UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTMENT OF APPLIED MATHEMATICS

On Updating Preconditioners for the Iterative Solution of Linear Systems

Author:
Danny Joel GUERRERO FLORES

Advisers:
Dra. Juana CERDÁN
Dr. José MARÍN
Dr. José MAS

Valencia, Spain, May 2018

Certification

José Marín, José Mas and Juana Cerdán, professors at the Valencia Polytechnic University,

CERTIFY THAT:

The present thesis, “On Updating Preconditioners for the Iterative Solution of Linear Systems”, presented by **Danny Joel GUERRERO FLORES**, has been directed under our supervision in the Department of Applied Mathematics of the Valencia Polytechnic University and makes up him thesis to obtain the doctorate in Applied Mathematics.

As stated in the report, in compliance with the current legislation, we authorize the presentation of the above Ph.D thesis before the doctoral commission of the Valencia Polytechnic University, signing the present certificate in Valencia at May 8, 2018.

José Marín	José Mas	Juana Cerdán

Declaration of Authorship

I, Danny Joel GUERRERO FLORES, declare that this thesis titled, "On Updating Preconditioners for the Iterative Solution of Linear Systems" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"There is also a mysterious yet real phenomenon that we can experience: a reality which is a sign of another reality . . . as we reach the top of the ladder in our examination of something, either analytical or sentimental, our human nature tells us there is something else beyond. This step also defines the concept of 'sign'. . . . It is the vanishing point which lies in every human experience, i.e., a point that does not close, but leads further"

Luigi Giussani

Abstract

The main topic of this thesis is updating preconditioners for solving large sparse linear systems $Ax = b$ by using Krylov iterative methods. Two interesting types of problems are considered. In the first one is studied the iterative solution of non-singular, non-symmetric linear systems where the coefficient matrix A has a skew-symmetric part of low-rank or can be well approximated with a skew-symmetric low-rank matrix. Systems like this arise from the discretization of PDEs with certain Neumann boundary conditions, the discretization of integral equations as well as path following methods, for example, the Bratu problem and the Love's integral equation. The second type of linear systems considered are least squares (LS) problems that are solved by considering the solution of the equivalent normal equations system. More precisely, we consider the solution of modified and rank deficient LS problems. By modified LS problem, it is understood that the set of linear relations is updated with some new information, a new variable is added or, contrarily, some information or variable is removed from the set. Rank deficient LS problems are characterized by a coefficient matrix that has not full rank, which makes difficult the computation of an incomplete factorization of the normal equations. LS problems arise in many large-scale applications of the science and engineering as for instance neural networks, linear programming, exploration seismology or image processing.

Usually, incomplete LU or incomplete Cholesky factorization are used as preconditioners for iterative methods. The main contribution of this thesis is the development of a technique for updating preconditioners by bordering. It consists in the computation of an approximate decomposition for an equivalent augmented linear system, that is used as preconditioner for the original problem.

The theoretical study and the results of the numerical experiments presented in this thesis show the performance of the preconditioner technique proposed and its competitiveness compared with other methods available in the literature for computing preconditioners for the problems studied.

Resumen

El tema principal de esta tesis es el desarrollo de técnicas de actualización de precondicionadores para resolver sistemas lineales de gran tamaño y dispersos $Ax = b$ mediante el uso de métodos iterativos de Krylov. Se consideran dos tipos interesantes de problemas. En el primero se estudia la solución iterativa de sistemas lineales no singulares y antisimétricos, donde la matriz de coeficientes A tiene parte antisimétrica de rango bajo o puede aproximarse bien con una matriz antisimétrica de rango bajo. Sistemas como este surgen de la discretización de PDEs con ciertas condiciones de frontera de Neumann, la discretización de ecuaciones integrales y métodos de puntos interiores, por ejemplo, el problema de Bratu y la ecuación integral de Love. El segundo tipo de sistemas lineales considerados son problemas de mínimos cuadrados (LS) que se resuelven considerando la solución del sistema equivalente de ecuaciones normales. Concretamente, consideramos la solución de problemas LS modificados y de rango incompleto. Por problema LS modificado se entiende que el conjunto de ecuaciones lineales se actualiza con alguna información nueva, se agrega una nueva variable o, por el contrario, se elimina alguna información o variable del conjunto. En los problemas LS de rango deficiente, la matriz de coeficientes no tiene rango completo, lo que dificulta el cálculo de una factorización incompleta de las ecuaciones normales. Los problemas LS surgen en muchas aplicaciones a gran escala de la ciencia y la ingeniería como, por ejemplo, redes neuronales, programación lineal, sismología de exploración o procesamiento de imágenes.

Los precondicionadores directos para métodos iterativos usados habitualmente son las factorizaciones incompletas LU, o de Cholesky cuando la matriz es simétrica definida positiva. La principal contribución de esta tesis es el desarrollo de técnicas de actualización de precondicionadores. Básicamente, el método consiste en el cálculo de una descomposición incompleta para un sistema lineal aumentado equivalente, que se utiliza como precondicionador para el problema original.

El estudio teórico y los resultados numéricos presentados en esta tesis muestran el rendimiento de la técnica de precondicionamiento propuesta y su competitividad en comparación con otros métodos disponibles en la literatura para calcular precondicionadores para los problemas estudiados.

Resum

El tema principal d'esta tesi és actualitzar precondicionadors per a resoldre sistemes lineals grans i buits $Ax = b$ per mitjà de l'ús de mètodes iteratius de Krylov. Es consideren dos tipus interessants de problemes. En el primer s'estudia la solució iterativa de sistemes lineals no singulars i antisimètrics, on la matriu de coeficients A té una part antisimètrica de baix rang, o bé pot aproximar-se amb una matriu antisimètrica de baix rang. Sistemes com este sorgixen de la discretització de PDEs amb certes condicions de frontera de Neumann, la discretització d'equacions integrals i mètodes de punts interiors, per exemple, el problema de Bratu i l'equació integral de Love. El segon tipus de sistemes lineals considerats, són problemes de mínims quadrats (LS) que es resolen considerant la solució del sistema equivalent d'equacions normals. Concretament, considerem la solució de problemes de LS modificats i de rang incomplet. Per problema LS modificat, s'entén que el conjunt d'equacions lineals s'actualitza amb alguna informació nova, s'agrega una nova variable o, al contrari, s'elimina alguna informació o variable del conjunt. En els problemes LS de rang deficient, la matriu de coeficients no té rang complet, la qual cosa dificultata el calcul d'una factorització incompleta de les equacions normals. Els problemes LS sorgixen en moltes aplicacions a gran escala de la ciència i l'enginyeria com, per exemple, xarxes neuronals, programació lineal, sismologia d'exploració o processament d'imatges.

Els precondicionadors directes per a mètodes iteratius utilitzats més a sovint són les factoritzacions incompletes tipus ILU, o la factorització incompleta de Cholesky quan la matriu és simètrica definida positiva. La principal contribució d'esta tesi és el desenvolupament de tècniques d'actualització de precondicionadors. Bàsicament, el mètode consistix en el càlcul d'una descomposició incompleta per a un sistema lineal augmentat equivalent, que s'utilitza com a precondicionador pel problema original.

L'estudi teòric i els resultats numèrics presentats en esta tesi mostren el rendiment de la tècnica de precondicionament proposta i la seua competitivitat en comparació amb altres mètodes disponibles en la literatura per a calcular precondicionadors per als problemes considerats.

Acknowledgements

To my family. A special feeling of gratitude to my wife Elisa Savoia and my daughter Lucía Guerrero. I also dedicate this thesis to my friends who have supported me throughout the process. To each professor I had during my education, in particular, my project coordinators José Marín, José Mas and Juana Cerdán who have been more than generous with their expertise and precious time spent with me for preparing this thesis. To the Universidad Politécnica de Valencia, Universidad Pedagógica Nacional Francisco Morazán and Fundación Carolina.

Contents

List of Figures

List of Tables

To all my beloved ones…
specially Elisa and Lucía

Chapter 1

Introduction

The aim of this thesis is to study techniques for updating preconditioners for iterative methods based on Krylov subspaces, to solve systems of linear equations arising in many different areas of science and engineering. The problems of interest have large coefficient matrices, generally sparse, ill-conditioned and even rank deficient, that suppose a challenge and an opportunity for the study and proposal of new techniques. The efficient solution of linear systems using iterative methods requires the use of preconditioning techniques. The purpose of the preconditioning is to improve the spectral properties of the system matrix so that the solution is obtained with a lower computational cost. Preconditioners can be roughly classified into direct preconditioners, that calculate an approximation of the coefficient matrix, and approximate inverse preconditioners that explicitly compute an approximation of its inverse. Among the first ones, are incomplete LU and Cholesky factorizations (ILU, IC). In this line, we will introduce a technique for updating a previously computed ILU-type preconditioner using a bordering method. Furthermore, we will show that for the problems considered in this work our proposal is competitive compared with other preconditioning methods that appear in the literature.

The contents of this thesis are structured as follows. In this chapter we present the general and specific objectives of this work. Then, we review some basic concepts and the notation used throughout the text. We also present a basic introduction to the theory of Krylov subspace methods and preconditioning. We describe the problems considered as well as an overview of existing techniques to solve them. The main objectives of our work are also stated. In Chapter 2, the basic framework for updating a preconditioner is described. The technique will be developed for each problem in subsequent chapters. The main problems studied are:

- Almost symmetric problems: in Chapter 3 we study the iterative solution of nonsingular, nonsymmetric linear systems where the coefficient matrix has skew-symmetric part that is low-rank, or can be well approximated by a skew-symmetric low-rank matrix. In general, any problem whose skew-symmetric part has a small number of dominant singular values can be described in this way.
- Least squares (LS) problems: the resolution of LS problems can be carried out by preconditioned iterative methods applied to the normal equations. In Chapter 4, we focus on least squares problems where the set of linear relations is updated with some new information, a new variable is added or, contrarily, some information or variable is removed from the original set. We call this kind of problems modified LS problems. In Chapter 5, we study the resolution of least squares problems when the coefficient matrix is rank deficient.

1.1 Objectives

This section includes descriptions of the general and specific objectives of this thesis. There are four primary objectives of this study:

1. Develop preconditioners for iteratively solving linear systems.
2. Update a preconditioner previously computed in order to obtain a new preconditioner to accelerate the convergence of a Krylov iterative method to solve linear system, giving details of its computation and application.
3. Apply the update preconditioning technique for solving almost symmetric linear systems $Ax = b$, where the coefficient matrix A nonsymmetric and has a skew-symmetric part that can be well approximated by a skew-symmetric low-rank matrix and for solving LS problems.
4. Compare the performance of the preconditioner proposed with others already available in the repositories, when solving almost symmetric linear systems and LS problems.

To accomplished these main objectives some specific ones are considered:

1. Study the preconditioners already available for solving almost symmetric linear systems and LS problems.

2. State and prove some results concerning to the approximation properties of the preconditioner proposed and the spectral properties of the preconditioning technique.

3. Verify the spectral properties of the proposed preconditioner by analysing applied problems.

4. Solve almost symmetric linear systems arising from the discretization of PDEs with certain Neumann boundary conditions, the discretization of integral equations as well as path following methods, as for instance, the Bratu problem, Love's integral equations and problems from the University of Florida sparse matrix collection [35].

5. Compare the performance of the preconditioner proposed when solving almost symmetric linear systems with another strategies that has been proposed to solve this kind of problems, mainly with the Schur complement method (SCM) developed in [37].

6. Solve LS problems when the set of linear relations is updated with some new information, new variables are added or, contrarily, some information or variables are removed from the set.

7. Study the numerical performance of the preconditioner proposed when solving modified LS problems arising in different areas of scientific computing. The performance of the technique is compared with other preconditioning strategies, as reusing a previously preconditioner computed for the normal equations of the unmodified matrix, computing a new almost Cholesky preconditioner for the modified matrix from scratch and without preconditioner.

8. Solve LS problems when the coefficient matrix is rank deficient arising in many large-scale applications of the science and engineering.

9. Compare the preconditioning technique proposed with the non updated one, by analysing the convergence criteria for the iterative method used, the choice of a needed parameter of regularization and the performance of the preconditioners, when solving rank deficient LS problems.

Summarizing, the main objective of this thesis is to develop a technique for updating preconditioners for solving large-scale and sparse linear systems of the form $Ax = b$ by using iterative Krylov methods. Furthermore, we are interested in the numerical results that demonstrate the performance of the proposed technique and its competitiveness in comparison with other methods available in the literature, to solve a wide collection of problems, which are frequently presented in different fields of science and engineering.

1.2 Preliminaries and Basic Concepts

We are interested in solving linear systems

$$Ax = b \tag{1.1}$$

where the $m \times n$ matrix A is large, sparse and real, and x, b are m-dimensional vectors.

Linear systems are among the most important and common problems encountered in scientific computing and engineering. The structure and properties of the coefficient matrix determine the method of choice to solve the linear system. Considering a square matrix $A = (a_{ij})$ we recall the following definitions:

- Symmetric matrix: $A^T = A$, where the super index T represents the transpose.
- Skew-symmetric matrix: $A^T = -A$.
- Normal matrix: $A^T A = AA^T$.
- Orthogonal matrix: $A^T A = I$.
- Diagonal matrix: $a_{ij} = 0$ if $i \neq j$. We denote a diagonal matrix as $A = \mathrm{diag}(a_{11}, \ldots, a_{nn})$.
- Tridiagonal matrix: $a_{ij} = 0$ for any pair of i, j with $|j - i| > 1$. We denote it by $A = \mathrm{tridiag}(a_{i,i-1}, a_{ii}, a_{i,i+1})$.
- Upper/Lower triangular matrix: $a_{ij} = 0$ for $i > j / a_{ij} = 0$ for $i < j$.
- Permutation matrix: a square matrix obtained from the same size identity matrix by a permutation of rows.
- Upper Hessenberg matrix: $a_{ij} = 0$ for i, j with $i > j + 1$. Lower Hessenberg matrix can be defined similarly.
- Nonnegative matrix: $a_{ij} \geq 0, \forall i, j$.

Some of these concepts can be extended to non-square matrices easily as is the case of Hessenberg and nonnegative matrices.

An inner product on a complex vector space $\mathbb{X}$ is an assignment that for any two vectors $x, y \in \mathbb{X}$ there is a real number (x, y) satisfying the following properties:

1. $(ax + by, z) = a(x, z) + b(y, z)$, $\forall x, y \in \mathbb{X}$ and $\forall a, b \in \mathbb{C}$.
2. $(x, x) \geq 0$, $\forall x \in \mathbb{X}$, and $(x, x) = 0$ if, and only if $x = 0$.
3. $(x, y) = \overline{(y, x)}$, $\forall x, y \in \mathbb{X}$.

In the case of $\mathbb{X} = \mathbb{C}^n$ it is defined the Euclidean inner product as

$$(x, y) = \sum_{i=1}^{n} x_i \bar{y}_i = y^H x \ .$$

The concept of vector and matrix norms are a powerful tool in numerical analsysis. A vector norm on a vector space $\mathbb{X}$ is a real-valued function $|| \cdot ||$ which satisfies the following three conditions:

1. $||x|| \geq 0$, $\forall x \in \mathbb{X}$, and $||x|| = 0$ if and only if $x = 0$.
2. $||\alpha x|| = |\alpha| ||x||$, $\forall x \in \mathbb{X}$, $\forall \alpha \in \mathbb{C}$.
3. $||x + y|| \leq ||x|| + ||y||$, $\forall x, y \in \mathbb{X}$.

When $\mathbb{X} = \mathbb{C}^n$ the *Euclidean norm* of a vector is defined by

$$||x||_2 = (x, x)^{1/2} = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2} . \tag{1.2}$$

As a consequence, an orthogonal matrix preserves the Euclidean norm metric, i.e., $||Qx||_2 = ||x||_2, \forall x$. Other important vector norms are particular cases of the Holder norms defined by

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} .$$

In particular, the values $p = 1$, $p = 2$ and $p = \infty$ define three important norms in numerical linear algebra. Note that by taking limits one has

$$||x||_\infty = \max_{i=1,\ldots,n} |x_i|.$$

The concept of norm can be extended to matrices. Given a matrix $A \in \mathbb{R}^{m \times n}$, it is defined the set of norms

$$||A||_{p,q} = \max_{x \in \mathbb{R}^n, x \neq 0} \frac{||Ax||_p}{||x||_q} \tag{1.3}$$

induced by the two vector norms $||\cdot||_p$ and $||\cdot||_q$. Note that the $||\cdot||_{p,q}$ satisfies the usual three properties of matrix norms:

1. $||A|| \geq 0, \forall A \in \mathbb{R}^{m\times n}$, and $||A|| = 0$ if and only if $A = O$.
2. $||\alpha A|| = |\alpha| ||A||, \forall A \in \mathbb{R}^{m\times n}, \forall \alpha \in \mathbb{R}$.
3. $||A + B|| \leq ||A|| + ||B||, \forall A, B \in \mathbb{R}^{m\times n}$.

The case $q = p$ is of particular interest and the associated norm is denoted by $||\cdot||_p$ and called a *p-norm*. The most important cases correspond with $p = 1, 2, \infty$. A very important property of a p-norm is that

$$||AB||_p \leq ||A||_p ||B||_p.$$

Another very common matrix norm used in the literature is the Frobenious norm, which is not an induced norm. It is a generalization of the euclidean norm for matrices, defined as

$$||A||_F = \left(\sum_{j=1}^{n} \sum_{i=1}^{m} |a_{ij}|^2 \right)^{1/2}. \tag{1.4}$$

One can establisth the following relations between matrix norms that imply that all the norms are equivalent,

$$\begin{aligned} ||A||_2 \leq& ||A||_F \leq \sqrt{\min\{m, n\}} ||A||_2 \\ \frac{1}{\sqrt{n}} ||A||_\infty \leq& ||A||_2 \leq \sqrt{m} ||A||_\infty \\ \frac{1}{\sqrt{m}} ||A||_1 \leq& ||A||_2 \leq \sqrt{n} ||A||_1. \end{aligned} \tag{1.5}$$

There are also some interesting relations between the norms defined above and some other concepts related to a matrix A, as for instance the eigenvalues, singular values (square of the eigenvalues of $A^T A$) and trace of the matrix $A^T A$. Some of these are presented in Equation (1.6)

$$\begin{aligned} ||A||_1 &= \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}| \\ ||A||_\infty &= \max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}| \\ ||A||_2 &= \left[\rho(A^T A)\right]^{1/2} = \sigma_{max}(A) \\ ||A||_F &= \left[\mathrm{tr}(A^T A)\right]^{1/2}, \end{aligned} \tag{1.6}$$

where tr denotes the trace, ρ indicates the *spectral radius*, that is, the maximum of the absolute eigenvalues, and σ_{max} is the largest singular value of a matrix, respectively.

Finally, a matrix A is *positive definite* if $(Ax, x) > 0, \forall x \in \mathbb{R}^n$ and $x \neq 0$. In addition, if A is symmetric, then it is said that A is *symmetric positive definite* (SPD). The condition number of a nonsingular matrix is defined as $\mathrm{cond}(A) = ||A^{-1}||_2 \cdot ||A||_2$, or equivalently, $\mathrm{cond}(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$, where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ are the maximum and minimum singular values of A, respectively. For moderate values of the condition number a matrix is said to be well-conditioned. If the condition number is very large, then the matrix is ill-conditioned. In practice, computations with such a matrix, as the computation of its inverse or the solution of a linear system of equations, are prone to large numerical errors.

1.3 Overview on Iterative Methods

In this section, we present a basic introduction into the theory of Krylov subspace methods. It is common that problems arising from the discretization of partial differential equations lead to large sparse systems of linear equations. For large-scale problems obtaining a solution with direct methods may be difficult due to well known reasons, as for instance, memory constraints. In theses cases an alternative is the use of iterative methods, that generate a sequence of approximate solutions. The main idea behind iterative methods for the solution of a linear system $Ax = b$ is, from a given vector x_k, derive a better approximation x_{k+1} of the solution at a low computational cost. In all this section, the main references considered are [10, 20, 43, 73]. Also [5, 8, 45, 74, 87, 92, 93].

1.3.1 Basic Iterative Methods

Given a linear system $Ax = b$, where A is a nonsingular matrix, the expression $A = M - N$ with M invertible, defines an splitting of A. Therefore, the linear system is equivalent to $Mx = Nx + b$, i.e.,

$$x = M^{-1}Nx + M^{-1}b.$$

This equation allows for the definition of the fixed point iteration

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b, \quad k \geq 0, \tag{1.7}$$

with x_0 any arbitrary initial vector. The convergence of basic iterative methods has been well studied by many authors. Basically, the convergence to the solution $x^* = A^{-1}b$ is guaranteed if the spectral radius of the iteration matrix is less than one, i.e., $\rho(M^{-1}N) < 1$ [43, Th 10.1.1].

Different methods correspond to different choices of the splitting of the matrix A. Writting the matrix as $A = D - E - F$, where D, $-E$ and $-F$ are the diagonal, strict lower and upper parts of A, respectively, the more popular basic iterative methods are defined as follows:

- ***Jacobi's method:*** $M = D$ and $N = E+F$. Here $M^{-1}N = D^{-1}(E+F)$ is known as Jacobi's matrix.
- ***Gauss-Seidel's method:*** $M = D - E$ and $N = F$ (backward Gauss-Seidel's with $M = D - F$ and $N = E$).
- ***The relaxation method:***

  $$M = \frac{D}{\omega} - E, \quad N = \frac{1-\omega}{\omega}D + F,$$

  where $\omega \neq 0$ is known as the relaxation parameter.

  The use of these methods have been superseded by Krylov iterative methods. In any case, they still have a rol in many scientific computing areas as preconditioners, mainly due to its simplicity.

1.3.2 Krylov Subspace Methods

A Krylov method is an special case of projection methods. The main goal of projection methods is to extract an approximate solution of $Ax = b$ from a m-dimensional subspace $\mathcal{K}_m$ of $\mathbb{R}^n$, by imposing m constraints. Specifically, the residual vector $b - Ax$ is constrained to be orthogonal to m linearly independent vectors. This defines another m-dimensional subspace $\mathcal{L}_m$ called the subspace of constraints. This scheme is common to many different mathematical methods and it is known as the Petrov-Galerkin conditions.

There are two classes of projection techniques:

- Orthogonal: $\mathcal{L}_m = \mathcal{K}_m$. In this case the Petrov-Galerkin conditions are often called the Galerkin conditions.
- Oblique: $\mathcal{L}_m \neq \mathcal{K}_m$. A typical choice is $\mathcal{L}_m = A\mathcal{K}_m$.

In general, a projection method onto the subspace $\mathcal{K}_m$ and orthogonal to $\mathcal{L}_m$ is a process summarized as follows:

$$\text{Find } \tilde{x} \in \mathcal{K}_m, \text{ such that } b - A\tilde{x} \perp \mathcal{L}_m. \tag{1.8}$$

If an initial guess x_0 is known, then the approximate problem should be redefined as

$$\text{Find } \tilde{x} \in x_0 + \mathcal{K}_m, \text{ such that } b - A\tilde{x} \perp \mathcal{L}_m. \tag{1.9}$$

Note that if $\tilde{x}$ is written in the form $\tilde{x} = x_0 + \delta$, with $\delta \in \mathcal{K}_m$, then the condition in Equation (1.9) is equivalent to $r_0 - A\delta \perp \mathcal{L}_m$, where $r_0 = b - Ax_0$ is the initial residual. Therefore, the projection method is defined as

$$\text{Find } \tilde{x} = x_0 + \delta,\ \delta \in \mathcal{K}_m, \text{ such that } (r_0 - A\delta, w) = 0,\ \forall w \in \mathcal{L}_m. \tag{1.10}$$

The orthogonal condition is graphically presented in Figure 1.1. This is a basic projection step. Most standard techniques use a succession of such projections.

A **Krylov subspace method** is a projection method for which the subspace $\mathcal{K}_m$ is the Krylov subspace

$$\mathcal{K}_m(A, r_0) = \mathrm{Span}\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\}, \tag{1.11}$$

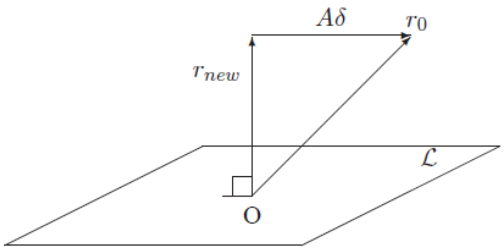


FIGURE 1.1: Representation of the orthogonal condition.

where r_0 is the initial residual and m is the dimension. For an initial solution x_0, approximations to the solution x are computed in every step by iterates x_m of the form:

$$x_m \in x_0 + \mathcal{K}_m(A, r_0)\ \text{, with}\ \ m > 1.$$

Let the vectors $v_1, v_2, \ldots, v_m$ be a basis of $\mathcal{K}_m$, and $V_m = [v_1, v_2, \ldots, v_m]$. Then, an expression for the m-iterate is given by

$$r_m = r_0 - AV_m y_m, \text{ with } y_m \in \mathbb{R}^m \ \text{ and } \ x_m = x_0 + V_m y_m. \tag{1.12}$$

To construct the matrix V_m two methods are proposed in the literature: Arnoldi's method and Lanczos's method.

Arnoldi's method [3] is an orthogonal projection method for general non-symmetric matrices. The method computes an orthogonal basis of the Krylov subspace $\mathcal{K}_m$. Assuming that $v_1 = r_0/\beta$ with $\beta = ||r_0||_2$, then one has that $V_m^T A V_m = H_m$ where H_m is an upper Hessenberg matrix, and $V_m^T r_0 = \beta e_1$. Therefore, the approximate solution is given by

$$x_m = x_0 + V_m H_m^{-1}(\beta e_1).$$

A variation of this approach, that uses the modified Gram-Schmidt method in the Arnoldi procedure, is the Full Orthogonalization Method (FOM). Moreover, for large problems there exists restarted and truncated variants of the FOM method.

The second method to find the basis of the Krylov subspace is the Lanczos's method [53]. It is an special case of Arnoldi's method for symmetric matrices in which the Hessenberg matrix H_m becomes symmetric and tridiagonal. Thus, the computation of the Arnoldi vectors can be done with a three-term recurrence formula.

Another class of Krylov subspace methods are based on a bi-orthogonalization algorithm [54, 75]. The Lanczos biorthogonalization algorithm is an extension to nonsymmetric matrices of the symmetric Lanczos algorithm. The algorithm builds a pair of biorthogonal bases.

In the rest of this subsection we present the most important Krylov subspace algorithms, some of them used in this thesis.

1.3.3 Iterative Methods for General Matrices

Conjugate Gradient

The Conjugate Gradient method (CG) was originally proposed by Hestenes and Stiefel in [50]. It is one of the most important iterative techniques for solving sparse linear systems when the matrix A is SPD. It is the result of an orthogonal projection technique onto the Krylov subspace $\mathcal{K}_m(r_0, A)$. Algorithm 1 shows the method.

Algorithm 1 Conjugate Gradient

```
1. Compute $r_0 = b - Ax_0,\ \ p_0 := r_0$.
2. For $j = 0, 1, 2, \ldots,$ until convergence Do:
3.      $\alpha_j := (r_j, r_j)/(Ap_j, p_j)$
4.      $x_{j+1} := x_j + \alpha_j p_j$
5.      $r_{j+1} := r_j - \alpha_j A p_j$
6.      $\beta_j := (r_{j+1}, r_{j+1})/(r_j, r_j)$
7.      $p_{j+1} := r_{j+1} + \beta_j p_j$
8. EndDo
```

The CG method minimizes $||x - x_k||_A$, where $||x||_A = (Ax, x)^{1/2}$, over $K_m(A, r_0)$. It requires only short-term recurrences, one matrix-vector product and a few vector updates. Anyway, for general matrices, as for instance, rectangular, nonsymmetric or indefinite matrices, the algorithm may not converge because the orthogonality condition can not be accomplished. A simple solution is to apply the CG algorithm to the normal equations A^TA. In this case, to solve the system $Ax = b$, one considers the equivalent system:

$$A^TAx = A^Tb. \tag{1.13}$$

This procedure is called Conjugate Gradient Normal Residual (CGNR) [20, 41]. In the context of LS problems it is also known as Conjugate Gradient for Least Squares (CGLS). It applies the CG to the normal equations without explicitly forming them.

Another alternative to overcome the indefiniteness is to set $x = A^T u$ and solve the equation for u, $AA^T u = b$, by applying the CG. This technique yields to the Conjugate Gradient Normal Equation method (CGNE) or CRAIG's method [20]. Once the solution u is computed, x could be obtained by multiplying u by A^T.

When the matrix is symmetric but indefinite, an option to the CG method is the Minimal Residual method (MINRES) [29, 66, 92].

Biconjugate Gradient Methods

These methods are based on Lanczos biorthogonalization and are good options when solving nonsymmetric linear systems. The Biconjugate Gradient method (BCG) [39] computes two mutually orthogonal sequences at the price of no longer satisfying an optimality condition. Unlike the CG method applied to the normal equations, BCG does not require the explicit computation of the matrix $A^T A$, but only matrix-vector multiplications with A^T. Its main drawback is that the convergence behaviour may be quite irregular in practice, and may even breakdown. The Conjugate Gradient Squared (CGS) [81] algorithm avoids using the transpose of A in the BCG, so there are not matrix-vector products with the transpose of A. Both methods tipically show very large variations of the residual vectors and, as result, the residual norms computed are inaccurate. The BICGSTAB method [91] was developed to remedy these troubles. In practice shows a more regular convergence pattern than the CGS method with lower overall computational cost.

Generalized Minimum Residual Method

One of the most popular methods for nonsymmetric linear systems is the Generalized Minimum Residual method (GMRES) [76], wich is an extension of the MINRES. It is an oblique projection method with $\mathcal{L} = A\mathcal{K}_m$, where $\mathcal{K}_m$ is the m-th Krylov subspace with $v_1 = r_0/||r_0||^2$. It minimizes the residual norm $||b - Ax_k||_2$ over all vectors in $x_0 + \mathcal{K}_m$. The basic GMRES is presented in Algorithm 2.

The full GMRES algorithm is guaranteed to converge in at most n steps, but in practice this could lead to high computational and memory costs to build the Krylov subspace basis vectors. To rectify this inconvenience, there exist restarted and truncated variants of GMRES. The restarted GMRES, GMRES(r), restarts the algorithm after r steps. A disadvantage of GMRES(r) is that it can stagnate when the matrix is

Algorithm 2 Generalized Minimum Residual, GMRES

```
1. Compute $r_0 = b - Ax_0$,  $\beta := ||r_0||_2$,  $v_1 := r_0/\beta$.
2. For $j = 0, 1, 2, \ldots, m$ Do:
3.         Compute $w_j = Av_j$
4.         For $i = 0, 1, 2, \ldots, j$ Do:
5.                 $h_{ij} := (w_j, v_i)$
6.                 $w_j := w_j - h_{ij}v_i$
7.         EndDo
8.         $h_{j+1,j} = ||w_j||$. If $h_{j+1,j} = 0$ set $m := j$ and go to 11
9.         $v_{j+1} = w_j/h_{j+1,j}$
10. EndDo
11. Define the $(m+1) \times m$ Hessemberg matrix $\tilde{H}_m = \{h_{ij}\}_{1 \leq i \leq m+1, 1 \leq j \leq m}$.
12. Compute $y_m$ the minimizer of $||\beta e_1 - \tilde{H}_m y||_2$ and $x_m = x_0 + V_m y_m$.
```

not positive definite. The use of preconditioners can be useful to increase the convergence rate and avoid the stagnation.

The literature is rich and there are many other variants of the methods described above. Which method to use for a particular problem will depend on the properties and structure of the coefficient matrix [10].

1.3.4 Iterative Methods for Least Squares Problems

The Least Squares (LS) problem is formulated as

$$\min_x ||b - Ax||_2, \tag{1.14}$$

where $A \in \mathbb{R}^{m \times n}$ $(m \geq n)$ and $b \in \mathbb{R}^m$. When the matrix A is large and sparse iterative methods may be an interesting alternative to direct methods. The iterative solution of LS problems can be done with the CGLS method presented above. Alternatively, one can use the Least Squares QR method (LSQR) presented in [67, 68], wich is based on the Golub–Kahan bidiagonalization [42]. LSQR has the property of reducing the norm of the residual $||r_k||_2$ monotonically. Mathematically, the CGLS and LSQR methods generate the same sequence of approximations since they apply implicitly the CG to the normal equations (1.15).

$$A^T Ax = A^T b. \tag{1.15}$$

Algorithm 3 Least Square Minimal Residual algorithm, LSMR

```
1. Initialize
   $\beta_1 u_1 = b$    $\alpha_1 v_1 = A^T u_1$    $\bar{\alpha}_1 = \alpha_1$    $\bar{\zeta}_1 = \alpha_1 \beta_1$
   $\rho_0 = \bar{\rho}_0 = \bar{c}_0 = 1$    $\bar{s}_0 = 0$    $h_1 = v_1$    $\bar{h}_0 = x_0 = \overrightarrow{0}$
2. For $k = 1, 2, 3, \ldots,$ until convergence repeat steps 3-6:
3. Continue the bidiagonalization
   $\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$
   $\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$
4. Construct and apply rotation $Q_{k,k+1}$
   $\rho_k = (\bar{\alpha}_k^2 + \beta_{k+1}^2)^{1/2}$
   $c_k = \bar{\alpha}_k / \rho_k$    $s_k = \beta_{k+1} / \rho_k$
   $\theta_{k+1} = s_k \alpha_{k+1}$    $\bar{\alpha}_{k+1} = c_k \alpha_{k+1}$
5. Construct and apply rotation $\bar{Q}_{k,k+1}$
   $\bar{\theta}_k = \bar{s}_{k-1} \rho_k$    $\bar{\rho}_k = ((\bar{c}_{k-1} \rho_k)^2 + \theta_{k+1}^2)^{1/2}$
   $\bar{c}_k = \bar{c}_{k-1} \rho_k / \bar{\rho}_k$    $\bar{s}_k = \theta_{k+1} / \bar{\rho}_k$
   $\zeta_k = \bar{c}_k \bar{\zeta}_k$    $\bar{\zeta}_{k+1} = -\bar{s}_k \bar{\zeta}_k$
6. Update $h$, $\bar{h}$ and $x$
   $\bar{h}_k = h_k - (\bar{\theta}_k \rho_k / (\rho_{k-1} \bar{\rho}_{k-1})) \bar{h}_{k-1}$
   $x_k = x_{k-1} + (\zeta / (\rho_k \bar{\rho}_k)) \bar{h}_k$
   $h_{k+1} = v_{k+1} - (\theta_{k+1} / \rho_k) h_k$
```

Another method is the Least Squares Minimal Residual (LSMR) presented in [38]. LSMR is also based on the Golub-Kahan bidiagonalization of A. But in contrast to the LSQR, LSMR is equivalent to MINRES applied to the normal equations and, in this case, the quantities $||A^T r_k||_2$ are monotonically decreasing. In practice, the norm of the residual also decreases monotonically, and it is never very far behind the corresponding value for LSQR. Hence, it is safer to use LSMR in situations where the solver must be terminated early. The Algorithm 3 summarizes the main steps of the LSMR method. For more details of the method see [38].

In this thesis, the methods GMRES, BICGSTAB, CG, CGNR and LSMR are mainly used. To better understand some modifications introduced later on regarding the application of the preconditioner for the LSMR method, the pseudo-codes of the CG, GMRES and LSMR methods are provided.

1.4 Preconditioning

In this subsection we describe the most important general purpose preconditioners, particularly those based on incomplete LU (ILU) and incomplete Cholesky (IC) type factorizations. See [8, 13, 20, 73, 77, 92] for more details.

1.4.1 Generalities

It is well known that for most problems the convergence of an iterative method can be slow, or even can fail to converge. In this situation, the application of a preconditioner is mandatory. Preconditioning is any technique that modifies the original linear system $Ax = b$ in such a way that an approximate solution can be obtained at a lower computational cost. Commonly, the preconditioned linear system is written as

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

where M is called the preconditioner matrix. Normally, the matrix M or its inverse is stored explicitly, but there are also matrix-free methods. In the context of Krylov subspace methods, the goal of precondiditioning is to find a matrix M such that $M^{-1}A$ has better spectral properties than A, i.e., an smaller condition number and with eigenvalues clustered away from zero. Note that $M = A$ would be the ideal choice but it is not practical for the obvious reasons. But the idea behind is that we should try to stay as close to A as possible. Other two important requirements for a preconditioner are that it should be cheap to compute and have limited memory requirements.

The preconditioner can be applied in three different ways:

- **Left preconditioning.** The iterative method is applied to $M^{-1}Ax = M^{-1}b$.
- **Right preconditioning.** It applies the iterative method to $AM^{-1}y = b$, with $x = M^{-1}y$. An advantage for the right preconditioning approach is that in exact arithmetic, the residuals for right preconditioned system are identical to

the true residuals (the right-hand side is not affect), and thus, the convergence behaviour can be monitored accurately.

- **Two-side preconditioning.** For a preconditioner M with $M = M_1M_2$, the iterative method is applied to $M_1^{-1}AM_2^{-1}z = M_1^{-1}b$, with $x = M_2^{-1}z$. This form of preconditioning is intended for factorized preconditioners.

Algorithms 4 to 7 show the preconditioned versions of the Krylov subspace methods that will be used later.

Right preconditioned GMRES. See Algorithm 4, essentially only the steps 3 and 12 change with respect to the non-preconditioned GMRES Algorithm 2.

Algorithm 4 Right preconditioned GMRES

```
1. Compute $r_0 = b - Ax_0,\ \ \beta := ||r_0||_2,\ \ v_1 := r_0/\beta$.
2. For $j = 0, 1, 2, \ldots, m$ Do:
3.     Compute $w_j = AM^{-1}v_j$
4.     For $i = 0, 1, 2, \ldots, j$ Do:
5.         $h_{ij} := (w_j, v_i)$
6.         $w_j := w_j - h_{ij}v_i$
7.     EndDo
8.     $h_{j+1,j} = ||w_j||$. If $h_{j+1,j} = 0$ set $m := j$ and go to 11
9.     $v_{j+1} = w_j/h_{j+1,j}$
10. EndDo
11. Define the $(m+1) \times m$ Hessemberg matrix $\tilde{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$.
12. Compute $y_m$ the minimizer of $||\beta e_1 - \tilde{H}_m y||_2$ and $x_m = x_0 + M^{-1}V_m y_m$.
```

Left and two-side preconditioned CG. Algorithms 5 and 6 show the left and two-side preconditioned conjugate gradient, respectively. It is assumed that M is SPD.

Algorithm 5 Left preconditioned CG

```
1. Compute $r_0 = b - Ax_0,\ z_0 = M^{-1}r_0,\ p_0 := z_0$.
2. For $j = 0, 1, 2, \ldots,$ until convergence Do:
3.     $\alpha_j := (r_j, z_j)/(Ap_j, p_j)$
4.     $x_{j+1} := x_j + \alpha_j p_j$
5.     $r_{j+1} := r_j - \alpha_j Ap_j$
6.     $z_{j+1} := M^{-1}r_{j+1}$
7.     $\beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j)$
8.     $p_{j+1} := z_{j+1} + \beta_j p_j$
9. EndDo
```

Algorithm 6 Two-side preconditioned CG

```
1. Compute $r_0 = b - Ax_0,\ \hat{r}_0 = M_1^{-1} r_0,\ p_0 := M_2^{-1}\hat{r}_0$.
2. For $j = 0, 1, 2, \dots,$ until convergence Do:
3.     $\alpha_j := (\hat{r}_j, \hat{r}_j)/(Ap_j, p_j)$
4.     $x_{j+1} := x_j + \alpha_j p_j$
5.     $\hat{r}_{j+1} := \hat{r}_j - \alpha_j M_1^{-1} Ap_j$
6.     $\beta_j := (\hat{r}_{j+1}, \hat{r}_{j+1})/(\hat{r}_j, \hat{r}_j)$
7.     $p_{j+1} := M_2^{-1}\hat{r}_{j+1} + \beta_j p_j$
8. EndDo
```

The sequence of approximate solutions obtained by the preconditioned CG algorithm are identical in both cases, so the choice of the preconditioning technique depends on the form of the preconditioner available. Right preconditioning can also be used. Note that the system solved with left preconditioning is no longer symmetric in general. Thus, to preserve symmetry the usual Euclidean inner product is replaced by the M-inner product, $(x, y)_M \equiv (Mx, y) = (x, My)$. This is mathematically equivalent to the right preconditioned CG algorithm with the M^{-1}-inner product.

Left preconditioned LSMR. The two-side preconditioned LSMR version is obtained from Algorithm 3 changing the matrix-vector products with A and A^T by $AM_1^{-1}v$ and $M_2^{-T}A^Tu$, respectively. But, we are specially interested on the left preconditioned version since, as we will show later, the kind of preconditioner updates developed can not be applied in factorized form. Following the idea in [4] where the authors derived a left preconditioned LSQR algorithm, we implemented the left preconditioned version of the LSMR presented in Algorithm 7. The changes introduced correspond to the initialization steps and the bidiagonalization stage.

Algorithm 7 Left preconditioned LSMR

```
1. Initialize
   $\beta_1 u_1 = b \quad \tilde{p} = A^T u_1 \quad v_1 = M^{-1}\tilde{p} \quad \alpha_1 = (v_1, \tilde{p})^{1/2}$
   $v_1 = v_1/\alpha_1 \quad \bar{\alpha}_1 = \alpha_1 \quad \bar{\zeta}_1 = \alpha_1\beta_1 \quad \rho_0 = \bar{\rho}_0 = \bar{c}_0 = 1$
   $\bar{s}_0 = 0 \quad h_1 = v_1 \quad \bar{h}_0 = x_0 = \overrightarrow{0}$
2. For $k = 1, 2, 3, \ldots,$ until convergence repeat steps 3-6:
3. Continue the bidiagonalization
   $\beta_{k+1}u_{k+1} = Av_k - \alpha_k u_k$
   $\tilde{p} = A^T u_{k+1} - \beta_{k+1}\tilde{p}, \quad v_{i+1} = M^{-1}\tilde{p}, \quad \alpha_{i+1} = (v_{i+1}, \tilde{p})^{1/2}, \quad \tilde{p} = \tilde{p}/\alpha_{i+1},$
   $v_{i+1} = v_{i+1}/\alpha_{i+1}$
4. Construct and apply rotation $Q_{k,k+1}$ (as Algorithm 3).
5. Construct and apply rotation $\bar{Q}_{k,k+1}$ (as Algorithm 3).
6. Update $h$, $\bar{h}$ and $x$ (as Algorithm 3).
```

1.4.2 Jacobi, GS and SOR preconditioners

In section (1.7) the basic iterative methods where reviewed. As mentioned, nowadays these methods are mainly used as preconditioners for multigrid and Krylov subspace iterative methods. Considering the splitting of A, $A = D - E - F$, the preconditioner matrix M for the Jacobi, Gauss-Seidel and SOR methods, respectively, is given by

$$M_{JA} = D, \quad M_{GS} = D - E, \quad M_{SOR} = \frac{D}{\omega} - E.$$

1.4.3 Incomplete LU and Cholesky preconditioners

Incomplete triangular factorizations are widely used as preconditioners. They can be obtained by approximately decomposing A into LU factors [43, 59, 73], where L and U are lower and upper triangular matrices, respectively. They are referred to as ILU-type preconditioners. In the case of symmetric positive definite systems, an incomplete Cholesky factorization LL^T (IC) is computed. There are many powerful variants available of the ILU factorization, such as ILUT [11, 72], ILUS [31], MILU[73] and Balanced Incomplete Factorization (BIF) [21, 22] among others. For SPD linear systems, there are efficient implementations as, for instance, the ICCG method proposed in [62, 63].

Zero fill-in implementations, as ILU(0) and IC(0), are of low computational cost and simple to implement. These algorithms take the nonzero pattern to be exactly the same as the one of the original matrix. Therefore, the preconditioner at worst, needs as much storage as the original matrix.

It is important to note that incomplete factorization methods may fail in attempting to solve strongly nonsymmetric or indefinite linear systems. Failure may occur due to the appearance of small or negative pivots. Besides breakdowns, small pivots may cause the triangular solves to be unstable, see [16, 30].

1.4.4 Convergence Criteria

Iterative methods must be stopped when certain convergence criteria is fullfilled. The choice of an adequate criteria depends of the method applied and the type of problem to be solved. As a standar, with initial guess set to zero, the iterative method is stopped whenever one or both of these conditions are met:

- A maximum number of iterations maxit have been performed.
- The relative residual is less than a given threshold, i.e., $\frac{||r_k||_2}{||r_0||_2} = \frac{||b-Ax_k||_2}{||b||_2} \leq \mathrm{tol}$.

Generally, in this thesis the values $\mathrm{maxit} = 2000$ and $\mathrm{tol} = 10^{-8}$ are considered for all the iterative methods tested.

1.5 Background and objectives of the thesis

In this subsection we present a description of the problems studied, almost symmetric linear systems and least squares problems, emphasizing their importance. State-of-the-art techniques used to solve them that can be found in the literature are also reported. In addition, the main objectives of this thesis are outlined.

1.5.1 Almost Symmetric Problems

The first kind of problems considered is the iterative solution of nonsingular, nonsymmetric linear systems where the coefficient matrix $A \in \mathbb{R}^{n\times n}$ is large, sparse and has a skew-symmetric part of low-rank or can be well approximated with a skew-symmetric low-rank matrix. Consider $A = H + K$, where:

- H is the symmetric part of A, i.e., $H = (A + A^T)/2$ and
- K is the skew-symmetric part of A, i.e., $K = (A - A^T)/2$.

We say that a matrix is almost symmetric when its skew-symmetric part can be written as $K = FCF^T + E$, where $F \in \mathbb{R}^{n\times s}$ is a full-rank rectangular matrix, $C \in \mathbb{R}^{s\times s}$ is a nonsingular skew-symmetric matrix with s even, $s \ll n$ and $E \in \mathbb{R}^{n\times n}$ with $\| E \| \ll 1$.

Systems like this arise from the discretization of PDEs with certain Neumann boundary conditions, the discretization of integral equations [37] as well as path following methods [12]. In general, any problem whose skew-symmetric part K has a small number of dominant singular values can be described in this way.

This kind of linear systems arise in many real applications as, for example, path following problems and the solution of integral equations.

Problem 1.1. ***The Bratu problem.*** *It consists of finding the solution $u(x, y)$ of the nonlinear boundary problem*

$$-\Delta u - \lambda \exp(u) = 0 \ \ in \ \ \Omega, \quad with \ \ u = 0 \ \ on \ \ \partial\Omega \tag{1.16}$$

as a function of the parameter λ, where Δ denotes the Laplacian, Ω the unit square and $\partial\Omega$ its boundary. By discretizing in a $t \times t$ grid and choosing the same finite difference discretization and parameters as in [12], it is obtained a matrix of order $n = t^2$ that is symmetric plus a rank-2 skew-symmetric modification, i.e., the skew-symmetric part K of the matrix obtained exactly has rank $s = 2$.

Problem 1.2. ***Love's integral equation.*** *It arises in electrostatics, see [6, 57], and it is given by*

$$f(y) + \frac{1}{\pi}\int_{-1}^{1} \frac{c}{(x-y)^2 + c^2} f(x)dx = g(y), \quad |y| \leq 1 \ \ 0 \leq c \in \mathbb{R}. \tag{1.17}$$

Love's equation is a particular case of Fredholm integral equations of the second kind [9, 70]

$$f(y) + \lambda \int_{-1}^{1} k(x-y) f(x) dx = g(y), \quad |y| \leq 1, \tag{1.18}$$

where $\lambda \in \mathbb{R}$, g and k are smooth functions, and f is the unknown. The kernels k depend on the difference $x - y$. To solve this particular type of integral equations different techniques are used. For example, direct methods for solving Toeplitz linear systems of equations [49, p. 60], or by applying the GMRES iterative method [88, p. 266].

Back to Love's integral equation, with $g(y) = \sqrt{1+y}$ and $c = 0.1$, the discretization by a Nyström method based on the composite trapezoidal rule with equidistant nodes $x_k = y_k = (k-1)/(n-1)$, $1 \leq k \leq n$ gives rise to a $n \times n$ linear system of equations whose coefficient matrix A has an skew-symmetric part with rank $s = 4$ exactly. A challenge that presents this problem is that A is completely dense, thus, excessively large values of n should be avoided.

Different strategies have been proposed to solve (1.1) when the skew-symmetric part K has exactly rank $s \ll n$, i.e., $E = 0$, as for example the problems described above. In [12] the authors present a progressive GMRES (PGMRES) method which shows that an orthogonal Krylov subspace basis can be generated with a short recurrence formula. This proposal incorporates two advantages over full GMRES: first, it only demands storage of three Arnoldi vectors, rather than the whole set; second, we can orthogonalize the new Arnoldi vector against the previous Krylov subspace. Unfortunately, as pointed out in [37], although the method is mathematically equivalent to full GMRES, in practice it may suffer from instabilities due to the loss of orthogonality of the generated Krylov subspace basis and the residual could stagnate long before convergence. In the same last paper, the authors propose a Schur complement method (SCM), that also permits the application of short-term recurrences formulas. The method obtains an approximate solution by applying the MINRES method $s+1$ times. The authors also suggest that it can be applied as a preconditioner for GMRES for the more general case when $E \neq 0$, problem that is considered in our work. Another method recently proposed is the Induced Dimension Reduction method, IDR(s), [80, 82]. The IDR(s) method is a new family of short-recurrence methods for large nonsymmetric systems of linear equations. In [82] the authors show that IDR(s) is competitive compared to most BICG based methods, and even outperforms BiCGSTAB when $s > 1$.

In this context, one of the objetives of the thesis is to develope an efficient preconditioning technique for preconditioning almost symmetric linear systems. The general framework of the technique is discused in Chapter 2. It is based on the work presented in [24]. Basically, it is a low-rank update of an incomplete LU factorization of the symmetric part of the system matrix H by a bordering method.

Finally, it is worth to note that to obtain low-rank approximations of the skew-symmetric part of a matrix, the Sparse Column Row approximation (SCR) algorithm will be used [19]. The SCR algorithm will be described and analyzed in Chapter 3. In the same chapter, the approximation and spectral properties of the preconditioner are studied. Moreover, the results of the numerical experiments for the problems described above will be presented.

1.5.2 Modified and Rank Deficient Least Squares Problems

As described in subsection 1.3.4, the LS problem is formulated as

$$\min_x \|b - Ax\|_2,$$

where $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ is large and sparse and $b \in \mathbb{R}^m$. Currently, the most used iterative method are the CGLS, LSQR and LSRM methods. As it is well known, to improve the convergence of any iterative method a preconditioner is normally needed. There are different preconditioners that can be used, like Incomplete QR [56]. But in this thesis we focus on Incomplete Cholesky (IC) preconditioners. These preconditioners have been successfully employed in different applications and allow for the computation of robust preconditioners for full-rank overdetermined least squares problems [17, 23].

Alternatively, the solution of the LS problem can be obtained from the equivalent $(m+n)\times(m+n)$ augmented linear system

$$\begin{pmatrix} I_m & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{1.19}$$

Note that since $r + Ax = b$ and $A^T r = 0$, one has $A^T Ax = A^T x$, see [20, 40, 44, 55, 92].

In this thesis, we concentrate on two particular types of LS problems, *modified* LS and *rank deficient* LS.

Modified Least Squares Problems

We consider the iterative solution of LS problems when the set of linear relations is updated with some new information, new variables are added or, contrarily, some information or variables are removed from the set.

For example, let us denote a set of k new relations as

$$Bx = c.$$

Then, the new $m + k$ system of linear equations is

$$\begin{bmatrix} A \\ B \end{bmatrix} x = \begin{bmatrix} b \\ c \end{bmatrix},$$

whose normal equations are

$$(A^T A + B^T B)x = A^T b + B^T c.$$

We propose a preconditioning method for the updated normal equations which computes a low-rank update of a previously computed preconditioner. As mentioned, the general framework of the method will be presented in the Chapter 2. Moreover, in Chapter 4, the method is analyzed in detail and numerical results for a wide variety of problems are presented.

The problem of updating a preconditioner arises in some applications from statistics and optimization, where it is necessary to solve a sequence of modified least squares problems. An example can be found in [27], where an efficient and stable method for adding and deleting equations to a regression model is required. In signal processing applications near real-time solutions are required. Thus, methods that allow to modify LS problems with few operations and little storage requirements are needed, see [1]. The same problem is present if some information is added to or deleted from the data set. On some occasions it may be convenient to add or to remove some variables. Such situations are usually referred to as updating or downdating least squares problems. This is an interested problem and there are many researchers that have discussed it, see [48, 65, 69]. Chapter 3 of the reference text [20] is devoted to analyzing how to deal with these modifications when the least squares problem is solved by a direct method, including full and rank revealing QR decomposition, Cholesky factorization and singular value decomposition. More recently other algorithms to update Cholesky factorizations have been proposed, see [32, 33, 34]. More efforts seem to be addressed to updating the QR factorization, see [2, 46, 64].

Recently in [44], the authors review the performance of a broad range of preconditioners, as for example, Jacobi preconditioning, IC factorizations and stationary inner iterations used with Krylov subspace methods. The iterative methods used are the LSQR and LSMR methods. In addition, iterative methods are compared with direct solvers applied to both, the normal equations and the augmented system (1.19).

Rank Deficient Least Squares Problems

Rank deficient least squares (RDLS) arise in many large-scale applications of the science and engineering as neural networks, linear programming, exploration seismology or image processing, to name a few.

If the matrix A is rank deficient then, the matrix C is a semidefinite positive matrix and the Cholesky factorization suffers breakdown because negative or zero pivots are encountered. Thus, rank deficient LS problems are in general much more harder to solve.

Basically, there are two types of approaches for solving this case. The first one consists of computing an incomplete factorization of a regularized matrix which can be used as a preconditioner for the original LS problem. The second type is solving a mathematically equivalent augmented linear system of order $m + n$, see [20, 44, 78], which is essentially a regularized version of Equation (1.19). The technique that we propose belongs to the first type. The details of the our proposal and the results are presented in Chapter 5.

The main idea is applying the general framework described in Chapter 2 to update an incomplete factorization computed for the regularized problem

$$\begin{bmatrix} A \\ \alpha^{1/2} I \end{bmatrix}.$$

The regularized normal equations are given by

$$C_\alpha = A^T A + \alpha I,$$

where α is known as Tikhonov regularization parameter. If α is choosen large enough the computation of an IC for the matrix C_α can be done easily. On the other hand, since the final purpose is to use this incomplete factorization as a preconditioner for the original (unregularized) linear system, the parameter α should be chosen as small as possible. Both requirements make difficult the choice of the appropriate α. In practice, the factorization is restarted more than once, increasing α on each restart until breakdown is avoided.

An extended review of a variety of preconditioners can be found in [78], for instance, diagonal preconditioning, limited memory incomplete Cholesky factorization developed for the HSL mathemathical software library [52], the Multilevel Incomplete QR (MIQR) factorization [56], the Robust Incomplete Factorization [17], the BA-GMRES for solving least squares problems [76] and incomplete Cholesky based on BIF preconditioner [23].

Some recent contributions to solve rank deficient least squares problems are presented in [78]. The authors used sparse direct solvers to compute the factors for the regularized normal equations and augmented linear system (1.19). The packages HSL_MA87 and HSL_MA97 from the HSL Mathematical Software Library [52] are used. Alternatively, incomplete factorizations are computed using the HSL_MI35 or HSL_MI28 packages. Both, direct solvers and incomplete factorizations are used as preconditioners for the LSMR method. The main conclusions of the paper are that direct solvers are more efficient as preconditioners for moderate size problems. But for large-scale problems, it may be impossible to compute and store in memory the factors. In this case, incomplete IC factorizations must be used since they provide robust and sparse preconditioners. Moreover, fine tuning of the Tikhonov regularization parameter is a paramount to obtain good preconditioners. One of the objectives of the method proposed in this thesis is to simplify the choice of this parameter by updating the incomplete factorization computed for the regularized LS problem.

Chapter 2

General Updated Preconditioner Method

In this chapter the general framework developed for updating preconditioners is presented. The kind of linear systems considered are those that can be written in the general form

$$(A_{11} - A_{12}A_{22}^{-1}A_{21})x = b, \tag{2.1}$$

where A_{11} is a $n \times n$ nonsingular matrix, and the order k of the matrix A_{22} is considerably smaller than n, i.e., $k \ll n$. The matrices A_{12} and A_{21} are the complementary rectangular matrices. The strategy is based on the computation of an approximate decomposition of the coefficient matrix of an equivalent augmented linear system. Some important results regarding the inverses of matrices, such as the Sherman–Morrison formula, and the concept of Schur complement are also reviewed.

2.1 Sherman-Morrison and Woodbury formulas

The Sherman-Morrison formula gives an expression for the inverse of a rank-1 modification of a nonsingular matrix.

Proposition 2.1. *Let A be a nonsingular $n \times n$ matrix, u and v be n-column vectors, then $A + uv^T$ is nonsingular if and only if $1 + v^T A^{-1}u \neq 0$. In that case, its inverse is given by*

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u}, \tag{2.2}$$

where uv^T is the outer product of u and v.

A proof of the previous statement can be found, for instance, in [79]. The Sherman-Morrison-Woodbury formula, or matrix inversion formula, is a generalization of the previous one to a rank-k update of the original matrix.

Proposition 2.2. *Let A be a nonsingular $n \times n$ matrix, U and V be $n \times k$ matrices, and C be a $k \times k$ nonsingular matrix. Then,*

$$\left(A + UCV^T\right)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + V^TA^{-1}U\right)^{-1}V^TA^{-1}, \tag{2.3}$$

provided that the matrix $C^{-1} + V^TA^{-1}U$ is nonsingular.

Proposition 2.3 can be proved using block matrix inversion as it is shown in Subsection 2.2. We will see that this formula is very useful in certain numerical computations, specially when an approximation of the inverse of the matrix $A + UCV^T$ is desired, provided that an incomplete factorization of A is available. The inverse of such a matrix is closely related to the Schur complement of a 2×2 block matrix, as it is shown in the next section. Using this relation, it is proposed a preconditioning method for the linear system (2.1).

2.2 Schur Complement

The Schur complement is a rich instrument in numerous fields of numerical analysis, statistics and matrix analysis, see [47, 95]. Let us consider a square matrix $\mathbf{A}$ of order $(n + k)$ partitioned in 2×2 block as

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{2.4}$$

where A_{11} and A_{22} are square matrices of dimensions $n \times n$ and $k \times k$, respectively. A_{12} and A_{21} are rectangular matrices of the corresponding complementary dimensions.

Definition 2.1. *If A_{11} is nonsingular, the Schur complement of $\mathbf{A}$ with respect to A_{11} (or Schur complement of A_{11} in $\mathbf{A}$) is defined as*

$$\mathbf{A}/A_{11} := A_{22} - A_{21}A_{11}^{-1}A_{12}. \tag{2.5}$$

Similarly, if A_{22} is nonsingular, the Schur complement of $\mathbf{A}$ with respect to A_{22} is defined as

$$\mathbf{A}/A_{22} := A_{11} - A_{12}A_{22}^{-1}A_{21}. \tag{2.6}$$

The Schur complement arises naturally in solving a system of linear equations such as

$$
\begin{aligned}
A_{11}x + A_{12}y &= a \\
A_{21}x + A_{22}y &= b,
\end{aligned}
\tag{2.7}
$$

where x, y, a and b are vectors of the corresponding dimensions. Assuming that A_{11} is nonsingular, multiplying the top equation by $A_{21}A_{11}^{-1}$ and then subtracting from the bottom equation one obtains:

$$
(A_{22} - A_{21}A_{11}^{-1}A_{12})y = b - A_{21}A_{11}^{-1}a,
$$

then, if $A_{22} - A_{21}A_{11}^{-1}A_{12}$ (the Schur complement of A_{11} in $\mathbf{A}$) is nonsingular, y is obtained from the last equation and x from $x = A_{11}^{-1}(a - A_{12}y)$.

An expression for the inverse of $\mathbf{A}$ in terms of A_{11}^{-1} and the inverse of the Schur complement of A_{11} in $\mathbf{A}$ is given by

$$
\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \\ -(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}A_{21}A_{11}^{-1} & (A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \end{bmatrix}. \tag{2.8}
$$

$$
\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \\ -(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}A_{21}A_{11}^{-1} & (A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \end{bmatrix}.
$$

Similarly, assuming that A_{22} and the Schur complement of A_{22} in $\mathbf{A}$ are both nonsingular, an equivalent expression for the inverse of $\mathbf{A}$ can be obtained,

$$
\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} = \begin{bmatrix} (A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & (A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1}A_{12}A_{22}^{-1} \\ -A_{22}^{-1} + A_{22}^{-1}A_{21}(A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & A_{22}^{-1} + A_{22}^{-1}A_{21}(A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1}A_{12}A_{22}^{-1} \end{bmatrix}. \tag{2.9}
$$

From Equations (2.2) and (2.9) one clearly observes that

$$
(A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} = A_{11}^{-1} + A_{11}^{-1}A_{12}(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}A_{21}A_{11}^{-1}, \tag{2.10}
$$

that corresponds to the Woodbury formula presented in Section 2.1. Thus, by approximating the inverse in Equation (2.9) a preconditioner for the linear system (2.1) can be obtained.

In the case of $\mathbf{A}$ being a symmetric and positive definite matrix, the following result holds [95].

Proposition 2.3. *Let* $\mathbf{A}$ *be an square matrix partitioned as in (2.4). Then* $\mathbf{A}$ *is an SPD matrix if and only if the Schur complements of* A_{11} *and* A_{22} *in* $\mathbf{A}$ *are SPD.*

2.3 Preconditioner Computation and Application

As mentioned, our goal is to compute a preconditioner for solving iteratively the linear system (2.1). Therefore, the inverse of the matrix $A_{11} - A_{12}A_{22}^{-1}A_{21}$ must be approximated in some way. The two main problems that are studied in this thesis can be written in this form:

- linear systems which are a modification of a nonsingular matrix,
- linear systems for which the coefficient matrix can be easily decomposed in this form.

Observe that the solution of (2.1) can be obtained from the solution of the equivalent augmented linear system

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x \\ -A_{22}^{-1}A_{21}x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{2.11}$$

From (2.2) and (2.9), the following relations between the linear operators (2.1) and (2.11) and their inverses can be established.

$$A_{11} - A_{12}A_{22}^{-1}A_{21} = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I \\ -A_{22}^{-1}A_{21} \end{bmatrix}, \tag{2.12}$$

and

$$(A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix}. \tag{2.13}$$

The preconditioner computation consists of obtaining an incomplete LU factorization for the augmented matrix in (2.11) that is used to approximate the inverse linear operator in (2.13) by direct preconditioning, i.e., solving the corresponding upper and lower triangular systems.

Assuming that we have calculated an incomplete LU factorization of $A_{11} \approx L_{11}U_{11}$, a preconditioner $\mathbf{M}$ is obtained by computing an incomplete block LDU factorization of the augmented matrix in (2.11). That is

$$\mathbf{M} = \begin{bmatrix} L_{11} & 0 \\ A_{21}U_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} U_{11} & L_{11}^{-1}A_{12} \\ 0 & I \end{bmatrix}, \tag{2.14}$$

where $S = A_{22} - A_{21}U_{11}^{-1}L_{11}^{-1}A_{12}$ is a $k \times k$ matrix that corresponds to the Schur complement of $\mathbf{A}$ with respect to A_{11}. Therefore, we avoid the explicit computation of the matrix $A_{11} - A_{12}A_{22}^{-1}A_{21}$ and a preconditioner from scratch. In this way, the method can be viewed as a technique to update a preconditioner previously computed. Algorithm 8 summarizes the preconditioner computation method.

Algorithm 8 Preconditioner update computation

```
**Input:** Matrices $A_{11}$, $A_{12}$, $A_{21}$, $A_{22}$.
**Output:** Triangular factors $L_{11}$, $U_{11}$, $L_S$, $U_S$.
1. Compute incomplete factorization $L_{11}U_{11} \approx A_{11}$.
2. Compute the blocks $T_1 := A_{21}U_{11}^{-1}$ and $T_2 := L_{11}^{-1}A_{12}$.
3. Compute $S = A_{22} - T_1T_2$.
4. Compute $L_SU_S \approx S$.
```

To maintain sparsity in these factors, some dropping strategy can be used when computing the matrices T_1 and T_2 in step 2. Additionally, an incomplete factorization of the Schur complement, i.e., $S \approx L_SU_S$, could be computed instead of computing an exact factorization. Note however that, if k is small enough, the matrix S can be factorized exactly with low computational cost. Furthermore, if the augmented matrix is symmetric, an IC factorization of A_{11} is preferred, hence $U_{11} = L_{11}^T$, and $T_1 = T_2^T$.

The preconditioning step for a Krylov subspace iterative method typically consists of obtaining the preconditioned vector $\bar{r} = M^{-1}r$ where M^{-1} is the preconditioner and r is the residual. M^{-1} should be a good sparse approximation of the inverse of the coefficient matrix of the linear system to be solved, this case (2.1). Thus, the preconditioning strategy proposed computes the preconditioned residual by applying Equation (2.13) with an incomplete factorization of the augmented matrix. That is, the preconditioned residual $\bar{r}$ is given by

$$\bar{r} = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix} r,$$

and it is computed from the solution of

$$\mathbf{M}\begin{bmatrix} \bar{r} \\ \bar{r}' \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}. \tag{2.15}$$

The preconditioning is done in three steps as Algorithm 9 shows. Step 2 in the algorithm represents the extra cost in the application of the preconditioner with respect to the case of non-updating the incomplete factorization of A_{11}. If T_1, T_2, L_S and U_S are kept sparse and $k \ll n$, this overhead is small and can be amortized even for moderate reductions on the number of iterations, see [24]. From now on, this technique will be referred to as Updated Preconditioner Method (UPD).

Algorithm 9 Preconditioner update application

```
**Input:** Matrices $L_{11}, U_{11}, T_1, T_2, L_S, U_S$ and residual vector $r$.
**Output:** Preconditioned vector $\bar{r}$
1. Solve the linear system $L_{11}\tilde{r} = r$.
2. Update $\tilde{r} \leftarrow \tilde{r} - T_2(L_S U_S)^{-1} T_1 \tilde{r}$.
3. Solve the linear system $U_{11}\bar{r} = \tilde{r}$.
```

To summarize, the preconditioning strategy proposed relies on computing a good approximation of the block A_{11} in the augmented matrix (2.11), which is updated in order to obtain a preconditioner to accelerate the convergence of a Krylov iterative method to solve the linear system (2.1).

Depending on the structure of the linear system to be solved, some variations can be introduced in the updated preconditioner method. Its application for different applications will be described in the following chapters.

2.4 Equivalent Augmented System

We also considered another approach that consists of solving the whole equivalent augmented system (2.11)

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{2.16}$$

by applying a Krylov subspace method. In this case, the preconditioner is the block LDU factorization given in Equation (2.14),

$$\mathbf{M} = \begin{bmatrix} L_{11} & 0 \\ A_{21}U_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} U_{11} & L_{11}^{-1}A_{12} \\ 0 & I \end{bmatrix}.$$

After obtaining a solution for the augmented linear system, the solution for the original one is recovered by selecting the first n components of the vector y. Note that any solution of (2.16) is a solution of (2.11), and vice versa. One of the main drawbacks of this approach is that an approximate solution is sought in a Krylov subspace of higher dimension that can be costly, specially when solving non-symmetric linear systems with the GMRES method.

There are many authors that have studied the problem of solving block structured linear systems, specifically by proposing methods for computing block incomplete factorization preconditioners [7, 28, 83, 90, 94]. A particular case is the one studied in [26], where block approximate inverse preconditioners to solve sparse nonsymmetric linear systems with iterative Krylov subspace methods are studied. In general, block incomplete factorizations seems to be more robust than standard (point) ones with respect to breakdowns, and often result in improved rates of convergence for difficult problems. Nevertheless, instabilities can happen in the block case as well [14]. Thus, numerical results with both, point and block ILU preconditioners for the augmented linear system, will be presented in order to compare their performance.

Chapter 3

Preconditioners for Almost Symmetric Linear Systems

In this chapter the update preconditioning technique described in Chapter 2 is applied for solving non-symmetric linear systems $Ax = b$, where the coefficient matrix A has a skew-symmetric part that can be well approximated by a skew-symmetric low-rank matrix. The method consists of updating a preconditioner obtained from the symmetric part of A. We present some results concerning to the approximation properties of the preconditioner and the spectral properties of the preconditioning technique. The results of the numerical experiments performed show that our strategy is competitive compared with other methods used in the bibliography to solve the same problem. The main results of this chapter has been submitted to Journal of Computational and Applied Mathematics [25].

3.1 Introduction

We are interested on the iterative solution of nonsingular, non-symmetric linear systems

$$Ax = b \tag{3.1}$$

where the matrix $A \in \mathbb{R}^{n\times n}$ is large, sparse and its skew-symmetric part has low rank or can be approximated by a skew-symmetric low-rank matrix. Consider $A = H + K$ where H and K are the symmetric and skew-symmetric parts of A, respectively. It is supposed that the skew-symmetric part can be written as $K = FCF^T + E$

where $F \in \mathbb{R}^{n\times s}$ is a full-rank rectangular matrix, $C \in \mathbb{R}^{s\times s}$ is a nonsingular skew-symmetric matrix with s even, $s \ll n$ and $\| E \| \ll 1$. Systems like this arise from the discretization of PDEs with certain Neumann boundary conditions, the discretization of integral equations [37] as well as path following methods.

Different strategies have been proposed to solve this kind of problems when $E = 0$, see Subsection 1.5.1. Among them, of particular interest is the SCM method presented in [37]. The method permits the application of short-term formulas and obtains an approximate solution by applying the MINRES method $s + 1$ times. The authors also suggest that it can be applied as a preconditioner for GMRES for the more general case $E \neq 0$. Thus, it will be compared with our proposal in this chapter. Also, the Progresive GMREs (PGMRES) [12] and the Induced Dimension Reduction IDR(s) method [80, 82] can be applied to solve this type of problems.

This chapter is organized as follows. In Section 3.2 the proposed preconditioning technique is described. Section 3.3 is devoted to analyze the approximations properties of the preconditioned matrix. In Section 3.4 the technique used to approximate the skew-symmetric part is described. The results of the numerical experiments for some real and artificial problems are presented in Section 3.5. Finally, some conclusions are given in Section 3.6.

3.2 Updated preconditioner method

To obtain a preconditioner for the system (3.1), an approximate LU factorization of the augmented matrix

$$\mathbf{A} = \begin{bmatrix} H + E & F \\ F^T & -C^{-1} \end{bmatrix} \tag{3.2}$$

is computed. This matrix is a particular case of the matrix in (2.4), with $A_{11} = H + E$, $A_{12} = F$, $A_{21} = F^T$, $A_{22} = -C^{-1}$. This preconditioner can be viewed as a low-rank update of an incomplete LU factorization of the symmetric part H.

The preconditioner $\mathbf{M}$ is obtained by computing an incomplete LU of the matrix $\mathbf{A}$ in (3.2). Assuming that we have calculated an incomplete LU factorization of the symmetric part H, $\hat{H} = L_H D_H L_H^T$, one has

$$\mathbf{M} = \begin{bmatrix} L_H & 0 \\ F^T L_H^{-T} D_H^{-1} & I \end{bmatrix} \begin{bmatrix} D_H & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} L_H^T & D_H^{-1} L_H^{-1} F \\ 0 & I \end{bmatrix} \tag{3.3}$$

with $R = -(C^{-1} + F^T L_H^{-T} D_H^{-1} L_H^{-1} F)$. The computation of the preconditioner is done following the steps presented in Algorithm 8. In this particular case,

Algorithm 10 Preconditioner update computation

```
**Input:** Matrices $H, F, C$.
**Output:** Triangular factors $L_H$, $D_H$, $L_R$, $U_R$, $T$.
1. Compute incomplete factorization $L_H D_H L_H^T \approx H$.
2. Compute block $T$ by solving $L_H T = F$.
3. Compute $R = -(C^{-1} + T^T D_H^{-1} T)$.
4. Compute $L_R U_R = R$.
```

Step 2 in Algorithm 10, may involve a sparsification of the matrix T after its computation to reduce the amount of fill-in introduced. Note that the factorization in step 4 is done exactly when $s \ll n$. Otherwise, an incomplete factorization of R may be necessary to control the amount of fill-in.

The application of the preconditioner is done by solving the triangular systems of the LU factorization of **M**, Equation (3.3). Thus, the preconditioning step is done by solving linear systems of the form

$$\mathbf{M} \begin{bmatrix} \bar{r} \\ \bar{r}' \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} .$$

Algorithm 11 details the steps to obtain the preconditioned residual vector $\bar{r}$.

Algorithm 11 Preconditioner update application

```
**Input:** Matrices $L_H$, $D_H$, $T$, $L_R$, $U_R$ and residual vector $r$.
**Output:** Preconditioned vector $\bar{r}$
1. Solve the linear system $L_H D_H \tilde{r} = r$.
2. Update $\tilde{r} \leftarrow \tilde{r} - T^T (L_R U_R)^{-1} T \tilde{r}$.
3. Solve the linear system $L_H^T \bar{r} = \tilde{r}$.
```

The computation and application of the preconditioner is inexpensive, that is, with low computational cost, provided that $s \ll n$. Note that step 2 includes the solution of a $s \times s$ linear system which can be done with a direct method.

3.3 Approximation properties of the updated preconditioner

In this section we study the approximation properties of the proposed updated preconditioner. We recall that $A = H + K$ where H and K are the symmetric and skew-symmetric parts of A, respectively, and $K = FCF^T + E$ where $F \in \mathbb{R}^{n\times s}$ is a full-rank rectangular matrix, $C \in \mathbb{R}^{s\times s}$ is a nonsingular skew-symmetric matrix with s even, $s \ll n$ and $\| E \| \ll 1$. We denote $H_E = H + E$.

The proposed preconditioning strategy relies on computing a good approximation of the augmented matrix in Equation (3.2) which is used to accelerate the convergence of a Krylov iterative method. Solving (3.1) with a preconditioned Krylov method involves the computation of matrix-vector products with A and an approximation of its inverse operator A^{-1} in the preconditioning step. We have the following relations between the linear operators A and $\mathbf{A}$,

$$A = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} H_E & F \\ F^T & -C^{-1} \end{bmatrix} \begin{bmatrix} I \\ CF^T \end{bmatrix} = \begin{bmatrix} I & O \end{bmatrix} \mathbf{A} \begin{bmatrix} I \\ CF^T \end{bmatrix} \tag{3.4}$$

and their inverses,

$$A^{-1} = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} H_E & F \\ F^T & -C^{-1} \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix} = \begin{bmatrix} I & O \end{bmatrix} \mathbf{A}^{-1} \begin{bmatrix} I \\ O \end{bmatrix}, \tag{3.5}$$

provided that H_E is nonsingular. Note that if H is a well conditioned matrix and $\| E \| \ll 1$, this condition can be easily satisfied (see Theorem 2.3.4 in [43]). Next result relates the condition numbers of the matrices A and $\mathbf{A}$.

Theorem 3.1. *Let* $\mathbf{A}$ *be the matrix given by Equation* (3.2) *associated to the linear system (3.1). Assume that* FCF^T *is a reduced unitary diagonalization of the matrix* $K - E$. *Then,*

$$\operatorname{cond}(A) \leq \operatorname{cond}(\mathbf{A})\sqrt{1 + \sigma_1^2(C)}, \tag{3.6}$$

where $\sigma_1(C)$ *is the maximum singular value of* C.

Proof. Considering the Equations (3.4) and (3.5), one has

$$\operatorname{cond}(A) = \|A\|_2\|A^{-1}\|_2 = \left\| \begin{bmatrix} I & O \end{bmatrix} \mathbf{A} \begin{bmatrix} I \\ CF^T \end{bmatrix} \right\|_2 \left\| \begin{bmatrix} I & O \end{bmatrix} \mathbf{A^{-1}} \begin{bmatrix} I \\ O \end{bmatrix} \right\|_2$$
$$\leq \operatorname{cond}(\mathbf{A}) \left\| \begin{bmatrix} I \\ CF^T \end{bmatrix} \right\|_2 .$$

Since FCF^T is a reduced unitary diagonalization of $K - E$, then $F^TF = I_s$ and $C \in \mathbb{R}^{s\times s}$ is a block diagonal matrix of the form

$$\begin{bmatrix} 0 & \lambda_i \\ -\lambda_i & 0 \end{bmatrix},$$

where λ_i with $i = 1, ..., s/2$ are the absolute values of the complex eigenvalues of C. Under these conditions the nonzero eigenvalues of the matrices FC^TCF^T and C^TC are equal and positive since $C^TC = \operatorname{diag}(\lambda_1^2, \lambda_1^2, \dots, \lambda_{s/2}^2\lambda_{s/2}^2)$. Therefore,

$$\left\| \begin{bmatrix} I \\ CF^T \end{bmatrix} \right\|_2^2 = \rho(I + FC^TCF^T) = \rho(I + C^TC) = 1 + \sigma_1^2(C).$$

□

This proposition suggests that one can expect a faster convergence of the iterative method used to solve the linear system (3.1) if the condition number of the matrix $\mathbf{A}$ is improved with a proper preconditioner.

To study the quality of the updated preconditioner, first we evaluate the approximation error norm. A comparison with the non-updated preconditioner is also presented. These preconditioners are given by

$$\mathbf{M} = \mathbf{LDU} = \begin{bmatrix} \hat{H} & \hat{F} \\ \hat{F}^T & -C^{-1} \end{bmatrix} \quad \text{and} \quad \mathbf{M_0} = \begin{bmatrix} \hat{H} & O \\ O & -C^{-1} \end{bmatrix}. \tag{3.7}$$

The expression for $\mathbf{M}$ is obtained multiplying the LDU factors in Equation (3.3). Assuming that in step 2 of the computation of the preconditioner a sparsification of the matrix T has been done, which is denoted by $\hat{T}$, one has that the matrix F is approximated by $\hat{F} = L_H\hat{T}$. Moreover, we assume that R is factorized exactly.

Theorem 3.2. *Let $\hat{H} = L_HD_HL_H^T$ be an incomplete LDU factorization of H. Let $\mathbf{M}$ and $\mathbf{M_0}$ be the matrices given in (3.7). Let $\epsilon = \| \hat{H} - H \|_F^2$, $\delta = \| L_H \|_F^2$, $\gamma = \| E \|_F^2$ and*

$c = \| \hat{T} - T \|_F^2$. Then

$$\| \mathbf{M} - \mathbf{A} \|_F \leq \sqrt{\epsilon + \gamma + 2\delta c}. \tag{3.8}$$

Moreover, if $c \leq \frac{\|F\|_F^2}{\delta}$ then

$$\| \mathbf{M} - \mathbf{A} \|_F \leq \| \mathbf{M_0} - \mathbf{A} \|_F \,. \tag{3.9}$$

Proof. From (3.7) we have

$$\mathbf{M} - \mathbf{A} \quad = \quad \begin{bmatrix} \hat{H} - H_E & \hat{F} - F \\ \hat{F}^T - F^T & O \end{bmatrix} = \begin{bmatrix} \hat{H} - H_E & L_H(\hat{T} - T) \\ (\hat{T} - T)^T L_H^T & O \end{bmatrix}.$$

Then

$$\begin{aligned} \| \mathbf{M} - \mathbf{A} \|_F^2 \quad &= \quad \| \hat{H} - H_E \|_F^2 + 2 \| L_H(\hat{T} - T) \|_F^2 \\ &\leq \quad \| \hat{H} - H \|_F^2 + \| E \|_F^2 + 2(\| L_H \|_F^2 \| \hat{T} - T \|_F^2) \\ &= \quad \epsilon + \gamma + 2\delta c \,. \end{aligned}$$

If $c \leq \frac{\|F\|_F^2}{\delta}$, then

$$\| \mathbf{M} - \mathbf{A} \|_F^2 \leq \| \hat{H} - H_E \|_F^2 + 2\delta c \leq \| \hat{H} - H_E \|_F^2 + 2 \| F \|_F^2 = \| \mathbf{M_0} - \mathbf{A} \|_F^2$$

□

As it could be expected, the above theorem shows that the approximation degree of $\mathbf{M}$ depends on $\hat{H}$ and $\hat{F}$ being a good approximation of H and F, respectively, and $\| E \| \ll 1$. Moreover, we have proved that if these approximations are good enough, the updated preconditioner $\mathbf{M}$ is closer to the matrix $\mathbf{A}$ than the initial one, $\mathbf{M_0}$.

Theorem 3.3. *Let the assumptions of Theorem 3.2 hold, then the preconditioned matrix $\mathbf{M^{-1}A}$ can be written as*

$$\mathbf{M^{-1}A} = \mathbf{I} - \mathbf{M^{-1}E_A}, \tag{3.10}$$

where

$$\| \mathbf{M^{-1}E_A} \|_F \leq \| \mathbf{M^{-1}} \|_F \sqrt{\epsilon + \gamma + 2\delta c} \tag{3.11}$$

Proof. Let $\mathbf{E_A} = \mathbf{M} - \mathbf{A}$, hence

$$\| \mathbf{M^{-1}E_A} \|_F^2 = \| \mathbf{M^{-1}}(\mathbf{M} - \mathbf{A}) \|_F^2 \leq \| \mathbf{M^{-1}} \|_F^2 \| \mathbf{M} - \mathbf{A} \|_F^2 \leq \| \mathbf{M^{-1}} \|_F^2 \; (\epsilon + \gamma + 2\delta c)$$

□

Corollary 3.4. *Let the assumptions of Theorem 3.3 hold. Then, the eigenvalues of the preconditioned matrix $\mathbf{M^{-1}A}$ are clustered at 1 in the right half complex plane provided that $\| \mathbf{M^{-1}} \|_F \sqrt{\epsilon + \gamma + 2\delta c} < 1$.*

Proof. Defining $\rho = \| \mathbf{M^{-1}} \|_F \sqrt{\epsilon + \gamma + 2\delta c}$, it inmediatelly follows from the bound in Equation (3.11) and Equation (3.10) that there is a cluster of eigenvalues of $\mathbf{M^{-1}A}$ at 1 in the right half complex plane with radius equal to $\rho < 1$. □

Corollary 3.4 basically means that the quality of the preconditioner depends on the accuracy of the approximations computed for the symmetric and skew-symmetric parts of A. With a clustered spectrum one can expect a faster convergence of an iterative method although we recall that other aspects may influence the behaviour of Krylov-based iterative methods.

Next, we consider the case in which the symmetric part of A is indeed positive definite. The following result characterizes the spectrum of $\mathbf{M^{-1}A}$.
Theorem 3.5. *Let* $\mathbf{A}$ *and* $\mathbf{M}$ *be the matrices given by*

$$\mathbf{A} = \begin{bmatrix} H_E & F \\ F^T & -C^{-1} \end{bmatrix} \quad \text{and} \quad \mathbf{M} = \begin{bmatrix} \hat{H} & \hat{F} \\ \hat{F}^T & -C^{-1} \end{bmatrix}.$$

Assume that H is SPD, F and $\hat{F}$ have full rank s, and the error matrix $E_F = F - \hat{F}$ has rank p, $p \le s$. Then, the eigenvalues of $\mathbf{M^{-1}A}$ are either one or real positive and bounded by

$$\lambda_{\min}(\hat{H}^{-1}H) \le \lambda \le \lambda_{\max}(\hat{H}^{-1}H)\,, \tag{3.12}$$

or complex bounded by

$$|\lambda| \le 1 + \frac{\| C \|_2 \| E_F \|_2}{\sqrt{1 + \sigma^2_{min}(\hat{F}C^T)}} \tag{3.13}$$

where σ_{min} represents the smallest singular value.

Proof. The technique to prove the result is standard and similar to the one that can be found in [18]. The eigenvalues and eigenvectors of $\mathbf{M^{-1}A}$ are solutions of the following generalized eigenvalue problem $\mathbf{A}w = \lambda \mathbf{M}w$ written as

$$\begin{bmatrix} H_E & F \\ F^T & -C^{-1} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} \hat{H} & F - E_F \\ (F - E_F)^T & -C^{-1} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

where the eigenvector w is partitioned according to the block structure of the matrix **A**.

One has equivalently that,

$$\begin{aligned} H_E x + Fy &= \lambda \hat{H} x + \lambda F y - \lambda E_F y \,, \\ F^T x &= \lambda F^T x - \lambda E_F^T x + (1-\lambda) C^{-1} y \,. \end{aligned} \tag{3.14}$$

We distinguish the following cases:

1. $x = 0$. From the second equation in (3.14) it follows that $0 = (1-\lambda)C^{-1}y$. Then $\lambda = 1$ and therefore $E_F y = 0$ from the first equation. Since $y \in \ker E_F$ that has dimension $s-p$, we obtain that there are $s-p$ eigenvectors $\begin{bmatrix} 0 \\ y \end{bmatrix}$ associated to the unit eigenvalue.

2. $x \neq 0$. We consider three cases:

   (a) $F^T x = 0$. Since F has rank s it follows that there are $n-s$ linearly independent vectors satisfying this condition. From the second equation we have $\lambda E_F^T x = (1-\lambda)C^{-1}y$. Although x is real, the eigenpair can be complex. Thus, the conjugate transpose is $\bar{\lambda} x^T E_F = (1-\bar{\lambda}) y^H C^{-T}$. By multiplying the first equation by x^T and substituting one has

   $$x^T H_E x = \lambda x^T \hat{H} x - \frac{\lambda}{\bar{\lambda}} (1-\bar{\lambda}) y^H C^{-T} y,$$

   or equivalently, since $H_E = H + E$

   $$x^T H x + x^T E x = \lambda x^T \hat{H} x - \frac{\lambda}{\bar{\lambda}} (1-\bar{\lambda}) y^H C^{-T} y \,.$$

   We recall that E and C are skew-symmetric matrices. Therefore, in the equation above the terms $x^T E x$ and $y^H C^{-T}$ nullify. Then

   $$x^T H x = \lambda x^T \hat{H} x \,.$$

   Since H and $\hat{H}$ are SPD matrices the eigenpairs are real, and by Courant-Fischer Minimax Theorem (see [43]) it follows that the eigenvalues are bounded by

   $$\lambda_{\min}(\hat{H}^{-1}H) \leq \lambda \leq \lambda_{\max}(\hat{H}^{-1}H) \,.$$

(b) $F^Tx \neq 0$ and $E_F^Tx = 0$. In this case $s - p$ linearly independent vectors satisfy these conditions. The second equation reduces to

$$(1-\lambda)F^Tx = (1-\lambda)C^{-1}y$$

and it is satisfied for eigenvalues equal to 1 or when $y = CF^Tx$. In this last case, by substituting in the first equation one has

$$H_Ex + FCF^Tx = \lambda\hat{H}x + \lambda FCF^Tx - \lambda E_FCF^Tx \ .$$

Multiplying by x^T we obtain

$$x^T(H_E + FCF^T)x = \lambda x^T(\hat{H} + FCF^T)x \ .$$

Since FCF^T is skew-symmetric, then reasoning similar as in 2.(a) these eigenvalues are bounded by

$$\lambda_{\min}(\hat{H}^{-1}H) \leq \lambda \leq \lambda_{\max}(\hat{H}^{-1}H) \ .$$

(c) $F^Tx \neq 0$ and $E_F^Tx \neq 0$. Multiplying the first equation by x^H and the second by y^H one has

$$\begin{array}{rcl} x^HH_Ex + x^HFy & = & \lambda x^H\hat{H}x + \lambda x^H\hat{F}y \ , \\ y^HF^Tx & = & \lambda y^H\hat{F}^Tx + (1-\lambda)y^HC^{-1}y \ . \end{array} \tag{3.15}$$

Adding both equations we obtain

$$x^HH_Ex + 2Re(x^HFy) - y^HC^{-1}y = \lambda(x^H\hat{H}x + 2Re(x^H\hat{F}y) - y^HC^{-1}y) \ .$$

As in case 2.(a), since E and C are skew-symmetric matrices, the equation above simplifies to

$$x^HHx + 2Re(x^HFy) = \lambda(x^H\hat{H}x + 2Re(x^H\hat{F}y)) \ .$$

We consider two possibilities: if $x^H\hat{H}x + 2Re(x^H\hat{F}y) = 0$, the eigenvalue λ can be complex. In this case from the second equation in (3.14) one has $(F^Tx - C^{-1}y) = \lambda(\hat{F}^Tx - C^{-1}y)$, equivalent to $(CF^Tx - y) = \lambda(C\hat{F}^Tx - y)$.

Note that $C\hat{F}^T x - y \neq 0$ since we are considering $E_F^T x \neq 0$. It follows that

$$
\begin{aligned}
|\lambda| &= \frac{\| CF^T x - y \|_2}{\| C\hat{F}^T x - y \|_2} = \frac{\| \begin{bmatrix} CF^T & -I \end{bmatrix} w\|_2}{\| \begin{bmatrix} C\hat{F}^T & -I \end{bmatrix} w \|_2} \\
&\leq 1 + \frac{\| \begin{bmatrix} CE_F^T & O \end{bmatrix} w \|_2}{\| \begin{bmatrix} C\hat{F}^T & -I \end{bmatrix} w \|_2} \leq 1 + \frac{\| C \|_2 \| E_F \|_2 \| w \|_2}{\| \begin{bmatrix} C\hat{F}^T & -I \end{bmatrix} w \|_2} \\
&= 1 + \frac{\| C \|_2 \| E_F \|_2}{\| \begin{bmatrix} C\hat{F}^T & -I \end{bmatrix} \frac{w}{\|w\|_2} \|_2} \leq 1 + \frac{\| C \|_2 \| E_F \|_2}{\sqrt{1 + \sigma_{min}^2(\hat{F}C^T)}}
\end{aligned}
$$

where $\sigma_{min}(\hat{F}C^T)$ represents the smallest singular value of a matrix $\hat{F}C^T$.

On the other hand, if $x^H \hat{H} x + 2Re(x^H \hat{F} y) \neq 0$ then $\lambda \in \mathbb{R}$. By subtracting the transpose of the second equation from the first one in (3.15), we obtain the same equation and the corresponding bound as in 2.(a). Note that $2p$ is the maximum number of complex eigenvalues.

$\square$

To illustrate the bounds deduced in this section we consider the matrix $ADD20$ from the University of Florida sparse matrix collection [35]. This matrix has order $2,395$ with $13,151$ nonzero elements and condition number $\text{cond}(A) = 1.7637 \times 10^4$. We approximate its skew-symmetric part with a matrix of rank $s = 42$, giving an error matrix with norm $\|E\|_2 = 9.88 \times 10^{-5}$. An incomplete Cholesky factorization of H with dropping parameter equal to 10^{-4} was computed. The matrix T was also sparsified with a dropping threshold of 10^{-3} with respect to its maximum absolute value. The results were obtained using MATLAB version 2016a.

First, we studied the bound (3.6) of Theorem 3.1. We computed for this matrix $\text{cond}\,(\mathbf{A})\sqrt{1+\sigma_1^2(C)} = 1.0149 \times 10^8$, that is greater than $\text{cond}(A)$, satisfying the bound.

Concerning Theorem 3.2, the values of the parameters involved in the statement were $\epsilon = 3.0834 \times 10^{-6}$, $\gamma = 2.1762 \times 10^{-7}$, $\delta = 335.6455$, $c = 2.6701 \times 10^{-9}$ and $\frac{\|F\|_F^2}{\delta} = 4.8627 \times 10^{-9}$. The quantities involved in Equations (3.8) and (3.9) are shown in Table 3.1 that clearly satisfy the inequalities.

The bound in Theorem 3.3 is also satisfied since it was obtained 3.0313 and 227.8091 for the left and right side values in inequality (3.11), respectively.

$\lVert \mathbf{M} - \mathbf{A} \rVert_F$	$\sqrt{\epsilon + \gamma + 2\delta c}$	$\lVert \mathbf{M_0} - \mathbf{A} \rVert_F$
1.8×10^{-3}	2.3×10^{-3}	2.6×10^{-3}

TABLE 3.1: Bounds for Theorem 3.2

Finally, with respect Theorem 3.5, the bounds computed according to the Equations (3.12) and (3.13) are $\lambda_{\min}(\hat{H}^{-1}H) = 0.8599$ and $\lambda_{\max}(\hat{H}^{-1}H) = 1.1311$ for the real eigenvalues, and $|\lambda| \leq 1.0208$ for the complex ones. These bounds are satisfied since the minimum and maximum real eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ are 0.9384 and 1.1309, respectively. Moreover, the norm of the largest complex eigenvalue was 1.0101. Figure 3.1 illustrates the spectrum of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$. It is observed that the eigenvalues are clustered at one in the right half complex plane.

To end this part, in Figure 3.2 we can observe that the spectrum of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ satisfies the bounds deduced in this section, independently of the threshold parameters chosen for the incomplete Cholesky factorization of H, and for the auxiliary matrix T.

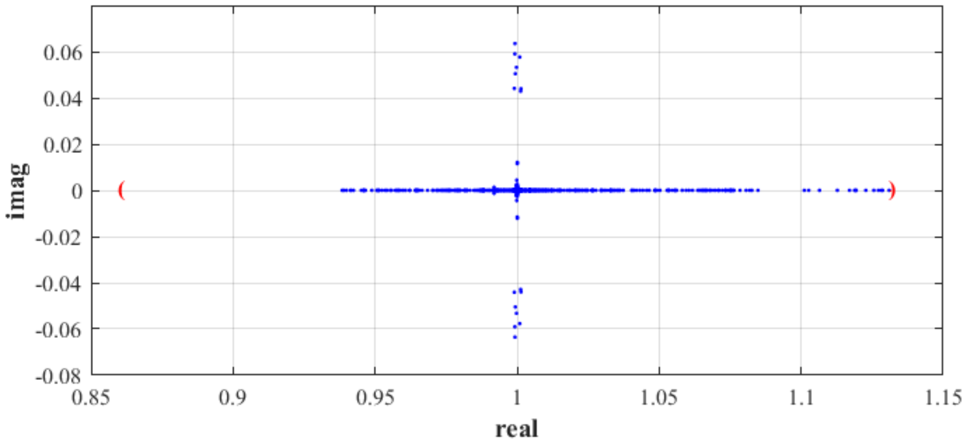
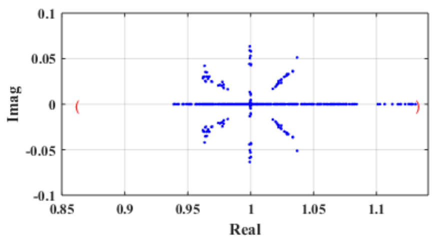


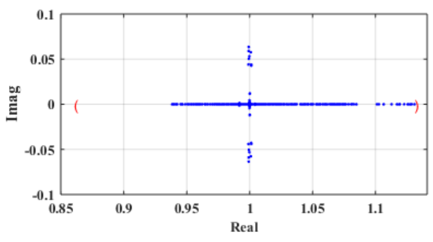
FIGURE 3.1: Spectrum of $\mathbf{M}^{-1}\mathbf{A}$ to illustrate the bounds of Theorem 3.5. The bounds for the real eigenvalues are indicated with a red parenthesis.

3.4 Low-rank approximation of the skew-symmetric part

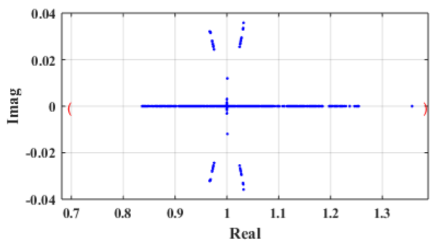
If the skew-symmetric part of a matrix is not low-rank, a crucial step is to obtain a good low-rank representation for it. There are different techniques that can be used.



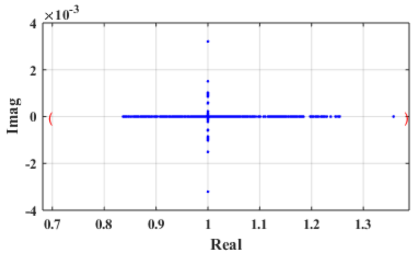
(A) Dropping IC of H with 10^{-4} and T with 10^{-1}



(B) Dropping IC of H with 10^{-4} and T with 10^{-5}



(C) Dropping IC of H with 10^{-3} and T with 10^{-1}



(D) Dropping IC of H with 10^{-3} and T with 10^{-5}

FIGURE 3.2: Spectrum of $\mathbf{M}^{-1}\mathbf{A}$ for different dropping parameters of T and the IC factorization of H. The bounds for the real eigenvalues are indicated with a red parenthesis.

Among them, as it was mentioned in Subsection 1.5.1, we use the Sparse Column Row aproximation (SCR) method presented in [19] that is well suited for this task. Its Matlab's implementation code was downloaded from [84]. With this method we are able obtain an approximation of the skew-symmetric part K of the form FCF^T, where F consists of columns of K and C is a $s \times s$ skew-symmetric matrix with s even.

In general, a low-rank approximation of a matrix K is commonly written in the form

$$K \approx FCG,$$

where the matrices F and G have full rank s, the order of the approximation, and C is a nonsingular matrix.

A widely used low-rank approximation is the singular value decomposition (SVD), which is known to be optimal in the sense of achieving the minimum error Frobenius norm. There are stable direct methods for its computation, see [51, 61].

Another strategy is to consider truncated pivoted QR approximations. For example in [86], the author proposes four algorithms to compute this type of approximations to a sparse matrix, based on the Gram–Schmidt algorithm and the Householder triangularization. Two of these methods are studied in detail in [19], the sparse pivoted QR (SPQR) and sparse column row (SCR) approximations, described below.

In general, a QR factorization has the form $KP = QR$, where K is the matrix to be approximated, P is a permutation matrix, Q is an orthogonal matrix and R is an upper triangular matrix. A rank-s approximation to K can be obtained by partitioning the QR factorization. Let $B = KP$, writing

$$B = \begin{pmatrix} B_1^{(s)} & B_2^{(s)} \end{pmatrix} = \begin{pmatrix} Q_1^{(s)} & Q_2^{(s)} \end{pmatrix} \begin{pmatrix} R_{11}^{(s)} & R_{12}^{(s)} \\ 0 & R_{22}^{(s)} \end{pmatrix},$$

where $B_1^{(s)}$ has s columns. It is obtained the approximation

$$B \approx \tilde{B} = Q_1^{(s)} \begin{pmatrix} R_{11}^{(s)} & R_{12}^{(s)} \end{pmatrix}.$$

Observe that, since $Q_2^{(s)}$ is orthogonal, the error in the approximation is

$$\epsilon_c = ||B - \tilde{B}|| = ||R_{22}^{(s)}||. \tag{3.16}$$

To compute the decomposition, a column of K can be successively brought at a time and use it to compute an additional column of Q and row of R. The process of selecting columns is called column pivoting, and the classical choice of a column is the one that corresponds to the column of $R_{22}^{(s-1)}$ of largest norm. A problem arises using the Gram-Schmidt algorithm, since Q is in general not sparse even when K has this property. Therefore, if s is very large it could be difficult to store Q. A possible solution is, observing that $Q = BR^{-1}$, the action of Q on a vector can be calculated by operations involving the matrices B and R without computing explicitely the matrix Q. The quasi-Gram-Schmidt method proposed in [19] takes advantage of this idea to compute a pivoted, Q-less PQR factorization, called sparse pivoted QR (SPQR) factorization. The matrix

$$(B_1^{(s)} R_{11}^{(s)^{-1}}) \begin{pmatrix} R_{11}^{(s)} & R_{12}^{(s)} \end{pmatrix}$$

is the SPQR approximation. The algorithm avoids the storage for Q, and also it replaces dense products involving Q with sparse products involving columns of K. A problem of the algorithm may be a progressive loss of orthogonality in the matrix BR^{-1}. However, an analysis of the quasi-Gram-Schmidt algorithm shows that the loss of orthogonality is not significative [85].

Based on SPQR, a sparse column row approximation can be derived by applying SPQR to the columns and rows of K, respectively. Specifically, first the quasi-Gram-Schmidt algorithm is applied to the columns of K to get a representative set of columns F of K, and an upper triangular matrix R that corresponds to $R_{11}^{(s)}$. Let the error in the corresponding reduced rank decomposition be ϵ_c, see Equation (3.16). Secondly, repeat the application of the algorithm to K^T to obtain a set G of rows and another upper triangular matrix S, with approximation error ϵ_r. In [86] the authors show that the matrix C that minimizes $||K - FCG||^2$, is

$$C = R^{-1}R^{-T}(F^TKG^T)S^{-1}S^{-T}, \text{with } ||K - FCG||^2 \leq \epsilon_c^2 + \epsilon_r^2.$$

Thus, the sparse column row approximation (SCR) approximation is given by

$$K \approx FR^{-1}R^{-T}(F^TKG^T)S^{-1}S^{-T}G. \tag{3.17}$$

As shown in [19], SPQR requires less time to be computed than SVD, especially, for large values of s, the rank of the approximation matrix. Regarding storage, SVD requires $2ns$ floating-point words, whereas SPQR requires only s^2 words.

For all these reasons, the SCR method will be used in this thesis to obtain a low-rank approximation of the skew-symmetric part K of a matrix. But, in this case it suffices to compute an SPQR approximation of K. Following the nomenclature adopted in the description of the SCR method, we get that $G = F^T$ and $S = R$. Then

$$K = FCF^T \text{ , with } C = -R^{-1}R^{-T}(F^TKF)R^{-1}R^{-T}. \tag{3.18}$$

Note that the matrix C is skew-symmetric. In fact,

$$C^T = -R^{-1}R^{-T}(F^TK^TF)R^{-1}R^{-T} = -R^{-1}R^{-T}F^T(-K)FR^{-1}R^{-T} = -C.$$

We show several examples of the quality of the approximation that can be obtained with the SCR method. A MATLAB version of the SCR algorithm has been used for the experiments. Consider the non-symmetric matrix CIRCUIT_1 from [35]. This is a square matrix of order 2624, whose skew-symmetric part K has 26400 nonzero elements.

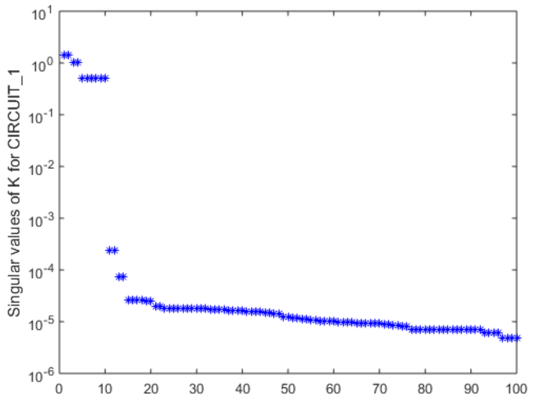


FIGURE 3.3: Singular values of K for the matrix CIRCUIT_1

Figure 3.3 shows a plot of the 100 biggest singular values of K. Observe that, since there is a big jump between the 10th and the 11th, a rank-10 approximation may be adequate. The approximation computed with the SCR algorithm has the same 10 biggest singular values than K, and the error of the approximation corresponds exactly to the 11th one, which is 2.4336×10^{-4}. Furthermore, the time spent for the computation is less than one hundred of a second, and only 30 nonzero elements are stored in memory, which is quite sparse compared with the matrix K as depicted in Figure 3.4.

For the matrix ADD20 a rank-42 was computed. The corresponding approximation error observed is 9.9×10^{-5} and requires only 49 nonzero elements, see Figure 3.5. As a conclusion, there are two interesting observations of the SCR method to be made. The error of a rank-s approximation is the $s + 1$ singular value of the matrix considered, and the approximation obtained is quite sparse.

Finally, we mention that in [36] the authors present an algorithm for computing a low-rank approximation to a sparse matrix based on a truncated LU factorization with column and row permutations (LU_CRTP). In this method, the selection of columns and rows at each step of the block factorization uses tournament pivoting based on QR. The authors claim that LU_CRTP provides a good balance between accuracy and speed. We have applied this method to the matrix CIRCUIT_1 as well. While the error and memory storage results obtained are similar to thouse reported with the SCR method, the execution time observed is much larger. Thus, we discarded its use in this work.

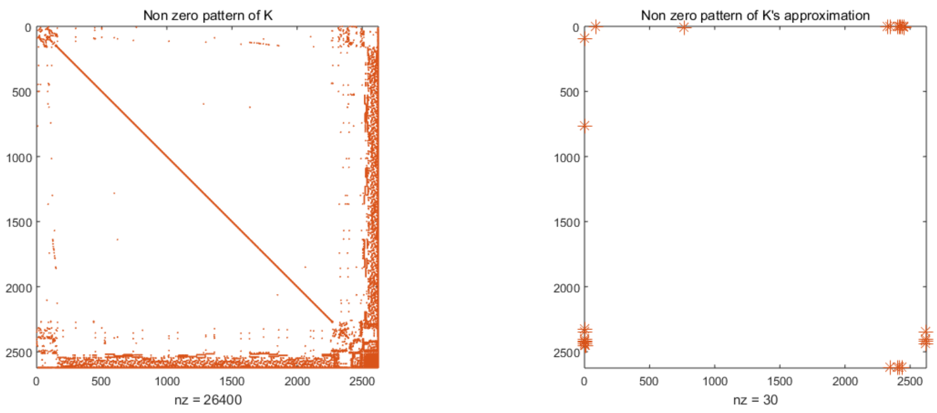


FIGURE 3.4: Nonzero patterns of K (left) and its approximation (right). Matrix CIRCUIT_1

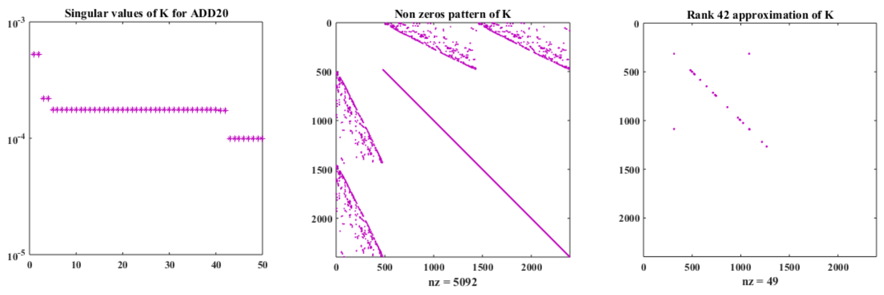


FIGURE 3.5: Rank-42 approximation of K for ADD20 using SCR

3.5 Numerical Experiments

In this section we compare the Updated Preconditioner Method, referred to as UPD, with the SCM method used as preconditioner and also an incomplete LU factorization of the symmetric part H. The iterative methods used are the full GMRES, restarted GMRES(m) and BiCGSTAB. The experiments have been performed with MATLAB version 2016a. In some case, the SCM method will be tested as a complete method. The iterative methods were run until the relative initial residual was reduced to 10^{-8}, allowing a maximum number of 2000 iterations. The incomplete factorization of the symmetric part H was computed with MATLAB's function **ilu**() that implements an ILU factorization with threshold [72]. We present the results obtained for different problems that appear in the bibliography and also using some matrices obtained from the University of Florida sparse matrix collection. Concerning the SCM preconditioner, it requires $s + 1$ applications of MINRES, which could be prohibitive to apply at each iteration of GMRES applied to the preconditioned system. Thus, as the authors suggest in [37], since s of these applications are needed to solve a linear system with multiple right-hand sides, the solution of this system is computed once and reused at each GMRES iteration.

Furthermore, we present the numerical results by considering the solution of the equivalent augmented system, which is solved with four different techniques, the GMRES and BiCGSTAB methods, both preconditioned with a point ILU computed for the augmented matrix and the block ILU described in Section 2.4.

3.5.1 A class of simple examples

The first example was used in [37] to show the performance of SCM method. Consider the block-diagonal matrix

$$A = \begin{bmatrix} \Lambda_- & & \\ & \Lambda_+ & \\ & & Z \end{bmatrix},$$

where $\Lambda_- = diag(\lambda_1, \dots, \lambda_p)$, $\Lambda_+ = diag(\lambda_{p+1}, \dots, \lambda_{n-s})$ with $\lambda_1, \dots, \lambda_p$ uniformly spaced in $[-\beta, -\alpha]$ and $\lambda_{p+1}, \dots, \lambda_{n-s}$ uniformly spaced in $[\alpha, \beta]$ for some positive constants $\alpha < \beta, p \ll n$ and s even such that $2 \leq s \ll n$. $Z = tridiag(-\gamma, 1, \gamma) \in \mathbb{R}^{s\times s}$ with $\gamma > 0$. The matrix A is indefinite with eigenvalues

- $\lambda_1, \dots, \lambda_p \in [-\beta, -\alpha]$,
- $\lambda_{p+1}, \dots, \lambda_{n-s} \in [\alpha, \beta]$,
- s complex eigenvalues of Z.

In this case $A = H + K$ with

$$H = \begin{bmatrix} \Lambda_- & & \\ & \Lambda_+ & \\ & & I \end{bmatrix}, \quad K = FCF^T = \begin{bmatrix} O & & \\ & O & \\ & & Z - I \end{bmatrix}.$$

For this first problem $E = 0$, that is, the skew-symmetric part is not approximated.

We study how to solve the system (3.1) with b equal to $1/\sqrt{n}$ in all its components, $n = 10^5$, $\alpha = 1/8$, $\beta = 1$, $\gamma = 1$. Figure 3.6 compares the CPU time of the different methods tested.

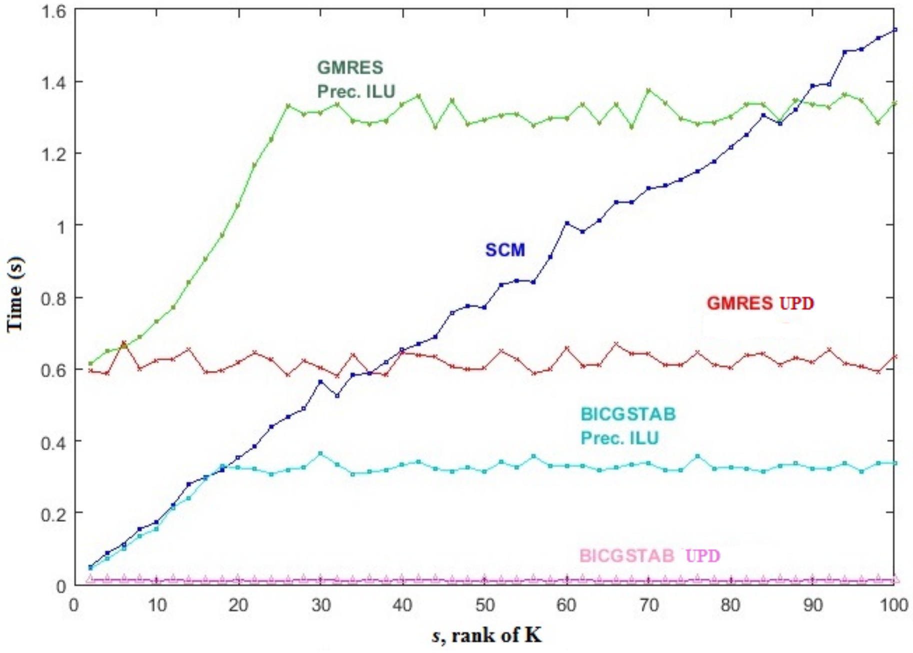


FIGURE 3.6: CPU solution time for the first example with the different methods tested for different values of the rank of the skew-symmetric part of A, s.

For all the values of the rank s it can be observed that using BiCGSTAB preconditioned with the updated preconditioned method performs the best. In the case of full GMRES, it starts to be competitive compared with SCM for values of s greater than 40. Note that the solution time of the SCM increases linearly with the rank of the skew-symmetric part, while its remains almost constant for the other methods.

The sequence of problems was also solved by considering the augmented system. Figure 3.7 shows the results for both, the full GMRES and BiCGSTAB methods, preconditioned with the point ILU and the block ILU for different ranks. It can be observed that the total solution time is similar for both preconditioners, and that the full GMRES converges slightly faster than BiCGSTAB.

Every preconditioner used in this example has a density around twice the number on nonzero elements of the initial coefficient matrix.

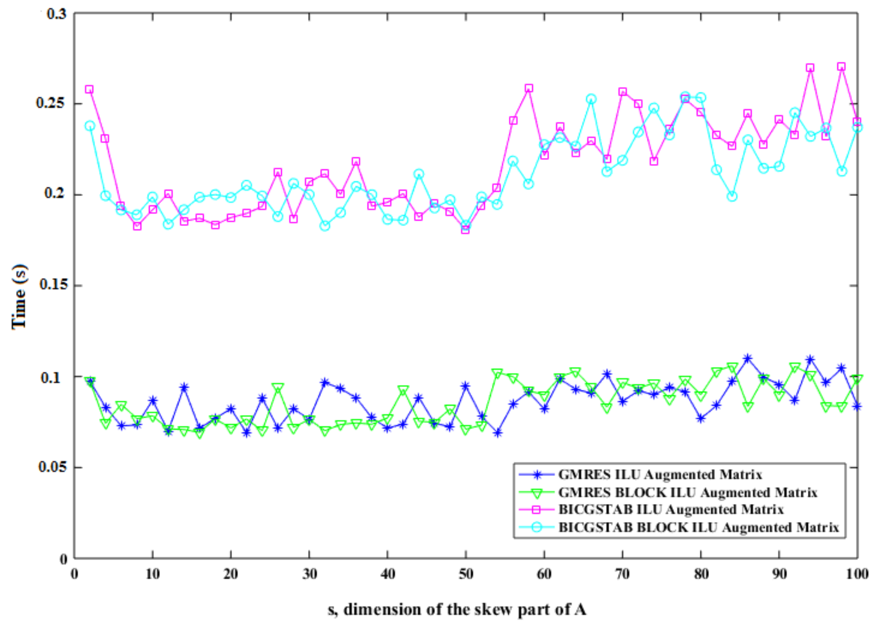


FIGURE 3.7: CPU solution time for the first example with different methods for the augmented systems and different values of the rank of the skew-symmetric part of A, s.

In the next example we modify the previous one in order to obtain a class of problems for which the skew-symmetric part of the coefficient matrix is approximated by a low-rank matrix, that is, $A = H + FCF^T + E$ with $E \neq 0$ and $\| E \| \ll 1$. The problem is defined with the following matrices,

$$A = \begin{bmatrix} \Psi & & \\ & \Gamma & \\ & & \Omega \end{bmatrix}, \; FCF^T = \begin{bmatrix} O & & \\ & O & \\ & & \frac{1}{2}(\Omega - \Omega^T) \end{bmatrix}, \; E = \begin{bmatrix} O & & \\ & \frac{1}{2}(\Gamma - \Gamma^T) & \\ & & O \end{bmatrix}$$

where Ψ is of size $n/2$ from the discretization of the 2D Poisson operator, $\Gamma = \text{tridiag}(-\gamma, -4, \gamma)$ and $\Omega = \text{tridiag}(-\omega, -4, \omega)$ are tridiagonal matrices of dimension $n/2 - s$ and $s \ll n$, respectively. We consider $n = 250000$, $\gamma = 0.01$, $\omega = 10$ and s an even number with values from 10 to 50 representing the rank of the matrix FCF^T. For these matrices the error matrix has $2-$norm equal to 0.02. We note that the norm of the skew symmetric part of these matrices are in the interval $[19.1899, 19.6824]$. Under these conditions the skew-symmetric part of A has rank equal to $n^2/2$ and it is approximated by a matrix of rank s. The matrix A is indefinite with eigenvalues lying in the intervals $(0, 8]$ and $[-4 - 20i, -4 + 20i]$, which is justified by the Gershgorin circle theorem, see [89]. Figure 3.8 shows the eigenvalue distribution for a matrix generated with $n = 50$ and $s = 20$.

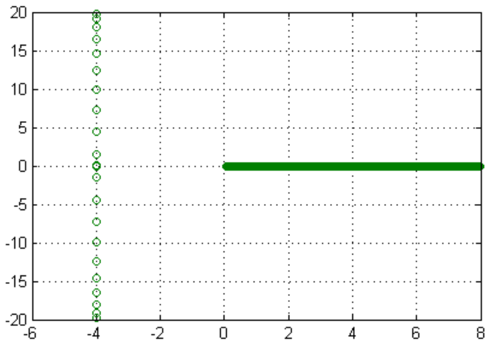


FIGURE 3.8: Eigenvalues for the matrix with n=50 and s=20

In Tables 3.2 and 3.3, respectively, we present the number of iterations (Iter) and time in seconds (Time) needed to solve the system $Ax = b$ with b a random vector. The restarted GMRES(100) and BiCGSTAB preconditioned with the SCM as presented in [19] were used. We also considered the restarted GMRES(100) and BiCGSTAB methods preconditioned with an ILU factorization computed for H with drop tolerance 10^{-2}, and with the updated preconditioner. Note that in this case, the SCM will not be used as a complete method, since when it was tested, it was not able to solve the problem to the tolerance 10^{-8}, in all cases it reached an error of the order of 10^{-4}, and regarding CPU time, over the reported for the GMRES preconditioned with SCM. The augmented system, referred to AS in the tables, was also solved with the GMRES(100) and BiCGSTAB, preconditioned with a point ILU with drop tolerance 10^{-2} and a block ILU (BILU). It is observed that the updated preconditioner and AS techniques show similar performance with respect number of iterations and CPU time. Moreover, both converge faster than the other ones. The density of all the preconditioners were in the interval $[1.56, 1.92]$.

As for the first class of examples, similar conclusions hold for the second one. There is a big improvement on the overall computational cost when solving the problems with the updated preconditioner technique, mainly for the BiCGSTAB method. The total solution time spent by the SCM as preconditioner increases considerably with the rank of the approximation. The BiCGSTAB with the SCM as preconditioner did

	Iter				
s	10	20	30	40	50
GMRES(100) Prec. ILU	221	254	260	299	401
GMRES(100) UPD	196	196	196	197	197
GMRES(100) SCM	206	206	206	207	207
BiCGSTAB Prec. ILU	272	936	1732	†	†
BiCGSTAB UPD	108	107	102	103	104
BiCGSTAB SCM	†	†	†	†	†
GMRES(100) AS ILU	196	196	196	197	197
GMRES(100) AS BILU	196	196	196	197	197
BiCGSTAB AS ILU	108	107	104	104	104
BiCGSTAB AS BILU	108	107	104	104	104

TABLE 3.2: Number of iterations for the second problem with different values of s. A † means no convergence in 2000 iterations.

	Time (s)				
s	10	20	30	40	50
GMRES(100) Prec. ILU	31.5	36.9	35.9	47.2	59.9
GMRES(100) UPD	30.2	29.8	29.8	30.3	29.8
GMRES(100) SCM	33.4	35.9	35.7	40.3	41.7
BiCGSTAB Prec. ILU	12.7	38.7	69.3	†	†
BiCGSTAB UPD	6.7	6.7	6.4	6.3	6.5
BiCGSTAB SCM	†	†	†	†	†
GMRES(100) AS ILU	28.9	29.2	29.4	29.5	29.3
GMRES(100) AS BILU	28.9	29.4	28.9	30.1	30.3
BiCGSTAB AS ILU	6.9	6.9	6.3	6.3	6.4
BiCGSTAB AS BILU	6.8	6.8	6.2	6.3	6.3

TABLE 3.3: CPU time for the second problem with different values of s. A † means no convergence in 2000 iterations.

not converge for any case. It is also important to emphasize that applying the iterative method to the augmented linear system also gives good results, specially with the BiCGSTAB method.

3.5.2 The Bratu problem and Love's equation

The next example corresponds to the 2-dimensional Bratu problem. To review the problem, it consists of finding the solution $u(x, y)$ of the nonlinear boundary problem

$$-\Delta u - \lambda \exp(u) = 0 \text{ in } \Omega, \quad \text{with } u = 0 \text{ on } \partial\Omega \tag{3.19}$$

depending on the parameter λ, Δ is the Laplacian, Ω the unit square and $\partial\Omega$ its boundary. We discretize this problem using the five-point finite differences as in [37, 12], in a grid of 500×500 points. After this, we obtain a system with coefficient matrix of order $n = 2.5 \times 10^5$ with skew-symmetric part of exactly rank equal to 2. Table 3.4 shows the results for the tested methods. Incomplete LU factorization with threshold parameter equals to 5×10^{-1} were used, giving preconditioners with density around twice the number of nonzero elements of the original problem. The non-preconditioned BiCGSTAB and restarted GMRES(m) methods were also tested. Note that we solve the Bratu problem presented in [37].

Method	Time (s)	Iter
GMRES(100)	†	
BiCGSTAB	26.6	827
GMRES(100) Prec. ILU	45.1	123
GMRES(100) UPD	46.3	131
BiCGSTAB Prec. ILU	13.1	194
BiCGSTAB UPD	11.3	156
SCM	38.2	255
GMRES(100) AS ILU	‡	
GMRES(100) AS BILU	50.7	125
BiCGSTAB AS ILU	‡	
BiCGSTAB AS BILU	59.7	166

TABLE 3.4: CPU solution time and iterations for the Bratu problem. † means no convergence in 2000 iterations. † means that ILU was not possible to compute.

It can be observed that BiGSTAB preconditioning with our technique has the edge over the SCM method and also works better than the ILU preconditioner computed for H. Compared with the preconditioned GMRES(100), both preconditioners performed similarly. In the case of the augmented system, it was not possible to compute a point ILU factorization for threshold parameter in the range 10^{-8} to 10^{-1}. It was possible, however, solving the augmented system with the block ILU preconditioner, but performance was poor.

Regarding the Love's equation, described in Subsection 1.5.1,

$$f(y) + \frac{1}{\pi}\int_{-1}^{1} \frac{c}{(x-y)^2 + c^2} f(x)dx = g(y), \quad |y| \leq 1 \;\; 0 \leq c \in \mathbb{R}.$$

choosing, $g(y) = \sqrt{1+y}$ and $c = 0.1$, discretizing the last equation by a Nyström method based on the composite trapezoidal rule with equidistant nodes $x_k = y_k =$

$(k-1)/(n-1)$, $1 \leq k \leq n$. This gives a linear system of equations with a $n \times n$ matrix whose skew-symmetric part is of exactly rank $s = 4$. Since the coefficient matrix is dense, we considered a moderated value $n = 2049$. The results are presented in Table 3.5. It can be observed that the number of iterations needed to converge is similar for all the preconditioners, but the CPU time with SCM was the largest one. Incomplete LU factorization with threshold parameter equals to 10^{-1} were used, giving preconditioners with density around twice the number of nonzero elements of the original problem. We note that in both problems, Bratu and Love, the SCM was used as a complete method, as presented in [19].

Method	Time (s)	Iter
GMRES Prec. ILU	0.04	5
GMRES UPD	0.03	4
BiCGSTAB Prec. ILU	0.04	2.5
BiCGSTAB UPD	0.04	2.5
SCM	0.51	55
GMRES(100) AS ILU	‡	
GMRES(100) AS BILU	†	
BiCGSTAB AS ILU	‡	
BiCGSTAB AS BILU	†	

TABLE 3.5: CPU solution time and iterations for the Love's equation. † means no convergence in 2000 iterations. † means that ILU was not possible to compute.

3.5.3 Problems from the University of Florida sparse matrix collection

Table 3.6 shows the matrices used in this subsection. These matrices arise from different applications. In this table n and nnz indicate the size and number of nonzeros of the matrices, respectively. The rank of the matrix FCF^T that approximates the skew-symmetric part is indicated with s, the norm of the skes-symmetric part and the norm of the error matrix E is indicated in the last two columns, $\|K\|_2$ and $\|E\|_2$ respectively. It can be observed that the SCRA method gives good low-rank approximations to the skew-symmetric part of the matrices sudied. $\|E\|_2 = 0$ means that the skew-symmetric part has low rank. The full and restarted GMRES methods were used. In this problems, as suggested in [19], the SCM will be use as preconditioner for the GMRES and BiCGSTAB. Tables 3.7, 3.8 and 3.9 show the results of the experiments.

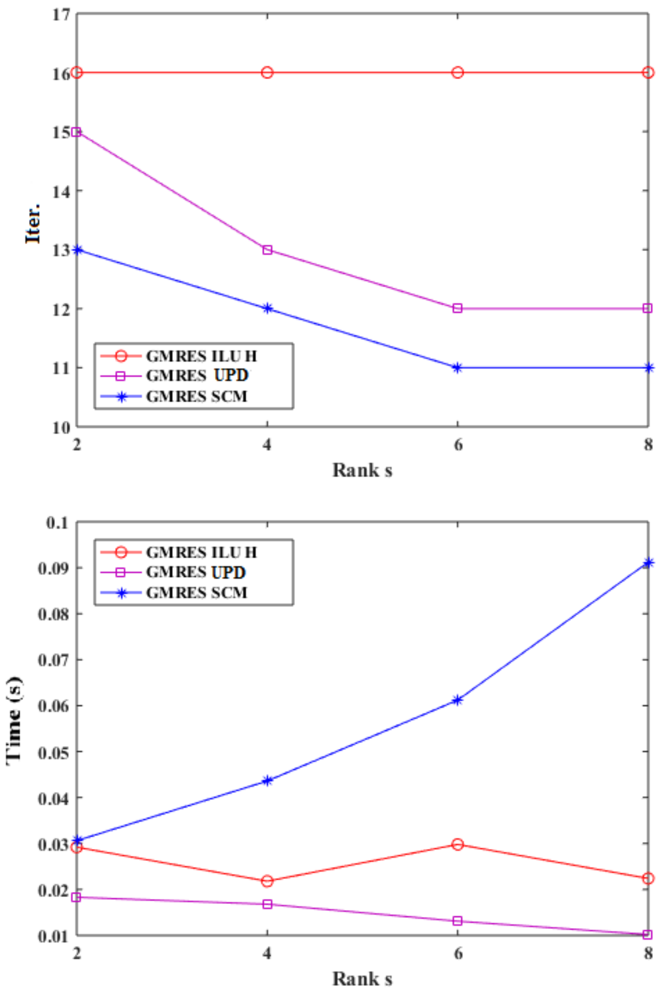


FIGURE 3.9: Iteration and CPU time with GMRES ILU H, GMRES UPD and GMRES SCM for the matrix ADDER_TRANS_01, $s = 2, 4, 6, 8$

Matrix name	Application	n	nnz	s	$\|K\|_2$	$\|E\|_2$
IPROB	Linear programming	3001	9000	4	66128	0
PESA	Directed weighted graph	11738	79566	2	16958	0
BIG	Directed weighted graph	13209	91465	2	19565	0
08BLOCKS	Combinatorial	300	592	16	189.9112	2×10^{-14}
ADDER_TRANS_01	Circuit simulation	1814	14579	8	3.5266	0.0314
ADD20	Circuit simulation	2395	13151	42	5×10^{-4}	1×10^{-4}
CIRCUIT_1	Circuit simulation	2624	35823	10	1.4212	0.0907
ASIC_100K	Circuit simulation	99340	940,621	6	138.8046	0.7091
HCIRCUIT	Circuit simulation	105676	513072	58	19.4064	0.0472
SCIRCUIT	Circuit simulation	170998	958936	126	14.8593	0.0026

TABLE 3.6: Set of tested matrices from the University of Florida sparse matrix collection

We can observe that for these matrices the UPD technique obtains the best results in terms of CPU time. The number of iterations is comparable to the SCM preconditioner, but this method spends more CPU time to obtain the solution because the preconditioner application is more expensive. Comparing with the incomplete LU factorization of H, when the skew-symmetric part K is significative, the updated preconditioner technique shows a better performance, specially when using the restarted GMRES. Specially significative is the improvement observed for the matrices in Table 3.9.

Finally, Figure 3.9 shows the results for the matrix ADDER_TRANS_01 for different values of the rank of the approximation s. Note that the updated preconditioner behaves better when the approximation of the skew-symmetric part is more precise, as it could be expected.

Regarding the augmented system, good results were observed for all the matrices tested, specially when the system was preconditioned with de block ILU (BILU).

3.6 Conclusions

We have presented a method for preconditioning non-symmetric matrices whose skew-symmetric can be well approximated by a low-rank matrix. The method can be viewed as an update of a preconditioner computed for the symmetric part of the system matrix. Some approximation properties of the preconditioner and the eigenvalue distribution of the preconditioned matrix have been presented. The method has been compared with others that appear in the literature for this kind of matrices.

	IPROB	PESA	BIG
	m= 10	m = 200	m = 200
	ρ/Iter/Time(s)	ρ/Iter/Time(s)	ρ/Iter/Time(s)
GMRES Prec. ILU	2.15/22/0.2	1.16/295/3.8	1.15/295/3.8
GMRES UPD	2.15/21/0.2	1.16/244/2.8	1.16/270/3.2
GMRES SCM	2.15/27/0.7	1.16/263/3.4	1.15/265/5.4
GMRES(m) Prec. ILU	2.15/30/0.2	1.16/698/5.5	1.15/1928/17.3
GMRES(m) UPD	2.15/30/0.1	1.16/398/3.4	1.16/832/7.5
GMRES(m) SCM	2.15/28/0.8	1.16/378/3.8	1.15/761/9.1
BiCGSTAB Prec. ILU	†	1.16/1353/2.7	†
BiCGSTAB UPD	†	1.16/1372/2.7	†
BiCGSTAB SCM	†	1.16/1256/2.9	†
GMRES AS ILU	†	1.16/263/2.8	1.15/275/3.3
GMRES AS BILU	1.36/23/0.2	1.16/263/3.1	1.15/275/3.7
GMRES(m) AS ILU	†	1.16/433/3.6	1.15/988/9.1
GMRES(m) AS BILU	1.36/67/0.2	1.16/433/4.1	1.15/991/9.3
BiCGSTAB AS ILU	†	1.16/1320/2.7	†
BiCGSTAB AS BILU	†	1.16/1082/2.6	†

TABLE 3.7: Results for the matrices IPROB, PESA, BIG

From the numerical results conducted it has been observed that the proposed preconditioner is competive in terms of solution time and number of iterations spent.

	08BLOCKS ρ/Iter/Time(s)	ADDER_TRANS_01 ρ/Iter/Time(s)	ADD20 ρ/Iter/Time(s)	CIRCUIT_1 ρ/Iter/Time(s)
GMRES Prec. ILU	2.30/18/0.01	0.58/16/0.12	0.89/10/0.01	0.42/15/0.03
GMRES UPD	2.35/13/0.01	1.40/12/0.04	0.92/9/0.01	0.78/9/0.01
GMRES SCM	2.30/12/0.02	0.58/11/0.14	0.89/9/0.11	0.42/9/0.12
BiCGSTAB Prec. ILU	2.30/19.5/0.01	0.58/13.5/0.05	0.89/5.5/0.01	0.42/15/0.02
BiCGSTAB UPD	2.35/10/0.01	1.40/7.5/0.03	0.92/5/0.01	0.78/6.5/0.01
BiCGSTAB SCM	2.30/13.5/0.03	0.58/8.5/0.17	0.89/5/0.38	0.42/7.5/0.18
GMRES AS ILU	1.18/26/0.01	0.39/12/0.04	0.90/10/0.01	0.33/10/0.01
GMRES AS BILU	1.58/13/0.01	0.74/12/0.02	0.91/9/0.01	0.60/8/0.01
BiCGSTAB AS ILU	1.18/10/0.01	0.39/10/0.04	0.90/5.5/0.01	0.33/7/0.01
BiCGSTAB AS BILU	1.18/10/0.01	0.74/7.5/0.02	0.91/5/0.01	0.60/5/0.01

TABLE 3.8: Results for the matrices 08BLOCKS, ADDER_TRANS_01, ADD20 and CIRCUIT_1

	ASIC_100K m = 20 ρ/Iter/Time(s)	HCIRCUIT m = 50 ρ/Iter/Time(s)	SCIRCUIT m=200 ρ/Iter/Time(s)
GMRES(m) Prec. ILU	0.87/120/2.1	0.86/368/8.4	1.09/1171/148.2
GMRES(m) UPD	0.87/38/0.8	0.89/80/1.6	1.12/568/61.4
GMRES(m) SCM	0.87/39/10.1	0.86/91/21.3	1.09/705/149.7
BiCGSTAB Prec. ILU	0.87/42.5/1.1	0.86/919.5/11.9	†
BiCGSTAB UPD	0.87/44.5/1.1	0.89/86.5/1.2	1.12/854/30.7
BiCGSTAB SCM	0.87/46.5/2.7	0.86/133.5/20.2	†
GMRES(m) AS ILU	0.48/31/0.9	0.83/143/3.0	1.09/561/70.5
GMRES(m) AS BILU	0.81/42/1.1	0.86/83/1.6	1.11/560/69.5
BiCGSTAB AS ILU	0.48/28.5/1.0	0.83/388.5/5.1	1.09/743/27.5
BiCGSTAB AS BILU	0.81/82/2.3	0.86/97.5/1.3	1.11/807.5/27.8

TABLE 3.9: Results for the matrices ASIC_100K, HCIRCUIT and SCIRCUIT

Chapter 4

Updating Preconditioners for Modified Least Squares Problems

In this chapter we analyze how to update incomplete Cholesky preconditioners to solve least squares problems using iterative methods when the set of linear relations is updated with some new information, a new variable is added or, contrarily, some information or variable is removed from the set. The proposed method computes a low-rank update of the preconditioner using a bordering method which is inexpensive compared with the cost of computing a new preconditioner. Moreover the numerical experiments presented show that this strategy gives, in many cases, a better preconditioner than other choices, including the computation of a new preconditioner from scratch or reusing an existing one. The main results of this chapter has been published in [60].

4.1 Introduction

Iterative methods are used for solving large and sparse linear least squares (LS) problems because they often require much less storage than their direct counterparts. One of the most used iterative methods for LS problems is CGLS [20]. CGLS is equivalent to applying the Conjugate Gradient method (CG) to the normal equations. To improve the convergence of the iterative method very often a preconditioner is needed.

Among other choices like Incomplete QR factorizations preconditioners, we will focus on Incomplete Cholesky (IC) preconditioners. These preconditioners have been successfully employed in different applications, see [72, 73], and allow for the computation of robust preconditioners for full rank overdetermined least squares problems [17, 23].

The problem of updating a preconditioner arise in some applications from statistics and optimization, where it is necessary to solve a sequence of modified least squares problems. An example can be found in [27], where an efficient and stable method for adding and deleting equations to a regression model is required. In signal processing applications near real-time solutions are required. Thus, methods that allow to modify LS problems with few operations and little storage requirements are needed, see [1]. The same problem is present if some information is added to or deleted from the data set. On some occasions it may be convenient to add or to remove some variables. Such situations are usually referred to as updating or downdating least squares problems. Chapter 3 of the reference text [20] is devoted to analyzing how to deal with these modifications when the least squares problem is solved by a direct method, including full and rank revealing QR decomposition, Cholesky factorization and singular value decomposition. More recently other algorithms to update Cholesky factorizations have been proposed, see [32, 33, 34]. More efforts seem to be addressed to updating the QR factorization, see [2, 46, 64].

In this chapter we present a method to modify an existing incomplete factorization with low computational cost. We note that when some columns are removed from an overdetermined system, obtaining a preconditioner for the modified LS problem can be done without additional cost by taking a block from the existing one. A similar situation occurs when some columns are added to an overdetermined system, or when new relations are added to an underdetermined one. In both cases the old preconditioner is the top left block of the new one. Thus, it is a preconditioner completion problem. The final result is equivalent to computing a new preconditioner from scratch.

The cases in which we are interested correspond to LS modified problems whose normal equations have a coefficient matrix that does not change in size but their entries do. These problems can be formulated as a low-rank update of the original normal equations and we propose updating the preconditioner following the the general framework described in Chapter 2. The goal is computing the update with smaller cost than obtaining a new preconditioner from scratch, but with comparable performance.

The chapter is organized as follows. In Section 4.2 we describe the preconditioning updating technique for this problem. In Section 4.3 we consider adding or deleting equations to overdetermined least squares problems. The opposite case, that is when the system is underdetermined, is analyzed in Section 4.4. We will see that there is a duality between both groups of problems. In Section 4.5 we present the results of the numerical experiments that show that the proposed strategy is effective.

4.2 Preconditioner update computation and application

Suppose that the least squares solution of the overdetermined linear system

$$Ax = b, \tag{4.1}$$

where A is a large and sparse $m \times n$ matrix, $m > n$, has been computed using a preconditioned iterative method. We assume that A has full rank, n, that is, its columns are linearly independent. As it is well known, the LS solution is given by the vector x that minimizes $\| b - Ax \|_2$, and can be obtained by solving the normal equations corresponding to (4.1) given by

$$A^T Ax = A^T b. \tag{4.2}$$

We are interested in computing the least squares solution of a new linear system obtained after the original system has been modified by adding or removing k equations. As it is shown in the next section, the normal equations for the modified linear system can be written in these cases as

$$(A^T A \pm B^T B)x = c, \tag{4.3}$$

where B is a $k \times n$ matrix.

Observe that the solution of (4.3) can be obtained from the solution of the equivalent linear system

$$\begin{bmatrix} A^T A & B^T \\ B & \mp I \end{bmatrix} \begin{bmatrix} x \\ \pm Bx \end{bmatrix} = \begin{bmatrix} c \\ 0 \end{bmatrix}. \tag{4.4}$$

Note that the augmented matrix in the last system is a particular case of the matrix in (2.4), defined in Chapter 2, with $A_{11} = A^T A$, $A_{12} = B^T$, $A_{21} = B$, $A_{22} = \mp I$.

One has the following relations between the linear operators in (4.3) and (4.4),

$$A^T A \pm B^T B = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} A^T A & B^T \\ B & \mp I \end{bmatrix} \begin{bmatrix} I \\ \pm B \end{bmatrix} \tag{4.5}$$

and their inverses

$$(A^T A \pm B^T B)^{-1} = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} A^T A & B^T \\ B & \mp I \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix}. \tag{4.6}$$

The preconditioner update technique consists in computing an incomplete factorization for the augmented matrix in (4.4) that is used to approximate the inverse linear operator in (4.6) by direct preconditioning, i.e., solving the corresponding upper and lower triangular systems. Therefore we avoid the computation of a new preconditioner for the updated matrix $A^T A \pm B^T B$ from scratch.

To be precise, let $A^T A \approx R^T R$ be an IC factorization of $A^T A$, where R is an upper triangular matrix. Then one gets a block LDL^T (almost Cholesky) factorization of the augmented matrix in (4.5) given by

$$\begin{bmatrix} A^T A & B^T \\ B & \mp I \end{bmatrix} = \begin{bmatrix} R^T & 0 \\ R_{12}^T & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \mp S \end{bmatrix} \begin{bmatrix} R & R_{12} \\ 0 & I \end{bmatrix}, \tag{4.7}$$

where $R_{12} = R^{-T} B^T$ is a $n \times k$ matrix and $S = I \pm R_{12}^T R_{12}$ is a $k \times k$ matrix. To maintain sparsity in these factors some dropping strategy can be used when computing R_{12} and an incomplete factorization of the Schur complement $S \approx R_S^T R_S$ as well, but if k is small enough this block can be factorized exactly. The computation of the preconditioner is done following the steps presented in Algorithm 8 in Chapter 2. Note that although the (approximate) inverse operator in the form of (4.6) is symmetric and positive definite, it is not stored nor can it be applied in factorized form. Therefore, only left or right preconditioning can be used when applying the conjugate gradient method to the normal equations (or the mathematically equivalent CGLS method). The preconditioning strategy proposed computes the preconditioned residual by applying Equation (4.6) with an incomplete factorization of the augmented matrix. That is, the preconditioned residual s is given by

$$s = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} A^T A & B^T \\ B & \mp I \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix} r,$$

and it is computed from the solution of

$$\begin{bmatrix} R^T & 0 \\ R_{12}^T & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \mp S \end{bmatrix} \begin{bmatrix} R & R_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} s \\ s' \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}. \tag{4.8}$$

The preconditioning is done in three steps as Algorithm 12 shows. Observe that this is a particular case of the Algorithm 9 presented in Chapter 2.

Algorithm 12 Preconditioner update application

```
**Input:** Matrices $R$, $R_{12}$, $R_S$ and residual vector $r$.
**Output:** Preconditioned vector $s$
1. Solve the linear system $R^T\tilde{r} = r$.
2. Update $\tilde{r} \leftarrow \tilde{r} \mp R_{12}(R_S^T R_S)^{-1} R_{12}^T \tilde{r}$.
3. Solve the linear system $Rs = \tilde{r}$.
```

Step 2 in Algorithm 12 represents the extra cost in the application of the preconditioner with respect to the case of non-updating an existing one. If R_{12} and R_S are kept sparse and the number of added or removed equations is small compared with the problem size, this overhead is small and can be amortized even for moderate reductions on the number of iterations, see [24]. The method in the rest of this chapter will be referenced as UPD, it essentially remains the same method presented in Chapter 2.

4.3 Updating preconditioners: Overdetermined case

In this section we consider all possible modifications of the overdetermined linear system (4.1). We analyze and propose strategies to get a preconditioner for the new normal equations when adding or/and removing equations, as well as when adding or/and removing some unknowns. As it is shown below some of theses cases lead to trivial computation of the preconditioner.

4.3.1 Adding equations to an overdetermined system

It may happen that some new relations among the unknowns are considered. If these relations are given as the system of k linear equations

$$Bx = c,$$

then we have the $m+k$ system of linear equations

$$\begin{bmatrix} A \\ B \end{bmatrix} x = \begin{bmatrix} b \\ c \end{bmatrix}.$$

If A has full rank, the new coefficient matrix $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ has also full rank, and the corresponding normal equations are

$$(A^TA + B^TB)x = A^Tb + B^Tc. \tag{4.9}$$

That is, the new normal equations are the result of a low-rank update of the initial ones. If we put $f = A^Tb + B^Tc$, the preconditioner update technique proposed in Section 4.2 can be applied to the augmented linear system

$$\begin{bmatrix} A^TA & B^T \\ B & -I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}.$$

4.3.2 Removing equations from an overdetermined system

This is just the opposite case. Suppose that instead of adding new information, some linear equations are removed from the initial linear system $Ax = b$. After a suitable row permutation, the new system can be written as

$$\begin{bmatrix} A_1 \\ B \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{4.10}$$

where $A_1 \in \mathbb{R}^{(m-k)\times n}$, $m-k > n$ and $B \in \mathbb{R}^{k\times n}$, and it is assumed that $\operatorname{rank}\left[\begin{smallmatrix} A_1 \\ B \end{smallmatrix}\right] = n$. Assume the information corresponding to the bottom block must be removed. The normal equations corresponding to (4.10) are

$$(A_1^TA_1 + B^TB)x = A_1^Tb_1 + B^Tb_2. \tag{4.11}$$

Observe that the row permutation is irrelevant when forming the normal equations. In fact if M is a matrix and P is a permutation matrix, $(PM)^T(PM) = M^T(P^TP)M = M^TM$.

After deleting the bottom block, one gets the linear system $A_1 x = b_1$, whose normal equations, $A_1^T A_1 x = A_1^T b_1$, can be related to (4.11) by

$$(A^T A - B^T B)x = A_1^T b_1.$$

This system is again the result of a rank k modification of the initial normal equations, and it has the same solution as component x in the solution of the augmented linear system

$$\begin{bmatrix} A^T A & B^T \\ B & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} A_1^T b_1 \\ 0 \end{bmatrix},$$

which allows for the application of the preconditioner update strategy described in Section 4.2, provided that A_1 has full rank. Otherwise, the singularity of $A_1^T A_1$ may produce poor preconditioners or even a breakdown during the computation of the preconditioner. Since the new coefficient matrix has less rows than the original one, it may happen that the remaining set of equations has rank less than n. In this case the square matrix $\left[\begin{smallmatrix} A^T A & B^T \\ B & I \end{smallmatrix}\right]$ is singular. To prove it, let $\operatorname{rank} A_1 = r < n$. Then, $\operatorname{rank} A_1^T A_1 = r < n$, and let $B \in \mathbb{R}^{k \times n}$ of rank k. Let us do a symmetric permutation, $\left[\begin{smallmatrix} O & I \\ I & O \end{smallmatrix}\right]$, to the matrix so that the permuted matrix is $\left[\begin{smallmatrix} I & B \\ B^T & A^T A \end{smallmatrix}\right]$. After eliminating the left bottom block by Gaussian elimination obtaining $\left[\begin{smallmatrix} I & B \\ 0 & A_1^T A_1 \end{smallmatrix}\right]$, which has rank $k + r < k + n$. Of course, computing a new preconditioner from scratch in this case can be difficult for the same reasons. This is illustrated with an example with the matrix ASH219 in the numerical experiments section, see figures 4.4, 4.5 and 4.6.

4.3.3 Adding and removing equations from an overdetermined system

Now suppose that both things occur simultaneously, that is, some equations are added and some others are deleted. To fix the notation, starting with the linear system $Ax = b$ written as

$$\begin{bmatrix} A_1 \\ B \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{4.12}$$

one wants to solve the linear system obtained after removing the bottom equations $Bx = b_2$, and then adding some new equations $Cx = c$, such that the new linear system is

$$\begin{bmatrix} A_1 \\ C \end{bmatrix} x = \begin{bmatrix} b_1 \\ c \end{bmatrix}. \tag{4.13}$$

The normal equations for systems (4.12) and (4.13) are

$$(A_1^T A_1 + B^T B)x = A_1^T b_1 + B^T b_2 \tag{4.14}$$

and

$$(A_1^T A_1 + C^T C)x = A_1^T b_1 + C^T c, \tag{4.15}$$

respectively. If we consider the augmented system

$$\begin{bmatrix} A^T A & C^T & B^T \\ C & -I & 0 \\ B & 0 & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} A_1^T b_1 + C^T c \\ 0 \\ 0 \end{bmatrix} \tag{4.16}$$

we obtain

$$\begin{gathered} A^T A x + C^T y + B^T z = A_1^T b_1 + C^T c \\ Cx - y = 0 \\ Bx + z = 0. \end{gathered}$$

Hence, $y = Cx$ and $z = -Bx$ and substituting in the first equation we obtain

$$(A^T A + C^T C - B^T B)x = (A_1^T A_1 + C^T C)x = A_1^T b_1 + C^T c.$$

Therefore, problem (4.15) is a low-rank update of the initial problem (4.14) and the proposed strategy can be used provided that $\left[\begin{smallmatrix} A_1 \\ C \end{smallmatrix}\right]$ has full rank. Note that the new coefficient matrix can have full rank even when A_1 has not, depending on C. Observe also that the coefficient matrix in (4.16) is nonsingular if and only if the matrix $\left[\begin{smallmatrix} A_1 \\ C \end{smallmatrix}\right]$ has full rank since the Schur complement of the $(1, 1)$ block is $A_1^T A_1 + C^T C$.

4.3.4 Adding columns to an overdetermined system

Assume now that a set of columns are added to the system (4.1). The natural choice is to put them at the end of the matrix as long as the solution can be easily reordered. An example can be if we decided to increase the degree of a fitting polynomial to a set of data, or one considers that some previously discarded variable is relevant to the phenomenon studied. Then the augmented system is

$$\begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = b,$$

and the corresponding normal equations are

$$\begin{bmatrix} A^T A & A^T B \\ B^T A & B^T B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} A^T b \\ A^T b \end{bmatrix}. \tag{4.17}$$

Updating an already computed preconditioner for $A^T A$ is straightforward, provided the new coefficient matrix in Equation (4.17) has still full rank. Observe that this matrix has full rank if and only if the coefficient matrix in the last linear system is positive definite. Then to update the preconditioner is as easy as to apply the standard method to complete the preconditioner by bordering. Thus, we do not consider this case in this thesis since it is equivalent to compute a new preconditioner from scratch but taking into account that a subblock is already at disposal, and therefore, with a lower cost.

4.3.5 Removing columns from an overdetermined system

Now we analyze the case of removing some columns of the coefficient matrix in (4.1). A suitable column permutation allows to move these columns to the right of the matrix, so that we can write the system as

$$\begin{bmatrix} A & C \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = b,$$

C being the block of columns to be removed. The normal equations of the original system are

$$\begin{bmatrix} A^T A & A^T C \\ C^T A & C^T C \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} A^T b \\ C^T b \end{bmatrix}. \tag{4.18}$$

The $(1,1)$ block of the coefficient matrix in the previous Equation (4.18) is the coefficient matrix of the normal equations we want to solve and the corresponding part of the incomplete Cholesky preconditioner computed for this matrix, is the preconditioner needed. This case is trivial and it is not considered in this work.

4.3.6 Adding and removing columns from an overdetermined system

Suppose now that in the system

$$\begin{bmatrix} A & C \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = b,$$

the last columns, block C, are removed and a new set of unknowns y and their corresponding block B are added to obtain the system

$$\begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = b.$$

Of course the update can be computed only if the matrix $\begin{bmatrix} A & B \end{bmatrix}$ has full column rank and in this case the preconditioner for the new system can be computed in two steps. First one must select the $(1,1)$ block in the IC preconditioner computed for $\begin{bmatrix} A^TA & A^TC \\ C^TA & C^TC \end{bmatrix}$ and then compute the part corresponding to the new block as proposed in subsection 4.3.4. As before, this case is not considered in this work.

4.4 Updating preconditioners: Underdetermined case

Consider now the LS problem

$$\min \|x\|_2 \qquad \text{subject to } Ax = b, \tag{4.19}$$

where $A \in \mathbb{R}^{m\times n}$, is a large and sparse full rank matrix with $m < n$. Problem (4.19) is solved using the second kind normal equations

$$AA^Tz = b, \qquad y = A^Tz. \tag{4.20}$$

Since $y = A^T(AA^T)^{-1}b$, y belongs to the row subspace of A which is orthogonal to the Kernel of A. Thus, y is the solution of (4.19).

As in the overdetermined case, it is assumed that an incomplete Cholesky factorization of the symmetric positive definite matrix AA^T has been computed. Then, new unknowns or columns are added to the linear system or some of them are deleted, or both. In the following, we will see that the preconditioner can be updated under the same conditions and with similar techniques as for the cases studied in Section 4.3.

4.4.1 Adding equations

Suppose that some new equations are added to the problem (4.19) obtaining a LS problem given by

$$\min \|x\|_2 \qquad \text{subject to} \begin{bmatrix} A \\ B \end{bmatrix} x = \begin{bmatrix} b \\ c \end{bmatrix} \tag{4.21}$$

where $B \in \mathbb{R}^{p \times n}$, with $m + p < n$. Clearly the coefficient matrix in (4.21) has full rank and the new normal equations of second kind are

$$\begin{bmatrix} AA^T & AB^T \\ BA^T & BB^T \end{bmatrix} z = \begin{bmatrix} b \\ c \end{bmatrix}. \tag{4.22}$$

Observe from (4.22) that we have a preconditioner completion problem similar to (4.17), and therefore the same strategy proposed in subsection 4.3.4 can be used and the same considerations hold.

4.4.2 Removing equations

To analyze this case we can consider, without lost of generality, that the last block of equations in

$$\begin{bmatrix} A \\ C \end{bmatrix} x = \begin{bmatrix} b \\ d \end{bmatrix} \tag{4.23}$$

is removed. Therefore, the normal equations for (4.23) are

$$\begin{bmatrix} AA^T & AC^T \\ CA^T & CC^T \end{bmatrix} z = \begin{bmatrix} b \\ d \end{bmatrix}. \tag{4.24}$$

Observe that the coefficient matrix in (4.24) is similar to the one in (4.18), hence, one must simply take the $(1, 1)$ block to obtain a preconditioner for solving the LS problem (4.20). Then, it is a trivial case.

4.4.3 Adding and removing equations

If we want to remove some equations and add some new ones, let us consider first that the equations to be removed have been permuted to the last positions. Then, the problem consists in removing equations $Cx = d$ in the Equation (4.23) and adding the new equations $Bx = c$ to get a system as the one presented in Equation (4.21).

The corresponding normal equations of the second kind are (4.24) and (4.22), respectively and thus, the new preconditioner is obtained by taking off the $(1, 1)$ block in (4.24) first and then completing the computation of the preconditioner.

4.4.4 Adding columns

Now we consider the problem of adding unknowns to (4.19), so that the new LS problem is

$$\min \|x\|_2 \qquad \text{subject to} \begin{bmatrix} A & B \end{bmatrix} x = b. \tag{4.25}$$

The normal equations of the second kind in this case are

$$(AA^T + BB^T)z = b, \tag{4.26}$$

that corresponds to a low-rank update similar to the one described in subsection 4.3.1 since the augmented linear system

$$\begin{bmatrix} AA^T & B^T \\ B & -I \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

provides the solution for (4.26). Therefore, the updating strategy proposed in Section 4.2 can be applied.

4.4.5 Removing columns

Assume that the linear system $Ax = b$ is splitted as

$$\begin{bmatrix} A_1 & B \end{bmatrix} x = b,$$

where B represents the block of columns to be removed. The corresponding normal equations of the second kind are

$$(A_1A_1^T + BB^T)z = b.$$

To solve the new normal equations $A_1A_1^Tz = b$, one can consider the augmented system

$$\begin{bmatrix} AA^T & B \\ B^T & I \end{bmatrix} \begin{bmatrix} z \\ w \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

This is similar to the situation in subsection 4.3.2. Then, the proposed strategy to update the preconditioner can be applied.

4.4.6 Adding and removing columns

The last problem that we study in this section corresponds to the case of removing the last set of columns in the underdetermined linear system $Ax = b$ given by

$$\begin{bmatrix} A_1 & B \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = b \tag{4.27}$$

and adding a new block C to get a new underdetermined problem given by

$$\begin{bmatrix} A_1 & C \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = b. \tag{4.28}$$

The normal equations of the second kind for (4.28) are

$$(A_1A_1^T + CC^T)z = b,$$

that can be written as

$$(AA^T - BB^T + CC^T)z = b.$$

The solution of these equations, z, can be obtained from the solution of the augmented system

$$\begin{bmatrix} AA^T & B & C \\ B^T & I & 0 \\ C^T & 0 & -I \end{bmatrix} \begin{bmatrix} z \\ v \\ w \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}.$$

Observe that the coefficient matrix of this system is similar to the one in (4.16). Then, the same comments and strategies used in subsection 4.3.3 apply to this case.

4.5 Numerical experiments

In this section we study the numerical performance of the preconditioner update method proposed (UPD). We present results obtained with matrices arising in different areas of scientific computing. The performance of the method is compared with other preconditioning strategies. The first one is reusing the initial preconditioner computed for the normal equations of the unmodified matrix. The second strategy corresponds to the computation of a new almost Cholesky preconditioner for the updated matrix from scratch. In addition, non-preconditioned iterations are also reported.

We present results for the modifications described in sections 4.3 and 4.4 that correspond to adding and removing equations or columns.

Matrix name	rows	cols	nnz	Application
PHOTOGRAMMETRY2	4472	936	37056	Computer graphics/vision problem
TESTBIG	17613	31223	61639	Linear programming problem
CAT_EARS_4_4	19020	44448	132888	Combinatorial problem
DELTAX	68600	21961	247424	High fillin with exact partial pivoting
FOME13	48568	97840	285046	Linear programming problem
LP_KEN_18	105127	154699	358171	Linear programming problem
FLOWER_8_4	55081	125361	375266	Combinatorial problem
FXM3_16	41340	85575	392252	Linear programming problem
LP_OSA_30	4350	104375	604488	Linear programming problem
MESH_DEFORM	234023	9393	853829	Image mesh deformation problem
WATSON_1	201155	386992	1055093	Linear programming problem
TS-PALKO	22002	47235	1076903	Linear programming problem
LP_NUG30	52260	379350	1567800	Linear programming problem
LARGEREGFILE	2111154	801374	4944201	Circuit simulation problem
SLS	1748122	62729	6804304	Statistics
TP-6	142752	1014301	11537419	Linear programming problem

TABLE 4.1: Set of test matrices

The tested matrices are shown in Table 4.1. All the matrices can be downloaded from the University of Florida Sparse Matrix Collection [35]. For each matrix we provide its number of rows and columns, the number of its nonzero entries, nnz, and the application field. The matrices with more rows than columns were used to obtain the numerical results corresponding to the overdetermined case, while the rest, mainly matrices arising from linear programming problems, were used for the undetermined one.

The preconditioned CGLS [20] or CGNR [73] and the preconditioned CGNE [73] for the overdetermined problems, were used for a relative initial residual norm decrease of 10^{-8}, allowing a maximum number of $3,000$ iterations. The right hand side vector was computed as a random vector. The initial approximation to the solution x was the vector of all zeros. The experiments where done with MATLAB version 2016a running on an Intel 5 CPU with 8 Gb of RAM in a Windows operating system. We used MATLAB's function $\mathbf{ilu}()$ to compute the incomplete factorizations since, for some matrices, the computation of a Cholesky factorization with the MATLAB's function $\mathbf{ichol}()$ stopped with a breakdown. Moreover, we found that permuting the coefficient matrix to block triangular form before computing the normal equations improved the quality of the preconditioner. Thus, all the matrices were permuted using the MATLAB's function $\mathbf{dmperm}()$ that obtains the Dulmage-Mendelsohn decomposition [71]. Symmetric diagonal scaling was applied to the matrices. The dropping parameter for managing the fill-in of the preconditioners was set to 0.1 except for the matrices DELTAX and MESH_DEFORM for which a value of 0.01 was used. We avoided fine tuning of the drop tolerance and with these values we computed very sparse preconditioners.

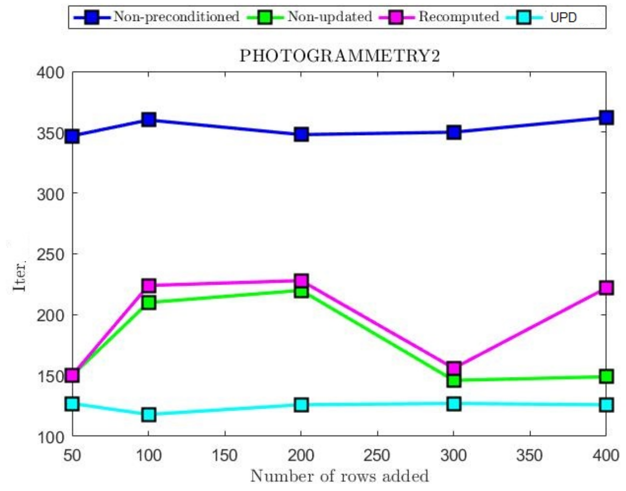
Tables 4.2 and 4.3 report the results for the cases of adding and removing equations or columns, depending of the problem. In these tables, k represents the rank of the update, i.e., the number of equations added or removed. This parameter is given in absolute number and also in percentage compared with the largest dimension of the matrix. We tested several values, but in the Tables we only report three results for each matrix that correspond to small, medium and large modifications up to a maximum of five percent. The relative density of the preconditioner with respect to the updated matrix is indicated in the column ρ. For simplicity the minimum and maximum density values observed for the preconditioners considered are shown. Normally, the minimum value corresponds to the non-updated preconditioner while the maximum was achieved for either, the recomputed or the UPD. The number of iterations and CPU solution time, measured in seconds, are indicated with Iter and Time, respectively. We recall that the application of the UPD and the recomputed preconditioners have, with respect to the two other strategies, an extra cost due to the computation of the UPD or the computation of the full preconditioner from scratch, respectively. Therefore, in the tables the value Time reports the total CPU time corresponding to the preconditioner computation and the iterative solution spent by these two strategies. As recommended in MATLAB's documentation, CPU times reported are the mean value of 10 successive runs of the experiment performed after 3 initial runs that were discarded. The maximum standard deviation observed relative to the mean value was 3 percent, and frequently less than 1 percent.

We start analyzing the results for the case of adding equations or columns that are shown in Table 4.2. The equations added were obtained by selecting at random k rows of the original matrix, and ordering in reverse order their column entries to avoid duplicated rows. In the case of adding columns, the modification was obtained similarly but with the rows of the matrix. The preconditioner density is very small for most of the preconditioners and always below one. It is important to note that in our algorithm, to compute an update with moderate fill-in, element dropping was applied in three different steps. First, a sparsification of the new block of rows (equations) added to the matrix was done before computing the block column R_{12} in Equation (4.7). Then, the computation of the block R_{12} itself was done incompletely by dropping small entries. Finally, an incomplete factorization was computed for the Schur-complement block S, see Equation (4.7). The respective tolerances were 1.0, 1.0 and 0.1 for all the matrices. Although for small values of k exact factorization of the Schur complement S could be done, we avoided fine tuning and performed incomplete factorization with the same drop tolerance used for the normal equations. We note that, with this aggressive dropping the total solution time was reduced, because the application of the preconditioner is cheaper, and also in some cases the number of iterations needed to converge was reduced. We recall that adding fill-in is not directly correlated with fewer number of iterations and, actually an increment is possible as is reported for instance in [23] for incomplete Cholesky factorizations for LS problems, see also [15].

From the number of iterations we see that the UPD performed better than the non-updated one, and similar to the case of recomputing the preconditioner. Taking into account the overall time, our strategy performed similarly to the best of the other strategies in most of the cases, and it was the best in several cases. For example, Figure 4.1 shows the evolution of the number of iterations when the number of added rows increases, for the matrix PHOTOGRAMMETRY2. In this case the proposed strategy performed better than the others with almost constant number of iterations.

We present in Figures 4.2 and 4.3 the comparison of the different methods when equations are added for the problem SLS and FOME13, respectively.

Analyzing the results in Table 4.3 we observe that, if instead of adding equations



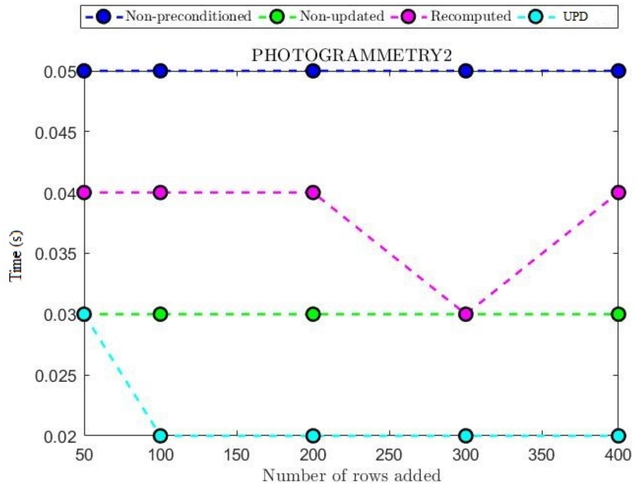
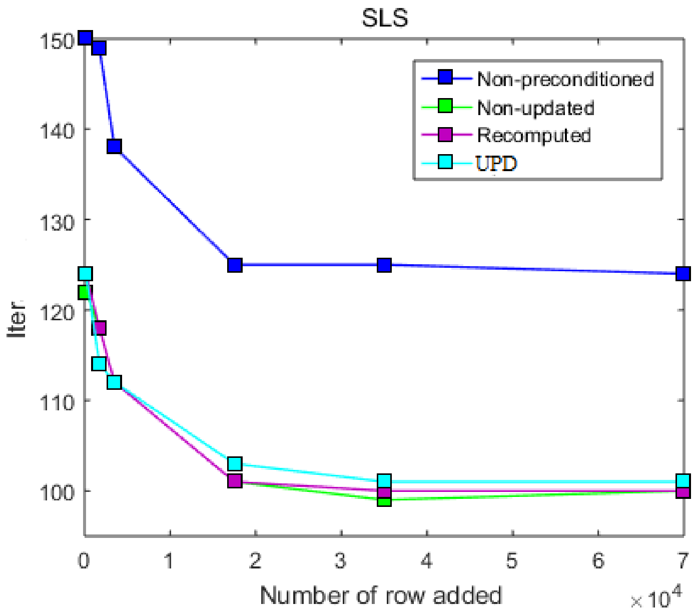


FIGURE 4.1: Effect of the number of equations added in the number of iterations and time for the matrix PHOTOGRAMMETRY2.



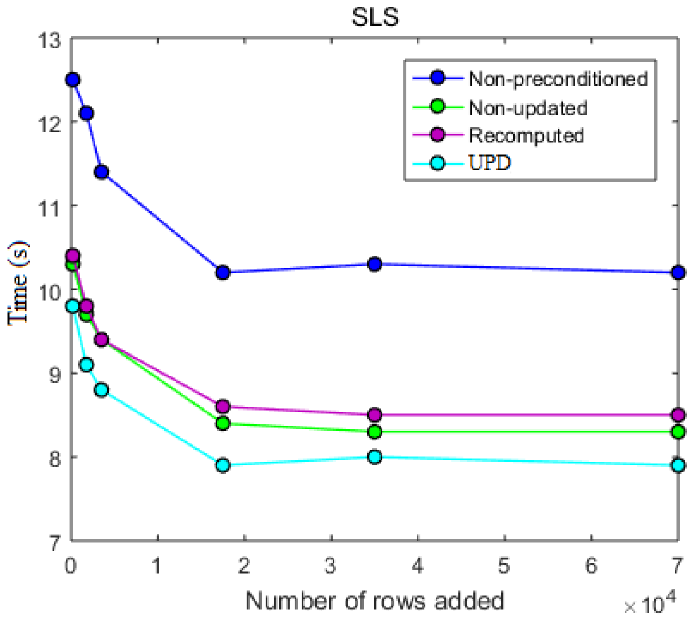


FIGURE 4.2: Effect of the number of equations added in the number of iterations and in the total time to get the solution for the matrix SLS.

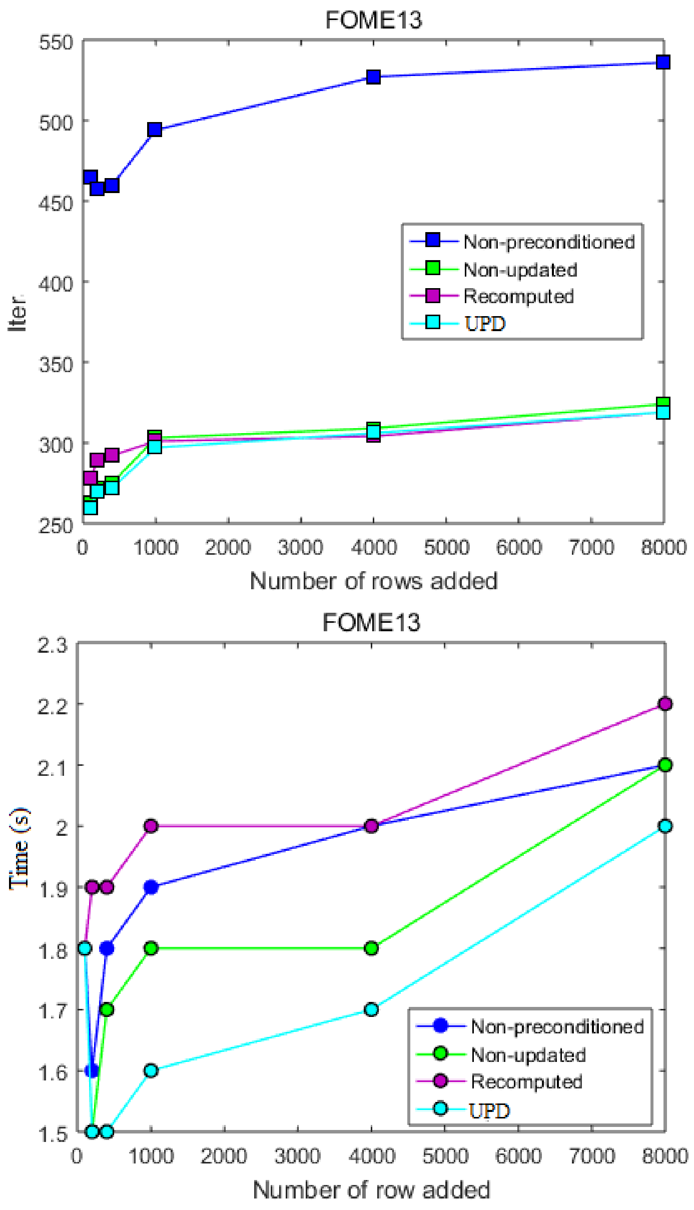


FIGURE 4.3: Effect of the number of equations added in the number of iterations and in the total time to get the solution for the matrix FOME13.

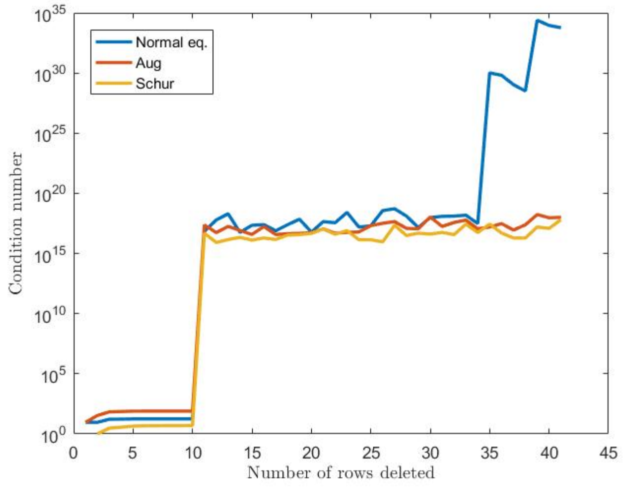


FIGURE 4.4: Condition numbers of the normal equations, equivalent bordered matrix and Schur complement matrix S for matrix ASH219 when removing rows from bottom.

(columns) the modification consists of removing a block of them, the situation changes in favor of the proposed algorithm. In this case, when the number of equations removed increases, sometimes the preconditioner can not be computed or it is very poor. Therefore, the application of the recomputed preconditioner can even lead to a divergence of the iterative solution method. Recall that, after removing equations, it is not warranted that the new matrix keeps its full rank. This can explain the big increment in the number of iterations needed to converge, and even the failure to converge in some cases. Under these conditions the proposed updating strategy performed nicely, and surprisingly it kept an almost constant performance independently of the number of equations or columns removed.

To illustrate the comments above we did an experiment with the matrix ASH219, also from the University of Florida Sparse Matrix Collection [35]. Its size is 219×85 and has full rank. Figure 4.4 illustrates the condition numbers of the normal equations, of the bordered matrix and the Schur complement block S, when successive rows are deleted from the end of the matrix. We observe that when a small number of rows are deleted all condition numbers remain low and have the same order up to some point where all them increase similarly. But eventually as more rows are deleted the condition number of the normal equations rises quickly while the other condition numbers remain almost constant. Figure 4.5 shows the decay evolution of

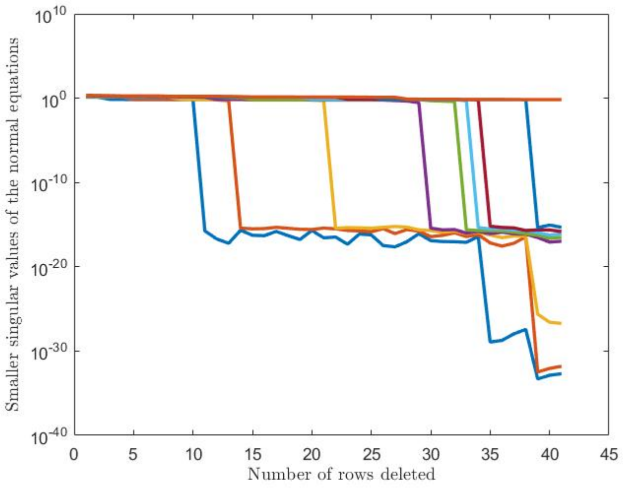


FIGURE 4.5: Decay evolution of the smallest 9 singular values of the normal equation matrix when removing rows from the bottom for matrix ASH219.

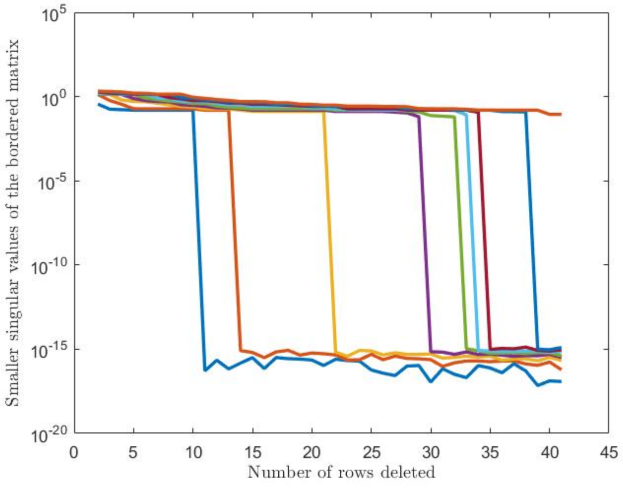


FIGURE 4.6: Decay evolution of the smallest 9 singular values of the augmented matrix when removing rows from the bottom for matrix ASH219.

the smallest singular values of the matrix when rows are deleted. It is observed that the increment of the condition number is acompanied by a progressive increment of the number of singular values that are clustered closer and closer to zero, while the decay in the singular values for the augmented system is less pronounced as Figure 4.6 shows. This may explain the degradation of the convergence degradation of the iterative method and why the updating technique gives better results than recomputing the preconditioner from scratch for some problems.

We note that as more rows (columns) are removed the matrix may loose its full rank, we observed this situation for example in the case of the matrix FXM3_16. CGLS then converges to the pseudo-inverse solution if the initial approximation x^0 is in the range of A^T, as in our choice of x^0 as the vector of all zeros [20, p. 291]. In practice this convergence can be very slow, or even can stagnate. Also it may be very difficult to compute the preconditioner due to breakdowns, as happen in some cases when recomputing it from scratch. Using our strategy the preconditioner was computed successfully in all cases and preconditioned CGLS converged quite fast.

Although for some matrices, for example TESTBIG and LP_OSA_30, the results are comparable in terms of time, we recall that the reduction of the number of iterations spent by the iterative method may have a bigger impact in the overall solution time when increasing the problem size, as the matrices SLS and FOME13 shows, see Figures 4.7 and 4.8.

Overall, we can conclude that the proposed algorithm is competitive and robust since it was able to successfully solve all the problems. The number of iterations and time spent was the best, or close to it, in the majority of cases. Another conclusion is that, in general, it is better to apply the UPD or recompute a new preconditioner from scratch instead of reusing the original one. In any case, these three strategies are better than non-preconditioned iterations. Computing a new preconditioner from scratch may have two drawbacks. The first one is an increment on the set-up time that usually only pays off in the case of adding equations when the size of the update is quite large. The second one is that when removing equations, the preconditioner computation may become unstable probably due to an increment of the condition number of the coefficient matrix of the normal equations.

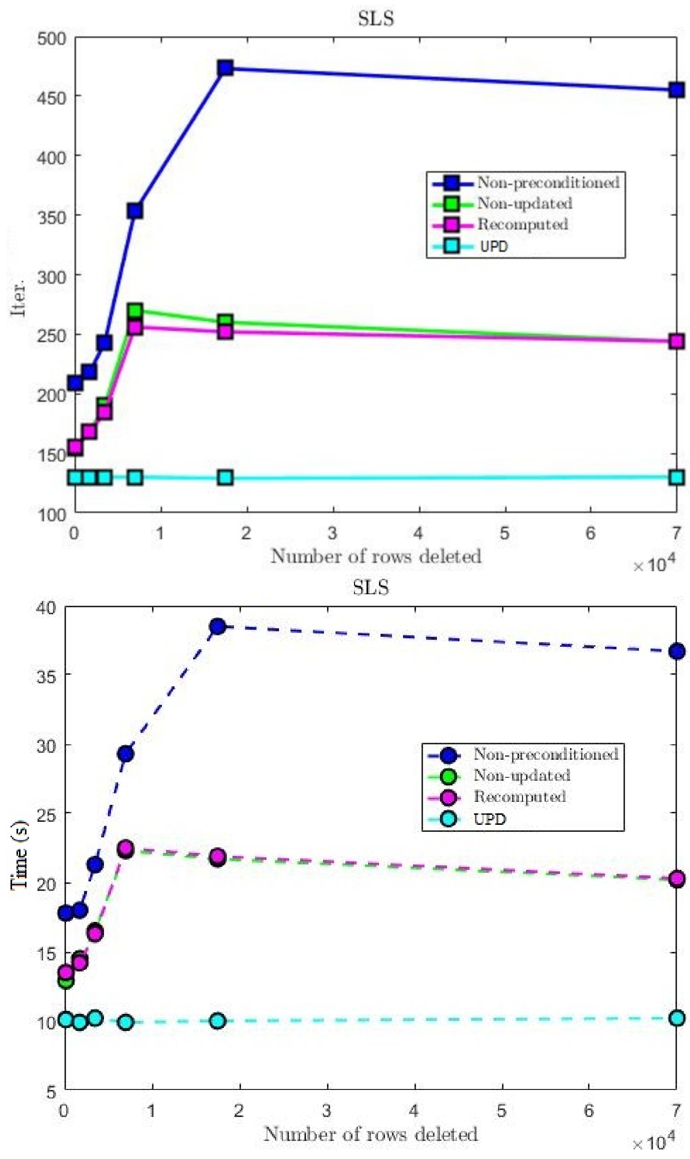
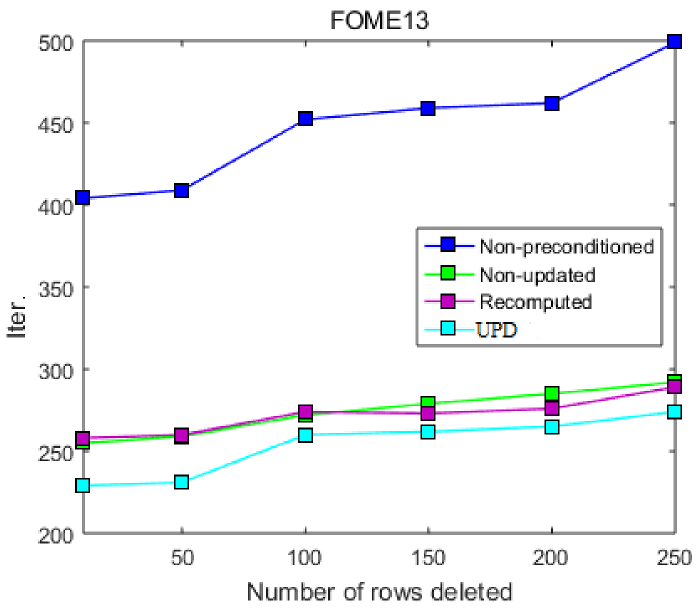


FIGURE 4.7: Effect of the number of equations deleted in the number of iterations and in the total time to get the solution for the matrix SLS.



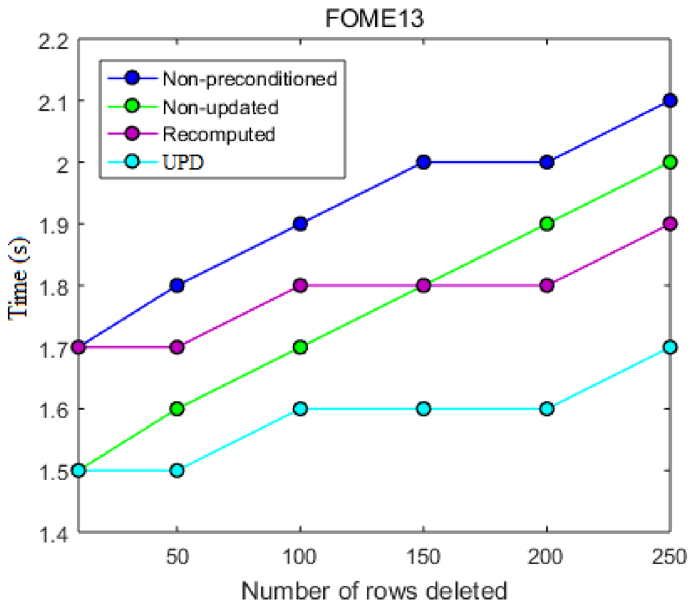


FIGURE 4.8: Effect of the number of equations deleted in the number of iterations and in the total time to get the solution for the matrix FOME13.

4.6 Conclusions

The main conclusion of this chapter is that adding equations, or removing them or both, as well as adding or removing or simply changing some variables is feasible when solving least squares problems with preconditioned iterative methods, provided that the resulting coefficient matrix has full rank.

In the most difficult cases, that is, when the coefficient matrix of the normal equations are modified by a low-rank matrix, we have introduced a technique, based on bordering, that allows to update the preconditioner in an inexpensive way. This technique has moderate memory and computational requirements, as demanded in [1]. Moreover, it is effective and robust as our numerical experiments show.

In Chapter 5 we combine the proposed strategy with other regularization techniques to compute the least squares solution of rank deficient linear systems.

Matrix	k	ρ	non-prec	non-updated	recomputed	UPD
			Iter/Time(s)	Iter/Time(s)	Iter/Time(s)*	Iter/Time(s)*
	300/0.96		86/0.1	72/0.1	66/0.2	70/0.1
TESTBIG	600/1.92	[0.54,0.57]	89/0.1	73/0.1	69/0.2	67/0.1
	1200/3.84		87/0.1	72/0.1	66/0.2	72/0.1
	400/0.90		151/0.2	83/0.3	82/0.4	82/0.2
CAT_EARS_4_4	1100/2.47	[0.70,0.77]	158/0.2	87/0.3	87/0.3	86/0.3
	2200/4.95		149/0.2	81/0.3	81/0.4	81/0.3
	300/0.44		768/1.4	919/3.3	920/3.8	873/3.0
DELTAX	1200/1.75	[0.95,0.98]	787/1.6	914/3.1	829/3.6	788/2.8
	3400/4.96		903/1.8	899/3.1	894/3.7	863/2.9
	200/0.20		457/1.5	272/1.5	289/1.9	270/1.4
FOME13	1000/1.02	[0.74,0.78]	494/1.6	303/1.6	301/2.0	297/1.6
	4000/4.08		527/1.8	309/1.8	304/2.0	306/1.7
	500/0.32		511/2.6	167/1.4	167/1.4	152/1.2
LP_KEN_18	1000/0.65	[0.75,0.78]	509/2.6	154/1.3	155/1.4	154/1.2
	5000/3.23		519/2.6	169/1.4	168/1.5	169/1.4
	500/0.40		177/0.6	104/0.7	103/1.2	103/0.7
FLOWER_8_4	1000/0.80	[0.77,0.80]	190/0.7	94/0.7	99/1.2	93/0.7
	6000/4.79		182/0.8	102/0.8	97/1.2	101/0.8
	100/0.12		1995/4.9	616/2.9	605/2.4	608/2.4
FXM3_16	1000/1.17	[0.34,0.38]	2870/7.1	754/2.8	712/2.8	714/2.8
	4000/4.68		†	781/3.1	802/3.1	794/3.1
	500/0.48		132/0.4	66/0.2	55/0.2	60/0.2
LP_OSA_30	1000/0.96	[0.01,0.02]	134/0.4	67/0.2	57/0.2	57/0.2
	5000/4.79		142/0.4	98/0.4	57/0.3	56/0.3
	230/0.10		473/2.9	228/1.7	228/1.8	196/1.3
MESH_DEFORM	2300/0.98	[0.07,0.11]	474/2.9	230/1.7	229/1.8	200/1.3
	9200/3.94		462/2.9	236/1.8	217/1.8	207/1.4
	500/0.13		638/7.0	342/6.4	420/7.7	342/6.5
WATSON_1	5000/1.29	[0.45,0.48]	622/6.9	343/6.8	576/9.8	342/6.8
	15000/3.88		627/7.1	333/6.7	612/11.1	332/6.6
	500/1.06		48/0.3	48/0.3	47/0.3	44/0.3
TS-PALKO	1000/2.12	[0.02,0.03]	49/0.2	48/0.3	48/0.4	44/0.3
	2000/4.23		48/0.2	48/0.3	48/0.4	44/0.3
	500/0.13		13/0.3	13/0.3	13/0.4	13/0.3
LP_NUG30	5000/1.32	[0.13,0.15]	14/0.3	14/0.3	14/0.4	16/0.3
	10000/2.64		16/0.3	16/0.3	16/0.4	16/0.3
	5000/0.24		68/5.2	48/5.7	48/6.6	42/4.9
LARGEREGFILE	10000/0.47	[0.42,0.45]	68/5.0	49/5.3	50/5.5	44/5.3
	50000/2.37		69/5.3	51/5.9	51/5.7	48/5.5
	1750/0.10		149/12.1	118/9.7	118/9.8	114/9.1
SLS	17500/1.00	[0.01,0.02]	125/10.2	101/8.4	101/8.6	103/7.9
	70000/4.00		124/10.3	100/8.4	100/8.6	101/7.9
	1400/0.14		20/1.7	17/1.6	17/1.7	16/1.6
TP-6	14000/1.38	[0.02,0.03]	21/1.8	18/1.7	18/1.8	19/1.7
	28000/2.76		21/1.8	18/1.7	18/1.7	19/1.8

TABLE 4.2: Effect of the rank of the update when adding equations or columns. k is the rank of the update in absolute number and percentage, ρ is the density range for all the preconditioners. * indicates total CPU time corresponding to the preconditioner computation and the iterative solution. A † means that the iterative method was unable to converge.

Matrix	k	ρ	non-prec Iter/Time(s)	non-updated Iter/Time(s)	recomputed Iter/Time(s)*	UPD Iter/Time(s)*
PHOTOGRAMMETRY2	50/1.12	[0.03,0.05]	357/0.04	143/0.02	142/0.02	70/0.01
	100/2.24		461/0.05	182/0.03	191/0.03	70/0.01
	200/4.47		859/0.11	495/0.07	497/0.12	70/0.01
TESTBIG	300/0.96	[0.55,0.59]	142/0.1	112/0.1	101/0.2	56/0.1
	600/1.92		141/0.1	113/0.1	99/0.2	56/0.1
	1200/3.84		149/0.1	112/0.1	104/0.2	57/0.1
CAT_EARS_4_4	400/0.9	[0.78,0.84]	146/0.2	78/0.2	78/0.9	87/0.3
	1100/2.47		163/0.3	92/0.3	88/0.9	87/0.3
	2200/4.95		369/0.5	213/0.5	144/0.9	109/0.4
FOME13	200/0.20	[0.79]	460/1.5	276/1.5	272/1.9	230/1.3
	1000/1.02		512/1.8	306/1.6	296/2.0	239/1.4
	4000/4.08		712/3.8	612/1.9	604/2.4	237/1.3
LP_KEN_18	500/0.32	[0.77,0.81]	531/2.6	217/1.7	216/1.8	132/1.1
	1000/0.65		532/2.6	231/1.8	229/1.9	132/1.1
	5000/3.23		573/2.7	227/1.7	216/1.8	134/1.1
FLOWER_8_4	500/0.40	[0.80,0.86]	175/0.7	102/0.7	101/1.2	107/0.8
	1000/0.80		181/0.7	93/0.7	92/1.4	95/0.7
	6000/4.79		315/1.2	186/1.3	144/1.7	104/0.8
FXM3_16	100/0.12	[0.38,0.42]	2115/5.3	1200/4.7	668/3.1	462/1.9
	1000/1.17		2212/5.4	1974/7.2	2228/8.5	470/1.8
	4000/4.68		2670/6.3	2067/7.4	1592/6.7	473/1.8
LP_OSA_30	500/0.48	[0.01,0.02]	132/0.4	60/0.2	55/0.2	56/0.2
	1000/0.96		134/0.4	68/0.2	61/0.3	56/0.2
	5000/4.79		142/0.4	111/0.4	68/0.3	56/0.2
MESH_DEFORM	230/0.10	[0.09,0.19]	482/2.9	261/1.9	230/1.9	198/1.3
	2300/0.98		518/3.3	508/3.6	680/5.1	192/1.2
	9200/3.94		1554/9.1	2464/17.1	†	197/1.4
WATSON_1	500/0.13	[0.48,0.56]	1136/12.4	705/13.0	332/7.0	330/6.5
	5000/1.29		†	†	414/7.9	315/6.4
	15000/3.88		†	†	381/7.1	331/6.5
TS-PALKO	500/1.06	[0.03,0.04]	75/0.4	73/0.4	72/0.8	44/0.3
	1000/2.12		842/3.7	831/3.8	744/3.7	44/0.3
	2000/4.23		2665/10.6	2620/12.3	2346/11.8	44/0.3
LP_NUG30	500/0.13	[0.13,0.15]	20/0.3	32/0.4	18/0.7	17/0.3
	5000/1.32		40/0.6	57/0.9	22/0.8	18/0.3
	10000/2.64		62/1.1	91/1.3	27/1.0	19/0.3
LARGEREGFILE	5000/0.24	[0.44,0.48]	91/6.7	53/5.7	48/6.6	50/5.4
	10000/0.47		93/6.7	53/5.7	47/6.2	56/6.2
	50000/2.37		98/7.0	57/6.2	46/6.4	60/6.3
SLS	175/0.01		209/17.8	154/12.9	155/13.5	129/10.1
	17500/1.00		473/38.5	260/21.7	252/21.9	129/10.0
	70000/4.00		455/36.7	244/20.2	244/20.3	130/10.2
TP-6	1400/0.14	[0.02,0.03]	191/15.3	182/16.0	137/12.3	18/1.7
	14000/1.38		†	†	†	20/1.8
	28000/2.76		†	†	†	21/1.9

TABLE 4.3: Effect of the rank of the update when removing equations or columns. k is the rank of the update in absolute number and percentage, ρ is the density range for all the preconditioners. * indicates total CPU time corresponding to the preconditioner computation and the iterative solution. A † means that the iterative method was unable to converge.

Chapter 5

Preconditioners for rank deficient least squares problems

We use the method presented in Chapter 2 for computing sparse preconditioners for iteratively solving rank deficient least squares problems by using the LSMR method. The main idea of the method proposed is to update an incomplete factorization computed for a regularized problem to recover the solution of the original one. The numerical experiments for a wide set of matrices arising from different science and engineering applications show that the preconditioner proposed, in most cases, can be successfully applied to accelerate the convergence of the iterative Krylov subspace method.

5.1 Introduction

Linear least squares (LS) problems arise in many large-scale applications of the science and engineering as neural networks, linear programming, exploration seismology or image processing, among others. The LS problem considered is formulated as

$$\min_x \|b - Ax\|_2, \tag{5.1}$$

where $A \in \mathbb{R}^{m \times n}$, $m \geq n$ is large and sparse and $b \in \mathbb{R}^m$. This problem can be also formulated in the following mathematically equivalent $n \times n$ normal equations system:

$$A^T Ax = A^T b. \tag{5.2}$$

Two types of methods are usually used to solve these linear systems, direct and iterative methods. Direct methods, in spite of their robustness, require the computation of an explicit factorization of the coefficient matrix of the linear system, that implies large computational time and memory storage. In contrast, iterative Krylov subspace methods may be preferred when the system matrix is large and sparse because they often are less demanding in memory requirements than their direct counterparts. In this case, Equation (5.2) is solved iteratively using conjugate gradient like methods, as the LSMR and CGLS methods, among others. Basically, these methods implicitly apply the conjugate gradient or minimal residual method to the normal equations. See Chapter 1 for a description of these methods and different preconditioning techniques for LS problems.

The chapter is organized as follows. In Section 5.2, we describe the bordering technique used to update an existing preconditioner by using an equivalent augmented system. In Section 5.3, we describe the test environment, and present the set of problems studied. Then, we report on the numerical experiments in Section 5.4, that show that the proposed technique is robust and effective. Finally the conclusions are presented.

5.2 Updated preconditioner method

When the matrix A in (5.1) is rank deficient, one of the approaches for solving the LS problem, as mentioned above, is based on the computation of a Cholesky factorization of the normal equations associated to the regularized matrix

$$\begin{bmatrix} A \\ \alpha^{1/2} I \end{bmatrix}, \tag{5.3}$$

which are given by

$$C_\alpha = A^T A + \alpha I. \tag{5.4}$$

The shift α is known as Tikhonov regularization parameter. If α is choosen large enough the computation of an IC for the matrix C_α can be done easily and without breakdowns. On the other hand, since the final purpose is to use this incomplete factorization as a preconditioner for the original (unregularized) linear system, the parameter α should be chosen as small as possible. Both requirements make difficult the choice of the appropriate α. In practice, one starts with a small value for α, if a breakdown occurs then α is increased successively until a successful incomplete factorization is computed.

We propose a method that simplifies the choice of the regularization parameter and at the same time allows for the use of very sparse preconditioners. It is similar to the technique presented in Chapter 4 in which it is studied how to update a preconditioner for LS problems when the linear system is modified by adding or removing equations.

The idea is to compute a preconditioner for the regularized matrix C_α in (5.4), with α large enough, and then update the preconditioner for the original problem. Consider the matrix

$$C_\alpha - \beta I \tag{5.5}$$

which is an update of the shifted matrix C_α in (5.4). Clearly, the closer β is to α, the closer this update is to the normal equations $A^T A$. Our technique consists of updating an incomplete Cholesky factorization obtained for C_α, and it relies on the relations that one can stablish between the augmented matrix

$$\begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix}, \tag{5.6}$$

and the matrix in (5.5). Observe that

$$C_\alpha - \beta I = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix} \begin{bmatrix} I \\ -\beta^{1/2} I \end{bmatrix}, \tag{5.7}$$

and

$$(C_\alpha - \beta I)^{-1} = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix}. \tag{5.8}$$

Thus, an IC factorization computed for the augmented matrix can be used to approximate the matrix $C_\alpha - \beta I$ and its inverse. Hence, it can be used as a preconditioner for the original normal equations. Note that the modification introduced by the updated in Equation (5.5), makes the choice of the shift α less restrictive. The only condition needed now is to select a large enough value to avoid breakdown during the computation of an IC factorization.

5.2.1 Preconditioner computation

The preconditioner is obtained from the block Cholesky factorization of the augmented matrix in (5.6) given by

$$\begin{pmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{pmatrix} = \begin{pmatrix} L_\alpha & 0 \\ \beta^{1/2} L_\alpha^{-T} & L_R \end{pmatrix} \begin{pmatrix} L_\alpha^T & \beta^{1/2} L_\alpha^{-1} \\ 0 & L_R^T \end{pmatrix}$$

where L_α is the Cholesky factor of C_α and L_R is the Cholesky factor of the Schur complement of C_α in the augmented matrix, $R = I - \beta L_\alpha^{-T} L_\alpha^{-1}$. The preconditioner is computed in four steps, summarized in Algorithm 13:

Algorithm 13 Preconditioner computation

```
**Input:** Matrix $A$, $\alpha$, $\beta$.
**Output:** Matrices $L_\alpha$ and $L_R$.
1. Compute and IC: $L_\alpha L_\alpha^T \approx C_\alpha = A^T A + \alpha I$.
2. Compute $T = \beta^{1/2} L_\alpha^{-1}$.
3. Compute $R = I - T^T T$.
4. Compute an IC: $L_R L_R^T \approx R$.
```

To keep the preconditioner sparse, the amount of fill-in may be limited by dropping small elements in steps 2 and 3.

5.2.2 Preconditioner application

The preconditioning step for a Krylov subspace iterative method involves the solution of systems of the form $Ms = r$ where M is the preconditioner and r is the residual. Thus, the preconditioning strategy proposed computes the preconditioned residual by applying Equation (5.8) with an incomplete factorization of the augmented matrix. That is, the preconditioned residual s is given by

$$s = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix} r,$$

and it is computed from the solution of the block linear system

$$\begin{pmatrix} L_\alpha & 0 \\ \beta^{1/2} L_\alpha^{-T} & L_R \end{pmatrix} \begin{pmatrix} L_\alpha^T & \beta^{1/2} L_\alpha^{-1} \\ 0 & L_R^T \end{pmatrix} \begin{pmatrix} s \\ s_1 \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}.$$

The preconditioning step is done as shown in Algorithm 14. These three steps will

Algorithm 14 Preconditioner update application

```
**Input:** $\beta$, matrices $L_\alpha$, $L_R$ and residual vector $r$.
**Output:** Preconditioned vector $s$
1. $s \leftarrow L_\alpha^{-T}(L_\alpha^{-1} r)$.
2. $s_R \leftarrow L_R^{-T}(L_R^{-1} \bar{r})$.
3. $s \leftarrow s + \beta L_\alpha^{-T}(L_\alpha^{-1} s_R)$.
```

be referenced as updated preconditioner method (UPD). Steps 2 and 3 represent the extra cost in the application of the preconditioner with respect to the non-updated case. We recall that the inverses of the triangular factors are applied by solving the corresponding triangular systems. If L_α and L_R are kept sparse, the additional cost is small and can be amortized even for moderate reductions on the number of iterations, see [24].

A final observation with respect to β is that, since $\operatorname{rank} A = k < n$, theoretically one should choose $\beta \neq \alpha$. Otherwise, the square augmented matrix in (5.6) is singular. To see this, let us do a symmetric permutation to the augmented matrix as follows

$$\begin{bmatrix} O & I \\ I & O \end{bmatrix} \begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix} \begin{bmatrix} O & I \\ I & O \end{bmatrix} = \begin{bmatrix} I & \beta^{1/2} I \\ \beta^{1/2} I & C_\alpha \end{bmatrix}.$$

After eliminating the left bottom block by Gaussian elimination we obtain

$$\begin{bmatrix} I & \beta^{1/2} I \\ 0 & C_\alpha - \beta I \end{bmatrix} = \begin{bmatrix} I & \beta^{1/2} I \\ 0 & A^T A + (\alpha - \beta) I \end{bmatrix},$$

which has no full rank if $\beta = \alpha$. In practice, as we will see in Section 5.4, a value of β equals to α was fine for the problems reported.

5.3 Numerical environment

The experiments were done with MATLAB version 2016a running on an Intel 5 CPU with 8 Gb of RAM in a Windows operating system. For the solution of each problem we set a limit of 600 seconds and 50,000 iterations for the total CPU time and number of iterations, respectively. As recommended in MATLAB documentation, CPU times

Matrix	m	n	nnz	nullity	Application
BAXTER	27441	30733	111576	3055	Linear programming
DBIR1	18804	45775	1077025	2	Linear programming
DBIR2	18906	45877	1158159	2	Linear programming
NSCT1	22901	37461	678739	1	Linear programming
NSCT2	23003	37563	697738	1	Linear programming
beaflw	492	500	53403	32	Economic
Pd_rhs	5804	4371	6323	3	Counter-example
162bit	3606	3476	37118	16	Combinatorial
176bit	7441	7150	82270	40	Combinatorial
192bit	13691	13093	154303	87	Combinatorial
208bit	24430	23191	299756	210	Combinatorial
wheel_601	902103	723605	2170814	600	Combinatorial
12month1	12471	872622	22624727	53	Bipartite graph
ND_actors	383640	127823	1470404	13061	Bipartite graph
IMDB	303617	896302	3782463	53101	Bipartite graph
Maragal_6	21251	10144	537694	92	Least squares
Maragal_7	46845	26525	1200537	659	Least squares
Maragal_8	33093	60845	1308415	14637	Least squares
mri1	65536	114637	589824	1019	graphics/vision
mri2	63240	104597	569160	14919	graphics/vision
tomographic1	142752	1014301	11537419	3700	graphics/vision

TABLE 5.1: Set of tested matrices

reported are the mean value of 10 successive runs of the experiment performed after 3 initial runs that were discarded. The maximum standard deviation observed relative to the mean value for the different runs was insignificantly.

Table 5.1 shows the set of tested matrices from the Florida Sparse Matrix Collection [35], arising in different areas of scientific computing. The matrices were cleaned by removing the null rows and columns before solving the LS problem. The number of rows and columns, number of nonzeros (nnz) and nullity of the matrix (estimated null space rank) are reported.

If a matrix has less rows than columns, then it is transposed. Each tested matrix was permuted using the MATLAB function **dmperm**() that obtains the Dulmage-Mendelsohn decomposition [71]. This decomposition estimates an upper bound of the structural rank of the matrices, that allows for the approximation of the nullity. The values obtained for the nullity coincide with the ones reported in [78]. The columns of the matrix corresponding to the normal equations $C = A^T A$ were normalised by their 2-norm. As result, the regularized normal equations matrix is

$$C_\alpha = SPCP^T S^T + \alpha I.$$

An IC factorization $L_\alpha L_\alpha^T$ of C_α was computed with the MATLAB function **ichol**$()$. With respect to the Schur complement R in Algorithm 13, small elements were dropped in step 2 before computing step 3, and finally an incomplete LU factorization using the MATLAB function **ilu**$()$ was computed in step 4. For the updated preconditioner, a value of $\alpha = \beta = 1$ was used for all the matrices, except for the matrices BAXTER and BEAFLW for which a value of $\alpha = \beta = 10^{-3}$ was needed. We have found that in practice, choosing $\alpha = \beta$, has given the best results. The right-hand side b, in all the cases, is the vector of all ones.

The LSMR method was used to solve the normal equations because the norm of the residual decrease monotonically as mentioned in the Section 4.1. The MATLAB impletation of LSMR, ALgorithm Algorithm 3, was downloaded from this repository [58]. Since we do not have an explicit factorization of the normal equations, we do not apply two side preconditioning as suggested by the authors. Therefore, following the idea in [4] where the authors derived a left preconditioned LSQR algorithm, we implemented a left preconditioned version of the LSMR presented in 7.

5.4 Numerical experiments

In this section we study the numerical performance of the preconditioner update method proposed. The method has been compared with an IC factorization of the regularized matrix C_α. Before analyzing the performance of the preconditioner, we study the convergence criteria for the LSMR method and the choice of the Tikhonov regularization parameter.

5.4.1 Study of the convergence criteria and choice of α

We have done an exhaustive study concerning the stopping criteria for the convergence of the LSMR method. In recent implementations of the LSMR algorithm, different convergence criteria are proposed in the bibliography.

FS: Fong and Saunders in [38] propose the following stopping rule

$$\frac{||A^T r_k||_2}{||A||_2 ||r_k||_2} < \epsilon. \tag{5.9}$$

GS: Gould and Scott in [44] proposed a different criterion defined by

$$\frac{||A^T r_k||_2 ||r_0||_2}{||A^T r_0||_2 ||r_k||_2} < \epsilon, \tag{5.10}$$

that reduces to

$$\frac{||A^T r_k||_2 ||b||_2}{||A^T b||_2 ||r_k||_2} < \epsilon,$$

when the initial solution guess is $x_0 = 0$. It can be easily observed the following relation between both criteria

$$\frac{||A^T r_k||_2}{||A||_2 ||r_k||_2} = \frac{||A^T r_k||_2 ||b||_2}{||A^T||_2 ||b||_2 ||r_k||_2} \leq \frac{||A^T r_k||_2 ||b||_2}{||A^T b||_2 ||r_k||_2}.$$

Thus, with the FS criterion an iterative method may converge in fewer iterations.

We remark that the convergence rule programmed by default in the LSMR depends on the preconditioner M applied, since it evaluates the norm $||(AM^{-1})^T r||_2$. To remove this dependency in the sense of Gould and Scott, we modify the FS criterion by computing instead the norm $||A^T r||_2$. Moreover, as the authors did in [44], we exclude the additional computational time needed to compute the corresponding residuals from the total solution time.

To study the effect of the stopping criteria on the convergence of the LSMR method, we consider for instance, the matrix DBIR1. The convergence tolerance was set to $\epsilon = 10^{-6}$. For the Tikhonov parameter α, values in the interval $[0.01, 2]$ were considered. We note that for very small values of α the MATLAB function **ichol**() produces very dense preconditioners. Therefore, the IC factorization $L_\alpha L_\alpha^T$ of C_α was obtained with the MATLAB function **ilu**() with drop tolerance 0.01. With this function sparser factorizations were obtained with a considerable reduction of the computational time.

Figure 5.1 shows the CPU time and the number of iterations that the LSMR method takes to converge with both stopping rules and both preconditioners. One can observe that the best results are obtained for values of α in the interval $[0.1, 1]$. But, with $\alpha = 1$ sparser preconditioners were obtained.

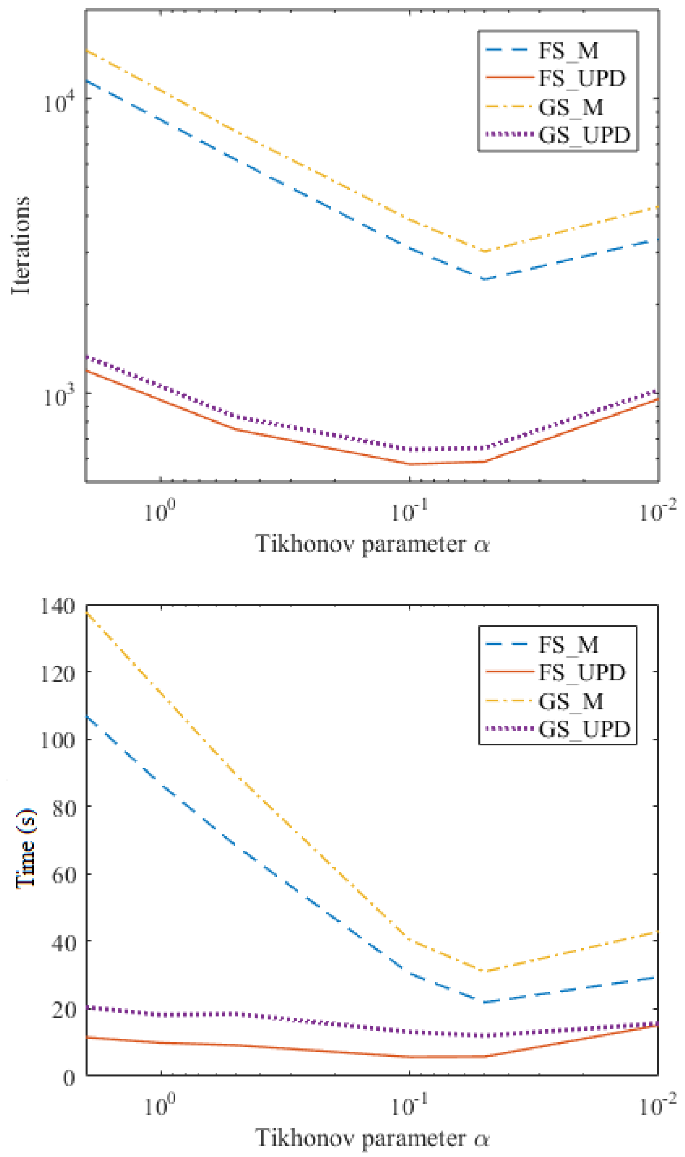


FIGURE 5.1: DBIR1 matrix. Number of iterations and total solution time, for the LSMR method with FS and GS criteria, and for the non-updated (M) and updated (UPD) preconditioners, α in $[0.01, 2]$.

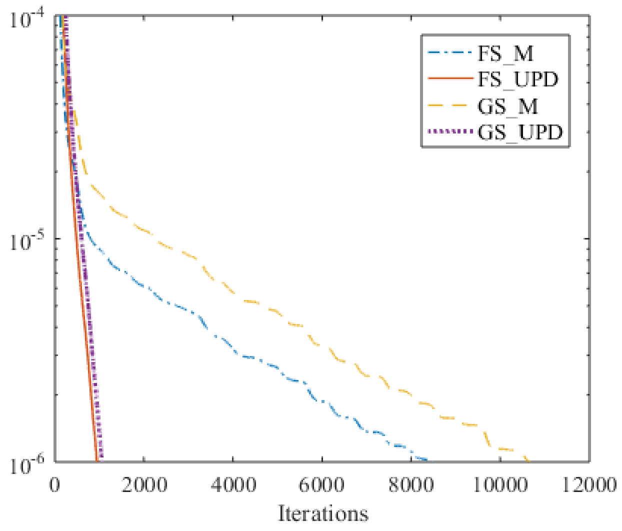


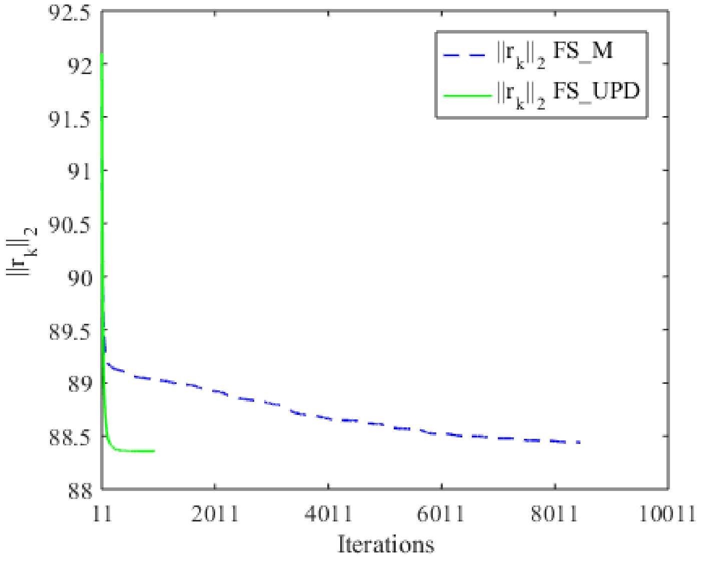
FIGURE 5.2: DBIR1 matrix. Evolution of the FS and GS criteria with respect to the number of iterations; y axis represents the value of the FS and GS criteria at each iteration.

In general, we found that the value $\alpha = 1$ was a good choice for the majority of the problems tested because no breakdowns were produced, and the performance of the iterative method was reasonable. Therefore, the results presented below are obtained with this value with some exceptions. We recall that one of the objectives was to avoid the iterative process for selecting the value of α.

With respect to the convergence criteria it is observed that the LSMR method with FS rule needs less number of iterations and computational time to converge for all values of α.

Figure 5.2 shows the evolution of the FS and GS criteria for a fixed value of $\alpha = 1$ during the iterative solution process. As mentioned before, the FS rule converges in less iterations, specially with the non-updated preconditioner.

Finally, Figure 5.3 compares the evolution of $||r_k||_2$ for the non-updated and updated preconditioner and both convergence criteria. It can be observed that the UPD preconditioner converges in less iterations than the non-updated one.



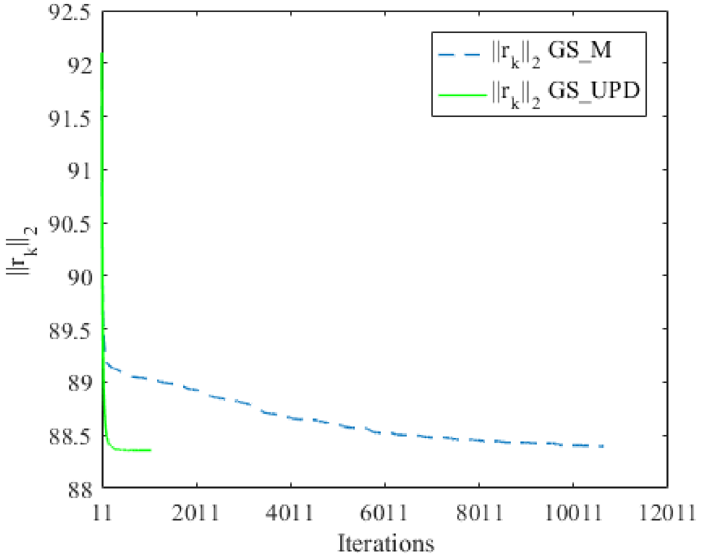


FIGURE 5.3: DBIR1 matrix. Evolution of $||r_k||_2$ for the non-updated and updated preconditioner.

Matrix	UPD				M			
	rho_UPD	Time(s)	$\|\|r\|\|_2$	Iter	rho_M	Time(s)	$\|\|r\|\|_2$	Iter
BAXTER	0.53	42.1	74.99	2115	0.28	**39.6**	74.99	**2114**
DBIR1	0.04	**1.7**	88.44	**294**	0.02	107.2	88.44	22163
DBIR2	0.03	**2.8**	87.57	**477**	0.02	16.7	87.57	3434
NSCT1	0.07	**1.2**	93.80	**261**	0.04	2.0	93.80	554
NSCT2	0.07	**3.0**	88.59	**701**	0.04	25.1	88.59	6939
BEAFLW	1.37	**6.1**	4.53	**4074**	1.36	9.6	4.53	6504
PD_RHS	1.40	**0.1**	34.62	**186**	0.71	0.2	34.62	830
162BIT	0.20	**0.2**	0.62	**389**	0.10	0.2	0.62	397
176BIT	0.18	**0.3**	0.80	**383**	0.09	0.3	0.80	394
192BIT	0.18	**0.6**	1.28	**360**	0.09	0.6	1.28	373
208BIT	0.16	**1.0**	1.62	**321**	0.08	1.0	1.62	336
WHEEL_601	0.83	**2.9**	497.51	**32**	0.50	3.8	497.51	45
12MONTH1	0.01	**24.3**	679.32	**154**	0.01	30.9	679.32	182
ND_ACTORS	0.18	**133.4**	301.65	**6491**	0.09	140.5	301.65	7188
IMDB	1.64	113.7	497.65	1442	0.08	**112.1**	497.65	**1427**
MARAGAL_6	0.04	5.8	93.98	**1637**	0.02	**5.6**	93.98	1771
MARAGAL_7	0.05	5.2	133.13	496	0.03	**4.5**	133.13	**492**
MARAGAL_8	0.06	141.6	238.90	15724	0.04	**129.5**	238.90	**15683**
MRI1	0.22	16.4	26.74	2616	0.11	**13.8**	26.74	**2537**
MRI2	0.27	11.3	141.26	1692	0.15	**9.7**	141.26	**1583**
TOMOGRAPHIC1	0.01	**10.7**	42.18	**1309**	0.01	13.1	42.18	1983

TABLE 5.2: Results for LSMR with the non-updated (M) and with the updated (UPD) preconditioners.

5.4.2 Results

The results that presented in this subsection are computed with the modified FS stopping criterion and $\alpha = 1$, except for the matrices BAXTER and BEAFLW for which a value of $\alpha = \beta = 10^{-3}$ was needed.

Table 5.2 shows the results for the different matrices tested. In this table, Time(s), $||r||_2$ and Iter represent the total time (in seconds, including computation of the preconditioner), residual norm ($||b - Ax||_2$) and the number of iterations needed to converge, respectively. M corresponds to the results obtained with the IC factorization of C_α while UPD corresponds to the ones obtained with the proposed updated preconditioning technique. The density of each preconditioner is also presented, rho_UPD and rho_M. The iterative method was stopped when the stopping rule FS was reduced to 10^{-6}. A maximum number of 50000 iterations was allowed. L_α was calculated with drop tolerance equal to 0.1, except for the matrices MRI1 and MRI2 for which a value of 0.2 was used, and a value of 10^{-5} for matrices BAXTER and BEAFLW.

The problems are classified into three blocks mainly taking into account the field of application. The best results for every problem, in total solution time and number of iterations are emphasized with bold type.

The first block of matrices were not cleaned because it was not necessary deleting null columns and rows. For this matrices the UPD preconditioner was able to reduce the time spent with the IC preconditioner considerably. Specially significant are the cases of the DBIR1 and NSCT2 matrices. For the second block we can observe that the UPD method is also competitive, although, the improvement with respect to the IC factorization is not so big as in the previous block of matrices. In the last block, the results were not so clear, and there were cases for which the UPD preconditioner performed better, and others where it was observed the opposite.

In conclusion, the results show that the updated preconditioner method is competitive and robust for solving rank deficient least squares problems. The number of iterations and time spent was the best, or close to it, for all the problems tested.

We recall that the preconditioners used were quite sparse, that is very important for solving much larger problems.

5.5 Conclusions

We have presented a method for preconditioning rank deficient least squares problems that can be viewed as an update preconditioner technique for the regularized normal equations. From the numerical results conducted it has been observed that the proposed preconditioner is competitive in terms of solution time and number of iterations spent. Furthermore, the method simplifies the choice of the Tikhonov regularization parameter α, and a fixed value equals to 1 was usually used. With this choice, we were able to compute very sparse preconditioners. Thus, we think that the preconditioner proposed can be successfully applied to accelerate the convergence rate of the LSMR method.

General conclusions

We summarize the contributions and conclusions of this thesis and finish with some suggestions for future research. We have focused on preconditioners for large sparse systems of linear equations $Ax = b$. In Chapter 2, we proposed a preconditioning technique based on updating an already calculated preconditioner referred to as Updated Preconditioner Method (UPD). The strategy is based on the computation of an approximate factorization for an equivalent augmented linear system. This technique has been used to solve non-symmetric linear systems and least squares problems.

In Chapter 3, the method was used for preconditioning non-symmetric systems whose skew-symmetric part is low-rank or can be well approximated by a low-rank matrix. Some approximation properties of the preconditioner and the eigenvalue distribution of the preconditioned matrix have been presented. It has been show that the technique sparse column row approximation (SCRA) produces good low-rank approximations of the skew-symmetric part. The updated method has been compared with others that appear in the literature for this kind of matrices, particularly with the SCM method. We solved several artificial and application problems arising from many areas of science and engineering. From the numerical results conducted it has been observed that the proposed preconditioner was competitive in terms of solution time and number of iterations.

In Chapter 4, the preconditioner UPD was adapted to be used as preconditioner for solving modified least squares problems, that is, when new equations are added, or removed or both, as well as adding or removing or simply changing some variables. For this kind of problems the CGLS method was used, without preconditioning, preconditioned with an ILU factorization of the normal equations or the original problem, preconditioned with a recomputed ILU factorization of the normal equations of the modified problem and preconditioned with the update technique proposed. From the numerical results, we can conclude that the preconditioner UPD is competitive and robust since it was able to successfully solve all the problems. Moreover, the number of iterations and time spent was the best, or close to it, for most of the problems tested. Another conclusion is that, in general, it is better to apply the updated preconditioner or recompute a new preconditioner from scratch instead of reusing the original one. In any case, any preconditioning strategy is better than non-preconditioning. Computing a new preconditioner from scratch may have two drawbacks: the first one is an increment on the set-up time that usually only pays off in the case of adding equations when the size of the update is quite large. The second one is that when removing equations, the preconditioner computation may become unstable probably due to an increment of the condition number of the coefficient matrix of the normal equations. In general, the technique proposed allows to update the preconditioner in an inexpensive way.

In Chapter 5 we deal with the least squares solution of rank deficient linear systems. To compute a preconditioner for these problems we combine the proposed strategy with Tykhonov's regularization. The LSMR method was preconditioned the UPD preconditioner and compared with an IC factorization of the regularized normal equations. UPD was competitive in terms of solution time and number of iterations needed to converge. Furthermore, with the proposed method the iterative process to estimate the right choice of the Tikhonov regularization parameter α was avoided. Indeed, for most of the problems a fixed value $\alpha = 1$ was used with good results.

In general, the Updated Preconditioner Method proposed has been effective and robust for solving the different problems considered in this thesis.

Bibliography

[1] Alexander S. T., Pan C. T. and Plemmons R. J. “Analysis of a recursive least squares hyperbolic rotation algorithm for signal processing”. In: *Linear Algebra Appl.* 98. Supplement C (1988), pp. 3 –40.

[2] Andrew R. and Dingle N. “Implementing QR factorization updating algorithms on GPUs”. In: *Parallel Comput.* 40.7 (2014). 7th Workshop on Parallel Matrix Algorithms and Applications, pp. 161 –172.

[3] Arnoldi W. E. “The principle of minimized iterations in the solution of the matrix eigenvalue problem”. In: *Q. Appl. Math* 9.17 (1951), pp. 17–29.

[4] Arridge S. R., Betcke M. M. and Harhanen L. “Iterated preconditioned LSQR method for inverse problems on unstructured grids”. In: *Inverse Probl.* 30.7 (2014), p. 075009.

[5] Atkinson K. E. *An introduction to numerical analysis*. Wiley, 1978.

[6] Atkinson K. E. “The Numerical Solution of Fredholm integral Equations of the Second Kind”. In: *SIAM J. Numer. Anal.* 4.3 (1967), pp. 337–348.

[7] Axelsson O. “A general incomplete block-matrix factorization method”. In: *Linear Algebra Appl.* 74. Supplement C (1986), pp. 179 –190.

[8] Axelsson O. *Iterative Solution Methods*. New York, NY, USA: Cambridge University Press, 1994.

[9] Baker C. *The Numerical Treatment of Integral Equations*. Monographs on Numerical Analysis Series. Oxford : Clarendon Press, 1977.

[10] Barrett R., Berry M., Chan T. F., Demmel J., Donato J., Dongarra J., Eijkhout V., Pozo R., Romine C. and Vorst H. V. der. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. Philadelphia, PA: SIAM, 1994.

[11] Beauwens R. “Modified incomplete factorization strategies”. In: *Preconditioned Conjugate Gradient Methods: Proceedings of a Conference held in Nijmegen, The Netherlands, June 19–21, 1989*. Ed. by O. Axelsson and L. Y. Kolotilina. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 1–16.

[12] Beckermann B. and Reichel L. "The Arnoldi process and GMRES for nearly symmetric matrices." In: *SIAM J. Matrix Anal. Appl.* 30.1 (1998), pp. 102–120.

[13] Benzi M. "Preconditioning Techniques for Large Linear Systems: A Survey". In: *J. Comput. Phys.* 182.2 (2002), pp. 418–477.

[14] Benzi M., Kouhia R. and Tůma M. "Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics". In: *Comput. Methods Appl. Mech. Eng.* 190.49 (2001), pp. 6533 –6554.

[15] Benzi M., Szyld D. B. and Duin A. V. "Orderings for incomplete factorization preconditioning of nonsymmetric problems". In: *SIAM J. Sci. Comput.* 20.5 (1999), pp. 1652–1670.

[16] Benzi M. and Tůma M. "A Comparative Study of Sparse Approximate Inverse Preconditioners". In: *Appl. Numer. Math.* 30.2-3 (1999), pp. 305–340.

[17] Benzi M. and Tůma M. "A robust incomplete factorization preconditioner for positive definite matrices". In: *Numer. Linear Algebra Appl.* 10.5-6 (2003), pp. 385–400.

[18] Bergamaschi L., Gondzio J., Venturin M. and Zilli G. "Inexact constraint preconditioners for linear systems arising in interior point methods". In: *Comput. Optim. Appl.* 36.2 (2007), pp. 137–147.

[19] Berry M. W., Pulatova S. A. and Stewart G. W. "Algorithm 844: Computing Sparse Reduced-rank Approximations to Sparse Matrices". In: *ACM Trans. Math. Softw.* 31.2 (2005), pp. 252–269.

[20] Bjorck A. *Numerical Methods for Least Squares Problems*. Siam Philadelphia, 1996.

[21] Bru R., Marín J., Mas J. and Tůma M. "Balanced incomplete factorization." In: *SIAM J. Sci. Comput.* 30.5 (2008), pp. 2302–2318.

[22] Bru R., Marín J., Mas J. and Tůma M. "Improved balanced incomplete factorization." In: *SIAM J. Matrix Anal. Appl.* 31.5 (2010), pp. 2431–2452.

[23] Bru R., Marín J., Mas J. and Tuma M. "Preconditioned Iterative Methods for Solving Linear Least Squares Problems". In: *SIAM J. Sci. Comput.* 36.4 (2014).

[24] Cerdán J., Marín J. and Mas J. "Low-rank updates of balanced incomplete factorization preconditioners". In: *Numer. Algorithms* 74.2 (2017), pp. 337–370.

[25] Cerdán J., Guerrero D., Marín J. and Mas J. "Preconditioners for nonsymmetric linear systems with low-rank skew-symmetric part". In: *J. Comput. Appl. Math.* (submitted).

[26] Cerdán J., Faraj T., Malla N., Marín J. and Mas J. "Block approximate inverse preconditioners for sparse nonsymmetric linear systems". In: *Electron. Trans. Numer. Anal.* 37 (2010), pp. 23–40.

[27] Chambers J. M. "Regression Updating". In: *J. Am. Stat. Assoc.* 66.336 (1971), pp. 744–748.

[28] Chapman A., Saad Y. and Wigton L. "High-order ILU preconditioners for CFD problems". In: *Int. J. Numer. Methods Fluids* 33.6 (2000), pp. 767–788.

[29] Choi S.-C. T., Paige C. C. and Saunders M. A. "MINRES-QLP: A Krylov Subspace Method for Indefinite or Singular Symmetric Systems". In: *SIAM J. Sci. Comput.* 33.4 (2011), pp. 1810–1836.

[30] Chow E. and Saad Y. "Experimental study of ILU preconditioners for indefinite matrices". In: *J. Comput. Appl. Math.* 86.2 (1997), pp. 387 –414.

[31] Chow E. and Saad Y. "ILUS: An incomplete LU preconditioner in sparse skyline format". In: *Int. J. Numer. Methods Fluids* 25.7 (1997), pp. 739–748.

[32] Davis T. A. and Hager W. W. "Modifying a Sparse Cholesky Factorization". In: *SIAM J. Matrix Anal. Appl.* 20.3 (1999), pp. 606–627.

[33] Davis T. A. and Hager W. W. "Multiple-Rank Modifications of a Sparse Cholesky Factorization". In: *SIAM J. Matrix Anal. Appl.* 22.4 (2000), pp. 997–1013.

[34] Davis T. A. and Hager W. W. "Row Modifications of a Sparse Cholesky Factorization". In: *SIAM J. Matrix Anal. Appl.* 26.3 (2005), pp. 621–639.

[35] Davis T. A. and Hu Y. "The University of Florida Sparse Matrix Collection". In: *ACM Trans. Math. Softw.* 38.1 (2011), 1:1–1:25.

[36] Demmel J., Grigori L. and Cayrols S. *Low Rank Approximation of a Sparse Matrix Based on LU Factorization with Column and Row Tournament Pivoting*. Tech. rep. UCB/EECS-2016-122. EECS Department, University of California, Berkeley, 2016.

[37] Embree M., Sifuentes J. A., Soodhalter K. M., Szyld D. B. and Xue F. "Short-term recurrence Krylov subspace methods for nearly hermitian matrices." In: *SIAM J. Matrix Anal. Appl.* 33.2 (2012), pp. 480–500.

[38] F D. C. L. and Saunders M. "LSMR: An Iterative Algorithm for Sparse Least-Squares Problems". In: *SIAM J. Sci. Comput.* 33.5 (2011), pp. 2950–2971.

[39] Fletcher R. "Conjugate gradient methods for indefinite systems". In: *Numerical Analysis*. Ed. by G. Watson. Vol. 506. Lecture Notes in Mathematics. Springer Berlin and Heidelberg, 1976. Chap. 7, pp. 73–89.

[40] Fong C., Saunders M., Gerritsen M. and Murray W. *Minimum-residual Methods for Sparse Least-squares Using Golub-Kahan Bidiagonalization*. Stanford University. Institute for Computational and Mathematical Engineering, 2011.

[41] Golub G. "Numerical Methods for Solving Linear Least Squares Problems". In: *Numer. Math.* 7.3 (1965), pp. 206–216.

[42] Golub G. and Kahan W. “Calculating the Singular Values and Pseudo-Inverse of a Matrix”. In: *SIAM J. Appl. Math.* 2.2 (1965), pp. 205–224.

[43] Golub G. H. and Loan C. F. V. *Matrix Computations (3rd Ed.)* Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[44] Gould N. and Scott J. “The State-of-the-Art of Preconditioners for Sparse Linear Least-Squares Problems”. In: *ACM Trans. Math. Softw.* 43.4 (2017), 36:1–36:35.

[45] Greenbaum A. *Iterative Methods for Solving Linear Systems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997.

[46] Hammarling S. and Lucas C. “Updating the QR factorization and the least squares problem”. In: (2008).

[47] Haynsworth E. V. “On the Schur complement, Basel Mathematical Notes, (University of Basel)”. In: *BMN 20* (1968).

[48] Heath M. T. “Numerical Methods for Large Sparse Linear Least Squares Problems”. In: *SIAM J. Sci. Stat. Comput.* 5.3 (1984), pp. 497–513.

[49] Heinig G. and Rost K. *Algebraic methods for Toeplitz-like matrices and operators*. Mathematical research. Akademie-Verlag, 1984.

[50] Hestenes M. R. and Stiefel E. “Methods of Conjugate Gradients for Solving Linear Systems”. In: *J. Res. Natl. Bur. Stand.* 49.6 (1952), pp. 409–436.

[51] Hopcroft J. and Kannan R. *Foundations of Data Science*. 2014.

[52] HSL. “A collection of Fortran codes for large scale scientific computation”. In: *http://www.hsl.rl.ac.uk/* (N.D.).

[53] Lanczos C. *An iterative method for the solution of the eigenvalue problem of linear differential and integral operators*. 1950.

[54] Lanczos C. “Solution of systems of linear equations by minimized iterations”. In: *J. Res. Natl. Bur. Stand* 49 (1952), pp. 33–53.

[55] Lawson C. and Hanson R. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995.

[56] Li N. and Saad Y. “MIQR: A Multilevel Incomplete QR Preconditioner for Large Sparse Least-Squares Problems”. In: *SIAM J. Matrix Anal. Appl.* 28.2 (2006), pp. 524–550.

[57] Love R. R. “The Electrostatic Field of Two Equal Circular Co-Axial Conducting Disks”. In: *Quart. J. Mech. Appl. Math.* 2.4 (1949), pp. 428–451.

[58] *LSMR Software for Linear Systems and Least Squares*. http://web.stanford.edu/group/SOL/software/lsmr/. 2010.

[59] Manteuffel T. A. “An Incomplete Factorization Technique for Positive Definite Linear Systems”. In: *Math. Comp.* 34 (1980), pp. 473–497.

[60] Marín J., Mas J., Guerrero D. and Hayami K. “Updating preconditioners for modified least squares problems”. In: *Num. Alg.* (2017), pp. 1–18.

[61] Markovsky I. *Low Rank Approximation: Algorithms, Implementation, Applications*. Springer Publishing Company, Incorporated, 2011.

[62] Meijerink J. A. and Vorst H. A. van der. “An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix”. In: *Math. Comp.* 31.137 (1977), pp. 148–162.

[63] Meijerink J. and Vorst H. van der. “Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems”. In: *J. Comput. Phys.* 44.1 (1981), pp. 134 –155.

[64] Olsson O. and Ivarsson T. *Using the QR Factorization to swiftly update least squares problems*. Student Paper. 2014.

[65] Olszanskyj S. J., Lebak J. M. and Bojanczyk A. W. “Rank-k modification methods for recursive least squares problems”. In: *Numer. Algorithms* 7.2 (1994), pp. 325–354.

[66] Paige C. C. and Saunders M. A. “Solution of Sparse Indefinite Systems of Linear Equations”. In: *SIAM J. Numer. Anal.* 12.4 (1975), pp. 617–629.

[67] Paige C. C. and Saunders M. A. “Algorithm 583: LSQR: Sparse Linear Equations and Least Squares Problems”. In: *ACM Trans. Math. Softw.* 8.2 (1982), pp. 195–209.

[68] Paige C. C. and Saunders M. A. “LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares”. In: *ACM Trans. Math. Softw.* 8.1 (1982), pp. 43–71.

[69] Pan C. T. and Plemmons R. J. “Least squares modifications with inverse factorizations: Parallel implications”. In: *J. Comput. Appl. Math.* 27.1 (1989). Special Issue on Parallel Algorithms for Numerical Linear Algebra, pp. 109 –127.

[70] Pastore P. “The Numerical Treatment of Love’s Integral Equation Having Very Small Parameter”. In: *J. Comput. Appl. Math.* 236.6 (2011), pp. 1267–1281.

[71] Pothen A. and Fan C.-J. “Computing the Block Triangular Form of a Sparse Matrix”. In: *ACM Trans. Math. Softw.* 16.4 (1990), pp. 303–324.

[72] Saad Y. “ILUT: A dual threshold incomplete LU factorization.” In: *Numer. Linear Algebra Appl.* 1.4 (1994), pp. 387–402.

[73] Saad Y. *Iterative methods for sparse linear systems.* Philadelphia: SIAM, 2003.

[74] Saad Y. “Krylov subspace methods for solving large unsymmetric linear systems”. In: *Math. Comput.* 37 (1981), pp. 105–126.

[75] Saad Y. “The Lanczos Biorthogonalization Algorithm and Other Oblique Projection Methods for Solving Large Unsymmetric Systems”. In: *SIAM J. Numer. Anal.* 19.3 (1982), pp. 485–506.

[76] Saad Y. and Schulz M. “GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.” In: *SIAM J. Sci. Comput.* 7.3 (1986), pp. 856–869.

[77] Saad Y. and Vorst H. A. van der. “Iterative Solution of Linear Systems in the 20th Century”. In: *J. Comput. Appl. Math.* 123.1-2 (2000), pp. 1–33.

[78] Scott J. “On Using Cholesky-Based Factorizations and Regularization for Solving Rank-Deficient Sparse Linear Least-Squares Problems”. In: *SIAM J. Sci. Comput.* 39 (2017), pp. C319–C339.

[79] Sherman J. and Morrison W. J. “Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix”. In: *Ann. Math. Statist.* 21.1 (1950), pp. 124–127.

[80] Simoncini V. and Szyld D. B. “Interpreting IDR as a Petrov–Galerkin Method”. In: *SIAM J. Sci. Comput.* 32 (2010), pp. 1898–1912.

[81] Sonneveld P. “CGS, a Fast Lanczos-type Solver for Nonsymmetric Linear Systems”. In: *SIAM J. Sci. Stat. Comput.* 10.1 (1989), pp. 36–52.

[82] Sonneveld P. and Gijzen M. B. van. “IDR(s): A Family of Simple and Fast Algorithms for Solving Large Nonsymmetric Systems of Linear Equations”. In: *SIAM J. Sci. Comput.* 31.2 (2009), pp. 1035–1062.

[83] Souhar O. and Guessous N. “Block ILU preconditioned iterative methods for reduced systems”. In: *Canad. Appl. Math. Quart.* 12 (2004), pp. 351–370.

[84] *Sparse Column Row Approximation (SCRA) Method Code, note= ftp://ftp.umiacs.umd.edu/pub/stewart/reports/Contents.html.*

[85] Stewart G. “Error Analysis of the Quasi-Gram–Schmidt Algorithm”. In: *SIAM J. Matrix Anal. Appl.* 27 (2005), pp. 493 –506.

[86] Stewart G. “Four algorithms for the the efficient computation of truncated pivoted QR approximations to a sparse matrix”. In: *Numerische Mathematik* 83.2 (1999), pp. 313–323.

[87] Stewart G. *Introduction to matrix computations*. Computer science and applied mathematics. Academic Press, 1973.

[88] Trefethen L. and Bau D. *Numerical Linear Algebra*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 1997.

[89] Varga R. *Matrix Iterative Analysis*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2009.

[90] Vassilevski P. *Multilevel Block Factorization Preconditioners: Matrix-based Analysis and Algorithms for Solving Finite Element Equations*. Springer New York, 2008.

[91] Vorst H. van der. "BICGSTAB: A fast and smoothly converging variant of BICG for the solution of non-symmetric linear systems." In: *SIAM J. Sci. Comput.* 12.4 (1992), pp. 631–644.

[92] Vorst H. van der. *Iterative Krylov Methods for Large Linear Systems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[93] Young D. and Rheinboldt W. *Iterative Solution of Large Linear Systems*. Computer science and applied mathematics. Elsevier Science, 2014.

[94] Yun J. H. "Block incomplete factorization preconditioners for a symmetric block-tridiagonal M-matrix". In: *J. Comput. Appl. Math.* 94.2 (1998), pp. 133 –152.

[95] Zhang F. *The Schur Complement and Its Applications*. Numerical Methods and Algorithms. Springer, 2005.