

Metody numeryczne

Mateusz Kwolek


3. Dana jest macierz

$$\mathbf{A} = \begin{bmatrix} \frac{19}{12} & \frac{13}{12} & \frac{5}{6} & \frac{5}{6} & \frac{13}{12} & -\frac{17}{12} \\ \frac{13}{12} & \frac{13}{12} & \frac{5}{6} & \frac{5}{6} & -\frac{11}{12} & \frac{13}{12} \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} & -\frac{1}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{5}{6} & \frac{5}{6} & -\frac{1}{6} & \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{13}{12} & -\frac{11}{12} & \frac{5}{6} & \frac{5}{6} & \frac{13}{12} & \frac{13}{12} \\ -\frac{17}{12} & \frac{13}{12} & \frac{5}{6} & \frac{5}{6} & \frac{13}{12} & \frac{19}{12} \end{bmatrix}. \quad (3)$$

Przy użyciu metody potęgowej znajdź jej dwie największe na moduł wartości własne i odpowiadające im wektory własne.

1) Kod źródłowy

Do stworzenia programu użyłem języka Python ze względu na jego prostotę, bogactwo bibliotek oraz częste zastosowanie do wykonywania obliczeń matematycznych.



```

1  import numpy as np
2
3  A = [
4      [19/12, 13/12, 5/6, 5/6, 5/6, -17/12],
5      [13/12, 13/12, 5/6, -1/6, 5/6, -11/12],
6      [5/6, 5/6, 5/6, -1/6, 5/6, 5/6],
7      [5/6, -1/6, -1/6, -1/6, 5/6, 5/6],
8      [13/12, -11/12, 5/6, 5/6, 13/12, 13/12],
9      [-17/12, 13/12, 5/6, 5/6, 13/12, 19/12]
10 ]
11
12 # Metoda potęgowa (znalezienie jednej największej wartości)
13 def power_method(A, num_iter=1000, tol=1e-9):
14     n = len(A)
15     b_k = np.random.rand(n)
16
17     for _ in range(num_iter):
18         b_k1 = np.dot(A, b_k)
19         b_k1_norm = np.linalg.norm(b_k1)
20         b_k = b_k1 / b_k1_norm
21
22         if np.linalg.norm(np.dot(A, b_k) - b_k1_norm * b_k) < tol:
23             break
24
25     eigenvalue = b_k1_norm
26     eigenvector = b_k
27     return eigenvalue, eigenvector
28
29 # Deflacja macierzy względem wektora własnego i wartości własnej
30 def deflate(A, eigenvector, eigenvalue):
31     eigenvector = eigenvector.reshape(-1, 1)
32     A = A - eigenvalue * np.dot(eigenvector, eigenvector.T)
33     return A
34
35 # Znajdywanie największej wartości własnej i odpowiadającego jej wektora własnego
36 eigenvalue1, eigenvector1 = power_method(A)
37
38 # Deflacja macierzy A (w celu znalezienia pozostałej wartości własnej)
39 A_deflated = deflate(np.array(A), eigenvector1, eigenvalue1)
40
41 # Znajdywanie DRUGIEJ największej wartości własnej i odpowiadającego jej wektora własnego
42 eigenvalue2, eigenvector2 = power_method(A_deflated)
43
44 # Wyświetlenie wyników
45 print("Matrix A:\n", A)
46 print("Largest eigenvalue:", eigenvalue1)
47 print("Corresponding eigenvector:", eigenvector1)
48 print()
49 print("Second largest eigenvalue:", eigenvalue2)
50 print("Corresponding eigenvector:", eigenvector2)

```

2) Wynik

Largest eigenvalue: 3.5424072619316975

Corresponding eigenvector: [0.29148779 0.24928162 0.4739556 0.26979013 0.52192033 0.53204519]

Second largest eigenvalue: 3.112082463544934

Corresponding eigenvector: [0.69746207 0.5326996 0.16680416 -0.03306591 -0.13856201 -0.42621753]