Obligatorio 1 - Implementación del método de "Residuo Mínimo Generalizado" (GMRES)

Grupo 36

Guillermo Daniel Toyos Marfurt, Juan José Mangado Esteves,
Martín Sebastián Piñeyro Olivera, Ricardo Alejandro Fasanello Guillermo

Tutor: Juan Piccini

Métodos Numéricos 2020
Instituto de Matemática y Estadística
Facultad de Ingeniería. Universidad de la República
Montevideo, Uruguay

Abstract

Se estudia la resolución de sistemas lineales $Ax = b$ donde A es una matriz dispersa utilizando el método iterativo de Residuo Mínimo Generalizado (GMRES). Para ello se implementa este método y se prueba la complejidad computacional de su implementación. Adicionalmente se encuentra una cota teórica dependiente de los valores propios de A. La cual asegura la convergencia y rapidez del método si el cociente entre el radio y el centro del disco donde los valores propios caen es pequeño. También se estudia experimentalmente el comportamiento de GMRES con matrices dispersas aleatorias y se presentan distintas gráficas que caracterizan el comportamiento del método. El informe finaliza con pruebas de tiempo medio de ejecución que comparan el método GMRES con el comando *"contrabarra"* de *Octave* mostrando la superioridad de GMRES a medida que la dimensión de A es cada vez más grande.

Keywords: Sistemas lineales, Métodos Iterativos, GMRES, Matrices dispersas

1 Introducción

El problema de resolver un sistema $Ax = b$ es uno de los problemas fundamentales del álgebra lineal. Éste problema, a priori, es fácil de resolver; se podría usar un método de resolución directo, como puede ser el método de Escalerización Gaussiana, y así obtener los valores de las incógnitas.

El problema aparece cuando la cantidad de incógnitas (la dimensión de A) es muy grande, ya que los métodos directos como la escalerización Gaussiana o descomposición LU no solo necesitan realizar demasiadas operaciones sino también requieren almacenar matrices auxiliares de tamaño prohibitivo. Por ello se idearon los métodos iterativos, estos son un tipo de método el cual se basa en aproximar sucesivamente la solución en cada iteración, mejorando la precisión conforme avanzamos en la iteración. Existen diversos métodos iterativos, los cuales dependiendo de las características del problema se "acercan" más rápido que otros. En este informe se estudiará el método iterativo GMRES propuesto en 1986 por Youcef Saad y Martin H. Schultz [**?**] . GMRES resulta particularmente eficiente para resolver problemas donde el número de incógnitas es enorme, pero estás aparecen contadas veces en cada ecuación. Esto es cuando A es una matriz dispersa.

El informe realiza los siguientes aportes:

- Se proporciona una síntesis de la metodología que utiliza GMRES y como se implementa el mismo
- Análisis de complejidad computacional de las distintas partes del algoritmo GMRES
- Obtención una cota teórica de la norma del residuo relativo. Generando un criterio de convergencia y rapidez de esta.
- Un estudio experimental con matrices dispersas de distribución aleatoria para confirmar la cota del residuo propuesta y la velocidad de convergencia de GMRES
- Comparación de tiempos de ejecución de la implementación de GMRES con el comando *"contrabarra"* de *Octave*

El informe se organiza de la siguiente manera. La sección (3) trata sobre la metodología, implementación y complejidad de GMRES. La sección (4) presenta los resultados del análisis experimental junto al desarrollo de la cota teoría para la convergencia del método. Finalmente, la sección (5) presenta las conclusiones halladas.

2 Problema a resolver

Dada una matriz $A \in \mathbb{M}^{n\times n}$, un vector $b \in \mathbb{M}^{n\times 1}$ se quiere hallar un vector $x \in \mathbb{M}^{n\times 1}$: $Ax = b$. Queremos resolver este problema cuando la matriz A y el vector b son dispersos, ya que es en estas condiciones que el algoritmo GMRES resulta eficiente.

3 Metodología

A continuación, se presenta una síntesis de la maquinaria matemática que utiliza GMRES para su implementación.

3.1 Rotaciones de Givens

Las rotaciones de Givens son transformaciones lineales ortogonales definidas como

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

Específicamente sus entradas no nulas se definen con $i > J$ en:

$$\begin{aligned} g_{kk} &= 1, \ (k \neq i \wedge k \neq j) \\ g_{kk} &= c, \ (k = j \vee k = i) \\ g_{ji} &= -g_{ij} = s \end{aligned}$$

G^t representa rotaciones antihorarias sobre el plano (i, j) de un espacio de $n > 1$ dimensiones de θ radianes: $G(i, j, \theta)^t$. Podemos escoger los parámetros c y s con el objetivo de anular una de las entradas de un vector $x \in \mathbb{R}^n$. Suponiendo que queremos anular las entradas x_i y x_j basta asignar los parámetros con:

$$c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \qquad s = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}$$

Existen varias formas de calcular s y c. Nuestra implementación de GMRES utiliza el algoritmo [5.1.3][**?**] que reduce el error de redondeo al evitar la división directa por la norma de x_i y x_j. De todas maneras el cálculo de los parámetros de una rotación de Givens es $O(1)$.

3.2 QR de una matriz de Hessenberg aplicando Givens

Dada una matriz $H \in \mathbb{R}^{(n+1)\times n}$, H se considera "Hessenberg superior" cuando los valores de la subdiagonal inferior son todos nulos.Ésto mismo también se aplica para matrices $n \times n$.
Podemos utilizar rotaciones de Givens para realizar la descomposición QR de H de forma eficaz.Por ejemplo, sea $H \in \mathbb{R}^{(n+1)\times n}$, con $n = 3$ y entradas $\times$ arbitrarias:

$$H = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{pmatrix}$$

donde las $\times$ son entradas arbitrarias. Para escalerizar H (y por ende hallar su descomposición QR), se le va aplicando rotaciones de Givens $G(i, j, \theta)$ de forma iterativa, donde $G(j, i, \theta)$ anula el elemento (i,j)-ésimo de H:

$$G(1,2,\theta_1)^T \cdot H = \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{pmatrix} = G(2,3,\theta_2)^T \cdot G(1,2,\theta_1)^T \cdot H = \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{pmatrix} =$$

$$G(3,4,\theta_3)^T \cdot G(2,3,\theta_2)^T \cdot G(1,2,\theta_1)^T \cdot H = \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{pmatrix} = R$$

Sí denominamos al producto de las rotaciones de Givens transpuestas como Q^T, se concluye el cómputo para la descomposición QR de H.

Según Golub y Van Loan (2013)[**?**], el algoritmo para el proceso descripto anteriormente requiere alrededor de n^2 $flops$. Si bien cuándo H es de Hessemberg no hay una ventaja evidente en la cantidad de operaciones entre utilizar rotaciones de Givens y Escalerización Gaussiana (el cual es $O(n^2)$ para matriz de Hessenberg), como este último realiza una cantidad de divisiones sobre las entradas mucho mayor va a tener más propagación de error. Especialmente cuando los valores de las entradas son pequeños. Además, las rotaciones de Givens eliminan la preocupación del pívot nulo. Simplificando la implementación.

3.3 Mínimos Cuadrados con QR

Se desea resolver el siguiente Problema de Mínimos Cuadrados Lineal. Hallar $x \in \mathbb{R}^m$ que minimice $||Hx - c||_2^2$, con $H \in \mathbb{R}^{(m+1)\times m}, c \in \mathbb{R}^{m+1}$.

$$\min_{x\in\mathbb{R}^m} ||Hx - c||_2^2$$

Tenemos que $R = Q^T H$, con R matriz triangular superior. $Q^{-1} = Q^T$. Utilizando este hecho junto con la proposición [4.3.2][**?**]: $||QX||_2 = ||X||_2$ tenemos que:

$$\min_{x\in\mathbb{R}^m} ||Hx-c||_2^2 = \min_{x\in\mathbb{R}^m} ||QRx-c||_2^2 = \min_{x\in\mathbb{R}^m} ||RX-Q^t c||_2^2 = \min_{x\in\mathbb{R}^m} ||R_1 x-(Q^t c)_1||_2^2+||(Q^t c)_2||_2^2$$

Separando R en R_1 (matriz triangular superior cuadrada formada por las primeras m filas de R) y escribiendo $(Q^t c) \in \mathbb{R}^m + 1$ como $(Q^t c)_1 \in \mathbb{R}^m$, $(Q^t c)_2 \in \mathbb{R}^{m+1-m}$ como los vectores formados por los primeros m y $m + 1 - m$ elementos de $Q^t c$ respectivamente. Como $||(Q^t c)_2||_2^2$ no depende de X el problema se traduce a resolver el sistema determinado $R_1 X = (Q^t c)_1$. Como R_1 es triangular superior, x se obtiene haciendo sustitución hacia atrás.

Ejemplo Solicitado: Probamos el método implementado con las siguientes matrices H y c:

$$H = \begin{pmatrix} 2 & 0 & 1 & 5 \\ -3 & 2 & -6 & 4 \\ 0 & 4 & 7 & 1 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}, \; c = \begin{pmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Obteniendo como resultado el vector solución $X^t \approx (1,398 \;\; 0,698 \;\; -0,3936 \;\; 0,060)$. Ejecutando el comando $H \setminus c$ en *Octave* obtenemos exactamente los mismos resultados. Esto es esperado dado que este comando busca la solución de mínimos cuadrados si la

matriz no es cuadrada[**?**].

3.4 Ortogonalización con Gram-Schmidt

El proceso de ortogonalización de Gram-Schmidt es un método utilizado principalmente en álgebra lineal y análisis numérico el cual toma como entrada un conjunto de vectores no ortogonales y linealmente independientes, y retorna una base ortonormal.

Dada una matriz $A = \left(a_1, \ldots, a_n\right) \in \mathbb{R}^{m\times n}$ tal que $m \geq n$, $rango(A) = n$ y sus columnas $a_1, \ldots, a_n$ son linealmente independientes entre sí, se puede aplicar Gram-Schmidt para hallar su descomposición QR fina Q_1R_1, donde $Q = \left(Q_1, Q_2\right) \in \mathbb{R}^{m\times m}$ (Q_1 son sus primeras n columnas, y Q_2 sus últimas $m - n$) , $R \in \mathbb{R}^{m\times n}$ tal que sus últimas $m - n$ filas son nulas ,y R_1 es el mismo R_1 definido en 3.3. Ésto también se conoce como el algoritmo de Gram-Schmidt Clásico (GSC):

Algorithm 1 GSC

Require: A ▷ $A \in \mathbb{R}^{m\times n}$

1: $R_1(1,1) \leftarrow \|A(:,1)\|_2$
2: $Q_1(:,1) \leftarrow \frac{A(:,1)}{R_1(1,1)}$
3: **for** $k = 2 : n$ **do**
4: $\quad R_1(1 : k-1, k) \leftarrow Q_1(1 : m, 1 : k-1)^T A(1 : m, k)$
5: $\quad z \leftarrow A(1 : m, k) - Q_1(1 : m, 1 : k-1)R_1(1 : k-1, k)$
6: $\quad R_1(k,k) \leftarrow \|z\|_2$
7: $\quad Q_1(1 : m, k) \leftarrow \frac{z}{R_1(k,k)}$
8: **end for**
9: **return** Q_1, R_1

Tiempo de ejecución de GSC: El bucle **for** del paso 3 se ejecuta $n - 2$ veces. Dentro del mismo **for** y por cada iteración, en el paso 4 se producen $k - 1$ productos internos ($2m - 1$ operaciones) correspondientes a las primeras $k - 1$ columnas de Q_1 y la k-ésima columna de A. De forma análoga ocurren la misma cantidad de operaciones para el paso 5,pero como son pasos secuenciales, no se toma en cuenta para el orden asintótico del algoritmo. Para los pasos 6 y 7 la cantidad de operaciones son $2m$ y m respectivamente, por lo que tampoco se tomarán en cuenta. El orden asintótico se termina de deducir mediante la siguientes ecuaciones:

$$\sum_{k=2}^{n}\sum_{i=1}^{k-1} 2m - 1 = \sum_{k=2}^{n}(2m-1)(k-1) \sim 2m\sum_{k=1}^{n} k = \frac{2m(n+1)n}{2} \sim mn^2$$

Por último, se concluye que el coste de operaciones de GSC es asintóticamente $O(mn^2)$.

Un problema que puede surgir a la hora de computar GSC es una pérdida de ortogonalidad de los vectores columna de Q_1, causada principalmente por errores de redondeo, o cuando los vectores de la matriz de entrada A ”rozan” la dependencia lineal.Por éstos motivos se considera que GSC es numéricametne inestable.Para evitar situaciones así, se pretende alcanzar una mayor estabilidad modificando GSC para que vaya calculando los vectores ortonormales de forma iterativa.Éste proceso alternativo deriva un nuevo algoritmo llamado Gram-Schmidt Modificado (GSM):

Algorithm 2 GSM

Require: A ▷ $A \in \mathbb{R}^{m \times n}$

for $k = 1 : n$ **do**
  $R_1(k,k) \leftarrow \|A(1:m,k)\|_2$
  $Q_1(1:m,k) \leftarrow A(1:m,k)/R_1(k,k)$
  for $j = k+1 : n$ **do**
    $R_1(k,j) \leftarrow Q_1(1:m,k)^T A(1:m,j)$
    $A(1:m,j) \leftarrow A(1:m,j) - Q_1(1:m,k)R_1(k,j)$
  end for
end for

Tiempo de ejecución de GSM: Se sigue un razonamiento similar al del tiempo de ejecución de GSC:
paso 2: $2m-1$ operaciones (m multiplicaciones y $m-1$ sumas).
paso 3: m divisiones.
paso 5: $2m-1$ operaciones (m multiplicaciones y $m-1$ sumas).
paso 6: $2m$ operaciones (m multiplicaciones y m restas).
Se termina de deducir el orden de operaciones con la siguiente ecuación:

$$\sum_{k=1}^{n} 3m-1 \sum_{j=k+1}^{n} 4m-1 = n(3m-1) + \sum_{k=1}^{n}(4m-1)(n-k)$$

$$= n(3m-1) + (4m-1)(n^2 - \frac{n(n+1)}{2}) = n(3m-1) + (4m-1)(\frac{n(n-1)}{2})$$

$$\sim \frac{4mn^2}{2} = 2mn^2$$

Se concluye que el orden de operaciones de GSM es asintóticamente igual al de GSC, osea, $O(mn^2)$. Cabe destacar que, a diferencia de GSC, en GSM no se retornan Q_1, R_1 , sino que A se sobrescribe con Q_1, mientras que R_1 se almacena aparte.

El seudocódigo para los algoritmos de GSC y GSM fueron extraídos del Matrix

Computations (Golub y Van loan, 2013)[**?**]

3.5 Base ortonormal del Subespacio Krylov

El subespacio de Krylov de dimensión m, generado por una matriz cuadrada $A \in \mathbb{R}^{n \times n}$ y un vector $r \in \mathbb{R}^n$ se define como sigue:

$$K = K^m(A, r) = \left[r, Ar, A^2r, ..., A^{m-1}r\right]$$

Siendo $B_0 = \left\{r, Ar, A^2r, ..., A^{m-1}r\right\}$ una base del subespacio de Krylov.

Si queremos obtener una base ortonormal del subespacio de Krylov, bastaría con aplicar el **Algoritmo de Arnoldi**, que consiste en tomar la base B_0 y aplicar GSM.

En el siguiente algoritmo, se pasan como parámetros la matriz A, el vector r, y la dimensión del subespacio de Krylov (m). Se retorna el subespacio de Krylov ortonormalizado, en forma de matriz de vectores columna; y a su vez, se retorna una matriz H Hessenberg.

Algorithm 3 Arnoldi

Require: A, r, m ▷ $A \in \mathbb{R}^{n \times n}, r \in \mathbb{R}^n, m \in \mathbb{N}$

$K[:, 0] \leftarrow \frac{r}{\|r\|_2}$
for $j = 0 : m$ **do**
  $w_j \leftarrow AK[:, j]$
  for $i = 0 : j + 1$ **do**
    $H[i, j] \leftarrow K[:, i] * w_j$
    $w_j \leftarrow w_j - H[i, j] * K[:, i]$
  end for
  $H[j + 1, j] \leftarrow \sqrt{w_j . w_j}$
  if $H[j + 1, j] = 0$ **then**
    break
  end if
  if $j \neq m - 1$ **then**
    $K[:, j + 1] = \frac{w_j}{H[j+1,j]}$
  end if
end for
return K, H

Este algoritmo es una adaptación del presentado por Y. Saad en Iterative methods for sparse linear systems[**?**]

3.6 Orden de operaciones de GMRES

Una vez presentados y desarrollados los métodos aplicados para el algoritmo de GMRES, se indica el orden de operaciones para una iteración del bucle **while**. Para lograr ésto, se toma como referencia el seudocódigo propuesto en la letra del Obligatorio 1:

- En el **paso 3**, realizar un paso de Arnoldi cuesta $m(2m-1)$ operaciones ($2m-1$ en realizar la multiplicación $A * K_j$, en m iteraciones).Ésto también puede deducirse como:

$$\sum_{i=1}^{m} 2m - 1 = m(2m-1) \sim O(m^2)$$

- Para el **paso 4**, realizar una rotación de Givens requiere de 6 operaciones.Si a ésto se lo multiplica por la cantidad de entradas debajo de la subdiagonal inferior de H de Hessenberg,se obtiene el siguiente orden de operaciones:

$$\sum_{i=1}^{m-1} 6i = \frac{6m(m-1)}{2} \sim O(m^2)$$

- En el **paso 5**, el orden para resolver PMCL se deduce del orden para hallar la descomposición QR ($O(3m^2)$ por paso 4), y aplicar sustitución hacia atrás ($O(m^2)$ según las notas del curso[**?**]) .Se sigue cumpliendo un orden asintótico de $O(m^2)$.
- Como en el **paso 6** solo se actualiza el residuo calculando una norma, se sabe que cuesta $2m$ operaciones.
- Finalmente, en el **paso 7** incrementar m solo cuesta 1 operación.

Por lo listado anteriormente, se concluye que una iteración del bucle **while** de GMRES es de orden $O(m^2)$ de operaciones.

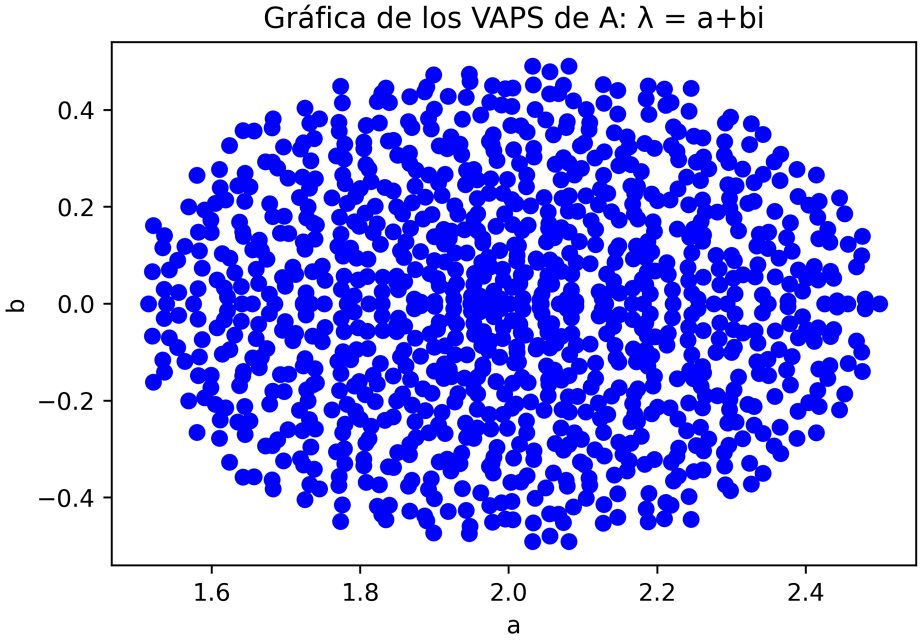
4 Análisis Experimental

4.1 Análisis y convergencia de GMRES

Para el análisis experimental se trabaja con una matriz $A \in \mathbb{M}_{n\times n}$ y un vector $b \in \mathbb{M}_{n\times 1}$ aleatorios y esparsos de densidad d con sus entradas no nulas generadas a partir de la siguiente distribución con parámetros:

$$N\left(\mu = 2, \sigma = \frac{0,5}{\sqrt{n \times d}}\right) \quad , n = 1000, \quad d = 0.01$$

Los parámetros de tolerancia mínima y máxima cantidad de iteraciones son $1tol = \times 10^{-6}$ y $Max = 30$ respectivamente.
Sea $\lambda(A)$ el conjunto formado por todos los valores propios de A. Graficando los mismos en $\mathbb{C}$ podemos observar que $\lambda(A)$ están dentro de $\mathscr{C}(2 + 0i, \frac{1}{2})$:



Esto implica que: $(\forall \lambda \in \lambda(A))(|\lambda - 2| \leq \frac{1}{2})$. A continuación generalizaremos el criterio de convergencia según el disco donde se encuentra $\lambda(A)$. Sea $c \in \mathbb{R}$ el centro de la circunferencia y $r \in \mathbb{R}$ el radio de la misma. Utilizando propiedades de complejos y reemplazando por los parámetros mencionados:

$$|\lambda - c| \leq r \Leftrightarrow |1 - \frac{Re(\lambda)}{c} - i\frac{Im(\lambda)}{c}| \leq \frac{r}{|c|} \tag{1}$$

Asumiendo que A es diagonalizable: $A = QDQ^{-1}$ y dada la proposición 6.32 del *Iterative methods for sparse linear systems*[**?**]:

$$\frac{||r^m||_2}{||r^0||_2} \leq k(Q) \max_{\lambda \in \lambda(A)} |p_m(\lambda)| \; \forall p_m \in \mathbb{P}_m / p(0) = 1 \tag{2}$$

Donde $k(Q)$ es el número de condición de Q. Utilizando el polinomio $p_m(z) = (\frac{-z}{c} + 1)$,

podemos obtener una cota para el residuo relativo de la siguiente forma:

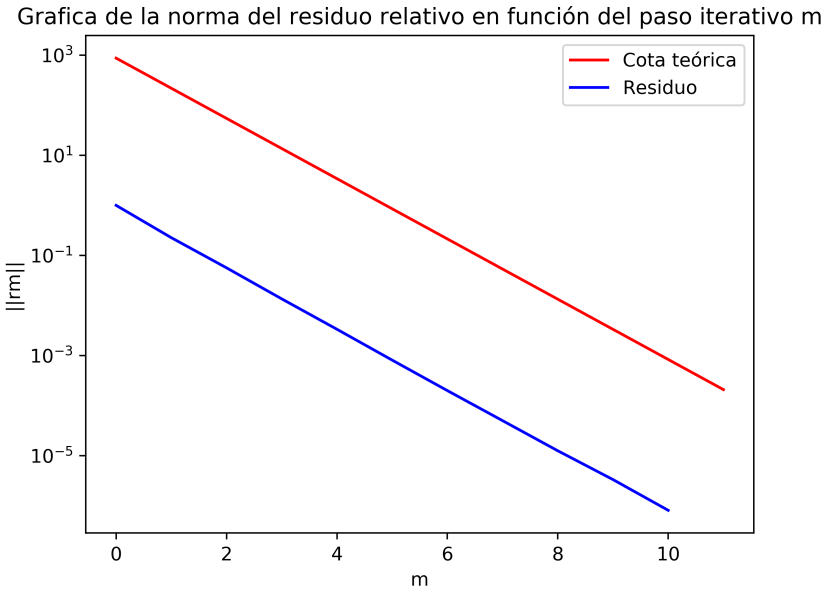
$$|p_m(z)| = |(\frac{-z}{c} + 1)^m| = |\frac{-z}{c} + 1|^m = |(1 - \frac{Re(z)}{c}) + i\frac{Im(z)}{c}|^m$$

El cual decrece en función de $m \Leftrightarrow |(1 - \frac{Re(z)}{c}) - i\frac{Im(z)}{c}| < 1$. Geométricamente esto sucede cuando z se encuentra dentro de $\mathscr{C}(c + 0i, c)$ en $\mathbb{C}$. Ahora bien, utilizando este hecho junto a (1) y (2):

$$\frac{||r^m||_2}{||r^0||_2} \leq k(Q) \max_{\lambda \in \lambda(A)} |p_m(\lambda)| = k(Q) \max_{\lambda \in \lambda(A)} |(1 - \frac{Re(\lambda)}{c}) - i\frac{Im(\lambda)}{c}|^m \leq k(Q)\Big(\frac{r}{|c|}\Big)^m \tag{3}$$

Lo cual muestra que la cota de la norma del residuo decrece exponencialmente en función del paso iterativo $m \Leftrightarrow r < |c|$. Es decir, sí $0 + 0i$ no se encuentra dentro de $\mathscr{C}(c, r)$ el método GMRES es convergente.

Entonces, como los $\lambda(A)$ están dentro de $\mathscr{C}(2 + 0i, \frac{1}{2})$, GMRES converge. Este hecho se confirma graficando la evolución de la norma del residuo relativo. Obsérvese que la norma se encuentra siempre debajo de la cota teórica:



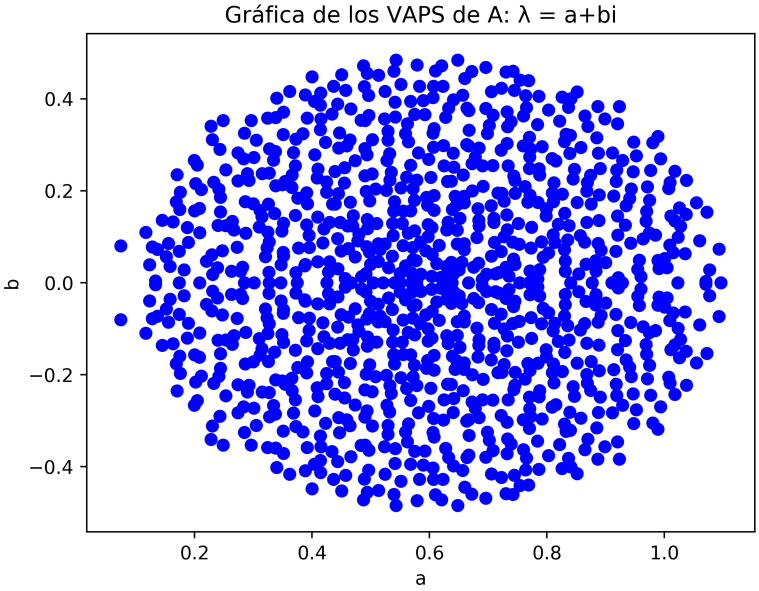
Adicionalmente, comparamos la solución de nuestra implementación de GMRES con la función *numpy.linalg.solve*, la cual resuelve sistemas lineales utilizando el método directo de descomposición LU[**?**] . Observamos que los 6 dígitos decimales más significativos para cada entrada del vector solución X son exactamente iguales para ambos

métodos.

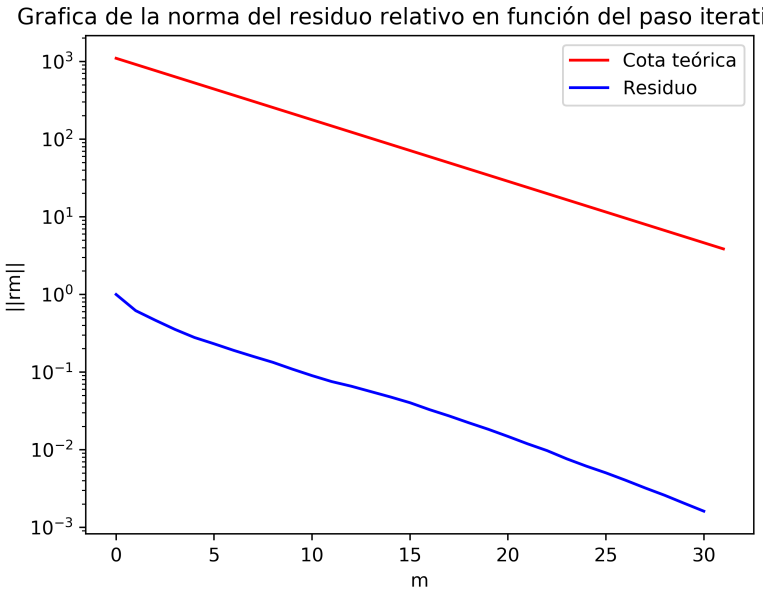
Alterando los parámetros de la distribución de las entradas no nulas de la matriz A nos encontramos con resultados análogos.

Con $A \sim N\left(\mu = \frac{3}{5}, \sigma = \frac{0{,}5}{\sqrt{1000 \times 0{,}01}}\right)$

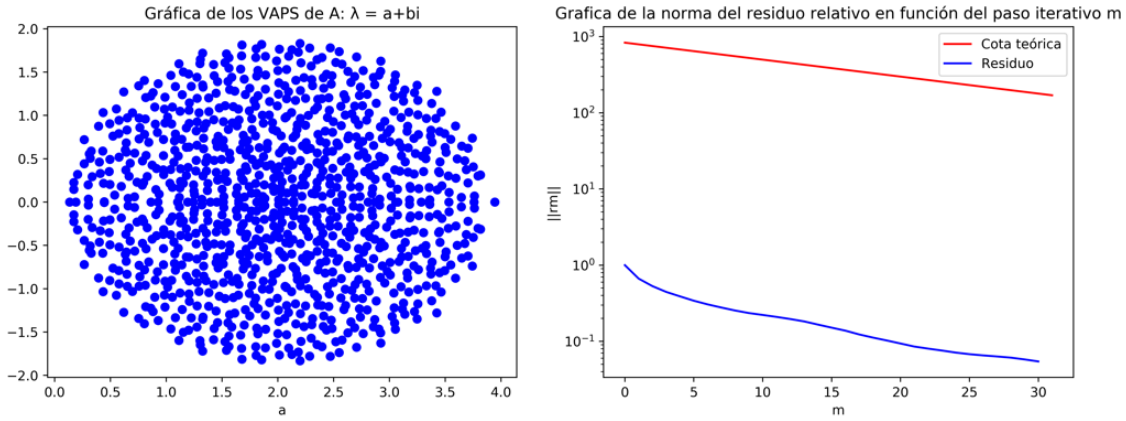
Graficando $\lambda(A)$ observamos que están dentro de $\mathscr{C}(\frac{1}{6} + 0i, \frac{1}{2})$:



Es decir, $c = \frac{3}{5}$, $r = \frac{1}{2}$ y $\frac{r}{c} \approx 0,8333$. Por lo que GMRES va a converger. Sin embargo, converger le llevará más iteraciones que en la matriz anterior y de hecho el algoritmo terminará por el tope máximo de iteraciones y no por alcanzar la tolerancia mínima:



Por último probamos con con $A \sim N\left(\mu = 2\ , \sigma = \frac{1{,}9}{\sqrt{1000 \times 0{,}01}}\right)$. Graficando el disco donde se encuentra $\lambda(A)$ y la evolución del residuo relativo:

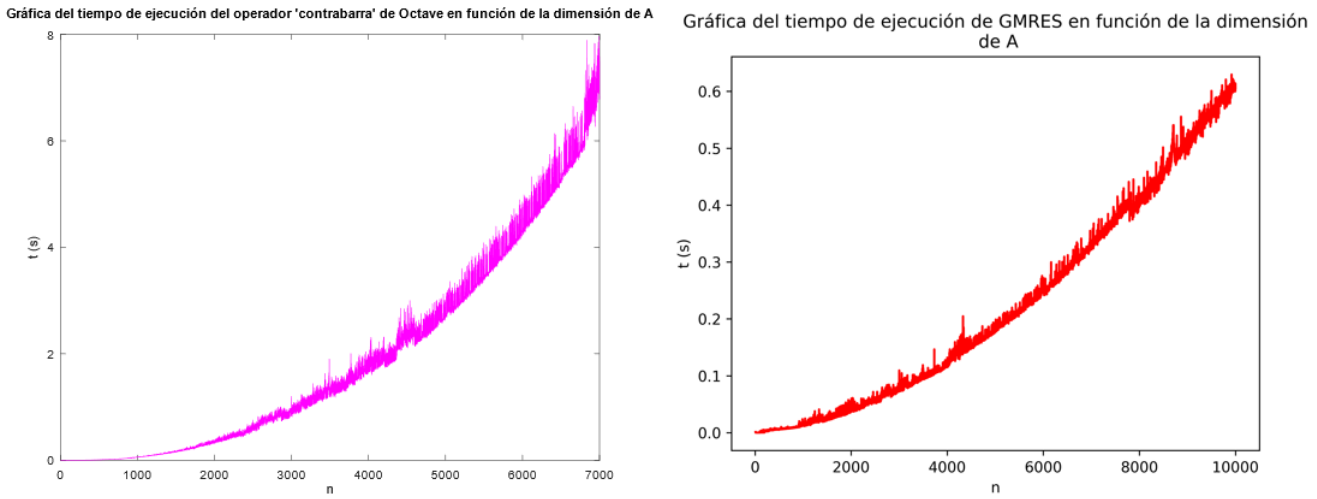


Lo cual muestra que (con $r = 1{,}9$, $c = 2$ y $\frac{r}{c} = 0{,}95$) GMRES converge pero lentamente. En efecto, el mismo cesa por el máximo de iteraciones y con un error de 5 órdenes mayor a la tolerancia mínima pedida.

Comparando los resultados con *numpy.linalg.solve* obtenemos soluciones acordes a la norma del error relativo en las últimas dos matrices estudiadas.

4.2 Comparación de pruebas de tiempo

Para finalizar el análisis experimental, comparamos los tiempos medios de ejecución de nuestra implementación de GMRES con el comando ”contrabarra” de *Octave* en función de la dimensión de la matriz A donde $A \sim N\left(\mu = 2, \sigma = \frac{0{,}5}{\sqrt{n \times d}}\right), \quad d = 0.01$



Las gráficas muestran que la tendencia de crecimiento del tiempo de ejecución medio de GMRES es mucho menor al del comando *“contrabarra”*. Por ejemplo, cuando $n = 3000$ nuestra implementación resuelve el sistema, en promedio, aproximadamente 20 veces más rápido que *“contrabarra”*, y cuando $n = 7000$ lo resuelve 27 veces más rápido. Nótese como el problema se torna rápidamente irresoluble en la práctica sí solo contásemos con el comando *“contrabarra”*

5 Conclusiones

Tras ahondar en la metodología que utiliza GMRES, hemos conseguido implementar el algoritmo y estudiar su comportamiento satisfactoriamente. También encontramos una cota teórica del residuo relativo de este método iterativo, asegurando su convergencia, según A.

Concluimos que el método GMRES resulta extremadamente eficiente para matrices dispersas con distribución normal que cumplen que sus valores propios caen en un disco de centro c y radio r donde el cociente entre c y r es menor a 1. Es decir sí $\frac{r}{|c|} < 1$ GMRES converge y cuanto más pequeño es este cociente mas rápido lo hace. También demostramos que en la práctica, el tiempo de ejecución de GMRES tiene un crecimiento mucho más lento que el comando *“contrabarra”* de *Octave* para resolver $Ax = b$ a medida que la dimensión de A crece para las matrices con las características mencionadas.

Esto muestra la importancia del uso de métodos no tradicionales para resolver el problema de hallar un x que cumpla $Ax = b$ dependiendo de las características de A. Sí A fuese de dimensión $n = 1000000$ no hubiese sido posible resolver el problema $Ax = b$ en un tiempo razonable utilizando el comando *"contrabarra"*. Mientras que el método GMRES lo resuelve en unos pocos segundos sí la matriz A cumple las características deseadas.