TEMA 2: ALGEBRA LINEAL COMPUTACIONAL

1. INTRODUCCIÓN

Objetivos del tema: estudiar algoritmos numéricos estables que nos permitan:

- Resolver sistemas lineales de la forma

$$Ax = b$$

donde A es una matriz de "gran tamaño".

- Calcular los autovalores y autovectores de una matriz grande A, es decir resolver el problema: encontrar escalares $\lambda \in \mathbb{R}$ y vectores no nulos $x \in \mathbb{R}^n$ tales que

$$Ax = \lambda x.$$

La eficiencia numérica de los algoritmos que estudiaremos depende de las propiedades de la matriz A, es decir, algunos algoritmos serán más útiles para una determinada ~~ma~~ tipo de matriz A, y otros métodos lo serán para otras matrices.

A modo de introducción, consideremos los 4 tipos siguientes de matrices:

1) Matrices de rigidez K_n, también llamadas de Toeplitz

$$K_n = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & \cdots & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & & \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & -1 & 2 & \end{bmatrix}_{n \times n}$$

Propiedades de K_n:

a) Es simétrica: $K_n = K_n^T$

b) Es sparse (dispersa), la mayoría de sus entradas $(K_n)_{ij}$ son cero.

Por ejemplo, K_{1000} tiene 1.000 000 entradas pero, a lo sumo, 1000 + 999 + 999 son no nulas.

c) Los elementos no nulos están situados en una banda alrededor de la diagonal principal.

2) Matrices circulantes C_n:

Son como las matrices de Toeplitz (o rigidez), pero se añade un -1 en la primera fila a la derecha y en la última a la izquierda. Por ejemplo:

$$C_4 = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

Comparte las propiedades (a) y (b) anteriores. La diferencia fundamental es que K_n es invertible y C_n no.

3) Matrices top row (o frontera) T_n:

Son como las matrices K_n, pero la primera entrada se sustituye por 1. Por ejemplo:

$$T_3 = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Son simétricas y tridiagonales.

La principal característica de este tipo de matrices es que son fácilmente reducibles a matrices triangulares superiores (U), cuyos sistemas lineales asociados son fácilmente resolubles. Veamos un ejemplo:

$$T_3 = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \xrightarrow{F_2 \to F_2 + F_1} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

$$\xrightarrow{F_3 \to F_3 + F_2} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} = U_3.$$

4) <u>Matrices tipo both ends</u> (ambas fronteras) B_n:

Son como las T_n pero la última entrada también vale 1. Por ejemplo.

$$B_3 = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

Eliminación gaussiana conduce a:

$$\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \xrightarrow{F_2 \to F_2 + F_1} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \xrightarrow{F_3 \to F_3 + F_2} \underbrace{\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}}_{U}.$$

MÉTODOS NUMÉRICOS PARA LA RESOLUCIÓN DE SISTEMAS LINEALES $Ax = b$

2. MÉTODOS DIRECTOS (Basados en eliminación Gaussiana)

- Son métodos muy eficaces cuando A es una matriz simétrica y definida positiva.
- La idea fundamental es que la eliminación Gaussiana (es decir, realizar operaciones elementales sobre las filas de una matriz) permite factorizar A en la forma $A = LU$, siendo L una matriz triangular inferior con 1s en la diagonal principal, y U una matriz triangular superior. L proviene de "lower" y U de "upper".

$$\underbrace{\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ 0 & & & u_{nn} \end{bmatrix}}_{U}$$

Así, el sistema $Ax = b$ se reescribe como

$LUx = b$ y llamando $z = Ux$ se llega a

$$\begin{cases} Lz = b \\ Ux = z \end{cases}$$

que son dos sistemas fácilmente resolubles al ser L y U matrices triangulares (inferior y superior). Esta idea conduce al llamado método de factorización LU.

- Además, si A es simétrica, entonces U se puede factorizar como $U = DL^T$, con D una matriz diagonal que recoge en su diagonal los "pivots" de la eliminación Gaussiana. Al igual que antes, el sistema $Ax = b$ se reescribe en la forma equivalente y más sencilla

$$\begin{cases} Lz = b \\ DL^T x = z \end{cases}$$

Esta idea conduce al llamado método de Cholesky.

Veamos ahora los detalles.

1) Resolución de sistemas triangulares (inferior / superior)

Consideremos el sistema triangular superior

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Este tipo de sistemas se resuelven en cascada, de abajo hacia arriba. En concreto, de la última ecuación se tiene $x_n = \dfrac{b_n}{a_{nn}}$. La penúltima ecuación es

$a_{n-1\,n-1}\, x_{n-1} + a_{n-1\,n}\, x_n = b_{n-1}$, de donde

$$x_{n-1} = \frac{b_{n-1} - a_{n-1\,n}\, x_n}{a_{n-1\,n-1}}.$$

En general, $$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}\, x_j}{a_{ii}}$$

Por tanto, el algoritmo es

$$\begin{cases} x_n = \dfrac{b_n}{a_{nn}} \\ x_i = \dfrac{b_i - \sum_{j=i+1}^{n} a_{ij}\, x_j}{a_{ii}}, \quad i = n-1, \ldots, 2, 1. \end{cases}$$

Nótese que para que esto funcione es esencial que $a_{ii} \neq 0$, $1 \leq i \leq n$, lo cual está garantizado si A es definida positiva.

Veamos cuál es el coste computacional de este algoritmo.

Para ello, consideremos el caso $n=4$.

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 \\ a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 \\ a_{33}x_3 + a_{34}x_4 &= b_3 \\ a_{44}x_4 &= b_4 \end{aligned}\right\}$$

$$x_4 = \frac{b_4}{a_{44}} \longrightarrow 1 \text{ operación}$$

$$x_3 = \frac{b_3 - a_{34}x_4}{a_{33}} \longrightarrow 3 \text{ operaciones}$$

$$x_2 = \frac{b_2 - a_{23}x_3 - a_{24}x_4}{a_{22}} \longrightarrow 5 \text{ operaciones}$$

$$x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3 - a_{14}x_4}{a_{11}} \longrightarrow 7 \text{ operaciones}$$

En total, 16 operaciones, es decir 4^2.

Se observa que para calcular x_i necesitamos

$2(n-i)+1$ operaciones.

Por tanto, el número de operaciones totales es

$$\sum_{i=1}^{n} [2(n-i)+1] = 2\sum_{i=1}^{n}(n-i) + n$$

$$= 2[(n-1)+(n-2)+\dots+(n-n)]+n$$

$$= 2\left(n^2 - \frac{n(n+1)}{2}\right)+n$$

$$= 2n^2 - n^2 - n + n = \boxed{n^2}$$

ya que $1 + \dots + n = \frac{n(n+1)}{2}$. En efecto:

$$S = 1 + 2 + \dots + n$$
$$S = n + n-1 + \dots + 1$$
$$2S = n \cdot (n+1) \Rightarrow S = \frac{n(n+1)}{2}$$

En resumen, el coste computacional de resolver un sistema triangular de tamaño n es n^2.

2) ¿Cómo se calculan L y U tales que $A = LU$?

Veamos un ejemplo concreto. Consideremos el sistema

$Ku = f$ $\quad \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$,

con $f = (4, 0, 0)$, de modo que el sistema es

$$\left.\begin{aligned} 2u_1 - u_2 &= 4 \\ -u_1 + 2u_2 - u_3 &= 0 \\ -u_2 + 2u_3 &= 0 \end{aligned}\right\}$$

Como de costumbre en la eliminación Gaussiana, reemplazamos la segunda ecuación por dicha ecuación $+ \frac{1}{2}$ primera ecuación (en términos de operaciones elementales $F_2 \to F_2 + \frac{1}{2}F_1$)

$$\begin{bmatrix} \textcircled{2} & -1 & 0 \\ 0 & \textcircled{\frac{3}{2}} & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + \frac{1}{2}f_1 \\ f_3 \end{bmatrix} \qquad \left.\begin{aligned} 2u_1 - u_2 &= 4 \\ \tfrac{3}{2}u_2 - u_3 &= 2 \\ -u_2 + 2u_3 &= 0 \end{aligned}\right\}$$

En círculo tenemos los "pivots"

Finalmente, para eliminar u_2 de la última ecuación, sumamos a la tercera ecuación la segunda multiplicada por $\frac{2}{3}$ (en términos de operaciones elementales $F_3 \to F_3 + \frac{2}{3}F_2$)

$$\begin{bmatrix} \textcircled{2} & -1 & 0 \\ 0 & \textcircled{\frac{3}{2}} & -1 \\ 0 & 0 & \textcircled{\frac{4}{3}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + \frac{1}{2}f_1 \\ f_3 + \frac{2}{3}f_2 + \frac{1}{3}f_3 \end{bmatrix} \qquad \left.\begin{aligned} 2u_1 - u_2 &= 4 \\ \tfrac{3}{2}u_2 - u_3 &= 2 \\ \tfrac{4}{3}u_3 &= \tfrac{4}{3} \end{aligned}\right\}$$

Resolviendo el sistema triangular hacia arriba se tiene:

$$\boxed{u_3 = 1}, \quad \boxed{u_2 = \tfrac{2}{3}\cdot 3 = 2}, \quad 2u_1 = 4 - 2 = 2 \to \boxed{u_1 = 1}$$

Mediante el proceso anterior hemos obtenido la matriz triangular superior que andábamos buscando

$$U = \begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & 0 & \frac{4}{3} \end{bmatrix}$$

que contiene a los pivots de la eliminación Gaussiana en su diagonal principal.

¿Dónde está L, la matriz triangular inferior con 1s en la diagonal principal?

En la eliminación Gaussiana, además de los pivots, juegan un papel fundamental los llamados "multiplicadores". En efecto, una vez conocemos el pivot en la fila j y la entrada que se desea

eliminar en la fila i, el multiplicador l_{ij} se define como

$$l_{ij} = \frac{\text{entrada a eliminar (en la fila } i\text{)}}{\text{pivot (en la fila } j\text{)}}.$$

El convenio es "restar", no sumar, l_{ij} veces una ecuación de la otra.

Así, en el ejemplo anterior

$$l_{21} = -\frac{1}{2}, \quad l_{31} = 0, \quad l_{32} = -\frac{2}{3}.$$

La matriz L es justo la matriz de multiplicadores en la eliminación Gaussiana, es decir,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{2}{3} & 1 \end{bmatrix}$$

de modo que

$$K = LU \qquad \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{2}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & 0 & \frac{4}{3} \end{bmatrix}.$$

Nótese que para que la descomposición LU sea posible es esencial que todos los pivots sean no nulos. De forma más precisa, si una columna tiene ceros en la posición del pivot y por debajo de él, entonces la matriz es singular (no invertible) y la descomposición LU no es posible. Es lo que sucede a las matrices circulantes.

Veamos ahora una forma algorítmica de calcular la factorización LU.

Sean $A = (a_{ij})$, $L = (l_{ij})$, $U = (u_{ij})$

Entonces,

$$a_{ij} = \sum_{k=1}^{n} l_{ik} u_{kj}$$

Como $l_{ik} = 0$ si $k > i$ y $u_{kj} = 0$ si $k > j$,

entonces podemos distinguir dos casos:

Caso 1: $i > j$ $\quad \min\{i, j\} = j$

$$a_{ij} = \sum_{k=1}^{j} l_{ik} u_{kj} = l_{i1} u_{1j} + l_{i2} u_{2j} + \dots + l_{ij} u_{jj}$$

y así $\quad l_{ij} = \dfrac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}}{u_{jj}}$

Caso 2: $i \leq j$ $\quad \min\{i, j\} = i$

$$a_{ij} = \sum_{k=1}^{i} l_{ik} u_{kj} = l_{i1} u_{1j} + l_{i2} u_{2j} + \dots + l_{ii} u_{ij}$$

y así $\quad u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$, ya que $l_{ii} = 1$ por definición.

Veamos cómo funciona este método en el caso de la matriz $K = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$

$u_{11} = a_{11} = 2.$

$u_{12} = a_{12} = -1$

$u_{13} = a_{13} = 0$

} primera fila de U

$l_{21} = \frac{a_{21}}{u_{11}} = -\frac{1}{2}$

$l_{31} = \frac{a_{31}}{u_{11}} = \frac{0}{2} = 0$

} primera columna de L

$u_{22} = a_{22} - (l_{21} u_{12}) = 2 - \left(-\frac{1}{2} \cdot (-1)\right) = \frac{3}{2}$

$u_{23} = a_{23} - (l_{21} u_{13}) = -1 - \left(-\frac{1}{2} \cdot 0\right) = -1$

} 2ª fila de U

$l_{32} = \frac{a_{32} - (l_{31} u_{12})}{u_{22}} = \frac{-1}{\frac{3}{2}} = -\frac{2}{3}$ 4 2ª columna de L.

$u_{33} = a_{33} - (l_{31} u_{13} + l_{32} u_{23}) = 2 - \left(0 \cdot 0 + \left(-\frac{2}{3}\right) \cdot (-1)\right)$

$= 2 - \frac{2}{3} = \frac{4}{3}$

Nótese que el orden de cálculo es:

fila de $U \rightarrow$ columna de L

y así sucesivamente.

Se puede probar que el coste computacional total, es decir, del cálculo de L y U, y resolver los sistemas triangulares asociados, es $\frac{n^3+3n^2-1}{3}$.

Nótese también que en el proceso anterior no hemos usado la propiedad de simetría de la matriz K. El carácter simétrico de K se pierde en la descomposición (o factorización) LU. En efecto:

$$K = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ 0 & -\frac{2}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ & \frac{3}{2} & -1 \\ & & \frac{4}{3} \end{bmatrix} = LU$$

La simetría de K se puede recuperar separando los pivots (entradas en la diagonal de U) en una matriz diagonal D y dividiendo cada fila de U por su correspondiente pivot. Se obtiene así:

$$K = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ 0 & -\frac{2}{3} & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & & \\ & \frac{3}{2} & \\ & & \frac{4}{3} \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ & 1 & -\frac{2}{3} \\ & & 1 \end{bmatrix}}_{L^T}$$

Además, D se puede ~~descom~~ factorizar como

$$\begin{bmatrix} 2 & & \\ & \frac{3}{2} & \\ & & \frac{4}{3} \end{bmatrix} = \begin{bmatrix} \sqrt{2} & & \\ & \sqrt{\frac{3}{2}} & \\ & & \sqrt{\frac{4}{3}} \end{bmatrix} \begin{bmatrix} \sqrt{2} & & \\ & \sqrt{\frac{3}{2}} & \\ & & \sqrt{\frac{4}{3}} \end{bmatrix}$$

donde simplemente hemos calculado la raíz cuadrada de los pivots. Finalmente,

$$\underbrace{\begin{bmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ 0 & -\frac{2}{3} & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \sqrt{2} & & \\ & \sqrt{\frac{3}{2}} & \\ & & \sqrt{\frac{4}{3}} \end{bmatrix}}_{\sqrt{D}} = \begin{bmatrix} \sqrt{2} & & \\ -\frac{\sqrt{2}}{2} & \sqrt{\frac{3}{2}} & \\ 0 & -\frac{2}{3}\cdot\sqrt{\frac{3}{2}} & \sqrt{\frac{4}{3}} \end{bmatrix}$$

L $\sqrt{D}$ columnas de $L \times \sqrt{\text{pivots}}$

$=$

A

y

$$\underbrace{\begin{bmatrix} \sqrt{2} & & \\ & \sqrt{\frac{3}{2}} & \\ & & \sqrt{\frac{4}{3}} \end{bmatrix}}_{\sqrt{D}} \underbrace{\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ & 1 & -\frac{2}{3} \\ & & 1 \end{bmatrix}}_{L^T} = \underbrace{\begin{bmatrix} \sqrt{2} & -\frac{\sqrt{2}}{2} & 0 \\ & \sqrt{\frac{3}{2}} & -\frac{2}{3}\sqrt{\frac{3}{2}} \\ & & \sqrt{\frac{4}{3}} \end{bmatrix}}$$

$\sqrt{D}$ L^T $\sqrt{\text{pivots}} \times$ filas de L^T

$=$

A^T

Se llega así a la factorización

$$K = A\,A^T$$

llamada factorización de Cholesky

Para el caso general de una matriz simétrica y definida positiva $A = (a_{ij})$, procediendo como en el caso LU, se obtiene, de forma algorítmica, la descomposición de Cholesky

$$A = L L^T$$

donde

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}$$

$$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} \frac{l_{ik} l_{jk}}{l_{jj}}\right), \quad i = j+1, \ldots, n.$$

Finalmente, para obtener la solución del sistema $Ax = b$, se resolverían los sistemas triangulares

$$Ly = b, \quad L^T x = y,$$

con un coste computacional del orden $\frac{n^3}{3}$.

ANÁLISIS DE ERRORES

A continuación analizaremos la cuestión de la estabilidad en la resolución de un sistema lineal $Ax = b$.

De forma más precisa, supongamos que b cambia por Δb, debido, por ejemplo a errores de redondeo. La solución cambiará en Δx, es decir la entrada real es $b + \Delta b$ y la salida $x + \Delta x$.

¿Cómo podemos estimar o controlar Δx en función de Δb?

Obviamente, la matriz A ha de jugar un papel esencial. Como pretendemos medir errores que involucran a vectores y matrices hemos de fijar un criterio de medida. Para el caso de vectores, usaremos, en general, la norma euclídea:

$$x = (x_1, \dots, x_n), \qquad \|x\| = \sqrt{x_1^2 + \dots + x_n^2},$$

aunque también son posibles otras normas, como

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

Para el caso de matrices A, también es posible definir varias normas. En principio, usaremos la norma

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

~~Dado que~~

obviamente, se tiene que si $x \neq 0$, entonces

$$\frac{\|Ax\|}{\|x\|} \leq \|A\|$$

y así, $\|Ax\| \leq \|A\| \, \|x\|$.

Para el caso de matrices invertibles (cuyos sistemas lineales asociados tienen una única solución), ~~el parámetro~~ asociamos un parámetro, llamado <u>número de condicionamiento</u> de la matriz A, que se define como

$$c(A) := \|A\| \, \|A^{-1}\|,$$

que va a jugar un papel esencial en el control/estimación de los errores en la resolución del sistema $Ax = b$.

<u>Nota</u>.- Como se puede intuir, no es fácil calcular el número de condicionamiento de una matriz. Afortunadamente, para matrices definidas positivas K, que aparecen muy frecuentemente en Ingeniería, sí que es "relativamente" fácil calcularlo. En efecto, como recordamos del Álgebra Lineal (tema de diagonalización de matrices),

las matrices simétricas y definidas positivas se pueden diagonalizar en la forma

$$K = Q D Q^T$$

con D una matrit diagonal que contiene en su diagonal principal los autovalores de K, y Q una matrit ortogonal que contiene en sus columnas los autovectores de K, que resultan ser ortonormales. Por tanto, Q y Q^T no cambian la longitud de un vector, es decir,

$$\| Qx \| = \| Q^T x \| = \| x \|.$$

Como consecuencia, $\| K \| = \lambda_{max}(K)$ su autovalor más grande.

De manera análoga, $\| K^{-1} \| = \lambda_{min}(K)$, su autovalor más pequeño.

De esta forma,

$$c(K) = \frac{\lambda_{max}(K)}{\lambda_{min}(K)}.$$

Nota.- Veremos más adelante cómo calcular los valores propios de una matrit grande.

Retomemos el problema del análisis de errores.

Sea $Ax=b$ un sistema compatible determinado con solución única $x = A^{-1}b$.

Si se perturba el término independiente b por Δb, la nueva solución $x+\Delta x$ verifica

$$A(x+\Delta x) = b+\Delta b$$

de modo que el error Δx es solución de

$$A(\Delta x) = \Delta b,$$

es decir, $\Delta x = A^{-1}(\Delta b)$

y así

$$\|\Delta x\| = \|A^{-1}(\Delta b)\| \leq \|A^{-1}\|\,\|\Delta b\|,$$

lo que nos da una cota del "error absoluto" Δx.

Más importante que el error absoluto, es el relativo, el cual estimamos a continuación:

Dado que $Ax=b$, entonces $\|Ax\| = \|b\|$. Así,

$$\|\Delta x\| \leq \|A^{-1}\|\,\|\Delta b\| \cdot \frac{\|Ax\|}{\|b\|} \leq \|A^{-1}\|\,\|A\|\,\|x\|\,\frac{\|\Delta b\|}{\|b\|}$$

de donde

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A\|\,\|A^{-1}\|\,\frac{\|\Delta b\|}{\|b\|} = c(A)\,\frac{\|\Delta b\|}{\|b\|}$$

es decir, el error relativo en x es menor o igual que $c(A)$ veces el error relativo en b.

Si $c(A)$ es grande, se dice que la matriz A está mal condicionada y la solución x puede ser muy sensible con respecto a pequeños cambios en el término independiente b.

La regla práctica a tener en cuenta es que para el sistema $Ax = b$, el ordenador pierde $\log(c(A))$ decimales en errores de redondeo.

Nota.- Para matrices no definidas positivas A e invertibles, se puede comprobar que

$$c(A) = \frac{\sigma_{max}}{\sigma_{min}}$$

donde σ^2_{max}, σ^2_{min} son los autovalores más grande y más pequeño de la matriz simétrica $A \cdot A^T$.

Ejemplo

Sean $A = \begin{bmatrix} 1 & 7 \\ 0 & 1 \end{bmatrix}$ y $b = \begin{bmatrix} 7 \\ 1 \end{bmatrix}$.

La solución del sistema $Ax = b$ es

$$\begin{bmatrix} 1 & 7 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 1 \end{bmatrix} \Rightarrow \begin{cases} x_1 + 7x_2 = 7 \\ x_2 = 1 \end{cases}$$

$$\boxed{x_1 = 0 \,,\; x_2 = 1}$$

Supongamos que b se perturba por $\Delta b = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$.

Calculamos el error en x resolviendo el sistema

$$A(\Delta x) = \Delta b \,;$$

$$\begin{bmatrix} 1 & 7 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

$$\begin{cases} \Delta x_1 + 7\Delta x_2 = 0 \\ \Delta x_2 = 0.1 \end{cases} \quad \Delta x_1 = -7 \times 0.1 = -0.7.$$

El error relativo en x es:

$$\frac{\|\Delta x\|}{\|x\|} = \frac{\sqrt{(-0.7)^2 + (0.1)^2}}{\sqrt{0^2 + 1^2}} = 0.1\sqrt{50}$$

Comparado con el error relativo en b

$$\frac{\|\Delta b\|}{\|b\|} = \frac{\sqrt{0^2 + 0.1^2}}{\sqrt{7^2 + 1^2}} = \frac{0.1}{\sqrt{50}}$$

Por tanto,

$\frac{\|\Delta x\|}{\|x\|} = 0.1\sqrt{50}$ es 50 veces más grande que $\frac{\|\Delta b\|}{\|b\|} = \frac{0.1}{\sqrt{50}}$.

Vamos a calcular el número de condicionamiento de A.

Recordemos $c(A) = \frac{\sigma_{max}}{\sigma_{min}}$

con σ_{max}^2, σ_{min}^2 los autovalores más grande y pequeño de AA^T.

$$A \cdot A^T = \begin{bmatrix} 1 & 7 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 7 & 1 \end{bmatrix} = \begin{bmatrix} 50 & 7 \\ 7 & 1 \end{bmatrix}$$

$$|A \cdot A^T - \lambda I_2| = \begin{vmatrix} 50-\lambda & 7 \\ 7 & 1-\lambda \end{vmatrix} = (50-\lambda)(1-\lambda) - 49$$

$$= 50 - 50\lambda - \lambda + \lambda^2 - 49$$

$$= \lambda^2 - 51\lambda + 1 = 0$$

$$\lambda = \frac{51 \pm \sqrt{51^2 - 4}}{2} = \frac{51 \pm 50.96}{2} = \begin{cases} 50.98 \\ 0.02. \end{cases}$$

Por tanto $\sigma_{max} = \sqrt{50.98} \simeq 7.14$

$$\sigma_{min} = \sqrt{0.02} \simeq 0.14$$

$$c(A) \simeq \frac{7.14}{0.14} = 51.$$

En resumen, se cumple:

$$\frac{\|\Delta x\|}{\|x\|} = 50 \frac{\|\Delta b\|}{\|b\|} \leq \underbrace{51}_{c(A).} \frac{\|\Delta b\|}{\|b\|}$$

3. MÉTODOS ITERATIVOS

Consideremos el caso en que la matriz A del sistema $Ax=b$, es demasiado grande o no es simétrica ni definida positiva, de modo que los métodos directos (LU, Cholesky, etc.) son demasiado caros computacionalmente.

La idea básica de los métodos iterativos (que estudiamos a continuación) consiste en introducir una matriz P, llamada Precondicionador, que sea próxima a la matriz A del sistema, pero que sea mucho más manejable a la hora de operar con ella. Veamos los detalles:

Dado que $Ax=b$, para toda matriz P se tiene

$$Px = (P-A)x + b.$$

Esta descomposición (splitting, en inglés) sugiere un algoritmo iterativo en el que a partir de un x_k calculamos x_{k+1} siguiendo el método

$$\boxed{P x_{k+1} = (P-A) x_k + b.}$$

El éxito de este método iterativo está garantizado siempre que el precondicionador P cumpla:

1) x_{k+1} se calcula rápidamente a partir de x_k, es decir, la ecuación

$$P x_{k+1} = (P - A) x_k + b$$

se resuelve fácilmente.

2) Los errores $e_k = x - x_k$ se aproximan a cero rápidamente.

Si a la ecuación $P x_{k+1} = (P - A) x_k + b$ restamos

$$P x = (P - A) x + b$$

se tiene:

$$P e_{k+1} = (P - A) e_k$$

de donde (suponiendo P invertible):

$$e_{k+1} = \underbrace{(I - P^{-1}A)}_{M} e_k$$

y así

$$\begin{aligned} \| e_{k+1} \| &\leq \| M \| \, \| e_k \| \\ &\leq \| M \| \cdot \| M \| \, \| e_{k-1} \| \\ &\leq \cdots \\ &\leq \| M \|^{k+1} \cdot \| e_0 \| \end{aligned}$$

lo que significa que $\|e_{k+1}\| \xrightarrow[k\to+\infty]{} 0$ siempre

que $\|M\| < 1$, que es el test de convergencia

de los métodos iterativos.

Conviene, en este punto, introducir el concepto de

"radio espectral" de una matriz.

Definición (Radio espectral).

Se define el radio espectral de la matriz M como

$$\rho(M) = \max\{\,|\lambda_i| \,;\, \lambda_i \text{ autovalor de } M\}$$

es decir, el radio espectral de una matriz es el

mayor (en valor absoluto) de sus autovalores.

Una propiedad muy interesante (que no demostramos)

del radio espectral es que siempre es menor que la

norma de la matriz, es decir, (o igual)

$$\boxed{\rho(M) \leq \|M\|}$$

Dado que es más fácil calcular el radio espectral

que la norma, el "test de convergencia" de los métodos

iterativos es

$$\boxed{\rho(M) = \rho(I - P^{-1}A) < 1}$$

La pregunta del millón sigue siendo:

¿Cómo elegir P?

Jacobi propuso tomar $P = D$, la parte diagonal de A, es decir, P es la matriz que contiene a la diagonal de A y el resto ceros.
Es el llamado <u>método iterativo de Jacobi</u>.
Veamos un ejemplo concreto. Consideremos la matriz

$$A = K_4 = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Se tiene: $P = \begin{bmatrix} 2 & & & \\ & 2 & & \\ & & 2 & \\ & & & 2 \end{bmatrix} = 2\, I_4$

$$M = I_4 - P^{-1} A = I_4 - \frac{1}{2} K_4$$

$$= \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

De esta forma, el sistema $Ax = b$ queda

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

$$\left.\begin{aligned} 2x_1 - x_2 &= b_1 \\ -x_1 + 2x_2 - x_3 &= b_2 \\ -x_2 + 2x_3 - x_4 &= b_3 \\ -x_3 + 2x_4 &= b_4 \end{aligned}\right\}$$

y el algoritmo de Jacobi:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}^{new} = \frac{1}{2} \begin{bmatrix} x_2 \\ x_1 + x_3 \\ x_2 + x_4 \\ x_3 \end{bmatrix}^{old} + \frac{1}{2} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

Como se puede apreciar, Jacobi decidió tomar P una matriz fácil para calcular, pero que puede estar muy alejada de A, pues P sólo contiene la información de la diagonal de A.

En el lado opuesto, podemos tomar como P una matriz más complicada para calcular, pero más próxima a A. Esta es la idea de <u>Gauss-Seidel</u>

donde se toma como P la parte triangular inferior de A, es decir, si $A = D + L + U$,

$$D = \begin{bmatrix} a_{11} & & & \\ & a_{22} & 0 & \\ & 0 & \ddots & \\ & & & a_{nn} \end{bmatrix}, \quad L = \begin{bmatrix} 0 & & & \\ a_{21} & 0 & & 0 \\ a_{31} & a_{32} & \ddots & \\ \vdots & \vdots & & \\ a_{n1} & a_{n2} & \cdots & a_{n\,n-1} \; 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & & \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix},$$

entonces $P = D + L$

y el llamado <u>algoritmo de "Gauss-Seidel"</u> queda como

$$\boxed{P\,x_{k+1} = -U x_k + b, \qquad P = D + L}$$

<u>Ejemplo:</u> Aplicado al caso de la matriz $A = K_4$ se tiene:

$$P = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \quad -U = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

y el algoritmo queda:

$$-1\begin{bmatrix}0\\x_1\\x_2\\x_3\end{bmatrix}^{new} + 2\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}^{new} = \begin{bmatrix}x_2\\x_3\\x_4\\0\end{bmatrix}^{old} + \begin{bmatrix}b_1\\b_2\\b_3\\b_4\end{bmatrix}$$

Este sistema se resuelve de la siguiente manera:
De la primera ecuación se calcula x_1^{new}, el cual introducimos en la segunda ecuación para calcular x_2^{new} y así sucesivamente.

Retomando la cuestión de la convergencia de los métodos de Jacobi y Gauss-Seidel, recordemos que la condición $\rho(I - P^{-1}A) < 1$, sobre el radio espectral, garantiza la convergencia de dichos métodos. Aún así, para matrices de gran tamaño, resulta complicado calcular o estimar dichos radios espectrales. Afortunadamente, tenemos una condición mucho más fácil de chequear en la práctica que nos garantiza dicha convergencia para un caso particular de matrices, llamadas "estrictamente diagonal dominantes"

Definición: Una matriz A se dice estrictamente diagonal dominante si para cada fila i se cumple

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|.$$

Ejemplo.

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

$|a_{11}| = 2 > 1$

$|a_{22}| = 4 > 1+2$

$|a_{33}| = 3 > 2.$

TEOREMA Supongamos que la matriz A del sistema $Ax = b$ es estrictamente diagonal dominante. Entonces los métodos de Jacobi y Gauss-Seidel son convergentes, siendo los radios espectrales de sus respectivas matrices ~~de~~ $M = I - P^{-1}A$, menores o iguales que

$$c = \max_{1\leq i\leq n} \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} \right\} < 1.$$

Una última cuestión que surje es sobre la elección entre Jacobi y Gauss-Seidel.

Ambos métodos tienen sus ventajas e inconvenientes.

Como norma general, Gauss-Seidel suele ser más rápido (menos iteraciones) que Jacobi, aunque en las primeras iteraciones Jacobi suele ser muy eficaz y barato por lo que, quizás, una combinación adecuada de ambos métodos sea la quizás óptima.

3. CÁLCULO NUMÉRICO DE AUTOVALORES Y AUTOVECTORES

Repaso de conceptos básicos

- Dada una matriz cuadrada A, se dice que λ es un autovalor (o valor propio) de A si existe un vector no nulo x tal que

$$Ax = \lambda x.$$

- Dos matrices cuadradas A y B se dicen semejantes si existe una matriz invertible P tal que

$$B = P^{-1} A P.$$

Si A y B son semejantes, sus valores propios son iguales. Sus vectores propios cumplen

$$x(B) = P^{-1} x(A)$$

es decir los vectores propios de B se calculan a partir de los de A multiplicando por P^{-1}.

- Las matrices simétricas tienen todos sus autovalores reales. Además son diagonalizables y existe una matriz ortogonal Q (cuyas columnas son vectores ortonormales) de modo que

$$A = Q D Q^T$$

con D una matriz diagonal que contiene los autovalores de A.

El interés en Ingeniería en el cálculo de autovalores es enorme. Por ejemplo, en Elasticidad los valores propios del tensor de deformaciones $\varepsilon = (\varepsilon_{ij})$ nos dan las deformaciones principales de un sólido y sus autovectores, las direcciones principales de deformación. En problemas continuos (modelados por ecuaciones en derivadas parciales) los valores propios del problema resultante de aplicar el método de separación de variables nos proporcionan, para problemas vibratorias, los modos de vibración naturales de una estructura, que conviene conocer con el fin de evitar fenómenos de resonancia.
En las nuevas tecnologías, Google ordena las páginas web de una búsqueda atendiendo al autovalor / autovector dominante en una determinada matriz, de tamaño enorme.
La siguiente pregunta es, por tanto, pertinente:
¿Cómo calcular autovalores y autovectores de matrices grandes?

Recordemos cómo hacerlo de manera analítica.

Los autovalores de una matriz A son las raíces de su polinomio característico asociado

$$P_A(\lambda) = \det(A - \lambda I).$$

Para cada autovalor λ, su(s) vector(es) propio(s) asociados se calculan resolviendo el sistema compatible indeterminado

$$(A - \lambda I)x = 0.$$

Como se puede observar, el grado del polinomio $P_A(\lambda)$ coincide con el tamaño de la matriz A.

Por tanto, el método analítico anterior es completamente ineficaz en el caso de matrices A de tamaño grande, salvo que A sea una matriz triangular

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ & & \ddots & & \\ 0 & & & & a_{nn} \end{bmatrix}$$

en cuyo caso los autovalores son las entradas a_{ii} de la diagonal principal.

o bien si A la matriz es triangular a bloques y de la forma

$$T = \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1n} \\ & \ddots & B_{22} & \cdots & B_{2n} \\ & & \ddots & & \\ 0 & & & & B_{nn} \end{bmatrix}$$

donde B_{ij} es 2×2 ó (1×1), caso triangular anterior. La matriz T se denomina de Schur y se cumple

$$\det(T-\lambda I) = \det(B_{11}-\lambda I)\det(B_{22}-\lambda I)\cdots \det(B_{nn}-\lambda I)$$

con lo que nos aparece un producto de polinomios de grado a lo sumo 2, que permiten calcular sus ceros (autovalores de T) fácilmente.

Veamos a continuación dos métodos numéricos para calcular autovalores / autovectores de matrices grandes. Como de costumbre, distinguiremos los casos de matrices simétricas y definidas positivas, y luego el caso general.

· Método de iteración Cholesky.

Consideremos el prototipo de matriz simetrica y definida positiva

$$K_4 = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

que introducimos en Octave / MatLab como:

```
K=[2 -1 0 0;-1 2 -1 0;0 -1 2 -1;0 0 -1 2]
```

A continuación calculamos su factorización Cholesky

$$K = K_0 = L\,L^T$$

con L una matriz triangular inferior.

· Escribimos en la línea de comandos de Octave / MatLab

```
L = chol(K, 'lower')
```

· A continuación actualizamos la matriz K en la forma

$K_1 = L' * L$ donde $L' = L^T$ es la traspuesta de L.

observamos que K_0 y K_1 son semejantes. En efecto

$$K_1 = L^{-1} * \underbrace{L * L'}_{K_0} * L = L^{-1} K_0 L$$

con lo que las matrices K_0 y K_1 tienen los mismos autovalores.

Tras repetir este proceso varias veces observamos que llegamos a una matriz K_n, que tiene los mismos autovalor que la original K, pero que es ¡triangular! . Por tanto, sus autovalores son las entradas $(K_n)_{ii}$ de la diagonal principal y nuestro problema está resuelto!

En resumen: aplicamos el método iterativo de Cholesky a matrices simétricas y definidas positivas, y procedemos del siguiente modo:

1) Inicialización: $A_0 = A$

2) Calculamos la factorización de Cholesky de A_0

$$A_0 = L_0 L_0^T$$

3) Actualizamos A en la forma

$$A_1 = L_0^T L_0$$

4) Volvemos al paso 2)

Repetimos el proceso hasta obtener una matriz A_n triangular cuyos valores propios son las

entradas $(A_n)_{ii}$ de la diagonal principal.
Este algoritmo iterativo está implementado en la función "eig" de Octave / MatLab.

Este algoritmo necesita de la factorización Cholesky, donde el carácter simétrico y definido positivo de A, es esencial. Para matrices no simétricas o no definidas positivas, ~~el~~ la función "eig" tiene implementado el llamado algoritmo "qz", que es una versión mejorada del famoso algoritmo de iteración QR, inventado por Francis y Kublanovskaya en 1961.
Como su nombre indica, este método se basa en la factorización $A = QR$, donde A es una matriz cuyas columnas a_i son vectores linealmente independientes, Q es una matriz ortogonal con columnas q_i formados por vectores ortonormales, y R es una matriz triangular superior.

La factorización QR se obtiene como consecuencia del método de ortogonalización de Gram-Schmidt (estudiado en primer curso) que permite obtener una base ortonormal $q_1, \dots, q_n$, columnas de Q, a partir de una base cualquiera $a_1, \dots, a_n$, columnas de la matriz original A.

Veamos un ejemplo concreto:

$$A = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} \quad a_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad a_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

· Método de Gram-Schmidt:

1º) $q_1 = \dfrac{a_1}{\|a_1\|} = \dfrac{(2,0)}{2} = (1,0)$

$r_{11} = \|a_1\| = 2$

$r_{12} = a_{12} = 1$

2º) $q_2' = a_2 + \alpha q_1$.

Pedimos que $q_2' \cdot q_1 = 0$ (ortogonales)

$0 = q_2' \cdot q_1 = a_2 \cdot q_1 + \alpha \underbrace{q_1 \cdot q_1}_{1} \Rightarrow \alpha = -a_2 \cdot q_1 =$

$= -(1,2) \cdot (1,0) = -1$

$q_2' = a_2 + \alpha q_1 = (1,2) - (1,0) = (0,2)$

$q_2 = \dfrac{q_2'}{\|q_2\|} = (0,1)$, $r_{22} = \|q_2\| = 2$.

Por tanto:

$$\underbrace{\begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}}_{R}$$

Veamos cómo podemos usar esta factorización para el cálculo de autovalores.

Introduzcamos en Octave / MatLab la matriz

$$A = \begin{bmatrix} 13 & 5 & 2 & 2 \\ 3 & 11 & 6 & 6 \\ 2 & 2 & 8 & 4 \\ 2 & 2 & 4 & 8 \end{bmatrix}$$

y a continuación repetimos varias veces las sentencias:

`[Q,R] = qr(A)` , que calcula la factorización QR de A

`A = R * Q` que actualiza A multiplicando, en este orden, R por Q.

Observamos que las matrices A_n así obtenidas se aproximan a una matriz triangular que contiene en su diagonal los números 20, 8, 8, 4, que son los autovalores de A.

Algoritmo iteración QR

1º) Tomar $A_0 = A$

2º) obtener la factorización $A_0 = Q_0 R_0$

3º) Actualización de A:

$$A_1 = R_0 Q_0.$$

4º) Volver a 2º).

Notas

- A_1 es equivalente a A_0; por tanto tiene sus mismos autovalores. En efecto:

$$Q_0^{-1} A_0 Q_0 = Q_0^{-1} Q_0 R_0 Q_0 = R_0 Q_0 = A_1$$

- La matriz a la que converge el algoritmo QR no tiene por qué ser triangular superior pero sí es una matriz de Schur.
- En ocasiones, el algoritmo QR no converge.