Treball final de grau

GRAU DE MATEMÀTIQUES

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Aspectos teóricos e implementación del método iterativo GMRES para la resolución de sistemas lineales

Autor: José Luis Dorado Ladera

Director:	Dr. Arturo Vieiro
Realizado en:	Departament de Matemàtiques i Informàtica
Barcelona,	27 de junio de 2018

Abstract[1]

In this work we study different theoretical aspects of the GMRES algorithm (Generalized Minimal RESidual), including the use of Krylov subspaces, general results on convergence and the adapted version with restarted GMRES(m). GMRES is an iterative method that approximates the solution of a linear system $Ax = b$ by looking for the solution that minimizes the residue within the Krylov subspace. This method can be applied to general linear systems, because it does not require an specific structure of the matrix A of the system. It is especially suitable for system of high dimension when it is not possible to solve it by direct methods and when the classical iterative methods like Jacobi, Gauss-Seidel or SOR do not converge or do not converge in a reasonable amount of time. We complement the work with several remarks and pseudo-code schemes useful for the implementation (in C language) that we have done of the method. We use our own implementation of GMRES in several examples, improving the implementation until we reach the final version (included in the Appendix). In the memory we present some examples to illustrate some aspects of the convergence of GMRES for different spectra of A.

Resumen

En este trabajo se estudian los aspectos teóricos del algoritmo GMRES (Generalized Minimal RESidual), incluyendo el uso de subespacios de Krylov, los resultados generales de convergencia y la adaptación con reinicio del método llamada GMRES(m). GMRES es un método iterativo que aproxima la solución de un sistema lineal $Ax = b$ buscando minimizar el residuo dentro del subespacio de Krylov. Este método se puede aplicar a sistemas lineales en general ya que no requiere condiciones específicas sobre la estructura de la matriz A del sistema. Es especialmente adecuado en sistemas de dimensión grande cuando no es factible la resolución por métodos directos y métodos iterativos clásicos como Jacobi, Gauss-Seidel o SOR no convergen o no lo hacen en un tiempo razonable. Se complementa el trabajo con comentarios y esquemas en pseudocódigo a lo largo del mismo, útiles para la implementación (en lenguaje C) que se ha hecho del método. Se ha utilizado la implementación propia de GMRES para ilustrar una gran variedad de ejemplos, que han servido para ir mejorándola hasta llegar a la versión actual (que se incluye en el Anexo). En la memoria del trabajo se presentan algunos ejemplos representativos que pretenden mostrar algunos aspectos relacionados con la convergencia de GMRES en función del espectro de A.

[1]2010 Mathematics Subject Classification: 65Fxx

Agradecimientos

A mi madre Manuela y a mi pareja Triana por apoyarme en mis estudios y no dejarme desfallecer.

A los profesores con los que he coincidido durante la carrera por los conocimientos transmitidos y la calidad de sus clases magistrales.

En especial, al Dr. Arturo Vieiro por las incontables horas dedicadas a mejorar el texto de este trabajo y su predisposición a ayudarme en todo lo que he necesitado.

Índice general

Capítulo 1

Introducción

En este trabajo estudiaremos la resolución de sistemas lineales $Ax = b$ donde $A \in \mathbb{R}^{n\times n}$ y $b \in \mathbb{R}^n$ son conocidos y $x \in \mathbb{R}^n$ es la incógnita. Nos centraremos en el método iterativo GMRES propuesto por primera vez en [6] por Yousef Saad y Martin H. Schultz en 1986. Este método consiste, para cada iterado i, en aproximar la solución al sistema en un subespacio, llamado subespacio de Krylov, que depende de la matriz A, del vector b y del índice i, y que denotaremos por $\mathcal{K}^i(A, b)$. Se tiene que $\mathcal{K}^{i-1}(A, b) \subset \mathcal{K}^i(A, b)$, así GMRES calcula, en cada iterado, una base ortonormal de $\mathcal{K}^i(A, b)$ a partir de la obtenida en el iterado anterior, y encuentra una aproximación de la solución en este subespacio.

Existen otros métodos de Krylov, es decir, métodos de resolución de sistemas lineales que utilizan el subespacio de Krylov. Las diferencias que caracterizan GMRES con los demás son:

- No requiere una estructura específica de la matriz, otros métodos requieren que la matriz A sea simétrica y/o definida positiva.
- Utiliza la reducción a la forma de Hessenberg superior.
- Impone la condición de mínimo residuo para aproximar la solución dentro del subespacio de Krylov.

Una característica de GMRES que comparte con otros métodos de Krylov es que se basa en el producto de A por un vector v. En realidad no se necesita conocer explícitamente los coeficientes de A, simplemente hace falta saber realizar el producto Av. Gran parte del coste computacional de éstos algoritmos viene dado por los productos Av, de esta forma estos métodos son eficaces para matrices con estructura tal que se puede implementar el producto usando un número de operaciones menor al habitual. Éste es el caso de, por ejemplo, las matrices «sparse», en las que la mayoría de sus coeficientes son nulos.

Por las consideraciones anteriores, GMRES suele asociarse a la resolución de sistemas lineales $Ax = b$ de dimensión muy grande, pero con la característica de estar definidos por una matriz A «sparse» de la que no podemos garantizar que sea simétrica ni definida positiva (o negativa).

En la Sección 2 presentaremos el subespacio de Krylov, motivaremos su uso, y abordaremos el problema de encontrar una base ortonormal que se utilizará en los siguientes pasos de GMRES. Los algoritmos discutidos para calcular dicha base son la iteración de

Arnoldi (Algoritmo 1), Arnoldi con reortogonalización (Algoritmo 2) y Lanczos (Algoritmo 3). Los tres se basan en los mismos principios, explicados a lo largo de la sección.

En la Sección 3 nos centraremos en como aproximar la solución del sistema en el subespacio de Krylov. Veremos que podemos reducir este problema a un problema de mínimos cuadrados de fácil solución. Una vez resueltos los aspectos teóricos estaremos en condiciones de proponer una implementación completa del algoritmo GMRES (Algoritmo 4) justificando todos sus pasos. En esta misma sección haremos un análisis del coste computacional y de memoria requerido, y propondremos una alternativa llamada Restarted GMRES o GMRES(m) (Algoritmo 5) para solucionar posibles problemas de memoria con sistemas de dimensión muy grande.

Se puede garantizar que GMRES converge en, como mucho, n iterados. Sin embargo, la convergencia de GMRES(m) no está garantizada, existiendo ejemplos explícitos en los que se produce estancamiento ([2]). Ésto motiva el estudio que se lleva a cabo en la Sección 4, donde se presentan algunos resultados teóricos que se ilustran en varios experimentos numéricos. En los diversos ejemplos considerados se observa la complejidad que hay detrás de la convergencia de GMRES(m).

Se ha realizado una implementación (en lenguaje C) de GMRES y GMRES(m) para poder llevar a cabo los experimentos numéricos realizados. En el Anexo se puede consultar el código básico que se ha desarrollado para este fin.

Capítulo 2

Subespacio de Krylov

GMRES es un algoritmo iterativo que en el iterado i-ésimo aproxima la solución en un subespacio de $\mathbb{R}^n$ de dimensión i generado por la matriz A y el vector b.

Antes de entrar en los detalles del algoritmo GMRES merece la pena estudiar este subespacio llamado subespacio de Krylov, que además nos servirá para saber «que hace» GMRES. Aunque será en la siguiente sección donde veremos «como lo hace».

Definición 2.0.1. *Dados $A \in \mathbb{R}^{n\times n}$ y $v \in \mathbb{R}^n$, se define el subespacio de Krylov i-dimensional como*

$$\mathcal{K}^i(A, v) = \langle v, Av, A^2v, ..., A^{i-1}v\rangle \tag{2.0.1}$$

Es decir, el subespacio generado por los vectores $v, Av, A^2v, ..., A^{i-1}v$.

2.1. Dimensión del subespacio de Krylov

En general no podemos garantizar que la dimensión de $\mathcal{K}^i(A, v)$ sea igual a i. De hecho, más adelante veremos que GMRES terminará cuando se produzca $\dim(\mathcal{K}^i(A, v)) < i$ (Proposición 3.1.11). Por este motivo excluimos el caso $\dim(\mathcal{K}^i(A, v)) < i$ en la construcción del algoritmo.

A continuación exponemos algunas consideraciones sobre la dimensión del subespacio de Krylov.

Observación 2.1.1. Los vectores de $\mathcal{K}^i(A, v)$ pueden ser vistos como los vectores $x \in \mathbb{R}^n$ que pueden escribirse como $x = p(A)v$ con $p(A)$ polinomio de grado menor o igual que $i-1$. Los coeficientes de $p(A)$ son los coeficientes de x en la base $\langle v, Av, A^2v, ...A^{i-1}v\rangle$ de $\mathcal{K}^i(A, v)$.

Definición 2.1.2. *Definimos el polinomio mínimo del vector v respecto a la matriz A, como el poliomio $p(x)$ mónico no nulo de grado menor tal que $p(A)v = 0$.*

Definición 2.1.3. *Definimos el grado del vector v respecto a la matriz A como el grado del polinomio mínimo de v respecto a A. Lo denotaremos como grado(A, v).*

Proposición 2.1.4. *Sea $\mu = grado(A, v)$, entonces se cumple*

1) *$\mathcal{K}^i(A, v)$ tiene dimensión i para todo $i \leq \mu$.*

II) $\mathcal{K}^i(A,v) = \mathcal{K}^\mu(A,v)$ *para todo* $i > \mu$.

Demostración.

I) Si $i \leq \mu$ entonces no existe polinomio $p(x)$ de grado menor o igual que $i-1$ tal que $p(A)v = 0$ y por tanto $v, Av, ..., A^{i-1}$ son linealmente independientes.

II) Sea $p(A) = \alpha_0 I + \alpha_1 A + ... + \alpha_{\mu-1}A^{\mu-1} + A^\mu$ el polinomio mímio de v respecto a A. Sea $q(x) = -(\alpha_0 I + \alpha_1 x + ... + \alpha_{\mu-1}x^{\mu-1})$. Notemos que $q(x)$ tiene grado $\mu - 1$ y

$$q(A)v = A^\mu v, \tag{2.1.1}$$

ya que $p(A)v = 0$. Por inducción sobre i. Si $i = \mu + 1$, por (2.1.1) tenemos que $A^\mu v$ es combinación lineal de $v, Av, ..., A^{\mu-1}v$ y, por tanto, $\mathcal{K}^{\mu+1}(A,v) = \mathcal{K}^\mu(A,v)$. Supongamos que $\mathcal{K}^{\mu+m}(A,v) = \mathcal{K}^\mu(A,v)$ entonces

$$A^{\mu+m}v = A^m A^\mu v = A^m q(A)v \in \mathcal{K}^{\mu+m}(A,v) = \mathcal{K}^\mu(A,v),$$

donde se sigue que $\mathcal{K}^{\mu+m+1}(A,v) = \mathcal{K}^\mu(A,v)$.

□

El siguiente corolario es consecuencia directa de la Proposición 2.1.4.

Corolario 2.1.5. *dim*$(\mathcal{K}^i(A,v)) = \text{mín}(i, grado(A,v))$.

Corolario 2.1.6. *La dimensión de* $\mathcal{K}^i(A,v)$ *es* i *si, y solo si,* $\mu = grado(A,v) \geq i$.

Demostración. Primero probemos la implicación hacia la derecha. Por la segunda parte de la Proposición 2.1.4, si $\mu < i$ entonces $\dim(\mathcal{K}^i(A,v)) = \mu < i$. La otra implicación es consecuencia directa de la primera parte de la Proposición 2.1.4. □

Observación 2.1.7. Una consecuencia del teorema del Cayley-Hamilton es que el grado de v respecto a A no supera la dimensión n de A ya que si $p(A) = 0$ entonces $p(A)v = 0$. Por el corolario 2.1.5 la dimensión de $\mathcal{K}^i(A,v)$ no supera la dimensión de A para ningún i. Notamos que esto es evidente ya que $m > n$ vectores no nulos de $\mathbb{R}^n$ son linealmente dependientes.

2.2. Motivación del subespacio de Krylov

En este apartado intentaremos mostrar porque es natural buscar la solución de un sistema lineal en los subespacios de Krylov. Las ideas principales han sido extraídas de [8].

Consideramos $A \in \mathbb{R}^{n\times n}$ regular y $b \in \mathbb{R}^n$, buscamos $x \in \mathbb{R}^n$ que satisface $Ax = b$. Supongamos que $\tilde{x}$ cumple la igualdad, entonces,

$$A\tilde{x} = b \Longrightarrow 0 = b - A\tilde{x} \Longrightarrow \tilde{x} = \tilde{x} + b - A\tilde{x},$$

de donde se obtiene el siguiente método iterativo,

$$x_{i+1} = x_i + b - Ax_i = x_i + r_i, \tag{2.2.1}$$

donde $r_i = b - Ax_i$.

Observación 2.2.1. Hemos supuesto A regular con lo que aseguramos la existencia y unicidad de la solución.

Proposición 2.2.2. *Se tiene $r_{i+1} = (I - A)r_i$ donde I denota la matriz identidad y, por tanto, $r_i = (I - A)^i r_0$.*

Demostración.

$$\begin{aligned} r_{i+1} &= b - Ax_{i+1} = b - A(x_i + r = b - Ax_i - Ar_i = b - Ax_i - Ar_i \\ &= r_i - Ar_i = (I - A)r_i \end{aligned}$$

□

Observación 2.2.3. Si $\|I - A\|_2 < 1$ entonces la aplicación $r_i \mapsto r_{i+1}$ es contractiva y el método (2.2.1) converge.

Expresemos ahora el iterado $(i+1)$-ésimo en función de x_0 y r_0,

$$\begin{aligned} x_{i+1} &= x_0 + r_0 + ... + r_i \\ &= x_0 + r_0 + (I - A)r_0 + (I - A)^2 r_0 + ... + (I - A)^i x_0 \\ &= x_0 + \sum_{j=0}^{i} (I - A)^j r_0. \end{aligned}$$

Sea $\mathcal{P}_i(A)$ el conjunto de los polinomios de grado i sobre A. Observamos que,

$$(I - A)^j \in \mathcal{P}_j(A) \Rightarrow \sum_{j=0}^{i} (I - A)^j r_0 \in \mathcal{P}_i(A)r_0,$$

y en consecuencia,

$$\begin{aligned} x_{i+1} \in x_0 + \mathcal{P}_i(A) &\Rightarrow x_{i+1} \in x_0 + \langle r_0, Ar_0, A^2 r_0, ..., A^i r_0 \rangle \\ &\Rightarrow x_{i+1} \in x_0 + \mathcal{K}^{i+1}(A, r_0). \end{aligned} \tag{2.2.2}$$

Proposición 2.2.4. *Sin pérdida de generalidad se puede suponer $x_0 = 0$. En tal caso $r_0 = b$.*

Demostración. Si $x_0 \neq 0$ escribimos $x = y + x_0$ y se tiene

$$A(y + x_0) = b \Leftrightarrow Ay = b - Ax_0 = \tilde{b},$$

con $y_0 = 0$. Si $x_0 = 0$ entonces,

$$r_0 = b - Ax_0 = b.$$

□

De esta forma consideramos $x_0 = 0$ y $r_0 = b$, con lo que tomando la expresión (2.2.2) tenemos

$$x_{i+1} \in \mathcal{K}^{i+1}(A, b).$$

El método iterativo (2.2.1) recibe el nombre de iteración de Richardson simple y si $x_0 = 0$, se reduce a

$$\begin{aligned} x_0 &= 0, \qquad x_{i+1} = x_i + r_i, \\ r_0 &= b, \qquad r_{i+1} = b - Ax_{i+1} = (I - A)r_i. \end{aligned}$$

Podríamos considerar otro método a partir de éste incluyendo una sucesión de parámetros α_i obteniendo la siguiente iteración

$$\begin{array}{ll} x_0 = 0, & x_{i+1} = x_i + \alpha_i r_i, \\ r_0 = b, & r_{i+1} = b - Ax_{i+1} = (I - \alpha_i A) r_i. \end{array} \tag{2.2.3}$$

Notamos que la condición de convergencia (observación 2.2.3) depende de la elección de los α_i. La iteración (2.2.3) al igual que (2.2.1) es tal que $x_{i+1} \in \mathcal{K}^{i+1}(A, b)$.

Proposición 2.2.5. *Dada una sucesión de parámetros α_i arbitraria, si consideramos el método iterativo definido en (2.2.3) entonces $x_{i+1} \in \mathcal{K}^{i+1}(A, b)$.*

Demostración.

$$x_{i+1} = x_i + \alpha_i r_i = ... = x_0 + \alpha_0 r_0 + ... + \alpha_i r_i.$$

Para $1 \leq j \leq i$ tenemos,

$$\begin{aligned} r_j &= (I - \alpha_{j-1}A) r_{j-1} = (I - \alpha_{j-1}A)(I - \alpha_{j-2}A) r_{j-2} \\ &= (I - \alpha_{j-1}A)(I - \alpha_{j-2}A)...(I - \alpha_0 A) r_0 \\ &= P_j(A) r_0, \end{aligned}$$

donde $P_j \in \mathcal{P}_j(A)$. Entonces,

$$x_{i+1} = x_0 + P_i'(A) r_0 = P_i'(A) b \in \mathcal{K}^{i+1}(A, b) = \langle b, Ab, ..., A^i b \rangle,$$

donde $P_i' \in \mathcal{P}_i(A)$. □

El problema es que dado un sistema $Ax = b$ con A y b arbitrarios, no conocemos, a priori, el valor de los α_i para definir un método convergente según (2.2.3). La idea que se propone a continuación es escoger los parámetros α_i de manera que minimicen el residuo $r = b - Ax$.

Definimos $\alpha^{(i-1)} = (\alpha_0, ..., \alpha_{i-1}) \in \mathbb{R}^i$ y consideramos la aplicación

$$\begin{array}{l} F \colon (\mathbb{R}^{n \times n}, \mathbb{R}^n, \mathbb{R}^i) \longrightarrow \mathbb{R} \\ \qquad (A, b, \alpha^{(i-1)}) \mapsto F(A, b, \alpha^{(i-1)}) = \alpha_i \end{array} \tag{2.2.4}$$

que a A, b y $\alpha^{(i-1)}$ les asigna el valor α_i tal que $\alpha^{(i)} = (\alpha_0, ..., \alpha_{i-1}, \alpha_i)$ minimice $\|r_{i+1}\|_2 = \|b - Ax_{i+1}\|_2$ obteniendo x_{i+1} según el método (2.2.3).

De esta forma podemos definir los parámetros del método (2.2.3) como,

$$\text{escogemos } \alpha_0 \text{ que minimiza } \|b - A(\alpha_0 b)\|_2 \text{ y } \alpha_i = F(A, b, \alpha^{(i-1)}). \tag{2.2.5}$$

Observación 2.2.6. Por la Proposición 2.2.5, $x_{i+1} \in \mathcal{K}^{i+1}(A, b)$ para todo i considerando el método (2.2.5).

No conocemos los parámetros que minimizan el residuo para definir el método (2.2.5), pero sí sabemos que $x_{i+1} \in \mathcal{K}^{i+1}(A, b)$ por la Observación 2.2.6. En (2.2.5) se fija $\alpha^{(i-1)}$ y se busca α_i que minimice el residuo. El método GMRES se basa en el mismo principio, pero considerando x_{i+1} el vector del subespacio de Krylov i-dimensional que minimiza el residuo. Es decir, GMRES es un método iterativo que calcula el espacio de Krylov $\mathcal{K}^{i+1}(A, b)$ a partir de $\mathcal{K}^i(A, b)$ y define x_{i+1} como el elemento de $\mathcal{K}^{i+1}(A, b)$ que minimiza la norma del residuo $\|b - Ax_{i+1}\|_2$.

2.2.1. Polinomio mínimo y la inversa de A

Otra forma de motivar el uso del subespacio de Krylov para resolver sistemas lineales es obteniendo la inversa de A a partir de su polinomio mínimo. La matriz A es invertible ya que consideramos sistemas en los que la solución existe y es única.

Proposición 2.2.7. *Sea $A \in \mathbb{R}^{n\times n}$ de rango máximo. Sea $m(x) = a_0 + a_1 x + ... + a_m x^m$, $a_0 \neq 0$, $a_m \neq 0$, el polinomio mínimo de A de grado $m \leq n$. Entonces,*

$$A^{-1} = -\left(\frac{a_1}{a_0} I + \frac{a_2}{a_0} A + ... + \frac{a_m}{a_0} A^{m-1}\right). \tag{2.2.6}$$

Es decir, la inversa de A viene dada por la evaluación en A de un polinomio de grado $m-1$.

Demostración. La demostración es inmediata utilizando la propiedad del polinomio mínimo, $m(A) = 0$, multiplicando A^{-1} por la derecha, $m(A)A^{-1} = 0$, y finalmente despejando A^{-1}. □

Corolario 2.2.8. *Sea $m \leq n$ el grado del polinomio mínimo de A, entonces la solución al sistema $Ax = b$ pertenece a $\mathcal{K}^m(A, b)$.*

Demostración. Podemos escribir la solución de $Ax = b$ como $x = A^{-1}b$. Tomando la expresión (2.2.6),

$$A^{-1}b = -\left(\frac{a_1}{a_0} b + \frac{a_2}{a_0} Ab + ... + \frac{a_m}{a_0} A^{m-1} b\right) \in \mathcal{K}^m(A, b) = \langle b, Ab, ..., A^{m-1}b\rangle.$$ □

Hemos definido GMRES como el método iterativo que en cada iterado i calcula $\mathcal{K}^i(A, b)$ y la aproximación $x_i \in \mathcal{K}^i(A, b)$ que minimiza $\|b - Ax\|_2$ en $\mathcal{K}^i(A, b)$. Aunque aún no sabemos como lo hace. Lo que sí sabemos por el Corolario 2.2.8 es que si m es la dimensión del polinomio mínimo de A entonces GMRES converge en como máximo $m \leq n$ iterados.

La convergencia en m iterados es independiente del vector b. Si recordamos la definición 2.1.2 podemos dar con el número exacto de iterados en los que convergirá GMRES. Si $m' \leq m \leq n$ es el grado del polinomio mínimo del vector b respecto a la matriz A, entonces GMRES converge en exactamente m' pasos. Para demostrarlo solo tenemos que repetir los argumentos que llevan al Corolario 2.2.8 utilizando en este caso el polinomio mínimo de b respecto A,

$$m_b(x) = a_0 + a_1 x + ... + a_{m'} x^{m'},\ a_0 \neq 0,\ a_{m'} \neq 0,$$

calculando $A^{-1}b$,

$$A^{-1}b = -\left(\frac{a_1}{a_0} b + \frac{a_2}{a_0} Ab + ... + \frac{a_{m'}}{a_0} A^{m'-1} b\right) \in \mathcal{K}^{m'}(A, b),$$

y observando que $A^{m'-1}b \neq 0$ dado que $m_b(x)$ es el polinomio mínimo de b respecto A, con lo que,

$$A^{-1}b \notin \mathcal{K}^i(A, b) \text{ para } i < m' - 1.$$

2.3. Ortogonalización del subespacio de Krylov

Esta sección está inspirada en los apartados 3.3 de [8] y 6.3 de [5].

Siguiendo con las conclusiones finales del apartado anterior vamos a explorar el subespacio $\mathcal{K}^i(A,b)$ con $i \leq n$. Descartando el caso $grado(A,v) < i$ nombrado en el apartado 2.1, los vectores $b, Ab, ..., A^{i-1}b$ forman una base de este subespacio. Esta base nos presenta un problema numérico.

La sucesión de vectores A^ib tiende al vector propio dominante a medida que incrementamos i (podemos consultar los detalles en [1] sección 4.4.1 o en [3] sección 7.3.1). Obsérvese la relación con el método de la potencia para hallar el vector propio dominante de la matriz A.

Si no disponemos de una aritmética exacta, como es en el caso numérico, habrá problemas a la hora de tener determinado el subespacio $\mathcal{K}^i(A,b)$ mediante la base $b, Ab, ..., A^{i-1}b$. En tal caso, es conveniente ortogonalizar la base. El proceso de ortogonalización puede llevarse a cabo mediante la iteración de Arnoldi. Veremos dos versiones (Algoritmos 1 y 2).

El algoritmo de Arnoldi construye una base ortogonal en el caso general, y por tanto es el usado por GMRES y lo analizaremos en detalle. Si A es simétrica es mejor usar una simplificación de Arnoldi llamada algoritmo de Lanczos (Algoritmo 3). El algoritmo de Lanczos es utilizado por otros métodos de Krylov «como MINRES» que intenta explotar la simetría.

Es importante mencionar que la iteración de Arnoldi hace algo más que encontrar una base ortonormal del subespacio de Krylov i-dimensional. Calcula una matriz Hessenberg superior, que denotaremos $\widetilde{H}_i$, y que jugará un papel importante en los siguientes pasos del algoritmo GMRES (aunque aún no estamos en disposición de justificar esta importancia). Gran parte del contenido de la siguiente sección se centra en encontrar las propiedades de esta matriz $\widetilde{H}_i$ y justificar su cálculo en el algoritmo de Arnoldi, junto con el ya motivado cálculo de los vectores ortonormales que generarán el subespacio de Krylov i-dimensional. Esta base ortonormal de $\mathcal{K}^i(A,b)$ aparecerá en forma de matriz ortogonal $Q_i = \begin{bmatrix} q_1 & \cdots & q_i \end{bmatrix}$ donde $\mathcal{K}^i(A,b) = \langle q_1, ..., q_i \rangle$.

2.3.1. Iteración de Arnoldi

Intentemos llegar al algoritmo de Arnoldi de forma constructiva.

Sea n la dimensión del sistema $Ax = b$. Dado $i \leq n$ consideremos la base de $\mathcal{K}^i(A,b)$ formada por los vectores

$$u_j = A^{j-1}b, \qquad 1 \leq j \leq i,$$

que cumplen,

$$u_j = Au_{j-1}, \qquad 2 \leq j \leq i,$$

Definimos la matriz $U_i \in \mathbb{R}^{n \times i}$ como la matriz con columnas $u_1, ..., u_i$. De forma compacta, escribimos

$$U_i = \begin{bmatrix} u_1 & \cdots & u_i \end{bmatrix}.$$

Observación 2.3.1. De momento supondremos que $\dim\langle u_1, ..., u_i \rangle = i$ descartando el caso $grado(A,v) < i$. El caso en que la dimensión de $\mathcal{K}^i(A,b)$ es menor que i la trataremos

más adelante y no es necesario considerarla ahora puesto que significa la parada del método GMRES que pretendemos construir (Proposición 3.1.11).

Los siguientes tres Teoremas 2.3.2, 2.3.3 y 2.3.5 proponen una factorización $U_i = Q_i R_i$ donde $Q_i \in \mathbb{R}^{n\times i}$, al igual que $U_i \in \mathbb{R}^{n\times i}$, tiene por columnas una base de $\mathcal{K}^i(A,b)$, pero en su caso esta base es ortonormal. Veremos que $R_i \in \mathbb{R}^{i\times i}$ es una matriz triangular superior. Estan basados en los Teoremas 5.2.1 y 5.2.2 de [3].

Teorema 2.3.2. *(Factorización QR). Si $M \in \mathbb{R}^{n\times i}$, $i \leq n$, entonces existe una matriz ortogonal $Q \in \mathbb{R}^{n\times n}$ y una matriz triangular superior $R \in \mathbb{R}^{n\times i}$ tal que $M = QR$.*

Demostración. La prueba es consecuencia de aplicar las transformaciones de Householder (sección 5.1.2 de [3]) a las columnas de la matriz M.

Si $M = (m_{kj})$ con $1 \leq k \leq n$ y $1 \leq j \leq i$, $P'_k \in \mathbb{R}^{(n-k)\times(n-k)}$ es la transformación de Householder asociada al vector $[m_{k,k+1}, m_{k,k+2}, ..., m_{kn}]^T$,

$$P_k = \begin{bmatrix} I_k & \\ & P'_k \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad \text{y} \quad P = P_i ... P_1 \in \mathbb{R}^{n\times n},$$

que es una matriz ortogonal por ser producto de matrices ortogonales. Entonces $PM = R$ con $R \in \mathbb{R}^{n\times n}$ triangular superior por construcción. Multiplicando por $Q = P^T \in \mathbb{R}^{n\times n}$ a ambos lados obtenemos el resultado $M = QR$. □

Teorema 2.3.3. *(Factorización QR fina). Si $M \in \mathbb{R}^{n\times i}$, $i \leq n$, entonces existen una matriz $Q_i \in \mathbb{R}^{n\times i}$ con columnas ortonormales $q_1, ..., q_i$ y una matriz triangular superior $R_i = (r_{kj}) \in \mathbb{R}^{i\times i}$ tal que $M = Q_i R_i$.*

Demostración. Consideramos una factorización QR de M según el Teorema 2.3.2. Sean,

$Q_i = \begin{bmatrix} q_1 & \cdots & q_i \end{bmatrix} \in \mathbb{R}^{n\times i}$, la matriz formada por las i primeras columnas de Q, y

$R_i = \begin{bmatrix} r_{11} & \cdots & r_{1i} \\ & \ddots & \vdots \\ & & r_{ii} \end{bmatrix} \in \mathbb{R}^{i\times i}$, la matriz formada por las i primeras filas de R,

entonces,

$$M = QR = \begin{bmatrix} Q_i & q_{i+1} & \cdots & q_n \end{bmatrix} \begin{bmatrix} R_i \\ 0 \\ \vdots \\ 0 \end{bmatrix} = Q_i R_i,$$

donde las columnas de Q_i son ortonormales por ser columnas de Q, y R_i es triangular superior por construcción. □

Observación 2.3.4. La factorización QR fina del Teorema 2.3.3 es única si consideramos los valores de la diagonal de R_i positivos. No necesitamos este resultado para nuestro propósito pero lo podemos consultar en [3] Theorem 5.2.3.

Teorema 2.3.5. *Si $M = \begin{bmatrix} m_1 & \cdots & m_i \end{bmatrix} \in \mathbb{R}^{n\times i}$, $i \leq n$, tiene rango máximo por columnas (i.e. tiene rango i), y $M = Q_i R_i$ es la factorización QR fina de la matriz M según el Teorema 2.3.3. Entonces*

$$\langle m_1, ..., m_i \rangle = \langle q_1, ..., q_i \rangle$$

y además los elementos de la diagonal de R_i son no nulos, es decir, $r_{kk} \neq 0$ para $1 \leq k \leq i$.

Demostración. Primero veamos que $r_{kk} \neq 0$. Como Q_i tiene rango máximo ya que sus columnas son ortogonales (i.e. rango$(Q_i) = i$), entonces,

$$\text{rango}(R_i) = \text{rango}(Q_i R_i) = \text{rango}(M) = i. \tag{2.3.1}$$

Además $R_i \in \mathbb{R}^{i\times i}$ es triangular superior. Si $r_{kk} = 0$ para algún $1 \leq k \leq i$ entonces rango$(R_i) < i$, en contradicción con (2.3.1).

Ahora veamos $\langle m_1, ..., m_i\rangle = \langle q_1, ..., q_i\rangle$. Como $M = Q_i R_i$ entonces,

$$m_k = \sum_{j=1}^{k} r_{jk} q_j \in \langle q_1, ..., q_i\rangle,$$

y por tanto,

$$\langle m_1, ..., m_i\rangle \subseteq \langle q_1, ..., q_i\rangle. \tag{2.3.2}$$

Como M es de rango máximo por columnas al igual que Q_i entonces,

$$\dim\langle m_1, ..., m_i\rangle = \dim\langle q_1, ..., q_i\rangle = i. \tag{2.3.3}$$

De (2.3.3) y (2.3.2) se sigue que,

$$\langle m_1, ..., m_i\rangle = \langle q_1, ..., q_i\rangle.$$

□

Ahora queremos aplicar la factorización QR fina a la matriz U_i, pero antes veamos el siguiente resultado.

Proposición 2.3.6. *$AU_i = U_i B_i + u_{i+1} e_i^T$ siendo e_i el i-ésimo vector de la base canónica en $\mathbb{R}^i$ y B_i la matriz cuadrada de dimensión i tal que sus elementos cumplen $b_{j+1,j} = 1$, $1 \leq j \leq i-1$, y el resto son nulos.*

Demostración. Notemos que,

$$U_i = \begin{bmatrix} b & Ab & ... & A^{i-1}b \end{bmatrix} \Rightarrow AU_i = \begin{bmatrix} Ab & A^2b & ... & A^ib \end{bmatrix},$$

$$\begin{aligned} B_i = \begin{bmatrix} e_2 & e_3 & ... & e_i & 0 \end{bmatrix} \Rightarrow U_i B_i &= \begin{bmatrix} U_i e_2 & U_i e_3 & ... & U_i e_i & 0 \end{bmatrix} \\ &= \begin{bmatrix} Ab & A^2b & ... & A^{i-1}b & 0 \end{bmatrix}, \end{aligned}$$

y,

$$u_{i+1} e_i^T = \begin{bmatrix} 0 & 0 & ... & 0 & A^ib \end{bmatrix}.$$

Entonces,

$$U_i B_i + u_{i+1} e_i^T = \begin{bmatrix} Ab & A^2b & ... & A^ib \end{bmatrix} = AU_i$$ □

Ahora consideramos la factorización QR fina de la matriz U_i,

$$U_i = Q_i R_i. \tag{2.3.4}$$

De la Proposición 2.3.6 y la factorización (2.3.4) obtenemos,

$$AQ_i R_i = Q_i R_i B_i + u_{i+1} e_i^T. \tag{2.3.5}$$

Notamos que R_i tiene inversa por el Teorema 2.3.5. Multiplicando por la derecha (2.3.5) por R_i^{-1} obtenemos,

$$AQ_i = Q_i R_i B_i R_i^{-1} + u_{i+1} e_i^T R_i^{-1}. \tag{2.3.6}$$

Proposición 2.3.7. *$R_iB_iR_i^{-1}$ es una matriz Hessenberg superior.*

Demostración. Dado que $R_iR_i^{-1} = I$ y R_i es triangular superior, tenemos que R_i^{-1} también es triangular superior. Con esta consideración resolvemos el producto,

$$R_iB_iR_i^{-1} =$$

$$= \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1i} \\ & r_{22} & \cdots & r_{2i} \\ & & \ddots & \vdots \\ & & & r_{ii} \end{bmatrix} \begin{bmatrix} & & \\ 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \begin{bmatrix} \bar{r}_{11} & \bar{r}_{12} & \cdots & \bar{r}_{1i} \\ & \bar{r}_{22} & \cdots & \bar{r}_{2i} \\ & & \ddots & \vdots \\ & & & \bar{r}_{ii} \end{bmatrix} =$$

$$= \begin{bmatrix} r_{12} & r_{13} & \cdots & r_{1i} & 0 \\ r_{22} & r_{23} & \cdots & r_{2i} & 0 \\ & r_{33} & & \vdots & \vdots \\ & & \ddots & \vdots & \vdots \\ & & & r_{ii} & 0 \end{bmatrix} \begin{bmatrix} \bar{r}_{11} & \bar{r}_{21} & \cdots & \cdots & \bar{r}_{i1} \\ & \bar{r}_{21} & \cdots & \cdots & \bar{r}_{i2} \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & & & & \bar{r}_{ii} \end{bmatrix} =$$

$$= \begin{bmatrix} \bar{h}_{11} & \bar{h}_{12} & \cdots & \cdots & \bar{h}_{1i} \\ \bar{h}_{21} & \bar{h}_{22} & \cdots & \cdots & \bar{h}_{2i} \\ & \bar{h}_{32} & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \bar{h}_{i,i-1} & \bar{h}_{i,i} \end{bmatrix} = \bar{H}_i$$

□

Sea $\bar{H}_i = R_iB_iR_i^{-1}$, y sustituyendo en (2.3.6) se tiene

$$AQ_i = Q_i\bar{H}_i + u_{i+1}e_i^TR_i^{-1}. \tag{2.3.7}$$

El producto del segundo sumando es

$$u_{i+1}e_i^TR_i^{-1} = \begin{bmatrix} u_{i+1,1} \\ \vdots \\ u_{i+1,n} \end{bmatrix} [0, \cdots, 0, 1]R_i^{-1}$$

$$= \begin{bmatrix} 0 & \cdots & 0 & u_{i+1,1} \\ \vdots & & \vdots & \vdots \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & u_{i+1,n} \end{bmatrix} \begin{bmatrix} \bar{r}_{11} & \bar{r}_{12} & \cdots & \bar{r}_{1i} \\ & \bar{r}_{22} & \cdots & \bar{r}_{2i} \\ & & \ddots & \vdots \\ & & & \bar{r}_{ii} \end{bmatrix}$$

$$= \bar{r}_{ii} \begin{bmatrix} 0 & \cdots & 0 & u_{i+1,1} \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & u_{i+1,n} \end{bmatrix}$$

$$= \bar{r}_{ii}u_{i+1}e_i^T.$$

Dado que $R_iR_i^{-1} = I$ y R_i es triangular superior, tenemos que $r_{ii}\bar{r}_{ii} = 1$ y entonces $\bar{r}_{ii} = \frac{1}{r_{ii}}$. Sustituyendo en la expresión (2.3.7) obtenemos

$$AQ_i = Q_i\bar{H}_i + \frac{1}{r_{ii}}u_{i+1}e_i^T. \tag{2.3.8}$$

En resumen, a partir de la Proposición 2.3.6 se tiene que

$$\begin{aligned} AU_i = U_iB_i + u_{i+1}e_i^T &\Longrightarrow AQ_iR_i = Q_iR_iB_i + u_{i+1}e_i^T \quad \text{Factorización } QR \text{ fina de } U_i \\ &\Longrightarrow AQ_i = Q_iR_iB_iR_i^{-1} + u_{i+1}e_i^TR_i^{-1} \\ &\Longrightarrow AQ_i = Q_i\bar{H}_i + \frac{1}{r_{ii}}u_{i+1}e_i^T \quad \text{Proposición 2.3.7.} \end{aligned}$$

Notemos que se ha sustituido la matriz U_i por la matriz Q_i, con columnas ortonormales y que generan, al igual que las columnas de U_i, el subespacio $\mathcal{K}^i(A,b)$.

El objetivo ahora es sustituir el vector u_{i+1} por q_{i+1} donde q_{i+1} sea ortonormal a $q_1, ..., q_i$ «y junto» a ellos formen una base ortonormal de $\mathcal{K}^{i+1}(A,b)$. Si conseguimos justificar la factorización QR fina de U_{i+1} de la siguiente forma,

$$\begin{aligned} U_{i+1} &= Q_{i+1}R_{i+1}, \\ Q_{i+1} &= \begin{bmatrix} Q_i & q_{i+1} \end{bmatrix}, \\ R_{i+1} &= \left[\begin{array}{c|c} & r_{1,i+1} \\ R_i & \vdots \\ & r_{i,i+1} \\ \hline & r_{i+1,i+1} \end{array}\right], \end{aligned}$$

obtendremos un resultado que justificará la propiedad iterativa en la iteración de Arnoldi (algoritmo 1).

Proposición 2.3.8. *Si $U_i = Q_iR_i$ y $U_{i+1} = Q_{i+1}R_{i+1}$ son las factorizaciones QR finas de las matrices U_i y U_{i+1} respectivamente, entonces,*

$$\begin{aligned} Q_{i+1} &= \begin{bmatrix} Q_i & q_{i+1} \end{bmatrix}, \\ R_{i+1} &= \begin{bmatrix} R_i & r' \\ & r_{i+1,i+1} \end{bmatrix}, \\ u_{i+1} &= Q_ir' + r_{i+1,i+1}q_{i+1}, \end{aligned}$$

donde u_{i+1} es la última columna de U_{i+1}, $r' \in \mathbb{R}^i$, $r_{i+1,i+1} \in \mathbb{R}$ y $q_{i+1} \in \mathcal{K}^{i+1}(A,b)$ es ortonormal a los vectores $q_1, ..., q_i$ que forman las columnas de Q_i.

Demostración. Como sabemos que $U_i = Q_iR_i$ tenemos que,

$$U_{i+1} = \begin{bmatrix} U_i & u_{i+1} \end{bmatrix} = \begin{bmatrix} Q_i & q_{i+1} \end{bmatrix} \begin{bmatrix} R_i & r' \\ 0 & r_{i+1,i+1} \end{bmatrix} = Q_{i+1}R_{i+1}$$

es la factorización QR fina de la matriz U_{i+1}.

Así,

$$u_{i+1} = \begin{bmatrix} Q_i & q_{i+1} \end{bmatrix} \begin{bmatrix} r' \\ r_{i+1,i+1} \end{bmatrix} = Q_ir' + r_{i+1,i+1}q_{i+1}.$$

□

Según la Proposición 2.3.8, sustituyendo $u_{i+1} = Q_ir' + r_{i+1,i+1}q_{i+1}$ en la expresión (2.3.8),

$$\begin{aligned} AQ_i &= Q_i(\bar{H}_i + \frac{1}{r_{ii}}r'e_i^T) + \frac{r_{i+1,i+1}}{r_{ii}}q_{i+1}e_i^T \\ &= Q_iH_i + \alpha_iq_{i+1}e_i^T, \end{aligned} \tag{2.3.9}$$

donde $\alpha_i \in \mathbb{R}$ y $H_i \in \mathbb{R}^{i\times i}$ es una matriz Hessenberg superior puesto que por la Proposición 2.3.7 $\bar{H}_i$ lo es.

Como se adelanta al inicio de la sección, la matriz H_i juega un papel importante en los siguientes pasos del algoritmo GMRES. La expresión que usaremos es la (2.3.9), pero la presentaremos de la siguiente manera

$$\begin{aligned} AQ_i &= Q_i H_i + \alpha_i q_{i+1} e_i^T = \begin{bmatrix} Q_i & q_{i+1} \end{bmatrix} \begin{bmatrix} H_i \\ e_i^T \alpha_i \end{bmatrix} \\ &= Q_{i+1} \widetilde{H}_i, \end{aligned} \tag{2.3.10}$$

donde hemos definido $\widetilde{H}_i = \begin{bmatrix} H_i \\ e_i^T \alpha_i \end{bmatrix}$.

Observación 2.3.9. La matriz $\widetilde{H}_i$ es Hessenberg superior puesto que H_i lo es.

La siguiente proposición será clave para justificar el cálculo de la matriz $\widetilde{H}_i$ en la iteración de Arnoldi (Algoritmo 1, Sección 2.3.2).

Proposición 2.3.10. *Si* $\widetilde{H}_i = (h_{kj})_{\substack{1\le k\le i+1 \\ 1\le j\le i}}$ *entonces* $h_{kj} = q_k^T A q_j$.

Demostración. Utilizando que $Q_i^T Q_i = I_{i\times i}$ y $Q_i^T q_{i+1} = 0$ por ser las columnas de $Q_{i+1} = \begin{bmatrix} Q_i & q_{i+1} \end{bmatrix}$ ortonormales, y multipliando por Q_i^T por la izquierda en la expresión (2.3.9) obtenemos

$$Q_i^T A Q_i = Q_i^T Q_i H_i + \alpha_i Q_i^T q_{i+1} e_i^T = H_i, \tag{2.3.11}$$

y, por tanto,

$$q_k^T A q_j = h_{kj} \text{ para } 1 \le k, j \le i. \,. \qquad \square$$

Del mismo modo que en la Proposición 2.3.8 hemos visto que Q_{i+1} y R_{i+1} correspondientes a la factorización QR fina de U_{i+1} contienen a las matrices Q_i y R_i de la factorización QR fina de U_i. También queremos ver que $\widetilde{H}_i$ esta contenida en $\widetilde{H}_{i+1}$ para justificar la propiedad iterativa en la iteración de Arnoldi.

Proposición 2.3.11. *Consideramos* $i+1 < n$. *La matriz* $\widetilde{H}_{i+1}$ *asociada al resultado* $AQ_{i+1} = Q_{i+2}\widetilde{H}_{i+1}$ *de la expresión (2.3.10) tiene la siguiente forma:*

$$\widetilde{H}_{i+1} = \left[\begin{array}{c|c} & h_{1,i+1} \\ \widetilde{H}_i & \vdots \\ & h_{i+1,i+1} \\ \hline 0 & \alpha_{i+1} \end{array}\right]$$

Demostración.

$$AQ_{i+1} = A\begin{bmatrix} Q_i & q_{i+1} \end{bmatrix} = \begin{bmatrix} AQ_i & Aq_{i+1} \end{bmatrix} = \begin{bmatrix} Q_{i+1}\widetilde{H}_i & Aq_{i+1} \end{bmatrix} =$$

$$= \begin{bmatrix} Q_{i+1} & q_{i+2} \end{bmatrix} \left[\begin{array}{c|c} & h_{1,i+1} \\ \widetilde{H}_i & \vdots \\ & h_{i+1,i+1} \\ \hline 0 & \alpha_{i+1} \end{array}\right] = Q_{i+2}\widetilde{H}_{i+1},$$

donde,

$$Aq_{i+1} = Q_{i+1} \begin{bmatrix} h_{1,i+1} \\ \vdots \\ h_{i+1,i+1} \end{bmatrix} + \alpha_{i+1} q_{i+2},$$

por la Proposición 2.3.10. □

Observación 2.3.12. Considerando las Proposiciones 2.3.10 y 2.3.11, el valor α_i de (2.3.10) que forma parte de la definición de la matriz $\widetilde{H}_i = \begin{bmatrix} H_i \\ e_i^T \alpha_i \end{bmatrix}$ es

$$\alpha_i = h_{i,i+1} = q_i^T A q_{i+1},$$

Observación 2.3.13. Si construimos un algoritmo iterativo que en la iteración $(i+1)$-ésima amplie la matriz Q_i obtenida en la iteración i-ésima de forma que $Q_{i+1} = \begin{bmatrix} Q_i & q_{i+1} \end{bmatrix}$; entonces tendremos un algoritmo iterativo que en cada iteración amplie una base ortonormal $q_1, ..., q_i$ de $\mathcal{K}^i(A, b)$ en una base ortonormal $q_1, ..., q_i, q_{i+1}$ de $\mathcal{K}^{i+1}(A, b)$. Podemos pensar en las matrices Q_i como en una única matriz Q que en cada iteración se amplia con una nueva columna. Por la Proposición 2.3.11 tambien podemos pensar en las matrices $\widetilde{H}_i$ como en una única matriz Hessenberg superior que en cada iteración se amplia con una nueva columna $[h_{1i}, ..., h_{i+1,i}]^T$. Por la Proposición 2.3.10 y la observación 2.3.12 el cálculo de sus coeficientes es $h_{ki} = q_k^T A q_i$ para $1 \leq k \leq i+1$.

Si logramos construir este algoritmo, en cada iterado i, además de obtener una base ortonormal de $\mathcal{K}^i(A, b)$ mediante la matriz Q_i, también habremos calculado la matriz $\widetilde{H}_i$ tal que $AQ_i = Q_{i+1}\widetilde{H}_i$ que será importante en los siguientes pasos del método GMRES.

2.3.2. Iteración de Arnoldi con ortogonalización de Gram-Schmidt modificada

Ahora presentamos la iteración de Arnoldi con ortogonalización de Gram-Schmidt modificada (Algoritmo 1). Veremos que este algoritmo realiza lo prometido en la observación 2.3.13. Conviene mencionar que uno puede construir la base ortonormal del subespacio de Krylov usando transformaciones de Housholder. De hecho, esta variante realiza menos operaciones (flops) y requiere menos memoria que la versión basada en Gram-Schmidt. Pero este beneficio es negligible dado que estamos interesados en el caso $i \ll n$ (Sección 3.5). En [5] capítulo 6.3.2 podemos encontrar una discursión más extensa de este hecho.

El algoritmo de Arnoldi basado en Gram-Schmidt se detalla a continuación (Algoritmo 1).

```
1  q_1 = b / ||b||_2;
2  for i = 1, 2, ... do
3  |   t = A q_i;
4  |   for k = 1, ..., i do
5  |   |   h_{k,i} = q_k^T t;
6  |   |   t = t - h_{k,i} q_k;
7  |   end
8  |   h_{i+1,i} = ||t||_2;
9  |   q_{i+1} = t / h_{i+1,i};
10 end
```

Algoritmo 1: Arnoldi con ortogonalización de Gram-Schmidt modificada.

Observación 2.3.14. En el Algoritmo 1 dividimos por $h_{i+1,i}$ sin justificar que sea diferente de cero. De hecho puede ser cero, esto ocurre cuando la dimensión de $\mathcal{K}^{i+1}(A,b)$ sea igual a i, es decir, cuando $q_{i+1}=0$. Veremos más adelante en la Proposición 3.1.11 que, en tal caso, el algoritmo GMRES termina.

Lema 2.3.15. *$q_j = P_{j-1}(A)b$ para todo $1 \leq j \leq n$, donde P_{j-1} es un polinomio de grado $j-1$.*

Demostración. Procedemos por inducción sobre j teniendo en cuenta la iteración del Algoritmo 1. Para $j=1$, $q_1 = P_0(A)b$ donde $P_0(x) = \frac{1}{||b||_2}$. Ahora suponemos que $q_1, ..., q_j$ cumplen la condición y queremos ver que $q_{j+1} = P_j(A)b$. Se tiene

$$\begin{aligned} h_{j+1,j}q_{j+1} &= Aq_j - \sum_{k=1}^{j} h_{k,j}q_k \\ &= AP_{j-1}(A)b - \sum_{k=1}^{j} h_{k,j}P_{k-1}(A)b \\ &= \tilde{P}_j(A)b - \sum_{k=1}^{j} h_{k,j}P_{k-1}(A)b \\ &= P_j(A)b. \end{aligned} \tag{2.3.12}$$

□

Proposición 2.3.16. *([5] Proposición 6.4) Consideremos $i \leq n$ fijado. Los vectores $q_1, ..., q_i$ que se construyen en el Algoritmo 1 forman una base ortonormal del subespacio de Krylov $\mathcal{K}^i(A,b)$.*

Demostración. Los vectores $q_1, ..., q_i$ son ortonormales por construcción ya que siguen el procedimiento de ortogonalización de Gram-Schmidt y están normalizados. Solo hace falta ver que generan el subespacio de Krylov.

Por el Lema 2.3.15, $q_j = P_{j-1}(A)b$, $1 \leq j \leq i$, donde P_{j-1} es un polinomio de grado $j-1$. Entonces,

$$\langle q_1, ..., q_i \rangle = \langle b, Ab, ..., A^{i-1}b \rangle = \mathcal{K}^i(A,b). \tag{2.3.13}$$

□

De la Proposición 2.3.16 concluimos que el Algoritmo 1 cumple el primer propósito que daba comienzo a esta sección, ortogonalizar el subespacio de Krylov. Queda pendiente comprobar que resuelve el cálculo de la matriz de Hessenberg $\widetilde{H}_i$ de la expresión (2.3.10).

Observación 2.3.17. Fijemos $i \leq n$. Con el cálculo de $q_1, ..., q_i$ hemos obtenido la matriz $Q_i = \begin{bmatrix} q_1 & \cdots & q_i \end{bmatrix}$ de la expresión (2.3.9), dado que $q_1, ..., q_i$ son ortonormales y generan $\mathcal{K}^i(A,b)$.

Proposición 2.3.18. *La matriz Hessenberg superior compuesta por los coeficientes h_{ki} definidos en el Algoritmo 1 y considerando nulos los no definidos, en el iterado i-ésimo es la matriz $\widetilde{H}_i$ de la expresión (2.3.10).*

Demostración. Recordemos que $h_{ki} = q_k^T A q_i$ por la observación 2.3.13.

Para $k \leq i$ es justamente el cálculo realizado en la fila 5 del algoritmo teniendo en cuenta que los vectores $q_1, ..., q_i$ son ortonormales como hemos visto en la Proposición 2.3.16.

Para $k = i+1$ tenemos que ver $h_{i+1,i} = q_{i+1}^T A q_i$. Observemos que $t = Aq_i - \sum_{k=1}^{i} h_{ki} q_k$. En la fila 8 del código vemos el cálculo de $h_{i+1,i}$ como el módulo del vector t y junto a la fila 9 llegamos a que,

$$Aq_i - \sum_{k=1}^{i} h_{ki} q_k = t = h_{i+1,i} q_{i+1},$$

y por tanto, como queriamos comprobar,

$$q_{i+1}^T A q_i = q_{i+1}^T (h_{i+1,i} q_{i+1} + \sum_{k=1}^{i} h_{ki} q_k) = h_{i+1,i},$$

nuevamente por la ortonormalidad de los vectores $q_1, ..., q_{i+1}$. □

Observación 2.3.19. Si $A \in \mathbb{R}^{n \times n}$ entonces existe una descomposición $Q^T A Q = H$ donde Q es ortogonal y H es Hessenberg superior. Esta descomposición es llamada descomposición de Hessenberg ([3] sección 7.4.3). Si consideramos la existencia de la descomposición de Hessenberg $Q^T A Q = H$, entonces podemos llegar al mismo resultado $AQ_i = Q_{i+1} \widetilde{H}_i$ de la expresión (2.3.10), considerando Q_i, Q_{i+1} y $\widetilde{H}_i$ submatrices de Q y H. Además, es fácil justificar que los valores son los calculados por la iteración de Arnoldi (algoritmo 1).

Esta justificación es la utilizada en gran parte de la bibliografia que podemos consultar (por ejemplo [1], [7] o [3]). La linea argumental que aquí se ha trabajado, inspirada en [8], utiliza en cambio la existencia de la factorización QR fina (Teorema 2.3.3), que da una idea más clara de que el subespacio generado por las columnas de Q_i y U_i es el mismo (Teorema 2.3.5), hecho que resuelve nuestra intención de ortogonalizar el subespacio de Krylov. Además esta argumentación creemos que es más constructiva respecto al algoritmo y utiliza únicamente las propiedades de U_i sin necesidad de recurrir a la descomposición de Hessenberg.

Observación 2.3.20. Si consideramos la iteración de Arnoldi (Algoritmo 1) la matriz $Q = \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}$ no solo depende de A si no del vector $b = \|b\|_2 q_1$. Dicho de otro modo, la descomposición de Hessenberg $Q^T A Q = H$ no es única.

2.3.3. Iteración de Arnoldi con reortogonalización

Notemos que en la iteración de Arnoldi (algoritmo 1) el vector t representa la componente ortogonal del nuevo vector a ortogonalizar Aq_i. Si esta componente t es muy pequeña respecto a Aq_i podemos no lograr la ortogonalidad deseada. Repitiendo el proceso de Gram-Schmidt en estos casos conseguimos una mejor ortogonalización. Esta idea se presenta en [8]. La condición de reortogonalizado o condición de refinamiento será $\frac{\|t\|_2}{\|Aq_i\|_2} < \kappa$ donde κ es un valor a determinar.

Proposición 2.3.21. *Consideremos el Algoritmo 1 y el paso iterativo i. Definimos $t = Aq_i - \sum_{k=1}^{i} h_{ki} q_k$ tal y como es calculado en el algoritmo. Entoces $\frac{\|t\|_2}{\|Aq_i\|_2} < 1$ si consideramos una aritmética exacta.*

Demostración. Recordemos por la Proposición 2.3.16 y considerando una aritmética exacta, que los vectores $q_1, ..., q_i$ y $t = \|t\|_2 q_{i+1}$ son ortogonales. Entonces t es ortogonal a $\sum_{k=1}^{i} h_{ki} q_k$ y por tanto,

$$\|Aq_i\|_2^2 = \|t + \sum_{k=1}^{i} h_{ki} q_k\|_2^2 = \|t\|_2^2 + \|\sum_{k=1}^{i} h_{ki} q_k\|_2^2 \Longrightarrow \|Aq_i\|_2^2 > \|t\|_2^2.$$

□

A partir de la Proposición 2.3.21 concluimos que el valor κ debe estar entre 0 y 1. Los valores cercanos a 1 serán los que más fuercen la reortogonalización y los valores cercanos a 0 los que menos.

```
1  Escoge un valor para κ ∈ (0, 1), e.g., κ = 0,25;
2  q_1 = b/‖b‖_2;
3  for i = 1, 2, ... do
4  |   t = Aq_i;
5  |   τ = ‖t‖_2;
6  |   for k = 1, ..., i do
7  |   |   h_{k,i} = q_k^T t;
8  |   |   t = t − h_{k,i} q_k;
9  |   end
10 |   if ‖t‖_2/τ ≤ κ then
11 |   |   for k = 1, ..., i do
12 |   |   |   ρ = q_k^T t;
13 |   |   |   t = t − ρ q_k;
14 |   |   |   h_{ki} = h_{ki} + ρ;
15 |   |   end
16 |   end
17 |   h_{i+1,i} = ‖t‖_2;
18 |   q_{i+1} = t/h_{i+1,i};
19 end
```

Algoritmo 2: Arnoldi con reortogonalización de Gram-Schmidt modificada

El algoritmo 2 muestra la iteración de Arnoldi propuesta en el Algoritmo 1 incluyendo la reortogonalización (filas de la 10 a la 16). En las filas 12 y 13 se vuelve a ortogonalizar por Gram-Schmidt y en la fila 14 se actualiza el valor h_{ki}.

Justificamos el cálculo de h_{ki} sumando ρ en la fila 14 del algoritmo. Recordemos de la observación 2.3.13 que $q_k^T A q_i = h_{ki}$. Si notamos los valores antes de la reortogonalización como h'_{ki} entonces después de ésta obtenemos

$$t = Aq_i - \sum_{k=1}^{i} h'_{ki} q_k - \sum_{k=1}^{i} \rho_k q_k = Aq_i - \sum_{k=1}^{i} (h'_{ki} + \rho_k) q_k$$

$$\Longrightarrow 0 = q_k^T t = q_k^T A q_i - (h'_{ki} + \rho_k) \Longrightarrow h_{ki} = q_k^T A q_i = h'_{ki} + \rho_k.$$

2.3.4. Algoritmo de Lanczos

Aunque no vamos a considerar el caso simétrico, por completitud en esta sección explicamos el algoritmo de Lanczos. Éste es una simplificación de la iteración de Arnoldi

considerando que A es una matriz simétrica. La siguiente propiedad nos permite reducir el número de operaciones en este contexto.

Proposición 2.3.22. *Consideremos la iteración de Arnoldi aplicada a una matriz simétrica A. Entonces la matriz H_i de la expresión (2.3.11) es tridiagonal y simétrica.*

Demostración. De la expresión (2.3.11) recordamos que $h_{kj} = q_k^T A q_j$. Para ver que H_i es simétrica tenemos que ver $h_{kj} = h_{jk}$,

$$h_{kj} = q_k^T (Aq_j) = (Aq_j)^T q_k = q_j^T A^T q_k = q_j^T A q_k = h_{jk},$$

donde hemos utilizado que $A = A^T$ por la hipótesis de simetría. Además H_i es Hessenberg superior como vimos en su definición (2.3.9) y en la Proposición 2.3.7. Entonces $0 = h_{kj} = h_{jk}$ para $k > j + 1$ por ser Hessenberg y simétrica, con lo que H_i es tridiagonal. □

La propiedad demostrada en la Proposición 2.3.22 ahorra muchos cálculos en la iteración de Arnoldi (Algoritmo 1). Recordemos que en el paso i-ésimo tenemos que ortogonalizar el vector Aq_i respecto $q_1, ..., q_i$. Utilizando Gram-Schmidt, la componente ortogonal sin normalizar es,

$$t = Aq_i - \sum_{k=1}^{i} h_{ki} q_k,$$

como $0 = h_{ki} = h_{ik}$ para $k > i + 1$ por la Proposición 2.3.22 entonces,

$$t = Aq_i - h_{i-1,i} q_{i-1} - h_{ii} q_i,$$

y de igual forma que en el caso general,

$$h_{i+1,i} = \|t\|_2, \quad q_{i+1} = \frac{t}{h_{i+1,i}}.$$

Notar que tampoco necesitamos calcular $h_{i-1,i}$ en el paso i-ésimo ya que tenemos calculado en el paso $(i-1)$-ésimo el valor de $h_{i,i-1} = h_{i-1,i}$.

Simplificando la iteración de Arnoldi (Algoritmo 1) según las consideraciones anteriores se obtiene el algoritmo de Lanczos (Algoritmo 3) que se detalla a continuación. En su descripción usamos la siguiente notación: $\alpha_k = h_{k,k}$, $\beta_j = h_{j-1,j} = h_{j,j-1}$.

```
1  β_1 = 0;
2  q_0 = 0;
3  q_1 = b/‖b‖_2;
4  for i = 1, 2, ... do
5  |   t = Aq_i − β_i q_{i−1};
6  |   α_i = q_i^T t;
7  |   t = t − α_i q_i;
8  |   β_{i+1} = ‖t‖_2;
9  |   q_{i+1} = t/β_{i+1};
10 end
```

Algoritmo 3: Algoritmo de Lanczos

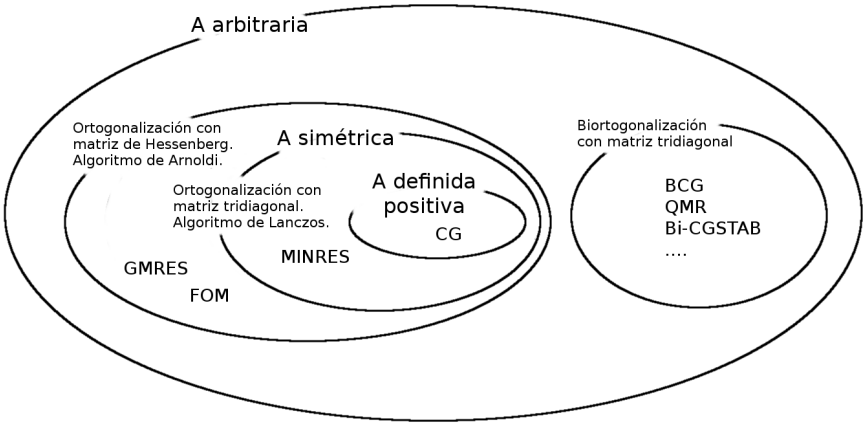

Figura 2.1: Esquema relacionando algunos métodos de Krylov con la estructura de la matriz del sistema a resolver.

2.4. Algunos métodos de Krylov para la resolución de sistemas lineales

Recordemos que nuestro objetivo es solucionar el sistema $Ax = b$. La estrategia a seguir es definir un método iterativo que utilice el subespacio de Krylov i-dimensional en el iterado i-ésimo para obtener una aproximación de la solución. Existen varios métodos dependiendo de la ortogonalización realizada al subespacio de Krylov.

En la sección 2.3 hemos visto la ortogonalización fabricando una matriz de Hessenberg mediante la iteración de Arnoldi (Algoritmo 1) obteniendo el resultado importante de la expresión (2.3.9) $AQ_i = Q_{i+1}\widetilde{H}_i$. Mediante este resultado se construye el método del mínimo residuo generalizado GMRES, en el que entraremos en profundidad, y el método llamado de ortogonalización completa FOM [5] que utiliza la condición de Galerkin en vez de la de mínimo residuo.

Si A es simétrica esta matriz de Hessenberg pasa a ser tridiagonal y podemos simplificar la iteración de Arnoldi. Esta simplificación es el algoritmo de Lanczos (Algoritmo 3). Bajo estas condiciones se construye el método del mínimo residuo MINRES [8]. Si además A es definida positva, tenemos el método de los gradientes conjugados CG que podemos consultar en prácticamente toda la bibliografía facilitada [1], [3], [7], [5] o [8], aunque para seguir a partir de los resultados obtenidos en la sección 2.3 se recomienda la lectura de [5] o [8].

Existen otros métodos utilizando una ortogonalización que no ha sido explicada. Esta ortogonalización es llamada biortogonalización y se basa en una factorización $A = VTV^{-1}$ que envuelve una matriz tridiagonal T, aunque en este caso no se requiere que A sea simétrica. Por contra, la diferencia está en que la matriz V no es ortogonal. Alguno de los métodos que utilizan esta técnica son BCG (gradientes biconjugados), QMR (casi mínimo residuo) o Bi-CGSTAB (BCG estabilizado) los cuales podemos consultar en [5] o [8].

La figura 2.1 presenta los citados métodos según las características de A y el tipo de ortogonalización utilizada.

Capítulo 3

Generalized Minimal RESidual method

3.1. Aproximando la solución en el subespacio de Krylov

En la Sección 2.3 se describe la iteración de Arnoldi (Algoritmo 1) que, para cada iteración i, calcula una base ortonormal $q_1, ..., q_{i+1}$ del subespacio $\mathcal{K}^{i+1}(A, b)$. Recordamos la relación (2.3.10): si $Q_k = \begin{bmatrix} q_1 & \cdots & q_k \end{bmatrix}$ se cumple que,

$$AQ_i = Q_{i+1}\widetilde{H}_i,$$

donde $\widetilde{H}_i \in \mathbb{R}^{(i+1)\times i}$ es una matriz Hessenberg superior que se calcula de forma iterativa (Observación 2.3.13).

Usando la relación (2.3.10), el método GMRES calcula, en cada iteración i de Arnoldi, una aproximación a la solución del sistema $Ax = b$ en $\mathcal{K}^i(A, b)$ que minimiza la norma del residuo. Notaremos esta aproximación como x_i.

Observación 3.1.1. Como las columnas de Q_i son una base de $\mathcal{K}^i(A, b)$ (Teorema 2.3.3), si $x_i \in \mathcal{K}^i(A, b)$ entonces existe $y_i \in \mathbb{R}^i$ tal que $x_i = Q_i y_i$. De esta forma, dada una condición sobre vectores de $\mathcal{K}^i(A, b)$, es equivalente encontrar $x_i \in \mathcal{K}^i(A, b)$ que cumpla la condición a encontrar $y_i \in \mathbb{R}^i$ tal que $Q_i y_i$ cumpla la condición.

3.1.1. Condición de mínimo residuo

Una condición que podemos imponer a $x_i \in \mathcal{K}^i(A, b)$ es que sea mínima la norma del error $(b - Ax_i)$. Esta condición da el nombre al método GMRES que queremos estudiar en profundiad, «método del mínimo residuo generalizado» o en inglés «Generalized Minimal RESidual method». El término «generalizado» aparece dado que el método considera matrices A generales. En la Sección 2.4 se mencionaron otros métodos de Krylov que explotan alguna propiedad de A como por ejemplo la simetría.

Por la Observación 3.1.1 buscamos $y_i \in \mathbb{R}^i$ que minimice

$$\|b - Ax_i\|_2 = \|b - AQ_i y_i\|_2,$$

que, por (2.3.10), es equivalente a minimizar

$$\|b - Q_{i+1}\widetilde{H}_i y_i\|_2. \tag{3.1.1}$$

Ahora el objetivo es transformar el problema de minimizar la expresión (3.1.1) en un problema de mínimos cuadrados fácil de solucionar.

Proposición 3.1.2. *Si* $1 \leq i \leq n$ *entonces*

$$Q_i^T b = \|b\|_2 e_1,$$

donde e_1 *denota el primer vector de la base canónica de* $\mathbb{R}^i$.

Demostración. Recordemos de la iteración de Arnoldi (Algoritmo 1) que al calcular los vectores $q_1, ..., q_i$ tomamos $q_1 = \frac{b}{\|b\|_2}$. Entonces dada la ortonormalidad de los vectores $q_1, ..., q_i$ (Proposición 2.3.16), se tiene $Q_i^T b = \|b\|_2 Q_i^T q_1 = \|b\|_2 e_1$. □

Proposición 3.1.3. *Sean* $Q_{i+1} \in \mathbb{R}^{n\times(i+1)}$ *con columnas* $q_1, ..., q_{i+1}$ *ortonormales entre si, y* $x \in \mathbb{R}^n$. *Entonces* $\|Q_{i+1}^T x\|_2 = \|x\|_2$.

Demostración. Utilizando $Q_{i+1}^T Q_{i+1} = I_{n\times n}$, por ser $q_1, ..., q_{i+1}$ ortonormales,

$$\|Q_{i+1}x\|_2^2 = (Q_{i+1}x)^T(Q_{i+1}x) = x^T Q_{i+1}^T Q_{i+1} x = x^T x = \|x\|_2^2. \qquad \square$$

Las siguientes igualdades muestran como llegamos al problema de mínimos cuadrados. Se tiene que

$$\begin{aligned} \|b - Ax_i\|_2 &= \|b - AQ_i y_i\|_2 = \|b - Q_{i+1}\widetilde{H}_i y_i\|_2 \quad \text{por el resultado (2.3.10)} \\ &= \|Q_{i+1}^T(b - Q_{i+1}\widetilde{H}_i y_i)\|_2 \quad \text{por la Proposición 3.1.3} \\ &= \|\|b\|_2 e_1 - \widetilde{H}_i y_i\|_2 \quad \text{por la Proposición 3.1.2} \end{aligned} \tag{3.1.2}$$

De esta forma hemos pasado de buscar,

$$x_i \in \mathcal{K}^i(A, b) \text{ correspondiente al } \min_{x\in\mathcal{K}^i(A,b)} \|b - Ax\|_2,$$

donde $A \in \mathbb{R}^{n\times n}$, a buscar,

$$y_i \in \mathbb{R}^i \text{ correspondiente al } \min_{y\in\mathbb{R}^i} \|\|b\|_2 e_1 - \widetilde{H}_i y\|_2, \tag{3.1.3}$$

donde $\widetilde{H}_i \in \mathbb{R}^{(i+1)\times i}$ es matriz Hessenberg superior y $x_i = Q_i y_i$.

3.1.2. Problema de mínimos cuadrados

Encontrar el vector de la expresión (3.1.3) se conoce como problema de mínimos cuadrados y es la estrategia que sigue GMRES para encontrar una aproximación en cada iterado.

El problema de mínimos cuadrados, en general, consiste en calcular

$$\min_{x\in\mathbb{R}^n} \|v - Mx\|_2,$$

donde $M \in \mathbb{R}^{m\times n}$, $v \in \mathbb{R}^m$ y $m > n$. Esto puede entenderse como encontrar la mejor aproximación al sistema sobredeterminado $Mx = v$.

Existen varias maneras de resolver el caso general del problema, podemos consultar, por ejemplo, [3] capítulo 5.3, [1] capítulo 3 o [7] lectura 11. Pero nuestro caso tiene ciertas propiedades que hacen interesante tratarlo de forma específica y así vamos a proceder.

Proposición 3.1.4. *(Rotaciones de Givens) Dado un vector $v \in \mathbb{R}^m$ y un índice $1 \leq i \leq m-1$ existe una matriz ortogonal G que llamaremos rotación de Givens tal que el vector producto Gv cumple $(Gv)^T e_j = v^T e_j$ para $j \neq i, i+1$ y $(Gv)^T e_{i+1} = 0$.*

Demostración. Consideremos el ángulo $\theta = \arctan(\frac{v_{i+1}}{v_i})$, los valores $s = \sin(\theta)$ y $c = \cos(\theta)$, y la matriz,

$$G = \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c & s & & & i \\ & & & -s & c & & & i+1 \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix}$$

Obtenemos inmediatamente $(Gv)^T e_j = v^T e_j \quad j \neq i, i+1$. Ahora veamos $(Gv)^T e_{i+1} = 0$,

$$v_{i+1} - \frac{s}{c} v_i = v_{i+1} - \tan(\arctan(\frac{v_{i+1}}{v_i}))v_i = 0,$$

entonces,

$$0 = c\,(v_{i+1} - \frac{s}{c}\,v_i) = c\,v_{i+1} - s\,v_i = (Gv)^T e_{i+1}.$$

Falta ver que G es ortogonal. Como

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} c^2 + s^2 & cs - sc \\ sc - cs & s^2 + c^2 \end{bmatrix} = I_{2\times 2}$$

entonces $GG^T = I_{m\times m}$. □

Observación 3.1.5. Los valores $s = \sin(\arctan(\frac{v_{i+1}}{v_i}))$ y $c = \cos(\arctan(\frac{v_{i+1}}{v_i}))$ pueden calcularse sin utilizar funciones trigonométricas mediante las expresiones,

$$s = \frac{v_{i+1}}{\sqrt{v_i^2 + v_{i+1}^2}} \qquad c = \frac{v_i}{\sqrt{v_i^2 + v_{i+1}^2}}.$$

Observación 3.1.6. Si $v_i = 0$, entonces $\theta = \frac{\pi}{2}$, $c = 0$ y $s = 1$. Si $v_{i+1} = 0$, entonces $\theta = 0$ y $G = I$.

Proposición 3.1.7. *Dada una matriz Hessenberg superior $H \in \mathbb{R}^{(i+1)\times i}$, existen i rotaciones de Givens $G_1, ..., G_i$ tal que,*

$$G_i ... G_1 H = \tilde{R} = \begin{bmatrix} R \\ 0 \end{bmatrix} \begin{matrix} i \times i \\ 1 \times i \end{matrix}$$

donde $R \in \mathbb{R}^{i\times i}$ es una matriz triangular superior.

Demostración. Utilizando la Proposición 3.1.4, notamos $H^{(0)} = H$, $H^{(k)} = G_k G_{k-1} ... G_1 H$ y escogemos las rotaciones de Givens $G_1, ..., G_i$ tal que,

$$(G_j h_j^{(j-1)})_{j+1} = 0, \qquad 1 \leq j \leq i, \tag{3.1.4}$$

donde $h_j^{(j-1)}$ represeta la j-ésima columna de la matriz $H^{(j-1)}$. □

Observación 3.1.8. Por la Proposición 3.1.4 el producto por G_j solamente altera los valores en las filas j y $j+1$ (anulando el valor de la fila $j+1$ y columna j). Por eso empezamos anulando la subdiagonal en la primera columna y repetimos el proceso hasta la última. En este orden conseguimos que G_j no altere las columnas previamente trianguladas.

Ilustramos el procedimiento de la Proposición 3.1.7 considerando $i = 5$.

$$G_5G_4G_3G_2G_1H =$$

$$G_5G_4G_3G_2G_1 \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix} = G_5G_4G_3G_2 \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix} =$$

$$= G_5G_4G_3 \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ & 0 & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix} = ... = G_5 \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ & 0 & * & * & * \\ & & 0 & * & * \\ & & & 0 & * \\ & & & & * \end{bmatrix} = \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ & 0 & * & * & * \\ & & 0 & * & * \\ & & & 0 & * \\ & & & & 0 \end{bmatrix}$$

El primer valor anulado corresponde al índice $j = 1$ en la expresión (3.1.4), el segundo a $j = 2$, y así sucesivamente hasta el último que corresponde a $j = i$. Las matrices que aparecen corresponden de igual forma a las matrices $H^{(j-1)}$ siendo la última $H^{(i)} = \tilde{R}$.

Volvamos a nuestro objetivo de minimizar la expresión (3.1.3). Sean $G_1, ..., G_i$ las rotaciones de Givens que triangulan $\widetilde{H}_i$ según la Proposición 3.1.7. Entonces,

$$\begin{aligned} \min_{y\in\mathbb{R}^i} \|\|b\|_2 e_1 - \widetilde{H}_i y\|_2 &= \min_{y\in\mathbb{R}^i} \|G_i...G_1(\|b\|_2 e_1 - \widetilde{H}_i y)\|_2 \quad \text{por la Proposición 3.1.3} \\ &= \min_{y\in\mathbb{R}^i} \|\tilde{b}_i - \tilde{R}_i y\|_2. \end{aligned}$$

Recordemos de la Proposición 3.1.7 que,

$$\tilde{R}_i = \begin{bmatrix} R_i \\ 0 \end{bmatrix} \begin{matrix} i \times i \\ 1 \times i \end{matrix},$$

donde $R_i \in \mathbb{R}^{i\times i}$ es una matriz triangular superior. Notemos $\tilde{b}_i = G_i...G_1(\|b\|_2 e_1) = [\tilde{b}_{i1}, ..., \tilde{b}_{i,i+1}]^T$. Entonces el problema de mínimos cuadrados se reduce a encontrar $y \in \mathbb{R}^i$ tal que

$$\min_{y\in\mathbb{R}^i} \|\|b\|_2 e_1 - \widetilde{H}_i y\|_2 = \min_{y\in\mathbb{R}^i} \|\tilde{b}_i - \tilde{R}_i y\|_2 = \min_{y\in\mathbb{R}^i} \| \begin{bmatrix} \tilde{b}_{i1} \\ \vdots \\ \tilde{b}_{ii} \\ \tilde{b}_{i,i+1} \end{bmatrix} - \begin{bmatrix} R_i \\ 0 \end{bmatrix} y \ \|_2. \tag{3.1.5}$$

Proposición 3.1.9. *(Existencia, unicidad y error de la solución al problema de mínimos cuadrados) Si la matriz triangular superior $R_i \in \mathbb{R}^{i\times i}$ tiene rango máximo, entonces el problema (3.1.5) tiene una única solución. El error de la aproximación mínimo-cuadrática es $|\tilde{b}_{i,i+1}|$.*

Demostración. Si tomamos la expresión (3.1.5) y notamos $v_y = \tilde{b}_i - \tilde{R}_i y$, observamos que $v_y^T e_{i+1} = \tilde{b}_{i,i+1}$ para todo $y \in \mathbb{R}^i$, es decir, v_y es de la forma

$$v_y = [*, \cdots, *, \tilde{b}_{i,i+1}]^T \quad \text{para todo } y \in \mathbb{R}^i.$$

Si existe $y_i \in \mathbb{R}^i$ tal que,

$$v_{y_i} = [0, \cdots, 0, \tilde{b}_{i,i+1}]^T, \tag{3.1.6}$$

entonces la solución al problema de mínimos cuadrados (3.1.3) es y_i y el error es $|\tilde{b}_{i,i+1}|$,

$$\min_{y \in \mathbb{R}^i} \|v_y\|_2 = \|v_{y_i}\|_2 = |\tilde{b}_{i,i+1}|.$$

Nótese que $y_i \in \mathbb{R}^i$ es la solución del sistema $R_i y = [\tilde{b}_{i1}, \cdots, \tilde{b}_{ii}]^T$ que, al ser R_i de rango máximo, existe y es única. □

Proposición 3.1.10. *(GMRES. Cálculo de la i-ésima aproximación y su error) Considerando la condición de mínimo residuo, la mejor aproximación $x_i \in \mathcal{K}^i(A, b)$ a la solución de $Ax = b$ en el paso i-ésimo de la iteración de Arnoldi (Algoritmo 1) y de su problema de mínimos cuadrados (3.1.3) asociado, se obtiene resolviendo el sistema $R_i y_i = [\tilde{b}_{i1}, \cdots, \tilde{b}_{ii}]^T$ y calculando $x_i = Q_i y_i$. Además, el error de esta aproximación es $|\tilde{b}_{i,i+1}|$.*

Demostración. El enunciado es consecuencia de la Proposicion 3.1.9, la observación 3.1.1 y las igualdades (3.1.2) que llevan al problema de mínimos cuadrados (3.1.3). □

Proposición 3.1.11. *(GMRES. Condición de parada) Consideremos el problema de mínimos cuadrados (3.1.3). Supongamos $q_1, ..., q_i$ no nulos y $q_{i+1} = 0$, o lo que es lo mismo, su norma $h_{i+1,i} = \|q_{i+1}\|_2 = 0$. Entonces la solución del sistema $Ax = b$ es $x_i = Q_i y_i$ donde y_i es la solución al sistema triangular $R_i y = [\tilde{b}_{i1}, \cdots, \tilde{b}_{ii}]^T$.*

Demostración. Por las Proposiciones 3.1.9 y 3.1.10, si R_i tiene rango máximo y $\tilde{b}_{i,i+1} = 0$ entonces existe una única solución y_i que resuelve el problema de mínimos cuadrados con error igual a cero. Además, y_i es la solución del sistema triangular $R_i y_i = [\tilde{b}_{i1}, \cdots, \tilde{b}_{ii}]^T$ que da como solución al sistema $Ax = b$ el vector $x = x_i = Q_i y_i$.

Primero veamos que $R_i \in \mathbb{R}^{i \times i}$ tiene rango máximo. Notemos $G^{(i)} = G_i ... G_1$ el producto de las rotaciones de Givens $G_1, ..., G_i$ que triangulan la matriz $\widetilde{H}_i$. Obviamente $G^{(i)}$ es ortogonal por ser producto de matrices ortogonales, $(G_i ... G_1)^T G_i ... G_1 = G_1^T ... G_i^T G_i ... G_1 = I$. Entonces,

$$\begin{aligned} AQ_i &= Q_{i+1}\widetilde{H}_i, && \text{expresión (2.3.10)} \\ &= Q_{i+1}(G^{(i)})^T G^{(i)} \widetilde{H}_i, && G^{(i)} \text{ es ortogonal} \\ &= Q_{i+1}(G^{(i)})^T \tilde{R}_i. && \text{Proposición 3.1.7} \end{aligned} \tag{3.1.7}$$

Observemos que $Q_{i+1}(G^{(i)})^T$ es una matriz invertible,

$$(Q_{i+1}(G^{(i)})^T)^T (Q_{i+1}(G^{(i)})^T) = G^{(i)} Q_{i+1}^T Q_{i+1} (G^{(i)})^T = I,$$

y por tanto tiene rango máximo. Recordemos que consideramos sistemas $Ax = b$ con solución única, con lo que A también tiene rango máximo. De esta forma,

$$\begin{aligned} rango(\tilde{R}_i) &= rango(Q_{i+1}(G^{(i)})^T\tilde{R}_i), \qquad Q_{i+1}(G^{(i)})^T \text{ tiene rango máximo} \\ &= rango(AQ_i), \qquad \text{expresión (3.1.7)} \\ &= rango(Q_i), \qquad A \text{ tiene rango máximo} \\ &= i. \qquad q_1, ..., q_i \text{ ortonormales y no nulos por hipótesis} \end{aligned} \tag{3.1.8}$$

Finalmente $\tilde{R}_i = \begin{bmatrix} R_i \\ 0 \end{bmatrix}$ con lo que $rango(R_i) = rango(\tilde{R}_i) = i$.

La segunda parte de la demostración consiste en comprobar que $\tilde{b}_{i,i+1} = 0$. Esto es consecuencia de la hipótesis $h_{i+1,i} = 0$,

$$\tilde{R}_i = G_i...G_1\widetilde{H}_i = G_i...G_1 \begin{bmatrix} * & * & * & \cdots & * & * \\ * & * & * & \cdots & * & * \\ & * & * & \cdots & * & * \\ & & * & \ddots & * & * \\ & & & \ddots & * & * \\ & & & & * & * \\ & & & & & 0 \end{bmatrix} =$$

$$= G_i...G_2 \begin{bmatrix} * & * & * & \cdots & * & * \\ 0 & * & * & \cdots & * & * \\ & * & * & \cdots & * & * \\ & & * & \ddots & * & * \\ & & & \ddots & * & * \\ & & & & * & * \\ & & & & & 0 \end{bmatrix} = ... = G_i \begin{bmatrix} * & * & * & \cdots & * & * \\ 0 & * & * & \cdots & * & * \\ & 0 & * & \cdots & * & * \\ & & 0 & \ddots & * & * \\ & & & \ddots & * & * \\ & & & & 0 & * \\ & & & & & 0 \end{bmatrix}.$$

Según la Observación 3.1.6 se tiene $G_i = I$, de donde

$$\tilde{b}_i = G_{i-1}...G_1(\|b\|_2 e_1) = G_{i-1}...G_1 \begin{bmatrix} \|b\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = G_{i-1}...G_2 \begin{bmatrix} * \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} = ... = G_{i-1} \begin{bmatrix} * \\ \vdots \\ * \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} * \\ \vdots \\ * \\ 0 \end{bmatrix},$$

y por tanto $\tilde{b}_{i,i+1} = 0$. □

Observación 3.1.12. Según la Proposición 3.1.11, si encontramos los vectores $q_1, ..., q_i$ iterativamente mediante Arnoldi hasta dar con $q_{i+1} = 0$, estaremos en condiciones de resolver el sistema $Ax = b$. Por este motivo podemos suponer $q_1, ..., q_i$ no nulos, si $q_k = 0$ para algún $1 \leq k \leq i$ el sistema estaría resuelto en el paso k.

Observación 3.1.13. Si $q_{i+1} = 0$ entonces $\mathcal{K}^{i+1}(A, b) = \mathcal{K}^i(A, b)$.

Corolario 3.1.14. *GMRES converge en como máximo n iterados.*

Demostración. Si n es la dimensión del sistema entonces $q_{n+1} = 0$. Esto es consecuencia de que q_{n+1} es la componente ortogonal normalizada de Aq_n respecto a $q_1, ..., q_n$ que ya forman una base de $\mathbb{R}^n$. □

3.2. Consideraciones en la iteración del método

En la iteración de Arnoldi (algoritmo 1) calculamos Q_{i+1} a partir de Q_i calculando q_{i+1}, y $\widetilde{H}_i$ a partir de H_{i-1}^+ calculando los valores de su última columna $[h_{1i}, \cdots, h_{i+1,i}]^T$, como adelantábamos en la Proposición 2.3.11,

$$Q_{i+1} = \begin{bmatrix} Q_i & q_{i+1} \end{bmatrix}, \qquad \widetilde{H}_i = \left[\begin{array}{c|c} & h_{1i} \\ H_{i-1}^+ & \vdots \\ & h_{ii} \\ \hline 0 & h_{i+1,i} \end{array}\right].$$

Podemos pensar las matrices $\widetilde{H}_i$ (y Q_i) como una única matriz que va aumentando su tamaño, añadiendo una nueva columna en cada iteración. Ésto hace que el problema de mínimos cuadrados (3.1.3) se pueda resolver de forma iterativa junto con la iteración de Arnoldi (Algoritmo 1).

Proposición 3.2.1. *Si $G_1, ..., G_{i-1}$ son las rotaciones de Givens que triangulan la matriz H_{i-1}^+ (Proposición 3.1.7) y G_i es la rotación de Givens que anula el elemento $h_{i+1,i}$ de $\widetilde{H}_i$ tras aplicar las rotaciones anteriores (Proposición 3.1.4), entonces $G_1, ..., G_{i-1}, G_i$ son las rotaciones de Givens que triangulan $\widetilde{H}_i$.*

Demostración.

$$\begin{aligned} G_i...G_1\widetilde{H}_i = G_i...G_1 \left[\begin{array}{c|c} & h_{1i} \\ H_{i-1}^+ & \vdots \\ & h_{ii} \\ \hline 0 & h_{i+1,i} \end{array}\right] = G_i \left[\begin{array}{c|c} & r_{1i} \\ \tilde{R}_{i-1} & \vdots \\ & r_{i-1,i} \\ & * \\ \hline 0 & h_{i+1,i} \end{array}\right] = \\ = \left[\begin{array}{c|c} & r_{1i} \\ \tilde{R}_{i-1} & \vdots \\ & r_{i-1,i} \\ & r_{ii} \\ \hline 0 & 0 \end{array}\right] = \tilde{R}_i. \end{aligned} \tag{3.2.1}$$

□

Observación 3.2.2. En la Proposición 3.2.1 se considera que las rotaciones de Givens $(G_i)_{1\leq i\leq i-1}$ son aquellas que anulan la posición $(i+1)$-ésima del vector a las que son aplicadas. Esta definición es independiente de la dimensión, pudiendo ser G_i de cualquier dimensión mayor o igual que $i+1$, dependiendo del vector a la que sea aplicada. Más formalmente, la relación entre las rotaciones de Givens que triangulan H_{i-1}^+ y $\widetilde{H}_i$ es,

$$G_j^{(i-1)} = \begin{bmatrix} G_j^{(i)} & \\ & 1 \end{bmatrix}, \qquad 1 \leq j \leq i-1.$$

Observación 3.2.3. Para resolver el problema de mínimos cuadrados (3.1.3), en la iteración i-ésima, solo necesitamos calcular una rotación de Givens G_i para triangular $\widetilde{H}_i$ ya que las anteriores han sido calculadas en el iterado $(i-1)$-ésimo. Además, como vemos en las igualdades (3.2.1), también aprovechamos el cálculo de la matriz $\tilde{R}_{i-1}$ obtenida

en el iterado $(i-1)$-ésimo para obtener $\tilde{R}_i$. Con lo que en la iteración i-ésima solo hace falta calcular los valores $r_{1i}, ..., r_{ii}$ que corresponden a la última columna de R_i. De esta forma, podemos pensar en las matrices R_i como una única matriz triangular R, que en cada iterado aumenta en uno su dimensión mediante una nueva columna $[r_{1i}, \cdots, r_{ii}]^T$.

Observación 3.2.4. Por la Proposición 3.2.1, si ampliamos la dimensión de $\tilde{b}_{i-1}$ en uno con una última posición nula, entonces

$$\tilde{b}_i = G_i ... G_1(\|b\|_2 e_1) = G_i \tilde{b}_{i-1}.$$

Si $\sin_i$ y $\cos_i$ son el seno y el coseno que definen G_i entonces,

$$\tilde{b}_{ik} = \tilde{b}_{i-1,k}, \qquad 1 \leq k \leq i-1,$$
$$\tilde{b}_{ii} = \cos_i \ \tilde{b}_{i-1,i}, \tag{3.2.2}$$
$$\tilde{b}_{i,i+1} = -\sin_i \ \tilde{b}_{i-1,i}. \tag{3.2.3}$$

De esta forma podemos pensar en los vectores $\tilde{b}_i$ como un único vector $\tilde{b} = [\tilde{b}_1, \cdots, \tilde{b}_i]^T$ que en cada iterado $i+1$ varía el valor de su última posición (3.2.2) y aumenta en uno su dimensión (3.2.3).

Hemos explicado al comienzo de la Sección 3.1 que el algoritmo GMRES, en el iterado i-ésimo, busca una aproximación $x_i \in \mathcal{K}^i(A, b)$. Para ello resuelve el problema de mínimos cuadrados (3.1.3). Hemos visto en la Proposición 3.1.11 que podemos saber si la solución será exacta o no en función del valor $h_{i+1,i}$. Este valor es dado por la iteración de Arnoldi (Algoritmo 1).

Por este motivo en realidad no es necesario calcular explícitamente el valor de la aproximación x_i en cada iterado i-ésimo, basta con realizar la iteración de Arnoldi hasta llegar al iterado en el que el valor de $h_{i+1,i}$ sea igual a 0. En este momento sabemos que no hay error en el problema de mínimos cuadrados y calculamos la solución como $x = x_i = Qy_i$ donde y_i es la solución del sistema $R_i y_i = [\tilde{b}_1, \cdots, \tilde{b}_i]^T$ (Proposición 3.1.11) que se obtiene por sustitución hacia atrás al ser R_i una matriz triangular superior.

Tras el último iterado $i = m$ de la iteración de Arnoldi, hay que calcular y_m para lo que es necesario guardar las rotaciones G_i, para así poder calcular la matriz R_m y el vector $[\tilde{b}_1, \cdots, \tilde{b}_m]^T$.

3.3. Algoritmo GMRES

Ya tenemos todo lo necesario para definir el algoritmo GMRES, véase el pseudocódigo en el Algoritmo 4.

Observamos que:

- De la fila 4 a la 14 se realiza la iteración de Arnoldi con reortogonalización de Gram-Schmidt (Algoritmo 2).
- En la fila 15 comprobamos la condición de parada según la Proposición 3.1.11. La condición $i = n$ es de seguridad para no sobrepasar la dimensión del sistema aunque con aritmética exacta siempre tenemos $h_{n+1,n} = 0$ en el iterado n-ésimo por el Corolario 3.1.14.

Escoge un valor para $\kappa \in (0,1)$, e.g., $\kappa = 0{,}25$;
$q_1 = \frac{b}{\|b\|_2}$; $\tilde{b}_1 = \|b\|_2$;
for $i = 1, ...$ **do**
    `// Arnoldi`
    $t = Aq_i$; $\tau = \|t\|_2$;
    for $k = 1, ..., i$ **do**
        $h_{k,i} = q_k^T t$; $t = t - h_{k,i} q_k$;
    end
    if $\frac{\|t\|_2}{\tau} \leq \kappa$ **then** `// Reortogonalización`
        for $k = 1, ..., i$ **do**
            $\rho = q_k^T t$; $t = t - \rho q_k$; $h_{ki} = h_{ki} + \rho$;
        end
    end
    $h_{i+1,i} = \|t\|_2$;
    if $h_{i+1,i} = 0$ *(con tolerancia)* **or** $i = n$ **then** `// Condición de parada`
        go to 28 ;
    end
    $q_{i+1} = \frac{t}{h_{i+1,i}}$;
    `// Cálculo de` R `y` $\tilde{b}$ `para resolver el problema de LS`
    $r_{1i} = h_{1i}$;
    for $k = 1, ..., i-1$ **do**
        $aux = r_{ki} \cos_k + h_{k+1,i} \sin_k$; $r_{k+1,i} = h_{k+1,i} \cos_k - r_{ki} \sin_k$; $r_{ki} = aux$;
    end
    $\sin_i = \frac{h_{i+1,i}}{\sqrt{r_{ii}^2 + h_{i+1,i}^2}}$; $\cos_i = \frac{r_{ii}}{\sqrt{r_{ii}^2 + h_{i+1,i}^2}}$;
    $r_{ii} = \cos_i r_{ii} + \sin_i h_{i+1,i}$; $\tilde{b}_{i+1} = -\sin_i \tilde{b}_i$; $\tilde{b}_i = c_i \tilde{b}_i$;
end
`// Cálculo de la última columna de` R
$r_{1i} = h_{1i}$;
for $k = 1, ..., i-1$ **do**
    $aux = r_{ki} \cos_k + h_{k+1,i} \sin_k$; $r_{k+1,i} = h_{k+1,i} \cos_k - r_{ki} \sin_k$; $r_{ki} = aux$;
end
`// Resolución del problema de LS`
for $k = i, ..., 1$ **do**
    $y_k = \frac{\tilde{b}_k - \sum_{j=k+1}^{i} r_{kj} y_j}{r_{kk}}$;
end
`// Cálculo del vector solución` $x = Q_j y$
$x = \sum_{j=1}^{i} y_j q_j$;

Algoritmo 4: GMRES sin límite con reortogonalización de Gram-Schmidt

- En las filas 20 a 25 y 28 a 31 se hacen los cálculos de R y $\tilde{b}$ necesarios para resolver posteriormente el problema de mínimos cuadrados (o LS por sus siglas en inglés «Least Squares»). A la hora de hacer una implementación, notar que la matriz $R = (r_{ki})$ puede escribirse sobre la matriz $H = (h_{ki})$ para tener un menor coste de memoria.
  - En las filas 20 a 23 se calcula $G_{i-1}...G_1\widetilde{H}_i$ según las consideraciones de la observación 3.2.3 derivada de la Proposición 3.2.1.
  - En la fila 24 se calculan el seno y el coseno que definen la rotación de Givens G_i.
  - En la fila 25 se aplica G_i calculando el valor r_{ii}, como podemos observar en las igualdades de la expresión (3.2.1), y actualizando las dos últimas posiciones de $\tilde{b}$ según la observación 3.2.4.
  - En las filas 28 a 31 se realiza la última iteración.
- En las filas 33 a 35 se resuelve el sistema $Ry = \tilde{b}$ asociado al problema de mínimos cuadrados.
- En la fila 37 se calcula la solución $x = Q_i y$.

3.4. Coste computacional y de memoria

En esta sección analizamos el número de operaciones elementales (flops): sumas, restas, multiplicaciones y divisiones que requiere el algoritmo GMRES. También estimamos la cantidad de memoria requirida. A continuación nos referiremos a los pasos del Algoritmo 4. Siguiendo la notación habitual, denotaremos la dimensión del sistema por n. Como no podemos controlar cuando se ejecutará la reortogonalización, supondremos que se realiza el algoritmo sin utilizar el refinamiento, es decir, ignorando las filas 9 a 13.

Primero calcularemos el número de operaciones de la iteración i-ésima (filas de la 4 a la 25):

- En la fila 5, se realiza el producto de la matriz A por un vector, al número de operaciones de este producto lo denotaremos por P.
- En la fila 7, se ortogonaliza la base (Arnoldi). Tenemos el producto escalar de dos vectores, el producto de un vector por un escalar y la suma de dos vectores, en un bucle de i iteraciones. En total,

$$\sum_{k=1}^{i} 2n - 1 + n + n = i(4n - 1).$$

- En la fila 22, aplicamos las rotaciones de Givens. Tenemos seis operaciones que se repiten en un bucle de $i - 1$ iterados. En total $6(i - 1)$ flops.
- El resto de operaciones de la iteración i-ésima son $\mathcal{O}(n)$.

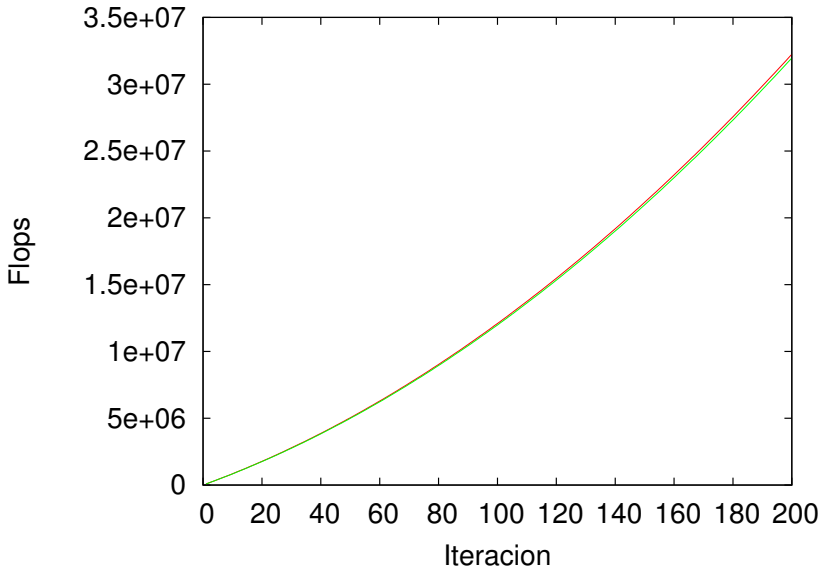

Figura 3.1: Se ha estimado un total de $iP + 2i^2n$ operaciones en i iteraciones de GMRES. Se muestra en verde dicha estimación. En rojo, el valor obtenido a partir de contar las operaciones en el código implementado.

Supongamos que GMRES termina en $m \leq n$ iterados. Entonces el coste de las m iteraciones es de

$$\sum_{i=1}^{m} P + i(4n-1) + 6(i-1) + \mathcal{O}(n) = mP + (4n-1)\frac{m^2+m}{2} + \mathcal{O}(m^2) + \mathcal{O}(mn)$$
$$= mP + 2m^2n + \mathcal{O}(mn).$$

Los pasos fuera del bucle analizado, filas 27 a 37, tienen un coste de $\mathcal{O}(m^2)$ flops. Así, el coste final es de $mP + 2m^2n + \mathcal{O}(mn)$ flops.

Observación 3.4.1. Recordemos que el coste de $mP + 2m^2n + \mathcal{O}(mn)$ flops es sin reortogonalización. Si llevamos a cabo la reortogonalización, en el peor de los casos, es decir, si reortogonalizamos siempre, el coste sería de $mP + 4m^2n + \mathcal{O}(mn)$ flops.

Para comprobar el resultado se ha añadido un contador de operaciones en la implementación de GMRES realizada y se ha resuelto un sistema con matriz y término independiente aleatorios de dimensión 200 (con valores generados a partir de una ley uniforme en $(-1, 1)$). Para cada iteración i, el gráfico de la Figura 3.1 muestra en color rojo el número de operaciones reales realizadas hasta ese punto y en color verde el valor $iP + 2i^2n$. Notemos que al ser una matriz aleatoria el valor de P es $n(2n-1)$.

Ahora calculemos el coste en memoria. En el iterado i-ésimo trabajamos con las matrices

$$Q_{i+1} \in \mathbb{R}^{n\times(i+1)}, \quad \widetilde{H}_i \in \mathbb{R}^{(i+1)\times i} \text{ y } R_i \in \mathbb{R}^{i\times i}.$$

Aunque en la práctica la matriz R_i puede escribirse sobre la matriz $\widetilde{H}_i$, con lo que debemos almacenar $n(i+1) + i(i+1) = ni + i^2 + n + i$ valores. También deberemos alamacenar los vectores sin $\in \mathbb{R}^i$, cos $\in \mathbb{R}^i$ que definen las rotaciones de Givens, y el vector $\tilde{b} \in \mathbb{R}^i$ para resolver el problema de mínimos cuadrados. Lo que hace un total de

$$ni + i^2 + n + 4i = ni + i^2 + \mathcal{O}(n)$$

valores. En el peor de los casos, si GMRES no converge hasta el iterado n-ésimo, necesitaremos almacenar $2n^2 + \mathcal{O}(n)$ valores.

En el Cuadro 3.1 aparece la memoria necesaria, en el peor de los casos, para dimesiones entre 10^3 y 10^{10} suponiendo que trabajemos con doble precisión. Notemos que para $n > 10^5$ se requieren más de ≈ 150 Gbytes de memoria.

$\log_{10}(n)$	Gbytes
3	$\approx 1{,}5 \times 10^{-2}$
4	$\approx 1{,}5$
5	$\approx 1{,}5 \times 10^{2}$
6	$\approx 1{,}5 \times 10^{4}$
7	$\approx 1{,}5 \times 10^{6}$
8	$\approx 1{,}5 \times 10^{8}$
9	$\approx 1{,}5 \times 10^{10}$
10	$\approx 1{,}5 \times 10^{12}$

Cuadro 3.1: Coste estimado de memoria necesaria del algoritmo GMRES (en Gbytes) en función de la dimensión del sistema (mostramos $\log_{10}(n)$ en la primera columna).

3.5. Restarted GMRES

En el Cuadro 3.1 podemos ver que a medida que crece la dimensión, la memoria necesaria para ejecutar GMRES puede hacer inviable su uso. Además, hemos visto que el coste computacional de la iteración i crece a medida que incrementamos i.

En la misma línea, recordamos que en la sección 2.3.3 y en las reflexiones anteriores al algoritmo 1, se justificó la ortogonalización de Gram-Schmidt ya que «típicamente» $i \ll n$. Esta observación da lugar a una variación del algoritmo 4 que pretende mitigar los problemas numéricos descritos y solucionar la posible falta de memoria. Esta variación recibe el nombre de Restarted GMRES o GMRES(m).

Este «restarted» o reiniciado no es más que truncar en un número de iterados m preestablecido. Si después de m iterados no se obtiene la solución, repetimos el algoritmo pero esta vez considerando el error obtenido como término independiente del sistema. De esta forma el método Restarted GMRES o GMRES(m) consiste en repetir iterativamente el algoritmo GMRES, descrito en 4, pero truncado en m iterados. Cada sucesión de m iterados consecutivos de GMRES da lugar a un ciclo de GMRES(m).

Observación 3.5.1. Si $b - Ax_1 = r_1$ y $r_1 - Ax_2 = 0$ entonces la solución del sistema $Ax = b$ es $x = x_1 + x_2$.

De la Observación 3.5.1 se deriva el método GMRES(m). Los detalles los podemos consultar en el Algoritmo 5. Recordemos, Corolario 3.1.14, que GMRES (Algoritmo 4) calcula la solución en n iterados como máximo, siendo n la dimensión del sistema. Si n es muy grande hemos visto que puede ser inviable realizar n iteraciones de GMRES por cuestiones de memoria. GMRES(m), Algoritmo 5, resuelve este problema de espacio al truncar en $m \ll n$ pasos. Pero hay una observación importante, no podemos garantizar resolver el sistema en un número finito de ciclos de GMRES(m). ¿Convergirá GMRES(m)? ¿Para qué valores de m? ¿Qué valor de m es el más adecuado?.

Escoge un valor para $\kappa \in (0,1)$, e.g., $\kappa = 0{,}25$;
Escoge un número de iterados m menor que la dimensión del sistema;
$m' = m$; $x = 0$; $err = b$; $\tilde{b}_1 = \|b\|_2$;
while $\tilde{b}_1 > 0$ *(con cierta tolerancia)* **do** `// bucle de reinicio`
  $q_1 = \frac{err}{\|err\|_2}$;
  for $i = 1, ..., m$ **do** `// GMRES m iteraciones`
    `// Arnoldi`
    $t = Aq_i$; $\tau = \|t\|_2$;
    for $k = 1, ..., i$ **do**
      $h_{k,i} = q_k^T t$; $t = t - h_{k,i} q_k$;
    end
    if $\frac{\|t\|_2}{\tau} \leq \kappa$ **then** `// Reortogonalización`
      for $k = 1, ..., i$ **do**
        $\rho = q_k^T t$; $t = t - \rho q_k$; $h_{ki} = h_{ki} + \rho$;
      end
    end
    $h_{i+1,i} = \|t\|_2$;
    if $h_{i+1,i} = 0$ *(con tolerancia)* **then**
      `// Cálculo de la última columna de` R
      $r_{1i} = h_{1i}$;
      for $k = 1, ..., i-1$ **do**
        $aux = r_{ki}\cos_k + h_{k+1,i}\sin_k$; $r_{k+1,i} = h_{k+1,i}\cos_k - r_{ki}\sin_k$; $r_{ki} = aux$;
      end
      $m' = i$; **go to** 36 ;
    end
    $q_{i+1} = t/h_{i+1,i}$;
    `// Cálculo de` R `y` $\tilde{b}$ `para resolver el problema de LS`
    $r_{1i} = h_{1i}$;
    for $k = 1, ..., i-1$ **do**
      $aux = r_{ki}\cos_k + h_{k+1,i}\sin_k$; $r_{k+1,i} = h_{k+1,i}\cos_k - r_{ki}\sin_k$; $r_{ki} = aux$;
    end
    $\sin_i = \frac{h_{i+1,i}}{\sqrt{r_{ii}^2 + h_{i+1,i}^2}}$; $\cos_i = \frac{r_{ii}}{\sqrt{r_{ii}^2 + h_{i+1,i}^2}}$;
    $r_{ii} = \cos_i r_{ii} + \sin_i h_{i+1,i}$; $\tilde{b}_{i+1} = -\sin_i \tilde{b}_i$; $\tilde{b}_i = c_i \tilde{b}_i$;
  end
  `// Resolución del problema de LS`
  for $k = m', ..., 1$ **do**
    $y_k = \frac{\tilde{b}_k - \sum_{j=k+1}^{m'} r_{kj} y_j}{r_{kk}}$;
  end
  `// Cálculo del vector aproximación` $x = x + Q_{m'} y$
  $x = x + \sum_{j=1}^{m'} y_j q_j$;
  $err = b - Ax$; $\tilde{b}_1 = \|err\|_2$;
end

Algoritmo 5: GMRES(m) con reortogonalización de Gram-Schmidt

Capítulo 4

Convergencia de GMRES(m)

En esta sección veremos que las respuestas a las preguntas con las que finaliza la sección anterior no tienen fácil respuesta. Primeramente, estudiaremos el comportamiento de los residuos bajo la iteración de GMRES. Denotamos por $r_i = b - Ax_i$ el residuo obtenido a partir de la aproximación x_i del iterado i-ésimo de GMRES. En particular, veremos que en algunos casos no se puede garantizar un decaimiento del residuo después de un ciclo de GMRES(m), dando lugar a un estancamiento del método. El estudio del comportamiento de la sucesión de residuos $\{r_i\}_{i\geq 1}$ se basa en los resultados en [5] que reproducimos, convenientemente adapatados, en la Sección 4.1. En la Sección 4.2 se caracteriza los residuos que dan lugar al estancamiento del Restarted GMRES en función de m.

4.1. Acotación del error según la distribución de los valores propios en el plano complejo

Seguidamente discutimos el comportamiento de los residuos. Veremos que los resultados presentados establecen relaciones entre el parámetro m adecuado y la velocidad de convergencia. Sea $\mathbb{P}_i$ el conjunto de polinomios de grado menor o igual que i.

Lema 4.1.1. *([5] Lema 6.31) Sea x_i la aproximación correspondiente al paso i-ésimo del algoritmo GMRES. Sea $r_i = b - Ax_i$. Entonces,*

$$x_i = P_{i-1}(A)b,$$

donde $P_{i-1} \in \mathbb{P}_{i-1}$. Además,

$$\|r_i\|_2 = \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \|P(A)b\|_2.$$

Demostración. Recordemos de la Observación 3.1.1 que $x_i = Q_i y_i$ con $y_i \in \mathbb{R}^i$ y $Q_i = \begin{bmatrix} q_1 & \cdots & q_i \end{bmatrix}$. En la demostración de la Proposición 2.3.16 vimos que $q_j = P^*_{j-1}(A)b$ para $j \leq i$ con $P^*_{j-1} \in \mathbb{P}_{j-1}$, y por tanto $x_i = y_{i1}q_1 + ... + y_{ii}q_i = P_{i-1}(A)b$ con $P_{i-1} \in \mathbb{P}_{i-1}$.

Si $x_i = P_{i-1}(A)b$ entonces los coeficientes de P_{i-1} son los coeficientes de x_i expresado en la base $\langle b, Ab, ..., A^{i-1}b\rangle = \mathcal{K}^i(A, b)$. Como x_i minimiza el residuo $\|b - Ax\|_2$ dentro del espacio de Krylov i-dimensional, entonces P_{i-1} es el polinomio que minimiza el residuo

$\|b - AP(A)b\|_2$ en el conjunto de polinomios de grado menor o igual que $i-1$. De esta forma tenemos,

$$\begin{aligned}\|r_i\|_2 &= \min_{x\in\mathbb{K}^i(A,b)} \|b - Ax\|_2 = \min_{P\in\mathbb{P}_{i-1}} \|b - AP(A)b\|_2 \\ &= \min_{P\in\mathbb{P}_{i-1}} \|(I - AP(A))b\|_2 = \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \|P(A)b\|_2.\end{aligned}$$

□

Proposición 4.1.2. *([5] Proposición 6.32) Supongamos que A es diagonalizable y $A = XDX^{-1}$ con $D = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ la matriz diagonal de valores propios. Sea,*

$$\epsilon_i = \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \max_{j=1,...,n} |P(\lambda_j)|.$$

Entonces obtenemos la siguiente cota del error en la i-ésima iteración del algoritmo GMRES para el sistema $Ax = b$,

$$\|r_i\|_2 \leq \kappa_2(X)\|b\|_2\,\epsilon_i, \tag{4.1.1}$$

donde $\kappa_2(X) = \|X\|_2\|X^{-1}\|_2$ es el número de condición de X.

Demostración. Utilizando el segundo resultado del Lema 4.1.1,

$$\begin{aligned}\|r_i\|_2 &= \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \|P(A)b\|_2 = \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \|P(XDX^{-1})b\|_2 = \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \|XP(D)X^{-1}b\|_2 \\ &\leq \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \|X\|_2\|P(D)\|_2\|X^{-1}\|_2\|b\|_2 = \kappa_2(X)\|b\|_2 \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \|P(D)\|_2.\end{aligned}$$

Como D es la matriz diagonal de valores propios $\lambda_1, \lambda_2, ..., \lambda_n$, entonces,

$$\|P(D)\|_2 = \max_{j=1,...,n} |P(\lambda_j)|,$$

y por tanto,

$$\|r_i\|_2 \leq \kappa_2(X)\|b\|_2 \min_{\substack{P\in\mathbb{P}_i \\ P(0)=1}} \max_{j=1,...,n} |P(\lambda_j)|.$$

□

Observación 4.1.3. El valor $\kappa_2(X)$ y la norma de b vienen dados por el sistema. De esta forma, únicamente se mejora la cota (4.1.1) al aumentar la dimensión del espacio de polinomios sobre el cual se minimiza $\max_{j=1,...,n} |P(\lambda_j)|$.

Observación 4.1.4. Con la cota (4.1.1) volvemos a ver que GMRES converge en como máximo n iterados, ya que existe un polinomio de grado n que interpola los valores propios $\lambda_1, ..., \lambda_n$ y, para este polinomio se tiene

$$\max_{j=1,...,n} |P(\lambda_j)| = 0.$$

La convergencia en n iterados ya la vimos en los Corolarios 2.2.8 y 3.1.14.

Observación 4.1.5. Se puede obtener una cota explícita de los residuos a partir de la Proposición 4.1.2, véase la sección 6.11.2 de [5]. Para ello se usan los polinomios de Chebychev sobre el plano complejo, que se definen recursivamente

$$C_0(z) = 1, C_1(z) = z,$$
$$C_{k+1}(z) = 2zC_k(z) - C_{k-1}(z).$$

Estos polinomios tienen la propiedad siguiente: si $E(c, d, a)$ es la elipse de centro $c \in \mathbb{C}$, distancia focal $d \in \mathbb{C}$ y semieje mayor $a \in \mathbb{C}$, entonces

$$\min_{\substack{P \in \mathbb{P}_i \\ P(0)=1}} \max_{\lambda \in E(c,d,a)} |P(\lambda)| \leq \frac{C_i(\frac{a}{d})}{|C_i(\frac{c}{d})|}.$$

Se obtiene así la cota del residuo

$$\|r_i\|_2 \leq \kappa_2(X)\|b\|_2\epsilon_i \leq \kappa_2(X)\|b\|_2 \frac{C_i(\frac{a}{d})}{|C_i(\frac{c}{d})|}.$$

4.1.1. Ejemplos numéricos: relación entre m y el espectro de A

Gracias a la Proposición 4.1.2 podemos intentar determinar un buen parámetro m para GMRES(m) dependiendo de como esten distribuidos los valores propios de A sobre el plano complejo. Supongamos que podemos agrupar los valores propios en $m < n$ grupos de tal forma que en cada grupo los valores propios sean cercanos. Entonces existe un polinomio $P \in \mathcal{P}_m$ cuyas raices aproximan los valores propios y, por tanto,

$$\epsilon_m \leq \max_{j=1,...,n} |P(\lambda_j)|,$$

es pequeño. El problema práctico de esta estrategia es que probablemente no sepamos nada de la distribución de los valores propios de la matriz A, y averiguarlo sea posiblemente más costoso que resolver el sistema.

En la Figura 4.1 se muestra el ejemplo de un polinomio $P \in \mathcal{P}_3$ que minimiza $\max_{j=1,...,9} |P(\lambda_j)|$ en $\mathcal{P}_3$ para nueve valores propios $\lambda_1, ..., \lambda_9$ agrupados en tres grupos. Para una matriz $A \in \mathbb{R}^{9\times 9}$ con valores propios $\lambda_1, ..., \lambda_9$, cabría esperar que GMRES(m) para $m = 3$ tendría un comportamiento óptimo en virtud de la cota de la Proposición 4.1.2.

Ejemplo 4.1.6. Sea $X = (x_{ij}) \in \mathbb{R}^{9\times 9}$ tal que $x_{kk} = 1$ para todo $1 \leq k \leq 9$, $x_{k,k+1} = 0{,}9$ para todo $1 \leq k \leq 8$, y el resto de coeficientes son nulos. Definimos las familias de matrices

$$D_\delta = (d_{ij}^\delta) = diag(2 - \delta, 2, 2 + \delta, 5 - \delta, 5, 5 + \delta, 8 - \delta, 8, 8 + \delta) \qquad \text{y} \qquad A_\delta = XD_\delta X^{-1},$$

donde $\delta \in [0, 1]$. Sean $b^\delta \in \mathbb{R}^9$ tal que $b_i^\delta = d_{ii}^\delta + d_{i+1,i+1}^\delta$ para $1 \leq i \leq 8$ y $b_9^\delta = d_{99}^\delta + d_{00}^\delta$.

Notemos que A_δ es diagonalizable para todo $\delta \in [0, 1]$ y A_0 tiene exactamente 3 valores propios. Por la Proposición 4.1.2 GMRES encuentra la solución al sistema $A_0x = b$ en el tercer iterado, para todo $b \in \mathbb{R}^9$. Para A_δ, $\delta > 0$, hay 9 valores propios agrupados en 3 grupos. Los valores propios de cada grupo son más cercanos cuanto más pequeño sea δ. Así, para δ pequeño, la situación es similar a la de la Figura 4.1. Para A_1 los valores propios son $1, 2, 3, 4, 5, 6, 7, 8$ y 9.

El objetivo es analizar el número de operaciones elementales o flops empleados por GMRES(m), para $1 \leq m \leq 9$, al resolver los sistemas

$$A_\delta x = b^\delta, \qquad \delta = 0, 10^{-9}, 10^{-8}, ..., 10^{-1}, 1.$$

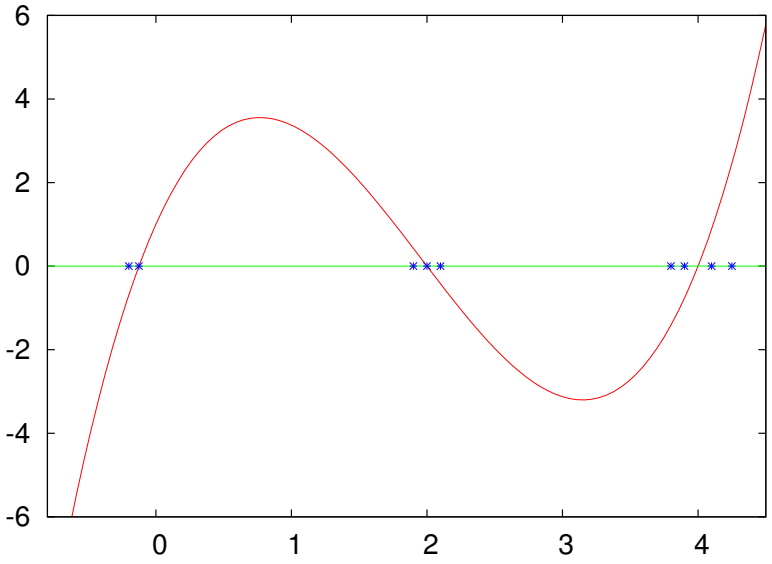

Figura 4.1: Ejemplo de polinomio de tercer grado que minimiza $\max_{j=1,\ldots,9} |P(\lambda_j)|$ en $\mathcal{P}_3$ para nueve valores propios $\lambda_1, \ldots, \lambda_9$ agrupados en tres grupos

Notemos que GMRES(9) es equivalente a GMRES sin límite. Los resultados se muestran en la Figura 4.2, cada línea muestra los resultados para un δ determinado. Podemos observar que el comportamiento es parecido hasta GMRES(2) para todo δ. GMRES(m), para $3 \leq m \leq 9$, emplea muy pocos flops para resolver el sistema para $\delta = 0$, de hecho utiliza los mismos ya que en la tercera iteración termina. Esto es debido a la existencia de un polinomio de tercer grado que interpola los tres valores propios de A_0. Vemos que GMRES(3) emplea más operaciones para el resto de valores $\delta > 0$. Cuanto más grande es el valor de δ, más distancia hay entre los valores propios en cada uno de los tres grupos, y peor se comporta GMRES(3). La línea verde muestra el caso con $\delta = 1$ para el que se aprecia un peor comportamiento.

Ejemplo 4.1.7. Sea $A = (a_{ij}) \in \mathbb{R}^{8\times 8}$ matriz triangular superior tal que $a_{kk} = 8$ para $1 \leq k \leq 4$, $a_{kk} = -8$ para $4 \leq k \leq 8$ y $a_{ij} = j - i$ para $j > i$. Sea $b \in \mathbb{R}^8$ tal que $b_k = k$. Queremos ver el comportamiento de GMRES(m), para $2 \leq m \leq 8$, al resolver el sistema $Ax = b$. Aplicamos el mismo experimento númerico utilizado en el Ejemplo 4.1.6. Notemos que A tiene dos valores propios, 8 y -8. Los resultados pueden consultarse en la Figura 4.3.

¿Porque no observamos el mismo comportamiento que con A_0 en el Ejemplo 4.1.6?, es decir, ¿porqué no observamos un buen comportamiento en GMRES(2)?. La respuesta es que no podemos aplicar la Proposición 4.1.2 porque la matriz A no diagonaliza, hipótesis necesaria para aplicar la cota.

En el siguiente Ejemplo 4.1.10 vamos a intentar generalizar la idea propuesta al principio de esta sección y que hemos aplicado al ejemplo 4.1.6. En lugar de utilizar la cota propuesta en la Proposición 4.1.2 que requiere que la matriz del sistema sea diagonalizable, utilizaremos la dimensión del polinomio mínimo de la matriz con $k < n$ valores propios que perturbamos dando lugar a matrices con n valores propios agrupados en k grupos. En el caso no diagonalizable, GMRES(m) tendrá un buen comportamiento para m igual a la dimensión del polinomio mínimo de A y no para $m = k$. En lo que sigue

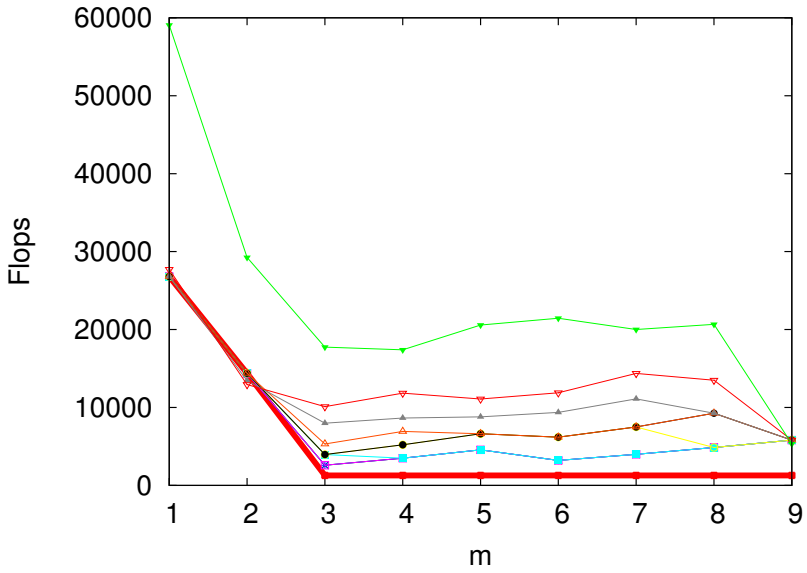

Figura 4.2: Se muestra el número de operaciones elementales (flops) respecto al parámetro de reinicio m para el Ejemplo 4.1.6. La línea gruesa corresponde a $\delta = 0$. El resto corresponden a $\log_{10}(\delta) = -9, \ldots, 0$ en orden ascendente.

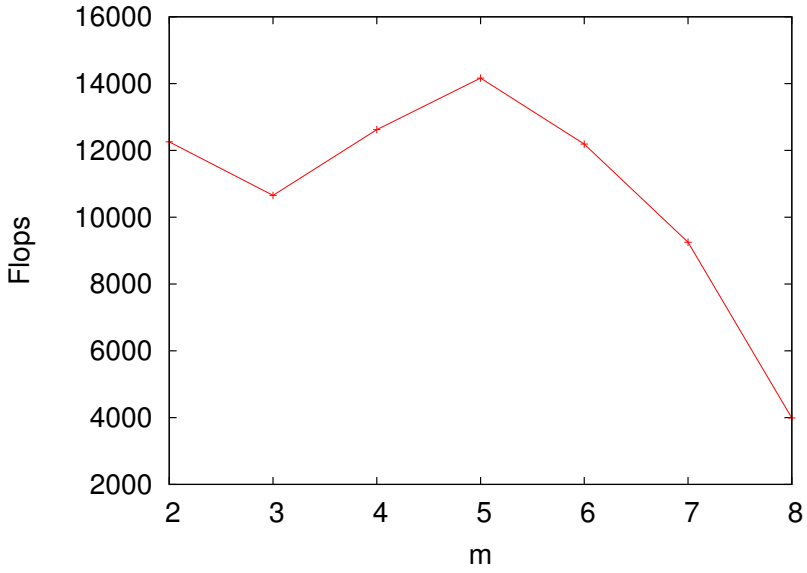

Figura 4.3: Se muestra el número de operaciones elementales (flops) respecto al parámetro de reinicio m para el Ejemplo 4.1.7.

jugará un papel importante el índice de un valor propio, que es igual a la dimensión de la mayor caja de Jordan asociada a éste.

Observación 4.1.8. Si la matriz A es diagonalizable y tiene m valores propios, entonces el polinomio mínimo de A tiene dimensión m. Esto no es así si la matriz no diagonaliza. Entonces la dimensión del polinomio mínimo depende del ínidice de los valores propios de A, es decir, de la dimensión del mayor bloque de Jordan para cada valor propio. Si A tiene valores propios $\lambda_1, ..., \lambda_k$ de índices $m_1, ..., m_k$ respectivamente, entonces la dimensión del polinomio mínimo de A es $m = m_1, ..., m_k$ [4].

Observación 4.1.9. Recordemos del Corolario 2.2.8, que si $A \in \mathbb{R}^{n\times n}$ tiene polinomio mínimo de dimensión m, entonces GMRES encuentra la solución al sistema $Ax = b$ en el m-ésimo iterado para todo $b \in \mathbb{R}^n$ (considerando aritmética exacta).

Ejemplo 4.1.10. Consideramos los vectores b^δ, la matriz $X \in \mathbb{R}^{9\times 9}$ y la familia de matrices $D_\delta \in \mathbb{R}^{9\times 9}$ definidas en el Ejemplo 4.1.6. Sea la matriz $U = (u_{ij}) \in \mathbb{R}^{9\times 9}$ tal que $u_{12} = 1$ y el resto de elementos son nulos. Definimos las familias de matrices

$$J_\delta = D_\delta + U \qquad \text{y} \qquad B_\delta = X J_\delta X^{-1},$$

donde $\delta \in [0, 1]$.

Repetimos el experimento numérico realizado en el Ejemplo 4.1.6 sin considerar $\delta = 1$, es decir, para $\delta = 0, 10^{-9}, 10^{-8}, ..., 10^{-1}$, y considerando los sistemas $B_\delta x = b^\delta$. Los resultados se encuentran en la Figura 4.4. Observamos el mismo comportamiento que en el Ejemplo 4.1.6, pero las diferencias entre los parámetros m se producen a partir de $m = 4$ en lugar de $m = 3$. Como adelantábamos antes de este ejemplo, el motivo es la dimensión del polinomio mínimo de B_0. En este caso tenemos uno de los tres valores propios con índice 2 y, por lo tanto, la dimensión del polinomio mínimo de B_0 es 4 en lugar de 3.

Repetiremos el experimento, pero esta vez haremos que B_0 tenga un polinomio mínimo de dimensión 5. Para ello modificamos la matriz U de manera que $u_{12} = 1$, $u_{23} = 1$ y el resto de valores son nulos. Los resultados se encuentran en la Figura 4.5.

Las matrices de los sistemas en los Ejemplos 4.1.6 y 4.1.10 tienen valores propios reales. Notemos que ni la Proposición 4.1.2 ni el Corolario 2.2.8 requieren que la matriz tenga valores propios reales. Así, los razonamientos en cuanto al parámetro m de GMRES(m) realizados a partir de estos dos resultados son válidos para matrices con valores propios complejos. Los siguientes ejemplos muestran esta observación.

Ejemplo 4.1.11. (Colisión de Krein) Consideramos una familia de matrices $A_\delta = A_0 + \delta B \in \mathbb{R}^{4\times 4}$, donde

$$A_0 = (a_{ij}) = \begin{bmatrix} 3/4 & 1/18 & 1/2 & 1/4 \\ -1/8 & 11/12 & 1/4 & -3/8 \\ -1/2 & 0 & 1 & 0 \\ 0 & 2/9 & 0 & 1 \end{bmatrix}, \qquad y$$

$$B = w \cdot e_2^T,\ e_2 = [0, 1, 0, 0]^T,\ w = [-1/8, 3/16, 0, -1/2]^T.$$

Se tiene que A_0 tiene dos valores propios $e^{\pm i\theta}$, $\theta = \arctan(\sqrt{23}/11)$, ambos con multiplicidad 2 e índice 2. De hecho, A_δ tiene cuatro valores propios $e^{\pm i\theta_1}, e^{\pm i\theta_2}$, $\theta_1 \neq \theta_2$, para $0 < \delta \ll 1$. Éstos coinciden (colision de Krein) para $\delta = 0$. Para $\delta < 0$, los cuatro valores propios salen del círculo unidad.

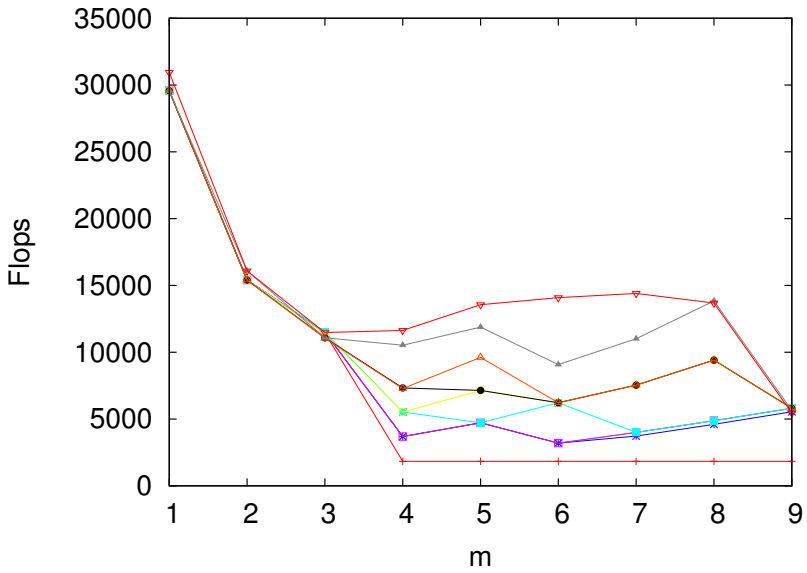

Figura 4.4: Se muestra el número de operaciones elementales (flops) respecto al parámetro de reinicio m para el primer caso del Ejemplo 4.1.10.

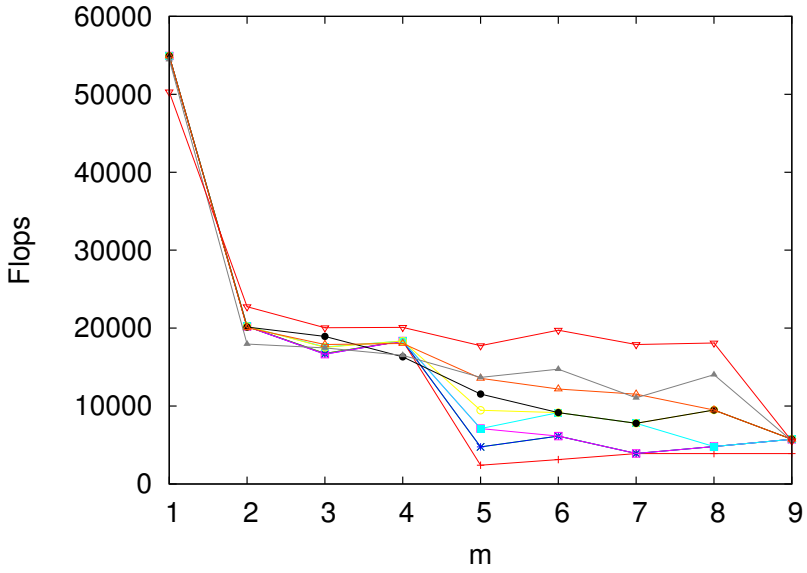

Figura 4.5: Se muestra el número de operaciones elementales (flops) respecto al parámetro de reinicio m para el segundo caso del Ejemplo 4.1.10.

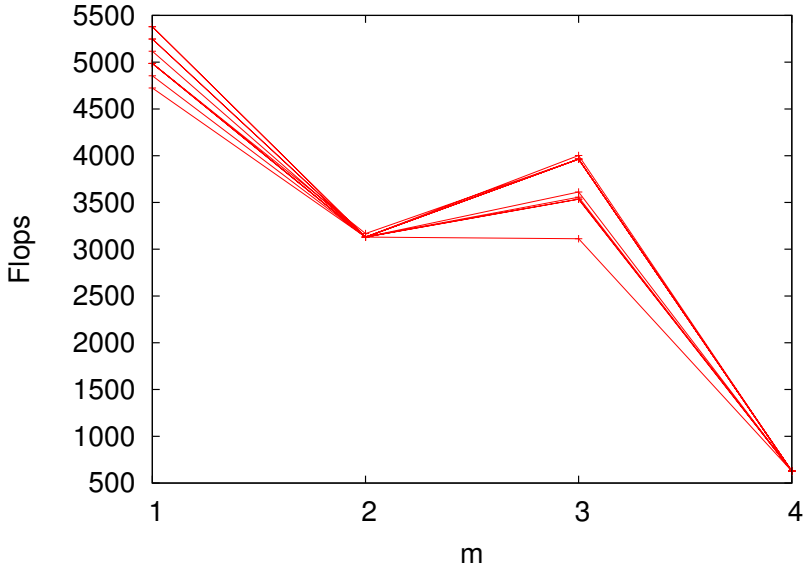

Figura 4.6: Se muestra el número de operaciones elementales (flops) respecto al parámetro de reinicio m para el Ejemplo 4.1.11.

Aplicamos el mismo experimento numérico que en el Ejemplo 4.1.6 para los sistemas $A_\delta x = b$, $\delta = -0{,}1 + 0{,}02\mu$, $\mu = 1, 2, ..., 10$ y

$$b = [a_{11} + a_{22}, a_{22} + a_{33}, a_{33} + a_{44}, a_{44} + a_{11}]^T.$$

Los resultados se pueden consultar en la Figura 4.6, cada línea corresponde con un valor de δ. Observamos que no hay un buen comportamiento de GMRES(m) para ningún $m < 4$. Además, observamos el mismo comportamiento para todo δ, aunque para $\delta = 0$ pasamos a tener 2 valores propios en vez de 4. Esto se debe a que el polinomio mínimo de A_0 es de grado 4, como lo es para el resto de las matrices de la familia A_δ.

Los dos últimos ejemplos los construimos a partir del ejemplo anterior.

Ejemplo 4.1.12. Consideramos la familia de matrices A_δ del Ejemplo 4.1.11 y definimos la familia de matrices

$$B_\delta = \begin{bmatrix} A_\delta & & & \\ & 0 & 2(1+\delta) & 0 & 0 \\ & -2(1+\delta) & 0 & 0 & 0 \\ & 0 & 0 & 0 & 2(1-\delta) \\ & 0 & 0 & -2(1-\delta) & 0 \end{bmatrix} \in \mathbb{R}^{8\times 8}.$$

Notemos que hemos añadido dos cajas antisimétricas que dependen del parámetro δ a las matrices A_δ. Para todo δ, los valores propios de A_δ también son valores propios de B_δ. Las cajas antisimétricas contribuyen con dos valores propios para $\delta = 0$, de multiplicidad 2 e índice 1; o con cuatro valores propios emparejados si $\delta \neq 0$. Es decir, de forma similar al Ejemplo 4.1.10, esperamos el mismo comportamiento de GMRES(m), $m < 6$, para todos los δ. GMRES(m), $m \geq 6$, acaba en el primer ciclo para $\delta = 0$ y para $\delta \neq 0$ esperamos un buen comportamiento de GMRES(6). Comprobamos que se cumplen los resultados esperados aplicando el experimento numérico descrito en el Ejemplo 4.1.6 para

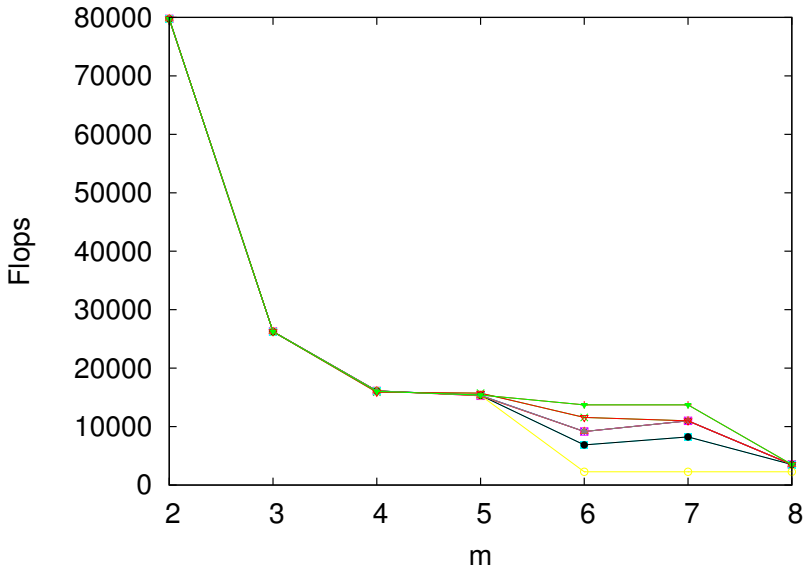

Figura 4.7: Se muestra el número de operaciones elementales (flops) respecto al parámetro de reinicio m para el Ejemplo 4.1.12.

los sistemas $B_\delta x = b$, $\delta = 0, \pm 10^{-4-\frac{\mu}{2}}$, $\mu = 1, 2, 3, 4$, y $b \in \mathbb{R}^8$ tal que $b_k = 0.1k$, $1 \leq k \leq 8$. Estos resultados pueden consultarse en la Figura 4.7. Cada línea corresponde a un valor de δ, la línea con un valor menor en $m = 6$ corresponde a $\delta = 0$, mientras que las otras líneas corresponden a valores de $|\delta|$ más elevados.

Ejemplo 4.1.13. Haciendo una pequeña variación en el Ejemplo 4.1.12 queremos construir una famíla de matrices que compartan la propiedad de tener un polinomio mínimo de grado menor a la dimensión del sistema. Consideramos la familia de matrices A_δ del Ejemplo 4.1.11 y definimos la familia de matrices

$$C_\delta = \begin{bmatrix} A_\delta & & & & \\ & 0 & 2+\delta & 0 & 0 \\ & -2+\delta & 0 & 0 & 0 \\ & 0 & 0 & 0 & 2-\delta \\ & 0 & 0 & -2-\delta & 0 \end{bmatrix} \in \mathbb{R}^{8\times 8}.$$

Se tiene que las cajas antisimétricas aportan dos valores propios de multiplicidad geométrica 2 e índice 1 para $-0{,}1 \leq \delta \leq 0{,}1$ y, por tanto, el polinomio mínimo es de grado 6 en todos los casos. Aplicamos el mismo experimento numérico del Ejemplo 4.1.6 para los sistemas $C_\delta x = b$, $\delta = -0{,}9 + 0{,}1\mu$, $\mu = 1, 2, ..., 18$, y $b \in \mathbb{R}^8$ tal que $b_k = 0.1k$, $1 \leq k \leq 8$. Los resultados los podemos consultar en la Figura 4.8, cada línea corresponde a un valor de δ. Confirmamos las conclusiones obtenidas a partir del Corolario 2.2.8 con matrices que tienen valores propios complejos, es decir, observamos que GMRES(m) termina en la sexta iteración del primer ciclo para $m \geq 6$.

4.2. Estancamiento

Vamos a intentar dar respuesta a una de las preguntas planteadas al final de la Sección 3.5. ¿Cuándo convergirá GMRES(m)?.

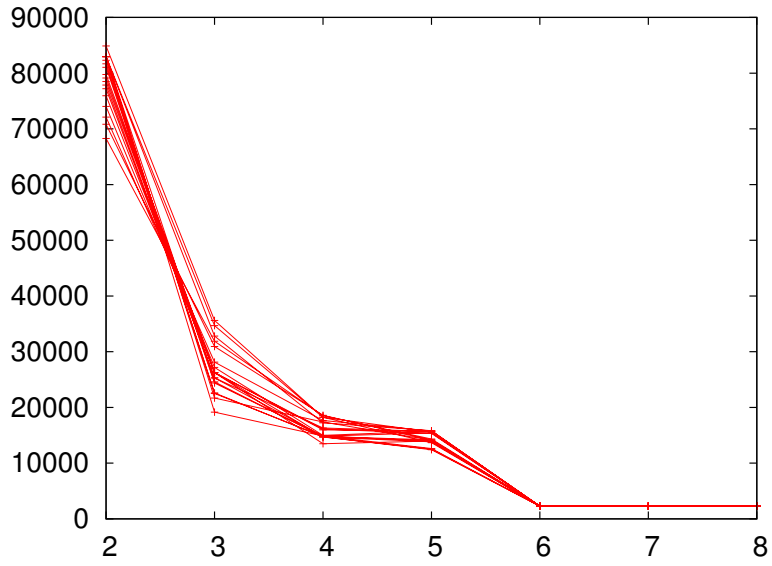

Figura 4.8: Se muestra el número de operaciones elementales (flops) respecto al parámetro de reinicio m para el Ejemplo 4.1.13.

Lema 4.2.1. *([6] Proposition 1) Sean $G_i, ..., G_1$ las rotaciones de Givens que hacen $G_i ... G_1 \widetilde{H}_i$ una matriz triangular superior (véase Sección 3.1.2). Sea $\sin_i$ el seno del ángulo de la rotación de Givens G_i. Entonces,*

$$\|r_i\|_2 = |\sin_i| \, \|r_{i-1}\|_2, \qquad i \geq 2. \tag{4.2.1}$$

Demostración. Por la Proposición 3.1.9, si $[\tilde{b}_1, \cdots, \tilde{b}_{i-1}]^T$ y $[\tilde{b}_1, \cdots, \tilde{b}_{i-1}, \tilde{b}_i]^T$ son los vectores correspondientes al problema de mínimos cuadrados con los que hallamos las aproximaciones x_{i-1} y x_i respectivamente, entonces

$$\|r_{i-1}\|_2 = |\tilde{b}_{i-1}| \qquad \text{y} \qquad \|r_i\|_2 = |\tilde{b}_i|.$$

Por la Observación 3.2.4,

$$\tilde{b}_i = -\tilde{b}_{i-1} \sin_i . \qquad \square$$

Observación 4.2.2. $\|r_1\|_2 = |e_2^T (G_1 \|b\|_2 \, e_1)| = |\sin_1| \, \|b\|_2$.

Observación 4.2.3. $\|r_i\|_2 = |\sin_1 ... \sin_i| \, \|b\|_2$.

En el Lema 4.2.1 podemos ver que la velocidad de convergencia de GMRES depende de los senos $\sin_1, ..., \sin_i$ de los ángulos de las rotaciones de Givens $G_1, ..., G_i$ que triangulan la matriz $\widetilde{H}_i = \begin{bmatrix} H_i \\ h_{i+1,i} e_i^T \end{bmatrix}$. Notemos que si $|\sin_i| = 1$ entonces $\|r_i\|_2 = \|r_{i-1}\|_2$. Veremos que el estancamiento de GMRES(m) se producirá cuando $|\sin_1| = |\sin_2| = ... = |\sin_m| = 1$.

Proposición 4.2.4. *Las siguientes tres condiciones son equivalentes:*

1. $r_i = r_{i-1}$.
2. $|\sin_i| = 1$.
3. H_i *es una matriz singular.*

Demostración. Primero demostraremos $|\sin_i| = 1$ si, y solo si, $r_i = r_{i-1}$. Por el Lema 4.2.1, $|\sin_i| = 1$ si, y solo si, $\|r_i\|_2 = \|b - Ax_i\|_2 = \|b - Ax_{i-1}\|_2 = \|r_{i-1}\|_2$.

Por un lado es evidente que si $r_i = r_{i-1}$ entonces $\|r_i\|_2 = \|r_{i-1}\|_2$. Veamos la implicación en el otro sentido. La aproximación x_i es el elemento que minimiza el residuo en el subespacio $\mathcal{K}^i(A,b)$ y x_{i-1} en el subespacio $\mathcal{K}^{i-1}(A,b) \subset \mathcal{K}^i(A,b)$. Por la unicidad de la aproximación que minimiza el residuo (Proposición 3.1.9), $x_i = x_{i-1}$ y por tanto $r_i = b - Ax_i = b - Ax_{i-1} = r_{i-1}$.

Ahora veremos $|\sin_i| = 1$ si, y solo si, H_i es una matriz singular. Recordemos que $G_i ... G_1 \widetilde{H}_i = \tilde{R}_i = \begin{bmatrix} R_i \\ 0 \end{bmatrix}$. Observemos que

$$G_i ... G_1 \widetilde{H}_i = G_i ... G_1 \left[\begin{array}{ccc|c} & \widetilde{H}_{i-1} & & \begin{matrix} h_{1,i} \\ \vdots \\ h_{i,i} \end{matrix} \\ \hline 0 & \cdots & 0 & h_{i+1,i} \end{array}\right] = G_i \left[\begin{array}{ccc|c} & \tilde{R}_{i-1} & & \begin{matrix} r_{1,i} \\ \vdots \\ r_{i-1,i} \\ \overline{r_{i,i}} \end{matrix} \\ \hline 0 & \cdots & 0 & h_{i+1,i} \end{array}\right] =$$

$$= \left[\begin{array}{ccc|c} & \tilde{R}_{i-1} & & \begin{matrix} r_{1,i} \\ \vdots \\ r_{i-1,i} \\ r_{i,i} \end{matrix} \\ \hline 0 & \cdots & 0 & 0 \end{array}\right] = \begin{bmatrix} R_i \\ 0 \end{bmatrix},$$

donde el ángulo de la rotación G_i queda definido por la relación

$$\begin{bmatrix} \cos_i & \sin_i \\ -\sin_i & \cos_i \end{bmatrix} \begin{bmatrix} \overline{r_{i,i}} \\ h_{i+1,i} \end{bmatrix} = \begin{bmatrix} r_{i,i} \\ 0 \end{bmatrix}. \tag{4.2.2}$$

Supongamos $|\sin_i| = 1$, entonces $\cos_i = 0$ y la ecuación (4.2.2) implica que $\overline{r_{i,i}} = 0$. Consideramos las rotaciones de Givens $G_{i-1}, ..., G_1$ aplicadas a la matriz H_i. Recordemos que las rotaciones de Givens son ortogonales y por tanto el módulo de su determinante es igual a uno. Con estas consideraciones,

$$G_{i-1} ... G_1 H_i = \begin{bmatrix} & & r_{1,i} \\ & \tilde{R}_{i-1} & \vdots \\ & & r_{i-1,i} \\ & & \overline{r_{i,i}} \end{bmatrix} = \begin{bmatrix} & & r_{1,i} \\ & \tilde{R}_{i-1} & \vdots \\ & & r_{i-1,i} \\ & & 0 \end{bmatrix},$$

entonces

$$|det(G_{i-1} ... G_1 H_i)| = |det(G_{i-1} ... G_1)|\,|det(H_i)| = |det(H_i)| = 0.$$

Supongamos que H_i es singular. Entonces la matriz

$$\begin{bmatrix} & & r_{1,i} \\ & \tilde{R}_{i-1} & \vdots \\ & & r_{i-1,i} \\ & & \overline{r_{i,i}} \end{bmatrix} = G_{i-1} ... G_1 H_i,$$

tambien es singular. Recordemos de la expresión (3.1.8) que $\tilde{R}_{i-1}$ tiene rango $i-1$. Entonces para que $G_{i-1} ... G_1 H_i$ sea singular es condición necesaria que $\overline{r_{i,i}} = 0$. De la igualdad (4.2.2), ya que $h_{i+1,i} \neq 0$, se tiene que $\cos_i = 0$ y, por tanto, $|\sin_i| = 1$. □

Corolario 4.2.5. *Se tiene $r_m = b$ si, y solo si, $|\sin_1| = |\sin_2| = ... = |\sin_m| = 1$ o, equivalentemente si, y solo si, H_i es singular para $1 \leq i \leq m$. Es decir,*

$$r_m = b \Longleftrightarrow |\sin_1| = |\sin_2| = ... = |\sin_m| = 1 \Longleftrightarrow H_i \text{ singular } \forall 1 \leq i \leq m.$$

Lema 4.2.6. *(Condición de estancamiento) El residuo tras m iterados de GMRES es igual a b si, y solo si, $h_{1j} = 0$ para todo $1 \leq j \leq m$. Es decir,*

$$r_m = b \iff h_{1j} = 0, \ \forall 1 \leq j \leq m.$$

Demostración. Por inducción sobre m. Para $m = 1$ se reduce a ver que

$$r_1 = b \iff h_{11} = 0,$$

que por la Proposición 4.2.4 es equivalente a

$$H_1 = (h_{11}) \text{ singular } \Longleftrightarrow h_{11} = 0,$$

que es cierto trivialmente.

Para el paso de inducción suponemos

$$r_{m-1} = r_{m-2} = ... = r_1 = b \iff h_{1j} = 0, \ \forall 1 \leq j \leq m-1,$$

y tenemos que ver

$$r_m = r_{m-1} \iff h_{1m} = 0.$$

Por el Corolario 4.2.5,

$$r_{m-1} = b \iff |\sin_1| = |\sin_2| = ... = |\sin_{m-1}| = 1.$$

Así, la rotación de Givens G_j, $1 \leq j \leq m-1$, tiene dos posibles formas

$$G_j = \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & 0 & 1 & & & \\ & & & -1 & 0 & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix} \text{ ó } \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & 0 & -1 & & & \\ & & & 1 & 0 & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ j \\ j+1 \\ \\ \\ \\ \end{matrix},$$

y, por tanto,

$$G_{m-1}...G_1\tilde{H}_m = G_{m-1}...G_1 \left[\begin{array}{c|c} \tilde{H}_{m-1} & \begin{matrix} h_{1m} \\ \vdots \\ h_{mm} \end{matrix} \\ \hline 0 & h_{m+1,m} \end{array}\right] = \left[\begin{array}{c|c} \tilde{R}_{m-1} & \begin{matrix} \pm h_{2m} \\ \vdots \\ \pm h_{mm} \\ \pm h_{1m} \end{matrix} \\ \hline 0 & h_{m+1,m} \end{array}\right].$$

Por la Proposición 4.2.4 tenemos que $r_m = r_{m-1}$ si, y solo si, H_m es singular. Recordemos de la expresión (3.1.8) que

$$\tilde{R}_{m-1} = \begin{bmatrix} R_{m-1} \\ 0 \end{bmatrix} \begin{matrix} (m-1) \times (m-1) \\ 1 \times (m-1) \end{matrix}$$

tiene rango $m-1$. Por tanto,

$$r_m = r_{m-1} \iff \det\left(\begin{bmatrix} & \pm h_{2m} \\ \tilde{R}_{m-1} & \vdots \\ & \pm h_{mm} \\ & \pm h_{1m} \end{bmatrix}\right) = 0 \iff h_{1m} = 0.$$

□

Dado el sistema $Ax = b$ y $m \leq n$, el Lema 4.2.6 nos proporciona una condición necesaria y suficiente para que el residuo permanezca estancado tras m iterados de GMRES, es decir, $r_m = r_{m-1} = ... = r_0 = b$. Utilizando este Lema, ahora queremos caracterizar los vectores $b \in \mathbb{R}^n$ para los cuales dados $A \in \mathbb{R}^{n\times n}$ y $m \leq n$ el residuo permanece estancado tras m pasos de GMRES.

Proposición 4.2.7. *(Ecuaciones de estancamiento) El residuo tras m iterados de GMRES es igual a b si, y solo si, $b^T A^j b = 0$ para todo $1 \leq j \leq m$. Es decir,*

$$r_m = b \iff b^T A^j b = 0, \ \forall 1 \leq j \leq m.$$

Demostración. Por el Lema 4.2.6 tenemos que probar,

$$h_{1j} = 0, \forall 1 \leq j \leq m \iff b^T A^j b = 0, \ \forall 1 \leq j \leq m.$$

Recordemos de la iteración de Arnoldi (Algoritmo 1) que $q_1 = \frac{b}{\|b\|_2}$, y de la Proposición 2.3.10 que $h_{1j} = q_1^T A q_j$. Por inducción sobre j. Para $j = 1$,

$$h_{11} = 0 \iff \left(\frac{b}{\|b\|_2}\right)^T A \left(\frac{b}{\|b\|_2}\right) = 0 \iff b^T A b = 0.$$

Supongamos que

$$h_{1j} = 0, \ \forall 1 \leq j \leq m-1 \iff b^T A^j b = 0, \ \forall 1 \leq j \leq m-1,$$

y veamos que,

$$h_{1m} = 0 \iff b^T A^m b = 0.$$

Por el Lema 2.3.15, tenemos que $q_m = P(A)b$ donde $P \in \mathbb{P}_{m-1}$. Supongamos que el polinomio P viene dado por

$$P(x) = \beta_0 + \beta_1 x + ... + \beta_{m-1} x^{m-1}, \quad \beta_{m-1} \neq 0.$$

Como $b^T A^j b = 0$ para todo $1 \leq j \leq m-1$, entonces,

$$h_{1m} = q_1^T A q_m = \frac{1}{\|b\|_2} b^T A P(A) b = \frac{\beta_{m-1}}{\|b\|_2} b^T A^m b.$$

□

Corolario 4.2.8. *Aplicando la Proposición 4.2.7. Si $A \in \mathbb{R}^{n\times n}$ es definida positiva (o definida negativa) entonces,*

$$\|r_m\|_2 < \|b\|_2, \qquad 1 \leq m \leq n,$$

al aplicar el algoritmo GMRES al sistema $Ax = b$ para todo $b \in \mathbb{R}^n$.

Ejemplo 4.2.9. En [5] y [8] consideran el sistema que ilustraremos a continuación como ejemplo para el cual los residuos quedan estancados hasta la última iteración de GMRES. De esta forma, GMRES(m) no converge para ningún $m < n$. Utilizando las ecuaciones de estancamiento pretendemos demostrar, sin necesidad de realizar el experimento númerico, este hecho para este sistema concreto. Sean,

$$A = \begin{bmatrix} e_2 & e_3 & \cdots & e_n & e_1 \end{bmatrix} \in \mathbb{R}^{n\times n}, \qquad b = e_1 \in \mathbb{R}^n,$$

donde e_i denota el i-ésimo vector de la base canónica. Entonces el sistema $Ax = b$ cumple las ecuaciones de estancamiento de la Proposicion 4.2.7 para todo $m < n$, es decir,

$$b^T A^j b = 0 \text{ para todo } 1 \leq j < n.$$

En efecto, notemos que

$$A^j = \begin{bmatrix} e_{j+1} & e_{j+2} & \cdots & e_n & e_1 & \cdots & e_j \end{bmatrix}, \qquad 1 \leq j \leq n-1$$

y, por tanto,

$$b^T A^j b = e_1^T A^j e_1 = e_1^T e_{j+1} = 0, \qquad 1 \leq j \leq n-1.$$

Observación 4.2.10. El número de condición de la matriz descrita en el Ejemplo 4.2.9 es 1. De esta forma parece que no podemos describir la convergencia de GMRES en términos únicamente del número de condición.

4.3. La iteración GMRES(m) como sistema dinámico

En esta sección denotamos b por b_0. Sea b_i, $i \geq 1$, el residuo que se tiene después de i ciclos de GMRES(m). Así, $b_1 = r_m^{b_0}$, es decir el residuo m-ésimo de GMRES aplicado al sistema $Ax = b_0$. Análogamente, $b_i = r_m^{b_{i-1}}$, $i \geq 1$, es decir el residuo m-ésimo de GMRES aplicado al sistema $Ax = b_{i-1}$.

Por el Lema 4.1.1 tenemos que

$$r_m^{b_{i-1}} = P_m(A) b_{i-1},$$

donde $P_m(x) \in \mathbb{P}_m$ es un polinomio tal que $P_m(0) = 1$ de la forma

$$P_m(x) = 1 + \eta_1 x + ... + \eta_m x^m,$$

siendo $\eta_j = \eta_j(b_{i-1})$, $1 \leq j \leq m$. Así,

$$r_m^{b_{i-1}} = b_{i-1} + \eta_1 A b_{i-1} + ... + \eta_m A^m b_{i-1}.$$

Consideramos la aplicación

$$\begin{aligned} g_{A,m} : \mathbb{R}^n &\longrightarrow \mathbb{R}^n \\ v &\longmapsto g_{A,m}(v) := r_m^v \end{aligned}$$

Así, $g_{A,m}(b_i) = b_{i+1}$, para todo $i \geq 0$. La aplicación $g_{A,m}$ define un sistema dinámico. Observamos que se tiene definido un semiflujo: dado $v \in \mathbb{R}^n$ tenemos unívocamente definida la semiórbita para iterados positivos pero no la semiórbita negativa (hay puntos con diferentes antiimágenes). Por ejemplo, $v = 0$ es imagen de todos los vectores propios de A (para cualquier m).

Definición 4.3.1. *Sean $A \in \mathbb{R}^{n\times n}$ una matriz regular, $b \in \mathbb{R}^n$ y $1 \leq m < n$. Definimos*

$$\mathcal{E}_m^A := \{v \in \mathbb{R}^n \,|\, v^T A^j v = 0, \forall 1 \leq j \leq m\},$$

es decir, el conjunto de vectores de $\mathbb{R}^n$ que cumplen las ecuaciones de estancamiento de la Proposición 4.2.7 para el sistema $Ax = b$ y el parámetro m.

Los puntos fijos cumplen

$$g_{A,m}(v) = r_m^v = v.$$

De la Proposición 4.2.7 concluimos que los puntos fijos cumplen

$$v^T A^j v = 0, \forall 1 \leq j \leq m,$$

Equivalentemente, v es punto fijo de $g_{A,m}$ si, y solo si, $v \in \mathcal{E}_m^A$.

Con la notación introducida, tenemos que GMRES(m) converge para el sistema $Ax = b$ si el ω-límite de la semiórbita positiva de b es $v = 0$. Por otro lado, si el ω-límite de la semiórbita positiva de b es un punto fijo diferente de $v = 0$ entonces GMRES(m) se estanca para el sistema $Ax = b$.

Observación 4.3.2. Cabe preguntarse si el sistema puede tener ω-límites más complejos. Las condiciones

$$\|g_{A,m}(v)\|_2 = \|r_m^v\|_2 = |sin_1 \cdot ... \cdot \sin_m |\|v\|_2 \leq \|v\|_2, \text{ y}$$

$$\|g_{A,m}(v)\|_2 = \|v\|_2 \Longrightarrow g_{A,m}(v) = v,$$

que se deducen de los resultados obtenidos en la Sección 4.2 parecen indicar que no es posible.

Ilustramos dos ejemplos, extraídos de [2], de sistemas de dimensión 3 en los que GMRES(2) se estanca y GMRES(1) converge. Es decir, el ω-límite de la semiórbita positiva de b es $v \in \mathcal{E}_2^A$ para $g_{A,2}$ pero en cambio es 0 para $g_{A,1}$. Estos ejemplos demuestran que aumentar el parámetro m no garantiza una mejor convergencia.

Ejemplo 4.3.3. Sean,

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ -4 \\ 1 \end{bmatrix}.$$

Se tiene que ω-límite de la semiórbita positiva de b para $g_{A,2}$ es

$$v \approx [0{,}776734950525330036797, -0{,}861117410336918354119, 1{,}277455818367493334 26]^T.$$

En cambio para $g_{A,1}$ es 0. Se puede comprobar que v es un punto fijo de $g_{A,2}$, es decir, cumple las ecuaciones de estancamiento $v^T Av = 0$ y $v^T A^2 v = 0$.

La matriz A definida en el Ejemplo 4.3.3 no es diagonalizable. Podríamos pensar que el comportamiento que se observa en la convergencia, siendo GMRES(1) convergente y GMRES(2) no, guarda relación con esta propiedad. El ejemplo siguiente descarta esta posibilidad viendo el mismo comportamiento para una matriz diagonalizable.

Ejemplo 4.3.4. Sean,

$$A = \begin{bmatrix} 1 & 2 & -2 \\ 0 & 2 & 4 \\ 0 & 0 & 3 \end{bmatrix}, \qquad b = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}.$$

Se tiene que ω-límite de la semiórbita positiva de b para $g_{A,2}$ es

$$v \approx [-0{,}29555039355570, 0{,}14377302752433, -0{,}34671023500259]^T.$$

En cambio para $g_{A,1}$ es 0. Se puede comprobar que v es un punto fijo de $g_{A,2}$, es decir, cumple las ecuaciones de estancamiento $v^T Av = 0$ y $v^T A^2 v = 0$.

El artículo [2] que contiene estos dos ejemplos ha servido de inspiración para derivar las ecuaciones de estancamiento en la Sección 4.2 y reinterpretar la iteración de GMRES(m) como sistema dinámico en esta sección.

Apéndice A

Implementación en C de GMRES y GMRES(m)

En este apéndice se proporciona una implementación en lenguaje C de los Algoritmos 4 y 5. El algoritmo GMRES corresponde a la función *gmresUnlimited*() y GMRES(m) corresponde a la función *gmresM*().

```
1
2  /*
3   * Functions to solve Ax=b with gmres method
4   *
5   * by Jose Luis Dorado
6   * University of Barcelona
7   * February 2018
8   */
9
10 #include <string.h>
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <math.h>
14 #include <time.h>
15
16 #define TOL 1e-15
17
18 /*
19  * GMRES algorithm to solve Ax=b
20  *
21  * PARAM:
22  *   dim:        dimension of the system
23  *   b:          b vector
24  *   A_product: the user define the matrix A like a subrutine that computes the
        product of the matrix A and a vector v
25  *               the parameters of this function are:
26  *                 first:  pointer to double that represents the vector v
27  *                 second: integer that is the dimension of the system (wich is the
        same as dim)
28  *                 third:  pointer to double where the function save the result (the
         space is already reserved
29  *                         before the fucntion called)
30  *   rc:         refinement condition of arnoldi (for example 0.25)
31  *   tol:        error tolerance
32  *
33  * RETURN:
34  *   the solution of the system
35  */
36 double* gmresUnlimited( int dim, double* b, void (*A_product)(double*,int,double*),
        double rc, double tol )
37 {
38   int       i;
```

```
39  int       j;
40  int       k;
41  double    aux;
42  double*   x;       /* global solution */
43  double*   bLS;     /* after QR givens (H=QR), Hy=||b||*e1 --> QRy=||b||*e1 --> Ry=
        bLS  */
44  double*   y;       /* solution of Hy=||b||*e1 */
45  double*   sine;    /* sine and cosine represent Q of QR givens */
46  double*   cosine;  /* sine and cosine represent Q of QR givens */
47  double** Q;        /* arnoldi's Q */
48  double** H;        /* arnoldi's H (and R of QR givens) */
49
50  /* allocate Q and H for the arnoldi iteration */
51  /* Q has dim+1 columns (maximum) */
52  Q = (double**)malloc((dim+1)*sizeof(double*));
53  if (Q == NULL)       { puts("memory␣problem"); exit(1); }
54  H = (double**)malloc((dim)*sizeof(double*));
55  if (H == NULL)       { puts("memory␣problem"); exit(1); }
56
57  /* allocate Q[0] */
58  Q[0] = (double*)malloc((dim)*sizeof(double));
59  if (Q[0] == NULL)    { puts("memory␣problem"); exit(1); }
60
61  /* sine and cosine for the QR givens */
62  sine = (double*)malloc((dim)*sizeof(double));
63  if (sine == NULL)    { puts("memory␣problem"); exit(1); }
64  cosine = (double*)malloc((dim)*sizeof(double));
65  if (cosine == NULL) { puts("memory␣problem"); exit(1); }
66
67  /* we use the vector bLS to solve least square problem
68   * we put the solution in the vector y
69  */
70  bLS = (double*)malloc((dim)*sizeof(double));
71  if (bLS == NULL)     { puts("memory␣problem"); exit(1); }
72  y   = (double*)malloc((dim)*sizeof(double));
73  if (y == NULL)       { puts("memory␣problem"); exit(1); }
74
75  /* Q[0] = b/||b|| and bLS[0] = ||b|| */
76  bLS[0] = 0;
77  for (i=0; i<dim; i++) bLS[0]+=b[i]*b[i];
78  bLS[0] = sqrt(bLS[0]);
79  for (i=0; i<dim; i++) Q[0][i]=b[i]/bLS[0];
80
81  /* set iterator 'j' */
82  j=0;
83  /*************GMRES UNLIMITED ITERATIONS******************/
84  while (1)
85  {
86    /***************** ARNOLDI ITERATION ********************/
87    /********************************************************/
88
89    Q[j+1] = (double*)malloc((dim)*sizeof(double));
90    if (Q[j+1] == NULL)    { puts("memory␣problem"); exit(1); }
91    A_product(Q[j],dim,Q[j+1]);
92    /* save ||Q[j+1]|| for the refinement check */
93    aux = 0;
94    for (k=0; k<dim; k++) aux+=Q[j+1][k]*Q[j+1][k];
95    aux = sqrt(aux);
96
97    /* allocate space for next column of H */
98    H[j] = (double*)malloc((2+j)*sizeof(double));
99    if (H[j] == NULL) { puts("memory␣problem"); exit(1); }
100
101   /** orthogonalization **/
102   for (k=0; k<j; k++)
103   {
104     /* product of Q[k]*Q[j+1] */
105     H[j][k] = 0;
106     for (i=0; i<dim; i++) H[j][k]+=Q[k][i]*Q[j+1][i];
107     /* Q[j+1] = Q[j+1] - H[j][k]*Q[k] */
```

```
108       for (i=0; i<dim; i++) Q[j+1][i]=Q[j+1][i]-H[j][k]*Q[k][i];
109     }
110     /* last iteration of the previous "for" with k=j (and code changes) to
            calculate H[j][j] and H[j][j+1]=||Q[j+1]|| */
111     /* product of Q[j]*Q[j+1] */
112     H[j][j] = 0;
113     for (i=0; i<dim; i++) H[j][j]+=Q[j][i]*Q[j+1][i];
114     /* Q[j+1] = Q[j+1] - H[j][i]*Q[i]  and  H[j][j+1] = ||Q[j+1]|| */
115     H[j][j+1] = 0;
116     for (i=0; i<dim; i++) { Q[j+1][i]=Q[j+1][i]-H[j][j]*Q[j][i]; H[j][j+1]+=Q[j+1][
            i]*Q[j+1][i]; }
117     H[j][j+1] = sqrt(H[j][j+1]);
118
119     /*** refinement ***/
120     if ( H[j][j+1]/aux <= rc)
121     {
122       for (k=0; k<j; k++)
123       {
124         /* aux = Q[k]*Q[j+1] */
125         aux = 0;
126         for (i=0; i<dim; i++) aux+=Q[k][i]*Q[j+1][i];
127         /* Q[j+1] = Q[j+1] - aux*Q[k] */
128         for (i=0; i<dim; i++) Q[j+1][i]=Q[j+1][i]-aux*Q[k][i];
129         H[j][k] = H[j][k] + aux;
130       }
131       /* last iteration of the previous "for" with k=j (and code changes) to
              calculate H[j][j] and H[j][j+1]=||Q[j+1]|| */
132       /* aux = Q[j]*Q[j+1] */
133       aux = 0;
134       for (i=0; i<dim; i++) aux+=Q[j][i]*Q[j+1][i];
135       /* Q[j+1] = Q[j+1] - aux*Q[j] and get ||Q[j+1]|| */
136       H[j][j+1] = 0;
137       for (i=0; i<dim; i++) { Q[j+1][i]=Q[j+1][i]-aux*Q[j][i]; H[j][j+1]+=Q[j+1][i
              ]*Q[j+1][i]; }
138       H[j][j] = H[j][j] + aux;
139       /* H[j][j+1] = ||Q[j+1]|| */
140       H[j][j+1] = sqrt(H[j][j+1]);
141     }
142
143     /************************** BREAKDOWN ****************************/
144     /**** if H[j][j+1]=0 or we have a 'dim' iterations --> break ****/
145     if ( fabs(H[j][j+1]) < tol || j == dim-1) { break; }
146
147     /* normalize Q[j+1] */
148     for (i=0; i<dim; i++) Q[j+1][i] = Q[j+1][i]/H[j][j+1];
149
150     /** GIVENS ROTATIONS TO FIND R AND bLS TO SOLVE LEAST SQUARE PROBLEM  **/
151     /*********************************************************************/
152
153     /* compute the product of the previous Givens rotations with H[j] */
154     for (k=0; k<j; k++)
155     {
156       aux       = H[j][k];
157       H[j][k]   = aux*cosine[k] + H[j][k+1]*sine[k];
158       H[j][k+1] = H[j][k+1]*cosine[k] - aux*sine[k];
159     }
160
161     /* compute and apply Givens rotation to put 0 in the subdiagonal */
162     if ( fabs(H[j][j+1]) > fabs(H[j][j]) ) { aux=H[j][j]/H[j][j+1]; sine[j]=1./sqrt
            (1+aux*aux); cosine[j]=sine[j]*aux;  }
163     else                                   { aux=H[j][j+1]/H[j][j]; cosine[j]=1./
            sqrt(1+aux*aux); sine[j]=cosine[j]*aux; }
164     /* apply the rotation found (remember that bLS[i+1]=0) */
165     H[j][j]  = H[j][j]*cosine[j] + H[j][j+1]*sine[j];
166     bLS[j+1] = -bLS[j]*sine[j];
167     bLS[j]   = bLS[j]*cosine[j];
168
169     /* increase iterator for the new iteration */
170     j++;
171   }
```

```
172   /************ END GMRES UNLIMITED ITERATIONS ************/
173
174   /* compute last R column of QR givens*/
175   /************************************/
176   for (k=0; k<j; k++)
177   {
178     aux       = H[j][k];
179     H[j][k]   = aux*cosine[k] + H[j][k+1]*sine[k];
180     H[j][k+1] = H[j][k+1]*cosine[k] - aux*sine[k];
181   }
182
183   /*********** SOLVE LEAST SQUARE PROBLEM ***************/
184   /******************************************************/
185   /* solve LS problem solving the system Ry=bLS */
186   for (k=j; k>=0; k--)
187   {
188     y[k] = bLS[k];
189     for (i=k+1; i<=j; i++) y[k]-=y[i]*H[i][k];
190     y[k] = y[k]/H[k][k];
191   }
192
193   /* we use the space allocated in Q[j+1] for the solution */
194   x = Q[j+1];
195
196   /* store the solution in x // x=Qy */
197   for (k=0; k<dim; k++)
198   {
199     x[k] = 0;
200     for (i=0; i<=j; i++) x[k]+=Q[i][k]*y[i];
201   }
202
203   /*** FREE MEMORY ***/
204   for(i=0; i<=j; i++) { free(H[i]); free(Q[i]); }
205   free(Q);
206   free(H);
207   free(sine);
208   free(cosine);
209   free(bLS);
210   free(y);
211
212   return x;
213 }
214
215
216 /*
217  * RESTARTED GMRES algorithm to solve Ax=b
218  *
219  * PARAM:
220  *   m:          restart parameter
221  *   dim:        dimension of the system
222  *   b:          b vector
223  *   A_product: the user define the matrix A like a subrutine that computes the
       product of the matrix A and a vector v
224  *               the parameters of this function are:
225  *                 first:  pointer to double that represents the vector v
226  *                 second: integer that is the dimension of the system (wich is the
       same as dim)
227  *                 third:  pointer to double where the function save the result (the
        space is already reserved
228  *                         before the fucntion called)
229  *   rc:         refinement condition of arnoldi (for example 0.25)
230  *   tol:        error tolerance
231  *   error:      we put in this pointer the error vector of the solution returned (
       it is not necessary to allocate previously)
232  *               we ignore if the input is NULL
233  *   x:          we put in this pointer the best approximation of the solution that
       we have found
234  *               (it is necessary to allocate previously)
235  *
236  * RETURN:
```

```
237  *      0  if we have found a solution with the 'tol' tolerance
238  *      -1 if we have not found a solution with the 'tol' tolerance
239  */
240 int gmresM( int m, int dim, double* b, void (*A_product)(double*,int,double*),
        double rc, double tol, double** error , double* x)
241 {
242   int       i;
243   int       j;
244   int       k;
245   double    aux;
246   int       mLS;      /* least square system dimension */
247   double*   err;      /* iterative error vector */
248   double*   bLS;      /* vector of Least Square system Ry=bLS  */
249   double*   y;        /* Least Square problem solution */
250   double*   sine;     /* sine and cosine represent Q of QR givens */
251   double*   cosine;   /* sine and cosine represent Q of QR givens */
252   double** Q;         /* arnoldi's Q */
253   double** H;         /* arnoldi's H (and R of QR givens) */
254   double    previousbNorm; /* to the stagnation check */
255
256   previousbNorm = -1;
257
258   /* set least square system dimension as m-1 */
259   mLS     = m-1;
260
261   /* allocate error vector and copy b vector inside */
262   err = (double*)malloc(dim*sizeof(double));
263   if (err == NULL)       { puts("memory␣problem"); exit(1); }
264   for(i=0; i<dim; i++)   { err[i]=b[i]; }
265
266   /* set solution x=0 */
267   for(i=0; i<dim; i++)   { x[i]=0; }
268
269   /* allocate Q and H for the arnoldi iteration */
270   Q = (double**)malloc((m+1)*sizeof(double*));
271   if (Q == NULL)         { puts("memory␣problem"); exit(1); }
272   H = (double**)malloc((m)*sizeof(double*));
273   if (H == NULL)         { puts("memory␣problem"); exit(1); }
274   /* allocate columns of Q */
275   for (i=0; i<=m; i++)
276   {
277     Q[i] = (double*)malloc((dim)*sizeof(double));
278     if (Q[i] == NULL)    { puts("memory␣problem"); exit(1); }
279   }
280   /* allocate columns of H */
281   for (i=0; i<m; i++)
282   {
283     H[i] = (double*)malloc((i+2)*sizeof(double));
284     if (H[i] == NULL)    { puts("memory␣problem"); exit(1); }
285   }
286
287   /* sine and cosine for the QR givens and least square problem */
288   sine = (double*)malloc((m)*sizeof(double));
289   if (sine == NULL)      { puts("memory␣problem"); exit(1); }
290   cosine = (double*)malloc((m)*sizeof(double));
291   if (cosine == NULL)    { puts("memory␣problem"); exit(1); }
292
293   /* we use the vector bLS to solve least square problem */
294   /* we put the solution in the vector y */
295   bLS = (double*)malloc((m+1)*sizeof(double));
296   if (bLS == NULL)       { puts("memory␣problem"); exit(1); }
297   y   = (double*)malloc((m)*sizeof(double));
298   if (y == NULL)         { puts("memory␣problem"); exit(1); }
299
300   /*** PREPARING THE FIRST GMRES ITERATION ***/
301   /* bLS[0] = ||b|| */
302   bLS[0] = 0;
303   for (i=0; i<dim; i++) bLS[0]+=b[i]*b[i];
304   bLS[0] = sqrt(bLS[0]);
305
```

```
306   /****** RESTARTED GMRES LOOP ******/
307   /* BREAKDOWN if overall error is less than the tolerance */
308   while( bLS[0] > tol )
309   {
310
311     /* Q[0] = err/||err|| */
312     for (i=0; i<dim; i++) Q[0][i]=err[i]/bLS[0];
313
314     /****************** GMRES m ITERATIONS ******************/
315     for(j=0; j<m; j++)
316     {
317       /***************** ARNOLDI ITERATION ******************/
318       /******************************************************/
319       A_product(Q[j],dim,Q[j+1]);
320       /* save ||Q[j+1]|| for the refinement check */
321       aux = 0;
322       for (k=0; k<dim; k++) aux+=Q[j+1][k]*Q[j+1][k];
323       aux = sqrt(aux);
324
325       /** orthogonalization **/
326       for (k=0; k<j; k++)
327       {
328         /* product of Q[k]*Q[j+1] */
329         H[j][k] = 0;
330         for (i=0; i<dim; i++) H[j][k]+=Q[k][i]*Q[j+1][i];
331         /* Q[j+1] = Q[j+1] - H[j][k]*Q[k] */
332         for (i=0; i<dim; i++) Q[j+1][i]=Q[j+1][i]-H[j][k]*Q[k][i];
333       }
334       /* last iteration of the previous "for" with k=j (and code changes) to
              calculate H[j][j] and H[j][j+1]=||Q[j+1]|| */
335       /* product of Q[j]*Q[j+1] */
336       H[j][j] = 0;
337       for (i=0; i<dim; i++) H[j][j]+=Q[j][i]*Q[j+1][i];
338       /* Q[j+1] = Q[j+1] - H[j][i]*Q[i]  and  H[j][j+1] = ||Q[j+1]|| */
339       H[j][j+1] = 0;
340       for (i=0; i<dim; i++) { Q[j+1][i]=Q[j+1][i]-H[j][j]*Q[j][i]; H[j][j+1]+=Q[j
              +1][i]*Q[j+1][i]; }
341       H[j][j+1] = sqrt(H[j][j+1]);
342
343       /*** refinement ***/
344       if ( H[j][j+1]/aux <= rc)
345       {
346         for (k=0; k<j; k++)
347         {
348           /* aux = Q[k]*Q[j+1] */
349           aux = 0;
350           for (i=0; i<dim; i++) aux+=Q[k][i]*Q[j+1][i];
351           /* Q[j+1] = Q[j+1] - aux*Q[k] */
352           for (i=0; i<dim; i++) Q[j+1][i]=Q[j+1][i]-aux*Q[k][i];
353           H[j][k] = H[j][k] + aux;
354         }
355         /* last iteration of the previous "for" with k=j (and code changes) to
                calculate H[j][j] and H[j][j+1]=||Q[j+1]|| */
356         /* aux = Q[j]*Q[j+1] */
357         aux = 0;
358         for (i=0; i<dim; i++) aux+=Q[j][i]*Q[j+1][i];
359         /* Q[j+1] = Q[j+1] - aux*Q[j] and get ||Q[j+1]|| */
360         H[j][j+1] = 0;
361         for (i=0; i<dim; i++) { Q[j+1][i]=Q[j+1][i]-aux*Q[j][i]; H[j][j+1]+=Q[j+1][
                i]*Q[j+1][i]; }
362         H[j][j] = H[j][j] + aux;
363         /* H[j][j+1] = ||Q[j+1]|| */
364         H[j][j+1] = sqrt(H[j][j+1]);
365       }
366
367       /***** if H[j][j+1] = 0 --> go to solve Least Square problem *****/
368       if ( fabs(H[j][j+1]/H[j][j]) < tol )
369       {
370         /* compute last R column */
371         /*************************/
```

```
372         for (k=0; k<j; k++)
373         {
374           aux        = H[j][k];
375           H[j][k]    = aux*cosine[k] + H[j][k+1]*sine[k];
376           H[j][k+1]  = H[j][k+1]*cosine[k] - aux*sine[k];
377         }
378         mLS      = j;
379         break;
380       }
381
382       /* normalize Q[j+1] */
383       for (i=0; i<dim; i++) Q[j+1][i] = Q[j+1][i]/H[j][j+1];
384
385       /** GIVENS ROTATIONS TO FIND R AND bLS TO SOLVE LEAST SQUARE PROBLEM  **/
386       /*************************************************************************/
387
388       /* compute the product of the previous Givens rotations with H[j] */
389       for (k=0; k<j; k++)
390       {
391         aux        = H[j][k];
392         H[j][k]    = aux*cosine[k] + H[j][k+1]*sine[k];
393         H[j][k+1]  = H[j][k+1]*cosine[k] - aux*sine[k];
394       }
395
396       /* compute and apply Givens rotation to put 0 in the subdiagonal */
397       if ( fabs(H[j][j+1]) > fabs(H[j][j]) ) { aux=H[j][j]/H[j][j+1]; sine[j]=1./
              sqrt(1+aux*aux); cosine[j]=sine[j]*aux;  }
398       else                                    { aux=H[j][j+1]/H[j][j]; cosine[j]=1./
              sqrt(1+aux*aux); sine[j]=cosine[j]*aux; }
399       /* apply the rotation found (remember that bLS[i+1]=0) */
400       H[j][j]  = H[j][j]*cosine[j] + H[j][j+1]*sine[j];
401       bLS[j+1] = -bLS[j]*sine[j];
402       bLS[j]   = bLS[j]*cosine[j];
403
404     }/***************** END m ITERATIONS OF GMRES ****************/
405
406     /*********** SOLVE LEAST SQUARE PROBLEM ***************/
407     /******************************************************/
408     /* solve LS problem solving the system Ry=bLS */
409     for (k=mLS; k>=0; k--)
410     {
411       y[k] = bLS[k];
412       for (i=k+1; i<=mLS; i++) y[k]-=y[i]*H[i][k];
413       y[k] = y[k]/H[k][k];
414     }
415
416     /* compute next aproximation of the solution */
417     for (k=0; k<dim; k++)
418       for (i=0; i<=mLS; i++) x[k]+=Q[i][k]*y[i];
419
420     /* compute the error and prepare the next iteration */
421     A_product(x,dim,err);
422     for(i=0; i<dim; i++) err[i]=b[i]-err[i];
423
424     /****** PREPARING THE NEXT ITERATION *******/
425     /* Q[0] = err/||err|| and bLS[0] = ||err|| */
426     bLS[0] = 0;
427     for (i=0; i<dim; i++) bLS[0]+=err[i]*err[i];
428     bLS[0] = sqrt(bLS[0]);
429
430     /****** STAGNATION CONDITION ******/
431     if( fabs( (bLS[0]-previousbNorm)/previousbNorm ) < TOL )
432     {
433       /*** save error vector or free it ***/
434       if(error) *error=err;
435       else      free(err);
436
437       /*** FREE MEMORY ***/
438       for(i=0; i<m; i++) { free(H[i]); free(Q[i]); }
439       free(Q[m]);
```

```
440         free(Q);
441         free(H);
442         free(sine);
443         free(cosine);
444         free(bLS);
445         free(y);
446 
447         /*** we have not obtained the wanted precision ***/
448         return -1;
449       }
450 
451     }/** END RESTARTED GMRES LOOP **/
452 
453     /*** save error vector or free it ***/
454     if(error) *error=err;
455     else      free(err);
456 
457     /*** FREE MEMORY ***/
458     for(i=0; i<m; i++) { free(H[i]); free(Q[i]); }
459     free(Q[m]);
460     free(Q);
461     free(H);
462     free(sine);
463     free(cosine);
464     free(bLS);
465     free(y);
466 
467     /*** we have get the wanted precision ***/
468     return 0;
469 }
```

Bibliografía

[1] Demmel, J. W.: *Applied Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997.

[2] Embree, M.: The Tortoise and the Hare Restart GMRES, *SIAM review*, 45(2):259-266, 2003.

[3] Golub, G. H.; Van Loan, C. F.: *Matrix Computations*, The Johns Hopkins University Press, cuarta edición, 2013.

[4] Ipsen, I.; Meyer C.: The Idea Behind Krylov Methods *The American Mathematical Monthly*, 105:889-899, 1998.

[5] Saad, Y.: *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, segunda edición, 2003.

[6] Saad, Y.; Schultz, M. H.: GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems *SIAM J. ScI. STAT. COMPUT.*, 7(3):856-869, 1986.

[7] Trefethen, L. N.; Bau, D.: *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997.

[8] Van der Vost, H. A.: *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monographs on applied and computational mathematics, No. 13, Cambridge University Press, 2003.

[9] Van der Vost, H. A.; Vuik, C.: The superlinear convergence behaviour of GMRES, *Journal of Computational and Applied Mathematics*, 48:327-341, 1993.