UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICA

# Métodos Iterativos Precondicionados para Resolver Problemas de Mínimos Cuadrados Lineales.

Trabajo Especial de Grado presentado ante la ilustre Universidad Central de Venezuela por la **Br. Ysamar Y. Meza G.** para optar al título de Licenciada en Matemática.
**Tutor: Brígida Molina**

Caracas, Venezuela

Mayo, 2011

## Dedicatoria

A mis padres, los mejores que pudo asignar Dios para mi.

A todas aquellas personas que aman la matemática con pasión y desean siempre aprender más de ella.

*"La matemática es el alfabeto con el cual Dios ha escrito el universo."*
*Galileo Galilei.*

**Agradecimientos**

A Dios por darme salud en todo momento para poder desarrollar este trabajo.

A mis padres por apoyarme y darme fuerzas desde los inicios hasta el último minuto y a mis dos hermanos por animarme y brindarme su ayuda para culminar este trabajo evitándome muchos inconvenientes.

A mi tutora, la Dra. Brígida Molina por ser mi guía en este paso tan importante.

A mis mega amigos Alfonso, Lorenzo, Ronaldys, Amanda, María, Javier, Hovsep, Frank, Freysimar y Carlitos por haber sido los mejores amigos que se pueden tener en una universidad, cada uno ayudándome de alguna forma durante toda la carrera incluyendo el apoyo recibido de su parte en el desarrollo de este trabajo.

A mi novio José David por ser una persona demasiado especial, dándome ánimos en los momentos en que pensaba rendirme, dedicando mucha parte de su tiempo a ayudarme con todo lo que estuvo a su alcance, por soportarme y comprenderme cuando yo estaba extremadamente estresada, supongo que no fue nada fácil.

Al profesor Jean Piero Suárez por haber sido el mejor profesor que tuve en toda la carrera, no sólo por haber compartido sus maravillosos conocimientos de matemática conmigo sino también por aconsejarme de la mejor manera que pudo y ayudarme en la realización de este trabajo dándome consejos que me ayudaron en gran cantidad.

A demás amigos, familiares y profesores que estuvieron presente ¡¡¡GRACIAS!!!

# ÍNDICE GENERAL

**Lista de Tablas** **vii**

**Lista de Figuras** **viii**

**Resumen** **1**

**Introducción** **2**

**1 Mínimos Cuadrados Lineales** **6**

1.1 Definición del Problema. . . . . 6

1.2 Existencia y Unicidad de Soluciones para Problemas de Mínimos Cuadrados Lineales. . . . . 7

1.3 Interpretación Geométrica del Problema de Mínimos Cuadrados. . . . . 10

1.4 Condicionamiento de Problemas de Mínimos Cuadrados. . . . . 12

1.5 Sensibilidad del Problema de Mínimos Cuadrados. . . . . 14

1.6 Métodos Directos para Resolver Problemas de Mínimos Cuadrados Lineales. . . . . 16

1.6.1 Factorización *QR*. . . . . 16

1.6.2 Factorización de Cholesky. . . . . 24

**2 Métodos Iterativos para Resolver Problemas de Mínimos Cuadrados Lineales** **27**

2.1 Gradientes Conjugados (CG). . . . . 28

2.1.1 CGLS. . . . . 31

2.2 LSQR. . . . . 33

2.2.1 El Proceso de Bidiagonalización . . . . . 33

2.2.2 Algoritmo LSQR. . . . . 34

2.2.3 Estimación de Normas. . . . . 38

2.2.4 Criterios de Parada. . . . . 39

2.3 Precondicionamiento. . . . . 40

2.4 PCGLS. . . . . 41

2.5 PLSQR. . . . . 43

**3 Técnicas de Precondicionamiento** **45**

3.1 Precondicionadores Mediante Factorización Incompleta. . . . . 46

3.2 Precondiciondores Jacobi, SOR y SSOR. . . . . 46

3.3 Precondicionadores Basados en Factorización $LU$. . . . . 49

3.3.1 Algoritmo de Factorización $LU$. . . . . 51

3.4 Cholesky Incompleto (IC). . . . . 53

3.5 Factorización $LQ$ Incompleta. . . . . 56

**4 Resultados Numéricos** **59**

**5 Conclusiones y Trabajos a Futuro** **70**

**Bibliografía** **73**

# Lista de Tablas


4.1 Problemas a Resolver. . . . . 60

4.2 Resolución de Problemas sin Precondicionar. . . . . 61

4.3 PCGLS usando SSOR como Técnica de Precondicionamiento ($\omega = 1$). . 63

4.4 Método PCGLS usando Cholesky Incompleto como Técnica de Precondicionamiento ($M = \tilde{L}$). . . . . 64

4.5 Método PCGLS usando $M = \tilde{L}^T$ de la Factorización Incompleta $\tilde{L}\tilde{Q}$ de $A^T$. . . . . 65

4.6 PCGLS y PLSQR Usando $M = A_1$ como Precondicionador Calculando su Factorización $LU$ Incompleta. . . . . 67

# Lista de Figuras

1.1 Interpretación geométrica del problema de mínimos cuadrados . . . . . 11

4.1 Comportamiento del residual relativo para el problema $WELL1033$ (escala logarítmica). . . . . 62

4.2 Comportamiento del residual relativo al resolver el problema *Amc* (escala logarítmica). . . . . 62

4.3 Resumen de iteraciones tomadas por PCGLS para resolver los nueve problemas aplicando los precondicionadores Cholesky incompleto, $LQ$ incompleto y Jacobi (escala logarítmica). . . . . 66

4.4 Número de condición de $AM^{-1}$ para los diversos precondicionadores usados (escala logarítmica). . . . . 68

**Resumen**

En este trabajo se estudian y comparan diferentes técnicas de precondicionamiento para resolver problemas de mínimos cuadrados lineales grandes y dispersos mediante métodos iterativos como LSQR (least square with QR factorization) y CGLS (conjugate gradient for least square). Las técnicas de precondicionamiento usadas son basadas en factorizaciones incompletas de Cholesky, LU, LQ, entre otras. Se presentan tablas y gráficos que muestran los resultados obtenidos de aplicar estas técnicas de precondicionamiento para resolver diversos problemas de mínimos cuadrados lineales provenientes del arte de medir tierras y de discretizar la ecuación de difusión-convección no transistoria mediante el método de Reciprocidad Dual en Multi-Dominios (DRM-MD).

## Introducción

El principal objetivo de esta investigación es resolver numéricamente problemas de mínimos cuadrados lineales de la forma

$$\text{mín}_{x\in\mathbb{R}^n} \|b - Ax\|_2, \tag{1}$$

donde $A$ es una matriz con valores reales de tamaño $m \times n$ ($m \geq n$), grande, dispersa (con muchos valores iguales a cero) y de rango completo. En este caso, resolver el problema consiste en minimizar la norma Euclideana del vector residual $r = b - Ax$.

La importancia de este estudio radica en que actualmente en muchas aplicaciones científicas y de ingeniería, como discretizaciones de ecuaciones de difusión vía métodos DRM-MD [20], simulación en mecánica de fluidos, análisis de datos, procesamiento de imágenes médicas y otros, surgen problemas de mínimos cuadrados lineales grandes que hoy por hoy es posible resolver, gracias a la gran evolución que han tenido los métodos numéricos y su implementación en potentes computadoras, planteando sistemas algebraicos de ecuaciones con varios cientos de miles (a veces de millones) de incógnitas. Muchos de los problemas que surgen en estos casos son difíciles de resolver debido al gran tamaño de la matriz asociada a los problemas y donde comúnmente está mal condicionada.

El problema (1) puede ser resuelto mediante métodos directos como factorizaciones ortogonales $QR$ [10]. Esta técnica fue propuesta por primera vez en el año 1965 por G. Golub y sigue siendo aplicada para resolver problemas de mínimos cuadrados lineales. Una manera alternativa de resolver el problema (1) es mediante el uso de métodos iterativos de Krylov, como por ejemplo, Gradientes Conjugados (CG) desarrollado

por Hestenes y Stiefel [12] en el año 1952 para resolver sistemas simétricos definidos positivos. Este método fue de gran importancia en esta investigación debido a que en la actualidad, CG es sumamente útil para resolver problemas de mínimos cuadrados lineales cuando la matriz asociada al problema es simétrica definida positiva y no presenta un número de condición tan elevado. Otro método iterativo considerado para resolver problemas de mínimos cuadrados grandes lineales es LSQR creado por Paige y Saunders en 1982 [19] y es de gran importancia cuando se necesita resolver problemas que están mal condicionados debido a su estabilidad numérica.

Los métodos directos basados en ortogonalizaciones de la matriz $A$ resultan muy costosos en cuanto a tiempo computacional y almacenamiento si la matriz es muy grande y dispersa, incluso cuando se realiza el proceso de ortogonalización existe una pérdida considerablemente grande en cuanto al patrón de dipersión [5]. Para estos problemas, los métodos iterativos poseen ventajas sobre los métodos directos si se aprovecha la característica dispersa de la matriz $A$. A pesar de esto, cuando la matriz asociada al problema está mal condicionada, los métodos iterativos suelen tardar mucho para lograr encontrar una buena aproximación a la solución, incluso en algunos casos podrían no encontrarla a pesar de hacer una cantidad grande de iteraciones (divergen). Una técnica para mejorar esta situación es precondicionar el problema (1) el cual consiste en transformar el sistema original en otro cuya matriz asociada posea propiedades más favorables en comparación con la matriz del problema original. En el desarrollo de precondicionadores para problemas de mínimos cuadrados lineales se han realizado muy pocos esfuerzos. Posiblemente una de las razones radica en que el problema (1) puede resolverse mediante el sistema de ecuaciones normales $A^TAx = A^Tb$ ya que poseen la misma solución. Sin embargo, el número de condición de la matriz de coeficientes $A^TA$ es el cuadrado del número de condición de la matriz $A$ y como resultado,

el sistema de ecuaciones normales tiende a estar muy mal condicionado si $A$ lo está. En estas situaciones, la técnica de precondicionamiento es de vital importancia para obtener métodos estables numéricamente, sin embargo, obtener un buen precondicionador no siempre es una tarea fácil ya que se requiere que satisfaga varias condiciones como veremos más adelante.

Otro enfoque también importante para este trabajo consiste en encontrar diversas técnicas de precondicionamiento que permitan mejorar el comportamiento de los métodos iterativos CG y LSQR al resolver problemas de mínimos cuadrados lineales grandes y dispersos provenientes de diversas aplicaciones científicas y de ingeniería. Realizar comparaciones entre los resultados de aplicar técnicas de precondicionamiento adecuadas para el sistema (1) y el uso de precondicionadores recomendados para el sistema de ecuaciones normales $A^TAx = A^Tb$.

La estructura de este trabajo consta de cinco capítulos. En el capítulo 1 se encuentra la teoría del problema de mínimos cuadrados lineales, se explica el uso de las ecuaciones normales para encontrar su solución así como también la interpretación geométrica del problema. Seguido de esto se detallan unos de los principales métodos directos que se han implementado para resolver el problema (1) como lo son la factorización $QR$ y factorización de Cholesky de $A$. El el capítulo 2 se explican dos métodos iterativos usados para resolver problemas de mínimos cuadrados lineales como son Gradientes Conjugados (CG) y LSQR. Se detalla en que consiste precondicionar al problema de mínimos cuadrados lineales y se explican las ventajas al momento de aplicar los métodos iterativos CG y LSQR a dicho problema. El capítulo 3 consta de la implementación de varias técnicas de precondicionamiento aplicadas a problemas de mínimos cuadrados como lo son precondicionadores de Jacobi, SOR, SSOR y fac-

torizaciones incompletas $LU$, Cholesky y $LQ$ de la matriz asociada al problema. El capítulo 4 contiene los resultados numéricos obtenidos después de resolver problemas de mínimos cuadrados lineales grandes y dispersos provenientes de varias aplicaciones implementando las técnicas de precondicionamiento tratadas en el capítulo 3. Se muestran tablas comparativas y gráficos para un mejor análisis y entendimiento de los resultados. Finalmente en el capítulo 5 se establecen las conclusiones y los trabajos a futuro.

# CAPÍTULO 1

# Mínimos Cuadrados Lineales

A continuación se presenta el contenido teórico del problema de mínimos cuadrados lineales. Como primera parte se define formalmente el problema a resolver. Una vez realizado esto se presenta el teorema que garantiza la existencia y unicidad de su solución. Luego, se explica el uso del sistema de ecuaciones normales para encontrar la solución al problema planteado y se ilustra la interpretación geométrica del mismo. Por último, se explicarán algunos métodos directos como lo son las factorizaciones $QR$ y $LL^t$ de una matriz, usadas para encontrar la solución al problema de mínimos cuadrados lineales. Para la factorización $QR$ se presentan dos diferentes maneras de realizarla.

## 1.1 Definición del Problema.

Cuando se desea resolver un sistema de ecuaciones lineales de la forma

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

donde $A \in \mathbb{R}^{m\times n}$ es una matriz no cuadrada y/o singular y $b \in \mathbb{R}^m$ un vector, la solución del sistema no siempre existe. En los casos donde $m > n$ se tiene un sistema sobrede-

terminado y este tipo de sistemas típicamente no tiene solución.
En estos casos, lo mejor que se puede esperar es encontrar un vector $x \in \mathbb{R}^n$ tal que haga al vector $Ax$ lo más cercano a $b$ , es decir, encontrar un vector $x$ tal que $||r||=||Ax-b||$ sea mínima. Cuando la norma Euclideana $||.||_2$ es usada, esa solución se refiere a la solución en el sentido de los mínimos cuadrados lineales para el sistema (1.1) .
El problema de mínimos cuadrados lineales se define formalmente de la siguiente manera [5]:

**Definición 1.1** *Dada una matriz real $A \in \mathbb{R}^{m\times n}$ y un vector real $b \in \mathbb{R}^m$, se quiere encontrar un vector $x \in \mathbb{R}^n$ tal que la función $||r(x)||_2=||Ax-b||_2$ sea mínima. Si el problema de mínimos cuadrados posee más de una solución, entonces la única que da la mínima norma Euclideana es llamada la solución de mínima norma.*

El interés de este trabajo está enfocado en estudiar sólo el caso sobredeterminado, es decir, dada una matriz $A \in \mathbb{R}^{m\times n}$ con $m > n$ se desea encontrar un vector $x \in \mathbb{R}^n$ tal que sea solución al problema de minimización

$$\text{mín}_{x\in\mathbb{R}^n} ||b - Ax||_2. \tag{1.2}$$

## 1.2 Existencia y Unicidad de Soluciones para Problemas de Mínimos Cuadrados Lineales.

En esta sección se enunciará el teorema de existencia y unicidad de la solución del problema de mínimos cuadrados lineales (1.2). Antes de hacerlo se dará la siguiente definición.

**Definición 1.2** *Una matriz A de tamaño $m\times n$ $(m > n)$ es de rango completo si y sólo si*

*las n columnas de A son linealmente independientes. En este caso se dice que el rango de A es igual a n (rank(A)=n).*

**Teorema 1.3** *(Existencia y unicidad de solución de mínimos cuadrados lineales): Siempre existe solución para cualquier problema de mínimos cuadrados lineales. La solución es única si y sólo si la matriz de coeficientes asociada al problema es de rango completo. Si dicha matriz no es de rango completo, entonces el problema de mínimos cuadrados posee infinitas soluciones.*

A continuación se dará la demostración de este teorema para el caso en que la matriz asociada al problema es de rango completo ya que es el caso que interesa en este trabajo. La demostración del caso en que la matriz no es de rango completo puede verse en [5]. Antes de demostrar el teorema 1.3 se presentará el siguiente lema y algunas definiciones útiles para la demostración.

**Lema 1** *Sean $A \in \mathbb{R}^{m\times n}$ $(m > n)$ y $b \in \mathbb{R}^m$, entonces $x \in \mathbb{R}^n$ es una solución de mínimos cuadrados lineales para el sistema $Ax = b$ si y sólo si $x$ satisface*

$$A^T Ax = A^T b. \tag{1.3}$$

*Demostración*: Se define $r(x) = b - Ax$. Sea $y \in \mathbb{R}^n$ un vector, entonces

$$r(y) = b - Ay = b - Ax + Ax - Ay = r(x) + A(x - y);$$

así

$$||r(y)||_2^2 = ||r(x)||_2^2 + 2(x - y)^T A^T r(x) + ||A(x - y)||_2^2 \tag{1.4}$$

Se asume que $x$ satisface $A^T Ax = A^T b$, es decir, $A^T r(x) = 0$ luego, por (1.4) se tiene que para un $y \in \mathbb{R}^n$ arbitrario

$$||r(y)||_2^2 = ||r(x)||_2^2 + ||A(x - y)||_2^2 \geq ||r(x)||_2^2$$

esto implica que $x$ es una solución de mínimos cuadrados lineales. Ahora se asume que el vector $x \in \mathbb{R}^n$ es solución del problema de mínimos cuadrados y que $A^T r(x) \neq 0$, es decir, $A^T r(x) = z \neq 0$. Se define un vector $y$ tal que $y = x + cz$, con $c \in \mathbb{R}$ entonces

$$r(y) = r(x) + A(x - y) = r(x) - cAz$$

así

$$\begin{aligned} ||r(y)||_2^2 &= ||r(x)||_2^2 + c^2||Az||_2^2 - 2cz^T A^T r(x) \\ &= ||r(x)||_2^2 + c^2||Az||_2^2 - 2c||z||_2^2, \end{aligned}$$

como $c \in \mathbb{R}$, entonces para valores de $c$ suficientemente pequeños se tendrá

$$||r(y)||_2^2 < ||r(x)||_2^2$$

y así $x$ no es solución de mínimos cuadrados lineales ya que se encontró un vector $y \neq x$ tal que hace $||r(y)||_2^2$ más pequeña pero este hecho contradice lo ya dado por cierto, por lo tanto

$$A^T r(x) = 0 \Rightarrow A^T Ax = A^T b \quad \blacksquare$$

El sistema (1.3) es llamado sistema de ecuaciones normales asociado al problema de mínimos cuadrados (1.2). En este sistema lineal la matriz de coeficientes $A^T A$ es cuadrada de tamaño $n \times n$ y el vector independiente $A^T b$ yace en $\mathbb{R}^n$.

**Definición 1.4** *Una matriz simétrica $A \in \mathbb{R}^{n \times n}$ es **definida positiva** si y sólo si para cada vector $x \in \mathbb{R}^n$ con $x \neq 0$,*

$$x^T Ax > 0.$$

Una propiedad importante que se cumple para este tipo de matrices es que toda matriz definida positiva es invertible y su inversa es también definida positiva.

*Demostración del teorema 1.3 (para el caso de rango completo)*: Sea $A \in \mathbb{R}^{m\times n}$ $m > n$ una matriz de rango completo y $b \in \mathbb{R}^m$. Se sabe por el lema 1 que $x$ es una solución de mínimos cuadrados para el sistema $Ax = b$ sí y sólo sí $x$ satisface $A^TAx = A^Tb$. Se debe demostrar que para el caso de rango completo esta solución es única. Es sabido que la solución del sistema $A^TAx = A^Tb$ es única sí y sólo sí $A^TA$ es no singular, así, todo lo que se necesita verificar es que $A^TA$ es no singular sí y sólo sí $A$ es de rango completo. Para esto se probará un fuerte resultado como lo es que $A^TA$ es definida positiva sí y sólo sí $A$ es de rango completo.
Sea $A \in \mathbb{R}^{m\times n}$ $(m > n)$. Si las columnas de $A$ son linealmente independientes, esto implica que para $x \neq 0$ se tiene $Ax \neq 0$ y por lo tanto $x \neq 0 \Rightarrow \|Ax\|_2^2 = x^TA^TAx > 0$. Por lo tanto $A^TA$ es definida positiva. Por otra parte, supóngase que $A^TA$ es definida positiva pero las columnas de $A$ son linealmente dependientes entonces para algún $x_0 \neq 0$ se tiene que $Ax_0 = 0$ y así $x_0^TA^TAx_0 = 0$ implica que $A^TA$ no es definida positiva lo que contradice el hecho dado por cierto. ■

Como consecuencia de la demostración del teorema 1.3 es notable que pueden ser aplicados al sistema (1.3) diversos métodos eficientes a la hora de resolver problemas simétricos definidos positivos para encontrar soluciones al problema (1.2) los cuales serán detallados en los capítulos siguientes.

## 1.3 Interpretación Geométrica del Problema de Mínimos Cuadrados.

Una manera de observar geométricamente el problema de mínimos cuadrados lineales es la siguiente. Suponga que se tiene una matriz $A$ de $m\times n$ con $m > n$. Es sabido que

$A$ es una aplicación lineal de $\mathbb{R}^n \rightarrow \mathbb{R}^m$. Llámese $R(A)$ al subespacio de $\mathbb{R}^m$ donde cada vector $u \in R(A)$ se puede escribir como $u = Ax$ para algún $x$ en $\mathbb{R}^n$, es decir,

$$R(A) = \{u = Ax | x \in \mathbb{R}^n\}.$$

Sea $b \in \mathbb{R}^m$. Como la norma $\|.\|_2$ es la norma Euclideana, $\|b - Ax\|_2$ es la distancia entre el puntio final de $b$ y $Ax$. Es claro que esa distancia es mínima, sí y sólo sí $b - Ax$ es perpendicular a $R(A)$. Esta interpretación geométrica se ilustra en la figura 1.1.

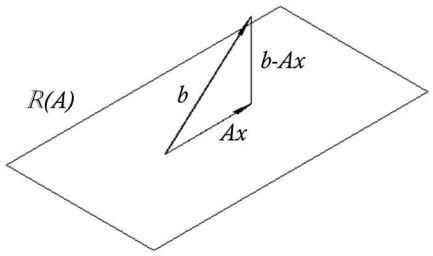


**Figura** 1.1: Interpretación geométrica del problema de mínimos cuadrados

De esta interpretación se puede notar que una solución en el sentido de los mínimos cuadrados para el sistema $Ax = b$ siempre puede encontrarse. Esta solución existe debido a que se puede proyectar el vector $b$ a $R(A)$ para obtener un vector $u \in R(A)$ tal que $u = Ax$ para algún vector $x \in \mathbb{R}^n$. Ese $x$ es una solución. Como $b - Ax$ es perpendicular a $R(A)$ y todo vector en $R(A)$ es una combinación lineal de los vectores columnas de $A$, entonces $b - Ax$ es ortogonal a toda columna de $A$, es decir,

$$A^T(b - Ax) = 0$$

o

$$A^T Ax = A^T b.$$

## 1.4 Condicionamiento de Problemas de Mínimos Cuadrados.

Una propiedad que permite de alguna forma conocer si la aproximación de la solución obtenida a un problema de mínimos cuadrados del tipo (1.2) es buena, lo indica el llamado *condicionamiento del problema*. El condicionamiento de un problema de mínimos cuadrados es una propiedad del problema en sí mismo y se refiere a cómo la solución del problema cambiará si los datos con los que se trabaja han sido alterados. Este estudio es importante ya que los datos usados para resolver un problema de mínimos cuadrados lineales generalmente provienen de observaciones experimentales donde las medidas calculadas pueden ser objeto de perturbaciones. Un ejemplo de esto son los errores que pueden originarse al discretizar ecuaciones diferenciales, etc. Así, al resolver un problema de mínimos cuadrados lineales frecuentemente no se trabaja con los datos originales sino con datos perturbados.

Un problema se dice frecuentemente que está mal condicionado si un pequeño error en los datos produce grandes errores en la solución independientemente del método utilizado para obtener dicha solución. Por el contrario, si al alterar un poco los datos de entrada no produce grandes cambios en la solución obtenida entonces se dice que el problema está bien condicionado.

Usualmente se trata de asociar a un problema de mínimos cuadrados un número llamado *número de condición*. El número de condición indica si el problema está mal o bien condicionado. Para dar una definición algebraica de lo que es el número de condición de un problema de mínimos cuadrados se plantea primero la siguiente definición.

**Definición 1.5** *La matriz* $A^{\dagger} = (A^T A)^{-1} A^T$ *cuando* $A$ *es* $m \times n$ $(m \geq n)$ *y de rango* $n$ *es llamada la* ***pseudoinversa*** *de* $A$.

Esta definición de pseudoinversa generaliza la definición básica de la inversa de una matriz cuadrada $A$. Nótese que cuando la matriz es cuadrada e invertible se tiene que

$$A^{\dagger} = (A^T A)^{-1} A^T = A^{-1} A^{-T} A^T = A^{-1}.$$

A continuación se define el número de condición de una matriz.

**Definición 1.6** *El número de condición de una matriz $A \in \mathbb{R}^{m \times n}$ de rango completo viene dado por*

$$Cond(A) = ||A|| ||A^{\dagger}||.$$

Si $Cond(A)$ es muy grande, entonces se dice que $A$ es una matriz mal condicionada. Por el contrario, si $Cond(A)$ es un número pequeño cercano a uno, se dice que $A$ está bien condicionada.

En la resolución de problemas de mínimos cuadrados lineales, el número de condición de la matriz asociada al problema juega un rol muy importante a la hora de estimar la eficiencia de los resultados cuando se usa un método numérico para resolver el problema. Uno de los principales problemas que se presenta a la hora de buscar la solución en el sentido de los mínimos cuadrados mediante las ecuaciones normales (1.3) es que el número de condición de la matriz $A^T A$ es el cuadrado del número de condición de la matriz $A$ y esto hace que resolver el problema vía ecuaciones normales reciba un estudio más cuidadoso. Este tipo de análisis se detallará más adelante.

## 1.5 Sensibilidad del Problema de Mínimos Cuadrados.

Esta sección se enfoca en el estudio de la sensibilidad de la solución de mínimos cuadrados cuando se realizan perturbaciones en la data. Este estudio es importante para entender los comportamientos de diferentes métodos para resolver problemas del tipo (1.2) que son discutidos en los siguientes capítulos. En este análisis el número de condición de la matriz $A$ juega un rol muy importante.

**Caso 1.** Perturbación en el vector $b$

Supóngase que para el problema (1.2) el vector $b$ ha sido perturbado a $\tilde{b} = b + \delta b$ pero $A$ permanece sin cambios.

**Teorema 1.7** *(Perturbación derecha de mínimos cuadrados): Sean $x$ y $\hat{x}$ la solución única del problema original de mínimos cuadrados y la del problema perturbado respectivamente. Si $\|b_R\| \neq 0$ entonces*

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq Cond(A)\frac{\|\delta b_R\|}{\|b_R\|}$$

*Donde $b_R$ y $\delta b_R$ son las proyecciones del vector $b$ y $\delta b$ al plano $R(A)$ respectivamente.*

Para detalles de la demostración ver [5].

El teorema 1.7 refleja que si sólo el vector $b$ es perturbado entonces, $Cond(A)$ sirve como base para el análisis de sensibilidad de la solución de mínimos cuadrados. Si $Cond(A)$ es muy grande se tiene que, así el error en la proyección de $b$ al plano $R(A)$ sea pequeño, se podrían tener drásticos cambios en la solución de mínimos cuadrados. Por otra parte, si tanto $Cond(A)$ como el error relativo en la proyección de $b$ son pequeños, entonces la solución de mínimos cuadrados no cambiará mucho [5].

**Caso 2.** Perturbación en la matriz $A$

Supóngase ahora que para el problema (1.2) la matriz $A$ es la perturbada. En este caso $\tilde{A} = A + E$ donde $E$ es la perturbación de $A$ lo suficientemente pequeña tal que se tenga que $rank(A) = rank(A + E)$. Sean $x$ y $\hat{x}$ las soluciones del problema original de mínimos cuadrados y del problema perturbado respectivamente. Sean $E_A$ y $E_N$ las proyecciones de $E$ en $R(A)$ y en el complemento ortogonal de $R(A)$ respectivamente. Entonces si $b_R \neq 0$ se tiene el siguiente teorema.

**Teorema 1.8** *(Perturbación izquierda de mínimos cuadrados): Sean $x$ y $\hat{x}$ las soluciónes de mínimos cuadrados para el problema $Ax = b$ y de $(A + E)\hat{x} = b$ respectivamente y sea $rank(A) = rank(A + E)$ entonces*

$$\frac{||\hat{x} - x||}{||x||} \leq 2Cond(A)\epsilon + 4(Cond(A))^2 \frac{||E_N||}{||A||} \frac{||b_N||}{||b_R||} + O\left(||E_N||/||A||\right)^2.$$

*Además*

$$\frac{||\hat{r} - r||_2}{||b||_2} \leq \epsilon(1 + 2Cond(A))$$

*donde* $\epsilon = \frac{||E||}{||A||}, \quad r = b - Ax, \quad \hat{r} = b - (A + E)\hat{x}.$

Para detalles del teorema ver [11]

El teorema 1.8 indica que en el caso donde solo la matriz $A$ es perturbada, la sensibilidad de la solución de mínimos cuadrados generalmente depende del cuadrado del número de condición de $A$.

## 1.6 Métodos Directos para Resolver Problemas de Mínimos Cuadrados Lineales.

En esta sección se detallan algunos métodos directos usados para la resolución de problemas de mínimos cuadrados lineales. Uno de ellos es la factorización $QR$ de la matriz asociada al problema, usada como un método directo en el año 1965 por Golub G. [10]. Se presenta cómo obtener la solución al problema de mínimos cuadrados usando dicha factorización y se plantean maneras diferentes de cómo obtenerla. Otro método directo detallado en esta sección y usado en la resolución de problemas lineales específicamente simétricos definido positivos es la factorización de Cholesky. En esta sección se explicará como puede usarse dicha factorización para obtener la solución de mínimos cuadrados vía ecuaciones normales (1.3) y se presentará el algoritmo que permite la obtención de esta factorización.

### 1.6.1 Factorización $QR$.

Una manera de resolver el problema (1.2) es realizando la factorización $QR$ de la matriz $A \in \mathbb{R}^{m\times n}$. Esta descomposición está definida como

$$A = QR, \tag{1.5}$$

donde la matriz $Q \in \mathbb{R}^{m\times m}$ es ortogonal y $R \in \mathbb{R}^{mxn}$ es triangular superior. El uso de la factorización $QR$ de $A$ para obtener solución al problema (1.2) es el siguiente. Considere la descomposición $QR$ de la matriz $A$. Entonces,

$$Q^T A = R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

con $R_1 \in \mathbb{R}^{n\times n}$. Como el tamaño de un vector es preservado cuando se multiplica por una matriz ortogonal, se tiene

$$\begin{aligned} \|Ax-b\|_2^2 &= \|Q^TAx - Q^Tb\|_2^2 \\ &= \|Rx - Q^Tb\|_2^2 \\ &= \|R_1x - c\|_2^2 + \|d\|_2^2, \end{aligned}$$

donde $Q^Tb = \begin{pmatrix} c \\ d \end{pmatrix}$. Así $\|Ax-b\|_2^2$ será mínima si $x$ es escogido tal que $R_1x - c = 0$ y el correspondiente residual está dado por $\|r\|_2 = \|d\|_2$. Estos cálculos sugieren los siguientes pasos para resolver el problema (1.2) vía factorización $QR$.

1. Descomponer $A_{m\times n} = Q_{m\times m}R_{m\times n}$.
2. Formar $Q^T_{m\times m}b_{m\times 1} = \begin{pmatrix} c \\ d \end{pmatrix}$, $c$ de tamaño $n\times 1$ y $d$ de tamaño $(m-n)\times 1$.
3. Resolver el sistema triangular superior de $n\times n$ $R_1x = c$ donde $R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$.

Existen diferentes maneras de obtener la factorización $QR$ de una matriz como lo son vía matrices de Householder [13], rotaciones de Givens [8] y el proceso de Gram-Schmidt. Estas dos últimas serán detalladas en esta sección ya que serán utilizadas en los capítulos siguientes como herramientas en el uso de métodos iterativos.

**Factorización $QR$ vía Rotaciones de Givens.**

Una matriz de la forma

$$G(i,j,\theta) = \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & i & & j & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array}$$

donde $c^2 + s^2 = 1$, es llamada matriz de Givens o rotación de Givens debido al analista numérico Wallace Givens [8].

En dos dimensiones, la matriz representa una rotación en el plano con un ángulo $\theta$

$$G(\theta) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c = cos(\theta), \quad s = sen(\theta).$$

Nótese que como $c^2+s^2 = 1$, $G(i,j,\theta)G(i,j,\theta)^T = I$, así, la matriz $G(i,j,\theta)$ es ortogonal.

Cuando un vector $x$ en $\mathbb{R}^n$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

es premultiplicado por una matriz de Givens $G(i,j,\theta) \in \mathbb{R}^{n\times n}$, sólo la $j$ -ésima y la $i$-ésima componentes del vector se ven afectadas y se obtiene

$$G(i,j,\theta)x = b = (\beta_1, ..., \beta_n)^T$$

donde $\beta_k = x_k, k \neq i, j$, y

$$\beta_i = cx_i + sx_j, \tag{1.6}$$

$$\beta_j = -sx_i + cx_j. \tag{1.7}$$

Si para las ecuaciones anteriores se tiene

$$c = x_i/\sigma, \quad s = x_j/\sigma, \quad \sigma = \sqrt{x_i^2 + x_j^2} \neq 0$$

entonces

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} x_i \\ x_j \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix},$$

es decir, se ha introducido el valor cero en la $j$-ésima componente del vector $x$.
La premultiplicación de una matriz $A \in \mathbb{R}^{m\times n}$ por una matriz de Givens $G(i, j, \theta) \in \mathbb{R}^{mxm}$ afectará solamente la fila $i$ y $j$ de $A$ las cuales son transformadas de la siguiente manera

$$a_{ik} = ca_{ik} + sa_{jk}, \tag{1.8}$$

$$a_{jk} = -sa_{ik} + ca_{jk}, \quad k = 1, 2, .., n. \tag{1.9}$$

donde $a_{i,j}$ es la entrada $(i, j)$ de $A$. De manera similar, postmultiplicar $A$ por una matriz de Givens $G(i, j, \theta)$ sólo afecta la columna $i$ y $j$.

Las rotaciones de Givens pueden ser utilizadas para calcular la factorización $QR$ de una matriz $A \in \mathbb{R}^{m\times n}$. En esta sección se denotará $G(i, j, \theta)$ simplemente como $G_{i,j}$.
La idea básica es construir matrices $Q_1, Q_2, ..., Q_k$ usando rotaciones de Givens tales que si $A^{(1)} = A$ entonces $Q_1A^{(1)} = A^{(2)}$ tenga ceros debajo de la entrada $(1, 1)$ de $A$ en toda la columna, $Q_2A^{(2)} = A^{(3)}$ tenga ceros debajo de la entrada $(2, 2)$ en la segunda columna y así sucesivamente para obtener

$$A^{(k+1)} = Q_kA^{(k)} \quad k = 1, ..., n.$$

$Q_k$ será escogido para hacer cero los elementos debajo de la diagonal en la $k$-ésima columna de la matriz reducida $A^{(k)}$ por lo tanto la matriz $A^{(k+1)}$ será triangular en las primeras $k$ columnas, es decir, toma la forma

$$A^{(k+1)} = Q_k Q_{k-1} ... Q_1 A = \begin{pmatrix} R_{11} & R_{12} \\ 0 & \tilde{A}^{(k+1)} \end{pmatrix}$$

con $R_{11} \in \mathbb{R}^{k \times k}$ triangular superior.

Por medio de rotaciones de Givens se pueden definir las matrices $Q_i$, $i = 1, ..., n$ como

$$Q_1 = G_{1,m} G_{1,m-1} ... G_{1,2}, \tag{1.10}$$

$$Q_2 = G_{2,m} G_{2,m-1} ... G_{2,3} \tag{1.11}$$

y así sucesivamente, donde cada $Q_i$, definidas como productos de matrices de Givens, son ortogonales. Note que en los pasos anteriores sólo la matriz $\tilde{A}^{(k+1)}$ es transformada y $(R_{11} \;\; R_{12})$ son las primeras $(k-1)$ filas de la matriz final $R$. De esto se sigue que

$$A^{(n+1)} = Q_n ... Q_2 Q_1 A = \begin{pmatrix} R \\ 0 \end{pmatrix},$$

así

$$Q = (Q_n ... Q_2 Q_1)^T \tag{1.12}$$

$$= Q_1 Q_2 ... Q_n. \tag{1.13}$$

Esta factorización $QR$ de $A$ por medio de matrices de Givens puede ser usada para resolver el problema de mínimos cuadrados lineales (1.2) de la forma explicada al inicio de esta sección.

**Algoritmo 1** Factorización $QR$ vía rotaciones de Givens.

**para** k=1,2,...,n **hacer**
  **para** i=k+1,...,m **hacer**
    **si** $a_{ik} = 0$ **entonces**
      $c = 1; s = 0; \sigma = a_{kk}$
    **si no**
      **si** $|a_{ik}| > |a_{kk}|$ **entonces**
        $t = a_{kk}/a_{ik}$
        $tt = \sqrt{1 + t^2}$
        $s = 1/tt; \quad c = ts; \quad \sigma = tta_{ik}$
      **si no**
        $t = a_{ik}/a_{kk}$
        $tt = \sqrt{1 + t^2}$
        $c = 1/tt; \quad s = tc; \quad \sigma = tta_{kk}$
      **fin si**
    **fin si**
    **para** j=k,...,n **hacer**
      $h = ca_{ik} + sa_{ij}$
      $a_{ij} = -sa_{kj} + ca_{ij}$
      $a_{kj} = h$
    **fin para**
  **fin para**
**fin para**

Por otra parte, si $A = QR$ es una factorización $QR$ de $A \in \mathbb{R}^{m \times n}$ entonces a la

descomposición

$$A = Q_1 R_1,$$

donde $Q_1 \in \mathbb{R}^{m\times n}$ es la matriz formada por las primeras $n$ columnas de $Q$ y $R_1 \in \mathbb{R}^{n\times n}$ la matriz formada por las $n$ primeras filas y columnas de $R$, es llamada *factorización QR reducida* de $A$ [11]. El uso de esta factorización para resolver el problema de mínimos cuadrados (1.2) se explicará mas adelante.

**Factorización $QR$ vía Proceso de Gram-Schmidt.**

Una técnica que puede ser usada para conseguir directamente la factorización $QR$ reducida de una matriz $A \in \mathbb{R}^{m\times n}$ se obtiene mediante el proceso llamado Gram-Schmidt. Dada una matriz $A = (a_1, a_2, ..., a_n)$ de orden $m \times n$, de rango completo, donde cada $a_i$, $i = 1, ..., n$ indica la $i$-ésima columna de $A$, el proceso de Gram-Schmidt consiste en construir una matriz $Q = (q_1, q_2, ..., q_n)$ de tamaño $m \times n$ con columnas ortonormales y una matriz triangular superior $R \in \mathbb{R}^{n\times n}$ tal que $A = Q_{m\times n} R_{n\times n}$. El siguiente algoritmo plantea la manera de obtener la factorización $QR$ reducida mediante este proceso.

**Algoritmo 2** Gram-Schmidt Clásico para factorización $QR$.

1: **para** k=1,2,...,n **hacer**
2: **para** i=1,2,...,k-1 **hacer**
3: $R_{ik} = q_i^T a_k$
4: $q_k = a_k - \sum_{i=1}^{k-1} R_{ik} q_i,$
5: $R_{kk} = ||q_k||_2$
6: $q_k = \frac{q_k}{R_{kk}}$
7: **fin para**
8: **fin para**

El algoritmo 2 es conocido por poseer dificultades numéricas. Durante el cálculo

de los vectores $q_k$ pueden aparecer cancelaciones que causen la pérdida de la ortonormalidad de los mismos [5]. Sin embargo, el algoritmo 2 puede ser modificado con el fin de que posea mejores propiedades numéricas. El siguiente algoritmo conocido como Gram-Schmidt modificado calcula la factorización $QR$ reducida de $A$ en un orden diferente, es decir, la matriz $Q$ es obtenida por columna y $R$ por filas (a diferencia del Gram-Schmidt clásico que suministra ambas matrices por columnas).

**Algoritmo 3** Gram-Schmidt Modificado para factorización $QR$.

Sea $q_k = a_k,\ k = 1, ..., n.$
**para** k=1,2,...,n **hacer**
    $R_{kk} = \|q_k\|_2$
    $q_k = \frac{q_k}{R_{kk}}$
    **para** j=k+1,...,n **hacer**
        $R_{kj} = q_k^T q_j,$
        $q_j = q_j - r_{kj} q_k$
    **fin para**
**fin para**

Los algoritmos 2 y 3 son matemáticamente equivalentes pero la diferencia entre ambos radica en que el orden en que se realizan las operaciones es diferente. Esta diferencia logra que el algoritmo Gram-Schmidt modificado sea más estable numéricamente. Para más detalles ver [5].

Como se dijo anteriormente, la factorización $QR$ reducida de $A$ también puede ser usada para resolver el problema de mínimos cuadrados lineales (1.2). Para este caso, es recomendado calcular mediante el algoritmo 3 la factorización $QR$ reducida de la

matriz extendida $(A \;\; b)$ para así obtener una expresión de la forma

$$(A \;\; b) = (Q_1 \;\; q_{n+1}) \begin{pmatrix} R_1 & z \\ 0 & \rho \end{pmatrix},$$

donde $\rho = \|q_{n+1}\|$ y $z = (z_1, z_2, ..., z_n)^T$ con $z_i = q_i^T q_{n+1}$ para $i = 1, 2, ..., n$. De esto se sigue que

$$\begin{aligned} Ax - b &= (A \;\; b) \begin{pmatrix} x \\ -1 \end{pmatrix} && (1.14) \\ &= (Q_1 \;\; q_{n+1}) \begin{pmatrix} R_1 & z \\ 0 & \rho \end{pmatrix} \begin{pmatrix} x \\ -1 \end{pmatrix} && (1.15) \\ &= Q_1(R_1 x - z) - \rho q_{n+1}. && (1.16) \end{aligned}$$

Así $\|Ax - b\|_2$ será mínima cuando $R_1 x = z$, luego la solución al problema (1.2) puede ser obtenida resolviendo

$$Rx = z$$

y el vector residual viene dado por $r = \rho q_{n+1}$. Para más detalles ver [5] y [1].

### 1.6.2 Factorización de Cholesky.

Sea $A \in \mathbb{R}^{nxn}$ una matriz cuadrada simétrica definida positiva, entonces la descomposición de la forma

$$A = LL^T$$

donde $L \in \mathbb{R}^{n \times n}$ es una matriz triangular inferior con elementos diagonales positivos es llamada factorización de Cholesky de $A$ debido al ingeniero francés Cholesky (1875-1918) y $L$ es llamado el factor de Cholesky de $A$.

Un método para calcular la factorización de Cholesky de $A$ surge de las siguientes

relaciones. Sea

$$A = LL^T$$

$$\begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & ... & 0 \\ l_{21} & l_{22} & 0... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & ... & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & ... & l_{n1} \\ 0 & l_{22} & ... & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & l_{nn} \end{pmatrix}$$

se tiene entonces

$$l_{11} = \sqrt{a_{11}}, \quad l_{i1} = \frac{a_{1i}}{l_{11}}, \ i = 2, ..., n.$$

$$\sum_{k=1}^{i} l_{ik}^2 = a_{ii}, \qquad a_{ij} = \sum_{k=1}^{j} l_{ik} l_{jk}, \quad j \leq i.$$

Esto conduce al siguiente algoritmo.

**Algoritmo 4** Factorización de Cholesky

1: **para** k=1,2,...,n **hacer**
2: **para** i=1,2,...,k-1 **hacer**
3: $l_{ki} = \frac{1}{l_{ii}}(a_{ki} - \sum_{j=1}^{i-1} l_{ij} l_{kj})$
4: $l_{ii} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$
5: **fin para**
6: **fin para**

En el algoritmo 4 el factor de Cholesky $L$ se obtiene por filas. Existen diferentes maneras de obtener el factor $L$. Para detalles de estas variantes ver [1].

Ahora considérese el caso en que se desea resolver el problema de mínimos cuadrados lineales (1.2). Como se dijo anteriormente, resolver el problema (1.2) es equivalente a encontrar solución al sistema de ecuaciones normales (1.3). Para el caso donde la matriz $A$ es de rango completo es sabido que el sistema (1.3) es cuadrado simétrico

positivo definido (verificado en la demostración del teorema 1.3) y, por lo tanto, puede ser aplicada la factorización de Cholesky a la matriz $A^TA$. Así $A^TA$ puede ser escrita de la forma

$$A^TA = LL^T.$$

Usando esta factorización para obtener la solución del problema (1.3) se obtiene lo siguiente:

$$A^TAx = A^Tb \Rightarrow LL^Tx = A^Tb,$$

sea $d = A^Tb$ entonces $LL^Tx = d$. Esto implica que para obtener $x$ solo se necesita resolver dos sistemas lineales de la forma

$$Lz = d, \qquad L^Tx = z.$$

# CAPÍTULO 2

# Métodos Iterativos para Resolver Problemas de Mínimos Cuadrados Lineales

Para la solución a problemas de mínimos cuadrados lineales se usarán dos metodos iterativos los cuales se describirán con detalle en el presente capítulo. Inicialmente se estudia el método Gradientes Conjugados (CG). Se detallará su construcción y se explicará cómo puede implementarse para obtener solución a problemas de mínimos cuadrados lineales resolviendo el sistema de ecuaciones normales (1.3). A continuación se explica el método LSQR dándose detalles de la construcción del mismo y su uso en la búsqueda de soluciones a problemas de mínimos cuadrados. Posteriormente se presentan sus versiones precondicionadas. Seguido de esto se define en que consiste precondicionar un problema de mínimos cuadrados lineales. La principal razón de hacer esto es mejorar las condiciones del mismo mediante una transformación de variables

para que el nuevo sistema posea propiedades más favorables y presente ventajas a la hora de implementar métodos iterativos para resolverlos.

## 2.1 Gradientes Conjugados (CG).

Gradientes Conjugados o CG por sus siglas en inglés, es un método iterativo originalmente propuesto por Hestenes y Stiefel [12] en 1952 como un método directo para resolver sistemas lineales de la forma

$$Ax = b \tag{2.1}$$

donde $A \in \mathbb{R}^{n\times n}$ es una matriz cuadrada, simétrica definida positiva y $b \in \mathbb{R}^n$.
En la actualidad CG es frecuentemente usado en la resolución de sistemas simétricos definidos positivos donde la matriz asociada al sistema es grande y dispersa. El método es basado en el siguiente resultado de optimización [17].

**Teorema 2.1** *Si $A$ es una matriz real simétrica definida positiva, entonces resolver (2.1) es equivalente a minimizar la funcional cuadrática*

$$q(x) = \frac{1}{2}x^T Ax - x^T b. \tag{2.2}$$

Sea $x^*$ la solución exacta de (2.1). En este método iterativo, se calcula una sucesión de vectores $x_k$, aproximaciones al vector solución $x^*$, de la siguiente manera:

$$x_{k+1} = x_k + \alpha_k p_k, \quad k = 0, ..., n-1. \tag{2.3}$$

donde los vectores $\{p_k\}$ se usan como direcciones, $x_0$ es un estimado inicial de $x^*$ y los escalares $\alpha_k$ son escogidos para minimizar $q(x)$ en las direcciones de $p_k$; esto es, los $\alpha_k$ son seleccionados para que la función $q_\alpha(x_k + \alpha p_k)$ sea mínima. Se puede demostrar

que los vectores residuales son ortogonales entre sí y satisfacen la siguiente recurrencia

$$r_{k+1} = r_k - \alpha_k A p_k, \quad k = 0, ..., n-1; \tag{2.4}$$

así, el valor mínimo de $q_\alpha(x_k + \alpha p_k)$ se alcanza tomando $\alpha_k$ como

$$\alpha_k = \frac{r_k^T r_k}{r_k^T A p_k}, \quad donde\ r_k = b - A x_k.$$

El Algoritmo CG genera automáticamente vectores direccionales $p_k$ que son combinación lineal de $r_k$ y del vector anterior $p_{k-1}$ (ver [11]) expresándose de la siguiente manera

$$p_{k+1} = r_{k+1} + \beta_k p_k, \quad k = 0, ..., n-1. \tag{2.5}$$

Se puede demostrar que estos vectores cumplen con la propiedad

$$p_j^T A p_k = 0 \ \ para \ \ i \leq k \leq j-1, \tag{2.6}$$

esto es, la sucesión de vectores $p_k$ son $A$-ortogonales.

De las expresiones (2.5) y (2.6) se deduce que $\alpha_k$ puede reescribirse de la forma

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}. \tag{2.7}$$

y además

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}. \tag{2.8}$$

La versión eficiente del algoritmo de CG es mostrado en el algoritmo 4.

**Algoritmo 5** Gradientes Conjugados

1: Escoger $x_0$ y calcular $r_0 = b - Ax_0, \quad p_0 = r_0$

2: **para** j = 0,1,..., hasta la convergencia **hacer**

3: $\alpha_j = r_j^T r_j / p_j^T A p_j$

4: $x_{j+1} = x_j + \alpha_j p_j$

5: $r_{j+1} = r_j - \alpha_j A p_j$

6: $\beta_j = r_{j+1}^T r_{j+1} / r_j^T r_j$

7: $p_{j+1} = r_{j+1} + \beta_j p_j$

8: **fin para**

Hestenes y Stiefel [12] indican que el método CG termina en a lo sumo $n$ pasos si no existe presencia de errores de redondeo. La norma $\|x_k - x^*\|$ con $x_k$ la aproximación de la solución $x^*$ en el paso $k$ decrece monótonamente al igual que $\|r_k - r\|$ donde $r_k$ es el residual calculado en el paso $k$. En el libro de Ortega [18] se puede observar la demostración del siguiente teorema.

**Teorema 2.2**

$$\|x^* - x_k\|_2 < \|x^* - x_{k-1}\|_2, \quad k = 1, ..., n$$

*donde $x^*$ es la solución exacta y $x_k$ la aproximación de $x^*$ en el paso k.*

A pesar de que el resultado del teorema anterior se mantiene al aplicar el algoritmo CG, la convergencia del método puede ser muy lenta si el número de condición de la matriz $A$ es muy alto. Esto se evidencia en el siguiente resultado.

**Teorema 2.3** *El k-ésimo iterado del método CG satisface*

$$\|x_k - x^*\|_A \leq 2\left(\frac{\sqrt{c} - 1}{\sqrt{c} + 1}\right)^k \|x_0 - x^*\|_A, \quad k = 1, ..., n$$

*donde* $c = Cond(A)$*,* $x_0$ *es un iterado inicial,* $x^*$ *la solución exacta,* $x_k$ *la aproximación de* $x^*$ *en el paso k y* $\|x^*\|_A = \sqrt{x^{*T}Ax^*}$.

Ver demostración en [15]

También se puede observar del teorema 2.3 que si $Cond(A) \approx 1$, entonces la convergencia del método es muy rápida.

### 2.1.1 CGLS.

Una manera de resolver el problema (1.2) es aplicar el algoritmo CG al sistema de ecuaciones normales (1.3) ya que éste es simétrico definido positivo cuando $A$ es de rango completo como se observó en el capítulo 1. Cuando CG se aplica al sistema (1.3), el algoritmo resultante frecuentemente recibe el nombre de CGLS por sus siglas en inglés. Hestenes y Stiefel en el año 1952 [12] dieron la versión del método de CG para la solución de sistemas de ecuaciones normales.

Aplicando las ecuaciones (2.3) hasta la (2.8) al sistema (1.3) se obtienen las siguientes iteraciones

$$\begin{aligned} r_0 = b - Ax_0, \quad p_0 = A^T r_0, \\ \alpha_k = \frac{\|A^T r_k\|^2}{\|Ap_k\|^2} \\ x_{k+1} = x_k + \alpha_k p_k \\ r_{k+1} = r_k - \alpha_k A p_k \\ \beta_k = \frac{\|A^T r_{k+1}\|^2}{\|A^T r_k\|^2} \\ p_{k+1} = A^T r_{k+1} + \beta_k p_k \end{aligned} \tag{2.9}$$

Aunque aplicar el algoritmo 5 al sistema (1.3) es matemáticamente equivalente a aplicar las ecuaciones anteriores, el sistema (2.9) posee mucha más ventaja ya que no es necesario presentar el producto de matrices $A^TA$, lo cual genera muchos errores de redondeo, sino que se utiliza solamente la matriz $A$ original.

En el algoritmo CGLS, la norma del residual obtenido en el paso $k$, $\|r_k\|$, decrece al igual que en CG [1], más aún

$$\|r - r_k\|_2 \leq 2\left(\frac{\sqrt{c}-1}{\sqrt{c}+1}\right)^k \|r_0 - r\|_2, \tag{2.10}$$

donde $r_0$ es el residual inicial. La estimación del teorema 2.3 se sigue manteniendo pero en este caso $c = Cond(A^TA)$. Para CGLS se puede verificar que $\|A^Tr_k\|$ frecuentemente presentará grandes oscilaciones cuando $Cond(A)$ sea muy grande.

**Algoritmo 6** CGLS

1: Calcular $r_0 = b - Ax_0,\ p_0 = s_0 = A^Tr_0,\ \gamma_0 = \|s_0\|_2^2$
2: **para** $k = 0, 1, 2, ..., mientras\ \gamma_k > tol$ **hacer**
3: $q_k = Ap_k,$
4: $\alpha_k = \gamma_k / \|q_k\|_2^2$
5: $x_{k+1} = x_k + \alpha_k p_k$
6: $r_{k+1} = r_k - \alpha_k q_k$
7: $s_{k+1} = A^T r_{k+1}$
8: $\gamma_{k+1} = \|s_{k+1}\|_2^2$
9: $\beta_k = \gamma_{k+1}/\gamma_k,$
10: $p_{k+1} = s_{k+1} + \beta_k p_k$
11: **fin para**

## 2.2 LSQR.

Se presenta ahora el método LSQR el cual es un procedimiento iterativo que genera una sucesión de vectores los cuales son aproximaciones de la solución $x^*$ del problema (1.1) o (1.2). El método LSQR es similar a CG al momento de resolver los problemas de mínimos cuadrados lineales, pero LSQR posee propiedades numéricas más favorables [19].

El algoritmo LSQR está basado en el proceso de bidiagonalización de Golub y Kahan [9]. Esto genera una secuencia de aproximaciones $\{x_k\}$ tales que en el paso $k$ la norma del residual $\|r_k\|_2$ decrece monótonamente. Analíticamente, la secuencia $\{x_k\}$ es idéntica a la secuencia generada por el algoritmo CG.

### 2.2.1 El Proceso de Bidiagonalización

La finalidad del proceso de bidiagonalización de Golub y Kahan [9] es escribir una matriz $A \in \mathbb{R}^{m\times n}$, de la forma $A = UBV^T$ donde $U \in \mathbb{R}^{m\times(n+1)}$ y $V \in \mathbb{R}^{n\times n}$ son matrices ortogonales y $B \in \mathbb{R}^{(n+1)\times n}$ es una matriz bidiagonal inferior. Este proceso consiste en lo siguiente.

De la relación $AV = UB$,

$$A\begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & \cdots & u_{n+1} \end{bmatrix} \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_n \\ & & & \beta_{n+1} \end{bmatrix}$$

Se tiene que

$$Av_i = \alpha_i u_i + \beta_{i+1} u_{i+1}, \quad para \;\; i = 1, ..., n. \tag{2.11}$$

Por otra parte, de la relación $A^T U = VB^T$ se tiene que

$$A^T \begin{bmatrix} u_1 & u_2 & \cdots & u_{n+1} \end{bmatrix} = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ & \alpha_2 & \beta_3 & & \\ & & \ddots & \ddots & \\ & & & \alpha_n & \beta_{n+1} \end{bmatrix}$$

así

$$A^T u_i = \beta_i v_{i-1} + \alpha_i v_i, \quad para \;\; i = 1, ..., n+1, \tag{2.12}$$

considerando $\alpha_{n+1} = 0$ y $v_0 = 0$.

Los escalares $\alpha_i \geq 0$ y $\beta_i \geq 0$ son escogidos tales que $\|u_i\| = \|v_i\| = 1$. Comenzando el proceso de bidiagonalización con un vector arbitrario $v_1 \in \mathbb{R}^n$, $\|v_1\| = 1$, las ecuaciones (2.11) y (2.12) pueden ser usadas recursivamente para generar los vectores $u_1, v_2, u_2, ..., v_n, u_n, u_{n+1}$. El proceso de bidiagonalización termina cuando $\beta_j = 0$ o $\alpha_j = 0$ para algún $j \leq n$ [19].

### 2.2.2 Algoritmo LSQR.

Para resolver el problema (1.2) se comienza bidiagonalizando la matriz $A$. Paige y Saunders [19] proponen comenzar el algoritmo de bidiagonalización con un vector inicial $u_1 \in \mathbb{R}^m$ tal que

$$\beta_1 u_1 = b \tag{2.13}$$

Realizando este procedimiento se tendrá en el paso $k$

$$V_k \equiv \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix}, \;\; U_{k+1} \equiv \begin{bmatrix} u_1 & u_2 & \cdots & u_{k+1} \end{bmatrix}, \;\; B_k \equiv \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}$$

donde $B_k \in \mathbb{R}^{(k+1)\times k}$ es una matriz rectangular.

Las relaciones de recurrencia (2.11), (2.12) y (2.13) se pueden reescribir en forma matricial de la siguiente manera:

$$U_{k+1}(\beta_1 e_1) = b, \tag{2.14}$$

$$AV_k = U_{k+1}B_k, \tag{2.15}$$

$$A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T, \tag{2.16}$$

Donde las matrices $U_{k+1}$ y $V_k$ cumplen con la relación $U_{k+1}^T U_{k+1} = I$ y $V_k^T V_k = I$.

Sean los siguientes vectores

$$x_k = V_k y_k, \tag{2.17}$$

$$r_k = b - Ax_k, \tag{2.18}$$

$$t_{k+1} = \beta_1 e_1 - B_k y_k \tag{2.19}$$

definidos en términos de algún vector $y_k$, entonces de las ecuaciones (2.14) y (2.15) se tiene que

$$r_k = U_{k+1} t_{k+1}. \tag{2.20}$$

Debido a que se busca que $\|r_k\|_2$ sea pequeña y ya que $U_{k+1}$ es teóricamente ortonormal, esto inmediatamente sugiere escoger un $y_k$ para minimizar $\|t_{k+1}\|_2$. Por lo tanto se llega naturalmente al problema de mínimos cuadrados

$$\text{mín}_y \|\beta_1 e_1 - B_k y_k\|_2. \tag{2.21}$$

Como $B_k$ es una matriz bidiagonal, tiene mucha ventaja desde el punto de vista computacional resolver el subproblema (2.21) usando la factorización QR de $B_k$ vía matrices de Givens descrita en el capítulo 1,

$$Q_k B_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix} \quad y \quad Q_k(\beta_1 e_1) = \begin{bmatrix} f_k \\ \bar{\phi}_{k+1} \end{bmatrix}, \tag{2.22}$$

donde $R_k$ es una matriz bidiagonal superior,

$$R_k = \begin{bmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \ddots & \ddots & \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \end{bmatrix}, \qquad f_k = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \\ \phi_k \end{bmatrix},$$

y $Q_k$ es el producto de rotaciones del plano $Q_k \equiv G_{k,k+1}...G_{2,3}G_{1,2}$ escogidos para eliminar los elementos $\beta_2, ..., \beta_{k+1}$ de $B_k$. El vector solución $y_k$ y el residual $t_{k+1}$ pueden ser obtenidos de

$$R_k y_k = f_k, \tag{2.23}$$

$$t_{k+1} = Q_k^T \begin{pmatrix} 0 \\ \bar{\phi}_{k+1} \end{pmatrix}. \tag{2.24}$$

La factorizacón $Q_k B_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix}$ no necesita ser calculada en cada paso. La relación de recurrencia se puede generar de la siguiente manera. Se asume que se tiene calculada la factorización de $B_{k-1}$. En el siguiente paso, la $k$-ésima columna es agregada y una matriz de rotación $Q_k = G_{k,k+1}Q_{k-1}$ es calculada tal que

$$G_{k,k+1}G_{k-1,k} \begin{pmatrix} 0 \\ \alpha_k \\ \beta_{k+1} \end{pmatrix} = \begin{pmatrix} \theta_k \\ \rho_k \\ 0 \end{pmatrix}, \quad G_{k,k+1} \begin{pmatrix} \bar{\phi}_k \\ 0 \end{pmatrix} = \begin{pmatrix} \phi_k \\ \bar{\phi}_{k+1} \end{pmatrix}.$$

Como proponen Paige y Saunders [19], una manera de combinar las ecuaciones (2.17) y (2.23) eficientemente es mediante la relación

$$x_k = V_k R_k^{-1} f_k \equiv D_k f_k,$$

donde $D_k$ satisface el sistema triangular inferior $R_k^T D_k^T = V_k^T$ y las columnas de la matriz $D_k = (d_1, d_2, ..., d_k)$ pueden ser calculadas sucesivamente mediante la sustitución $d_0 = x_0 = 0$ dando como resultado

$$d_k = \frac{1}{\rho_k}(v_k - \theta_k d_{k-1}), \tag{2.25}$$

$$x_k = x_{k-1} + \phi_k d_k. \tag{2.26}$$

Algunos cálculos en (2.25) pueden ser eliminados usando los vectores $w_k \equiv \rho_k d_k$ en vez de $d_k$.

Resumiendo, con las fórmulas desarrolladas con anterioridad se deriva el algoritmo dado por Paige y Saunders [19] en 1973. Matemáticamente LSQR genera la misma secuencia de aproximaciones $x_k$ que CGLS. Los vectores $v_{i+1}$ y $d_i$ en LSQR son proporcionales a $s_i$ y $p_i$ en CGLS, respectivamente. Ambos requieren acceso a la matriz $A$ sólo para realizar los productos matriz-vector $Av_k$ y $A^T u_k$, sin embargo, se ha visto [19] que LSQR es más estable numéricamente al momento en que se deben realizar muchas iteraciones y cuando $A$ está mal condicionada.

**Algoritmo 7** LSQR

Inicializar: $\beta_1 u_1 = b,\ \alpha_1 v_1 = A^T u_1,\ w_1 = v_1,\ x_0 = 0,\ \bar{\phi} = \beta_1,\ \bar{\rho_1} = \alpha_1$

**para** $i = 1, 2, 3, ...$ hasta la convergencia **hacer**

$\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i$

$\alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$

Construir y aplicar la siguiente transformación ortogonal.

$\rho_i = (\bar{\rho}_i^2 + \beta_{i+1}^2)^{1/2}$

$c_i = \bar{\rho}_i / \rho_i$

$s_i = \beta_{i+1} / \rho_i$

$\theta_{i+1} = s_i \alpha_{i+1}$

$\bar{\rho}_{i+1} = -c_i \alpha_{i+1}$

$\phi_i = c_i \bar{\phi}_i$

$\bar{\phi}_{i+1} = s_i \bar{\phi}_i$

Calcular $x$ y $w$.

$x_i = x_{i-1} + (\phi / \rho) w_i$

$w_{i+1} = v_{i+1} - (\theta_{i+1} / \rho_i) w_i$

**fin para**

### 2.2.3 Estimación de Normas.

En problemas de mínimos cuadrados es importante saber el estimado de $\|A^T r_k\|$ para determinar criterios de convergencia. De las ecuaciones (2.20) y (2.24) se tiene que

$$r_k = \bar{\phi}_{k+1} U_{k+1} Q_k^T e_{k+1}, \tag{2.27}$$

Luego, de (2.27) , (2.16) y (2.22) se tiene

$$\begin{aligned} A^T r_k &= \bar{\phi}_{k+1}(V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T) Q_k^T e_{k+1} \\ &= \bar{\phi}_{k+1} V_k [R_k^T \quad 0] e_{k+1} + r_k = \bar{\phi}_{k+1} \alpha_{k+1} (e_{k+1}^T Q_k^T e_{k+1}) v_{k+1} \end{aligned}$$

El primer término desaparece y además el elemento diagonal $(k+1)$ de $Q_k$ es $-c_k$. Así se tiene

$$A^T r_k = -(\bar{\phi}_{k+1} \alpha_{k+1} c_k) v_{k+1} \tag{2.28}$$

y

$$||A^T r_k|| = \bar{\phi}_{k+1} \alpha_{k+1} |c_k|.$$

### 2.2.4 Criterios de Parada.

Paige y Saunders [19] proponen un criterio de parada para LSQR determinado por una cantidad denominada ATOL que debe ser dato de entrada del algoritmo LSQR. El criterio es el siguiente

$$Parar\ si\ \ \frac{||A^T r_k||}{||A||||r_k||} \leq ATOL. \tag{2.29}$$

La justificación de este criterio es el siguiente. Stewart [**?**] observó que si $r_k = b - Ax_k$ y $\tilde{r}_k = b - (A + E_k)x_k$ donde

$$E_k = -\frac{r_k r_k^T A}{||r_k||^2},$$

entonces $(A + E_k)^T \tilde{r}_k = 0$, así $x_k$ y $\tilde{r}_k$ son, respectivamente, la solución exacta y el residual para el sistema con $A$ perturbado . Como $||E_k||_2 = ||A^T r_k||/||r_k||$, la perturbación a $A$ será insignificante si (2.29) se satisface.

En particular, para LSQR sucede que $E_k x_k = 0$. Por lo tanto $\tilde{r}_k = r_k$ y además $x_k$ y $r_k$ son exactos para el sistema perturbado. Esto fortalece el argumento para el uso de la regla (2.29). Este criterio es suficiente para garantizar que $x_k$ es una solución aceptable del problema de mínimos cuadrados [19].

## 2.3 Precondicionamiento.

Precondicionar un sistema de ecuaciones lineales consiste en transformar el sistema original en otro el cual posea la misma solución pero sea mas rentable resolverlo desde el punto de vista computacional con algún método iterativo. Ciertos precondicionadores necesitan una pequeña fase de construcción pero otros pueden necesitar un trabajo más elaborado.

Para el problema (1.2), precondicionar es equivalente a la siguiente transformación.

Sea $M \in \mathbb{R}^{nxn}$ una matriz no singular y considérese ahora el problema

$$\text{mín}_{y\in\mathbb{R}^n} \left\|(AM^{-1})y - b\right\|_2, \quad Mx = y. \tag{2.30}$$

Si $M$ es escogido tal que $AM^{-1}$ posea un número de condición menor que el de $A$, entonces $M$ mejorará la convergencia de un método iterativo aplicado al sistema (2.30). El sistema (2.30) es llamado *sistema precondicionamiento por la derecha* y es recomendable para los casos en que la matriz posea más filas que columnas.

Es importante notar que el producto de matrices $AM^{-1}$ no debe ser formado explícitamente, sino tratado como un producto de dos operadores. En un método iterativo precondicionado por la matriz $M$ ocurrirán los productos matriz-vector de la forma $AM^{-1}y$ y $M^{-T}A^Tr$. Así, el costo extra de precondicionar estará en resolver sistemas lineales de la forma $Mx = y$ y $M^Tq = s$. Por lo tanto, $M$ se escoge tal que estos sistemas sean fáciles de resolver.

En resumen, un buen precondicionador $M$ debería cumplir con las siguientes propiedades:

- $AM^{-1}$ debería estar mejor condicionada que $A$.
- $M$ debe tener aproximadamente el mismo número de elementos distintos de cero que $A$.

- Debería ser poco costoso computacionalmente resolver ecuaciones con matrices $M$ y $M^T$.

Al precondicionar un problema del tipo (1.2) se deben estudiar los costos y los beneficios que esto implica; es decir, el costo de construir y aplicar un precondicionador y la ganancia de rapidez en la convergencia de los métodos iterativos aplicados para obtener soluciones, considerando que la cantidad de cálculos requeridos por iteracion para resolver el problema (2.30) no debería ser considerablemente mayor que la necesitada para resolver (1.2). Existe otra manera de precondicionar el sistema (1.2) llamada *precondicionar por la izquierda* y consiste en transformar este sistema en uno de la forma

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

donde $M \in \mathbb{R}^{m\times m}$. Esta manera de precondicionar no será aplicada en este trabajo ya que, como se dijo antriormente, los problemas a solucionar serán problemas sobredeterminados con $A \in \mathbb{R}^{m\times n}$ y cuando $m$ es mucho mayor que $n$, esta forma de precondicionar no es recomendable ya que el tamaño del precondicionador será mucho más grande necesitando más cálculos tanto para construir el precondicionador como al usarlo para resolver los problemas lo que trae como consecuencia un costoso computacional mucho mayor.

## 2.4 PCGLS.

En esta sección se estudiará lo que sucede con el algoritmo CGLS cuando es aplicado para resolver el problema (2.30). Es sabido que el sistema de ecuaciones normales asociado al problema (2.30) viene dado por:

$$M^{-T}A^TAM^{-1}y = M^{-T}A^Tb \tag{2.31}$$

Al aplicar el algoritmo CGLS al problema (2.31), el error en el paso $k$ $\|r - r^{(k)}\|_2$ se sigue minimizando. Por lo tanto, para la rata de convergencia se tiene que la estimación del residual en el paso $k$ cumple con la condición

$$\|r - r^{(k)}\|_2 < 2\left(\frac{\sqrt{c}-1}{\sqrt{c}+1}\right)^k \|r - r^{(0)}\|_2, \quad c = Cond(M^{-T}A^TAM^{-1}) \tag{2.32}$$

Donde $r_0$ es el residual inicial. Es conveniente formular el método precondicionado en términos de la variable original $x$. Para el método gradientes conjugados precondicionado (PGCLS) se obtiene el siguiente algoritmo.

**Algoritmo 8** PGCLS

1: Calcular $r_0 = b - Ax_0,\ p_0 = s_0 = M^{-T}(A^T r_0),\ \gamma_0 = \|s_0\|_2^2$
2: **para** $k = 0, 1, 2, ..., hasta\ que\ \gamma_k \leq tol$ **hacer**
3: $t_k = S^{-1}p_k,$
4: $q_k = At_k,$
5: $\alpha_k = \gamma_k / \|q_k\|_2^2,$
6: $x_{k+1} = x_k + \alpha_k t_k,$
7: $r_{k+1} = r_k - \alpha_k q_k,$
8: $s_{k+1} = S^{-T}(A^T r_{k+1}),$
9: $\gamma_{k+1} = \|s_{k+1}\|_2^2,$
10: $\beta_k = \gamma_{k+1}/\gamma_k,$
11: $p_{k+1} = s_{k+1} + \beta_k p_k$
12: **fin para**

## 2.5 PLSQR.

En este caso se desea resolver el problema (2.30). Para esto se denota $C = AM^{-1}$. Entoces, similar a como se desarrolló en la sección 2.3.1 se obtienen las siguientes relaciones

$$Cv_i = \alpha_i u_i + \beta_{i+1} u_{i+1}, \tag{2.33}$$

$$C^T u_{i+1} = \beta_{i+1} v_i + \alpha_{i+1} v_{i+1}. \tag{2.34}$$

para obtener la factorización $C = UBV^T$. Sustituyendo esta factorización en (2.30) se llega al subproblema de mínimos cuadrados

$$\|Bz - \beta_1 e_1\|_2$$

donde $y = Vz \;\; ; \;\; x = My$.

De esta forma, el algoritmo PLSQR queda de la siguiente manera.

**Algoritmo 9** PLSQR

1: Inicializar: $\beta_1 u_1 = b,\ \alpha_1 v_1 = M^{-T} A^T u_1,\ w_1 = v_1,\ x_0 = 0,\ \bar{\phi} = \beta_1,\ \bar{\rho_1} = \alpha_1$

2: **para** $i = 1, 2, 3, ...$ hasta la convergencia **hacer**

3: $\quad Resolver\ \ Mh = v_i$

4: $\quad \beta_{i+1} u_{i+1} = Ah - \alpha_i u_i$

5: $\quad Resolver\ \ M^T g = A^T u_{i+1}$

6: $\quad \alpha_{i+1} v_{i+1} = g - \beta_{i+1} v_i$

7: $\quad \rho_i = (\bar{\rho}_i^2 + \beta_{i+1}^2)^{1/2}$

8: $\quad c_i = \bar{\rho}_i / \rho_i$

9: $\quad s_i = \beta_{i+1} / \rho_i$

10: $\quad \theta_{i+1} = s_i \alpha_{i+1}$

11: $\quad \bar{\rho}_{i+1} = -c_i \alpha_{i+1}$

12: $\quad \phi_i = c_i \bar{\phi}_i$

13: $\quad \bar{\phi}_{i+1} = s_i \bar{\phi}_i$

14: $\quad x_i = x_{i-1} + (\phi / \rho) w_i$

15: $\quad w_{i+1} = v_{i+1} - (\theta_{i+1} / \rho_i) w_i$

16: **fin para**

# CAPÍTULO 3

# Técnicas de Precondicionamiento

Este capítulo trata acerca de algunas técnicas de precondicionamiento que se aplican para mejorar la condición de un problema de mínimos cuadrados lineales con la finalidad de obtener soluciones óptimas en cuanto a exactitud y tiempo al usar métodos iterativos. Primero se define en que consiste la factorización incompleta de una matriz, luego se detalla el uso de precondicionadores de Jacobi, SOR y SSOR para resolver problemas de mínimos cuadrados lineales. A continuación se plantea un precondicionador basado en la factorización $LU$ incompleta de una matriz. Seguido de esto, se explica el uso de la factorización incompleta de Cholesky y la descomposición $LQ$ incompleta de una matriz como técnicas de precondicionamiento.

## 3.1 Precondicionadores Mediante Factorización Incompleta.

En la actualidad existen varios precondicionadores que resultan de factorizaciones hechas a matrices arbitrarias. Sin embargo, existen algunas variantes que se benefician de la estructura especial que pueda tener una matriz: simétrica definida positiva, dispersas, etc. Para el caso de las matrices con un patrón de dispersión, se tiene un procedimiento denominado factorización incompleta, es decir, si $A$ es una matriz dispersa, las matrices resultantes de cualquier factorización usualmente no poseen el mismo patrón de dispersión de la matriz $A$, y como la idea es conservar este patrón, entonces se descartan los elementos diferentes de cero en aquellas posiciones donde la matriz $A$ tenía un cero. Tales posiciones se denominan posiciones de llenado (fill-in). (Tomado de [3])

La idea de generar factorizaciones aproximadas ha sido estudiada por varios investigadores. El primero en hacerlo fue R. Varga en 1960 [23]. Posteriormente J. Meijerink y H. Van der Vorst en 1977 [14] la hicieron popular cuando la usaron como una herramienta generadora de precondicionadores para el método Gradientes Conjugados y otros métodos iterativos.

## 3.2 Precondiciondores Jacobi, SOR y SSOR.

Supóngase que se desea resolver un sistema lineal de la forma

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

donde la matriz $A \in \mathbb{R}^{n\times n}$ es una matriz cuadrada descompuesta de la forma

$$A = E + D + F$$

donde $D$ es la diagonal de $A$, $E$ es la parte inferior estricta de $A$ (sin la diagonal) y $F$ la parte superior estricta de $A$. Para este caso, Jacobi, SOR y SSOR han sido definidos y usados como métodos iterativos los cuales no serán implementados de esa forma en esta investigación pero se considerarán algunos pasos para determinar como se ven expresadas las matrices precondicionadoras que llevan su mismo nombre. El método de Jacobi para resolver el sistema de la forma (2.1) esta definido como [22]:

$$x_{k+1} = -D^{-1}(E + F)x_k + D^{-1}b, \quad k = 0, 1, ....$$

o equivalentemente

$$x_{k+1} = Gx_k + f, \tag{3.2}$$

donde $G = -D^{-1}(E + F) = -D^{-1}(A - D)$ y $f = D^{-1}b$.
La iteración (3.2) puede ser usada para resolver sistemas de la forma

$$(I - G)x = f. \tag{3.3}$$

Ya que $G = I - D^{-1}A$, el sistema (3.3) puede ser reescrito como

$$D^{-1}Ax = D^{-1}b. \tag{3.4}$$

El sistema (3.4) puede verse como un sistema *precondicionado por la izquierda* de (2.1) con $D$ la matriz precondicionadora de Jacobi; esto es,

$$M_J = D.$$

Para el caso de mínimos cuadrados, el precondicionador de Jacobi para el sistema (1.3) es tomado como $M = D_1$ con $A^TA = E_1 + D_1 + F_1$.

Se puede definir otra descomposición de la matriz $A$ (ver [22]) de la forma

$$\omega A = (D + \omega E) - (-\omega F + (1 - \omega)D), \tag{3.5}$$

que da lugar al método iterativo conocido como método de SOR. SOR depende del parámetro $\omega$ con $0 < \omega < 2$ (ver [1]) y está dado por la formula de recurrencia

$$(D + \omega E)x_{k+1} = [-\omega F + (1 - \omega)D]x_k + \omega b. \tag{3.6}$$

Análogamente se puede definir otro médoto de SOR de la forma

$$(D + \omega F)x_{k+1} = [-\omega E + (1 - \omega)D]x_k + \omega b. \tag{3.7}$$

Mediante una especie de combinación entre las ecuaciones (3.6) y (3.7) se deriva otro método conocido como SOR simétrico (SSOR) y viene definido exactamente por las ecuaciones

$$(D + \omega E)x_{k+1/2} = [-\omega F + (1 - \omega)D]x_k + \omega b.$$

$$(D + \omega F)x_{k+1} = [-\omega E + (1 - \omega)D]x_{k+1/2} + \omega b.$$

El parámetro $\omega$ en SOR debe ser escogido con sumo cuidado cumpliendo con ciertos teoremas, por su parte SOR simétrico (SSOR) no necesita tanto cuidado para su escogencia [1]. Si las recurrencias anteriores se escriben de la forma

$$x_{k+1} = G_\omega x_k + f_\omega,$$

entonces mediante operaciones algebraicas se puede determinar que los precondicionadores para SOR y SSOR son respectivamente

$$M_{SOR} = \frac{1}{\omega}(D + \omega E),$$

$$M_{SSOR} = \frac{1}{\omega(2 - \omega)}(D + \omega E)D^{-1}(D + \omega F).$$

El parámetro $\omega$ en SSOR, como ya se mencionó, no necesita de mucho cuidado para escogerlo y frecuentemente tomando $\omega = 1$ se puede obtener un buen resultado [1]. Tomando $\omega = 1$ el precondicionador de SSOR queda expresado de la forma

$$M_{SSOR} = (D + E)D^{-1}(D + F).$$

Una interesante observación es que $M_{SSOR}$ toma la forma

$$M_{SSOR} = LU$$

donde

$$L \equiv (D + E)D^{-1} = I + ED^{-1}, \;\; U = D + F.$$

Si se desea calcular $A - LU$ se puede notar que si se obliga a la matriz $L$ a tener la misma estructura que la parte inferior de $A$ y la matriz $U$ poseer la misma estructura que la parte triangular superior de $A$ entonces puede ser posible hacer que la norma de la diferencia $A - LU$ sea más pequeña. Esto induce a calcular factorizaciones incompletas de las que se hablaron al inicio de este capítulo y que se estarán desarrollando a lo largo del mismo.

En el caso en que se desea resolver un problema de mínimos cuadrados, los precondicionadores de SOR y SSOR son escogidos tomando en cuenta la descomposición $A^T A = E_1 + D_1 + F_1$.

## 3.3 Precondicionadores Basados en Factorización $LU$.

Se considera el problema de mínimos cuadrados (1.2). Se sabe que para mejorar la rata de convergencia de un método iterativo se puede precondicionar el problema de manera que ahora el sistema se convierte en (2.30).

En [2] Björk y Yuan consideran un tipo de precondicionador basado en la partición de $A$

$$PA = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \tag{3.8}$$

donde $P$ es una matriz de permutación, $A_1 \in \mathbb{R}^{n,xn}$ una submatriz de $A$ no singular que se toma como precondicionador y $A_2 \in \mathbb{R}^{(m-n)\times n}$. Tal técnica fue sugerida con anterioridad

por Läuchli [16] y luego investigado por Chen [4].

En este trabajo se desarrollará esta técnica similar a como trabajaron Björck y Yuan en [2]. La matriz de permutación $P$ debe ser escogida tal que las $n$ primeras filas de $A_1$ sean linealmente independientes y $AA_1^{-1}$ este bien condicionada.
Se asume que $PA$ ya está formada y que se tiene calculada la matriz $C = A_2A_1^{-1} \in \mathbb{R}^{(m-n)xn}$ donde

$$AA_1^{-1} = \begin{pmatrix} I_n \\ A_2A_1^{-1} \end{pmatrix} = \begin{pmatrix} I \\ C \end{pmatrix}$$

Con esto, el problema precondicionado puede ser escrito de la forma

$$min_{y\in\mathbb{R}^n} \left\| \begin{pmatrix} I_n \\ C \end{pmatrix} y - \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right\|_2, \quad y = A_1x, \quad C = A_2A_1^{-1}, \tag{3.9}$$

donde $b$ ha sido particionado de la forma $b = Pb$. El sistema de ecuaciones normales para el sistema (3.9) es

$$(I_n + C^TC)y = b_1 + C_2^Tb_2, \tag{3.10}$$

donde $x$ puede ser obtenido de $A_1x = y$.
Läuchli [16] propuso aplicar el algoritmo de Gradientes Conjugados al sistema de ecuaciones (3.10) para resolver de manera eficiente el problema. Esto conduce al siguiente algoritmo:

**Algoritmo 10** CGLS precondicionado con LU

Calcular $r_1^{(0)} = b_1,\ r_2^{(0)} = b_2,\ \ p^{(0)} = s^{(0)} = b_1 + C^T b_2,\ \gamma_0 = \|s^{(0)}\|_2^2$

**para** $k = 0, 1, 2, ..., hasta\ que\ \gamma_k > tol$ **hacer**

$q^{(k)} = Cp^{(k)}$,

$\alpha_k = \gamma_k/(\|p^{(k)}\|_2^2 + \|q^{(k)}\|_2^2)$,

$r_1^{(k+1)} = r_1^{(k)} - \alpha_k p^{(k)}$,

$r_2^{(k+1)} = r_2^{(k)} - \alpha_k q^{(k)}$,

$s^{(k+1)} = r_1^{(k+1)} + C^T r_2^{(k+1)}$,

$\gamma_{k+1} = \|s^{(k+1)}\|_2^2$,

$\beta_k = \gamma_{k+1}/\gamma_k$,

$p^{(k+1)} = s^{(k+1)} + \beta_k p^{(k)}$

**fin para**

Obtener $x$ de $A_1 x = b_1 - r_1$.

Se sabe que aplicando el método CG a las ecuaciones normales precondicionadas (2.31), la $\|r - r_k\|_2$ satisface la siguiente desigualdad:

$$\|r - r_k\|_2 \leq 2\left(\frac{\sqrt{c} - 1}{\sqrt{c} + 1}\right)^k \|r - r_0\|_2,$$

donde $r_0$ es el residual inicial y $c = Cond(M^{-T}A^T A M^{-1})$. Para las ecuaciones (3.10) esta relación se sigue manteniendo y en este caso $c = cond(I_n + C^T C)$.
Para más detalles sobre la convergencia de CG aplicado al sistema (3.10) ver [2] y [6].

### 3.3.1 Algoritmo de Factorización $LU$.

En los casos donde $A$ tiene muchas más filas que columnas es recomendable factorizar sólo $A_1$ en vez de $A$ y no formar la matriz $C = A_2 A_1^{-1}$ explícitamente. Si $C$ no es

calculada, se necesita entonces presentar en cada paso iterativo dos multiplicaciones matriz-vector con $A_2$ y $A_2^T$ y resolver dos sistemas lineales de la forma

$$A_1 y = c \quad y \quad A_1^T z = d.$$

Para hacer esto de manera eficiente se calcula la factorización $LU$ de $A_1$.
Se usará el algoritrmo de factorización $LU$ dado por Gilbert y Peierls [7] modificado para obtener la factorización $LU$ de $A_1$.

El algoritmo de Gilbert y Peierls puede ser modificado para trabajar con una matriz de tamaño $n \times m$ con $n \leq m$. Para el problema de mínimos cuadrados (2.30) la matriz $A$ es de $m \times n$ con $m \geq n$, por lo tanto se aplica el algoritmo de Gilbert y Peierls a la matriz $A^T$ para obtener la factorización $A_1 = U^T L^T$ de una submatriz $A_1$ de tamaño $n \times n$ no singular.
**Descripción de los términos**: $j$ será el índice de la fila de $L$ y de $U$ a calcular.
$a_j = (a_{j1}, ..., a_{j,j-1})^T$ , $a_j^{'} = (a_{jj}, ..., a_{jn})^T$,

$$L_j = \begin{pmatrix} l_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ l_{j-1,1} & \cdots & l_{j-1,j-1} \end{pmatrix} \quad L_j^{'} = \begin{pmatrix} l_{j1} & \cdots & l_{j,j-1} \\ \vdots & \ddots & \vdots \\ l_{n,1} & \cdots & l_{n,j-1} \end{pmatrix}$$

y $l_j^{'} = (l_{jj}, ..., l_{nj})^T$, $u_j = (u_{1j}, ..., u_{j-1,j})^T$ donde $l_{jj} = 1$. Se usará $c_j^{'} = (c_{jj}, ..., c_{nj})^T$ como un resultado intermedio.

**Algoritmo 11** LUGP

1: Dar valores iniciales $l = m,\ index(i) = i,\ i = 1, .., m$;
2: **para** $j = 1, 2, ..., n$ **hacer**
3: Resolver $L_j u_j = a_j$ para $u_j$; ($a_j^T$ es la $j$-ésima fila de $A$)
4: $c_j = a_j - L_j u_j$;
5: **si** $\|c_{jj}\| \geq \epsilon$ **entonces**
6: ir al paso 11
7: **si no**
8: intercambiar la $j$-ésima columna y la $l$-ésima columna de $A^T$ y actualizar los índices y $l$;
9: **si** $l \leq j$ **entonces**
10: $rank(A) < n$ **parar**
11: **si no**
12: ir al paso 3
13: **fin si**
14: **fin si**
15: $u_{jj} = c_{jj}$;
16: $l_j = c_j / u_{jj}$
17: **fin para**

## 3.4 Cholesky Incompleto (IC).

Sea $A \in \mathbb{R}^{nxn}$ una matriz cuadrada simétrica definida positiva (SDP) y $L$ una matriz triangular inferior con elementos diagonales positivos tal que

$$A = LL^T \tag{3.11}$$

entonces, como se observó en el capítulo 1, $L$ es llamado el **factor de Cholesky** de $A$ y la factorización (3.11) es llamada **factorización de Cholesky** de $A$. Nótese que factorización de Cholesky es un caso particular de la factorización $LU$ cuando la matriz a factorizar es simétrica definida positiva [5].
Matemáticamente el factor de Cholesky de la matriz $A^TA$ es igual al factor $R$ de la descomposición $QR$ de $A$ debido a que

$$A^TA = R^TQ^TQR = R^TR,$$

ya que $Q$ es ortogonal. En la resolución de problemas de mínimos cuadrados grandes y dispersos, la factorización de Cholesky es usada como una técnica de precondicionamiento para las ecuaciones normales. Se puede notar que tomando como precondicionador a $M = L$ donde $L$ es el factor de Cholesky de $A^TA$ se tiene que

$$Cond(AM^{-1}) = Cond(Q) = 1,$$

salvo errores de redondeo. Por lo tanto, escoger a $L$ como un precondicionador para las ecuaciones normales implica que todos los métodos iterativos usados para resolver el problema de mínimos cuadrados deben converger en una iteración en aritmética exacta. Así, escoger un precondicionador $M$ que de alguna forma se aproxime mucho a $L$ puede esperarse que sea eficiente. Una aproximación a $L$ es tomar el factor de Cholesky incompleto de $A^TA$, es decir, $M = \tilde{L}$ donde

$$C = A^TA = \tilde{L}\tilde{L}^T - E,$$

y $E$ es una matriz de error. Se quiere que $\|E\|$ sea pequeña y que $\tilde{L}$ sea dispersa.

Se puede notar que no es necesario calcular la matriz completa $C = A^TA$. Todo lo que se requiere es poder tener acceso a una fila a la vez, y así, los elementos diferentes

de cero en la fila *i* de *C* pueden ser calculados cuando sea necesario y luego se descarta. A continuación se presenta el algoritmo de factorización incompleta de Cholesky *sin llenado*, es decir, que mantiene al factor $L$ con el mismo patrón de dispersión que la matriz a factorizar.

**Algoritmo 12** Factorización Incompleta de Cholesky

1: Se elige $l_{11} = \sqrt{c_{11}}$
2: **para** i=2,...,n **hacer**
3:  **para** j=1,2,...,i-1 **hacer**
4:  **si** $c_{ij} = 0$ **entonces**
5:  $l_{ij} = 0$
6:  **si no**
7:  $l_{ij} = \frac{1}{l_{jj}}(c_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk})$
8:  **fin si**
9:  $l_{ii} = \sqrt{c_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$
10:  **fin para**
11: **fin para**

Cuando se factoriza a la matriz $C = A^T A$ mediante el algoritmo 12, se necesita estudiar muy bien los resultados obtenidos ya que dos diferentes dificultades se pueden encontrar. La primera es que la factorización de Cholesky podría no estar definida debido a que por construcción, el factor $L$ del algoritmo debe tener elementos diagonales no negativos. La segunda dificultad es que, asumiendo que la factorización no cause problema, el factor $L$ obtenido podría estar muy mal condicionado. Esto puede pasar incluso si la matriz residual $E = C - LL^T$ es razonablemente pequeña [21].

## 3.5 Factorización $LQ$ Incompleta.

Considere una matriz general $A \in \mathbb{R}^{m\times n}$ dispersa y denótese sus filas por $a_1, a_2, ..., a_m$. La factorización (completa) $LQ$ de $A$ está definida por

$$A = LQ,$$

donde $L$ es una matriz triangular inferior y $Q$ es ortogonal, es decir, $QQ^T = I$. El factor $L$ en esta factorización es idéntico al factor de Cholesky de la matriz $B = AA^T$. De hecho, si $A = LQ$ donde $L$ posee elementos diagonales positivos, entonces

$$B = AA^T = LQQ^TL^T = LL^T.$$

Esta relación, al igual que la generada en la factorización de Cholesky, puede ser explotada para obtener precondicionadores para las ecuaciones normales.

El proceso para construir la factorización completa $LQ$ de $A$ se presenta en el siguiente algoritmo.

**Algoritmo 13** Factorización LQ

1: **para** i=1,...,n **hacer**
2: Calcular $l_{ij} = q_j^T a_i$, para $j = 1, 2, ..., i-1$
3: Calcular $q_i = a_i - \sum_{j=1}^{i-1} l_{ij}q_j$, and $l_{ii} = \|q_i\|_2$,
4: Si $l_{ii} = 0$ entonces Parar; si no calcular $q_i = q_i/l_{ii}$
5: **fin para**

En el caso de resolver problemas del tipo (1.2), una técnica de precondicionamiento es realizar la factorización $LQ$ incompleta de $A$. Hay dos formas de obtener la matriz $L$. La primera es formar la matriz $B$ explícitamente y usar una factorización de Cholesky

para matrices dispersas. Esto requiere presentar todos los datos de $AA^T$ la cual es mucho mas densa que $A$. Una segunda forma es usar el proceso de Gram-Schmidt. En principio esta idea no parece ser la mas óptima debido a la pérdida de ortogonalidad que se produce cuando se desea ortogonalizar una gran cantidad de vectores. Sin embargo, debido a que las filas de $Q$ permanecen con una gran cantidad de entradas iguales a cero al implementar el algoritmo de factorización incompleta $LQ$, una fila dada de $A$ será ortogonal a muchas de las filas previas de la matriz $Q$. Como resultado, el proceso de Gram-Schmidt es mucho menos propenso a dificultades numéricas [21]. Desde el punto de vista computacional Gram-Schmidt es óptimo debido a que no requiere asignar más espacio del necesario. Otra ventaja es la simplicidad a la hora de realizar el proceso de ortogonalización.

Para definir una factorización incompleta se usa una estrategia de llenado. En términos generales se puede hacer de la manera que se propone en [22]. Sean $P_L$ y $P_Q$ los patrones de escogidos para $L$ y $Q$ respectivamente. En este caso, la única restricción para $P_L$ es que

$$P_L \subset \{(i, j) | i \neq j\}.$$

En cuanto a $P_Q$ se exige que para cada fila de $Q$ debe haber al menos un elemento diferente de cero. Estos conjuntos serán seleccionados usando una estrategia de llenado basada en la magnitud de los elementos generados. Se define entonces el siguiente algoritmo donde $x_i$ denota la $i$-ésima fila de la matrix $X$ y $x_{ij}$ su entrada $(i, j)$.

**Algoritmo 14** Factorización LQ Incompleta

1: **para** i=1,...,n **hacer**
2: Calcular $l_{ij} = q_j^T a_i$, para $j = 1, 2, ..., i-1$
3: Reemplazar $l_{ij}$ por cero si $(i, j) \in P_L$
4: Calcular $q_i = a_i - \sum_{j=1}^{i-1} l_{ij} q_j$,
5: Reemplazar cada $q_{ij}$, $j = 1, 2, .., n$ por cero si $(i, j) \in P_Q$
6: $l_{ii} = \|q_i\|_2$
7: Si $l_{ii} = 0$ entonces Parar; si no calcular $q_i = q_i / l_{ii}$
8: **fin para**

En el algoritmo 14 después de que el $i$-ésimo paso es presentado se mantiene la siguiente relación

$$q_i = l_{ii} q_i + r_i = a_i - \sum_{j=1}^{i-1} l_{ij} q_j$$

o

$$a_i = \sum_{j=1}^{i} l_{ij} q_j + r_i$$

donde $r_i$ es la fila de elementos que han sido sustituidos por la indicación de la tolerancia de la fila $q_i$. Esta ecuación se convierte en

$$A = LQ + R$$

donde $R$ es la matriz la cual su $i$-ésima fila es la $r_i$.

# CAPÍTULO 4

# Resultados Numéricos

En este capítulo se presentan los resultados numéricos después de resolver problemas de mínimos cuadrados lineales extraidos de varias aplicaciones científicas y de ingeniería. Para obtener estos resultados se utilizó un computador con un procesador de 2.53 GHz y una memoria RAM de 1GB. El software para programar los algoritmos fue Matlab versión 7.8.0.347 (R2009a). En este trabajo se resolvieron cuatro problemas de mínimos cuadrados grandes y dispersos todos provenientes de la agrimensura (arte de medir tierras), extraídos de la colección Harwell-Boeing. Además se resolvieron otros cuatro problemas grandes y dispersos provenientes de discretizar la ecuación de difusión-convección no transistoria mediante el método DRM-MD desarrollados por M. Portapila y H. Power. [20]. El criterio de parada para el método CGLS fue detenerse si la norma del residual relativo $\frac{\|A^T r\|}{\|A^T r_0\|}$ era menor a la tolerancia especificada, donde $r_0$ es el residual inicial. La tolerancia usada para los criterios de parada fue de $\frac{1}{2} \times 10^{-8}$. A lo largo de este capítulo se presentan tablas mostrando los resultados obtenidos al resolver estos problemas sin precondicionar así como también los resultados de haber aplicado las diversas técnicas de precondicionamiento tratadas en el capítulo 3. Se muestran

gráficos que expresan el comportamiento que tuvo el residual para ciertos problemas cuando fueron ejecutados los códigos de LSQR y CG.

Tabla 4.1: Problemas a Resolver.

| *Matriz* | *m* | *n* | *nnz* | *N Cond* | *Fuente* | *Autor* |
|---|---|---|---|---|---|---|
| *ILLC*1033 | 1033 | 320 | 4719 | $1{,}88 \times 10^4$ | Harwell-Boeing | Michael Saunders |
| *ILLC*1850 | 1850 | 712 | 8636 | $1{,}40 \times 10^3$ | Harwell-Boeing | Michael Saunders |
| *WELL*1033 | 1033 | 320 | 4732 | $1{,}66 \times 10^2$ | Harwell-Boeing | Michael Saunders |
| *WELL*1850 | 1850 | 712 | 8755 | $1{,}11 \times 10^2$ | Harwell-Boeing | Michael Saunders |
| *Mesh*0 | 512 | 467 | 9280 | $9{,}17 \times 10^2$ | Portapila M. | Portapila M. y Power H. |
| *MeshIV* | 792 | 735 | 15996 | $2{,}34 \times 10^2$ | Portapila M. | Portapila M. y Power H. |
| *MeshVI* | 1040 | 983 | 24128 | $2{,}39 \times 10^3$ | Portapila M. | Portapila M. y Power H. |
| *MeshVIII* | 1800 | 1653 | 34884 | $2{,}08 \times 10^3$ | Portapila M. | Portapila M. y Power H. |
| *Amc* | 1099 | 437 | 3580 | $9{,}45 \times 10^{10}$ | Propia | Meza Y. |

En la tabla 4.1 se resumen las características de las nueve matrices asociadas a los problemas de mínimos cuadrados resueltos en esta investigación. Se indica el nombre de las matrices, el número de filas y columnas de cada matriz los cuales están representados por las letras *m* y *n*, respectivamente, *nnz* indica el número de elementos diferentes de cero que posee cada matriz y *N Cond* su número de condición. La matriz *Amc* fue una matriz dispersa creada en matlab tal que tuviera un número de condición alto.

A continuación se presenta una serie de tablas que muestran los efectos de resolver los nueve diferentes problemas sin precondicionar así como también los resultados al usar las diversas técnicas de precondicionamiento descritas en el capítulo 3. Para todas las tablas, *Iter* indica el número de iteraciones que necesitó cada método iterativo para converger a la aproximación de la solución, *Res Rel* representa el residual relativo que se obtuvo en la última iteración y *Tiempo (s)* señala el tiempo en segundos que

tomó cada método para satisfacer la tolerancia.

Tabla 4.2: Resolución de Problemas sin Precondicionar.

| *Problema* | *Método* | *Iter* | *Res rel* | *Tiempo (s)* |
|---|---|---|---|---|
| $ILLC1033$ | CGLS | 2722 | $4{,}5793 \times 10^{-9}$ | 0,28 |
| | LSQR | 3281 | $3{,}4151 \times 10^{-9}$ | 0,37 |
| $ILLC1850$ | CGLS | 1742 | $4{,}9453 \times 10^{-9}$ | 0,17 |
| | LSQR | 2191 | $6{,}0818 \times 10^{-9}$ | 0,37 |
| $WELL1033$ | CGLS | 163 | $3{,}2061 \times 10^{-9}$ | 0,01 |
| | LSQR | 185 | $3{,}1269 \times 10^{-9}$ | 0,03 |
| $WELL1850$ | CGLS | 438 | $4{,}6674 \times 10^{-9}$ | 0,04 |
| | LSQR | 479 | $5{,}1802 \times 10^{-9}$ | 0,09 |
| $Mesh0$ | CGLS | 1334 | $4{,}0949 \times 10^{-9}$ | 0,96 |
| | LSQR | 1913 | $1{,}6129 \times 10^{-10}$ | 1,56 |
| $MeshIV$ | CGLS | 4133 | $4{,}2819 \times 10^{-9}$ | 11,71 |
| | LSQR | 5787 | $6{,}9843 \times 10^{-11}$ | 17,25 |
| $MeshVI$ | CGLS | 4155 | $4{,}5990 \times 10^{-9}$ | 21,01 |
| | LSQR | 5946 | $7{,}0014 \times 10^{-11}$ | 31,11 |
| $MeshVIII$ | CGLS | 5479 | $4{,}1428 \times 10^{-9}$ | $8{,}28 \times 10^{1}$ |
| | LSQR | 8880 | $3{,}5953 \times 10^{-11}$ | $1{,}30 \times 10^{2}$ |
| *Amc* | CGLS | 3306 | $4{,}3581 \times 10^{-9}$ | 7,19 |
| | LSQR | 2655 | $4{,}8339 \times 10^{-8}$ | 6,25 |

La tabla 4.2 muestra los resultados de aplicar los métodos iterativos CGLS y LSQR para resolver los nueve diferentes problemas de mínimos cuadrados lineales sin usar técnica de precondicionamiento. Se puede notar que para los problemas que están mejor condicionados ($WELL1033$ y $WELL1850$) ambos métodos iterativos necesitaron de menos pasos para satisfacer la tolerancia, en comparación con los problemas mal condicionados como *Amc* o $ILLC1033$ que necesitaron de muchas más iteraciones para poder obtener una norma de residual relativo menor o igual a $\frac{1}{2}\times 10^{-8}$ observándose también una gran diferencia en cuanto al tiempo de ejecución empleado para estos problemas. A pesar de que el número de condición de la matriz $MeshVIII$ es menor que

el de la matriz *Amc*, la diferencia en los esfuerzos que deben realizar ambos métodos iterativos para trabajar con dichos problemas se ve justificada debido a la gran cantidad de elementos diferentes de cero que posee la matriz $MeshVIII$ en comparación con *Amc* haciendo que los métodos aplicados a $MeshVIII$ se tomen mucho más tiempo debido a la necesidad de realizar muchas más operaciones por iteración. Es importante destacar que para el problema *Amc* que está muy mal condicionado, el método LSQR funcionó mejor que CGLS realizando menos iteraciones y tomándose menos tiempo que CGLS, característica que no ocurrió en la resolución de los problemas mejores condicionados.

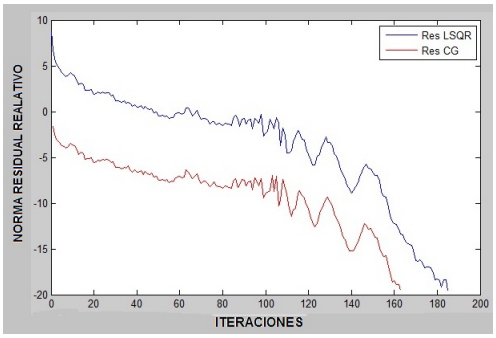


**Figura** 4.1: Comportamiento del residual relativo para el problema $WELL1033$ (escala logarítmica).

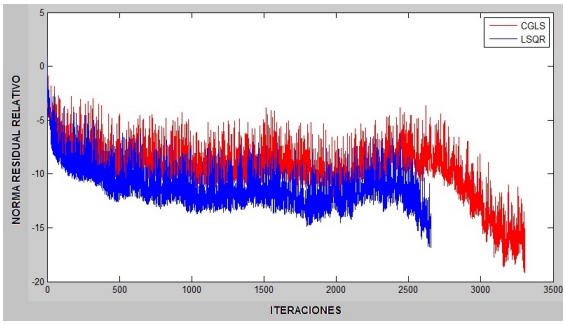


**Figura** 4.2: Comportamiento del residual relativo al resolver el problema *Amc* (escala logarítmica).

En la figura 4.1 se muestra el comportamiento que tuvo el residual relativo durante la resolución del problema $WELL1033$ usando los métodos CGLS y LSQR. En esta gráfica se puede observar que a medida que los métodos iterativos realizaron iteraciones, la norma del residual relativo no seguía un comportamiento monótono decreciente sino que fue alternando valores produciendo una función con altibajos estabilizándose en las últimas iteraciones hasta lograr la convergencia. En este caso se puede notar

que el método CGLS tuvo un mejor desempeño ya que necesitó menos iteraciones que LSQR para lograr que el valor de la norma del residual relativo fuese menor que la tolerancia.

En la figura 4.2 se muestra el comportamiento que tuvo el residual relativo al implementarse los métodos CGLS y LSQR para resolver el problema *Amc*. En esta figura se observa que ambos métodos iterativos necesitaron de más esfuerzo en cuanto al número de iteraciones que se debió realizar para conseguir la convergencia. El comportamiento del residual presenta oscilaciones bastante frecuentes a lo largo de todo el proceso donde los altibajos son muy notorios y se puede ver que no hubo una reducción del residual tan significativa hasta las últimas iteraciones donde logra disminuir un poco hasta satisfacer la tolerancia. A pesar de esto, LSQR mostró un mejor desempeño en cuanto a iteraciones necesitadas para poder lograr una reducción satisfactoria en la norma del residual relativo.

La razón por la que estos dos problemas fueron escogidos para expresar el comportamiento del residual relativo fue debido a la gran diferencia en cuanto al número de condición que poseen ambos problemas y así poder notar los distintos desempeños que muestran los métodos iterativos a la hora de resolver un problema mal condicionado o un problema el cual posee condiciones más favorables.

Tabla 4.3: PCGLS usando SSOR como Técnica de Precondicionamiento ($\omega = 1$).

| *Problema* | *Iter* | *Res rel* | *Tiempo(s)* |
|---|---|---|---|
| *ILLC*1033 | 1427 | $4{,}0610 \times 10^{-9}$ | 0,31 |
| *ILLC*1850 | 1072 | $4{,}8644 \times 10^{-9}$ | 0,48 |
| *WELL*1033 | 107 | $4{,}6797 \times 10^{-9}$ | 0,03 |
| *WELL*1850 | 218 | $3{,}6456 \times 10^{-9}$ | 0,10 |

La tabla 4.3 muestra los resultados de aplicar la técnica de precondicionamien-

to SSOR a los problemas extraidos de la colección Harwell-Boeing. Los resultados obtenidos indican que el precondicionador logró reducir el número de iteraciones para la resolución de cada problema comparándolo con la versión sin precondicionar. En cuanto al tiempo, la técnica SSOR no logra ganarle al método no precondicionado. Para las matrices resultantes del método DRM-MD también se realizaron estas pruebas pero no se muestran en la tabla debido a que la implementación del método PGCLS con esta técnica no mejoró en cuanto a tiempo de ejecución ni a iteraciones realizadas mostrando incluso muchas más iteraciones que al resolver los problemas sin precondicionar. Es importante indicar que el precondicionador SOR también fue probado con el método PCGLS y los resultados obtenidos fueron mejores que con SSOR. Se decidió usar este precondicionador no simétrico para observar también su comportamiento y comparar con los resultados obtenidos en SSOR.

Tabla 4.4: Método PCGLS usando Cholesky Incompleto como Técnica de Precondicionamiento ($M = \tilde{L}$).

| *Problema* | *Iter* | *Res rel* | *Tiempo (s)* |
|---|---|---|---|
| $ILLC1033$ | 65 | $7{,}6435 \times 10^{-10}$ | 0,04 |
| $ILLC1850$ | 21 | $3{,}1602 \times 10^{-9}$ | 0,24 |
| $WELL1033$ | 5 | $1{,}1502 \times 10^{-9}$ | 0,05 |
| $WELL1850$ | 5 | $2{,}4768 \times 10^{-9}$ | 0,31 |
| $Mesh0$ | 5 | $1{,}1699 \times 10^{-10}$ | 0.04 |
| $MeshIV$ | 7 | $5{,}4857 \times 10^{-10}$ | 0,10 |
| $MeshVI$ | 6 | $2{,}2757 \times 10^{-9}$ | 0,18 |
| $MeshVIII$ | 7 | $3{,}9089 \times 10^{-10}$ | 0,45 |
| $Amc$ | 9 | $2{,}2071 \times 10^{-9}$ | 0,20 |

La tabla 4.4 muestra los resultados obtenidos al implementar el método PCGLS usando como precondicionador $M = \tilde{L}$ donde $\tilde{L}$ es el factor incompleto de Cholesky de las matrices $A^T A$ para cada problema. Usar esta técnica hizo que el método PCGLS

presentara una reducción bastante notoria en cuanto a la cantidad de iteraciones necesitadas para converger en comparación con las realizadas sin precondicionar, además de conseguir una mejor disminución del residual relativo. Nótese que el rendimiento de PCGLS usando Cholesky incompleto como precondicionador para resolver el problema $ILLC1033$ no fue tan satisfactorio como en los otros problemas.

Tabla 4.5: Método PCGLS usando $M = \tilde{L}^T$ de la Factorización Incompleta $\tilde{L}\tilde{Q}$ de $A^T$.

| *Problema* | *Iter* | *Res rel* | *Tiempo (s)* |
|---|---|---|---|
| $ILLC1033$ | 244 | $2{,}8994 \times 10^{-9}$ | 0,24 |
| $ILLC1850$ | 445 | $4{,}5320 \times 10^{-9}$ | 4,01 |
| $WELL1033$ | 106 | $1{,}7300 \times 10^{-9}$ | 0,13 |
| $WELL1850$ | 254 | $4{,}5974 \times 10^{-9}$ | 2,34 |
| $Mesh0$ | 173 | $4{,}6335 \times 10^{-9}$ | 0,9 |
| $MeshIV$ | 295 | $4{,}3794 \times 10^{-9}$ | 4,02 |
| $MeshVI$ | 414 | $4{,}8855 \times 10^{-9}$ | 9,9 |
| $MeshVIII$ | 521 | $4{,}8040 \times 10^{-9}$ | $3{,}55 \times 10^{1}$ |
| *Amc* | 17 | $4{,}3501 \times 10^{-10}$ | 0,26 |

En la tabla 4.5 se muestran los resultados obtenidos después de haber implementado el método PCGLS utilizando la técnica de precondicionamiento $M = \tilde{L}^T$ con $\tilde{L}$ la matriz triangular inferior de la factorización $\tilde{L}\tilde{Q}$ de $A^T$ obtenida usando el algoritmo 14. En este caso, el precondicionador dio resultados buenos, para todos los casos logró disminuir el número de iteraciones en comparación con la resolución de los problemas sin precondicionar pero le tomó a PCGLS más iteraciones para converger que aplicando la técnica Cholesky incompleto. Particularmente para resolver el problema $MeshVIII$ usar esta técnica de precondicionamiento no arrojó resultados tan efectivos en cuanto al tiempo de ejecución del método iterativo.

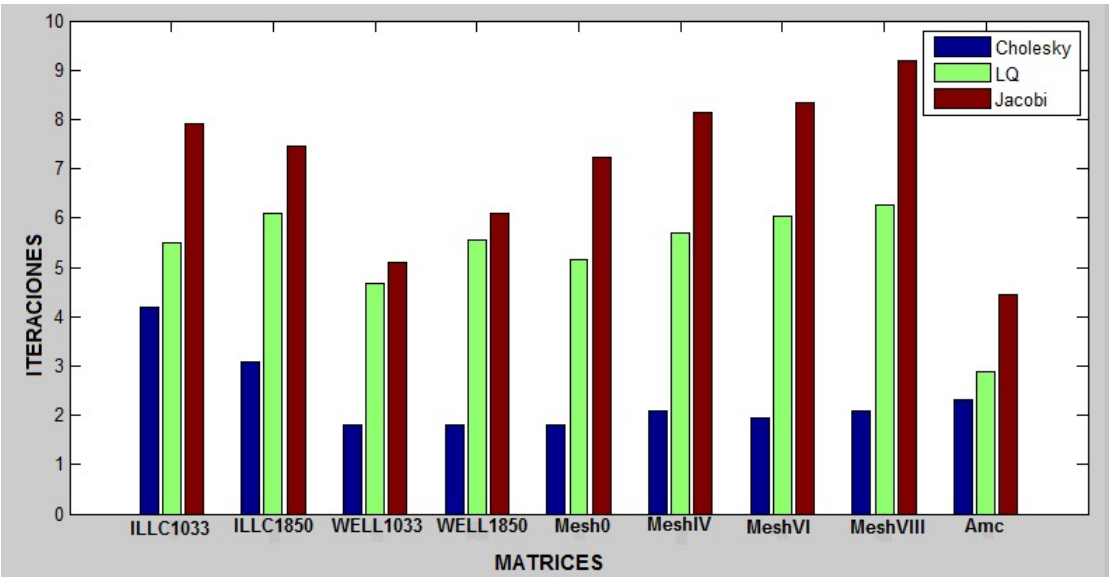


**Figura** 4.3: Resumen de iteraciones tomadas por PCGLS para resolver los nueve problemas aplicando los precondicionadores Cholesky incompleto, *LQ* incompleto y Jacobi (escala logarítmica).

La figura 4.3 muestra un resumen de las iteraciones que necesitó PCGLS para resolver los nueve problemas usando como técnica de precondicionamiento Cholesky incompleto, *LQ* incompleto y Jacobi. Se puede notar que Cholesky incompleto fue una técnica muy eficiente en cuanto a la reducción de iteraciones del método PCGLS a diferencia de Jacobi que incluso en algunos casos produjo que PCGLS necesitara de más iteraciones para satisfacer la tolerancia que al resolver los problemas sin precondicionar.

Tabla 4.6: PCGLS y PLSQR Usando $M = A_1$ como Precondicionador Calculando su Factorización $LU$ Incompleta.

| *Problema* | *Método* | *Iter* | *Res rel* | *Tiempo (s)* |
|---|---|---|---|---|
| $ILLC1033$ | PCGLS | 53 | $4{,}4493 \times 10^{-9}$ | 0,01 |
| | PLSQR | 68 | $2{,}6342 \times 10^{-9}$ | 0,02 |
| $ILLC1850$ | PCGLS | 80 | $4{,}1418 \times 10^{-9}$ | 0,038 |
| | PLSQR | 101 | $5{,}1626 \times 10^{-9}$ | 0,06 |
| $WELL1033$ | PCGLS | 52 | $3{,}9923 \times 10^{-9}$ | 0,01 |
| | PLSQR | 68 | $2{,}8548 \times 10^{-9}$ | 0,03 |
| $WELL1850$ | PCGLS | 79 | $4{,}9214 \times 10^{-9}$ | 0,04 |
| | PLSQR | 102 | $5{,}1352 \times 10^{-9}$ | 0,06 |
| $Mesh0$ | PCGLS | 11 | $1{,}3703 \times 10^{-9}$ | 0,015 |
| | PLSQR | 14 | $1{,}8204 \times 10^{-10}$ | 0,031 |
| $MeshIV$ | PCGLS | 12 | $2{,}1768 \times 10^{-9}$ | 0,04 |
| | PLSQR | 19 | $1{,}2473 \times 10^{-11}$ | 0,078 |
| $MeshVI$ | PCGLS | 12 | $2{,}2995 \times 10^{-9}$ | 0,04 |
| | PLSQR | 19 | $2{,}0693 \times 10^{-11}$ | 0,15 |
| $MeshVIII$ | PCGLS | 12 | $1{,}4010 \times 10^{-9}$ | 0,04 |
| | PLSQR | 17 | $1{,}5793 \times 10^{-11}$ | 0,32 |
| $Amc$ | PCGLS | 12 | $3{,}0854 \times 10^{-10}$ | 0,01 |
| | PLSQR | 12 | $2{,}8247 \times 10^{-9}$ | 0,06 |

En la tabla 4.6 se muestran los resultados de aplicar los métodos PCGLS y PLSQR para la resolución de los problemas indicados con la técnica de precondicionamiento $M = A_1$ calculando su factorización $LU$ incompleta. El método PCGLS fue usado para resolver el sistema $(I_n + C^T C)y = b_1 + C_2^T b_2$ mientras que LSQR se usó para resolver directamente el problema

$$min_{y \in \mathbb{R}^n} \left\| \begin{pmatrix} I_n \\ C \end{pmatrix} y - \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right\|_2 .$$

Se puede observar que con ésta técnica ambos métodos iterativos logran una buena reducción tanto en iteraciones como en tiempo de ejecución con respecto a la resolución

de los mismos problemas sin precondicionar. Se obtuvo una reducción en iteraciones desde un 63.24 % hasta un 99.68 % dependiendo del problema mientras que en tiempo, la reducción fue desde un 33.33 % hasta un 99.95 %. Nótese que para el problema $WELL1033$ el tiempo se mantuvo igualmente pequeño al resolver el problema sin precondicionar que con esta técnica.

Para el problema $WELL1033$, la norma del residual relativo usando la técnica de precondicionamiento $M = A_1$ fue disminuyendo de manera monótona al aplicar ambos métodos, sin presentar cambios bruscos a medida que se realizaron iteraciones. Para el problema *Amc*, el comportamiento del residual relativo usando este mismo precondicionador fue similar para ambos métodos, manteniendo un valor alto en las primeras iteraciones para luego descender de manera brusca hasta obtener una norma del residual relativo menor que la tolerancia, muy diferente de lo que expresó la figura 4.2.

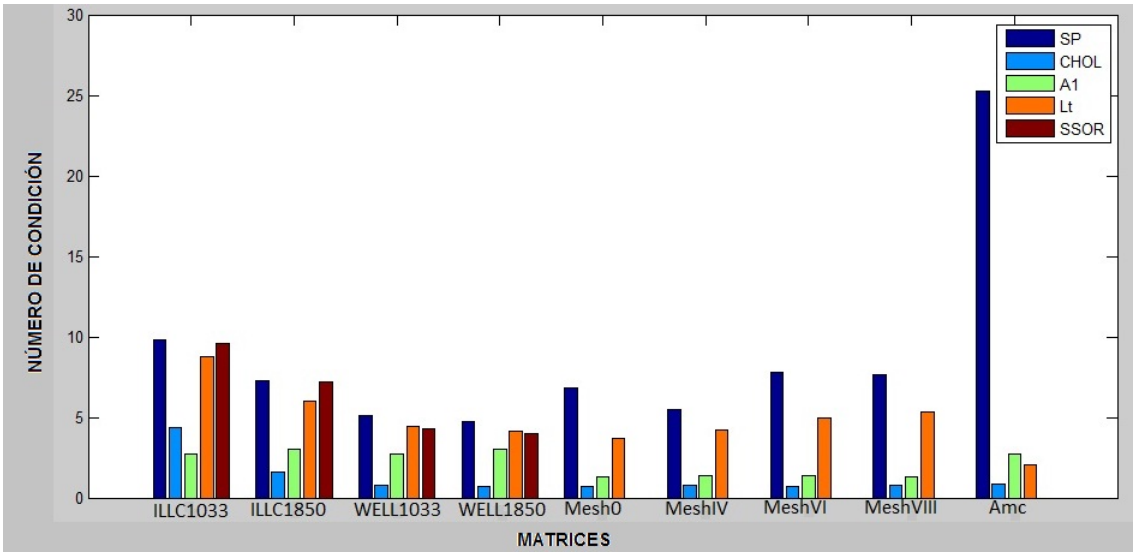


**Figura** 4.4: Número de condición de $AM^{-1}$ para los diversos precondicionadores usados (escala logarítmica).

En la figura 4.4 se puede observar mediante un gráfico de barras el número de condición de la matriz $AM^{-1}$ para cada precondicionador $M$ utilizado a lo largo de la investigación. En la leyenda del gráfico SP indica el número de condición de $A$, CHOL indica el número de condición de $A\tilde{L}^{-1}$ donde L es el factor incompleto de Cholesky de $A^TA$, $A1$ indica el número de condición de $A(A_1)^{-1}$, Lt es el número de condición de $A(\tilde{L}^T)^{-1}$ donde $\tilde{L}$ es la matriz triangular de la factorización $\tilde{L}\tilde{Q}$, SSOR es el número de condición de la matriz $MS^{-1}$ donde $S = (D + E)D^{-1}(D + F)$ es el precondicionador de SSOR escogiendo $\omega = 1$. Se puede notar que el número de condición de $AL^{-1}$ es el menor en la mayoría de los casos. Esto justifica el hecho de que aplicar PCGLS para resolver los problemas usando Cholesky incompleto como técnica de precondicionamiento haya presentado una convergencia rápida para la mayoría de los casos.

# CAPÍTULO 5

# Conclusiones y Trabajos a Futuro

Este trabajo permitió la aplicación de diversas técnicas de precondicionamiento para resolver problemas de mínimos cuadrados lineales grandes y dispersos, con el fin de estudiar su eficacia y comparar resultados con los obtenidos al resolver los mismos problemas sin precondicionar.

Los métodos iterativos CGLS y LSQR requirieron de una gran cantidad de iteraciones y tiempo de ejecución para resolver sin precondicionar los problemas de mínimos cuadrados lineales tratados en este trabajo. Al momento de resolver los problemas que estaban mal condicionados, los métodos CGLS y LSQR necesitaron de un esfuerzo mucho mayor al realizado para resolver problemas que poseían un número de condición más favorable. Para estos casos, el método LSQR presentó resultados más precisos viéndose reflejado en la norma del residual relativo obtenida en la última iteración y particularmente, para el problema *Amc* que está muy mal condicionado, LSQR logró presentar mejores resultados que CGLS en cuanto a iteraciones necesitadas para satisfacer la tolerancia y también en tiempo de ejecución. Por otra parte, cuando se aplicaron

técnicas de precondicionamiento para resolver los problemas de mínimos cuadrados, ambos métodos necesitaron realizar una menor cantidad de iteraciones lográndose en su mayoría una reducción de tiempo de ejecución.

En cuanto a las técnicas de precondicionamiento aplicadas para resolver los diferentes problemas de mínimos cuadrados, Jacobi y SSOR fueron precondicionadores poco efectivos permitiendo que el método CGLS realizara en algunos casos más iteraciones para converger que las necesitadas para resolver los mismos problemas sin precondicionar además de aumentar el tiempo de ejecución empleado. Esto se debe principalmente a que las matrices de precondicionamiento correspondientes a Jacobi y SSOR no lograron mejorar en gran magnitud el número de condición de la matrices $AM_{SSOR}^{-1}$ y $AM_{J}^{-1}$ en comparación con el número de condición de A. De esta manera se observo que dichas matrices de precondicionamiento no cumplen con esta propiedad, la cual es una de las propiedades más importantes que debe cumplir un precondicionador.

En cuanto al uso de precondicionadores para las ecuaciones normales, se recomienda aplicar el método Cholesky incompleto ya que presentó resultados muy eficientes en la resolución de casi todos los problemas logrando reducir en gran cantidad las iteraciones necesitadas por el método PCGLS para converger. En cuanto al tiempo de ejecución, esta técnica no logró ser la más rápida de todas, más sin embargo, no presentó tiempos de ejecución muy altos.

El método LSQR logró arrojar buenos resultados para todos los problemas al usar la técnica de precondicionamiento $M = A_1$ realizando muchas menos iteraciones para converger comparado con las realizadas sin precondicionar además de observarse una reducción bastante grande en cuanto al tiempo de ejecución, siendo incluso la técnica más rápida.

Tomando en cuenta los estudios realizados en este trabajo los cuales presentaron los resultados ya expuestos, es posible tomar estos últimos como base para realizar es-

tudios profundos que busquen la solución a problemas de mínimos cuadrados lineales grandes y dispersos mediante la creación e implementación de nuevas técnicas de precondicionamiento con el fin de comparar con resultados derivados de aplicar técnicas de precondicionamiento ya existentes. Trabajar con problemas mucho más grandes y que estén peor condicionados que los tratados en este trabajo.

# Bibliografía


[1] Björck A. *Numerical Methods for Least Squares Problems*. SIAM, 1996.

[2] Björck A. y Yuan Y. *Preconditioners for Least Squares Problems by LU Factorization*. ETNA, Vol. 8, 1999, 26-35.

[3] Cervantes G. y Carlos M. *Precondicionamiento de Métodos Iterativos.* Reimpreso de la Revista de la Academia Colombiana de Ciencias. Vol.28, No.106, 2004, 49-55.

[4] Chen Y. T. *Iterative Methods for Linear Least Squares Problems*. Technical Report CS-75-04, Universidad de Waterloo, 1975.

[5] Datta B. *Numerical Linear Algebra and Aplications*. Publicación Brooks/Cole, CA, 1995.

[6] Freund R. *A note on two block SOR methods for sparse least squares problems*, Linear Algebra Applications, 88/89, 1987, 211-221.

[7] Gilbert J. y Peierls T. *Sparse Partial Pivoting in Time Proportional to Arithmetic Operations*. TR 86-783, Departament of Computer Science, Universidad Cornell,1986.

[8] Givens W. *Computation of Plane Unitary Rotations Transforming a General Matrix to Triangular Form*. Journal of the Society for Industrial and Applied Mathematics. Vol. 6, No. 1, 1958, 26-50.

[9] Golub G. y Kahan W. *Calculating the Singular Values and Pseudo-inverse of a Matrix*. J. SIAM, Vol.2, No.2, 1965.

[10] Golub G. *Numerical Methods for Solving Least Squares Problems.* Numerical Mathematics, 1965, 206-216.

[11] Golub G. y Van Loan C. *Matrix Computation*. 3ra. edición, The Johns Hopkins University Press, 1996.

[12] Hestenes M. y Stiefel E. *Methods of Conjugate Gradients for Solving Linear Systems*. Journal of Research of the National Bureau of Standards Vol. 49, No.6, 1952.

[13] Householder A. *Unitary Triangularization of a Nonsymmetric Matrix*. J. Assoc. Comput. Mach., 5, 1958, 339-342.

[14] Meijerink J. y Van Der Vorst H. *An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-matrix*. Mathematics of Computation, Vol.31, No.137, 1977, 148–162.

[15] Molina B. y Raydán M. *Métodos Iterativos tipo Krylov para Sistemas Lineales.*. IV Escuela de Matemáticas de América Latina y el Caribe - XVII Escuela Venezolana de Matemáticas, Septiembre 2004, Mérida, Venezuela.

[16] Läuchli P. *Jordan- Elimination und Ausgleichung nach kleinsten Quadraten*. Numerische Mathematik, Vol.3, 1961, 226-240.

[17] Luenberger D. *An Approach to Nonlinear Programming*. Reimpreso de Journal of Optimization Theory and Applications, Vol.11, No.3, 1973.

[18] Ortega J. *Introduction to Parallel and Vectors Solutions for Linear Systems*. Plenum Press, Segunda edición, 1989.

[19] Paige C. y Saunders M. *LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares*. ACM Transactions on Mathematical Software, Vol. 8, No. 1, 1982, 43-71.

[20] Portapila M. y Power H. *Iterative Schemes for the Solution of System of Equations Arising from the DRM in Multi Domain Approach, and a Comparative Analysis of the Performance of two Different Radial Basis Functions used in the Interpolation*. Engineering Analysis with Boundary Elements 29, 2005,107-125.

[21] Saad Y. y Sosonkina M. *Enhanced Preconditioners for Large Sparse Least Squares Problems*. 2001.

[22] Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing, 2da. edición, 2000.

[23] Varga R. *Factorization and Normalized Iteratives Methods.*. Reimpreso de Boundary Problems in Differential Equations, Editado por R. Langer, The University of Wisconsin Press, 1960, 121–142.