SPRAWOZDANIE NUMERYCZNE

NUM6

Zadanie

Zadana jest macierz

$$\mathbf{M} = \begin{pmatrix} 8 & 1 & 0 & 0 \\ 1 & 7 & 2 & 0 \\ 0 & 2 & 6 & 3 \\ 0 & 0 & 3 & 5 \end{pmatrix}.$$

(a) Stosując metodę potęgową znajdź największą co do modułu wartość własną macierzy M oraz odpowiadający jej wektor własny. Na wykresie w skali logarytmicznej zilustruj zbieżność metody w funkcji ilości wykonanych iteracji.

(b) Stosując algorytm QR bez przesunięć, opisany w zadaniu nr 6, znajdź wszystkie wartości własne macierzy M. Zademonstruj, że macierze Ai upodabniają się do macierzy trójkątnej górnej w kolejnych iteracjach. Przeanalizuj i przedstaw na odpowiednim wykresie, jak elementy diagonalne macierzy Ai ewoluują w funkcji indeksu i.

(c) Zastanów się, czy zbieżność algorytmu z pkt. (a) i (b) jest zadowalająca. Jak można usprawnić te algorytmy?

Wyniki sprawdź używając wybranego pakietu algebry komputerowej lub biblioteki numerycznej.

Cel badania

Celem tego badania numerycznego jest zastosowanie i analiza dwóch różnych metod obliczania wartości własnych macierzy **M**:

(a) Metoda potęgowa:

Stosując metodę potęgową, celem jest znalezienie największej co do modułu wartości własnej macierzy **M** oraz odpowiadającego jej wektora własnego. Badanie ma na celu ilustrowanie zbieżności metody w funkcji liczby wykonanych iteracji. Poprzez przedstawienie wyników na wykresie w skali logarytmicznej, chcemy zobaczyć, jak szybko metoda potęgowa zbliża się do poprawnego rozwiązania.

Na samym początku wyznaczamy wektor początkowy, u nas to będzie wektor

$$z^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Dla tego wektora obliczamy wektor, na ktorym my będziemy mieli nasze przybliżone wartości własne

$$w^{(i)} = Az^{(i)} = \begin{bmatrix} 8 & 1 & 0 & 0 \\ 1 & 7 & 2 & 0 \\ 0 & 2 & 6 & 3 \\ 0 & 0 & 3 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Teraz musimy znaleźc największą wartość własną, czyli największą wśród wartości wektora

$$\lambda^{(i)} = max_j |w_j^{(i)}|$$

Wyliczamy błąd iteracji według wzoru

$$e^i = ||w^{(i)} - \lambda^{(i)} z^{(i)}||$$

Jest on nam potrzebny dla monitorowania w którym momencie algorytm musi się zatrzymać.

Na końcu wyliczamy wektor Z dla kolejnej iteracji:

$$z^{(i+1)} = w^{(i)} / \lambda^{(i)}$$

Powtarzamy to dopóki

$$e^{(i)} > \epsilon$$

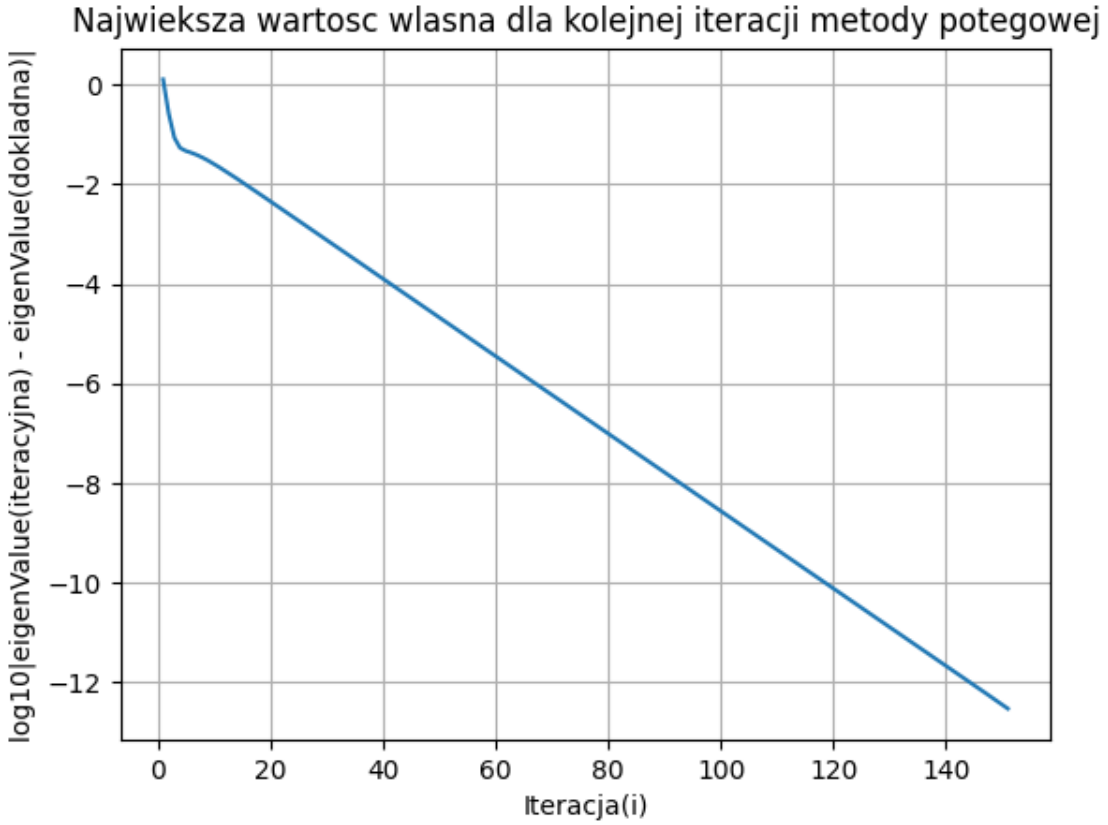
Wyniki

Dany wykres jest zrobiony dla wektoru startowego

$$z^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Następny wykres dla wektoru startowego

$$z^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$



Z drugiego wykresu widać, że dla metody potęgowej istotnym może być wybór wektoru startowego, co w naszym eksperymencie widocznie zmniejszyło ilość iteracji dla osiągnięcia potrzebnego nam wyniku. Dla przyspieszenia zbieżności wartości własnej możemy wybrać również inne wektory, które mogą zmniejszyć ilość iteracji dla obliczenia dokładnej wartości największej wartości własnej macierzy M.

Najwieksza wartosc wlasna obliczona metoda iteracyjna:

9.742393758888628

Odpowiedni wektor:

[-0.33272255 -0.57973369 -0.62856775 -0.39762688]

Najwieksza wartosc wlasna obliczona metoda biblioteczna:

9.742393758888332

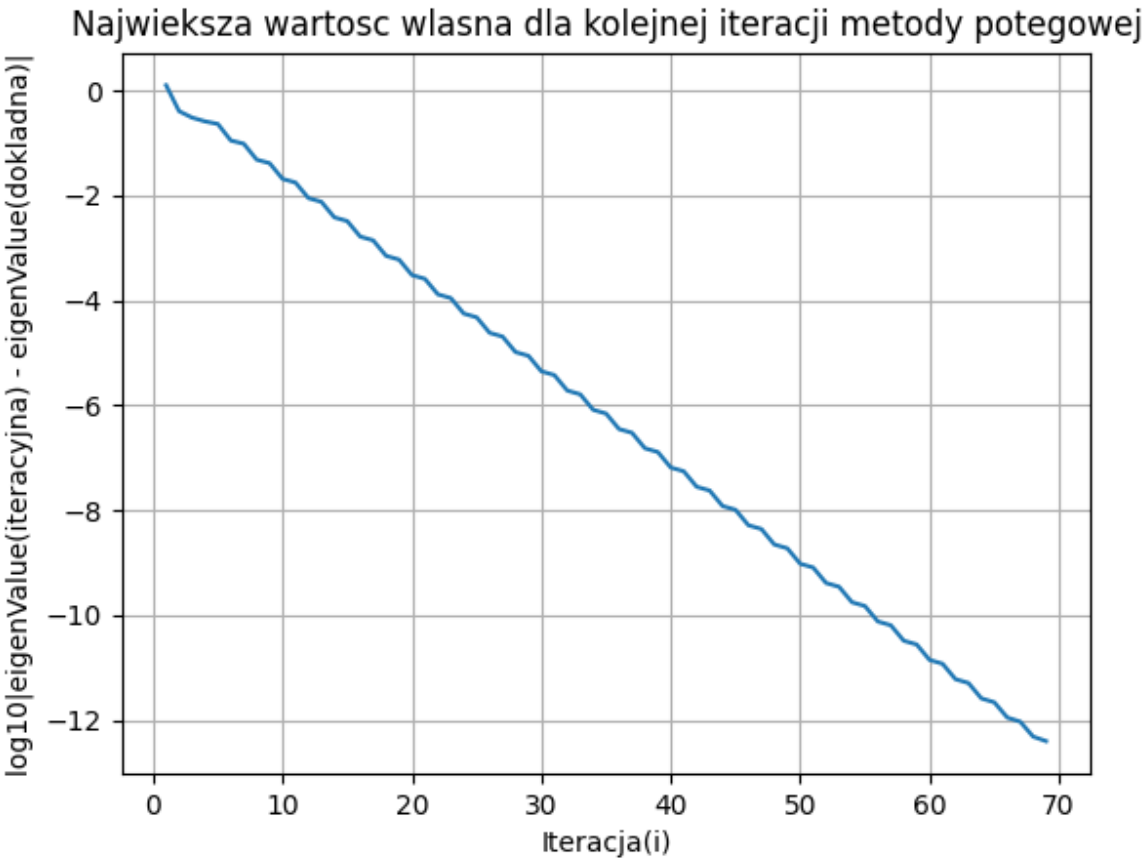
Odpowiedni wektor:

[-0.33272255 -0.57973369 -0.62856775 -0.39762688

Ustaliśmy, że dokładność wyniku musi być do 12 miejsca po przycinku, widzimy z wyniku, że wartości odpowiadają oczekiwanią.

Przyspieszenie działania metody potęgowej

Dodatkowo chciałbym omówić metoda przyśpieszenia metody potęgowej, która polega na przesunięciu macierzy o pełny skalar p, który trzeba odpowiednio dostosować do naszego przypadku. Okazuje się, że dla znacznego przyśpieszenia szybkości działania naszej metody p musi być równe sumie najmniejszej wartości własnej i grugiej największej, rozdzielonej na dwa. Wynika to z tego, że największy błąd w naszym równaniu iteracyjnym daje iloraz pierwszej i drugiej wartości własnej, a przesunięcie o taki skalar praktycznie zeruje nam tą wartość, co robi cały algorytm znacznie wydajniejszym.



Widać z wykresu, że dana zmiana przyśpieszyła naszą metodę dwukrotnie, co stanowi znaczną optymicację. Shodki są spowodowane tym, że podałem wartości własne z dużą precyzyjnością, co robi względny błąd przy obliczeniu iteracji bardziej widocznym.

(b) Algorytm QR bez przesunięć:

Stosując algorytm QR bez przesunięć, celem jest znalezienie wszystkich wartości własnych macierzy **M**. Badanie ma na celu zilustrowanie ewolucji macierzy **A_i**, które są generowane w kolejnych iteracjach algorytmu QR. Analizujemy, jak macierze **A_i** upodabniają się do macierzy trójkątnej górnej, a także jak elementy diagonalne tych macierzy ewoluują w funkcji indeksu iteracji.

W trakcie implementacji algorytmu QR dla znalezienia wszystkich wartości własnych korzystaliśmy z metod bibliotecznych numpy dla znalezienia rozkładu QR oraz dla dodawania i mnożenia macierzy. Algorytm jest zaimplementowany według następnego pomysłu:

Mamy rozkład QR, gdzie

$$A_1 = A$$
$$A_1 = Q_1 R_1$$
$$A_2 = R_1 Q_1 = Q_2 R_2$$
$$A_3 = R_2 Q_2 = Q_3 R_3$$

Kontynując podany rozkład możemy doprowadzić go do postaci

$$A_i = Q_i R_i$$
$$A_{i+1} = R_i Q_i = Q_i^T Q_i R_i Q_i \Rightarrow A_{i+1} = Q_i^T A_i Q_i$$

Z ostatniego wynika, że wartości na diagonali macierzy A są naszymi wartościamy własnymi w kolejnej iteracji.

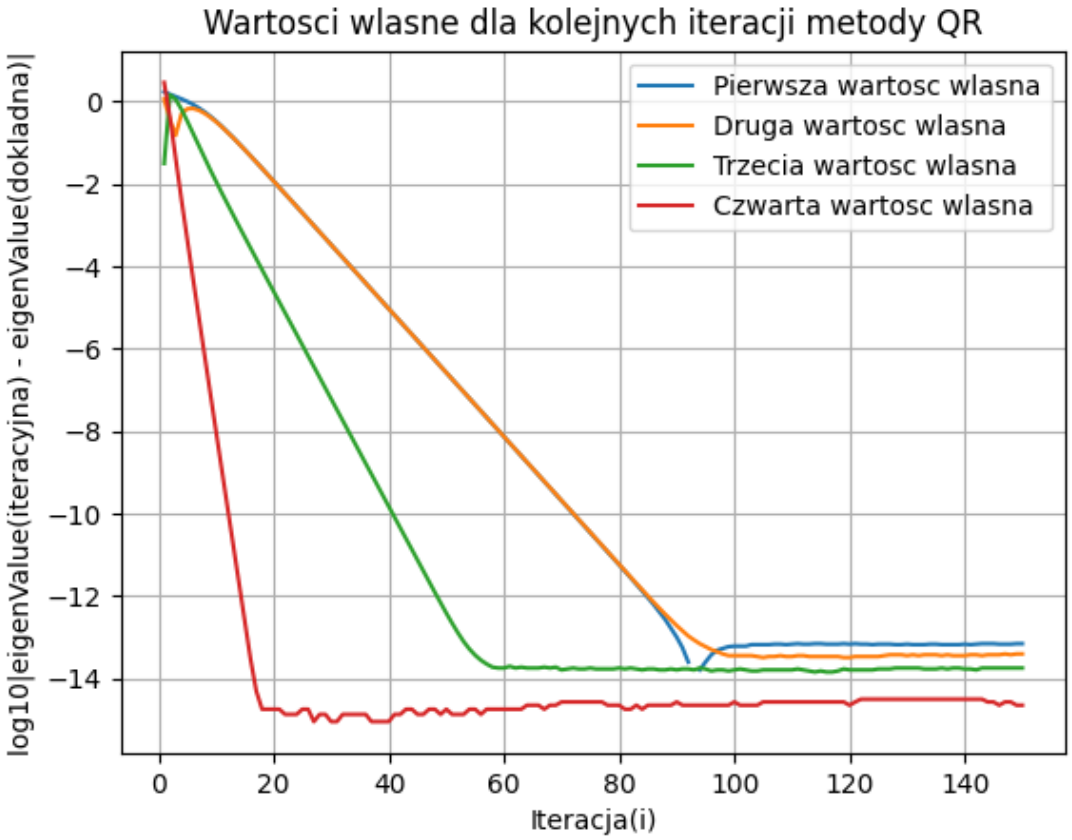
Rozpiszemy sobie teraz ostatni wyraz według podanej rekurencji, otrzymamy

$$A_{i+1} = Q_i^T A_i Q_i = Q_i^T Q_{i-1}^T A_{i-1} Q_{i-1} Q_i = Q_i^T Q_{i-1}^T Q_{i-2}^T A_{i-2} Q_{i-2} Q_{i-1} Q_i$$
$$A_{i+1} = Q_i^T Q_{i-1}^T Q_{i-2}^T \dots Q_1^T A_1 Q_1 \dots Q_{i-2} Q_{i-1} Q_i$$
$$Niech \quad P_i^T = Q_i^T Q_{i-1}^T Q_{i-2}^T \dots Q_1^T \quad i \quad P_i = Q_1 \dots Q_{i-2} Q_{i-1} Q_i$$
$$Wtedy \quad A_{i+1} = P_i^T A P_i$$

Widzimy teraz, że dla obliczenia kolejnych iteracji potrzebujemy jedynie macierzy Q oraz jej transponowania, zgodnie z tym i będziemy obliczać nasze wartości własne, które jak juz wiemy są położone na diagonali macierzy A.

Wyniki

Poniżej podaje zbieżność różnicy między dokładnym znaczeniem wartości własnych i ich przybliżeniem w kolejnych iteracjach.



**Wykres niebieski znajduję się pod wykresem pomarańczowym do 90 iteracji

Macierz M obliczony metoda QR:

$$\begin{bmatrix} 9.74239376e+00 & 1.11807596e-11 & -1.73604061e-17 & 3.76755119e-16 \\ 1.11809747e-11 & 8.14771771e+00 & 1.03408517e-16 & 5.23015418e-17 \\ 6.24935368e-17 & 7.25302222e-17 & 6.03172371e+00 & -1.61241114e-16 \\ [9.21724562e-16 & 5.56040656e-17 & -1.45691606e-16 & 2.07816482e+00 \end{bmatrix}$$

To jest wartosc wlasna macierzy M obliczony metoda QR:

9.7423937588884

To jest wektor wlasny macierzy M obliczony metoda QR:

[-0.33272255 -0.57973369 -0.62856775 -0.39762688]

To jest wartosc wlasna macierzy M obliczony metoda QR:

8.147717711696322

To jest wektor wlasny macierzy M obliczony metoda QR:

[0.8603918 0.12709511 -0.35726125 -0.34049551]

To jest wartosc wlasna macierzy M obliczony metoda QR:

6.031723710866654

To jest wektor wlasny macierzy M obliczony metoda QR:

[-0.38309775 0.75404221 -0.17351172 -0.50452962]

To jest wartosc wlasna macierzy M obliczony metoda QR:

2.078164818548761

To jest wektor wlasny macierzy M obliczony metoda QR:

[-0.04751607 0.28138235 -0.66870073 0.68658978]

Z wykresu oraz wyniku wnioskujemy, że od pewnego momentu wartości poza diagonalą stają się tak małe, że już nie są w stanie korygować wartości własne i błąd obliczeniowy dla rożnych wartości wynosi od 10e-13 do 10e-15, co jest zgodne z naszym epsilon, który ustaliliśmy.

Wektor wlasny macierzy M obliczony metoda biblioteczna:

[9.74239376 8.14771771 6.03172371 2.07816482]

Odpowiedni wektor:

[-0.33272255 -0.57973369 -0.62856775 -0.39762688]

Dla bibliotecznej reprezentacji liczb(8 miejsc po przecinku) warości są sobie równe. W rzeczywistości z wykresów widzimy, że błąd obliczeń występuje po 13-15 miejscu po przecinku.

Podbijając wnioski możemy stwierdzić, że dla podanej macierzy metoda QR jest bardizej optymalna jeżeli nie robimy dodatkowych założeń, ale dobierając odpowiedni wektor startowy i przesunięcie p możemy znacznie zmniejszyć ilośc iteracji dla metody potęgowej, co robi ją szybszą od QR.

Autor

Maksym Yankovenko