[2024-2] Tarea 1 - Análisis Númerico

Pablo Garcia, Nixon Lizcano, Emanuel Velez

December 2024

1. Polinomio característico

1.1. Descripción del método

Este método para hallar los valores propios de una matriz se basa en las raíces del polinomio obtenido al realizar el siguiente cálculo:

$$det(A - \lambda I) = 0$$

De donde: A es la matriz a la que queremos hallar sus valores propios; λ es la variable del polinomio; I es la matriz identidad.

Y para hallar sus vectores propios, teniendo los valores propios, sería resolver la ecuación:

$$(A - \lambda_0 I)v = 0$$

De donde: A es la matriz a la que queremos hallar sus vectores propios; λ_0 es algún valor propio de A; I es la matriz identidad; v será aquel vector propio correspondiente a λ.

Una vez planteado el polinomio característico de la primera parte, hallamos sus raíces, donde dichas raíces serán sus valores propios. Usamos cada valor propio λ_0 para hallar un vector propio relacionado a dicho valor.

Sea $A = \left[a_{ij}\right]$ una matriz de tamaño nxn Para hallar el determinante de la matriz A se recurre a la fórmula iterativa

$$det(A) = \sum_{k=1}^{n} (-1)^{k-1} a_{1k} det(A_{1k})$$

Donde A_{1k} es la submatriz cuadrada al eliminar la fila 1 y la columna k, para cada k.
Donde usamos el método de Newton-Raphson para poder hallar las raíces (valores propios) con aproximaciones de las raíces suficientemente buenas (Que cumplan las

condiciones para la convergencia de este método)
Para hallar el vector v tal que se cumpla la ecuación

$$(A - \lambda_0 I)v = 0$$

Para hallar dichos vectores podemos usar la función de python np.linalg.solve para hacerlo de forma eficiente, al estar basada en un método númerico eficiente para esta tarea

1.2. Análisis de error

En este caso, el método para hallar los valores propios de A de tamaño nxn tendremos una convergencia de $O(n!)$ para hallar el polinomio característico y una convergencia de $O(n^2 log(\frac{1}{\varepsilon}))$.

1.2.1. Teorema

La convergencia para hallar la determinante de una matriz de tamaño n es de $O(n!)$ (con el método planteado)

Prueba: Sea $T(n)$ la convergencia para hallar el determinante de una matriz de nxn, esta función esta dada por la siguiente fórmula:

$$T(n) = nT(n-1) + O(n)$$

Dado que se realizan n multiplicaciones y $(n-1)$ sumas, para luego realizar nuevamente la expansión para una matriz de $(n-1)x(n-1)$ en cada término Así obtenemos la recursión:

$$T(n) = n(n-1)T(n-2) + O(n^2)$$
$$T(n) = n(n-1)(n-2)T(n-3) + O(n^3)$$
$$.$$
$$.$$
$$.$$

Donde es fácilmente ver que $T(1) = O(1)$.

$$\therefore T(n) = O(n!) \blacksquare$$

1.2.2. Teorema

La convergencia para hallar todas las raíces del polinomio característico (valores propios de la matriz) es de $O(n^2 log(\frac{1}{\varepsilon}))$ (con el método planteado)

Prueba: Sea ε la tolerancia para hallar cada raíz del polinomio y la iteración del método de Newton Raphson

$$x_{k+1} = x_k + \frac{P(x_k)}{\frac{\mathrm{d}}{\mathrm{d}x}P(x_k)}$$

.

Veamos la complejidad para cada paso del método.
Paso 1: Evaluar el polinomio, lo cual tiene una convergencia de $O(n)$
Paso 2: Evaluar la derivada, lo cual, al ser la derivada de un polinomio tiene la misma convergencia que el paso anterior $O(n)$
Paso 3: Calcular lo siguiente $\frac{P(x_k)}{\frac{\mathrm{d}}{\mathrm{d}x}P(x_k)}$, lo cual tiene un complejidad de $O(1)$ por lo que es despreciable
Paso 4: Hallar x_{k+1}, al restar obtenemos una complejidad de $O(1)$ por lo que es despreciable
El número de iteraciones tiene una complejidad de $O(log(\frac{1}{\varepsilon}))$ teniendo una buena estimación para la raíz original
Por lo que para hallar una raíz tenemos una complejidad de $O(nlog(\frac{1}{\varepsilon}))$
Paso 5: Hallando una raíz λ_0, podemos hallar un polinomio de grado $(n-1)$ $K(x)$ con las demás raíces dado por

$$K(x) = \frac{P(x)}{(x-\lambda_0)}$$

Paso 6: Reiterando los anteriores pasos para $K(x)$ n veces para hallar las demás raíces, nos queda que la convergencia esta dada por $O(n^2log(\frac{1}{\varepsilon}))$ ■

En este caso, el método para hallar los vectores propios de A de tamaño nxn tendremos una convergencia de $O(n^3)$ dado que la función de python np.linalg.solve tiene una convergencia similar a métodos como LU

1.2.3. Teorema

El método LU para hallar vectores v de tales que se cumpla la ecuación

$$Av = b$$

Donde A sea una matriz y b un vector dado, tiene convergencia de $O(n^3)$

Prueba: Veamos la complejidad para cada paso del método.
Paso 1: La descomposición de A en las matrices L y U tales que: A=LU; L es una matriz triangular inferior; U es una matriz triangular superior. Usando eliminación Gaussiana podemos hallar dichas matrices, en donde realizamos lo siguiente.
Eliminamos $(n-1)$ elementos de la primer columna $\Rightarrow (n-1)$ operaciones de división para hallar los multiplicadores $\wedge$ $(n-1)^2$ restas y multiplicaciones para actualizar las

filas restantes. Luego, iterativamente, para la k-columna eliminamos $(n-k)$ elementos Por lo que el número total de operaciones a realizar es de

$$\sum_{k=1}^{n-1}(n-k)^2 = \sum_{i=1}^{n-1} i^2 = \frac{n(n-1)(2n-1)}{6}$$

$\therefore$ Este paso tiene convergencia de $O(n^3)$

Paso 2: En este paso debemos resolver la ecuación $Ly = b$ hallando y por medio de sustitución hacia adelante lo cual tiene convergencia de $O(n^2)$

Se omitirá esta prueba dado que el siguiente paso tiene un razonamiento análogo
Paso 3: En este paso debemos resolver la ecuación $Uv = y$ hallando y por medio de sustitución hacia atrás lo cual tiene convergencia de $O(n^2)$
Sean v_i la i-entrada del vector v; y_i la i-entrada del vector y; u_{ij} la entrada de la matriz U donde i determina la fila y j determina la columna. Realizamos la siguiente operación para cada valor de $1 \leqslant i \leqslant n$

$$v_i = \frac{1}{u_{ii}}\left(y_i - \sum_{j=i+1}^{n} u_{ij}v_j\right)$$

Así fácilmente veremos que para v_n tendremos que realizar 0 operaciones, para v_{n-1} realizamos 1 operación, y en general para v_k haremos $(n-k)$ operaciones, por lo que el número total de operaciones sera de

$$\sum_{k=1}^{n}(k-1) = \frac{n(n-1)}{2}$$

O equivalente a decir que tendremos una convergencia de $O(n^2)$

$\therefore$ Tendremos una convergencia total de $O(n^3)$ para este metodo ■

2. Método de la potencia

En este método se encuentra el autovalor "dominante.º con mayor valor absoluto de una matriz A cuadrada y diagonalizable y su vector propio normalizado respectivo. Este proceso de normalizado seda cuando la componente más grande del vector V es 1.

Supongamos que la matriz A tiene un valor propio dominante λ_1 y un vector propio V normalizado correspondiente. Esta pareja se puede encontrar con el siguiente proceso iterativo.

$$Y_k = AX_k \qquad X_0 = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^\top$$
$$X_{k+1} = \frac{1}{c_{k+1}}Y_k \qquad c_{k+1} = \max_{1\leq i\leq n} |(Y_k)_i|$$

c_{k+1} es la coordenada de Y_k con mayor valor absoluto. Tenemos que las secuencias $\{X_k\}$ y $\{c_k\}$ convergen a

$$\lim_{n\to\infty} X_k = V_1 \qquad \lim_{n\to\infty} c_k = \lambda_1$$

Esta convergencia se puede probar de la siguiente forma:

Prueba: Como Atiene n valores propios también se tienen n valores propios V_i linealmente independientes que forman una base para $\mathbb{R}^n$, con esta base se puede representar el vector inicial X_0 como

$$X_0 = b_1V_1 + \cdots + b_nV_n$$

Aplicando las iteraciones obtenemos lo siguiente:

$$\begin{aligned} Y_0 = AX_0 &= \mathcal{A}(b_1V_1 + \cdots + b_nV_n) \\ &= b_1\lambda_1V_1 + \ldots b_n\lambda_nV_n \\ &= \lambda_1\left[b_1V_1 + b_2\left(\frac{\lambda_2}{\lambda_1}\right)V_2 + b_n\left(\frac{\lambda_n}{\lambda_1}\right)V_n\right] \end{aligned}$$

y con c_i el componente de mayor valor absoluto de Y_k

$$\begin{aligned} X_1 &= \frac{\lambda_1}{c_1}\left[b_1V_1 + b_2\left(\frac{\lambda_2}{\lambda_1}\right)V_2 + \cdots + b_n\left(\frac{\lambda_n}{\lambda_1}\right)V_n\right] \\ &\vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\ X_k &= \frac{\lambda_1^k}{c_1c_2\ldots c_{k-1}}\left[b_1V_1 + b_2\left(\frac{\lambda_2}{\lambda_1}\right)^k V_2 + \cdots + b_n\left(\frac{\lambda_n}{\lambda_1}\right)^k V_n\right] \end{aligned}$$

Como asumimos que $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$ tenemos que:

$$\lim_{k\to\infty} b_i\left(\frac{\lambda_i}{\lambda_1}\right)^k V_i = 0 \qquad \to \qquad \lim_{k\to\infty} X_k = \lim_{k\to\infty} \frac{b_1\lambda_1^k}{c_1c_2\ldots c_{k-1}}V_1$$

Como en cada iteración se busca normalizar V_1 tal que su mayor componente es 1 se tiene que el coeficiente escalar de V_1 también tiene que ser 1, entonces:

$$\lim_{k\to\infty} \frac{b_1\lambda_1^k}{c_1c_2\ldots c_{k-1}} = 1 \qquad \to \qquad \lim_{k\to\infty} X_k = V_1$$

Agarrando este resultado para $k-1$ en el siguiente limite tenemos lo siguiente

$$\lim_{k\to\infty} \frac{\lambda_1}{c_k} = \lim_{k\to\infty} \frac{b_1\lambda_1^k/(c_1c_2\ldots c_k)}{b_1\lambda_1^{k-1}/(c_1c_2\ldots c_{k-1})} = 1$$

Con esto tenemos que $\lim_{k\to\infty} c_k = \lambda_1$ ■.

Para la velocidad de convergencia tomamos el siguiente ratio de error

$$X_k = \frac{b_1 \lambda_1^k}{c_1 \dots c_k} V_1 + \frac{1}{c_1 \dots c_k} \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^k V_i$$
$$= \frac{b_1 \lambda_1^k}{c_1 \dots c_k} V_1 + \mathcal{O}(r^k)$$

Con $r = \frac{\lambda_2}{\lambda_1}$ y $e_k \propto r^k$ con lo que se llega a

$$\|e_{k+1}\| \approx r \|e_k\|$$

Por lo que el error decrece linelmente.

3. Descomposición QR

3.1. Descripción del método

Este método se basa en usar la descomposición de una matriz cuadrada n x n para determinar de forma directa los valores propios de la matriz. En específico, la descomposición requerida es de la siguiente forma:

$$A = QR$$

Donde Q es una matriz ortogonal (es decir, $Q^T Q = I$) y R es una matriz triangular superior (es decir, $a_{ij} = 0$ para $j < i$).

Mediante la descomposición y un proceso iterativo de la forma $A_{k+1} = R_k Q_k$, la secuencia de matrices de la forma $(A_k)_{k \in N}^{\infty}$ converge a una matriz T cuya diagonal contiene los valores propios de la matriz A.

Antes de proceder con el algoritmo, se procede con una revisión teórica de su funcionamiento:

3.1.1. Proceso Gram-Schmidt para ortogonalización de matrices

Dada una matriz $A \in \mathbb{C}_{nxn}$ cuyas columnas son linealmente independientes, entonces se puede hallar una descomposición de la forma $A = QR$, donde Q es una matriz ortogonal y R es una matriz triangular superior. La construcción se da de la siguiente forma:

Escribiendo a A en términos de sus columnas:

$$A = [a_1, a_2, ..., a_n]$$

Entonces con el siguiente proceso iterativo:

$$u_1 = a_1$$
$$u_2 = a_2 - (a_2 \cdot e_1) e_1, \qquad e_1 = \frac{u_1}{||u_1||}$$

$$u_{k+1} = a_{k+1} - (a_{k+1} \cdot e_1)e_1 - \cdots - (a_{k+1} \cdot e_k)e_k, \quad e_{k+1} = \frac{u_{k+1}}{||u_{k+1}||}$$

Se puede representar a la matriz A de la siguiente forma:

$$A = [a_1, a_2, ..., a_n] = [e_1, e_2, ..., e_n] \begin{bmatrix} a_1 \cdot e_1 & a_2 \cdot e_1 & \cdots & a_n \cdot e_1 \\ 0 & a_2 \cdot e_2 & \cdots & a_n \cdot e_2 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & a_n \cdot e_n \end{bmatrix}$$

Es importante notar que este proceso genera vectores u_k ortogonales entre ellos (es decir, $u_i \cdot u_j = 0 \ \forall i \ \neq j$). Aún más, los vectores e_k también serán ortogonales entre ellos por ser múltiplos de los u_k. Por tanto, la matriz conformada por elementos e_k será una matriz ortogonal.

Por otro lado, se puede notar que la matriz de la derecha queda como una matriz triangular superior. Así, se puede ver que A se puede descomponer como producto de Q ortogonal y R triangular superior.

Ahora, será importante hacer una relación entre matrices de la forma $A = Q^T A' Q$, donde Q es una matriz ortogonal. Diremos que el par A, A' corresponden a matrices similares.

3.1.2. Teorema 3.1

Los valores propios de dos matrices similares A, A' son iguales.

Demostración:
Como son similares, entonces $A' = Q^T AQ$. Ahora, veamos que aquellos $\lambda \in \mathbb{C}$ y $v \in \mathbb{R}^n$ que cumplen que:

$$Av = \lambda v$$

Tambien cumplen que:

$$A'(Q^T v) = Q^T AQ(Q^T v) = Q^T Av = Q^T \lambda v = \lambda(Q^T v)$$

Por tanto, también serán valores propios de A' (en este caso, el vector propio de A' será $Q^T v$ en vez de v).

3.2. Algoritmo de descomposición QR

El algoritmo de descomposición QR consiste en los siguientes pasos:

Dada una matriz A diagonalizable y cuyas columnas son linealmente independientes, entonces se pueden hallar los valores propios de la matriz mediante el siguiente proceso iterativo:

1. Defina $A_0 = A$ y halle su descomposición QR de la forma $A = Q_0 R_0$
2. Defina $A_i = R_{i-1} Q_{i-1}$. Halle su descomposición QR de la forma $A_i = Q_i R_i$

3. Itere hasta cierta cantidad M de iteraciones, definida por el usuario o hasta que se cumpla cierta tolerancia en los valores en el triangulo inferior de la matrizA_i.

4. Cuando se haya dejado de iterar, defina los valores propios de A como las entradas en la diagonal de la última matriz A_i calculada.

3.2.1. Convergencia del algoritmo de descomposición QR

Sin proporcionar una demostración formal, tratemos de entender por qué funciona este algoritmo.

Dado que $Q_i^{-1} = Q_i^T$ (debido a que todas las Q_i son ortogonales), tenemos $R_i = Q_i^T A_i$, y por lo tanto:

$$A_{i+1} = Q_i^T A_i Q_i.$$

Así, hemos llegado a una relación de similitud entre las matrices A_{i+1} y A_i; como se demostró por el teorema, tendrán los mismos valores propios. Si iteramos la ecuación anterior, obtenemos:

$$A_{i+1} = Q_i^T A_i Q_i = Q_i^T Q_{i-1}^T A_{i-1} Q_{i-1} Q_i = \cdots = Q_i^T \cdots Q_1^T A_1 Q_1 \cdots Q_i = P_i^T A_1 P_i,$$

donde $P_i = Q_1 \cdots Q_i$. Notemos que P_i es el producto de matrices ortogonales y, por lo tanto, es ortogonal.

Bajo nuestras suposiciones de diagonalización para la matriz A, podemos escribir $A = XDX^{-1}$, donde $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$ y X es una matriz real de vectores propios de A. Sabemos que para X también existe una factorización $X = QR$. Entonces:

$$A = QRDR^{-1}Q^{-1}.$$

Por lo tanto, $Q^{-1}AQ = RDR^{-1}$. Dado que RDR^{-1} es el producto de matrices triangulares superiores, también es triangular superior. De esta forma, sabemos que $Q^{-1}AQ = Q^T AQ$ es triangular superior. Notemos que los valores propios de A se encontrarán en la diagonal de $Q^T AQ$.

El teorema se demuestra mostrando que $\lim_{i\to\infty} P_i = Q$, ya que esto implica que:

$$\lim_{i\to\infty} A_{i+1} = \lim_{i\to\infty} P_i^T A_1 P_i = Q^T AQ.$$

En otras palabras, las matrices A_i están convergiendo hacia una matriz triangular superior cuyos elementos diagonales son los valores propios de A.

Para ver por qué $\lim_{i\to\infty} P_i = Q$, examinamos la cantidad $P_i U_i$, donde $U_i = R_i R_{i-1} \cdots R_1$. Entonces:

$$P_i U_i = Q_1 \cdots Q_i R_i \cdots R_1 = Q_1 \cdots Q_{i-1} A_i R_{i-1} \cdots R_1 = P_{i-1} A_i U_{i-1}.$$

Pero como $A_{i+1} = P_i^T A_1 P_i$, tenemos $P_i A_{i+1} = A_1 P_i$, o, reduciendo los índices en uno:

$$P_{i-1} A_i = A_1 P_{i-1}.$$

Por lo tanto:

$$P_i U_i = A_1 P_{i-1} U_{i-1}.$$

Si iteramos esta identidad, obtenemos:

$$P_i U_i = (A_1)^{i-1} P_1 U_1 = (A_1)^{i-1} Q_1 R_1 = A^i.$$

Utilizando el hecho de que $A = XDX^{-1}$, también tenemos que $A_i = XD_iX^{-1}$. Sabemos que $X = QR$ y, por hipótesis, $X^{-1} = LU$ (es posible encontrarle una descomposición LU). Por lo tanto:

$$A_i = QRD_iLU = QR(D_iLD_i^{-1})D_iU.$$

Al igualar estas dos expresiones para A_i, obtenemos:

$$P_i U_i = QR(D_iLD_i^{-1})D_iU.$$

El paso clave en la demostración es mostrar que:

$$\lim_{i\to\infty} D_iLD_i^{-1} = I.$$

Asumiendo por el momento que esto es cierto, el lado derecho se convierte en QRD_iU. Sin embargo, RD_iU es el producto de matrices triangulares superiores y, por lo tanto, también es triangular superior. Así, esencialmente podemos identificar $\lim_{i\to\infty} P_i$ con Q, ya que ambas son matrices ortogonales, y $\lim_{i\to\infty} U_i$ con $\lim_{i\to\infty} RD_iU$, ya que ambas cantidades son matrices triangulares superiores.

Volviendo al paso clave, observamos que la matriz $D_iLD_i^{-1}$ es una matriz triangular inferior cuyo elemento en la posición j, k está dado por $l_{jk}(\lambda_j/\lambda_k)^i$, cuando $j > k$. Dado que $|\lambda_j/\lambda_k| < 1$ para $j > k$, tenemos que:

$$\lim_{i\to\infty} D_iLD_i^{-1} = I.$$

De lo anterior, se puede concluir también que la tasa de convergencia de este método depende de los valores propios de la matriz A, puesto que los elementos fuera de la diagonal dependen de la razón entre los valores propios.

3.3. Posibles mejores al algoritmo de descomposición QR

Para mejorar los resultados del algoritmo QR, se pueden aplicar diversas técnicas que optimizan tanto las propiedades de la matriz A como el propio proceso del algoritmo. A continuación, se describen algunas de estas mejoras:

3.3.1. 1. Mejoras en la matriz A

1. **Reordenamiento espectral:** Si los valores propios de A están muy cercanos entre sí, el algoritmo QR puede converger más lentamente. Una transformación inicial, como la reordenación espectral, puede separar mejor los valores propios dominantes y acelerar la convergencia.

2. **Reducción a forma Hessenberg:** Reducir la matriz A a una forma Hessenberg (triangular con una banda subdiagonal) preserva los valores propios y disminuye el número de operaciones necesarias en cada iteración. Esta reducción mantiene la estabilidad numérica del algoritmo.

3. **Transformación de similitud previa:** Aplicar una transformación de similitud, como $B = P^T AP$, donde P es una matriz ortogonal adecuada, puede hacer que A sea más fácil de procesar numéricamente. Por ejemplo, si A es simétrica, esta transformación puede garantizar mejor la convergencia.

4. **Normalización de la matriz:** Si los elementos de A son muy grandes o muy pequeños, pueden producir errores numéricos. Escalar A para que tenga una norma adecuada (por ejemplo, $\|A\| \approx 1$) puede mejorar la precisión.

3.3.2. 2. Mejoras en el algoritmo QR

1. **Uso del algoritmo QR con desplazamiento (shift):** Incorporar un desplazamiento espectral μ mejora la convergencia. Esto se logra al redefinir $A_i - \mu I$ en cada paso, donde μ es una aproximación de los valores propios dominantes. Los desplazamientos pueden calcularse con métodos como el desplazamiento de Rayleigh.

2. **Descomposición QR implícita:** En lugar de calcular explícitamente Q y R, se pueden usar métodos implícitos para reducir el tiempo computacional. Un ejemplo es el uso del algoritmo de Francis (también conocido como el método QR implícito).

3. **Aceleración por bloques:** Procesar bloques de la matriz en lugar de realizar iteraciones sobre toda A puede aprovechar mejor las arquitecturas modernas de hardware y reducir los tiempos de cómputo.

4. **Uso de precondicionadores:** Los precondicionadores pueden modificar A para mejorar la convergencia del algoritmo. Por ejemplo, aplicar un precondicionador diagonal puede mejorar las propiedades espectrales de la matriz.

5. **Factores de estabilización numérica:** En ciertas implementaciones, es necesario garantizar que R tenga elementos diagonales positivos para evitar inestabilidad numérica. Asegurarse de esta propiedad mejora la precisión en cálculos sucesivos.

4. Referencias

[1] Jing Liu, 11.2 Power Method, *Max Planck Institut für Physik*, n.d. Available at `https://www.mpp.mpg.de/~jingliu/ECPI/PowerMethodProof.pdf`

[2] Prabhakar Misra, *Numerically Reliable Computation of Characteristic Polynomials*, Proceedings of the American Control Conference, 1995.

[3] Igor Yanovsky, QR Decomposition with Gram-Schmidt, *Math 151B Handout*, n.d. Available at `https://www.math.ucla.edu/~yanovsky/Teaching/Math151B/handouts/GramSchmidt.pdf#:~:text=QR%20Decomposition%20with%20Gram-Schmidt%20Igor%20Yanovsky%20(Math%20151B%20TA)%20The`