Modified MSDO-CG in partitions t=2 and 4 and was exactly the same at t=8,16,32,64 and 128. This is because we were so close to the stopping tolerance 10^{-8} and so the needed enlarged Krylov subspace was built already. As for the time of convergence, it was very close to the time taken by Modified MSDO-CG .

The increase in the number of iterations in flexible Modified MSDO-CG for the switchTols 10^{-3} and 10^{-5} was expected as the dimension of the enlarged Krylov subspace is smaller compared to the original Krylov subspace, this is due to building half the number of basis vectors after the switch. Thus, finding the solution will require more iterations.The decrease in time isn't a surprise either and it is due to the reduction of the number of the basis vectors built.

Based on the results presented in table 4.3, the best switchTol would be 10^{-5}. Although it needed more iterations as compared to flexible MSDO-CG with switchTol $= 10^{-7}$, but taking into consideration the three variables: iterations till convergence, runtime and memory storage, it is better than switchTol$= 10^{-3}$ and 10^{-7}.

Chapter 5

Conclusion

In this thesis, a Flexible version of MSDO-CG and of Modified MSDO-CG were introduced. The flexible MSDO-CG and the Flexible Modified MSDO-CG showed their effectiveness in reducing time till convergence and even though the number of iterations was a bit higher than MSDO-CG and Modified MSDO-CG respectively, they remain acceptable and the results in general seems promising.
Future work would be to introduce new variant to MSDO-CG and Modified MSDO-CG and test its effectiveness.

Bibliography

[1] S. Moufawad, "Enlarged krylov subspace methods and preconditioners for avoiding communication," English, Ph.D. dissertation, 2014.

[2] L. Grigori, S. Moufawad, and F. Nataf, "Enlarged Krylov subspace conjugate gradient methods for reducing communication," *SIAM J. Matrix Anal. Appl.*, vol. 37, no. 2, pp. 744–773, 2016, ISSN: 0895-4798. DOI: 10.1137/140989492. [Online]. Available: https://doi-org.ezproxy.aub.edu.lb/10.1137/140989492.

[3] T. Gu, X. Liu, Z. Mo, and X. Chi, "Multiple search direction conjugate gradient method. I. Methods and their propositions," *Int. J. Comput. Math.*, vol. 81, no. 9, pp. 1133–1143, 2004, ISSN: 0020-7160. [Online]. Available: https://doi-org.ezproxy.aub.edu.lb/10.1080/00207160410001712305.

[4] S. M. Moufawad, "s-step enlarged Krylov subspace conjugate gradient methods," *SIAM J. Sci. Comput.*, vol. 42, no. 1, A187–A219, 2020, ISSN: 1064-8275. DOI: 10.1137/18M1182528. [Online]. Available: https://doi-org.ezproxy.aub.edu.lb/10.1137/18M1182528.

[5] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Research Nat. Bur. Standards*, vol. 49, 409–436 (1953), 1952, ISSN: 0160-1741.

[6] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.

[7] Y. Saad, S. for Industrial, and A. Mathematics, *Iterative methods for sparse linear systems*, 2nd. Philadelphia, Pa: Society for Industrial and Applied Mathematics, 2003.

[8] C. T. Kelley, S. for Industrial, and A. Mathematics, *Iterative methods for linear and nonlinear equations*, English. Philadelphia, Pa: Society for Industrial and Applied Mathematics, 1995, vol. 16.

[9] C. Lanczos, "Solution of systems of linear equations by minimized-iterations," *J. Research Nat. Bur. Standards*, vol. 49, pp. 33–53, 1952, ISSN: 0160-1741.

[10] R. Fletcher, "Conjugate gradient methods for indefinite systems," in, ser. Numerical Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 73–89.

[11] H. A. VAN DER VORST, "Bi-cgstab : A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems," *SIAM journal on scientific and statistical computing*, vol. 13, no. 2, pp. 631–644, 1992.

[12] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *J. Comput. Phys.*, vol. 182, no. 2, pp. 418–477, 2002, ISSN: 0021-9991. DOI: 10.1006/jcph.2002.7176. [Online]. Available: https://doi-org.ezproxy.aub.edu.lb/10.1006/jcph.2002.7176.

[13] G. Karypis and V. Kumar, "Metis 4.0: Unstructured graph partitioning and sparse matrix ordering system," English, 1998.