Proyecto: "MÉTODO LANCZOS "

Análisis Numérico II - 2024

Diaz Solana - Villarroel Carolina

13 de noviembre de 2024

Resumen

Se expone el primer algoritmo iterativo del método de Lanczos, desarrollado por Cornelius Lanczos.

1. Introducción

El tema de este proyecto es encontrar una buena aproximación de los autovalores extremos de una matriz A cuadrada, simétrica, dispersa y de gran tamaño. Para ello el algoritmo, que tiene como entrada una matriz A cuadrada de tamaño nxn. un vector inicial q y m número máximo de iteraciones; genera una matriz T_k tridiagonal de tamaño mxm y una matriz Q ortogonal de tamaño nxm talque $AQ = QT$, donde veremos luego que los autovalores de T serán buenas aproximaciones de los autovalores de A ; y que Q tiene columnas ortonormales que generan $K(A, q1, k)$. Sin embargo, la precisión de los resultados depende de la cantidad de iteraciones realizadas, ya que es un método iterativo.

2. Trabajo realizado

2.1. Deducción del método de Lanczos

Supongamos que $T = Q^T AQ$, donde $Q = [q_1| \dots |q_n]$ y

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \dots & 0 \\ \beta_1 & \alpha_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \beta_{k-1} \\ 0 & \dots & & \beta_{k-1} & \alpha_k \end{bmatrix}$$

Igualando las columnas de $AQ = QT$ se deduce que para $k = 1, \dots, n-1$:

$$Aq_k = \beta_{k-1}q_{k-1} + \alpha_k q_k + \beta_k q_{k+1}, \qquad \beta_0 q_0 \equiv 0 \tag{1}$$

Notar que $T_{ij} = q_i^T Aq_j$, igualando coordenada a coordenada en $T = Q^T AQ$, tenemos que las entradas de la diagonal son

$$T_{kk} = q_k^T Aq_k \tag{2}$$

Por otro lado, obtenemos una expresión para calcular sucesivos q_k de la siguiente manera.

Definimos el siguiente vector (conocido por vector de residuos):

$$r_k = \beta_k q_{k+1} = (A - \alpha_k I)q_k - \beta_{k-1}q_{k-1} \tag{3}$$

y en caso de que $r_k \neq 0$, encontramos nuestro siguiente vector de Lanczos sin más que dividir por su norma:

$$q_{k+1} = r_k/\beta_k \tag{4}$$

donde

$$\beta_k = \|r_k\|_2 \tag{5}$$

De esta forma nos queda que el vector q_k son ortogonales y cómo los q_k son las columnas de Q, entonces obtendremos la matriz Q ortogonal.

Evidentemente, no hay pérdida de generalidad en haber elegido los β_k positivos. Por ello, podíamos haber tomado β_k = - $\|r_k\|_2$

Si encontramos un $r_k = 0$, la iteración termina; pero como veremos en el teorema siguiente, no sin llegar a un subespacio invariante. Cómo $r_k = 0$, entonces $\beta_k = 0$, lo cual es raro de encontrar en la práctica, pero igualmente se producen buenas aproximaciones incluso antes de encontrar un β_k pequeño. Por cual se suelen pedir un número m máximo de iteraciones de forma que si no se llega en m iteraciones a un subespacio invariante, finalizamos el algoritmo y habremos aproximado m autovalores de la matriz dada.

El siguiente teorema nos muestra cuándo termina el teorema de forma autónoma (es decir; sin llegar al número máximo de iteraciones que hayamos seleccionado).

Teorema 1. *El algoritmo de Lacnzos continúa hasta que $k = m$, donde*

$$m = rank(K(A, q_1, n)).$$

Es más, para $k = 1, \ldots, m$ se tiene que

$$AQ_k = Q_k T_k + r_k e_k^T \tag{6}$$

donde

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \ldots & 0 \\ \beta_1 & \alpha_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \beta_{k-1} \\ 0 & \ldots & & \beta_{k-1} & \alpha_k \end{bmatrix}$$

e_k es el vector canónico de tamaño k x 1 con el 1 en su última componente, y $Q = [q_1| \ldots |q_k]$ tiene columnas ortonormales que generan $K(A, q_1, k)$.

2.2. Primer algoritmo del método de Lanczos

Este es el primer algoritmo de Lanczos:

Algoritmo 1: Algoritmo de Lanczos

Entrada: Matriz $A \in \mathbb{R}^{n\times n}$, un vector aleatorio $q \in \mathbb{R}^n$, y m el número de iteraciones

Salida : Matriz tridiagonal $T_k \in \mathbb{R}^{k\times k}$, y $Q_k = [q_1 \mid \ldots \mid q_n] \in \mathbb{R}^{n\times k}$ tal que $AQ_k = Q_k T_k$

$q_1 = q/\|q\|$, $\beta_0 = 1$, $q_0 = 0$.
for $k = 1$ **to** m **do**
  $\alpha_k = q_k^T A q_k$;
  $r_k = Aq_k - \alpha_k q_k - \beta_{k-1} q_{k-1}$
  $\beta_k = \|r_k\|$;
  if $\beta_k \neq 0$ **then**
    $q_{k+1} = r_k/\beta_k$
  end
end

Por todo lo mencionado anteriormente podemos construir el algoritmo de la siguiente manera. Primero, debe tener como entradas:

- Una matriz A cuadrada, simétrica, dispersa y de gran tamaño,
- Un vector inicial q
- Un número máximo de iteraciones m

Iniciamos con una matriz T con todos sus elementos nulos, cuyo tamaño dependerá de la cantidad máxima de iteraciones, donde luego se guardará α_k (vector de dimensión m+1) en la diagonal principal y β_k (vector de dimensión m+1) en las diagonales adyacentes por debajo y por encima de esta. También creamos una matriz Q con todos sus elementos nulos, donde el número de sus filas es equivalente al número de filas de A y el número de sus columnas dependerá de la máxima cantidad de iteraciones.

Notemos que la dimensión de alpha y beta es de $m+1$ y no m, la razón de esto es que estén bien definidos los elementos de ambos vectores en las iteraciones.

Luego, siguiendo con el algoritmo, definimos un vector q_0 de dimensión n con todos sus elementos nulos, ponemos 1 en la primera coordenada de beta y en la primera columna de la matriz Q el vector q_1, siendo q_1 el vector de entrada q normalizado.

Una vez definidas las matrices y vectores necesarios, empezamos con las iteraciones k desde $k = 0$ hasta $k = m$. Por cada iteración calculamos el elemento de la coordenada $k+1$ de alpha a partir de (2), el vector residuo r con la fórmula dada por (3) y el elemento de la coordenada $k+1$ de beta, siendo r normalizado. Podemos notar que el vector residuo en cada iteración va a ir pisando la información anterior, en cambio a los vectores alpha y beta se va a ir agregando nueva información, sin pisar al anterior.

En caso de que el elemento de la coordenada $k+1$ de beta sea distinto de 0, podemos despejar q_1 de la fórmula del vector residuo y nos queda (4), esto es lo que hacemos en el if utilizado en el agoritmo. Antes de calcular el nuevo q_1, actualizamos q_0 cómo q_1 y agregamos en la columna k de Q el vector q_0 actualizado anteriormente.

Posteriormente construimos T, cuya diagonal principal pondremos todos los elementos de alpha menos el primer elemento que cómo habiamos definido es igual a 0 y sigue así después de las iteraciones, ya que cuando calculamos alpha con $k = 0$ tenemos alpha $[k+1]$ = alpha $[1] = 0$, es decir en ningún momento se modifica el primer elemento de alpha. Luego ponemos en la diagonal inferior y en la superior el vector beta, sin el elemento de la primera coordenada, que

es igual a 1 ya que la iteración no lo altera en ningún, y tampoco se agrega el último elemento (que se calculado de más).

Cómo resultado este algoritmo devuelve la matriz T, la matriz Q, el vector residuo r, el vector alpha y beta. Por el teorema 1, tenemos que

$$AQ_k = Q_k T_k + r_k e_k^T \tag{7}$$

Por lo que si queremos verificar si encontramos T y Q talque $AQ = TQ$ debemos verificar (7).

Ahora sólo nos falta calcular los autovalores de la matriz T, para ello utilizamos el método de Jacobi (que vimos en el teórico de esta materia). Creemos que este es el método más eficiente y menos costoso computacionalmente, a diferencia de otros métodos para calcular autovalores vistos en el teorico de la materia. Y de esta forma calculamos aproximaciones de los autovalores de la matriz A.

2.3. Ejemplo de la utilización del algoritmo

A continuación veremos un ejemplo donde se pretende ilustrar cómo se produce la convergencia de los autovalores cuando crece el número de iteraciones (de forma que podamos comparar con nuestras aproximaciones numéricas). Vamos a tomar una matriz cuyos autovalores exactos conozcamos, y aplicaremos el algoritmo de Lanczos para un vector inicial q_1.

Consideremos A una matriz diagonal de tamaño 1000 con entradas tomados de una distribución $N(\mu = 0, \sigma^2 = 1)$, que han sido reordenadas posteriormente, de forma que $a_{1,1} \geq a_{2,2} \geq ... \geq a_{1000,1000}$. Tomamos como vector inicial $q_1 = [1, ..., 1]^T$, es decir, el vector columna de 1000 entradas donde todas ellas son 1.

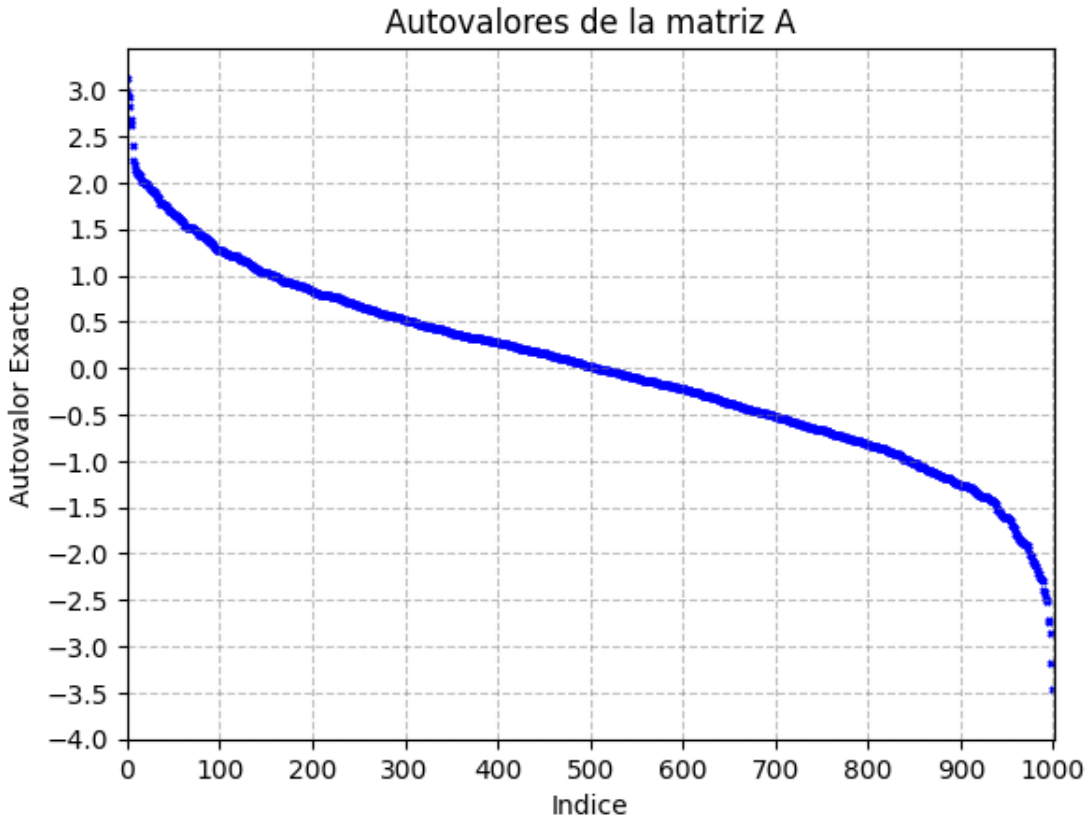

Figura 1: Autovalores exactos de la matriz A

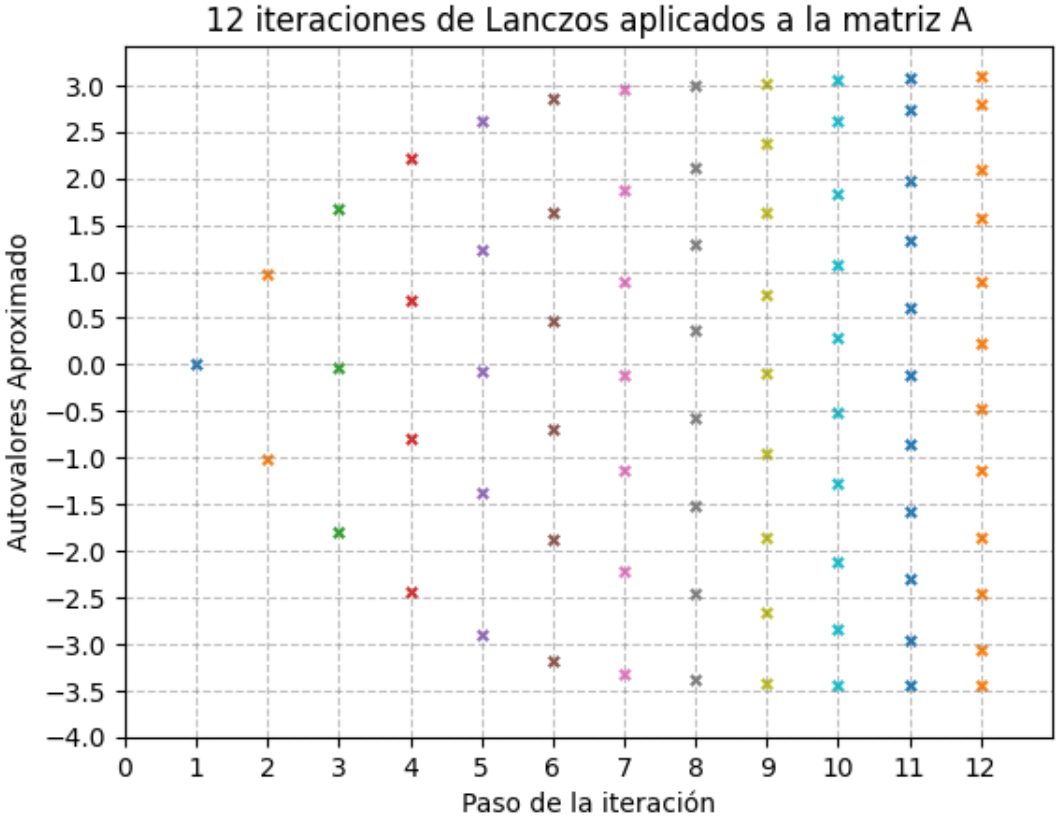

Figura 2: Autovalores aproximados por cada iteración de Lanczos frente a autovalores exactos para el vector inicial $q_1 = [1, 1, \ldots, 1]^T$

En la figura 1, podemos ver claramente que los autovalores exactos de A van desde 3 a -3. En la figura 2, se observa que cuantas más iteraciones tengamos más se aproximan los autovalores extremos de T a los autovalores extremos de A.

Hablemos un poco de cómo realizamos estos gráficos. Para hacer la figura 1, creamos una lista x donde cada elemento es un número del 1 al 1000 ordenados de forma ascendente ($x = [1, \ldots, 1000]$) , y una lista y donde cada elemento es un elemento de la diagonal de A, es decir, en y están los autovalores de A. Utilizamos la función $scatter(...)$ de matplotlib.pyplot para crear un gráfico, donde cada punto en el gráfico está representado por una coordenada en el plano (x, y). Cómo el vector x equivale a los indices de la matriz A, cada elemento del vector x se le asigna un elemento de y, osea a cada índice de A le corresponde un autovalor.

En cambio, para el otro gráfico utilizamos el algoritmo 1 de Lanczos que desarrollamos con la matriz A, q_1 y m, que fue cambiando desde 1 hasta 12, cómo entradas. Osea, utilizamos el algoritmo 12 veces cambiando la cantidad máxima de iteraciones. Para cada T_k dada por los algoritmos, calculamos sus autovalores con el método de jacobi para encontrar autovalores. Luego creamos una lista x, donde cada elemento es un número del 1 al 12 ordenados de forma ascendente ($x = [1, \ldots, 12]$), y una lista y, donde por cada índice están los autovalores de T_k con k el índice dicho. Por lo tanto los elementos de la lista y son vectores de dimensión k con k índice del elemento en la lista. Usamos, entonces, la función $scatter(...)$ de matplotlib.pyplot pero para cada índice de y copiamos x del tamaño del vector ubicado en el índice dicho, de esta forma creamos una gráfica donde por cada iteración, en el eje de las x, tenemos los autovalores aproximados de A, en el eje de las y.

2.4. Teoría de la convergencia del algoritmo

En esta sección mencionaremos teoremas que muestran la calidad de los autovalores aproximados mediante el algoritmo, así como la convergencia hacia los autovalores exactos de la matriz A en función del número de iteraciones k.

Teorema 2. *Supongamos que se han aplicado k iteraciones de Lanczos descritas en el algoritmo 1, y que* $S_k^T T_k S_k = diag(\theta_1, ..., \theta_k)$ *es la descomposición de Schur de la matriz* T_k. *Si* $Y_k = [y_1, ..., y_k] = Q_k S_k \in R^{nxk}$, *entonces para* $i = 1, \ldots, k$ *se tiene que:*

$$\|Ay_i - \theta_i y_i\|_2 = |\beta_k||s_{ki}|, \qquad donde \qquad S_k = (spq). \tag{8}$$

Siendo θ_i el i-ésimo autovalor de T_k, que son aproximaciones de los autovalores λ_i de la matriz A, y cada columna y_i de Y_k un vector aproximado del autovector correspondiente de A. En base a esto se puede deducir que:

$$Ay_i - \theta_i y_i = r_k(e_k^T S_k e_i), \tag{9}$$

donde r_k es el residuo y S_k matriz de la descomposición de Schur de la matriz T_k. Podemos notar que en el lado izquierdo de la ecuación (8), el error en la aproximación de los autovalores, depende del residuo r_k. Cuanto más pequeño es el residuo, mejor es la aproximación de los autovalores θ_i a los autovalores de A. En este sentido, se tiene el siguiente lema:

Lema 1. *Sea A una matriz simétrica, y supongamos que*

$$Ax = r \qquad con \qquad x \neq 0. \tag{10}$$

Entonces

$$\min_{\mu \in \lambda(A)} |\mu - \theta| \leq \frac{\|r\|_2}{\|x\|_2} \tag{11}$$

Este lema proporciona una cota para error al aproximar un autovalor λ de A mediante un valor θ. Y este resultado se recoge en el siguiente colorario:

Corolario. *Supongamos que se han aplicado k iteraciones del método de Lanczos. Si* $\theta_i = \lambda_i(T_k)$, *entonces*

$$\min_{\mu \in \lambda(A)} |\theta_i - \mu| \leq \frac{|\beta_k||s_{ki}|}{\|Q_k S_k e_i\|} \qquad para \qquad i = 1, ..., k, \tag{12}$$

donde $\lambda(A)$ *representa el espectro de la matriz A.*

En ausencia de errores de redondeo, $\|y\|_2 = \|Q_k S_k e_i\|_2 = 1$, por lo que sin más que sustituir en (12), tenemos que existe un autovalor μ de A tal que

$$|\mu - \theta_i| \leq |\beta_k||s_{ki}|. \tag{13}$$

Sin embargo, al implementar numéricamente este algoritmo, la pérdida de ortogonalidad de los vectores q_i puede destruir la cota anterior. Esta pérdida de ortogonalidad motiva, que nosotros no veremos, el desarrollo de diversas técnicas de reortogonalización.

Los siguientes teoremas estudian cómo los autovalores de T_k aproximan de bien los autovalores de A en función del número de iteraciones k. Estos resultados se enmarcan en lo que se conoce por Teoría de convergencia de Kaniel-Paige.

Teorema 3. *Sea A una matriz $n \times n$ real, y consideremos su descomposición de Schur*

$$Z^T AZ = diag(\lambda_1, \ldots, \lambda_n), \quad \lambda_1 \geq \cdots \geq \lambda_n, \quad Z = [z_1 | \ldots | z_n].$$

Supongamos que se han realizado k iteraciones de Lanczos (ver algoritmo 1), y que T_k es la matriz tridiagonal . Si $\theta_1 = \lambda_1(T_k)$, entonces

$$\lambda_1 \geq \theta_1 \geq \lambda_1 - (\lambda_1 - \lambda_n) \left(\frac{\tan(\phi_1)}{c_{k-1}(1 + 2\rho_1)} \right)^2 \tag{14}$$

donde $\cos(\phi_1) = |q_1^T z_1|$

$$\rho_1 = \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n} \tag{15}$$

y $c_{k-1}(x)$ es el polinomio de Chebyshev de grado $k-1$.

La desigualdad (15) nos proporciona una cota inferior para el autovalor θ_1 de la matriz T_k, lo que permite controlar el error en la aproximación del mayor autovalor λ_1 de A. Esta cota te dice que tan cerca está θ_1 del verdadero λ_1, si el valor de la cota inferior está cerca de λ_1, esto indica que el algoritmo de Lanczos convergió correctamente, y que la aproximación de θ_1 es bastante precisa.

Este resultado indica que la calidad de la aproximación de λ_1 mejora cuando k aumenta, especialmente si el polinomio de Chebyshev amplifica la componente en la dirección del autovector asociado al autovalor λ_1

Corolario. *Usando la misma notación que en el teorema anterior, si $\theta_k = \lambda_k(T_k)$, entonces*

$$\lambda_n \leq \theta_k \leq \lambda_n + (\lambda_1 - \lambda_n) \left(\frac{\tan(\phi_n)}{c_{k-1}(1 + 2\rho_n)} \right)^2,$$

donde

$$\cos(\phi_1) = |q_1^T z_n|, \quad y \quad \rho_1 = \frac{\lambda_{n-1} - \lambda_n}{\lambda_1 - \lambda_{n-1}}.$$

Este último teorema y corolario ofrecen información de la calidad de las aproximaciones al autovalor máximo y mínimo respectivamente en función del número de iteraciones k.

El siguiente teorema muestra cotas de error para las aproximaciones de los autovalores interiores. Notemos que debido al factor k_i, y al grado reducido del polinomio de Chebyshev, es claro que las cotas se deterioran cuando i crece.

Teorema 4. *Usando la misma notación que en el teorema 3, si $1 \leq i \leq k$ y $\theta_i = \lambda_i(T_k)$, entonces*

$$\lambda_i \geq \theta_i \geq \lambda_i - (\lambda_i - \lambda_n) \left(\frac{\kappa_i \tan(\phi_i)}{c_{k-i}(1 + 2\rho_i)} \right)^2, \tag{16}$$

donde

$$\rho_i = \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}, \quad \kappa_i = \prod_{j=1}^{i-1} \frac{\theta_j - \lambda_n}{\theta_j - \lambda_i}, \quad \cos(\phi_i) = |q_1^T z_i|.$$

Dado todo esto, podemos concluir que el Método de Lanczos converge hacia los autovalores exactos de una matriz simétrica A a medida que se incrementa el número de iteraciones k. Las aproximaciones de los autovalores extremos (máximo y mínimo) son especialmente buenas y mejoran significativamente con k. Sin embargo, las aproximaciones de los autovalores interiores son menos precisas a medida que k crece, debido a la presencia del factor k_i , que puede deteriorar la calidad de la aproximación. Para obtener mejores resultados en la práctica, especialmente para los autovalores interiores, a menudo se utiliza reortogonalización para mantener la ortogonalidad de los vectores de Lanczos, lo que mejora la precisión y la estabilidad del algoritmo. Esta reortogonalización consiste en otro algoritmo que no vamos a ver.

3. Conclusión

Cómo conclusión se puede decir que el algoritmo de Lanczos es un método de iteración muy bueno para la aproximación de autovalores y autovectores. Uno de sus principales beneficios del algoritmo es que esta diseñado específicamente para matrices grandes y dispersas, es decir, a diferencias de otros métodos, el algoritmo de Lanczos es más eficiente y adecuado para grandes escalas debido a su baja complejidad computacional por iteración.

El método es computacionalmente eficiente ya que agarra una matriz de gran tamaño y produce una matriz tridiagonal que es mucho mas fácil de manejar que la matriz original. En esta situación es mas simple aproximar los autovalores de una matriz tridiagonal que de una matriz de gran tamaño y dispersa.

Y como lo mencionamos anteriormente el algoritmo tiene una muy buena tasa de convergencia, es decir, que la precisión de las aproximaciones de los autovalores mejora cuando aumenta las iteraciones.

A. Copiando y pegando el código

```
def Lanczos(A,q,m):
    #Entrada: A una matriz nxn, q un vector aleatorio nx1 y m el N de iteraciones
    #Salida : Tk  matriz tridiagonal real de tamao kxk , y Qk = [q1 | . . . | qn]
    #matriz de tamaNo nxk tal que A Qk = Qk Tk

    n = np.shape(A)[0]
    T = np.zeros((m,m))
    Q = np.zeros((n,m))
    beta = np.zeros(m+1)
    alpha = np.zeros(m+1)
    q0 = np.zeros(n)

    q1 = q / np.linalg.norm(q,ord=2)
    beta[0]  = 1
    Q[:,0] = q1

    for k in range(m):
        alpha[k+1] =  q1.T@A@q1
        r = A@q1 - alpha[k+1]*q1 - beta[k]*q0
        beta[k+1] = np.linalg.norm(r,ord=2)

        if beta[k+1] !=0:
            q0 = q1
```

```
            Q[:,k] = q0
            q1 = r / beta[k+1]

    T = np.diag(alpha[1:]) + np.diag(beta[1:m], -1) + np.diag(beta[1:m], 1)

    return  T, Q, r, alpha, beta
```

B. Poniendo un hipervínculo

Pueden ver el código del algoritmo en este link: Código

Referencias

[1] Autor: Alfonso Martín Mozo, Tutora: Begoña Cano Urdiales
Universidad de Valladolid, Faculad de Ciencias, Método de Lanczos, Junio 2021
Pueden ver la referencia en este link: Referencia