Cómputo científico para probabilidad y estadística. Tarea 4. Cálculo de eigenvalores

Juan Esaul González Rangel

Septiembre 2023

1. Dado el siguiente

   Teorema 1 (Gershgorin)**.** *Dada una matriz $A = a_{ij}$ de $m \times m$, cada eigenvalor de A está en al menos uno de los discos en el plano complejo con centro en a_{ii} y radio $\sum_{j \neq i} |a_{ij}|$. Además, si n de estos discos forman un dominio conexo, disjunto de los otros $m - n$ discos, entonces hay exactamente n eigenvalores en ese dominio.*

   Deduce estimaciones de los eigenvalores de

   $$A = \begin{pmatrix} 8 & 1 & 0 \\ 1 & 4 & \varepsilon \\ 0 & \varepsilon & 1 \end{pmatrix}$$

   con $|\varepsilon| < 1$.

   Por el Teorema de Gershgorin, los tres eigenvalores de A se encuentran en el dominio,

   $$\{x \in \mathbb{C} : |8 + x| \leq 1\} \cup \{x \in \mathbb{C} : |4 + x| \leq 1 + |\varepsilon|\} \cup \{x \in \mathbb{C} : |1 + x| \leq |\varepsilon|\}.$$

   Como $|\varepsilon| < 1$, cada disco es disjunto de los otros dos, y además es conexo. Por lo tanto,

   $$\begin{aligned} &\lambda_1 \in \{x \in \mathbb{C} : |8 + x| \leq 1\}, \\ &\lambda_2 \in \{x \in \mathbb{C} : |4 + x| \leq 1 + |\varepsilon|\}, \\ &\lambda_3 \in \{x \in \mathbb{C} : |1 + x| \leq |\varepsilon|\}. \end{aligned}$$

   Notemos que si $\varepsilon \in \mathbb{R}$, entonces A es hermitiana y por lo tanto todos sus valores propios son reales, de donde obtenemos una mejor estimación de los eigenvalores,

   $$\begin{aligned} &\lambda_1 \in \{x \in \mathbb{C} : |8 + x| \leq 1\} \cap \mathbb{R} \qquad \Rightarrow 7 \leq \lambda_1 \leq 9, \\ &\lambda_2 \in \{x \in \mathbb{C} : |4 + x| \leq 1 + |\varepsilon|\} \cap \mathbb{R} \qquad \Rightarrow 4 - (1 + |\varepsilon|) \leq \lambda_2 \leq 4 + (1 + |\varepsilon|), \\ &\lambda_3 \in \{x \in \mathbb{C} : |1 + x| \leq |\varepsilon|\} \cap \mathbb{R} \qquad \Rightarrow 1 - \varepsilon \leq \lambda_1 \leq 1 + \varepsilon. \end{aligned}$$

2. Implementa la iteración QR con shift. Aplícala a la matriz A del Ejercicio 1 con $\varepsilon = 10^{-N}$ para $N = 1, 3, 4, 5$.

   En el archivo `Tarea4.py`, la función se encuentra implementada con el nombre de `QR_shift`. La función toma tres argumentos; `Matrix`, la matriz a la cuál se le aplicará el algoritmo; `shift`, el desplazamiento que se aplicará a la matriz, y `iterations`, la cantidad de veces que se repetirá el algoritmo antes de parar.

   El siguiente es un ejemplo de uso del algoritmo,

```
B = np.array( [[1,2,3],[4,5,6],[7,8,9]] )
vals = QR_shift(B,2,200)
```

El tiempo de ejecución del código anterior es menor a un segundo y la salida es `array([ 1.61168440e+01, -1.11684397e+00, 1.55431223e-15])`.

Para la matriz A, se iteró un total de 10,000 veces por cada valor de ε y se obtuvieron las siguientes estimaciones de los eigenvalores de la matriz,

```
[array([8.23614095, 3.76735469, 0.99650436]),
array([8.23606798, 3.76393237, 0.99999965]),
array([8.23606798, 3.76393203, 1.        ]),
array([8.23606798, 3.76393202, 1.        ])]
```

Es decir, para $N = 1$ los eigenvalores estimados son $(8.23614095, 3.76735469, 0.99650436)$, para $N = 3$, son $(8.23606798, 3.76393237, 0.99999965)$, para $N = 4$ son $(8.23606798, 3.76393203, 1.)$, y para $N = 5$ son $(8.23606798, 3.76393202, 1.)$.

Cuando ejecutamos el código con 10 iteraciones, obtenemos los siguientes valores

```
[array([8.23614095, 3.76735469, 0.99650436]),
array([8.23606798, 3.76393237, 0.99999965]),
array([8.23606798, 3.76393203, 1.        ]),
array([8.23606798, 3.76393202, 1.        ])]
```

Los valores coinciden exactamente con los que obtenemos para 10,000 iteraciones. Concluimos que en este caso la convergencia del algoritmo bastante rápida.

3. Determina todos los eigenvalores y eigenvectores de una matriz de Householder.

   Sea H una matriz de Householder de tamaño $n \times n$, entonces H es de la forma

   $$H = I - 2vv^*,$$

   con $v \in \mathbb{C}^n$ un eigenvector de norma 1. Sea u cualquier vector linealmente independiente a v, entonces

   $$Hu = (I - 2vv^*)u = u - 2v(v^*u) = u - 2v(0) = u.$$

   Por lo tanto, u es un eigenvector de H con eigenvalor 1. Como $\mathbb{C}^n$ tiene dimensión n, existe un total de $n-1$ vectores linealmente independientes entre sí y linealmente independientes a v (pues de otra forma se podría formar una base de $\mathbb{C}^n$ de tamaño distinto a n). El valor 1 entonces es un eigenvalor con multiplicidad $n-1$.

   Ahora, para Hv se cumple,

   $$Hv = (I - 2vv^*)v = v - 2v\|v\| = -v.$$

   Por lo tanto v es un eigenvector con eigenvalor -1. Los eigenvalores de H son 1 con multiplicidad $n-1$ y -1 con multiplicidad 1.

4. Demuestra que no es posible construir la transformación de similaridad del teorema de Schur con un número finito de transformaciones de similaridad de Householder.

   Demostración. Supongamos que existe un algoritmo que nos permita encontrar de manera exacta los eigenvalores de cualquier matriz A con un número finito de transformaciones de similaridad de Householder.

   Sea entonces $p(\lambda)$ cualquier polinomio de orden mayor a 4. Sabemos que podemos encontrar una matriz A acompañante de $p(\lambda)$ tal que los eigenvalores de A son las raíces de $p(\lambda)$. Aplicando nuestro algoritmo, podemos encontrar los eigenvalores de A en una cantidad finita de transformaciones de Householder. Pero las transformaciones de Householder son tranformaciones lineales, por lo que en particular son operaciones algebraicas.

   Lo anterior significa que podemos encontrar las raíces de cualquier polinomio de grado mayor que 4 usando una cantidad finita de operaciones algebraicas, lo que es una contradicción del Teorema de Abel-Rufini.

   □

5. ¿Qué pasa si aplicas la iteración QR sin shift a una matriz ortogonal? o **hagan el que quieran**. Sea A una matriz de Hessenberg superior y sea $QR = A$ la factorización QR de A. Muestra que RQ es una matriz superior de Hessenberg.

   Contestando a la primera pregunta. Sea A una matriz ortogonal, sabemos que la descomposición QR en una matriz ortogonal y una triangular superior es única (salvo un signo). Además se satisface que $A = AI$, y como A es ortogonal y I es triangular superior, esta es la única descomposición QR de A.

   Por lo anterior, $Q_1 R_1 = A_0 = A$, entonces $Q_1 = A, R_1 = I$, por un argumento recursivo llegamos a que para cada paso del algoritmo tenemos

$$\begin{aligned} &Q_k R_k = A_{k-1} = A, \qquad \Rightarrow Q_k = A, R_k = I, \\ &\Rightarrow A_k = R_k Q_k = IA = A. \end{aligned}$$

   Es decir, la matriz no cambia después de todas las iteraciones, por lo que el algoritmo no representa ninguna utilidad para encontrar los eigenvalores.