Reporte de laboratorio
Análisis numérico II

Alumno: Octavio Alberto Agustín Aquino
Profesora: M. C. Graciela Castro
Grupo: 605
Licenciatura en Matemáticas Aplicadas
Universidad Tecnológica de la Mixteca

5 de junio de 2005

Índice

1. Planteamiento

1.1. Introducción

Existen varios métodos para calcular los autovalores de una matriz; algunos son generales y otros se restringen a ciertas matrices especiales, generalmente simétricas. En la siguiente práctica se abordan algunos métodos númericos para

hallar los autovalores reales de una matriz real, a saber: los métodos de la potencia, la iteración QR y el método de Jacobi. Todos ellos son de carácter iterativo. Los métodos se implementaron en MATLAB, y se reproduce su código fuente.

1.2. Objetivo

Conocer e implementar en MATLAB el método de la potencia, la iteración QR y el método de Jacobi para el cálculo de los autovalores reales de una matriz.

2. Descripción de los métodos

2.1. Métodos de la potencia

Los métodos de la potencia son directos y sencillos de implementar, y además pueden servir para inicializar otros métodos más complejos. Tienen la desventaja de que sólo calculan un autovalor. En este caso sólo consideraremos el método de la potencia regular y el inverso.

2.1.1. Método de la potencia regular

Consideremos una matriz $A \in \mathcal{M}_{n\times n}$. Denotaremos al conjunto de sus autovalores por $\text{spec}(A) = \{\lambda_1, \ldots, \lambda_n\}$. Supongamos que el valor propio más grande en valor absoluto es real y aislado (es decir, que no es valor propio doble de A). Supongamos, además, que los valores propios están ordenados en orden creciente de valor absoluto,

$$|\lambda_1| \leq |\lambda_2| \leq \cdots < |\lambda_n|.$$

El método de la potencia se inicia con una estimación inicial del vector propio $u^{(0)}$, que puede ser cualquier vector no nulo con $\|u\| = 1$. La primera iteración es entonces

$$u^{(1)} = Au^{(0)}$$

y las siguientes iteraciones son

$$u^{(k+1)} = \frac{1}{\lambda_k} Au^{(k)} \tag{1}$$

donde

$$\lambda^{(0)} = 1, \lambda^{(k)} = \left\| Au^{(k)} \right\|.$$

Al continuar la iteración tenemos que $\lambda_k \to \lambda_n$ y $u^{(k)}$ tiende a al vector propio correspondiente. Ahora veamos que el método de potencias converge. El vector inicial puede escribirse en términos de la base que forman los valores propios de A,

$$u^{(0)} = \sum_{i=1}^{n} a_i u_i. \tag{2}$$

Consierando las ecuaciones (1) y (2) vemos que la k-ésima estimación del vector propio de A es

$$u^{(k)} = \frac{\lambda_n^k}{\lambda^{(1)} \dots \lambda^{(k-1)}} \left[a_1 \left(\frac{\lambda_1}{\lambda_n} \right)^k u_1 + a_2 \left(\frac{\lambda_2}{\lambda_n} \right)^k u_2 + \dots + a_n u_n \right]. \quad (3)$$

Como $\frac{\lambda_i}{\lambda_n} < 1$ para $i = 1, \dots, n-1$ entonces, cuando $k \to \infty$, todos los sumandos de (3) salvo $a_n u_n$ tienden al vector cero. Por el teorema del valor de normas matriciales tenemos

$$0 < \lambda^{(k)} \leq |\lambda_n|,$$

por lo tanto

$$0 < \frac{1}{\lambda^{(k)}} \leq \frac{1}{|\lambda_n|}$$

de donde

$$0 < \frac{\lambda_n^k}{\lambda^{(1)} \dots \lambda^{(k-1)}} \leq 1.$$

Esto indica que $u^{(k)}$ converge a un vector propio asociado al valor propio λ_n. Obsérvese, finalmente, que

$$u^{(k)} = \frac{1}{\prod_{i=1}^{k-1} \lambda^{(i)}} A^{(k)} u^{(0)},$$

es decir, la esencia del método de la potencia es multiplicar la estimación inicial por una potencia de A normalizada adecuadamente; de lo contrario, el método podría diverger.

2.1.2. Método de la potencia inversa

Existe una relación entre el espectro de una matriz invertible A el de su inversa. Sea λ un valor propio de A y v su vector propio asociado. Entonces

$$v = Iv = A^{-1}Av = A^{-1}(\lambda v) = \lambda A^{-1} v,$$

de donde puede deducirse

$$A^{-1}v = \frac{1}{\lambda} v,$$

esto es, $\frac{1}{\lambda}$ es un valor propio de A^{-1}. Supongamos ahora que el valor propio mínimo de A es real, aislado y no nulo:

$$|\lambda_1| < |\lambda_2| \leq \dots \leq |\lambda_n|.$$

Por lo tanto,

$$\frac{1}{|\lambda_n|} \leq \dots \leq \frac{1}{|\lambda_2|} < \frac{1}{|\lambda_1|},$$

lo que implica que si aplicamos el método de la potencia a A^{-1}, obtendremos el valor $\frac{1}{|\lambda_1|}$; y así calculamos el valor de λ_1, el menor valor propio de A.

El primer paso de la iteración es

$$Au^{(1)} = u^{(0)}$$

o, equivalentemente,

$$u^{(1)} = A^{-1}u^{(0)},$$

donde $u^{(0)}$ es un vector no nulo, dado como estimación inicial. Los pasos subsecuentes son

$$Au^{(k+1)} = \frac{1}{\lambda^{(k)}}u^{(k)} \tag{4}$$

con

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

Para resolver el sistema (4) se calcula la factorización QR vía transformaciones de Householder de A:

$$Ax = (QR)x = u^{(k)}.$$

Así,

$$Rx = Q^t u^{(k)}$$

pues $Q^{-1} = Q^t$. Este sistema puede resolverse por sustitución hacia atrás. Finalmente,

$$u^{(k+1)} = \frac{1}{\lambda^{(k)}}x.$$

El método de la potencia inversa converge por las mismas razones que converge el método de la potencia regular.

2.2. Iteración QR

Es una sucesión de transformaciones de similaridad. Cada paso de la iteración consiste en la descomposición de la matriz en la forma QR y en la transformación de similaridad. Denotamos a la matriz inicial por $A_0 = A$, donde A es la matriz original de la cual deseamos calcular los valores. La matriz A_0 se descompone en

$$A_0 = Q_0 R_0$$

donde Q_0 es una matriz ortonormal y R_0 es una matriz triangular superior. La transformación de similaridad se escribe como

$$A_1 = Q_0^{-1}A_0Q_0 = Q_0^{-1}Q_0R_0Q_0 = R_0Q_0.$$

Los siguientes paso son esencialmente identicos y se escriben como

$$\begin{aligned} A_k &= Q_k R_k \\ A_{k+1} &= R_k Q_k. \end{aligned}$$

Nuestro criterio de paro es

$$\|A_{k+1} - A_k\| < \epsilon$$

donde ϵ es la tolerancia. De hecho, la iteración QR y el método de la potencia pueden verse como generalizaciones de un esquema más general. Sea $A \in \mathcal{M}_{n\times n}$, y r un entero que satisface $1 \leq r \leq n$. Dada una matriz $Q_0 \in \mathcal{M}_{n\times r}$ con columnas ortonormales, el método de la iteración ortogonal genera una sucesión de matrices $\{Q_k\} \subseteq \mathbb{C}^{n\times r}$ como sigue

$$\begin{aligned} Z_k &= AQ_{k-1} \\ Q_k R_k &= Z_k. \end{aligned}$$

Nótese que si $r = 1$, entonces tenemos el método de la potencia. Más aún, la sucesión $\{Q_k e_1\}$ es precisamente la sucesión de vectores producida por el método de potencias con el vector inicial $q_0 = Q_0 e_1$.

Para analizar el comportamiento de esta iteración, supongamos que

$$Q^* A Q = T = \operatorname{diag}(\lambda_i) + N \quad |\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$$

es una descomposición de Schur de $A \in \mathbb{C}^{n\times n}$. Supongamos que $1 \leq r \leq n$ y particionemos Q, T y N como sigue

$$\begin{aligned} Q &= \begin{pmatrix} Q_\alpha & Q_\beta \end{pmatrix} \\ T &= \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix} \\ N &= \begin{pmatrix} N_{11} & N_{12} \\ 0 & N_{22} \end{pmatrix}, \end{aligned}$$

donde Q_α, T_{11} y N_{11} tienen r columnas y Q_β, T_{12}, T_{22}, N_{12} y N_{22} tienen $n - r$ columnas. Si $|\lambda_r| > |\lambda_{r+1}|$, entonces el subespacio $D_r(A) = \operatorname{ran}(Q_\alpha)$ se dice que es un subespacio invariante dominante. El siguiente teorema muestra que, bajo suposiciones razonables, los subespacios $\operatorname{ran}(Q_k)$ generados por el esquema iterativo converge a $D_r(A)$ a una tasa proporcional a $\left|\frac{\lambda_{r+1}}{\lambda_r}\right|^k$. No se da su demostración, pues no es el objetivo de la práctica.

Teorema 1. *Sea la descomposición de Schur de $A \in \mathbb{C}^{n\times n}$ con $n \geq 2$. Supóngase $|\lambda_r| > |\lambda_{r+1}|$ y que $\theta \geq 0$ satisface*

$$(1+\theta)\,|\lambda_r| > \|N\|_F\,.$$

Si $Q_0 \in \mathbb{C}^{n\times r}$ tiene columnas ortonormales y

$$\begin{aligned} d &= \operatorname{dist}(D_r(A^*), \operatorname{ran}(Q_0)) < 1, \\ d_k &= \operatorname{dist}(D_r(A^*), \operatorname{ran}(Q_k)), \end{aligned}$$

entonces las matrices Q_k generadas por la iteración ortogonal satisfacen

$$d_k \leq \frac{(1+\theta)^{n-2}}{\sqrt{1-d^2}} \left(1 + \frac{\|T_{12}\|_F}{\operatorname{sep}(T_{11}, T_{22})}\right) \left(\frac{|\lambda_{r+1}| + \|N\|_F/(1+\theta)}{|\lambda_r| - \|N\|_F/(1+\theta)}\right)^k,$$

donde $\operatorname{sep}(T_{11}, T_{22})$ es el valor singular más pequeño de $T_{11}X - XT_{22}$.

Podemos derivar ahora la iteración QR y examinar su convergencia. Tomemos en la iteración ortonormal $r = n$ y los valores propios de A satisfacen

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| .$$

Particionamos la matriz Q y Q_k en el esquema como sigue:

$$Q = (q_1, \ldots, q_n) \quad Q_k = \left(q_1^{(k)}, \ldots, q_n^{(k)}\right)$$

Si

$$\text{dist}(D_i(A^*), \text{gen}\{q_1^{(0)}, \ldots, q_n^{(0)}\}) < 1 \quad i = 1, \ldots, n$$

se sigue, del teorema (1), que

$$\text{dist}(\text{gen}\{q_1^{(k)}, \ldots, q_i^{(k)}\}, \text{gen}\{q_1, \ldots, q_i\}) \to 0$$

para $i = 1, \ldots, n$. Por lo tanto, la iteración QR calcula la descomposición de Schur bajo las hipótesis del teorema, esto es, $Q = \lim_{k\to\infty} Q_1 \cdots Q_k$.

Aunque la iteración QR calcula todo el espectro de una matriz A, es un cómputo $O(n^3)$, donde n es la dimensión de la matriz; su convergencia es lineal, cuando existe, por lo cual para matrices grandes es un método impráctico.

2.3. Iteración de Jacobi

El método de Jacobi permite encontrar todos los autovalores de una matriz simétrica A. Se construye una sucesión de matrices ortogonales $R_1, R_2, \ldots, R_n$ como sigue

$$\begin{aligned} D_0 &= A \\ D_j &= R_j^t D_{j-1} R_j, \end{aligned}$$

para $j = 1, \ldots, n$. Mostraremos cómo construir la sucesión $\{R_j\}$ tal que

$$\begin{aligned} \lim_{j\to\infty} D_j &= D = \text{diag}(\lambda_1, \ldots, \lambda_n), \\ \lim_{j\to\infty} R_j &= R \end{aligned}$$

y además

$$R^t A R = D.$$

En la práctica nos detendremos cuando los elementos distintos de la diagonal sean cercanos a cero. La construcción produce

$$D_n = R_n^t R_{n-1}^t \cdots R_1^t A R_1 R_2 \cdots R_{n-1} R_n.$$

En cada paso de la iteración de Jacobi conseguiremos el limitado objetivo de anular dos elementos d_{pq} y d_{qp} con $p \neq q$. Denotemos R_1a la primera matriz ortogonal utilizada. Supóngase que

$$D_1 = R_1^t A R_1$$

reduce los elementos d_{pq} y d_{qp} a cero, donde R_1 tiene la forma

$$R_1 = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}$$

con $(R_1)_{pp} = (R_1)_{qq} = c$, $(R_1)_{pq} = s$, $(R_1)_{qp} = -s$ y $(R_1)_{ij} = \delta_{ij}$ siempre que $i \neq p, q$ y $j \neq p, q$. Veremos que de hecho

$$\begin{aligned} c &= \cos\phi \\ s &= \operatorname{sen}\phi, \end{aligned}$$

donde ϕ es el ángulo de rotación que produce el efecto deseado. Definamos

$$\theta = \cot 2\phi = \frac{c^2 - s^2}{2cs}.$$

Después de algunos cálculos directos vemos que $d_{pq} = d_{qp} = (c^2 - s^2)a_{pq} + cs(a_{pp} - a_{qq})$, y lo que queremos es

$$(c^2 - s^2)a_{pq} + cs(a_{pp} - a_{qq}) = 0,$$

lo que implica que

$$\frac{c^2 - s^2}{2cs} = \frac{a_{qq} - a_{pp}}{2a_{pq}} = \theta. \tag{5}$$

Para aumentar la estabilidad numérica, calculamos $\tan\phi$. Definimos

$$t = \tan\phi = \frac{s}{c}.$$

Dividimos los numeradores de (5) entre c^2 para obtener

$$\theta = \frac{1 - s^2/c^2}{2s/c} = \frac{1 - t^2}{2t},$$

lo que arroja la ecuación

$$t^2 + 2t\theta - 1 = 0. \tag{6}$$

Puesto que $t = \tan\phi$, la raíz más pequeña (6) corresponde al ángulo de rotación más pequeño con $|\phi| \leq \pi/4$. La forma particular de la fórmula cuadrática en este caso es

$$t = -\theta \pm \sqrt{\theta + 1} = \frac{\operatorname{signo}(\theta)}{|\theta| + \sqrt{\theta + 1}},$$

y así c y s pueden calcularse con las fórmulas

$$\begin{aligned} c &= \frac{1}{\sqrt{t^2+1}} \\ s &= ct. \end{aligned}$$

El esquema de Jacobi continúa anulando siguiendo el órden de las columnas hasta que todos los elementos distintos de la diagonal se acercan a cero tanto como lo indique la tolerancia, así que nuestro criterio de paro es

$$\|D_{j+1} - D_j\| < \epsilon$$

donde ϵ es la tolerancia. Las desventajas del método de Jacobi es que está restringido para matrices simétricas y que su eficiencia se ve afectada por el modo en que se eligen los elementos de la matriz a eliminar.

3. Implementación en MATLAB

En las secciones que vienen, los algoritmos siempre tendrán como supuesto que se tiene una tolerancia *tol* y un número máximo de iteraciones MAX. No se examina el algoritmo para obtener la factorización QR vía transformaciones de Householder, pues no es el aspecto fundamental de los algoritmos expuestos. Por simplicidad, en las implementaciones en MATLAB no se despliegan mensajes de error por división entre cero o exceso de iteraciones.

3.1. Método de la potencia regular

Algoritmo 1 (Potencia regular). *Sea la matriz $A \in \mathcal{M}_{n\times n}$, cuyo valor propio más grande en valor absoluto es real y aislado (es decir, que no es valor propio doble de A). Supongamos, además, que los valores propios están ordenados en orden creciente de valor absoluto,*

$$|\lambda_1| \leq |\lambda_2| \leq \cdots < |\lambda_n|$$

Sea v una estimación inicial del vector propio, que puede ser cualquier vector no nulo con $\|v\| = 1$.

1. *Hacer $i = 1$ y $err = tol + 1$.*
2. *Mientras $i \leq MAX$ y $tol \leq err$, repetir:*
   a) *Calcular la estimación $y \leftarrow Av$.*
   b) *Calcular la estimación del valor propio $\lambda \leftarrow \|y\|$. Si $\lambda = 0$, marcar error e ir al paso .*
   c) *Calcular el error de la estimación $err = \left\| v - \frac{y}{\|y\|} \right\|$.*
   d) *Actualizar la estimación $v \leftarrow \frac{y}{\|y\|}$ y el contador $i \leftarrow i + 1$. Si $i = MAX + 1$, marcar exceso de iteraciones.*

3. *Fin.*

Al terminar el algoritmo, se obtiene o bien una aproximación a λ_n y v, su vector propio con precisión tol; o bien, un mensaje de exceso de iteraciones, o bien, un mensaje de error.

La función `potencia` tiene como parámetros la matriz cuadrada $A \in \mathcal{M}_{n\times n}$, el vector de estimación inicial v y la tolerancia *tol*. Como salida se obtiene *lambda*, el valor propio máximo en valor absoluto y v, el vector propio asociado a dicho autovalor. El algoritmo se repite un máximo de 100 veces.

```
function [lambda,v]=potencia(A,v,tol)
i=1;
err=tol+1;
while (i<=100)&(tol<=err)
    v=v/norm(v,inf);
    y=A*v;
    lambda=norm(y,inf);
    if (lambda==0)
        break;
    end
    vs=y/lambda;
    err=norm(v-vs,inf);
    i=i+1;
    v=vs;
end
```

3.2. Método de la potencia inversa

Antes de implementar el método de la potencia inversa, se requiere un algoritmo auxiliar que resuelve un sistema de ecuaciones $Ax = b$, con $A \in \mathcal{M}_{n\times n}$.

Algoritmo 2 (Sistema de ecuaciones). *Sea la matriz $A \in \mathcal{M}_{n\times n}$ invertible y b un vector de $\mathbb{R}^n$.*

1. *Calcular $Q, R \in \mathcal{M}_{n\times n}$ con Q ortonormal y R triangular superior tales que $A = QR$ a través de transformaciones de Householder.*
2. *Hacer $b' = Q^t b$ y $x = 0$.*
3. *Para i desde $i = n$ hasta $i = 1$, repetir:*
   a) *Hacer $x_i = \frac{\left(b'_i \sum_{j=i+1}^{n} A_{ij} x_{i+1}\right)}{R_{ii}}$.*
4. *Fin.*

Entonces el vector x es la solución del sistema $Ax = b$.

La función `resuelve` toma como parámetros una matriz invertible $A \in \mathcal{M}_{n\times n}$ y un vector b de $\mathbb{R}^n$ y devuelve la solución x del sistema de ecuaciones $Ax = b$.

```
function x=resuelve(A,b)
[Q, R]=factoqr(A);
b1=Q'*b;
x=zeros(length(b),1);
for i=length(b):-1:1
    x(i)=(b1(i)-R(i,i+1:length(b))*x(i+1:length(b)))/R(i,i);
end
```

Algoritmo 3 (Potencia inversa). *Sea la matriz $A \in \mathcal{M}_{n\times n}$, cuyo valor propio mínimo en valor absoluto es real y aislado (es decir, que no es valor propio doble de A). Supongamos, además, que los valores propios están ordenados en orden creciente de valor absoluto,*

$$|\lambda_1| < |\lambda_2| \leq \cdots \leq |\lambda_n|.$$

Sea v una estimación inicial del vector propio, que puede ser cualquier vector no nulo con $\|v\| = 1$.

1. *Hacer $i = 1$ y $err = tol + 1$.*
2. *Mientras $i \leq MAX$ y $tol \leq err$, repetir:*
   a) *Resolver para y la ecuación $Ay = v$ a través del algoritmo (2).*
   b) *Calcular la estimación del valor propio $\lambda \leftarrow \|y\|$. Si $\lambda = 0$, marcar error e ir al paso 3.*
   c) *Calcular el error de la estimación $err = \left\| v - \frac{y}{\|y\|} \right\|$.*
   d) *Actualizar la estimación $v \leftarrow \frac{y}{\|y\|}$ y el contador $i \leftarrow i + 1$.*
3. *Fin.*

Al terminar el algoritmo, se obtiene o bien una aproximación a $\frac{1}{\lambda_n}$ y v, su vector propio con precisión tol; o bien, un mensaje de exceso de iteraciones, o bien, un mensaje de error.

La función `potenciainv` tiene como parámetros la matriz cuadrada A, el vector de estimación inicial v y la tolerancia tol. Como salida se obtiene *lambda*, el valor propio mínimo en valor absoluto y v el vector propio asociado a dicho autovalor. El algoritmo se repite un máximo de 100 veces.

```
function [lambda,v]=potenciainv(A,v,tol)
i=1;
```

```
err=tol+1;
while (i<=100)&(tol<=err)
    v=v/norm(v,inf);
    y=resuelve(A,v);
    lambda=norm(y,inf);
    if (lambda==0)
        error('El valor propio es cero o muy cercano a cero.');
        break;
    end
    vs=y/lambda;
    err=norm(v-vs,inf);
    i=i+1;
    v=vs;
end
lambda=1/lambda;
```

3.3. Iteración QR

Algoritmo 4 (Iteración QR). *Sea la matriz $A \in \mathcal{M}_{n\times n}$, cuyo espectro puede ordenarse del siguiente modo:*

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|\,.$$

Particionamos la matriz Q y Q_k en el esquema como sigue:

$$Q = (q_1, \ldots, q_n) \quad Q_k = \left(q_1^{(k)}, \ldots, q_n^{(k)}\right).$$

Supongamos que

$$\operatorname{dist}(D_i(A^*), \operatorname{gen}\{q_1^{(0)}, \ldots, q_n^{(0)}\}) < 1 \quad i = 1, \ldots, n.$$

1. *Hacer $A_0 \leftarrow A$, $A_1 \leftarrow A - (tol+1)I, Q_f \leftarrow I$, con I la matriz identidad de $n \times n$.*
2. *Mientras $\|A_0 - A_1\| > tol$ e $i \leq MAX$, repetir:*
   a) *Calcular la factorización QR de A_0.*
   b) *Hacer $Q_f \leftarrow Q_f Q$, $A_0 \leftarrow RQ$, $A_1 \leftarrow QR$ e $i \leftarrow i+1$. Si $i = MAX+1$, marcar exceso de iteraciones.*
3. *Hacer $S = A_0$.*
4. *Fin.*

Al terminar el algoritmo, se obtiene o bien una aproximación a la descomposición de Schur $Q_f^t A Q_f = S$, con precisión tol; o bien, un mensaje de exceso de iteraciones.

La función `itQR` tiene como parámetros la matriz cuadrada A y la tolerancia *tol*. Como salida se obtiene Q_f y S de la descomposición de Schur $Q_f^t A Q_f = S$. El algoritmo se repite un máximo de 100 veces.

```
function[S,Qf]=itQR(A,tol)
A0=A;
A1=A+(tol+1)*eye(size(A));
Qf=eye(size(A));
i=1;
while(norm(A1-A0,inf)>tol)&(i<=100)
    [Q,R]=factoqr(A0);
    Qf=Qf*Q;
    A0=R*Q;
    A1=Q*R;
    i=i+1;
end
S=A0;
```

3.4. Iteración de Jacobi

Algoritmo 5 (Iteración de Jacobi). *Sea la matriz $A \in \mathcal{M}_{n\times n}$ simétrica.*

1. *Hacer $R \leftarrow I$, $P \leftarrow A$.*
2. *Mientras $\|P - diag(P)\| > tol$ e $l \leq MAX$, repetir:*
   a) *Para i desde $i = 1$ hasta $i = n$, repetir:*
   b) *Para j desde $j = i + 1$ hasta $i = n$, repetir:*
      1) *Hacer $\theta \leftarrow \frac{A_{qq}-A_{pp}}{2A_{pq}}$, $t \leftarrow \frac{\text{sign}(\theta)}{|\theta|+\sqrt{\theta+1}}$, $c \leftarrow \frac{1}{\sqrt{t^2+1}}$ y $s \leftarrow ct$.*
      2) *Hacer $U \leftarrow I$.*
      3) *Hacer $U_{ii}, U_{jj} \leftarrow c$, $U_{ij} \leftarrow s$ y $U_{ji} \leftarrow -s$.*
      4) *Hacer $P \leftarrow U^t PU$, $R \leftarrow RU$ e $l \leftarrow l+1$. Si $l = MAX+1$, marcar exceso de iteraciones.*
3. *Fin.*

Al terminar el algoritmo, se obtiene o bien una aproximación a la descomposición $R^t AR = \text{diag}(\text{spec}(A))$, con precisión tol; o bien, un mensaje de exceso de iteraciones.

La función `eigJacobi` tiene como parámetros la matriz cuadrada A y la tolerancia *tol*. Como salida se obtiene P y R de la descomposición $R^t AR = P$ donde $P = \text{diag}(\text{spec}(A))$. El algoritmo verifica que A sea simétrica (de no serlo se despliega un mensaje de error) y se repite un máximo de 100 veces.

```
function [P,R]=eigJacobi(A,tol)
if A==A'
    n=length(A(:,1));
    R=eye(n);
    P=A;
    l=1;
    while norm(P-diag(diag(P)))>tol&(l<=100)
        for i=1:n
        for j=i+1:n
         if P(j,j)~=P(i,i)
             theta=(P(j,j)-P(i,i))/(2*P(i,j));
             t=sign(theta)/(abs(theta)+sqrt(theta*theta+1));
             c=1/sqrt(t*t+1);
             s=c*t;
         else
             c=sqrt(2)/2;
             s=c;
         end
            aux=eye(n);
            aux(i,i)=c;
            aux(j,j)=c;
            aux(i,j)=s;
            aux(j,i)=-s;
            P=aux'*P*aux;
            R=R*aux;
        end
    end
   l=l+1;
end
else
   error('La matriz debe ser simetrica.');
end
```

4. Conclusiones y observaciones

- Si bien no existen métodos que sean numéricamente estables, rápidos y que calculen todos los autovalores de una matriz $A \in \mathcal{M}_{n\times n}$, es claro que tenemos que debemos tener en mente las ventajas y desventajas de cada uno de los métodos estudiados al momento de aplicarlos.
- El método de la potencia regular tiene el gran inconveniente de que no podemos saber a priori si una matriz A tiene una valor propio aislado simple y de máximo valor absoluto.
- El método de la potencia inversa tiene el inconveniente de que no converge

si el mínimo valor propio en valor absoluto es 0, que sería el caso de las matrices no invertibles.

- Para la iteración QR es difícil comprobar los casos en los que no converge (que no son demasiado comunes); además, si los valores propios son próximos entre sí, el método se vuelve lento, según el teorema (1).
- Aunque la iteración de Jacobi está enfocada a las matrices simétricas, éstas surgen frecuentemente en la práctica (en las ecuaciones en derivadas parciales, por ejemplo). Además, la transformación de similiaridad que nos devuelve es útil pues es rígida, al ser ortogonal.

Referencias

[1] Golub, Gene *et al. Matrix computations*, tercera edición, Johns Hopkins University Press, Baltimore, 1996.

[2] Mathews, John H. *et al. Numerical Methods using MATLAB*, cuarta edición, Prentice-Hall, New Jersey, 2004.

[3] Nakamura, Shoichiro. *Métodos numéricos aplicados con software*. Prentice-Hall, México, 1992.