Métodos Iterativos No Estacionarios

por

Bibiana López Rodríguez

Trabajo presentado como requisito parcial
para optar al Título de

Magister en Matemáticas

Director: Carlos Enrique Mejía Salazar

Universidad Nacional de Colombia
Sede Medellín

Facultad de Ciencias

Escuela de Matemáticas

Diciembre 2003

Este trabajo ha sido apoyado parcialmente la Dirección Nacional de Investigaciones de la Universidad Nacional de Colombia,
Proyecto DI00C1236.

Resumen

Se estudian métodos iterativos para la solución de un sistema $Ax = b$, cuando la matriz A posee características especiales, desde dos puntos de vista, teórico y práctico. Con relación al primero, se concentra la atención sobre la fundamentación matemática de los métodos, enfocándolos desde el punto de vista general utilizando subespacios de Krylov. Con relación al segundo se ofrecen rutinas MATLAB originales que los implementan y se ilustran los resultados con la solución de ejemplos seleccionados. Las rutinas que preparamos son eficientes en el sentido de usar poco almacenamiento en memoria y requerir tiempos cortos de ejecución.

Contenido

Agradecimientos

Quiero expresar mis agradecimientos a Dios ... a mi familia, en especial a mis padres por su constante e incondicional apoyo. A Carlos Enrique Mejía Salazar, profesor de la Universidad Nacional de Colombia y director de este trabajo, su confianza, orientación y valiosos aportes. A Margarita María Toro Villegas, profesora de la Universidad Nacional de Colombia y jurado de este trabajo. A Francisco Mauricio Toro Botero, profesor de la Universidad Nacional de Colombia y jurado de este trabajo. A mis profesores y todas aquellas personas que de una u otra forma colaboraron con la realización de este trabajo.

Introducción

Queremos estudiar métodos iterativos para resolver el problema $Ax = b$ con A matriz no singular. Por un método iterativo, entendemos aquel en el que se construye una sucesión de aproximaciones a la solución. Existen dos clases de métodos iterativos: Métodos estacionarios los cuales son más antiguos y más simples de comprender pero, usualmente, menos efectivos y los Métodos no estacionarios, relativamente recién desarrollados, su análisis usualmente es más difícil de comprender pero usualmente más efectivos. Los métodos no estacionarios están basados en la idea de sucesiones de vectores ortogonales. La rapidez en la convergencia del método depende grandemente del espectro de la matriz de coeficientes.

Los métodos iterativos usualmente envuelven una segunda matriz que transforma la matriz de coeficientes en una con espectro más favorable. La matriz transformada se llama precondicionada. El uso de un buen precondicionamiento lleva a que la convergencia del método iterativo sea más rápida.

Uno de los métodos iterativos no estacionarios que más nos compete es el Gradiente Conjugado (CG). El método del CG fue presentado por Hestenes y Stiefel (1952) como un método directo, mas tarde Reid (1971) y Concus et al (1976) le descubrieron su verdadero potencial al mirarlo como un método iterativo bastante adecuado para resolver sistemas de la forma $Ax = b$, donde A es simétrica y definida positiva. El método de CG está basado en un principio de conjugación teniendo un almacenamiento modesto y siempre converge. Subsiguientemente, la investigación se orientó a estudiar el problema cuando la matriz A no tiene alguna o ninguna de las características anteriores. Así que una considerable parte de la investigación ha sido implementada para desarrollar generalizaciones del CG o para construir estrategias de precondicionamiento.

Los métodos Generalized Minimal Residual (GMRES), aplicable a matrices no simétricas y Minimal Residual (MINRES), aplicable a matrices simétricas y no definidas positivas, están basados en un principio de minimización, el almacenamiento es elevado y siempre convergen.

Entre los métodos que han sido construidos como generalizaciones del CG tenemos:
a. Biconjugate Gradient (BiCG) aplicable a sistemas no simétricos, está basado en un principio de conjugación, genera dos sucesiones de vectores bi-ortogonales teniendo un almacenamiento modesto, pero tiene la desventaja de que hay casos en que el método fracasa.
b. Quasi-Minimal Residual (QMR) diseñado para sistemas no simétricos y para mejorar la convergencia de BiCG.
c. Conjugate Gradient Squared (CGS) aplicable a matrices no simétricas fue diseñado para evitar el uso de la matriz traspuesta de A y ganar rapidez en BiCG, al igual que BiCG hay casos en que el método fracasa.

d. Biconjugate Gradient Stabilized (Bi-CGSTAB) es una variación de CGS para mejorar su convergencia. Decidir cual de ellos usar, en un momento determinado, depende del problema particular o de la estructura de los datos.

En este trabajo nos concentramos en CG y GMRES y hacemos referencias ocasionales a otros métodos. Nuestro objetivo es comprender la teoría subyacente y preparar rutinas eficientes para el cálculo. La eficiencia la conseguimos gracias a que las matrices con las que trabajamos tienen estructura de matriz por bandas o tridiagonal por bloques y en lugar de almacenarlas, usamos rutinas para definir su acción.

Para todos los métodos iterativos que estudiamos preparamos ejemplos ilustrativos y presentamos comparaciones con los resultados obtenidos directamente por MATLAB. El alcance del trabajo lo limitamos a la consideración de los métodos y no introducimos estrategias de precondicionamiento. Los programas están a disposición en la sección *documentos* de la página web

$$http://www.unalmed.edu.co/\tilde{}cemejia/.$$

Este trabajo está organizado así: el capítulo 1 lo dedicamos a Gradiente Conjugado y sus variaciones para ecuaciones normales. Después, en el capítulo 2 presentamos los métodos GMRES y MINRES y en el capítulo final discutimos con menos detalle otros métodos. En todos los capítulos hay ejemplos y comparaciones que ilustran el comportamiento de los algoritmos estudiados.

Capítulo 1

GRADIENTE CONJUGADO (CG)

Los métodos de subespacios de Krylov se usan para resolver sistemas de ecuaciones lineales $Ax = b$ y encontrar los valores propios de A. Nosotros trabajaremos sobre la primera de estas ideas.

En los algoritmos de Krylov suponemos que A tiene estructura y es posible conocer la acción de la matriz sobre un vector, es decir, funciones que retornan $z = Av$ para un v dado.

Este capítulo esta organizado como sigue. En la Sección 1.1 de los preliminares,enunciamos las ideas básicas y definiciones sobre métodos de Krylov. Enseguida, en la Sección 1.2 damos un repaso de los métodos iterativos estacionarios, posteriormente en la sección 1.3 vemos una descripción de la forma de los métodos no estacionarios, que continúa en la sección 1.4 con una descripción detallada del método CG y unos casos particulares cuando la matriz A no cumple la condición de ser simétrica definida positiva. En la última sección describimos el precondicionamiento para el método de CG con miras a mejorar su tiempo de convergencia y terminamos con un pequeño comentario sobre la generalización del método CG para problemas en dimensión infinita.

La teoría que aquí se trabaja esta basada en el libro de Kelley [2] y para los resultados numéricos se utilizan rutinas propias que no requieren el almacenamiento de la matriz, solamente la acción de la matriz. Las rutinas las preparamos con base en los algoritmos de Barrett et al [5].

1.1 Preliminares

Definición 1.1.1 *Denotaremos $\mathbb{M}_n(\mathbb{R})$ como el conjunto de matrices cuadradas de orden n con elementos en $\mathbb{R}$.*

a. *Una matriz simétrica $A \in \mathbb{M}_n(\mathbb{R})$ es definida positiva si $x^T Ax > 0$, para todo $x \neq \mathbf{0}$.*

b. *Para* $A \in \mathbb{M}_n(\mathbb{R})$, *la función*

$$\begin{aligned} f: \quad & \mathbb{R}^n \to \mathbb{R}^n \\ & x \longmapsto x^T A x \end{aligned}$$

se llama forma cuadrática.

c. *Dada una norma vectorial* $\|.\|$, *tenemos que la norma matricial inducida es*

$$\|A\| = \sup\{\|Au\| : u \in \mathbb{R}^n, \|u\| = 1\}.$$

d. *El número de condición de una matriz* A *se denota por* $k(A)$ *y se define*

$$k(A) = \|A\| \left\|A^{-1}\right\|.$$

Si A *es singular,* $k(A) = \infty$. *Si* $k(A)$ *es grande se dice que* A *es mal condicionada.*

e. *Si* x *es la solución del sistema* $Ax = b$ *y* $\widetilde{x}$ *es una aproximación de la solución decimos que:*

$r = b - A\widetilde{x}$ *es el vector residual*

$e = x - \widetilde{x}$ *es el vector error y*

$\frac{\|x-\widetilde{x}\|}{\|x\|}$ *es el error relativo en* $\widetilde{x}$.

Teorema 1.1.1 *(Cayley Hamilton) Sea* $p(t)$ *el polinomio característico de* $A \in \mathbb{M}_n(\mathbb{R})$. *Entonces* $p(A) = 0$.

Teorema 1.1.2 *Si* $A \in \mathbb{M}_n$ *tiene valores propios* $\lambda_1, \lambda_2, ..., \lambda_n$, *las siguientes afirmaciones son equivalentes:*

(a) A *es normal.*

(b) A *es ortogonalmente diagonalizable.*

(c) *Existe un conjunto ortonormal de* n *vectores propios de* A.

Como simetría implica normalidad, las matrices simétricas son ortogonalmente diagonalizables.

Definición 1.1.2 *Un método de proyección para resolver un sistema lineal* $Ax = b$ *es un*

procedimiento que construye aproximaciones $x_k \in V_k$ bajo las condiciones

$$r_k = b - Ax_k \perp W_k \tag{1.1}$$

donde V_k y W_k son subespacios de dimensión k de $\mathbb{R}^n$. En el caso $W_k = V_k$ se tiene un método de proyección ortogonal y (1.1) *se llama condición Galerkin y el caso general $W_k \neq V_k$ se llama método de proyección oblicua con una condición de Petrov-Galerkin en la ecuación* (1.1) .

Definición 1.1.3 *Un método de minimización de norma para resolver un sistema lineal $Ax = b$ es un procedimiento que construye soluciones aproximadas $x_k \in V_k$ que cumplen*

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \min_{x \in V_k} \{\|b - Ax\|_2\},$$

donde V_k es un subespacio de dimensión k de $\mathbb{R}^n$.

Para los subespacios V_k partimos de una aproximación inicial x_0 y V_k será un subespacio afín, ésto es, $V_k = x_0 + \widetilde{V_k}$. Existen varias posibilidades de escoger $\widetilde{V_k}$, una de ellas es tomar subespacios de Krylov pues éstos tienen propiedades ventajosas.

Definición 1.1.4 *Un método de subespacio de Krylov es un método de proyección ó un método de minimización de norma para resolver el sistema lineal $Ax = b$, donde $\widetilde{V_k}$ es un subespacio de Krylov*

$$\widetilde{V_k} = K_k = gen\left\{r_0, Ar_0, ..., A^{k-1}r_0\right\}$$

$k = 1, 2, ...$ *y* $r_0 = b - Ax_0$.

Para trabajar eficientemente sobre estos subespacios necesitamos construir una base ortonormal. Los algoritmos de Arnoldi y Lanczos son métodos que computan bases ortonormales para espacios de Krylov.

1.2 Métodos Estacionarios

Un método iterativo estacionario para la solución del sistema $Ax = b$, con A no singular, tiene la forma

$$x^{(k+1)} = Mx^{(k)} + C$$

con C y M matrices no dependientes de k. Generalmente se plantea con base en una matriz no singular Q, llamada matriz de descomposición y el problema $Ax = b$ es equivalente a

$$Qx = (Q - A)\, x + b.$$

Siguiendo el proceso iterativo

$$Qx^{(k+1)} = (Q - A)\, x^{(k)} + b,$$

para que el método sea convergente para x_0 arbitrario debemos escoger Q de manera que se pueda calcular fácilmente la sucesión $\left\{x^{(k)}\right\}$ y la sucesión converja rápidamente a la solución. Nótese que

$$x^{(k+1)} = \left(I - Q^{-1}A\right) x^{(k)} + Q^{-1}b.$$

Ejemplos de métodos iterativos estacionarios para matrices A con diagonal no nula, son:
Jacobi: la matriz $Q = diag\,(A)$ y la iteración está dada por

$$x_i^{(k+1)} = \left(b_i - \sum_{j \neq i}^{n} a_{ij} x_j^{(k)} \right) / a_{ii}$$

Gauss-Seidel: la matriz Q es la parte triangular inferior de A incluyendo la diagonal y la iteración está dada por

$$x_i^{(k+1)} = \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} \right) / a_{ii}$$

SOR: método de sobrerrelajación sucesiva y la matriz $Q = \alpha D - C$, α real, D una matriz simétrica y definida positiva y con C tal que $C + C^H = D - A$. Comúnmente el método de SOR se conoce cuando D es la diagonal de A y $-C$ es la parte triangular inferior de A excluyendo la diagonal. Generalmente $\alpha = 1/w$ y $0 < w < 2$. La iteración de SOR es

$$x_i^{(k+1)} = (1 - w)\, x_i^{(k)} + \frac{w}{a_{ii}} \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} \right)$$

1.3 Métodos no Estacionarios

Tienen la forma

$$x_{k+1} = x_k + \alpha_k p_k$$

donde α_k y p_k son escalar y vector respectivamente y dependen del número de iteración k. Aquí el subíndice corresponde al número de iteración y la igualdad es entre vectores y no entre componentes. Un ejemplo de estos métodos es el Método de Richardson

$$x_{k+1} = x_k + \alpha_k r_k$$

donde

$$r_k = b - Ax_k \quad \text{y} \quad \alpha_k = -\frac{r_k^T r_k}{r_k^T A r_k}.$$

1.4 Gradiente Conjugado (CG)

El primero de los métodos del tipo Krylov es el método CG, desarrollado por Hestenes y Stiefel en 1952, [10]. Se usa para resolver sistemas de ecuaciones lineales con A matriz simétrica definida positiva.

Su nombre viene del siguiente hecho: En el proceso se generan vectores p_{k+1} que cumplen $p_{k+1}^T Aq = 0$ para todo $q \in K_k$. A esta condición se le llama A-conjugación.

Además, la $k\_ésima$ iteración x_k es el minimizador de un funcional cuadrático Φ sobre $x_0 + K_k$. El gradiente de este funcional tiene la misma dirección del vector residual r_k obtenido en la iteración. Más adelante nos volveremos a referir a este tema en más detalle.

1.4.1 Métodos de Krylov y la propiedad de minimización

Los métodos de Krylov CG y GMRES, minimizan la medida del error en la k-ésima iteración sobre un espacio afín

$$x_0 + K_k$$

donde x_0 es la iteración inicial y K_k es el k-ésimo subespacio de Krylov

$$K_k = gen\left\{r_0, Ar_0, ..., A^{k-1}r_0\right\} \qquad k \geq 1$$

donde $r_0 = b - Ax_0$. Denotamos a $\{r_k\}$ la sucesión de residuales con $r_k = b - Ax_k$. Suponemos que A es no singular $n \times n$ y $x^* = A^{-1}b$. El Gradiente Conjugado (CG) iterativo fue inventado como un método directo, pero desde 1980 más o menos, se implementa como un método iterativo y en gran medida ha tomado el lugar de la familia de los métodos iterativos estacionarios como Jacobi, Gauss-Seidel, SOR. El CG sirve para resolver sistemas simétricos positivamente definidos (spd). Definimos para A spd la A-norma: $\|x\|_A = \sqrt{x^T A\ x}$.

Comenzamos con una descripción del algoritmo y las consecuencias de la minimización de los errores en las iteraciones, describiremos un criterio de terminación, ejecución y la implementación, acompañados de unos ejemplos ilustrativos. Los datos de partida para el algoritmo son: la matriz de coeficientes A, el vector del lado derecho b, la aproximación inicial x_0, los valores tol y max_it que corresponden al máximo error que se permitirá y un número máximo de iteraciones correspondientemente. Los algoritmos son tomados de Barrett et al [5].

Algoritmo de CG

Lea A, x_0, b, tol, max_it

$r_0 = b - Ax_0$

$ep = tol * \|b\|_2$

para $k = 1, 2, ..., max\_it$

$\rho_{k-1} = r_{k-1}^T r_{k-1}$

si $k = 1$

$p_1 = r_0$

en otro caso

$\beta_k = \frac{\rho_{k-1}}{\rho_{k-2}}$

$p_k = r_{k-1} + \beta_k p_{k-1}$

fin

$q_k = A * p_k$

$\alpha_k = \frac{\rho_{k-1}}{p_k^T q_k}$

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

$r_k = r_{k-1} - \alpha_k q_k$

si $\|r_k\|_2 \leq ep$ *entonces termine*

fin

fin

En síntesis

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

donde

$$\alpha_k = \frac{r_{k-1}^T r_{k-1}}{p_k^T q_k}$$

y

$$\begin{aligned} p_k &= r_{k-1} + \beta_{k-1} p_{k-1} \\ \beta_k &= \frac{\rho_{k-1}}{\rho_{k-2}} = \frac{r_{k-1}^T r_{k-1}}{r_{k-2}^T r_{k-2}} \end{aligned}$$

Definimos la función:

$$\Phi(x) = \frac{1}{2} x^T A x - x^T b \tag{1.2}$$

la $k\_$*ésima* iteración de CG es el minimizador de esta función sobre $x_0 + K_k$. Vemos que si $\Phi(\widetilde{x})$ es el valor mínimo, entonces

$$\nabla \Phi(\widetilde{x}) = A\widetilde{x} - b = 0$$

y así $\widetilde{x} = x^*$.

Minimizar Φ sobre cualquier subconjunto de $\mathbb{R}^n$ es lo mismo que minimizar $\|x - x^*\|_A$ sobre ese conjunto.

Lema 1.4.1 *Sea* $S \subset \mathbb{R}^n$. *Si* x_k *minimiza* Φ *sobre* S *entonces* x_k *también minimiza* $\|x - x^*\|_A = \|r\|_{A^{-1}}$ *sobre* S.

Prueba.

$$\|x - x^*\|_A^2 = (x - x^*)^T A (x - x^*) = x^T A x - x^T A x^* - (x^*)^T A x + (x^*)^T A x^*$$

como A es simétrica y $Ax^* = b$

$$-x^T Ax^* - (x^*)^T Ax = -2x^T Ax^* = -2x^T b$$

por lo tanto

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

como $(x^*)^T Ax^*$ es independiente de x, minimizar Φ es equivalente a minimizar $\|x - x^*\|_A^2$ y a minimizar $\|x - x^*\|_A$. Ahora, dado que $e = x - x^*$ entonces

$$\begin{aligned} \|e\|_A &= e^T Ae = (Ae)^T e = (Ae)^T A^{-1} Ae = \|Ae\|_{A^{-1}} \\ &= \|Ax - b\|_{A^{-1}} = \|b - Ax\|_{A^{-1}} = \|r\|_{A^{-1}} \end{aligned}$$

así la A-norma del error es también la A^{-1} -norma del residual. ■

Usaremos el lema en particular cuando $S = x_0 + K_k$.

1.4.2 Consecuencias de la propiedad de minimización

Del lema anterior si x_k minimiza Φ sobre $x_0 + K_k$ se cumple

$$\|x^* - x_k\|_A \leq \|x^* - w\|_A$$

para todo $w \in x_0 + K_k$. Todo $w \in x_0 + K_k$ se puede escribir

$$w = x_0 + \sum_{j=0}^{k-1} \gamma_j A^j r_0$$

para algunos coeficientes $\{\gamma_j\}$ y podemos expresar a $x^* - w$ por

$$x^* - w = x^* - x_0 - \sum_{j=0}^{k-1} \gamma_j A^j r_0$$

como $Ax^* = b$ y $r_0 = b - Ax_0 = A(x^* - x_0)$ entonces

$$\begin{aligned} x^* - w &= x^* - x_0 - \sum_{j=0}^{k-1} \gamma_j A^{j+1} (x^* - x_0) \\ &= p(A)(x^* - x_0) \end{aligned}$$

donde $p(z) = 1 - \sum_{j=0}^{k-1} \gamma_j z^{j+1}$, es de grado k y satisface $p(0) = 1$.
De aquí se sigue

$$\|x^* - x_k\|_A = \min_{p \in P_k,\, p(0)=1} \|p(A)(x^* - x_0)\|_A .$$

El teorema espectral para matrices spd afirma que $A = U\Lambda U^T$ donde U es una matriz ortogonal de columnas los vectores propios de A y Λ es una matriz diagonal con diagonal los valores propios de A. Por la ortoganalidad de U se cumple

$$A^j = U\Lambda^j U^T$$

de aquí que $p(A) = Up(\Lambda)U^T$. Para $A^{1/2} = U\Lambda^{1/2}U^T$ notamos que $\|x\|_A^2 = x^T Ax = \left\|A^{1/2}x\right\|_2^2$, así para cualquier $x \in \mathbb{R}^n$

$$\|p(A)x\|_A = \left\|A^{1/2}p(A)x\right\|_2 \leq \|p(A)\|_2 \left\|A^{1/2}x\right\|_2 = \|p(A)\|_2 \|x\|_A$$

y reemplazando

$$\|x^* - x_k\|_A \leq \|x^* - x_0\|_A \min_{p \in P_k,\, p(0)=1} \max_{z \in \sigma(A)} |p(z)| .$$

Este resultado nos permite tener una cota sobre las aproximaciones, a su vez nos permite observar que el método de CG puede ser visto como un método directo.

Corolario 1.4.2 *Sea A spd y sea $\{x_k\}$ las iteraciones de CG. Sea k dada y $\overline{p}_k$ cualquier polinomio de grado k tal que $\overline{p}_k(0) = 1$ entonces*

$$\frac{\|x_k - x^*\|_A}{\|x_0 - x^*\|_A} \leq \max_{z \in \sigma(A)} |\overline{p}_k(z)| . \tag{1.3}$$

Aquí $\overline{p}_k$ es un polinomio residual de orden k.

Definición 1.4.1 *El conjunto de polinomios residuales de grado k es*

$$\mathbb{P}_k = \{p/\ p \text{ es polinomio de grado } k \text{ y } p(0) = 1\}.$$

Teorema 1.4.3 *Sea A una matriz spd entonces el algoritmo CG encuentra la solución en a lo más n iteraciones.*

Prueba. Sea $\{\lambda_i\}_{i=1}^n$ los valores propios de A. Tomemos el polinomio

$$\overline{p}(z) = \prod_{i=1}^{n} \frac{(\lambda_i - z)}{\lambda_i}$$

$\overline{p} \in \mathbb{P}_n$ y

$$\|x_n - x^*\|_A \leq \|x_0 - x^*\|_A \max_{z \in \sigma(A)} |\overline{p}(z)| = 0.$$

■

En particular si el número de valores propios distintos es menor que el orden de la matriz la convergencia de CG es más rápida.

Teorema 1.4.4 *Sea A una matriz spd, supongamos que existen exactamente $k \leq n$ valores propios distintos de A. Entonces la iteración CG termina en a lo más k iteraciones.*

Prueba. La demostración es similar a la anterior. Sean $\{\lambda_i\}_{i=1}^k$ los valores propios distintos de A, basta tomar el polinomio

$$\overline{p}(z) = \prod_{i=1}^{k} \left(1 - \frac{z}{\lambda_i}\right)$$

donde $\overline{p} \in \mathbb{P}_k$ y por corolario la iteración CG termina en a lo más k iteraciones. ■

En general n es grande y los valores propios diferentes también lo son, por tanto es mejor considerar CG como un método iterativo.

1.4.3 Terminación de la iteración

Un criterio típico es terminar cuando los residuales son relativamente pequeños. Las iteraciones terminan cuando

$$\|b - Ax_k\|_2 = \|r_k\|_2 \leq tol * \|b\|_2$$

para una tolerancia *tol* dada.

Para el método CG, dados los valores propios mayor y menor de A, podemos obtener una cota de error si consideramos el polinomio minimal sobre el intervalo $[\lambda_{\min}, \lambda_{\max}]$ y obtener el siguiente resultado:

Teorema 1.4.5 *Sea e_k el error en el paso k_ésimo del algoritmo CG aplicado al sistema $Ax = b$, A matriz spd. Entonces*

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2\left[\left(\frac{\sqrt{k(A)}-1}{\sqrt{k(A)}+1}\right)^k + \left(\frac{\sqrt{k(A)}+1}{\sqrt{k(A)}-1}\right)^k\right]^{-1} \leq 2\left(\frac{\sqrt{k(A)}-1}{\sqrt{k(A)}+1}\right)^k, \qquad (1.4)$$

donde $k(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$ número de condición de la matriz A.

La demostración la podemos encontrar en [1].

Ejemplo 1.4.1 *Supongamos A spd, $x_0 = 0$ y que $\sigma(A) \subset (1, 1.1) \cup (2, 2.2)$. Obtengamos un estimado para el número de iteraciones que CG requiere para reducir la A norma del error por un factor 10^{-3}.*

La mejor estimación para el número de condición de A es $k(A) \leq 2.2$, reemplazando en (1.4) :

$$\frac{\|x^* - x_k\|_A}{\|x^*\|_A} \leq 2 \times \left(\frac{\sqrt{2.2}-1}{\sqrt{2.2}+1}\right)^k \approx 2 \times (0.194\,6)^k$$

queremos que

$$\frac{\|x^* - x_k\|_A}{\|x^*\|_A} \leq 10^{-3},$$

se cumple cuando

$$2 \times (0.194\,6)^k < 10^{-3}$$

es decir $k > -\log_{10}(2000)/\log_{10}(0.194\,6) \simeq 4.643\,73$, por lo tanto el número de iteraciones será 5.

Si usamos como polinomio residual $\overline{p}_{2k}(z) \in \mathbb{P}_{2k}$

$$\overline{p}_{2k}(z) = \frac{(1.05 - z)^k * (2.1 - z)^k}{1.05^k * 2.1^k}$$

es fácil ver que

$$\max_{z\in\sigma(A)} |\overline{p}_{2k}(z)| \leq \left(\frac{1.15*0.1}{1.05*2.1}\right)^k$$

utilizando (1.3) *queremos que*

$$\frac{\|x^* - x_k\|_A}{\|x^*\|_A} \leq 10^{-3}$$

converge en $2k$ *iteraciones cuando*

$$\left(\frac{1.15*0.1}{1.05*2.1}\right)^k \leq 10^{-3}$$

es decir $k > -3/\log_{10}\left(\frac{1.15*0.1}{1.05*2.1}\right) \simeq 2.338$*, por lo tanto el número de iteraciones será 6 (el menor múltiplo entero de 2 ya que 2*2.338=4.676).*

Si usamos como polinomio residual $\overline{p}_{3k}(z) \in \mathbb{P}_{3k}$

$$\overline{p}_{3k}(z) = \frac{(1.05-z)^k * (2.2-z)^{2k}}{1.05^k * 2.2^{2k}}$$

es fácil ver que

$$\max_{z\in\sigma(A)} |\overline{p}_{3k}(z)| \leq \left(\frac{0.05*1.2^2}{1.05^k * 2.2^2}\right)^k$$

utilizando (1.3) *queremos que*

$$\frac{\|x^* - x_k\|_A}{\|x^*\|_A} \leq 10^{-3}$$

converge en $3k$ *iteraciones cuando*

$$\left(\frac{0.05*1.2^2}{1.05^k * 2.2^2}\right)^k \leq 10^{-3}$$

es decir $k > -3/\log_{10}\left(\frac{0.05*1.2^2}{1.05^k*2.2^2}\right) \simeq 1.623$*, por lo tanto el número de iteraciones será 6.*

Del resultado anterior y los ejemplos en [2] se puede concluir que si el número de condición de A es cercano a 1 el método de CG converge más rápidamente, pero si el número de condición de A es grande la convergencia será más lenta. La transformación del problema en uno con valores propios más cercanos a 1 se logra por medio de un **precondicionamiento** del problema inicial.

1.4.4 Implementación

La implementación de CG esta basada en el hecho de que una vez obtenido x_k buscamos la dirección $p_{k+1} \neq 0$ que satisfaga $x_{k+1} = x_k + \alpha_{k+1} p_{k+1}$ para un escalar α_{k+1}. Una vez se tiene p_{k+1}, α_{k+1} es fácil de computar de la propiedad de minimización de la iteración. En efecto

$$\left.\frac{d\Phi\left(x_k + \alpha p_{k+1}\right)}{d\alpha}\right|_{\alpha=\alpha_{k+1}} = 0$$

de (1.2) tenemos

$$\begin{aligned}
\Phi\left(x_k + \alpha p_{k+1}\right) &= \frac{1}{2}\left(x_k + \alpha p_{k+1}\right)^T A\left(x_k + \alpha p_{k+1}\right) - \left(x_k + \alpha p_{k+1}\right)^T b \\
&= \frac{1}{2}\left(x_k^T A x_k + \alpha p_{k+1}^T A x_k + \alpha x_k^T A p_{k+1} + \alpha^2 p_{k+1}^T A p_{k+1}\right) \\
&\quad - x_k^T b - \alpha p_{k+1}^T b
\end{aligned}$$

como $x^T A \alpha p = \alpha\left(Ax\right)^T p = \alpha p^T A x$

$$\Phi\left(x_k + \alpha p_{k+1}\right) = \frac{1}{2} x_k^T A x_k + \alpha p_{k+1}^T A x_k + \frac{1}{2}\alpha^2 p_{k+1}^T A p_{k+1} - x_k^T b - \alpha p_{k+1}^T b$$

por lo tanto

$$\left.\frac{d\Phi\left(x_k + \alpha p_{k+1}\right)}{d\alpha}\right|_{\alpha=\alpha_{k+1}} = p_{k+1}^T A x_k + \alpha_{k+1} p_{k+1}^T A p_{k+1} - p_{k+1}^T b = 0$$

despejando α_{k+1}

$$\alpha_{k+1} = \frac{p_{k+1}^T\left(b - A x_k\right)}{p_{k+1}^T A p_{k+1}} = \frac{p_{k+1}^T r_k}{p_{k+1}^T A p_{k+1}}$$

si $x_k = x_{k+1}$ entonces $\alpha_{k+1} = 0$, esto solo se cumple si $x_k = x^*$.

Primero notaremos que si las $\{x_k\}$ son iteraciones de CG, se cumple que $r_l \in K_k$ para todo $l < k$. Sea $l \in \mathbb{N}$

$$r_l = b - A x_l = b - A\left(x_0 + \sum_{j=0}^{l-1} \gamma_j A^j r_0\right) = r_0 - \gamma_0 A r_0 - ... - \gamma_{l-1} A^l r_0$$

para que $r_l \in K_k$ dado que $K_k = \left\{r_0, A r_0, ..., A^{k-1} r_0\right\}$, se debe cumplir que $l \leq k - 1$.

En el proceso iterativo de CG vamos construyendo una sucesión de vectores residuales ortogonales.

Lema 1.4.6 *Sea A una matriz spd y $\{x_k\}$ iteraciones de CG entonces*

$$r_k^T \; r_l = 0 \qquad \textit{para todo } 0 \leq l < k.$$

Prueba. Como x_k minimiza Φ sobre $x_0 + K_k$, para $\xi \in K_k$

$$\frac{d\Phi\left(x_k + t\xi\right)}{dt} = \nabla\Phi\left(x_k + t\xi\right)^T \xi = 0$$

en $t = 0$, luego

$$0 = \nabla\Phi\left(x_k\right)^T \xi = \left(Ax_k - b\right)^T \xi = -r_k^T \; \xi$$

para todo $\xi \in K_k$. En particular para $r_l \in K_k$ con $l < k$. ■

Si $x_k = x_{k+1}$, $r_k = r_{k+1}$ el lema anterior implica

$$\|r_k\|_2^2 = r_k^T r_k = r_k^T r_{k+1} = 0$$

es decir $Ax_k = b$.

Lema 1.4.7 *Sea A spd y $\{x_k\}$ iteraciones de CG. Si $x_k \neq x^*$ entonces $x_{k+1} = x_k + \alpha_{k+1} p_{k+1}$ y p_{k+1} queda determinado, salvo por múltiplo por escalar, por las condiciones*

$$p_{k+1} \in K_{k+1}, \quad p_{k+1}^T A\xi = 0 \quad \textit{para todo } \xi \in K_k.$$

La demostración de este lema la encontramos en [2]. Ahora veremos como construir las direcciones.

Teorema 1.4.8 *Sea A matriz spd y supongamos que $r_k \neq 0$. Definimos $p_0 = 0$, entonces*

$$p_{k+1} = r_k + \beta_{k+1} p_k$$

para algún β_{k+1} y $k \geq 0$.

Prueba. Por lema anterior y el hecho que $K_k = gen\{r_0, r_1, ..., r_{k-1}\}$, solo necesitamos verificar que β_{k+1} puede ser encontrado. Si p_{k+1} esta dado por $p_{k+1} = r_k + \beta_{k+1} p_k$ y $p_{k+1}^T A r_l = 0$ para $0 \leq l \leq k-1$

$$r_k^T A r_l + \beta_{k+1} p_k^T A r_l = 0 \qquad l \leq k-1$$

si $l \leq k-2$, entonces $r_l \in K_{l+1} \subset K_{k-1}$. Por lema anterior se cumple

$$p_{k+1}^T A r_l = 0 \quad 0 \leq l \leq k-2$$

solo resta resolver para β_{k+1}, tomando $l = k-1$

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

es necesario que $p_k^T A r_{k-1} \neq 0$. Como $r_k = r_{k-1} - \alpha_k A p_k$ tenemos

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

como $r_k^T r_{k-1} = 0$ por lema,

$$p_k^T A r_{k-1} = \frac{\|r_{k-1}\|_2^2}{\alpha_k} \neq 0.$$

■

Para realizar un algoritmo menos costoso la implementación común de CG utiliza otras expresiones para α_k y β_k.

Lema 1.4.9 *Sea A una matriz spd, si $r_k \neq 0$ entonces*

$$\alpha_{k+1} = \frac{\|r_k\|_2^2}{p_{k+1}^T A \; p_{k+1}}$$

y

$$\beta_{k+1} = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2}$$

Prueba. Por lema anterior para $k \geq 0$

$$p_{k+1}^T r_{k+1} = r_k^T r_{k+1} + \beta_{k+1} p_k^T r_{k+1} = 0$$

similarmente se cumple que $p_k^T r_k = 0$ y así

$$p_{k+1}^T r_k = r_k^T r_k + \beta_{k+1} p_k^T r_k = \|r_k\|_2^2 .$$

Tomando producto escalar a ambos lados de $r_{k+1} = r_k - \alpha_{k+1} A p_{k+1}$ con p_{k+1}

$$0 = p_{k+1}^T r_k - \alpha_{k+1} p_{k+1}^T A p_{k+1} = \|r_k\|_2^2 - \alpha_{k+1} p_{k+1}^T A p_{k+1},$$

lo que es equivalente a

$$\alpha_{k+1} = \frac{\|r_k\|_2^2}{p_{k+1}^T A\ p_{k+1}}.$$

Para β_{k+1} por la propiedad de A-conjugación tenemos que $p_{k+1}^T A p_k = 0$

$$0 = p_{k+1}^T A p_k = \left(r_k + \beta_{k+1} p_k\right)^T A p_k = r_k^T A p_k + \beta_{k+1} p_k^T A p_k$$

lo cual implica que

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

También tenemos

$$\begin{aligned} p_k^T A p_k &= p_k^T A \left(r_{k-1} + \beta_k p_{k-1}\right) \\ &= p_k^T A r_{k-1} + \beta_k p_k^T A p_{k-1} \\ &= p_k^T A r_{k-1} \end{aligned}$$

combinando los resultados anteriores y $p_k^T A r_{k-1} = \frac{\|r_{k-1}\|_2^2}{\alpha_k} \neq 0$

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

tomando producto escalar a ambos lados de $r_k = r_{k-1} - \alpha_k A p_k$ con r_k y usando la ortogonalidad de los vectores residuales

$$\|r_k\|_2^2 = -r_k^T A p_k \alpha_k$$

y por tanto

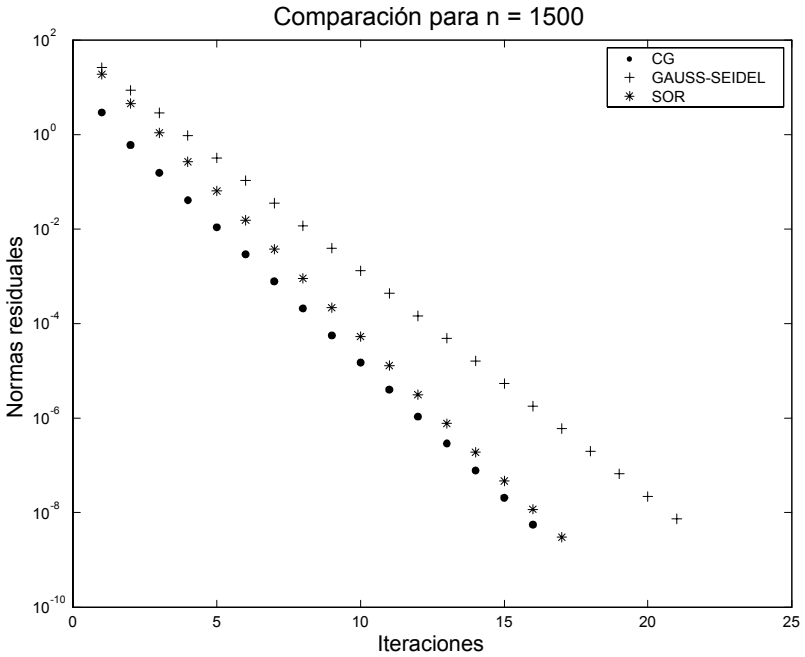
$$\beta_{k+1} = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2}.$$

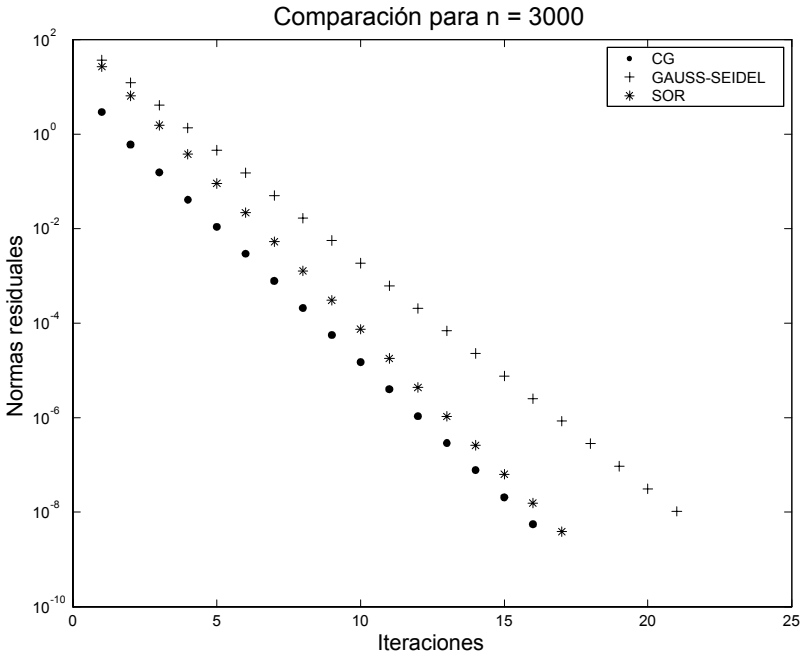
■

Ejemplo 1.4.2 *Comparamos los resultados para los métodos G-S, SOR con* $w = 1.1$ *y CG para matrices de la forma* $A = trid\,(-1, 4, -1)$, *con tol* $= 10^{-10}$, *partimos del vector de ceros y obtenemos:*

Método	*Orden A*	*Iteración*	*Norma Residual*
CG	*1500*	*16*	*5.5544e-09*
G-S	*1500*	*21*	*7.3654e-09*
SOR	*1500*	*17*	*3.0008e-09*
CG	*3000*	*16*	*5.5577e-09*
G-S	*3000*	*21*	*1.0444e-08*
SOR	*3000*	*17*	*3.8949e-09*

y las gráficas de las normas residuales vs. iteraciones son:





Podemos observar que el método de CG converge más rápido que los métodos estacionarios y el comportamiento de los residuales nos permiten observar que las aproximaciones son mejores que las de los métodos estacionarios.

Ahora vemos la ventaja de trabajar con la acción de la matriz sobre un vector y no almacenar la matriz, comparamos rutinas propias (sin almacenar) vs. rutinas internas de matlab versión 6.5 (almacenando)

	Orden A	*Iteración*	*tiempo (seg.)*
CGP	*5000*	*16*	*0.0620000*
PCG(matlab_sparse)	*5000*	*16*	*0.7190000*
PCG(matlab)	*5000*	*16*	*11.391000*

PCG(matlab_sparse) se refiere al caso en que no almacenamos la matriz pero trabajamos con la instrucción sparse(A), es decir, utilizando el almacenamiento especial de matlab para matrices ralas.

Y por último comparamos rutinas propias vs. rutinas internas matlab versión 6.5 ambas

con la acción de la matriz sobre vector

	Orden A	*Iteración*	*tiempo (seg.)*
CGP	*60000*	*15*	*0.875000*
PCG(matlab)	*60000*	*15*	*1.297000*
CGP	*600000*	*14*	*7.812000*
PCG(matlab)	*600000*	*14*	*16.23500*

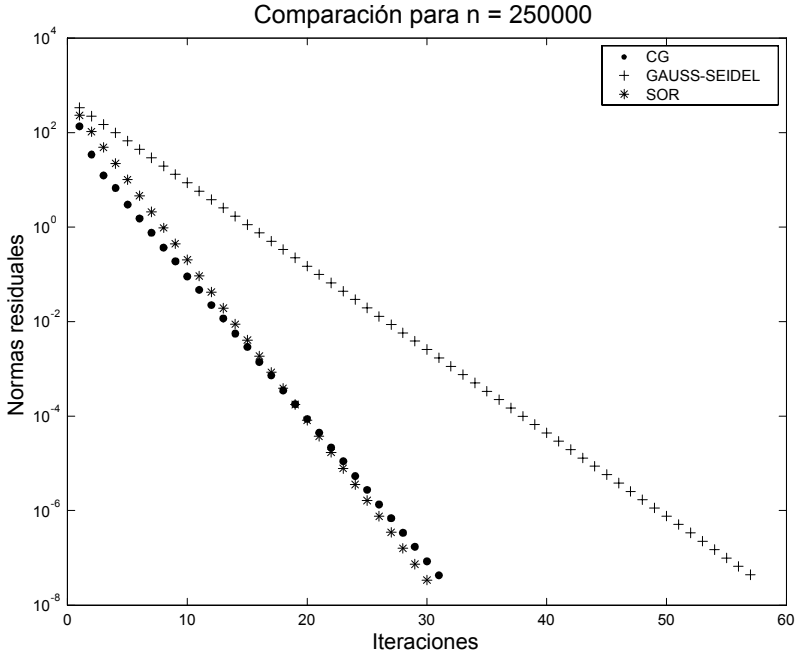
Ejemplo 1.4.3 *Comparamos los resultados para los métodos G-S, SOR con* $w = 13$ *y CG para matrices tridiagonales por bloques de la forma*

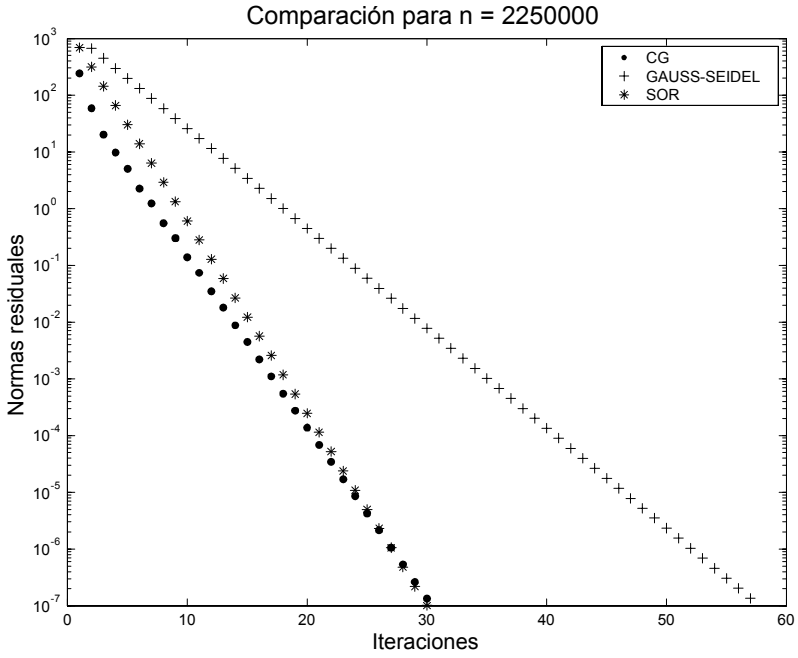
$$A = \begin{bmatrix} D & -I & & & \\ -I & D & -I & & \\ & -I & \ddots & \ddots & \\ & & \ddots & D & -I \\ & & & -I & D \end{bmatrix}$$

donde D=trid(-1,5,-1), con tol $= 10^{-10}$*, partimos del vector de ceros y obtenemos:*

Método	*Orden A*	*Iteración*	*Norma Residual*
CG	*250000*	*31*	*4.29830e-008*
G-S	*250000*	*57*	*4.43533e-008*
SOR	*250000*	*30*	*3.41219e-008*
CG	*2250000*	*30*	*1.34233e-007*
G-S	*2250000*	*57*	*1.36143e-007*
SOR	*2250000*	*30*	*1.01606e-007*

y las gráficas de las normas residuales vs. iteraciones son:





1.4.5 Gradiente Conjugado y ecuaciones normales (CGNE y CGNR):

Estos métodos se basan en la aplicación del método CG a una de las ecuaciones normales para $Ax = b$. CGNE resuelve el sistema $\left(AA^T\right) y = b$ para y y entonces computamos la solución $x = A^T y$. CGNR resuelve $\left(A^T A\right) x = \widetilde{b}$ para x, donde $\widetilde{b} = A^T b$. Como A es no singular, tanto $A^T A$ como AA^T son spd y por tanto CG se puede aplicar.
La razón para el nombre CGNR es que la propiedad de minimización de CG aplicada a $A^T Ax = A^T b$ lleva a minimizar

$$\begin{aligned} \|x^* - x\|^2_{A^T A} &= (x^* - x)^T A^T A (x^* - x) \\ &= (Ax^* - Ax)^T (Ax^* - Ax) \\ &= (b - Ax)^T (b - Ax) \\ &= \|b - Ax\|_2^2 = \|r\|_2^2 \end{aligned}$$

sobre $x_0 + K_k$ en cada iteración, por tanto el CG de la ecuación normal minimiza el residual. Similarmente, en CGNE se minimiza el error, pues

$$\begin{aligned} \|y^* - y\|^2_{AA^T} &= (y^* - y)^T AA^T (y^* - y) \\ &= \left(A^T y^* - A^T y\right)^T \left(A^T y^* - A^T y\right) \\ &= \|x^* - x\|_2^2 . \end{aligned}$$

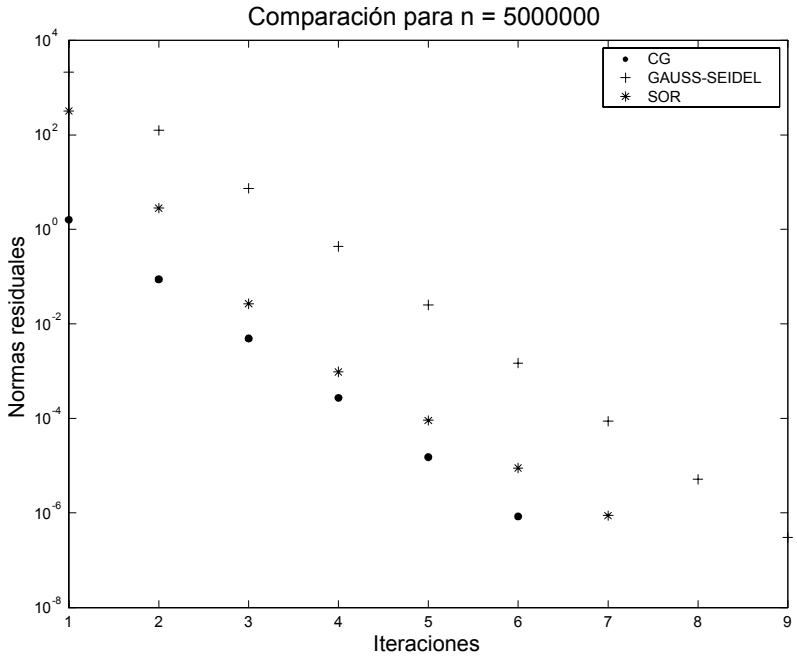
Ejemplo 1.4.4 *Para matrices de la forma $A = trid\,(-1, 4, 1)$ que no son simétricas usamos CGNR donde*

$$A^T A = \begin{bmatrix} 17 & 0 & -1 & & & \\ 0 & 18 & 0 & -1 & & \\ -1 & 0 & 18 & 0 & \ddots & \\ & -1 & 0 & \ddots & \ddots & -1 \\ & & \ddots & \ddots & 18 & 0 \\ & & & -1 & 0 & 17 \end{bmatrix},$$

comparamos los resultados para G-S, SOR con $w = 1.05$ y CG con $tol = 10^{-10}$, partimos del

vector de ceros y para A de orden 5000000 obtenemos:

Método	*Iteración*	*Norma Residual*
CG	*6*	*8.44577e-07*
G-S	*9*	*3.01689e-07*
SOR	*7*	*8.77578e-07*



1.5 Precondicionamiento

Consideramos el sistema $n \times n$ lineal simétrico definido positivo $Ax = b$. La idea del precondicionamiento en el Gradiente Conjugado es aplicar el método de CG al sistema transformado

$$\widetilde{A}\widetilde{x} = \widetilde{b}$$

donde $\widetilde{A} = L^{-1}AL^{-1}$, $\widetilde{x} = Lx$, $\widetilde{b} = L^{-1}b$, donde L es una **matriz simétrica definida positiva**. La escogencia de L será tal que $\widetilde{A}$ sea bien condicionada o una matriz con valores propios pequeños. Si aplicamos CG al sistema transformado tenemos

Algoritmo CG sobre $\widetilde{A}\widetilde{x} = \widetilde{b}$

Lea $\widetilde{A}$, $\widetilde{x}_0$, $\widetilde{b}$, tol, max_it

$\widetilde{r}_0 = \widetilde{b} - \widetilde{A}\widetilde{x}_0$

$ep = tol * \left\|\widetilde{b}\right\|_2$

$para\ k = 1, 2, ..., max\_it$

$\rho_{k-1} = \widetilde{r}_{k-1}^T \widetilde{r}_{k-1}$

$si\ k = 1$

$\widetilde{p}_1 = \widetilde{r}_0$

$en\ otro\ caso$

$\beta_k = \frac{\rho_{k-1}}{\rho_{k-2}}$

$\widetilde{p}_k = \widetilde{r}_{k-1} + \beta_k \widetilde{p}_{k-1}$

fin

$\alpha_k = \rho_{k-1} / \ \widetilde{p}_k^T \widetilde{A} \widetilde{p}_k$

$\widetilde{x}_k = \widetilde{x}_{k-1} + \alpha_k \widetilde{p}_k$

$\widetilde{r}_k = \widetilde{r}_{k-1} - \alpha_k \widetilde{A} \widetilde{p}_k$

$si\ \|\widetilde{r}_k\|_2 \leq ep\ entonces\ termine$

fin

fin

Aquí, $\widetilde{x}_k$ será una aproximación a $\widetilde{x}$ y $\widetilde{r}_k$ es el residual del sistema transformado $\widetilde{b} - \widetilde{A}\widetilde{x}_k$. Por supuesto si tenemos $\widetilde{x}$ podemos obtener x como $x = L^{-1}\widetilde{x}$, para ello deberíamos tener una expresión para L^{-1}. Definamos

$$\begin{aligned} \widetilde{p}_k &= L\ p_k \\ \widetilde{x}_k &= L\ x_k \\ \widetilde{r}_k &= L^{-1}\ r_k. \end{aligned}$$

Como $\widetilde{b} = L^{-1}b$ y $\widetilde{x} = Lx$ del algoritmo anterior obtenemos:

Algoritmo

$Lea\ A, x_0, b, tol, max\_it$

$r_0 = b - Ax_0$

$ep = tol * \left\|\widetilde{b}\right\|_2$

para $k = 1, 2, ..., max\_it$

$\rho_{k-1} = \left(L^{-1} r_{k-1}\right)^T \left(L^{-1} r_{k-1}\right)$

si $k = 1$

$Lp_1 = L^{-1} r_0$

en otro caso

$\beta_k = \frac{\rho_{k-1}}{\rho_{k-2}}$

$L\ p_k = L^{-1} r_{k-1} + \beta_k\ L\ p_{k-1}$

fin

$\alpha_k = \rho_{k-1} / \left(L\ p_k\right)^T\ L^{-1} A L^{-1}\ \left(L\ p_k\right)$

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

$L^{-1} r_k = L^{-1} r_{k-1} - \alpha_k\ L^{-1} A L^{-1} \left(L\ p_k\right)$

si $\|r_k\|_2 \leq ep$ *entonces termine*

fin

fin

Si definimos el precondicionamiento M por $M = L^2$ (también definida positiva) y hacemos z_k la solución del sistema $M\ z_k = r_k$, entonces

$$p_1 = L^{-1} L^{-1} r_0 = M^{-1} r_0 = z_0.$$

Para ρ_{k-1} tendremos

$$\begin{aligned} \rho_{k-1} &= \left(L^{-1} r_{k-1}\right)^T \left(L^{-1} r_{k-1}\right) \\ &= r_{k-1}^T L^{-T} L^{-1} r_{k-1} \\ &= r_{k-1}^T L^{-1} L^{-1} r_{k-1} \\ &= r_{k-1}^T M^{-1} r_{k-1} \\ &= r_{k-1}^T z_{k-1}. \end{aligned}$$

Además, la dirección p_k será

$$\begin{aligned} p_k &= L^{-1}L^{-1}r_{k-1} + L^{-1}\beta_k L p_{k-1} \\ &= M^{-1}r_{k-1} + \beta_k L^{-1}L p_{k-1} \\ &= z_{k-1} + \beta_k\, p_{k-1} \end{aligned}$$

y el escalar α_k quedará

$$\begin{aligned} \alpha_k &= \rho_{k-1} \,/\, (L\, p_k)^T\, L^{-1}AL^{-1}\,(L\, p_k) \\ &= \rho_{k-1} \,/\, p_k^T L^T L^{-1}AL^{-1}L\, p_k \\ &= \rho_{k-1} \,/\, p_k^T A\, p_k \\ &= \rho_{k-1} \,/\, p_k^T A\, p_k. \end{aligned}$$

Luego la iteración x_k estará dada por

$$x_k = L^{-1}L\, x_{k-1} + L^{-1}\alpha_k L\, p_k = x_{k-1} + \alpha_k\, p_k$$

y por último los residuales serán

$$r_k = LL^{-1}r_{k-1} - L\alpha_k L^{-1}AL^{-1}\,(L\, p_k) = r_{k-1} - \alpha_k A\, p_k.$$

Con estas nuevas variables, se obtiene el algoritmo de CG precondicionado siguiente: Dadas la matriz A, precondicionamiento M, vector de términos independientes b,

Algoritmo PCG

Lea A, x_0, b, tol, max_it

$r_0 = b - Ax_0$

$ep = tol * \left\|\widetilde{b}\right\|_2$

para $k = 1, 2, ..., max\_it$

resuelva $Mz_{k-1} = r_{k-1}$

$\rho_{k-1} = r_{k-1}^T z_{k-1}$

si $k = 1$

$p_1 = z_0$

en otro caso

$\beta_k = \frac{\rho_{k-1}}{\rho_{k-2}}$

$p_k = z_{k-1} + \beta_k \; p_{k-1}$

fin

$q_k = A * p_k$

$\alpha_k = \rho_{k-1} / \; p_k^T \; q_k$

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

$r_k = r_{k-1} - \alpha_k \; q_k$

si $\|r_k\|_2 \leq ep$ *entonces termine*

fin

fin

El método de CG se puede generalizar para operadores lineales en dimensión infinita. Para un operador lineal acotado T de X sobre Y, X, Y espacios de Hilbert, ambos sobre el campo real o complejo, la ecuación lineal $Tx = y$, para $y \in Y$, puede tener o no solución dependiendo si y está o no en $R(T)$, rango de T. Además si $y \in R(T)$ la solución no necesariamente es única. Lo que buscamos es, en cada paso, encontrar la mejor solución en el sentido de mínimos cuadrados, es decir, una solución la cual minimiza el funcional cuadrático $f(x) = \|Tx - y\|_2$. La implementación para el método de CG que minimiza $f(x)$ en cada paso es: partimos de un valor inicial $x_0 \in X$, computamos $r_0 = p_0 = T^*(Tx_0 - y)$ donde T^* es el adjunto de T. Si $p_0 \neq 0$ encontramos $x_1 = x_0 - \alpha_0 p_0$ donde $\alpha_0 = \frac{\|r_0\|^2}{\|Tp_0\|^2}$ para $i = 1, 2, ...,$

$$r_i = T^*(Tx_i - y) = r_{i-1} - \alpha_{i-1} T^* T p_{i-1}$$

donde

$$\alpha_{i-1} = \frac{\langle r_{i-1}, p_{i-1} \rangle}{\|Tp_{i-1}\|^2}.$$

Si $\alpha_{i-1} \neq 0$ la dirección será $p_i = r_i + \beta_{i-1} p_{i-1}$ donde

$$\beta_{i-1} = -\frac{\langle r_i, T^* T p_{i-1} \rangle}{\|Tp_{i-1}\|^2}$$

y la aproximación está dada por $x_{i+1} = x_i - \alpha_i p_i$. La convergencia de la iteración está dada si T es un operador lineal acotado que mapea X sobre Y, con inversa acotada. A través del tiempo se han realizado más estudios para reducir las condiciones sobre el operador y poder seguir garantizando convergencia del método, ver por ejemplo [3].

Capítulo 2

GMRES Y MINRES

En este capítulo discutimos el método GMRES para resolver sistemas $Ax = b$ donde A es una matriz no singular, posiblemente no simétrica y analizamos brevemente el método MINRES para resolver sistemas $Ax = b$, donde A es matriz simétrica. Este último se diferencia del método de CG en que la iteración minimiza la norma del residual.

Este capítulo está organizado como sigue. En la sección 2.1 GMRES, se realiza una descripción detallada del método GMRES y se estudia una pequeña variación del método que permite economizar grandes espacios de memoria. En la sección 2.2 se describe el precondicionamiento del método de GMRES para obtener mejores tiempos de ejecución. En la sección 2.3 MINRES, se estudia la idea básica del método MINRES y una pequeña descripción de su implementación y terminamos con la sección 2.4 dimensión infinita, en la cual damos una breve generalización del método GMRES para espacios de dimensión infinita.

2.1 GMRES

El método GMRES (Generalized Minimum Residual) es aplicable a sistemas no simétricos. Para GMRES la iteración x_k minimiza $\|b - Ax\|_2$ sobre el conjunto $x_0 + K_k$ donde $K_k = gen\left\{r_0, Ar_0, ..., A^{k-1}r_0\right\}$. La idea fundamental es expresar x_k en términos de un conjunto $\{v_1, ..., v_k\}$ generador de K_k, en el cual v_1 es un múltiplo escalar del vector residual $r_0 = b - Ax_0$. En el método de gradiente conjugado se forma una base ortogonal para el espacio K_k, en GMRES esta base está formada explícitamente por el proceso de ortogonalización de Gram - Schmidt Modificado. La teoría que aquí se trabaja está basada en los libros de Kelley [2] y Saad [19].

2.1.1 Propiedad de minimización y sus consecuencias

GMRES fue propuesto por Saad y Schultz [18] en 1986 como un método de subespacios de Krylov para sistemas no simétricos. A diferencia de CGNR, GMRES no requiere computar la acción de A^T sobre un vector. Encontrar estimados de error para GMRES es mucho más difícil que para CG pues la matriz A, fuera de no ser simétrica, a menudo ni siquiera es diagonalizable (ver [2], [18]).

Comenzamos con una descripción de la propiedad de la minimización de los errores en las iteraciones y sus consecuencias, describiremos un criterio de terminación, la ejecución y la implementación acompañados de unos ejemplos ilustrativos.

Recordamos que nuestro objetivo es minimizar el error en la $k\_ésima$ $(k \geq 1)$ iteración sobre el espacio afín $x_0 + K_k$. La $k\_ésima$ $(k \geq 1)$ iteración de GMRES la denotamos x_k y es solución del problema de mínimos cuadrados $\min\limits_{x \in x_0 + K_k} \|b - Ax\|_2$.

El comienzo es similar al análisis realizado para CG, notemos que si $x \in x_0 + K_k$ entonces

$$x = x_0 + \sum_{j=0}^{k-1} \gamma_j A^j r_0$$

y así

$$b - Ax = b - Ax_0 - \sum_{j=0}^{k-1} \gamma_j A^{j+1} r_0 = r_0 - \sum_{j=1}^{k} \gamma_{j-1} A^j r_0,$$

de aquí que si $x \in x_0 + K_k$ entonces $r = \overline{p}(A) r_0$ donde $\overline{p} \in \mathbb{P}_k$, $\mathbb{P}_k$ el conjunto de polinomios residuales de grado k que se caracterizan por tener término constante igual a 1 (ver sección 1.4.2 de este trabajo). Hemos probado el siguiente resultado:

Teorema 2.1.1 *Sea A no singular y sea x_k la $k\_ésima$ iteración con GMRES. Entonces para todo $\overline{p_k} \in \mathbb{P}_k$*

$$\|r_k\|_2 = \min_{\overline{p} \in \mathbb{P}_k} \|\overline{p}(A) r_0\|_2 \leq \|\overline{p_k}(A) r_0\|_2 . \tag{2.1}$$

De aquí se sigue el corolario

Corolario 2.1.2 *Sea A no singular y sea x_k la $k\_ésima$ iteración con GMRES. Entonces para todo $\overline{p_k} \in \mathbb{P}_k$*

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \|\overline{p_k}(A)\|_2 . \tag{2.2}$$

De este corolario se puede ver que el método GMRES termina en un número finito de pasos.

Teorema 2.1.3 *Sea A no singular. Entonces el algoritmo GMRES encuentra la solución en n iteraciones.*

Prueba. Sea $p(z) = \det(A - zI)$ el polinomio característico de A, p tiene grado n, $p(0) = \det(A) \neq 0$ ya que A es no singular. Por lo tanto

$$\overline{p_n}(z) = \frac{p(z)}{p(0)} \in \mathbb{P}_n$$

es un polinomio residual. Por teorema de Cayley Hamilton $p(A) = \det(A)\ \overline{p_n}(A) = 0$. Por (2.2) $r_n = b - Ax_n = 0$ y de aquí que x_n es la solución. ■

Para el método de CG aplicamos el teorema espectral para obtener información de la rapidez de convergencia. Esta es una opción única para matrices simétricas. Sin embargo, si A es diagonalizable podemos usar (2.1) para saber sobre el espectro de A como en CG. Recordemos que si A es diagonalizable existe V matriz no singular (posiblemente compleja) tal que $A = V\Lambda V^{-1}$ donde Λ matriz diagonal con los valores propios de A en su diagonal. Si A es matriz diagonalizable y p es un polinomio entonces

$$p(A) = Vp(\Lambda)V^{-1}.$$

La matriz A es normal si la matriz cambio de base V es ortogonal. En este caso las columnas de V son una base ortogonal de vectores propios de A y $V^{-1} = V^H$. Para matrices diagonalizables tenemos el siguiente resultado:

Teorema 2.1.4 *Sea $A = V\Lambda V^{-1}$ no singular diagonalizable, x_k la k_ésima iteración con GMRES. Entonces para todo $\overline{p_k} \in \mathbb{P}_k$*

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq k_2(V) \max_{z \in \sigma(A)} |\overline{p_k}(z)|. \tag{2.3}$$

La prueba de este teorema la encontramos en [2]. Pero no es claro cómo estimar el número de condición de V, excepto en casos sencillos, como por ejemplo, A normal, en cuyo caso $k_2(V) = 1$. Análogamente a CG, se dan los siguientes resultados.

2.1.2 Terminación de la iteración

Como en el caso de CG, GMRES es considerado como un método iterativo. La rapidez de convergencia para cuando A es diagonalizable depende de $k_2(V)$. Por ser más sencillo, consideramos solamente matrices diagonalizables. Como para el método CG, un típico criterio es residuales relativos pequeños, las iteraciones terminan cuando

$$\|b - Ax_k\|_2 = \|r_k\| \leq tol * \|b\|_2$$

para una tolerancia *tol* dada, por lo tanto podemos estimar el número de iteraciones necesarias de (2.2) o (2.3) con $x_0 = 0$.

La convergencia para GMRES en el caso no normal es mucho más complicada que las de CG, CGNR, CGNE, o GMRES en el caso normal (ver [18]).

2.1.3 Implementación de GMRES : Ideas básicas

Recordamos que el problema de mínimos cuadrados definido en la $k\_ésima$ iteración GMRES es

$$\min_{x \in x_0 + K_k} \|b - Ax\|_2 . \tag{2.4}$$

Supongamos que tenemos V_k matriz de proyección ortogonal sobre K_k. Entonces cualquier $z \in K_k$ puede escribirse como

$$z = \sum_{l=1}^{k} y_l v_l,$$

donde v_l son las columnas de V_k. Por lo tanto podemos convertir (2.4) al problema de mínimos cuadrados en $\mathbb{R}^k$ para el vector de coeficientes y de $z = x - x_0$. Dado que

$$x - x_0 = V_k y$$

para algún $y \in \mathbb{R}^k$, tenemos que $x_k = x_0 + V_k y$ donde y minimiza

$$\|b - A(x_0 + V_k y)\|_2 = \|r_0 - AV_k y\|_2$$

por lo tanto nuestro problema de mínimos cuadrados en $\mathbb{R}^k$ es

$$\min_{y\in\mathbb{R}^k} \|r_0 - AV_k y\|_2 .$$

Ahora encontraremos una base ortogonal para K_k, para resolver nuestro problema de mínimos cuadrados en $\mathbb{R}^k$, para ello usaremos el proceso de Gram-Schmidt. El proceso de Gram-Schmidt para la formación de una base ortonormal para K_k se llama proceso de Arnoldi. El algoritmo de Arnoldi se construye partiendo de un vector unitario (por simplicidad, tomaremos un vector y lo normalizamos dentro del algoritmo) y de la matriz A, tenemos así

Algoritmo de Arnoldi

Lea A, v_1

$v_1 = \frac{v_1}{\|v_1\|_2}$

para $j = 1, ..., k$

  para $i = 1, 2, ..., j$

    $h_{i,j} = (Av_j)^T v_i$

  fin

$w_j = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$

$h_{j+1,j} = \|w_j\|_2$

si $h_{j+1,j} = 0$ *entonces pare*

fin

$v_{j+1} = w_j / h_{j+1,j}$

fin

Si no existe división por cero en el algoritmo de Arnoldi, entonces las columnas de V son una base ortonormal para K_k. Partiremos con $v_1 = \frac{r_0}{\beta}$ donde $\beta = \|r_0\|_2$. La división por cero aparece si la solución de $Ax = b$ está en K_{k-1}. Antes de probar esta afirmación revisaremos unas relaciones dadas por el algoritmo de Arnoldi.

Proposición 2.1.5 *Denotamos por V_k la matriz $n \times k$ con columnas $v_1, v_2, ..., v_k$, $\widetilde{H}_k$ la matriz $k+1 \times k$ de Hessenberg con entradas no cero h_{ij} definidas por el algoritmo de Arnoldi y H_k la*

matriz obtenida de $\widetilde{H}_k$ borrando la última fila . Entonces se cumplen las siguientes relaciones:

$$AV_k = V_k H_k + w_k e_k^T \tag{2.5}$$

$$AV_k = V_{k+1}\widetilde{H}_k \tag{2.6}$$

$$V_k^T AV_k = H_k \tag{2.7}$$

Prueba. La relación (2.6) se deriva del hecho de que en el algoritmo de Arnoldi $w_j = Av_j - \sum_{i=1}^{j} h_{i,j}v_i$ de donde $Av_j = w_j + \sum_{i=1}^{j} h_{i,j}v_i$, como $v_{j+1} = w_j/h_{j+1,j}$ tenemos $w_j = h_{j+1,j}v_{j+1}$ que al reemplazar en Av_j obtenemos que

$$Av_j = \sum_{i=1}^{j+1} h_{ij}v_i \qquad j = 1, 2, ..., k. \tag{2.8}$$

La relación (2.5) es la forma matricial de (2.8) . La relación (2.7) se sigue de multiplicar ambos lados de (2.5) por V_k^T y usando la ortonormalidad de $\{v_1, v_2, ..., v_k\}$. ■

Podemos así probar la afirmación anterior

Lema 2.1.6 *Sea A no singular, sean los vectores v_i generados por el algoritmo Arnoldi y sea j el menor entero tal que*

$$Av_j - \sum_{i=1}^{j} h_{i,j}v_i = 0$$

entonces $x = A^{-1}b \in x_0 + K_j$.

Prueba. Por hipótesis $Av_j \in K_j$ y por tanto $AK_j \subset K_j$, si $Av_j - \sum_{i=1}^{j} h_{i,j}v_i = 0$ de la relación (2.6) tenemos que $AV_j = V_j H_j$ donde H_j es una matriz $j \times j$ no singular. Tomando $\beta = \|r_0\|_2$ y $e_1 = (1, 0, ..., 0)^T \in \mathbb{R}^j$ tenemos

$$\|r_j\|_2 = \|b - Ax_j\|_2 = \|r_0 - A(x_j - x_0)\|_2$$

como $x_j - x_0 \in K_j$ existe $y \in \mathbb{R}^j$ tal que $x_j - x_0 = V_j y$. Como $r_0 = \beta V_j e_1$ y V_j es una matriz ortogonal

$$\|r_j\|_2 = \|\beta V_j e_1 - AV_j y\| = \|V_j(\beta e_1 - H_j y)\|_2 = \|\beta e_1 - H_j y\|_{\mathbb{R}^{j+1}}$$

donde $\|.\|_{\mathbb{R}^{j+1}}$ denota la norma euclideana en $\mathbb{R}^{j+1}$. Tomando $y = \beta H_j^{-1} e_1$ produce que $r_j = 0$ por la propiedad de minimización. ■

Para derivar el algoritmo de GMRES retomamos la propiedad de minimización y las relaciones obtenidas por el algoritmo de Arnoldi. Cualquier vector x de $x_0 + K_k$ puede escribirse

$$x = x_0 + V_k y$$

donde $y \in \mathbb{R}^k$, luego

$$\|b - Ax\|_2 = \|b - A(x_0 + V_k y)\|_2$$

y por (2.6) en forma similar a la prueba del lema, tenemos que

$$\begin{aligned} b - Ax &= b - A(x_0 + V_k y) = r_0 - AV_k y = \beta v_1 - V_{k+1}\widetilde{H}_k y = \beta V_{k+1} e_1 - V_{k+1}\widetilde{H}_k y \\ b - Ax &= V_{k+1}\left(\beta e_1 - \widetilde{H}_k y\right) \end{aligned}$$

y como las columnas de V_{k+1} son ortonormales

$$\|b - Ax\|_2 = \left\|\beta e_1 - \widetilde{H}_k y\right\|_2. \tag{2.9}$$

La aproximación GMRES es el único vector de $x_0 + K_k$ que minimiza $\|b - Ax\|_2$. Por análisis anterior esta aproximación puede ser obtenida como $x_k = x_0 + V_k y_k$ donde y_k minimiza (2.9), es decir

$$\begin{aligned} x_k &= x_0 + V_k y_k \text{ donde} \\ y_k &= \min_{y \in \mathbb{R}^k} \left\|\beta e_1 - \widetilde{H}_k y\right\|_2. \end{aligned}$$

Para el algoritmo GMRES partimos del valor inicial x_0, el vector lado derecho b, la matriz de coeficientes A y utilizamos el proceso de Gram-Schmidt Modificado para obtener la base de K_k.

Algoritmo GMRES

Lea A, x_0, b

$r_0 = b - Ax_0$

$\beta = \|r_0\|_2$

$v_1 = r_0/\beta$

para $j = 1, ..., k$

$w_j = Av_j$

para $i = 1, ..., j$

$h_{i,j} = w_j^T v_i$

$w_j = w_j - h_{i,j} v_i$

fin

$h_{j+1,j} = \|w_j\|_2$ *si* $h_{j+1,j} = 0$ *salir del ciclo*

$v_{j+1} = w_j / h_{j+1,j}$

fin

Encuentre y_j *que minimize* $\left\|\beta e_1 - \widetilde{H}_j y\right\|_2$

$x_j = x_0 + V_j y_j.$

Para el valor de k en el ciclo sabemos que el método converge en n o menos iteraciones, nos falta aún estudiar cual será la forma de minimizar $\left\|\beta e_1 - \widetilde{H}_j y\right\|_2$, para ello nos basaremos en las rotaciones planas.

2.1.4 Implementación: Rotaciones planas

Para resolver el problema de mínimos cuadrados $\min_y \left\|\beta e_1 - \widetilde{H}_k y\right\|_2$, es natural transformar la matriz Hessenberg en una matriz triangular superior usando Rotaciones.

Una matriz de rotación 2×2 tiene la forma

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

donde $c^2 + s^2 = 1$. Esta matriz puede ser usada para introducir ceros en un vector, específicamente, sean a y b dados con $a \neq 0$ y sean $v = \sqrt{a^2 + b^2}$,

$$c = \frac{a}{v} \qquad \text{y} \qquad s = -\frac{b}{v};$$

entonces

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \frac{a^2+b^2}{v} \\ 0 \end{bmatrix}.$$

También podemos aplicar matriz de rotaciones para introducir ceros en una matriz de orden mayor, nuestro objetivo es usar las matrices de rotación para reducir el sistema original y llevarlo a una forma triangular. Definimos matrices cuadradas de Rotación de orden superior así

$$G_j = \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & c_j & -s_j & 0 & \cdots & 0 \\ \vdots & & \vdots & s_j & c_j & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

con $c_j^2 + s_j^2 = 1$. Si se realizan k pasos de GMRES esta matriz tiene orden $(k+1) \times (k+1)$. Multiplicamos a izquierda la matriz $\widetilde{H}_k$ y el vector $\widetilde{g}_0 = \beta e_1$ por las matrices de rotaciones. Los coeficientes s_j y c_j se escogen de tal modo que eliminen la entrada $h_{j+1,j}$ en cada paso. Así si $k = 5$ tenemos:

$$\widetilde{H}_5 = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & h_{1,5} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & h_{2,5} \\ 0 & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} \\ 0 & 0 & h_{4,3} & h_{4,4} & h_{4,5} \\ 0 & 0 & 0 & h_{5,4} & h_{5,5} \\ 0 & 0 & 0 & 0 & h_{6,5} \end{bmatrix} \qquad \widetilde{g}_0 = \begin{bmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

entonces premultiplicamos por la matriz de rotaciones G_1 de orden 6×6 con

$$s_1 = -\frac{h_{2,1}}{\sqrt{h_{1,1}^2 + h_{2,1}^2}} \qquad c_1 = \frac{h_{1,1}}{\sqrt{h_{1,1}^2 + h_{2,1}^2}}$$

y obtenemos

$$\widetilde{H}_5^{(1)} = G_1\widetilde{H}_5 = \begin{bmatrix} h_{1,1}^{(1)} & h_{1,2}^{(1)} & h_{1,3}^{(1)} & h_{1,4}^{(1)} & h_{1,5}^{(1)} \\ 0 & h_{2,2}^{(1)} & h_{2,3}^{(1)} & h_{2,4}^{(1)} & h_{2,5}^{(1)} \\ 0 & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} \\ 0 & 0 & h_{4,3} & h_{4,4} & h_{4,5} \\ 0 & 0 & 0 & h_{5,4} & h_{5,5} \\ 0 & 0 & 0 & 0 & h_{5,6} \end{bmatrix} \qquad \widetilde{g}_1 = \begin{bmatrix} c_1\beta \\ s_1\beta \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Premultiplicamos $\widetilde{H}_5^{(1)}$ y $\overline{g}_1$ por la matriz de rotación G_2 donde

$$s_2 = -\frac{h_{3,2}}{\sqrt{\left(h_{2,2}^{(1)}\right)^2 + h_{3,2}^2}} \qquad c_2 = \frac{h_{2,2}^{(1)}}{\sqrt{\left(h_{2,2}^{(1)}\right)^2 + h_{3,2}^2}}$$

para eliminar la entrada $h_{3,2}$. Este proceso de eliminación continúa hasta la quinta ($k\_ésima$) matriz de rotación, las cuales transforman el problema inicial en un sistema que involucra la matriz y término independiente siguientes

$$\widetilde{H}_5^{(5)} = \begin{bmatrix} h_{1,1}^{(1)} & h_{1,2}^{(1)} & h_{1,3}^{(1)} & h_{1,4}^{(1)} & h_{1,5}^{(1)} \\ 0 & h_{2,2}^{(2)} & h_{2,3}^{(2)} & h_{2,4}^{(2)} & h_{2,5}^{(2)} \\ 0 & 0 & h_{3,3}^{(3)} & h_{3,4}^{(3)} & h_{3,5}^{(3)} \\ 0 & 0 & 0 & h_{4,4}^{(4)} & h_{4,5}^{(4)} \\ 0 & 0 & 0 & 0 & h_{5,5}^{(5)} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \widetilde{g}_5 = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix}. \tag{2.10}$$

Los escalares c_j y s_j de la $j\_ésima$ rotación G_j están definidos por

$$s_j = -\frac{h_{j+1,j}}{\sqrt{\left(h_{j,j}^{(j-1)}\right)^2 + h_{j+1,1}^2}} \qquad c_j = \frac{h_{j,j}^{(j-1)}}{\sqrt{\left(h_{j,j}^{(j-1)}\right)^2 + h_{j+1,1}^2}}$$

con el fin de volver cero la entrada $(j+1, j)$ de la matriz Hessenberg. Definimos Q_k como el

producto de las G_j

$$Q_k = G_k G_{k-1} ... G_1$$

y

$$\widetilde{R}_k = \widetilde{H}_k^{(k)} = Q_k \widetilde{H}_k \tag{2.11}$$

$$\widetilde{g}_k = Q_k (\beta e_1) = \left(\gamma_1, \gamma_2, ..., \gamma_{k+1}\right)^T \tag{2.12}$$

como Q_k es unitaria

$$\min \left\| \beta e_1 - \widetilde{H}_k y \right\|_2 = \min \left\| Q_k \left(\beta e_1 - \widetilde{H}_k y \right) \right\|_2 = \min \left\| \widetilde{g}_k - \widetilde{R}_k y \right\|_2$$

la solución de este último problema es obtenido por solución del sistema triangular (2.10), quitando las últimas filas de $\widetilde{H}_k^{(k)}$ y $\widetilde{g}_k$ dadas en (2.10).

El algoritmo GMRES tiene la propiedad que la norma del residual puede ser computada sin la iteración x_j, así la iteración puede ser pospuesta hasta que el residual sea tan pequeño como se desee. En la siguiente proposición veremos el por qué de esta propiedad.

Proposición 2.1.7 *Sean G_j $j = 1, 2, ..., k$ matrices de rotaciones utilizadas para transformar $\widetilde{H}_k$ en una matriz triangular superior $\widetilde{R}_k$ y transformar βe_1 en $\widetilde{g}_k = \left(\gamma_1, \gamma_2, ..., \gamma_{k+1}\right)^T$ como se da en (2.11) y (2.12). Denotamos por R_k la matriz triangular superior $k \times k$ obtenida de $\widetilde{R}_k$ quitando la última fila y g_k vector de dimensión k obtenido de $\widetilde{g}_k$ quitando la última componente. Entonces*

1. *Rango de AV_k es igual al rango de R_k, en particular si $r_{k,k} = 0$ entonces A es singular.*
2. *El vector y_k que minimiza $\left\| \beta e_1 - \widetilde{H}_k y_k \right\|_2$ esta dado por $y_k = R_k^{-1} g_k$.*
3. *El vector residual en el paso k satisface*

$$b - A x_k = V_{k+1} \left(\beta e_1 - \widetilde{H}_k y_k \right) = V_{k+1} Q_k^T \left(r_{k+1} e_{k+1} \right)$$

y como resultado

$$\left\| b - A x_k \right\|_2 = \left| \gamma_{k+1} \right|$$

Prueba. Para la parte 1 usamos (2.6)

$$AV_k = V_{k+1}\widetilde{H}_k = V_{k+1}Q_k^T Q_k \widetilde{H}_k = V_{k+1}Q_k^T \widetilde{R}_k.$$

Como $V_{k+1}Q_k^T$ es unitaria, el rango de AV_k es el mismo rango de $\widetilde{R}_k$ el cual es igual al rango de R_k (estas matrices solo difieren en la última fila de ceros) . Si $r_{k,k} = 0$ entonces el rango de R_k es menor o igual a $k-1$ y como por tanto AV_k tiene rango $k-1$, dado que V_k tiene rango completo entonces A es singular.

La parte 2 está esencialmente probada arriba. Para cualquier y

$$\left\|\beta e_1 - \widetilde{H}_k y_k\right\|_2^2 = \left\|Q_k\left(\beta e_1 - \widetilde{H}_k y_k\right)\right\|_2^2 = \left\|\widetilde{g}_k - \widetilde{R}_k y\right\|_2^2 = \left|\gamma_{k+1}\right|^2 + \|g_k - R_k y\|_2^2 \qquad (2.13)$$

el mínimo valor se alcanza cuando $\|g_k - R_k y\|_2^2$ es cero, como R_k es no singular, esto ocurre cuando $y = R_k^{-1} g_k$.

Para la tercera parte tenemos

$$\begin{aligned} b - Ax_k &= V_{k+1}\left(\beta e_1 - \widetilde{H}_k y_k\right) = V_{k+1}Q_k^T Q_k\left(\beta e_1 - \widetilde{H}_k y_k\right) \\ b - Ax_k &= V_{k+1}Q_k^T\left(\widetilde{g}_k - \widetilde{R}_k y\right) \end{aligned}$$

como vimos en la parte anterior, la norma 2 de $\widetilde{g}_k - \widetilde{R}_k y$ es minimizada cuando $y = R_k^{-1} g_k$ y anulamos las componentes de $\widetilde{g}_k$ excepto la última, la cual es igual a γ_{k+1} luego

$$b - Ax_k = V_{k+1}Q_k^T\left(\gamma_{k+1} e_{k+1}\right).$$

El resultado $\|b - Ax_k\|_2 = \left|\gamma_{k+1}\right|$ se sigue de la normalidad de las columnas de $V_{k+1}Q_k^T$. ■

Algoritmo GMRES (basado en las matrices de rotaciones)

Lea A, x_0, b, *tol*, max $\_it$

$r_0 = b - Ax_0$

$\beta = \|r_0\|_2$

$v_1 = r_0/\beta$

$j = 1$

$g = \beta\,(1, 0, ..., 0) \in \mathbb{R}^{\max\_it+1}$

$ep = tol * \|b\|_2$

$Mientras\ \beta > ep\ y\ j \leq \max\_it$

$w_j = Av_j$

$para\ \ i = 1, ..., j$

$h_{i,j} = w_j^T v_i$

$w_j = w_j - h_{i,j} v_i$

fin

$h_{j+1,j} = \|w_j\|_2$

$v_{j+1} = w_j / h_{j+1,j}$

$si\ j > 1\ aplique\ Q_{j-1}\ a\ la\ j\_ésima\ columna\ de\ \widetilde{H}_j$

$v = \sqrt{h_{j,j}^2 + h_{j+1,j}^2}$

$c_j = h_{j,j}/v$

$s_j = -h_{j+1,j}/v$

$h_{j,j} = c_j h_{j,j} - s_j h_{j+1,j}$

$h_{j+1,j} = 0$

$g = G_j g$

$\beta = |g_{j+1}|$

$j = j + 1$

fin

$r_{i,k} = h_{i,k}\ para\ 1 \leq i, k \leq j$

$t_i = g_i\ para\ 1 \leq i \leq j$

$Resuelva\ el\ sistema\ triangular\ superior\ Ry = t$

$x_j = x_0 + V_j y.$

2.1.5 Variación GMRES

La mayor desventaja de GMRES es que la cantidad de trabajo de almacenamiento requerido por iteración aumenta linealmente con el conteo de iteraciones. A menos que uno sea bastante afortunado para obtener convergencia extremadamente rápido, el costo será alto. La forma usual para vencer esta limitación es reiniciar la iteración. Se escoge un número m de iteraciones, los

datos acumulados se borran y los resultados intermedios se usan como el dato inicial para las siguientes m iteraciones. Este procedimiento se repite hasta que la convergencia se alcance.

Algoritmo $GMRES(m)$

1. $r_0 = b - Ax_0$

$\beta = \|r_0\|_2$

2. Generar una base de Arnoldi y matriz $\widetilde{H}_m$ usando algoritmo de Arnoldi con $v_1 = r_0/\beta$

3. calcular y_m que minimiza $\left\|\beta e_1 - \widetilde{H}_m y\right\|_2$ y $x_m = x_0 + V_m y_m$

4. si satisface criterio de parada termine, si no $x_0 = x_m$ y volver a 1.

Notemos que en cada subpaso j conocemos la norma del residual sin computar x_j, esto permite al programa salir tan pronto como la norma sea suficientemente pequeña. Si no se usa GMRES(m), GMRES (como cualquier método de ortogonalización de subespacio de Krylov) converge en no más de n pasos. Por supuesto cuando n es grande el método tendrá una convergencia lenta y el costo de almacenamiento será inmenso. La dificultad está en la escogencia de un valor apropiado para m. Si m es "pequeño" GMRES(m) puede ser lento para la convergencia, o falla completamente. Desafortunadamente no existen reglas definitivas que definan la escogencia del m. Existen ejemplos para los cuales el método se estanca y la covergencia da lugar en el $n\_ésimo$ paso. Para tales sistemas cualquier escogencia de m menor que n falla en convergencia.

Saad y Schultz en [18] tienen probados varios resultados útiles, en particular ellos muestran que si la matriz de coeficientes A es real y cercanamente definida positiva, entonces un valor razonable para m puede ser seleccionado.

Ejemplo 2.1.1 *Para matrices de la forma* $A = trid\,(-1, 4, 1)$ *que no son simétricas compararemos los resultados obtenidos por CGNR con los obtenidos por GMRES(m) con* $m = 10$, $tol = 10^{-10}$ *y partimos del vector de ceros.*

	Orden A	*Iteración*	*Norma Residual*	*tiempo (seg.)*
CGNR	*5000*	*7*	*4.705609599e-08*	*0.063000*
GMRES	*5000*	*2-4*	*5.26921648e-011*	*0.0470000*
CGNR	*100000*	*7*	*4.706611964e-08*	*0.7820000*
GMRES	*100000*	*2-3*	*4.99213994e-011*	*2.2190000*

Para el caso de matrices de orden menor que 7500 mas o menos el método de GMRES converge en un menor tiempo, para matrices mayores es mejor la convergencia de CG en las ecuaciones normales, como era de esperarse debido al almacenamiento en memoria necesario para GMRES.

Ejemplo 2.1.2 *Para matrices pentadiagonales de la forma* $A = (5, 12, 25, -13, -8)$ *comparamos rutinas propias vs. rutinas internas matlab versión 6.5 ambas con la acción de la matriz sobre vector*

	Orden A	*Iteración*	*tiempo (seg.)*
GMRES	*10000*	*4-4*	*0.203000*
GMRES(matlab)	*10000*	*4-4*	*0.422000*
GMRES	*100000*	*4-2*	*4.735000*
GMRES(matlab)	*100000*	*4-2*	*8.781000*

Si A es matriz tridiagonal por bloques de la forma

$$A = \begin{bmatrix} D & I & & & \\ -I & D & I & & \\ & -I & \ddots & \ddots & \\ & & \ddots & D & I \\ & & & -I & D \end{bmatrix}$$

donde $D = trid(-5, 12, 5)$, *con tol* $= 10^{-10}$ *y vector inicial vector de ceros, tenemos:*

	Orden A	*Iteración*	*tiempo (seg.)*
GMRES	*10000*	*3-4*	*0.171000*
GMRES(matlab)	*10000*	*3-4*	*0.391000*
GMRES	*1000000*	*3-2*	*49.76600*
GMRES(matlab)	*1000000*	*3-2*	*3.382190e+002*

2.2 Precondicionamiento

El algoritmo de precondicionamiento requiere una solución de un sistema lineal con la matriz M en cada paso. Resolvemos el sistema $M^{-1}Ax = M^{-1}b$ precondicionamiento a izquierda o el sistema $AM^{-1}u = b$ donde $x = M^{-1}u$ precondicionamiento a derecha. El precondicionamiento

que se utilizó en CG se basa en la idea de conservar la simetría del sistema, utilizando una factorización de Cholesky incompleta $M = LL^T$, se resolvió el sistema $L^{-1}AL^{-T}u = L^{-1}b$ donde $x = L^{-T}u$ el cual envuelve una matriz simétrica definida positiva. A este precondicionamiento se le conoce como precondicionamiento dividido (derecha e izquierda).

2.2.1 Precondicionamiento izquierdo

El algoritmo precondicionado a izquierda consiste en aplicar GMRES(m) al sistema

$$M^{-1}Ax = M^{-1}b. \tag{2.14}$$

La aplicación directa de GMRES(m) a (2.14) produce solo los cambios para los r y w_j así

$$r = M^{-1}(b - Ax) \qquad \text{y} \qquad w_j = M^{-1}Av_j$$

el resto del algoritmo queda igual. En el ciclo de Arnoldi se construye una base ortogonal del subespacio de Krylov

$$gen\{r_0, M^{-1}Ar_0, \left(M^{-1}A\right)^2 r_0, ..., \left(M^{-1}A\right)^{m-1} r_0\}$$

todos los vectores residuales y las normas calculadas en el algoritmo corresponden al sistema precondicionado (2.14) , a saber, $r_m = M^{-1}(b - Ax_m)$, en vez del residual original $b - Ax_m$. Si el criterio de parada involucra los residuales originales, el algoritmo requiere un paso adicional que será premultiplicar los residuales por el precondicionador M aumentando así un producto matriz vector en cada paso y produciendo así mayor costo computacional.

2.2.2 Precondicionamiento derecho

El algoritmo precondicionado derecho de GMRES se basa en la solución de

$$AM^{-1}u = b \text{ donde } u = Mx \tag{2.15}$$

por GMRES(m) pero la nueva variable u nunca necesita ser llamada específicamente. Notemos que el residual inicial para el sistema precondicionado es $b - AM^{-1}u_0 = b - AM^{-1}(Mx_0) = r_0$ es decir, igual al original. La aproximación a la solución que se obtiene por este método es

$$u_m = u_0 + \sum_{j=1}^{m} \eta_i v_i$$

con $u_0 = Mx_0$. Multiplicando por M^{-1} la aproximación en términos de la variable x es

$$x_m = x_0 + M^{-1}\left[\sum_{j=1}^{m} \eta_i v_i\right].$$

Así, una operación de precondicionamiento es necesaria al final del ciclo de salida, en lugar del comienzo como en la versión precondicionada a izquierda. La aplicación de GMRES(m) a (2.15) produce solo cambios sobre w_j y x_m así

$$w_j = AM^{-1}v_j \qquad y \qquad x_m = x_0 + M^{-1}V_m y_m$$

las instrucciones restantes se conservan. En este caso el ciclo de Arnoldi construye una base ortogonal del subespacio precondicionado derecho de Krylov

$$gen\left\{r_0, AM^{-1}r_0, \left(AM^{-1}\right)^2 r_0, ..., \left(AM^{-1}\right)^{m-1} r_0\right\}.$$

Notemos que la norma residual es ahora relativa al sistema inicial, es decir, $b - Ax_m$, puesto que el algoritmo obtiene el residual $b - Ax_m = b - AM^{-1}u_m$ implícitamente. Esta es una diferencia esencial con el algoritmo GMRES(m) precondicionado a izquierda.

2.2.3 Precondicionamiento Dividido

En muchos casos M es el resultado de una factorización de la forma $M = LU$, luego existe la opción de usar GMRES en el sistema precondicionado dividido

$$L^{-1}AU^{-1}u = L^{-1}b \text{ donde } x = U^{-1}u.$$

En este caso, necesitamos premultiplicar el residual inicial por L^{-1} en el comienzo del algoritmo, por U^{-1} en la combinación lineal $V_m y_m$ (como se hizo en el precondicionamiento derecho con M^{-1}) formando la solución aproximada. La norma del residual para volver a empezar será $L^{-1}(b - Ax_m)$.

La diferencia entre los diferentes tipos de precondicionamiento GMRES radica en las diferentes normas de los residuales ya que pueden afectar el criterio de parada y pueden causar que el algoritmo pare ó prematuramente ó con retraso. Puede ser particularmente perjudicial en caso que M sea mal condicionada, por ejemplo, un precondicionador dividido puede ser mucho mejor si A es cercanamente simétrica.

2.2.4 Comparación entre precondicionamiento a Derecha e Izquierda

Cuando comparamos las opciones de precondicionamiento, una primera observación es que el espectro de los tres operadores asociados $M^{-1}A$, AM^{-1} y $L^{-1}AU^{-1}$ son idénticos, por lo tanto en principio uno espera que la convergencia sea similar.

Comparemos las propiedades de optimalidad para GMRES precondicionado a derecha e izquierda. Para el precondicionamiento izquierdo GMRES minimiza la norma residual

$$\left\| M^{-1}b - M^{-1}Ax \right\|_2$$

a lo largo de todos los vectores sobre el subespacio afín

$$x_0 + gen\left\{ z_0, M^{-1}Az_0, \left(M^{-1}A\right)^2 z_0, ..., \left(M^{-1}A\right)^{m-1} z_0 \right\} \tag{2.16}$$

en el cual z_0 es el residual inicial del sistema precondicionado $z_0 = M^{-1}r_0$. Así la solución aproximada se expresa como

$$x_m = x_0 + \sum_{j=0}^{m-1} \gamma_j \left(M^{-1}A\right)^j z_0 = x_0 + s_{m-1}\left(M^{-1}A\right) z_0$$

donde $s_{m-1}(z) = \sum_{j=0}^{m-1} \gamma_j \; z^j$ es el polinomio de grado menor o igual a $m-1$ que minimiza la

norma

$$\left\| z_0 - M^{-1}As\left(M^{-1}A\right)z_0 \right\|_2$$

entre todos los polinomios s de grado menor o igual a $m-1$. Esta condición de optimalidad es posible expresarla con respecto al vector residual r_0 así:

$$z_0 - M^{-1}As\left(M^{-1}A\right)z_0 = M^{-1}\left[r_0 - As\left(M^{-1}A\right)M^{-1}r_0\right]$$

pero

$$s\left(M^{-1}A\right)M^{-1}r = M^{-1}s\left(AM^{-1}\right)r \tag{2.17}$$

obteniendo la relación

$$z_0 - M^{-1}As\left(M^{-1}A\right)z_0 = M^{-1}\left[r_0 - AM^{-1}s\left(AM^{-1}\right)r_0\right]$$

por lo tanto

$$\left\| z_0 - M^{-1}As\left(M^{-1}A\right)z_0 \right\|_2 = \left\| M^{-1}\left[r_0 - AM^{-1}s\left(AM^{-1}\right)r_0\right] \right\|_2 . \tag{2.18}$$

Consideremos ahora el caso de GMRES precondicionado a derecha, en donde es necesario distinguir entre la variable original x y la variable transformada u relacionada con x y mediante $x = M^{-1}u$. Para la variable u del proceso GMRES precondicionado derecho se minimiza la norma dos de $r = b - AM^{-1}u$ donde u pertenece al subespacio afín

$$u_0 + gen\left\{r_0, AM^{-1}r_0, \left(AM^{-1}\right)^2 r_0, ..., \left(AM^{-1}\right)^{m-1} r_0\right\} \tag{2.19}$$

en el cual r_0 es el residual $r_0 = b - AM^{-1}u_0$, este residual es idéntico al residual asociado con la variable x puesto que $M^{-1}u_0 = x_0$. Multiplicando (2.19) a izquierda por M^{-1} y usando (2.17) , observamos que la variable x asociada con un vector del subespacio afín (2.19) pertenece al subespacio afín

$$x_0 + gen\left\{z_0, M^{-1}Az_0, \left(M^{-1}A\right)^2 z_0, ..., \left(M^{-1}A\right)^{m-1} z_0\right\},$$

que es idéntico al subespacio afín (2.16) dado en el precondicionamiento izquierdo. En otras palabras para el GMRES derecho la solución aproximada x_m puede además expresarse como

$$x_m = x_0 + \sum_{j=0}^{m-1} \zeta_j \left(AM^{-1}\right)^j r_0 = x_0 + M^{-1} s_{m-1}\left(AM^{-1}\right) r_0$$

donde $s_{m-1}(z) = \sum_{j=0}^{m-1} \zeta_j \; z^j$ es un polinomio de grado menor o igual a $m-1$ el cual minimiza la norma

$$\left\| r_0 - AM^{-1} s\left(AM^{-1}\right) r_0 \right\|_2 \tag{2.20}$$

entre todos los polinomios s de grado menor igual a $m-1$. Las cantidades a minimizar (2.18) y (2.20) difieren únicamente por una multiplicación por M^{-1}. Tenemos así que el GMRES precondicionado izquierdo minimiza $M^{-1}r$ y el GMRES precondicionado derecho minimiza r, donde r se toma sobre el mismo subespacio en ambos casos.

Resumimos ésto en

Proposición 2.2.1 *La aproximación a la solución obtenida por precondicionamiento izquierdo o derecho de GMRES es de la forma*

$$x_m = x_0 + s_{m-1}\left(M^{-1}A\right) z_0 = x_0 + M^{-1} s_{m-1}\left(AM^{-1}\right) r_0$$

donde $z_0 = M^{-1}r_0$ *y* s_{m-1} *es el polinomio de grado menor o igual a* $m-1$ *que minimiza la norma del residual* $\|b - Ax_m\|_2$ *en el caso precondicionamiento derecho y en el caso del precondicionamiento izquierdo* $\left\|M^{-1}(b - Ax_m)\right\|_2$.

2.3 MINRES

El método MINRES (Minimum Residual) se basa en el método de tridiagonalización de Lanczos para problemas simétricos, posiblemente indefinidos, $Ax = b$. En este método la iteración x_k minimiza $\|b - Ax\|_2$ sobre el subespacio afín $x_0 + K_k$ donde $K_k = gen\left\{r_0, Ar_0, ..., A^{k-1}r_0\right\}$. Nuevamente la idea fundamental es expresar x_k en términos de un conjunto $\{v_1, v_2, ..., v_k\}$ generador de K_k, en el cual v_1 es un múltiplo escalar del vector residual $r_0 = b - Ax_0$.

2.3.1 Proceso de tridiagonalización de Lanczos

El proceso de tridiagonalización de Lanczos [7] consiste en construir una base ortonormal que permite factorizar una matriz simétrica dada. Para más detalles ver Saunders [15]. El método de tridiagonalización de Lanczos es una simplificación del método de Arnoldi. Cuando A es simétrica, la matriz de Hessenberg se vuelve tridiagonal simétrica y dicha simetría nos permite reducir el proceso de Arnoldi a generar dos sucesiones de valores α_j para $j = 1, ..., k$ y β_j para $j = 1, ..., k-1$.

Teorema 2.3.1 *Si el algoritmo de Arnoldi se aplica a una matriz A simétrica real, se obtiene*

$$\begin{aligned} h_{i,j} &= 0 \qquad 1 \leq i < j-1 \\ h_{j,j+1} &= h_{j+1,j} \qquad j = 2, ..., k \end{aligned}$$

es decir, la matriz H_k obtenida es tridiagonal simétrica.

Prueba. El resultado es consecuencia de (2.7) , es decir, $H_k = V_k^T A V_k$ la cual es simétrica y además, es también matriz de Hessenberg. ■

La notación estándar es

$$\alpha_j = h_{j,j} \qquad y \qquad \beta_j = h_{j-1,j}$$

y si denotamos por T_k la matriz H_k del algoritmo de Arnoldi, entonces

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & \beta_4 & \alpha_4 & \ddots & \\ & & & \ddots & \ddots & \beta_k \\ & & & & \beta_k & \alpha_k \end{bmatrix}$$

concluimos así que el algoritmo de tridiagonalización de Lanczos está dado por

Algoritmo Lanczos

Lea v_1 vector unitario

para $j = 1, ..., k$

$w_j = Av_j - \beta_j v_{j-1}$

$\alpha_j = v_j^T w_j$

$w_j = w_j - \alpha_j v_j$

$\beta_{k+1} = \|w_j\| \qquad si\ \beta_{j+1} = 0 \quad termine$

$v_{k+1} = w_j/\beta_{j+1}$

end

donde $\beta_1 = 0$ y $v_0 = 0$.

2.3.2 Relación con los mínimos cuadrados

Después de k pasos del algoritmo de Lanczos obtenemos la factorización

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T = V_{k+1} \widetilde{H}_k$$

donde $\widetilde{H}_k = \begin{bmatrix} T_k \\ \beta_{k+1} e_k \end{bmatrix}_{(k+1)\times k}$ es matriz de Hessenberg y T_k como se mostró antes. La aproximación x_k del método MINRES, ver [16], minimiza $\|b - Ax\|_2$ sobre el conjunto

$$x_0 + gen\{v_1, ..., v_k\},$$

de aquí

$$x_k = x_0 + V_k y$$

donde $y \in \mathbb{R}^k$; notemos que

$$\begin{aligned} \|b - A(x_0 + V_k y)\|_2 &= \|(b - Ax_0) - AV_k y\|_2 \\ &= \left\|(b - Ax_0) - V_{k+1}\widetilde{H}_k y\right\|_2 \end{aligned}$$

si tomamos $\beta = \|b - Ax_0\|$ y $q_1 = \frac{b - Ax_0}{\beta}$ vector unitario, tenemos que $b - Ax_0 = \beta v_1 = \beta V_{k+1} e_1^T$ por lo tanto

$$\|b - A(x_0 + V_k y)\|_2 = \left\|V_{k+1}\left(\beta e_1^T - \widetilde{H}_k y\right)\right\|_2 = \left\|\beta e_1^T - \widetilde{H}_k y\right\|_2.$$

Se resuelve en forma similar a GMRES, con la ventaja de que el costo computacional disminuye debido a la forma de $\widetilde{H}_k$. La convergencia es más complicada y discutida en Paige, Parlett y Van Der Vorst [17].

2.4 Dimensión infinita

En el campo de ecuaciones lineales en H espacio complejo de Hilbert

$$Au = f$$

la $k\_ésima$ iteración de GMRES u_k minimiza la $H-norma$ del residual $r_k = f - Au_k$ sobre los vectores en el espacio afín $u_0 + K_k$, donde u_0 es la iteración inicial y K_k es el espacio de Krylov dado por $K_k = gen\left\{r_0, Ar_0, ..., A^{k-1}r_0\right\}$. Podemos expresar la propiedad de minimización como

$$\|r_k\|_H = \min_{u \in u_0 + K_k} \|f - Au\|_H . \tag{2.21}$$

Si tomamos $\mathbb{P}_k$ la clase de polinomios residuales

$$\mathbb{P}_k = \{p \ / \ p \text{ es polinomio de grado } k \text{ y } p(0) = 1\}$$

es fácil ver que

$$r_k = f - Au_k = p_k(A)\, r_0 \tag{2.22}$$

para algún $p_k \in \mathbb{P}_k$. La ecuaciones (2.21) y (2.22) nos permiten estimar

$$\|r_k\|_H = \min_{p \in \mathbb{P}_k} \|p(A)\, r_0\|_H \leq \|r_0\|_H \min_{p \in \mathbb{P}_k} \|p(A)\|_{L(H)} \tag{2.23}$$

donde $L(H)$ denota el espacio de operadores acotados en H con la norma estándar para operadores. Normalmente para realizar estimados para la convergencia de GMRES seleccionamos un polinomio específico $p_k \in \mathbb{P}_k$ y estimamos el tamaño del lado derecho de (2.23) usando

$$\min_{p \in \mathbb{P}_k} \|p(A)\|_{L(H)} \leq \|p_k(A)\|_{L(H)} .$$

Capítulo 3

OTROS MÉTODOS

Los métodos para sistemas lineales no simétricos GMRES, CGNR y CGNE son más fáciles de implementar, CGNR y CGNE tienen la desventaja de que en cada iteración se computa la acción de la traspuesta y que las matrices de coeficientes A^TA y AA^T tienen número de condición el cuadrado del de la matriz original. GMRES necesita sólo producto matriz vector y usa solo a A pero en cada iteración requiere un gran almacenamiento. El método ideal para sistemas no simétricos sería un método parecido al CG, que sólo necesite productos matriz vector, que esté basado en un principio de minimización o conjugación, que tenga un modesto almacenamiento y converja en n iteraciones para toda matriz no singular A. Un análisis del por qué estas condiciones no pueden construirse para una matriz general A lo podemos encontrar en [4]. Los métodos que aquí comentaremos se basan en esta idea.

El capítulo está distribuido como sigue. Sección 3.1 Biortogonalización de Lanczos, realizamos una construcción que permite buscar métodos similares a CG para sistemas cuya matriz A es no simétrica. Sección 3.2 BiCG, una descripción del método BiCG acompañado de unos ejemplos ilustrativos y sección 3.3 otros métodos para sistemas no simétricos, damos un breve paso por diferentes métodos que permiten evitar comportamientos irregulares en la convergencia y trabajan sólo con la acción de la matriz A y no con A^T.

3.1 Biortogonalización de Lanczos

El proceso de biortogonalización de Lanczos consiste en extender el proceso de tridiagonalización de Lanczos a matrices no simétricas y construir sucesiones biortogonales en lugar de sucesiones ortogonales.

El algoritmo propuesto por Lanczos para sistemas no simétricos permite generar un par de

bases biortogonales para los dos subespacios

$$K_k = gen\left\{v_1, Av_1, ..., A^{k-1}v_1\right\} \qquad y \qquad K_k^T = gen\left\{w_1, A^T w_1, ..., \left(A^T\right)^{k-1} w_1\right\}$$

de la siguiente forma.

Algoritmo Lanczos (matriz no simétrica)

Lea v_1, w_1 *tales que* $v_1^T w_1 = 1$

$\beta_1 = \delta_1 = 0$

$w_0 = v_0 = 0$

para $j = 1, 2, ..., k$

$\alpha_j = (Av_j)^T w_j$

$\widehat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$

$\widehat{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$

$\delta_{j+1} = \left|\widehat{v}_{j+1}^T \widehat{w}_{j+1}\right|^{1/2}$ *si* $\delta_{j+1} = 0$ *termine*

$\beta_{j+1} = \widehat{v}_{j+1}^T \widehat{w}_{j+1}/\delta_{j+1}$

$w_{j+1} = \widehat{w}_{j+1}/\beta_{j+1}$

$v_{j+1} = \widehat{v}_{j+1}/\delta_{j+1}$

fin

Buscamos que los vectores w_{j+1} y v_{j+1} sean ortogonales, luego si $w_{j+1} = \widehat{w}_{j+1}/\beta_{j+1}$ y $v_{j+1} = \widehat{v}_{j+1}/\delta_{j+1}$

$$w_{j+1}^T v_{j+1} = \frac{\widehat{w}_{j+1}^T \widehat{v}_{j+1}}{\beta_{j+1}\delta_{j+1}}$$

es decir necesitamos que $\beta_{j+1}\delta_{j+1} = \widehat{w}_{j+1}^T \widehat{v}_{j+1}$. Existen numerosas formas para esta igualdad, tomaremos $\delta_{j+1} = \left|\widehat{v}_{j+1}^T \widehat{w}_{j+1}\right|^{1/2}$ y $\beta_{j+1} = \widehat{v}_{j+1}^T \widehat{w}_{j+1}/\delta_{j+1}$, los cuales cumplen δ_{j+1} es positivo y $\beta_{j+1} = \pm\delta_{j+1}$, además, por construcción los vectores $v_j \in K_k$ y $w_j \in K_k^T$.

Similar al proceso de tridiagonalización de Lanczos, denotemos por T_k la matriz tridiagonal

$$\begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \delta_3 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_k \\ & & & \delta_k & \alpha_k \end{bmatrix}$$

obtenida del algoritmo anterior. De $\widehat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$ y $v_{j+1} = \widehat{v}_{j+1}/\delta_{j+1}$ se sigue

$$AV_k = V_k T_k + \delta_{k+1} v_{k+1} e_k^T \tag{3.1}$$

y de $\widehat{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$ y $w_{j+1} = \widehat{w}_{j+1}/\beta_{j+1}$ obtenemos

$$AW_k = W_k T_k + \beta_{k+1} w_{k+1} e_k^T.$$

Se puede probar (ver por ejemplo [19]) que los vectores $v_1, v_2, ..., v_k$ y $w_1, w_2, ..., w_k$ son bior-togonales, es decir, $v_j^T w_i = \delta_{ij}$ para $1 \leq i, j \leq k$ y por lo tanto se cumple la relación

$$W_k^T A V_k = T_k.$$

Para resolver el sistema $Ax = b$ por medio del algoritmo de Lanczos para sistemas no simétricos, tomamos $v_1 = r_0/\beta$ con $\beta = \|r_0\|_2$, la aproximación esta dada por

$$x_k = x_0 + V_k y_k \tag{3.2}$$

y buscamos que los residuales sean ortogonales a K_k^T, es decir

$$0 = W_k^T (b - Ax_k) = W_k^T r_0 - W_k^T A V_k y_k = W_k^T r_0 - T_k y_k$$

pero $W_k^T r_0 = W_k^T (\beta v_1) = \beta e_1$ por la bi-ortogonalidad de los w_j y v_j, por lo tanto si tomamos

$y_k = T_k^{-1}(\beta e_1)$ la aproximación está dada por

$$x_k = x_0 + V_k T_k^{-1}(\beta e_1).$$

Dado que en el algoritmo construimos base para K_k^T se puede resolver al mismo tiempo un sistema dual $A^T\widehat{x} = \widehat{b}$, tomando $w_1 = \widehat{r}_0/\widehat{\beta}$ con $\widehat{\beta} = \|\widehat{r}_0\|$ en forma similar al anterior procedimiento buscamos que los residuales de

$$\widehat{x}_k = \widehat{x}_0 + W_k\widehat{y}_k$$

sean ortogonales a K_k, es decir

$$0 = V_k^T\left(\widehat{b} - A^T\widehat{x}_k\right) = V_k^T\widehat{r}_0 - V_k^T A^T W_k\widehat{y}_k = V_k^T r_0 - T_k^T\widehat{y}_k$$

de aquí que

$$\widehat{x}_k = \widehat{x}_0 + W_k T_k^{-T}\left(\widehat{\beta}e_1\right).$$

El algoritmo será

Algoritmo

Lea A, b, x_0, tol

$r_0 = b - Ax_0$

$\beta = \|r_0\|_2$

$v_1 = r_0/\beta$ *tome* w_1 *tal que* $w_1^T v_1 = 1$

para $k = 1$ *hasta* $\|r_k\|_2 \leq tol * \|b\|_2$

generar los $v_1, v_2, ..., v_k, w_1, w_2, ..., w_k$ *y* T_k *con algoritmo anterior*

$y_k = T_k^{-1}(\beta e_1)$

$x_k = x_0 + V_k y_k$

fin

Para determinar la convergencia no es necesario computar explícitamente la aproximación, basta con computar

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \left|\delta_{k+1} e_k^T v_k\right| \ \ \|v_{k+1}\|_2$$

ya que por la relación (3.1)

$$\begin{aligned} b - Ax_k &= b - A\left(x_0 + V_k y_k\right) = r_0 - V_k T_k y_k - \delta_{k+1} v_{k+1} e_k^T y_k \\ &= V_k\left(\beta e_1 - T_k y_k\right) - \delta_{k+1} v_{k+1} e_k^T y_k \end{aligned}$$

como $y_k = T_k^{-1}\left(\beta e_1\right)$ se sigue que

$$\|r_k\|_2 = \left\|-\delta_{k+1} v_{k+1} e_k^T y_k\right\|_2 .$$

El algoritmo genera la base para K_k^T, pero como sólo estamos interesados en resolver el sistema $Ax = b$ omitimos las expresiones para $A^T\widehat{x} = \widehat{b}$, claro está que si queremos resolverlos ambos al tiempo agregamos las líneas correspondientes. En este caso los requerimientos de memoria son elevados y nuestro objetivo es obtener métodos similares al CG.

3.2 BiCG

El método de BiCG (Bi-gradiente conjugado) se obtiene del algoritmo de Lanczos para sistemas no simétricos, fue primero propuesto por Lanczos [8] en 1952 y después en una forma diferente por Fletcher [6] en 1974. Implícitamente el algoritmo no sólo resuelve el sistema $Ax = b$, también $A^T\widehat{x} = \widehat{b}$ el sistema lineal dual y genera dos sucesiones de vectores semejantes a las del método CG, una basada en un sistema con la matriz original A y otra con A^T. En lugar de ortogonalizar cada sucesión, se toman mutuamente ortogonales ó "bi-ortogonal". Este método, semejante al CG usa un almacenamiento limitado. Útil cuando la matriz es no simétrica y no singular, sin embargo su convergencia puede ser irregular y a menudo el método fracasa. BiCG requiere una multiplicación con la matriz de coeficientes y una con la traspuesta en cada iteración.

Las dos sucesiones de vectores son los residuales $\{r_k\}$ del sistema asociado a la matriz A y los $\{\widehat{r}_k\}$ a su traspuesta, se generan dos sucesiones de direcciones $\{p_k\}$ y $\{\widehat{p}_k\}$ tales que

$$\begin{aligned} \widehat{r}_k^T r_l &= 0 \qquad k \neq l \\ \widehat{p}_k^T A p_l &= 0 \qquad k \neq l \end{aligned}$$

los residuales cumplen la condición de bi-ortogonalidad y las direcciones de bi-conjugación.

Implementación

Tomamos la descomposición LU de T_k como $T_k = L_k U_k$ donde

$$L_k = \begin{bmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \lambda_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & \lambda_k & 1 \end{bmatrix} \qquad U_k = \begin{bmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_2 & & \\ & & \eta_3 & \ddots & \\ & & & \ddots & \beta_k \\ & & & & \eta_k \end{bmatrix}$$

luego las aproximaciones están dadas por

$$\begin{aligned} x_k &= x_0 + V_k U_k^{-1} L_k^{-1} (\beta e_1) \\ \widehat{x}_k &= \widehat{x}_0 + W_k L_k^{-1} U_k^{-1} \left(\widehat{\beta} e_1\right), \end{aligned}$$

si definimos

$$\begin{aligned} P_k &= V_k U_k^{-1} \qquad & \widehat{P}_k &= W_k L_k^{-1} \\ z_k &= L_k^{-1} (\beta e_1) \qquad & \widehat{z}_k &= U_k^{-1} \left(\widehat{\beta} e_1\right) \end{aligned}$$

tenemos que

$$\begin{aligned} x_k &= x_0 + P_k z_k \\ \widehat{x}_k &= \widehat{x}_0 + \widehat{P}_k \widehat{z}_k. \end{aligned}$$

Para P_k, $U_k P_k = V_k$ si tomamos $\beta_1 = 0,\ p_0 = 0$

$$p_j = \eta_j^{-1} \left(v_j - \beta_j p_{j-1}\right) \qquad j = 1, ..., k$$

de la descomposición LU de T_k tomando $\lambda_1 = 0$

$$\begin{aligned} \eta_j &= \alpha_j - \lambda_j \beta_j \qquad j = 1, ..., k \\ \lambda_j &= \frac{\delta_j}{\eta_{j-1}} \qquad j = 2, ..., k \end{aligned}$$

y por último se cumple

$$z_j = \begin{bmatrix} z_{j-1} \\ \xi_j \end{bmatrix}$$

donde $\xi_j = -\lambda_j \xi_{j-1}$ con $\xi_1 = \beta$.
En forma similar para $\widehat{P}_k$, $L_k \widehat{P}_k = W_k$ tomando $\widehat{p}_0 = 0$

$$\begin{aligned} \widehat{p}_j &= w_j - \lambda_j \widehat{p}_{j-1} \qquad j = 1, ..., k \\ \widehat{z}_j &= \begin{bmatrix} \widehat{z}_{j-1} \\ \widehat{\xi}_j \end{bmatrix} \end{aligned}$$

donde $\widehat{\xi}_j = -\frac{\beta_j}{\eta_j} \widehat{\xi}_{j-1}$ con $\widehat{\xi}_1 = -\widehat{\beta}/\eta_1$.
Por lo tanto

$$x_k = x_0 + \begin{bmatrix} P_{k-1} & p_k \end{bmatrix} \begin{bmatrix} z_{j-1} \\ \xi_j \end{bmatrix} = x_0 + P_{k-1} z_{j-1} + p_k \xi_j = x_{k-1} + p_k \xi_j$$

y similarmente

$$\widehat{x}_k = \widehat{x}_{k-1} + \widehat{p}_k \widehat{\xi}_j.$$

Los vectores $p_j \in P_k$ y $\widehat{p}_j \in \widehat{P}_k$ son A-conjugados puesto que

$$\left(\widehat{P}_k\right)^T A P_k = L_k^{-T} W_k^T A V_k U_k^{-1} = L_k^{-T} T_k U_k^{-1} = L_k^{-T} L_k U_k U_k^{-1} = I_k$$

y por construcción los residuales r_j y $\widehat{r}_j$ son ortogonales a K_k^T y K_k respectivamente y dado que $K_k = gen\left\{r_0, Ar_0, ..., A^{k-1} r_0\right\} = gen\left\{r_0, r_1, ..., r_{k-1}\right\} = gen\left\{p_1, p_2, ..., p_k\right\}$ se cumple que los residuales son ortogonales.

Nuestro objetivo es tener un método similar a CG. Del análisis anterior tenemos que

$$\begin{aligned} x_{j+1} &= x_j + \alpha_j p_j \\ \widehat{x}_{j+1} &= \widehat{x}_j + \alpha_j \widehat{p}_j \end{aligned}$$

luego los residuales

$$\begin{aligned} r_{j+1} &= r_j - \alpha_j A p_j \\ \widehat{r}_{j+1} &= \widehat{r}_j - \alpha_j A^T \widehat{p}_j, \end{aligned} \tag{3.3}$$

las direcciones p_{j+1} son una combinación lineal de los vectores r_{j+1} y p_j (ya que los v_j son base para K_k y $K_k = gen\{r_0, r_1, ..., r_{k-1}\}$)

$$p_{j+1} = r_{j+1} + \beta_j p_j \tag{3.4}$$

$$\widehat{p}_{j+1} = \widehat{r}_{j+1} + \beta_j \widehat{p}_j. \tag{3.5}$$

Para hallar los α_j utilizamos la propiedad de bi-ortogonalidad de los residuales

$$0 = r_j^T \widehat{r}_{j+1} = r_j^T \left(\widehat{r}_j - \alpha_j A^T \widehat{p}_j\right) = r_j^T \widehat{r}_j - \alpha_j \left(r_j^T A^T \widehat{p}_j\right)$$

a su vez de (3.4) y de la bi-conjugación de las direcciones

$$\left(r_j^T A^T \widehat{p}_j\right) = \left(A^T \widehat{p}_j\right)^T r_j = \left(A^T \widehat{p}_j\right)^T \left(p_j - \beta_{j-1} p_{j-1}\right) = \left(A^T \widehat{p}_j\right)^T p_j = \widehat{p}_j^T A p_j$$

luego

$$\alpha_j = \frac{r_j^T \widehat{r}_j}{\widehat{p}_j^T A p_j}. \tag{3.6}$$

En el caso de β_j utilizamos la propiedad de bi-conjugación de las direcciones y (3.5)

$$0 = (Ap_j)^T \widehat{p}_{j+1} = (Ap_j)^T \left(\widehat{r}_{j+1} + \beta_j \widehat{p}_j\right) = (Ap_j)^T \widehat{r}_{j+1} + \beta_j (Ap_j)^T \widehat{p}_j$$

luego

$$\beta_j = -\frac{\widehat{r}_{j+1}^T A p_j}{\widehat{p}_j^T A p_j},$$

notamos que de (3.3)

$$Ap_j = -\frac{1}{\alpha_j}\left(r_{j+1} - r_j\right)$$

por lo tanto de (3.6)

$$\begin{aligned} \beta_j &= \frac{1}{\alpha_j}\frac{\widehat{r}_{j+1}^T\left(r_{j+1} - r_j\right)}{\widehat{p}_j^T A p_j} = \frac{\widehat{r}_{j+1}^T r_{j+1}}{\alpha_j\ \widehat{p}_j^T A p_j} \\ \beta_j &= \frac{\widehat{r}_{j+1}^T r_{j+1}}{r_j^T \widehat{r}_j} \end{aligned}$$

Para el algoritmo de BiCG se parte de $\widehat{r}_0 = r_0$ y por ende $\widehat{p}_0 = p_0 = 0$, si se quiere resolver el sistema dual $A^T\widehat{x} = \widehat{b}$ y se agrega la línea $\widehat{x}_{j+1} = \widehat{x}_j + \alpha_j\widehat{p}_j$ y $\widehat{r}_0 = \widehat{b} - A^T\widehat{x}_0$.

Algoritmo BiCG

Lea A, x_0, b, tol, max_it

$r_0 = b - Ax_0$

$\widehat{r}_0 = r_0$

$ep = tol * \|b\|_2$

para $k = 1, 2, ..., max\_it$

$\rho_{k-1} = \widehat{r}_{k-1}^T r_{k-1}$ *si* $\rho_{k-1} = 0$ *el método falla*

si $k = 1$

$p_1 = r_0$

$\widehat{p}_1 = \widehat{r}_0$

en otro caso

$\beta_{k-1} = \frac{\rho_{k-1}}{\rho_{k-2}}$

$p_k = r_{k-1} + \beta_{k-1}p_{k-1}$

$\widehat{p}_k = \widehat{r}_{k-1} + \beta_{k-1}\widehat{p}_{k-1}$

fin

$q_k = A * p_k$

$\widehat{q}_k = A^T * \widehat{p}_k$

$\alpha_k = \frac{\rho_{k-1}}{\widehat{p}_k^T q_k}$

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

$r_k = r_{k-1} - \alpha_k q_k$

$\widehat{r}_k = \widehat{r}_{k-1} - \alpha_k \widehat{q}_k$

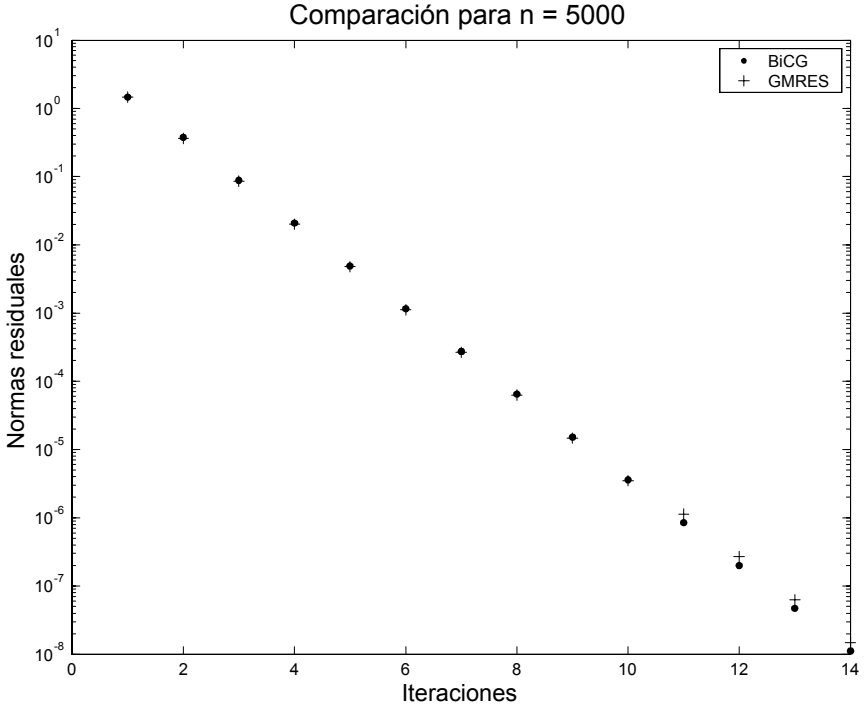
$si\ \|r_k\|_2 \leq ep\ entonces\ termine$

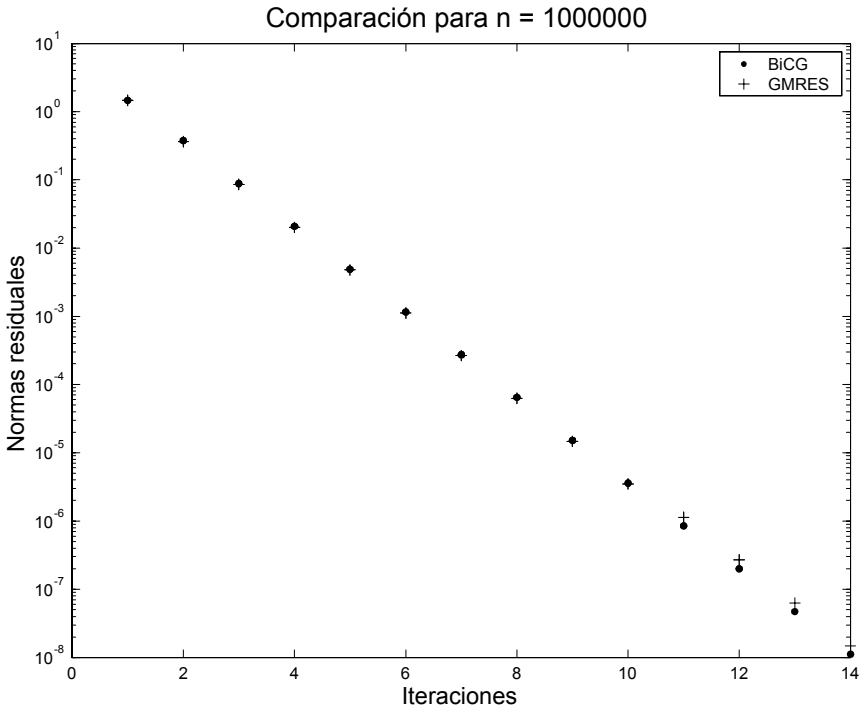
fin

fin

Ejemplo 3.2.1 *Para matrices de la forma* $A = trid\,(-1, 4, 1)$ *que no son simétricas compararemos los resultados obtenidos por GMRES(m) con* $m = 10$ *con los obtenidos por Bi-CG tomando* $tol = 10^{-10}$ *y vector inicial de ceros.*

	Orden A	*Iteración*	*Norma Residual*	*Norma Error*	*tiempo (seg.)*
GMRES	*5000*	*2-4*	*5.26921648e-011*	*3.534757e-009*	*0.172000*
Bi-CG	*5000*	*14*	*1.11430434e-08*	*2.634609e-009*	*0.063000*
GMRES	*1000000*	*2-2*	*6.68751793e-011*	*6.353604e-008*	*29.42200*
Bi-CG	*1000000*	*12*	*1.99998407e-07*	*4.728670e-008*	*20.62500*

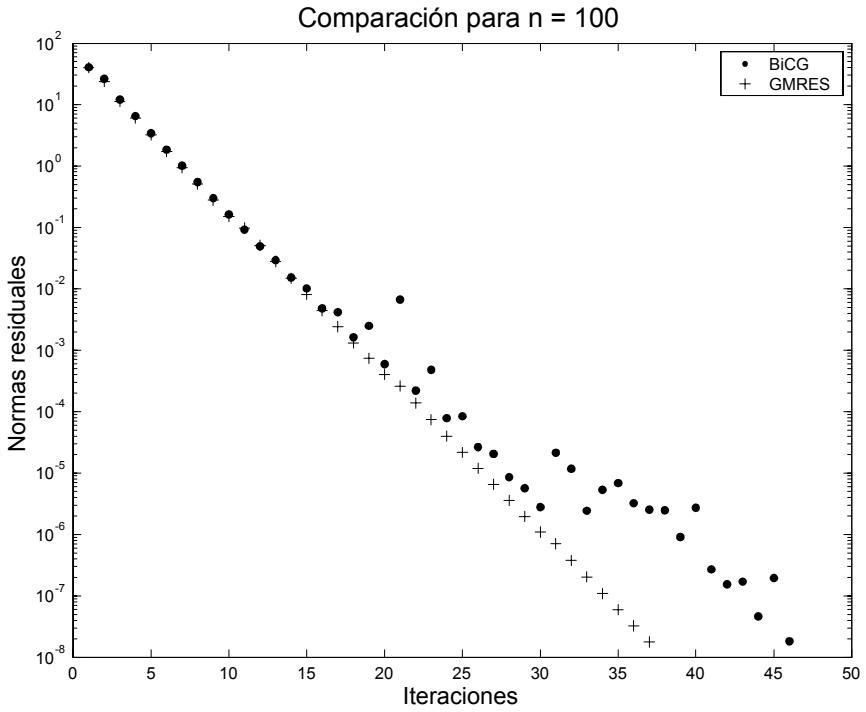




en ambos casos vemos que ganamos tiempo de ejecución y las aproximaciones son mejores, en las gráficas de las normas residuales vs. iteraciones se observa que los residuales obtenidos por BiCG son mejores.

En algunas ocasiones la convergencia del método se vuelve irregular y el método puede ser un fracaso, esto es, cuando $\widehat{r}_{k-1}^T r_{k-1} \approx 0$ ó $\widehat{p}_k^T q_k \approx 0$. Si $\widehat{p}_k^T q_k \approx 0$ ocurre, la descomposición LU falla y puede ser reemplazada por otra descomposición.

Ejemplo 3.2.2 *Para la matriz pentadiagonal de la forma* $A = pent\,(5, 12, 25, -13, -8)$ *comparamos los resultados obtenidos por GMRES(m) con* $m = 10$ *con los obtenidos por Bi-CG tomando tol* $= 10^{-10}$*, vector inicial de ceros y la gráfica de las normas residuales vs. iteraciones obtenida es*



en este caso el método de BiCG necesita más iteraciones para satisfacer el criterio de parada.

3.3 Otros métodos para sistemas no simétricos

Los métodos que aquí mencionamos se utilizan para evitar los casos de fracaso en BiCG y evitar el utilizar la acción de A^T.

QMR (Residual Cuasi-minimal): Este método resuelve un problema de mínimos cuadrados y actualiza en el BiCG los residuales, de este modo dejamos por fuera el comportamiento irregular de convergencia de BiCG. QMR en gran parte evita el fracaso que puede ocurrir en BiCG en particular si la descomposición LU falla.

La idea principal es minimizar el residual en forma similar como se hace en GMRES. Puesto que la base es bi-ortogonal la solución obtenida es como una solución residual cuasi-minimal lo cual explica su nombre.

Si trabajamos (3.2) desde el punto de vista de mínimos cuadrados, es decir, hallaremos y_k que minimice la norma del residual r_k. Del algoritmo de Lanczos para matrices no simétricas

se cumple la relación

$$AV_k = V_{k+1}\widetilde{T}_k$$

donde $\widetilde{T}_k$ es de orden $(k+1) \times k$ tridiagonal

$$\widetilde{T}_k = \begin{bmatrix} T_k \\ \delta_{k+1} e_k^T \end{bmatrix}$$

de forma similar al método GMRES

$$\|b - Ax_k\|_2 = \left\| V_{k+1} \left(\beta e_1 - \widetilde{T}_k y_k \right) \right\|$$

en el algoritmo de Lanczos los $v_j \in V_{k+1}$ no son ortonormales, sin embargo, es razonable minimizar $\left\| \beta e_1 - \widetilde{T}_k y \right\|$ para $y \in \mathbb{R}^k$.
Así la aproximación será $x_k = x_0 + V_k y_k$ donde $y_k = y \in \mathbb{R}^k \min \left\| \beta e_1 - \widetilde{T}_k y \right\|$, similar como en GMRES, excepto que el proceso de Arnoldi es reemplazado por el proceso de Lanczos. Para una implementación y análisis de la familia de los métodos QMR nos referimos a [11] y [12].

CGS (Gradiente conjugado cuadrado): El CGS fue desarrollado por Sonneveld en 1984 principalmente para evitar usar traspuesta de A en el método BiCG y ganar rapidez de convergencia aproximadamente al mismo costo computacional. La idea principal está basada en lo siguiente: en BiCG el residual r_j puede ser obtenido como el producto de r_0 por un polinomio de grado j en A, es decir,

$$r_j = p_j(A)\, r_0$$

este mismo polinomio satisface

$$\widehat{r}_j = p_j\left(A^T\right) \widehat{r}_0$$

así que

$$\rho_{j-1} = \widehat{r}_j^T r_j = \left(p_j\left(A^T\right) \widehat{r}_0 \right)^T \left(p_j(A)\, r_0 \right) = \widehat{r}_0^T\ p_j^2(A)\, r_0$$

esto sugiere que si $p_j(A)$ reduce r_0 a un vector más pequeño r_j, puede ser ventajoso aplicar este operador "contracción" dos veces y computar $p_j^2(A)\, r_0$. Más aún, podemos encontrar unas

fórmulas equivalentes a las de BiCG sólo en términos de la acción de A. CGS requiere al menos el mismo número de operaciones por iteración que en BiCG pero no involucra computación con A^T. Para detalles de este método nos referimos a [14] y para el análisis de su convergencia a [13] y [9].

Bi-CGSTAB (Gradiente biconjugado estabilizado): CGS se basa en polinomios cuadrados de los residuales y en caso de convergencia irregular se producen muchos errores de redondeo y posiblemente lleva a un desbordamiento del método. El Bi-CGSTAB es una variación de CGS para remediar esta dificultad. En lugar de computar $p_j^2(A)\, r_0$ Bi-CGSTAB computa $Q_j(A)\, P_j(A)\, r_0$ donde Q_j es un polinomio de grado j definido recursivamente en cada paso con el objetivo de "estabilizar" o "suavizar" la variación de la convergencia del algoritmo original. Los ejemplos en [9] indican la efectividad de las aproximaciones, una descripción detallada e implementación del algoritmo la encontramos en [9].

Bibliografía

[1] A. Greenbaum, *Iterative Methods for Solving Linear Systems,* SIAM, (1997).

[2] C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations,* SIAM, (1995).

[3] W. J. Kammerer y M. Z. Nashed, *On the convergence of the conjugate gradient method for singular linear operator equations,* SIAM J. Num. Anal., (1972), pp. 165 - 181.

[4] V. Faber and T. A. Manteuffel, *Necessary and sufficient conditions for the existence of a conjugate gradient method,* SIAM J.Numer. Anal., (1984), pp. 352-362.

[5] R. Barrett , M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods,* SIAM, Philadelphia, PA, 1995.

[6] R. Fletcher, *Conjugate gradient methods for indefinite systems,* in Numerical Analysis Dundee 1975, G. Watson, ed., Springer - Verlag, Berlin, New York, 1976, pp. 73 - 89.

[7] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential operators,* J. Res. N.B.S., 45 (1950), pp. 255-282

[8] _____, *Solution of linear equations by minimized iterations,* J. Res. Natl. Bur. Stand., 49 (1952), pp. 33 - 53.

[9] H. A. Van Der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant to Bi-CG for the solution of nonlinear systems,* SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631 - 664.

[10] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems,* J. Res. Nat. Bur. Stand., 49 (1952), pp. 409-436.

[11] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *QMR: a quasi-minimal residual algorithm for non-hermitian linear systems,* Numer. Math., 60 (1991), pp. 315-339.

[12] _____, *An implementation of the QMR method based on coupled two-term recurrences,* SIAM J. Sci. Comput., 15 (1994), pp. 313-337.

[13] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?,* SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778-795.

[14] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear system,* SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36-52.

[15] M. A. Sanders, *Solution of sparse rectangular systems,* BIT 35 (1995), pp. 588-604.

[16] C. C. Paige and M. A. Saunders, *Solution of sparse Indefinite systems of linear equations,* SIAM J. Num. Anal. 12 (1975), pp. 617-629.

[17] C. C. Paige, B. N. Parlett, and H. A. Van Der Vorst, *Aproximate solutions and eigenvalue bounds from Krylov subspaces,* Numer. Linear Algebra wiht Applic., 2 (1995), pp. 115-134.

[18] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear system,* SIAM J. Scientific and Stat. Comp. 7 (1986), pp. 856-869.

[19] Y. Saad, *Iterative Methods for Sparse Linear Systems,* SIAM, 2003.