Krzysztof Czarnowus	zadanie NUM6	grupa 3

1. Wstęp

Problem liczenia wartości własnych macierzy jest istotnym i przydatnym w wielu sytuacjach zagadnieniem, do którego podejście może być zróżnicowane w zależności od tego, czy potrzebne jest uzyskanie tylko największej z nich, czy wszystkich naraz.

Aby otrzymać pierwszą z wartości własnych macierzy A najkorzystniej jest zastosować metodę potęgową; polega ona na wybraniu losowego wektora y i wielokrotnego wykonywania mnożenia:

$$A\,y^{(i)} = y^{(i+1)} \tag{1}$$

wystarczająco wiele razy, aby wektor przestał zmieniać swoją wartość przy kolejnych iteracjach. Po każdym mnożeniu wektor należy normować; na samym końcu otrzymuje się dobre przybliżenie wektora własnego badanej macierzy A, ponieważ przyczynki pochodzące od wektorów o mniejszych wartościach własnych stopniowo zanikają. Wartość własna od tego wektora wynosi:

$$\lambda_1 = \frac{y^T A\, y}{y^T y} \tag{2}$$

Otrzymanie wszystkich wartości własnych macierzy jest w sytuacji problemem wyraźnie bardziej kosztownym obliczeniowo, ponieważ wymaga wielokrotnego wykonania rozkładu QR, mającego złożoność obliczeniową $O(n^3)$, chociaż dla macierzy trójdiagonalnych złożoność ta zmniejsza się do $O(n)$. Algorytm opiera się na poprawianiu macierzy A w kolejnych iteracjach:

$$A^{(i)} = Q^{(i)} R^{(i)} \tag{3}$$

$$A^{(i+1)} = R^{(i)} Q^{(i)} \tag{4}$$

tak długo, aż elementy diagonalne się ustabilizują na zadanym poziomie dokładności, a wszystkie wartości pod diagonalą zbliżą się do zera, co sprawi, że badana macierz przyjmie postać zbliżoną do trójkątnej górnej. Elementy diagonalne będą wówczas odpowiednim przybliżeniem wartości własnych macierzy.

2. Opracowanie wyników

Badanym zagadnieniem jest obliczenie wartości własnych macierzy o postaci:

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

W tym celu napisano program w języku Python, który generuje wektor o losowych wartościach z przedziału [0, 10], po czym za pomocą metody potęgowej wykonuje odpowiednie mnożenia tak długo, aż norma z różnicy wektorów $|| y^{(i+1)} - y^{(i)} ||$ przyjmie wartość mniejszą od 10^{-12}.

Ilość wykonanych iteracji różni się w zależności od dokładnej postaci wektora początkowego, za każdym razem jednak przyjmując wartość ok. 140. Możliwe jest poprawienie tej wartości przez zastosowanie metody Wilkinsona, polegającej na wykonywaniu obliczeń na macierzy:

$$M' = M - Ip \tag{5}$$

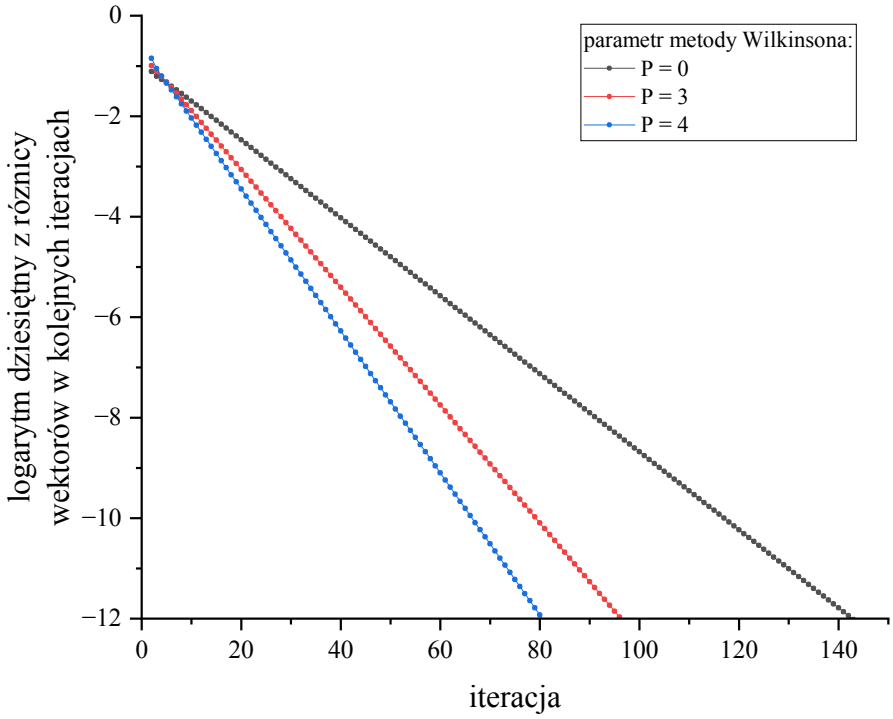
gdzie I jest macierzą jednostkową, a p jakąś wartością; wartości własne otrzymanej macierzy przyjmują postać:

$$\lambda' = \lambda - p \tag{6}$$

Przetestowano metodę Wilkinsona, aby poprawić szybkość metody potęgowej. Wykonano dwa kolejne powtórzenia obliczeń dla p = 3 oraz p = 4, po czym dodano wartość p do otrzymanych wartości własnych. Wyniki przedstawiono w tabeli 1., a szybkość zbieżności badanych wektorów własnych zilustrowano na rysunku 1.

Tabela 1. Zestawienie największej wartości własnej uzyskanej w wyniku obliczeń z zastosowaniem metody potęgowej oraz biblioteki NumPy.

metoda	największa wartość własna λ
metoda potęgowa bez stosowania metody Wilkinsona	9.74239376
metoda Wilkinsona o p = 3	9.74239376
metoda Wilkinsona o p = 4	9.74239376
biblioteczka NumPy	9.74239376



Rysunek 1. Zależność wartości logarytmu dziesiętnego kryterium zbieżności od liczby wykonanych iteracji dla trzech obliczeń liczenia największej wartości własnej macierzy M z użyciem metody potęgowej; dla dwóch z nich zastosowano metodę Wilkinsona.

Odpowiadający tej wartości wektor własny ma postać:

$$y_1 = [0.33272255;\ 0.57973369;\ 0.62856775;\ 0.39762688]^T$$

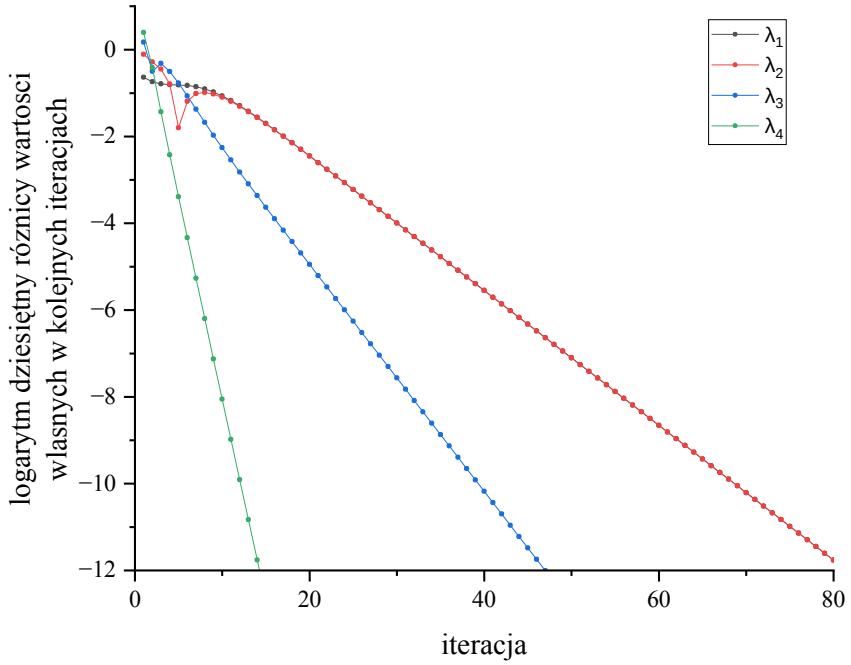
Napisano również program w języku Python służący wykonaniu algorytmu QR, aby upodobnić badaną macierz do macierzy trójkątnej górnej i z wartości na diagonali odczytać wszystkie wartości własne. Jako kryterium zbieżności przyjęto różnicę między wartościami na diagonali w kolejnych iteracjach mniejszą od 10^{-12}. Po wykonaniu 82 iteracji otrzymano macierz w postaci:

$$M^{(82)} = \begin{pmatrix} 9.74 & 1.78 \times 10^{-6} & 1.51 \times 10^{-15} & 2.83 \times 10^{-16} \\ 1.78 \times 10^{-6} & 8.15 & 4.29 \times 10^{-11} & 4.64 \times 10^{-16} \\ 0 & 4.29 \times 10^{-11} & 6.03 & 1.07 \times 10^{-15} \\ 0 & 0 & 3.29 \times 10^{-38} & 2.08 \end{pmatrix}$$

Zestawienie otrzymanych wartości diagonalnych z wartościami własnymi macierzy obliczonymi za pomocą biblioteki NumPy przedstawiono w tabeli 2., a na rysunku 3. zilustrowano zbieżność wszystkich wartości własnych w funkcji wykonanych iteracji.

Tabela 2. Zestawienie wartości własnych uzyskanych z zastosowaniem algorytmu QR oraz biblioteki NumPy.

Metoda	λ_1	λ_2	λ_3	λ_4
biblioteka NumPy	9.74239376	8.14771771	6.03172371	2.07816482
algorytm QR	9.74239376	8.14771771	6.03172371	2.07816482



Rysunek 2. Zależność logarytmu dziesiętnego z przyjętego kryterium zbieżności każdej z wartości własnych od liczby wykonanych iteracji.

3. Podsumowanie

Dla macierzy 4x4 zastosowano metodę potęgową oraz algorytm QR w celu obliczenia odpowiednio największej oraz wszystkich wartości własnych, w obu przypadkach uzyskując zgodność wyników z oczekiwanymi w stosunkowo szybkim czasie. Można się jednak spodziewać, że dla macierzy o większych wymiarach obliczenia wydłużyłyby się znacząco.

Zauważono, że dla metody potęgowej można znacznie poprawić szybkość osiągnięcia zbieżności poprzez zastosowanie metody Wilkinsona, polegającej na odjęciu od elementów na diagonali pewnej stałej wartości, po czym dodania jej do otrzymanej w wyniku obliczeń wartości własnej. Jest to algorytm wyraźnie mniej złożony obliczeniowo od algorytmu QR, służy jednak jedynie do obliczenia największej (bądź najmniejszej) z wartości własnych macierzy – aby otrzymać wszystkie, dużo bardziej korzystnie jest zastosować drugi z nich.