Sprawozdanie NUM6

Szymon Tomaszewski

30 listopada 2024

1. Wstęp

1.1. Treść zadania

Zadana jest macierz

$$\mathbf{M} = \begin{pmatrix} 9 & 2 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}.$$

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 liczby wykonanych iteracji.

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

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

d) Wyniki sprawdź, używając wybranego pakietu algebry komputerowej lub biblioteki numerycznej.

1.2. Wprowadzenie

Metoda Potęgowa

Służy do wyznaczania największej co do modułu wartości własnej i odpowiadającemu jej wektorowi własnemu. Algorytm polega na wybraniu dowolnego niezerowego wektora początkowego y_1 o ustalonej normie $||y_1|| = 1$. Następnie obliczamy $z_k = M_{yk}$ i tworzymy nowy poprzez normalizację $z_k y_{k+1} = \frac{z_k}{||z_k||}$. Zbiega nam to do unormowanego wektora własnego macierzy M odpowiadającego największej wartości własnej. Gdy iteracja zbiegnie do punktu stałego, zmiana między kolejny wektorami własnymi jest bardzo mała, wtedy wartość własną obliczamy jako $\lambda_1 = ||z_k||$.

Algorytm QR
Metoda ta polega na dekompozycji macierzy M na macierz ortogonalną Q_i i macierz trójkątną górną R_i postaci $M_i = Q_i R_i,\ gdzie\ i = 1, 2,...$. Proces ten jest powtarzany aż macierz M_i stanie się dostatecznie podobna macierzy trójkątnej górnej. Elementy na diagonali tej macierzy odpowiadają wartościom własnym macierzy M.

W tej metodzie wykorzystywany jest fakt że wszystkie macierze M_i mają te same wartości własne a co za tym idzie powtarzając daną czynność macierz jest coraz mniej złożona. Zauważamy też, że macierz jest symetryczna więc algorytm QR będzie działał efektywnie.

2. Kod

Program został zaimplementowany w Pythonie 3, a do analizy wyników wykorzystano NumPy oraz Matplotlib. Program został podzielony na pliki aby poprawić czytelność.

Kryterium stopu dla obu programów jest osiągnięte gdy różnica między sąsiednimi wynikami jest mniejsza od zadanej tolerancji lub gdy osiągnie określoną ilość iteracji.

Struktura projektu

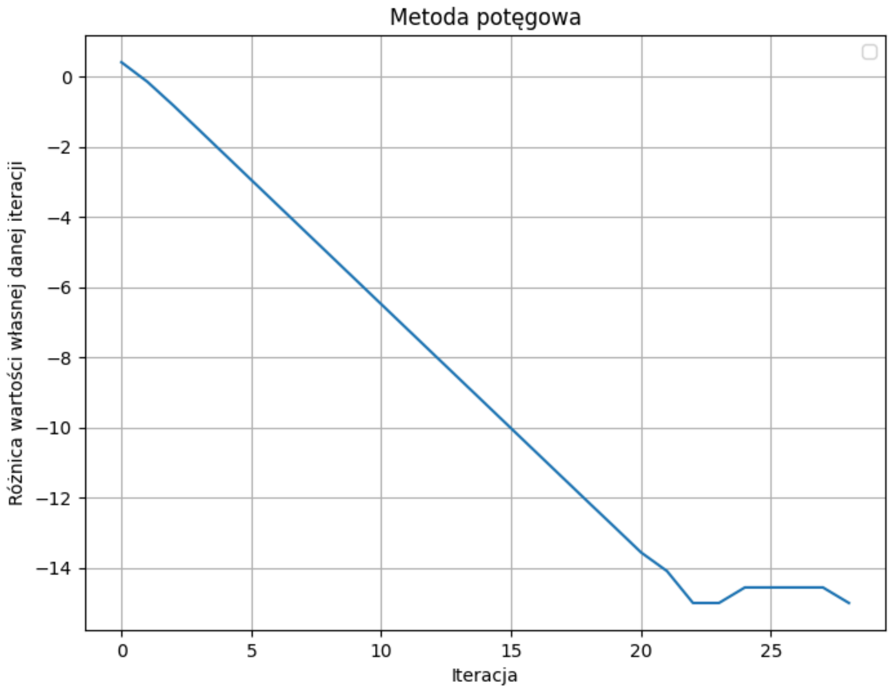
1. power.py — implementacja metody potęgowej.
2. qr.py — implementacja algorytmu QR.
3. check.py — sprawdzenie wyników

3. Wyniki

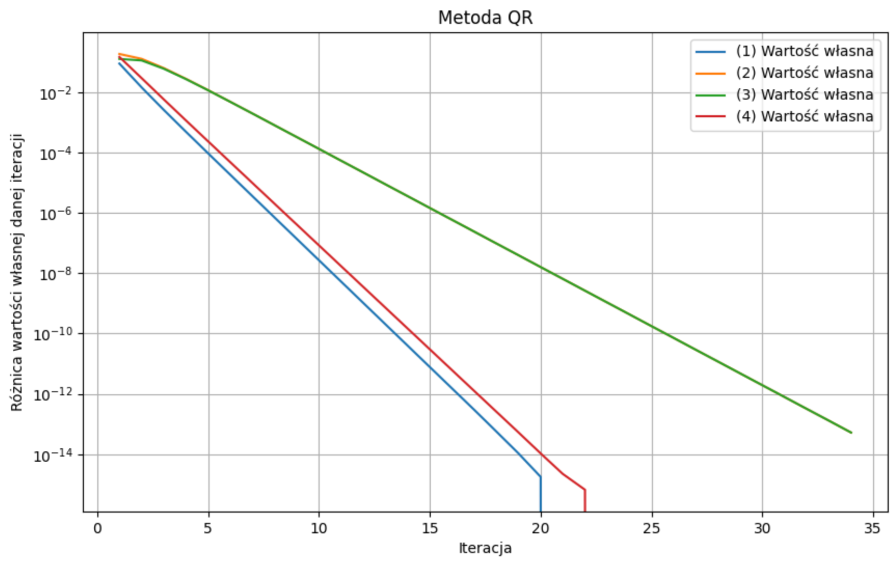
Wyniki przedstawione są na wykresach poniżej. Wartość błędu wyświetlana jest w obu przypadkach w skali logarytmicznej.

Podpunktu (a), program power.py, **metoda potęgowa**:
Największa wartość własna: 9.718548254119625
Znormalizowany wektor własny: [0.9398475755382383, 0.33766291735630644, 0.05124653505054892, 0.00663940077390805]

Podpunktu (b), program qr.py, **algorytm QR**:
Wartości własne: [9.718548254119618, 4.301704905012378, 2.740194113151937, 1.2395527277160612]



1. *Wykres dla podpunktu (a), program power.py.*



2. *Wykres dla podpunktu (b), program qr.py.*

4. **Dyskusja i Wnioski**

4.1. Wnioski

- Z wykresu widzimy, że błąd przybliżenia po około 23 operacjach schodzi poniżej 10^{-14} i jest on zależny od wyboru wartości własnych
- Metoda potęgowa i algorytm QR przydają się gdy nie potrzebujemy dokładności rozwiązania.
- Zaletą algorytmu QR jest jego zdolność do zachowania postaci macierzy.
- Jeśli przy pomocy metody potęgowej trzeba znaleźć kilka wartości metoda ta staje się bardziej czasochłonna.
- W metodzie potęgowej zbieżność jest bardzo wolna lub praktycznie znikoma jeśli wartości są bliskie modułu

Dodatkowo jeśli będziemy wypisywać wartości macierzy w każdej iteracji jesteśmy w stanie zauważyć, że macierz zbiega do trójkątnej górnej, wartości pod diagonalą zbliżają się z każdą iteracją do zera

Iteracja 10:
[[9.71854822e+00 4.26195047e-04 -4.30255314e-17 6.35706603e-17]
 [**4.26195047e-04** 4.30148228e+00 1.86447545e-02 -2.02719162e-16]
 [0.00000000e+00 **1.86447545e-02** 2.74041666e+00 3.97877494e-04]
 [0.00000000e+00 0.00000000e+00 **3.97877494e-04** 1.23955283e+00]]

Iteracja 40:
[[9.71854825e+00 1.03616928e-14 -4.42873919e-17 6.35662595e-17]
 [**1.02507116e-14** 4.30170491e+00 2.48221859e-08 -2.04168531e-16]
 [0.00000000e+00 **2.48221853e-08** 2.74019411e+00 1.82692902e-14]
 [0.00000000e+00 0.00000000e+00 **1.83750585e-14** 1.23955273e+00]]

Iteracja 63:
[[9.71854825e+00 1.10981252e-16 -4.42873937e-17 -6.35662595e-17]
 [**7.40723393e-23** 4.30170491e+00 7.77015439e-13 2.04168533e-16]
 [0.00000000e+00 **7.76383759e-13** 2.74019411e+00 1.05768135e-16]
 [0.00000000e+00 0.00000000e+00 **-2.18954204e-22** 1.23955273e+00]]

4.2. Poprawa wydajności

Działanie algorytmów pozostawia wiele do życzenia lecz na szczęście jesteśmy w stanie to poprawić.

Metoda Wilkinsona wspomaga liczenie jeśli pierwsze wyliczone wartości własne są do siebie zbliżone, tym samym szybkość zbieżności drastycznie maleje. Wilkinson wpadł na pomysł, że przy odejmowaniu jakiejś wartości x z diagonali macierzy M szybciej osiągniemy zbieżność. Jeśli jednak źle dobierzemy x otrzymamy efekt przeciwny do zamierzonego, dlatego ogólnym wzorem do wyznaczania x jest średnia arytmetyczna z wartości własnych y postaci: $x = \frac{y_i + y_{i+1}}{2}$, gdzie y_i to kolejne przybliżenia wartości własnych

Metodę QR złożoność, która sama w sobie z powodu rozkładu wynosi $O(n^3)$, można poprawić przekształcając macierz do postaci trójdiagonalnej lub Hessenberga.

Ciekawą metodą, którą warto napomnieć, jest metoda **ilorazu Rayleigha**. Umożliwia ona przyspieszenie zbieżności metody potęgowej, ale wymaga, aby macierz była symetryczna, ponieważ w takim przypadku jej wektory własne są ortogonalne. Metoda ta pozwala na precyzyjne przybliżenie największej wartości własnej macierzy.