Zadanie numeryczne 06

Autor: Eryk Stępień

06.12.2023

Spis treści:

1. Problem

Zadana jest macierz

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

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 $A_i$upodabniają się do macierzy trójkątnej górnej w kolejnych iteracjach. Przeanalizuj i przedstaw na odpowiednim wykresie, jak elementy diagonalne macierzy A_i 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

2. Program

2.1 Użyte narzędzia

Program został napisany w języku Python 3.10. Przy zastosowaniu środowiska PyCharm 2023.2.2. Korzysta on z następujących bibliotek:

- Numpy
- Matplotlib.pyplot

2.2 Kompilacja i uruchomienie

W celu kompilacji należy wywołać poniższą komendę w terminalu:

python NUM6.py

2.3 Opis działania programu

2.3.1 Podpunkt a)

Rozwiązanie podpunktu a) w całości zostało umieszczone w funkcji *ANajwiekszyModul*. Największa wartość własna obliczana jest w następujący sposób:

$$\lambda_k = \frac{{y_{k-1}}^T A y_{k-1}}{{y_{k-1}}^T y_{k-1}}$$

$$y_k = \frac{A y_{k-1}}{\|A y_{k-1}\|}$$

Początkowym y jest dowolny $\|y\| = 1$. W przypadku mojego programu jest to $y = (1 \quad 0 \quad 0 \quad 0)$.

Powyższy wzór jest obliczany do osiągnięcia maksymalnej dozwolonej liczby iteracji lub do spełnienia warunku $\|\lambda_d - \lambda_k\| < dozwolona\ tolerancja$, gdzie λ_d jest dokładną wartością własną o największym module obliczoną za pomocą biblioteki numerycznej.

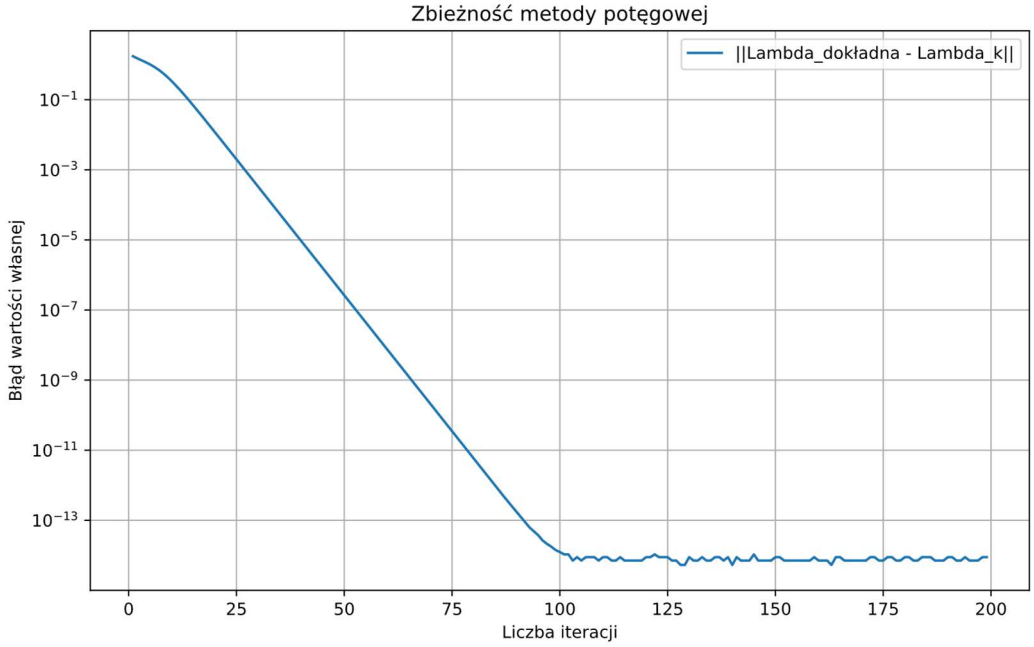
Po zakończeniu iteracji program wyświetla największą co do modułu wartość, odpowiadający jej wektor własny oraz tworzy wykres zależności $\|\lambda_d - \lambda_k\|$ od liczby iteracji w skali logarytmicznej.

Wyniki działania programu

$$Wartości\ własne\ obliczone\ za\ pomocą\ biblioteki\ numerycznej:$$
$$\lambda_1 = 9.74239376\ \lambda_2 = 8.14771771\ \lambda_3 = 6.03172371\ \lambda_4 = 2.07816428$$

$$Największa\ co\ do\ modułu\ wartość\ własna\ obliczona\ za\ pomocą\ algorytmu:$$
$$\lambda_{max} = 9.742393758888323$$

$$Odpowiadający\ jej\ wektor\ własny: [0.03415203\ 0.05950629\ 0.06451882\ 0.04081408]$$



Wykres obrazuje zbieżność lambdy w zależności od iteracji. Błąd wartości własnej stabilizuje się przy osiągnięciu około setnej iteracji.

2.3.2 Podpunkt b)

Rozwiązanie podpunktu b) w całości zostało umieszczone w funkcji *BQR.* Algorytm wykonuje kolejne iteracje do osiągnięcia maksymalnej dozwolonej liczby iteracji lub do momentu, gdy suma modułów elementów pod diagonalą jest mniejsza niż dana tolerancja. Jedna iteracja polega na:

$$A = QR$$

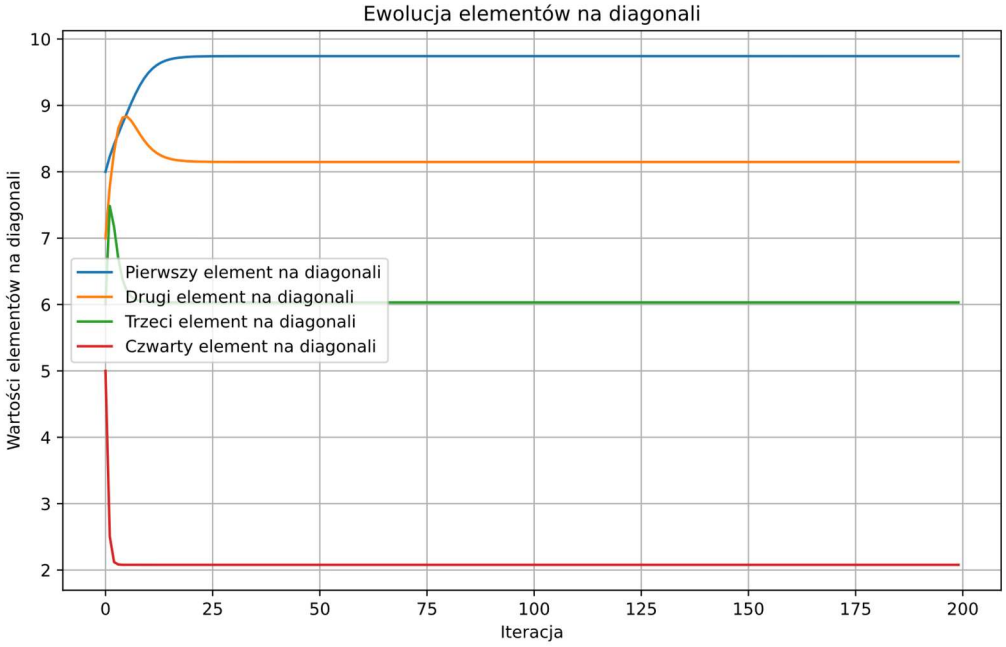
$$A = RQ$$

Po zakończeniu procesu iterowania na diagonali macierzy A będą znajdowały się wartości własne macierzy. Wynikiem działania algorytmu jest uzyskanie wszystkich wartości własnych macierzy:

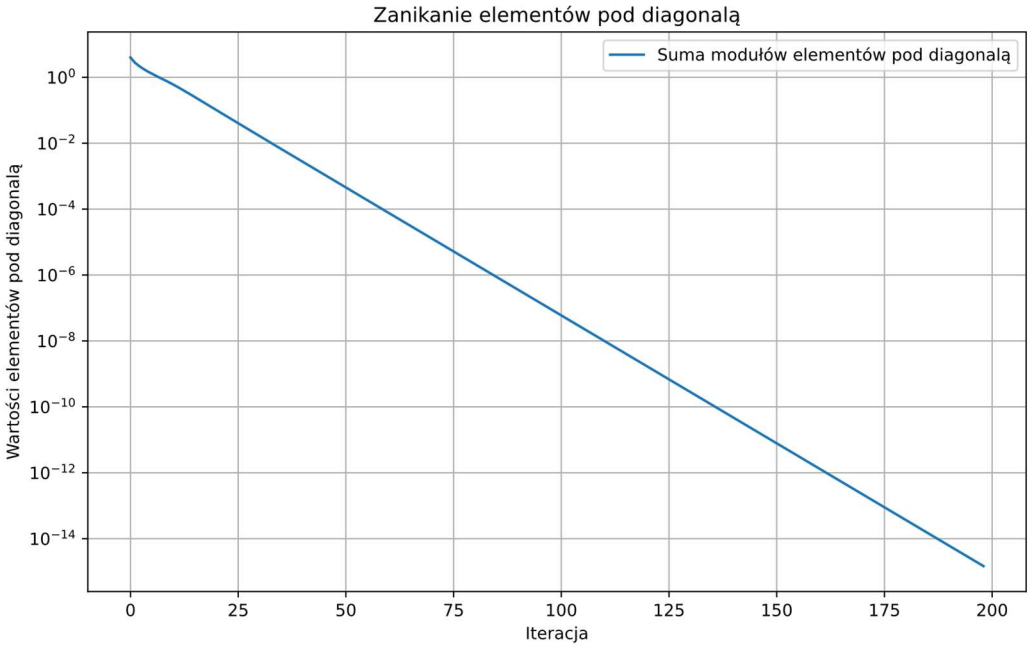
$$Wartości\ własne\ obliczone\ za\ pomocą\ biblioteki\ numerycznej:$$
$$\lambda_1 = 9.74239376\ \lambda_2 = 8.14771771\ \lambda_3 = 6.03172371\ \lambda_4 = 2.07816428$$

$$Wartości\ własne\ Macierzy\ M:$$
$$\lambda_1 = 9.74239376\ \lambda_2 = 8.14771771\ \lambda_3 = 6.03172371\ \lambda_4 = 2.07816428$$

Oraz utworzenie dwóch wykresów:



Pierwszy z nich obrazuje ewolucje elementów na diagonali. Łatwo z niego zauważyć, że najszybciej do końcowej wartości własnej zbliża się czwarty element diagonalny (a_{44}). Już po kilku iteracjach osiąga on wartość zbliżoną do końcowej. Kolejno element trzeci, drugi i pierwszy osiągają wartości zbliżone do końcowych.



W przypadku zachowania elementów pod diagonalą wraz z kolejnymi iteracjami możemy wykorzystać sumę modułów elementów pod diagonalą. Wykres obrazuje, że elementy te zanikają w sposób liniowy wraz z kolejnymi iteracjami. Macierz A dąży, więc do postaci macierzy trójkątnej górnej.

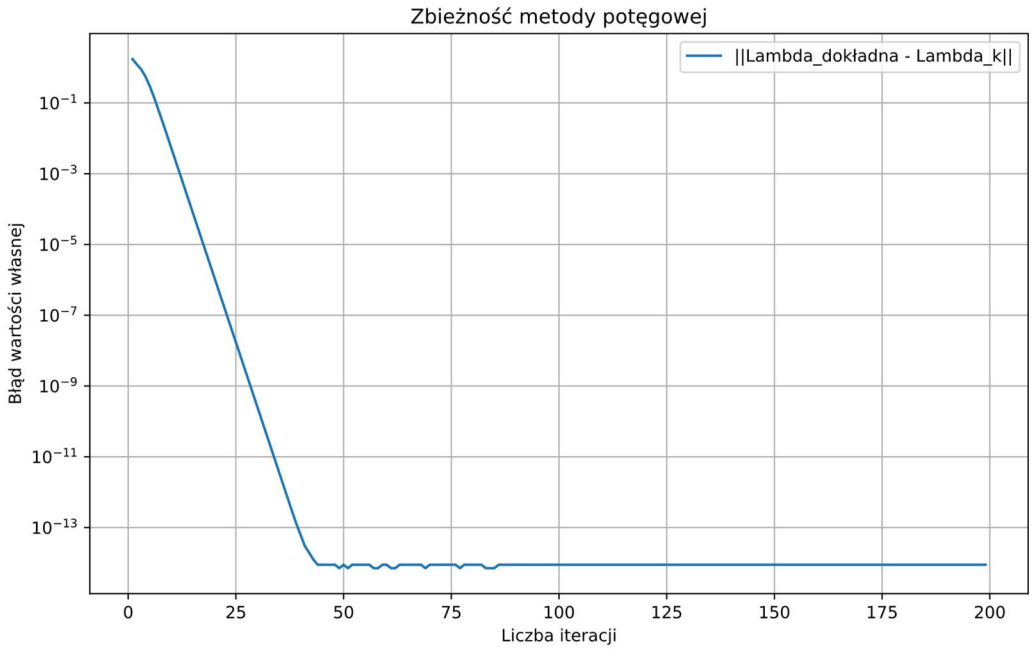
2.3.3 Podpunkt c)

W celu przyśpieszenia zbieżności algorytmu z pkt. A) i B). Możemy zastosować metodę Wilkinsona.

$$A - p\mathbb{I}$$

$$gdzie\ p = \frac{\lambda_2 - \lambda_n}{2}$$

Tak przekształcona macierz zapewnia przyśpieszenie metod.



Algorytm obliczający największą co do modułu wartość własną zapewnia teraz obliczenia λ_{max} już przed 50 iteracją.