Cómputo Científico para Probabilidad, Estadística y Ciencia de Datos

Ezau Faridh Torres Torres

TAREA 4: Cálculo de eigenvalores.

Fecha de entrega: 25/Sep/2024.

NOTA: Los ejercicios se encuentran repartidos en los archivos:

- QR_iteration.py
- ejercicio2_tarea4.py

---

1. Dado el siguiente:
   Teorema (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 $n - m$ discos, entonces hay exactamente n eigenvalores en ese dominio.

   Deduce estimaciones de los eigenvalores de

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

   con $|\varepsilon| \leq 1$.

Respuesta:

Consideremos los discos de Gershgorin:

$$R_i = \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \right\} \tag{2}$$

En particular, como la matriz (1) es simétrica y diagonal dominante, es definida positiva. Con esto, se garantiza que los eigenvalores son reales positivos (se encuentran sobre la recta real).

Por el Teorema de Gershgorin, se sabe que los valores propios de (1) se encuentran en los discos:

- $R_1 = \{|\lambda - 8| \leq 1\}$
- $R_2 = \{|\lambda - 4| \leq 1 + |\varepsilon|\}$
- $R_3 = \{|\lambda - 1| \leq |\varepsilon|\}$

Notemos primero que R_1 nunca tiene intersección con los otros dos círculos para ningún valor de $|\varepsilon| \leq 1$. El radio mayor para R_2 y R_3 se da cuando $|\varepsilon| = 1$ y en este caso, tampoco existe intersección entre los círculos. Por lo que para ningún valor de $|\varepsilon| \leq 1$, existe intersección entre los círculos de Gershgorin y se garantiza que, al ser una matriz de 3×3, hay un eigenvalor en cada círculo debido a que cada uno de estos es un dominio conexo, disjunto de los demás (por el Teorema de Gershgorin).

En particular, si $\varepsilon = 0$, se puede garantizar que $\lambda_3 = 1$ es un eigenvalor de (1). Con esto, la submatriz

$$\begin{pmatrix} 8 & 1 \\ 1 & 4 \end{pmatrix} \tag{3}$$

tiene eigenvalores cercanos a 8 y 4 ya que es simétrica y es diagonal dominante. Para ser exactos, el polinomio característico es $(8-\lambda)(4-\lambda) - 1 = 0$ que tiene como soluciones: $\lambda_1 = 8.236$ y $\lambda_2 = 3.764$. Entonces, para la matriz

$$\begin{pmatrix} 8 & 1 & 0 \\ 1 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{4}$$

los eigenvalores son exactamente

$$\lambda_1 = 8.236,\ \lambda_2 = 3.764 \text{ y } \lambda_3 = 1. \tag{5}$$

Con esto, la forma general para cualquier $|\varepsilon| \leq 1$, (1), es una perturbación de (4). Dado que $a_{23} = a_{32} = \varepsilon$, se genera un acoplamiento entre las variables en las filas 2 y 3, lo que hace que los eigenvalores λ_2 y λ_3 interactúen entre sí, esto provoca que estos eigenvalores varíen más a medida que $|\varepsilon|$ crece, mientras que λ_1 permanecerá casi sin cambios ya que no está directamente conectado con ε.

En resumen, la teoría de perturbaciones dice que si los eigenvalores de la matriz sin perturbación (es decir, cuando $\varepsilon = 0$) están separados lo suficiente, las perturbaciones pequeñas causarán solo pequeños cambios en los eigenvalores. Es decir, los valores dados en (5) sufrirán pequeños cambios según $|\varepsilon|$ crece (a excepción de λ_1, que tendrá cambios casi nulos).

---

2. Implementa la iteración QR. Aplícala a la matriz A del Ejercicio 1 con $\varepsilon = 10^{-N}$ para $N = 0, 1, 2, 4, 5$. Compara lo obtenido por el algoritmo de Scipy.

Respuesta:

En el archivo QR_iteration.py, se define la función *QR_iteration()* la cual recibe como argumentos a la matriz A de interés, el número máximo de iteraciones (default $= 1000$), la tolerancia para la convergencia y un booleano que indica si se debe sobreescribir la matriz A original. La función devuelve un arreglo con los eigenvalores de la matriz A. Dado que en cada iteración se realiza la factorización QR de la matriz, se hace uso de la función *MODIFIED_GRAM_SCHMIDT()*, implementada en las tareas pasadas.

Para el criterio de paro se toman en cuenta las magnitudes de las entradas fuera de la diagonal principal, si son demasiado pequeñas, se termina el algoritmo (se toma un valor default de 10^{-7}).

Para revisar que la función *QR_iteration()* funcione correctamente, se usaron muchos ejemplos prácticos para ver su desempeño. En particular, se adjuntaron 3 ejemplos en QR_iteration.py en los cuales se puede destacar lo siguiente:

- Cuando la matriz tiene todos sus eigenvalores reales y distintos, el desempeño de la función *QR_iteration()* es practicamente el mismo que el de la función *eig()* de Scipy, encontrando rápidamente todos los eigenvalores.

- Cuando la matriz tiene eigenvalores reales pero repetidos, el desempeño de la función *QR_iteration()* es muy similar al de la función *eig()* de Scipy, ya que en la mayoría de los casos se notó una pequeña variación en los eigenvalores.
- Si la matriz tiene eigenvalores complejos, la función *QR_iteration()* no es capaz de encontrar la parte compleja del eigenvalor, mientras que *eig()* de Scipy sí los encuentra. Sin embargo, la parte real sí coincide con la esperada.

Estos resultados no sorprenden, ya que fueron comentados en clase y son los esperados según la parte teórica.

En el archivo ejercicio2_tarea4.py() se realiza la segunda parte del ejercicio: se comparan los resultados obtenidos por la función *QR_iteration()* con los obtenidos por la función *linalg.eig()* de Scipy para la matriz del ejercicio 1 para los diferentes valores de N y ε.

Se observa que los resultados obtenidos por ambas funciones son iguales, debido a que en cada caso, los eigenvalores de las matrices son todos reales positivos (que ya habíamos visto que era definida positiva).

Además, se confirma lo mencionado en el ejercicio 1: a medida que N crece, ε tiende a 0, generando que el eigenvalor λ_3 se acerque a 1, mientras que los otros se mantienen más cerca de los valores dados en (5) (a excepción de λ_2 que varía más su valor cuando N hace la transición de 0 a 1). A continuación, se da la tabla de resultados:

N	ε	**Iteración QR**	**Scipy**
0	1	[8.2435, 4.0710, 0.6853]	[8.2435, 4.0710, 0.6853]
1	0.1	[8.2361, 3.7673, 0.9965]	[8.2361, 3.7673, 0.9965]
2	0.01	[8.2360, 3.7639, 0.9999]	[8.2360, 3.7639, 0.9999]
3	0.001	[8.2360, 3.7639, 0.9999]	[8.2360, 3.7639, 0.9999]
4	0.0001	[8.2360, 3.7639, 1.0]	[8.2360, 3.7639, 1.0]
5	0.00001	[8.2360, 3.7639, 1.0]	[8.2360, 3.7639, 1.0]

Cuadro 1: Comparación de eigenvalores entre Iteración QR y Scipy para distintos valores de N y ε.

De hecho, a partir de $N = 2$, ya no se notan cambios importantes en los eigenvalores de (1).