

Resumen

En este trabajo se desarrollaron dos métodos para encontrar los eigenvalores y los eigenvectores propios de una matriz tridiagonal. Se utilizó el método de la potencia para encontrar el eigenvalor propio más grande y el método de la potencia inversa para el eigenvalor más pequeño. Además, se generalizaron los métodos para encontrar los n valores y vectores propios.

1. Introducción

Los vectores propios o eigenvectores son los vectores no nulos que cuando son transformados por un operador, dan lugar a un múltiplo escalar de sí mismos [1]. Este escalar, recibe el nombre de eigenvalor o valor propio. Los eigenvalores de una matriz se encuentran calculando su polinomio característico ($p(\lambda) = \det(A - \lambda I)$), se le resta a la diagonal principal un parámetro λ , después se calcula el determinante y se iguala a cero y de esta forma se encuentran sus eigenvalores.

2. Metodología

2.1. Método de la potencia

Este es un método iterativo que aproxima los valores propios. Para aplicar este método, se supone que la matriz A cuadrada tiene n valores característicos, y además, que la longitud de uno de estos valores característicos es la mayor. Para generalizar el método, se resta la contribución de cada eigenvector encontrado en cada iteración, es decir, si se encontró el eigenvalor dominante, para encontrar el segundo más grande, se le restará (en cada proceso iterativo) la multiplicación escalar del vector inicial v_0 por el vector propio encontrado, por el vector propio en cada iteración. Sin embargo, esto se recomienda sólo cuando interesan los más grandes, no todos los vectores o valores propios, ya que el proceso iterativo acarrea errores en cada iteración. En este método se parte de $Av_0 = v_1$, donde v_0 y v_1 son vectores escritos en términos de los vectores propios de A .

Algorithm 1 Método de la potencia

Entrada: Matriz A , vector inicial (v_0), vector vacío v_1 , tolerancia, valor viejo y valor

Salida: Eigenvalor propio más grande, eigenvector propio asociado, N : nodos

```
1: function NORMALIZAR VECTOR(vector 1, nodos)
2: function PRODUCTO VECTOR VECTOR(vector 1, vector 2, nodos)
3: function PRODUCTO MATRIZ VECTOR(Matriz, vector, nodos)
4: Inicializar  $v_0$  con cualesquiera elementos excepto 0.0
5: Normalizar ( $v_0$ )

6: error =  $10^{10}$ ;
7: Método generalizado
8: for  $n = 0, n < \text{cantidad de eigenvalores a buscar}$  do
9:   Reiniciar parámetros: valor,  $v_0, v_1$ , valor viejo, valor.
10:  while error > tolerancia do
11:    for  $i = 0; i < N; i++$  do
12:      for  $k = 0; k < \text{cantidad de eigenvalores}; k++$  do
13:         $v_0[i] -= \text{producto vector vector}(v_0 \text{ vector propio}, N) * \text{vector propio}[i]$ 
14:       $v_1 = \text{producto matriz vector}(A, v_0, N)$ 
15:      valor =  $\text{producto vector vector}(v_0, v_1, N)$ 
16:      error =  $\text{abs}(\text{valor} - \text{valor viejo})$ 
17:       $v_1 = \text{Normalizar vector}(v_0, N)$ 
18:      valor viejo = valor
19:      Reemplazar vector de inicio
20:      for  $i = 0; i < N; i++$  do
21:         $v_0[i] = v_1[i]$ 
22:      vector propio = guardar eigenvector
23: Return: vector propio y eigenvalor (valor).
```

2.2. Método de la potencia inversa

Este método genera una convergencia más rápida, y en lugar de encontrar el valor dominante, encuentra el valor más pequeño. Este supone resolver un sistema de ecuaciones, y puede simplificarse desde el inicio al hacer una factorización matricial. Esto es porque hacer el cálculo de la inversa es más caro (computacionalmente), que hacer la factorización. En este método se parte de resolver:

$$Av_1 = v_0, \quad (1)$$

que es parecido a resolver un sistema $Ax = b$.

Algorithm 2 Método de la potencia

Entrada: Matriz A, vector inicial (v_0), vector vacío v_1 , tolerancia, lambda


Salida: Eigenvalor propio más grande, eigenvector propio asociado, N: nodos

```
1: function NORMALIZAR VECTOR(vector 1, nodos)
2: function PRODUCTO VECTOR VECTOR(vector 1, vector 2, nodos)
3: function PRODUCTO MATRIZ VECTOR(Matriz, vector, nodos)
4: Factorizar la matriz con LU, LDU, o  $LL^T$ 
5: Inicializar  $v_0$  con cualesquiera elementos excepto 0.0
6: Normalizar ( $v_0$ )

7: error =  $10^{10}$ ;
8: Método generalizado
9: for n = 0, n < cantidad de eigenvalores a buscar do
10:   Reiniciar parámetros: valor,  $v_0$ ,  $v_1$ , valor viejo, valor.
11:   while abs(valor) > tolerancia do
12:     for i = 0; i < N; i ++ do
13:       for k = 0; k < cantidad de eigenvalores; k ++ do
14:          $v_0[i] = \text{producto vector vector}(v_0 \text{ vector propio, N}) * \text{vector propio}[i]$ 
15:       Soluciona sistema de ecuaciones:  $Av_1 = v_0$ 
16:       suma = 0
17:       for i = 0; i < N; i ++ do
18:         suma +=  $v_0[i] * v_1[i]$ 
19:       lambda = 1/suma
20:       valor = abs(lambda - lambda old)
21:       Normalizar ( $v_1$ )
22:       for i = 0; i < N; i ++ do
23:          $v_0[i] = v_1[i]$ 
24:       lambda old = lambda
25: Return: vector propio y eigenvalor (valor).
```

3. Resultados

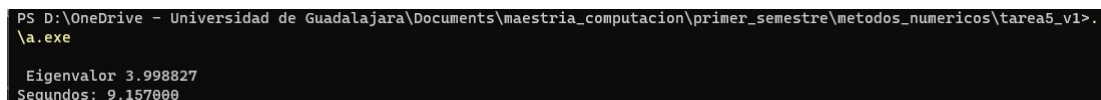
Se utilizó una matriz tridiagonal con 2 en la diagonal principal y con -1 en los otros términos de la matriz, con 1000 nodos. Para el método de la potencia, se tuvo que utilizar una tolerancia baja, ya que al aumentar la tolerancia, tardaba mucho en encontrar el eigenvalor más grande. En este caso, se utilizó una tolerancia de 0.01. Y para el método de la potencia inversa, se resolvió el sistema factorizando con el método de Cholesky para después resolver con funciones triangular superior e inferior. Cuando la matriz tenía valores negativos, a partir de aproximadamente el nodo 150 el valor propio colapsaba y se hacía muy pequeño, lo que estaba incorrecto como se muestra en la fig. 1. Sin embargo, si utilizaba la matriz con todos positivos, esta convergía muy rápido y con un buen resultado (fig. 2).



```
PS D:\OneDrive - Universidad de Guadalajara\Documents\maestria_computacion\primer_semestre\metodos_numericos\tarea5_v1>.
\la.exe

Eigenvalor 0.002000
Eigenvalor 0.044721
Segundos: 3.933000
```

Figura 1: Método de la potencia para una matriz con 2 en la diagonal y -1 en los términos vecinos.



```
PS D:\OneDrive - Universidad de Guadalajara\Documents\maestria_computacion\primer_semestre\metodos_numericos\tarea5_v1>.
\la.exe

Eigenvalor 3.998827
Segundos: 9.157000
```

Figura 2: Método de la potencia con todos los términos de la tridiagonal positivos.

Además, se probó el método para la misma matriz con distintos nodos, en el caso de 5 nodos, los resultados se muestran en la fig. (3).

```

PS D:\OneDrive - Universidad de Guadalajara\Documents\maestria_computacion\primer_semestre\metodos_numericos\tarea5_v1>.
\la.exe

2.000000  -1.000000  0.000000  0.000000  0.000000
-1.000000  2.000000  -1.000000  0.000000  0.000000
0.000000  -1.000000  2.000000  -1.000000  0.000000
0.000000  0.000000  -1.000000  2.000000  -1.000000
0.000000  0.000000  0.000000  -1.000000  2.000000

Eigenvvalor 3.732049
Eigenvvalor 2.999992
Eigenvvalor 1.998200
Eigenvvalor 0.543124
Eigenvvalor 0.924777
Segundos: 0.004000

```

Figura 3: Matriz tridiagonal con 5 nodos.

Estos resultados se compararon con otros obtenidos por la aplicación WolframAlpha,

```

Valores propios
λ1 ≈ 3,73205
λ2 = 3
λ3 = 2
λ4 = 1
λ5 ≈ 0,267949

```

Figura 4: Resultados con otra aplicación para 5 nodos.

donde se puede ver que los primeros términos los aproxima muy bien, pero los últimos aparecen en desorden, y sus resultados numéricos empeoran. En el caso del método de la potencia inversa para $N = 1000$, el resultado convergió rápido, y el eigenvalor mínimo que encontró se muestra en la fig. 5.

```

PS D:\OneDrive - Universidad de Guadalajara\Documents\maestria_computacion\primer_semestre\metodos_numericos\tarea5_v1>.
\la.exe

Eigenvvalor 0 : 0.000010
Segundos: 0.046000

```

Figura 5: Método de la potencia inversa para 1000 nodos.

y también se probó para una matriz de 5x5, y se encontraron:

```

PS D:\OneDrive - Universidad de Guadalajara\Documents\maestria_computacion\primer_semestre\metodos_numericos\tarea5_v1>.
cc inverse_power_method.c
PS D:\OneDrive - Universidad de Guadalajara\Documents\maestria_computacion\primer_semestre\metodos_numericos\tarea5_v1>.
\la.exe

Eigenvvalor 0 : 0.267949
Eigenvvalor 1 : 1.037701
Eigenvvalor 2 : 1.873493
Eigenvvalor 3 : 3.223994
Eigenvvalor 4 : 3.149461
Segundos: 0.426000

```

Figura 6: Potencia inversa para 5 nodos.

donde los primeros resultados son muy buenos, pero conforme se acerca al dominante, estos pierden precisión. Asimismo, para calcular el error en los métodos, se tomó la matriz con todos los valores positivos, y sus resultados se muestran en el Cuadro 1. Los resultados mostrados se compararon con los obtenidos utilizando la paquetería Numpy de Python.

n	$ \lambda - \lambda_{aprox} $	$ \phi_n - \phi_{n-aprox} $
1	$(4-3.998827) \approx 0.0011$	0.03693
1000	$(9.866240e-16-0.000039) \approx 3.89e-05$	0.03572

Cuadro 1: Resultados para $N = 1000$ con una matriz tridiagonal con todos los elementos positivos.

4. Discusión y conclusiones

Un error encontrado fue el orden en el que aparecían los valores conforme se buscaban los restantes. Aunque estos métodos son para encontrar valores más grandes o más chicos, no tiene sentido que no aparecieran en orden los siguientes ya que se quitaba la contribución del pasado más grande o del pasado más chico. Sin embargo, como se mostró en los resultados, esto apareció en el método de la potencia. Otra de las cosas fue el cambio de tiempo y de solución de ambos métodos. Por ejemplo, al utilizar la factorización de Cholesky para esta matriz, el método convergía rápido y con buenos resultados pese a la tolerancia, cuando este parámetro era importante en el método de la potencia. Además, los eigenvalores y eigenvectores para la matriz A con elementos con distinto signo no mostraba resultados correctos a partir del nodo 150, que también puede depender de la tolerancia del método y de los errores numéricos en cada iteración.

Referencias

[1] R.L. Burden, J.D. Faires, and A.M. Burden. *Numerical Analysis*. Cengage Learning, 2015.