NUM 6

Łukasz Kowalik

1 Polecenie

Zadana jest macierz

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

(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 ilości 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 ewoluują w funkcji indeksu i.*

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

2 Wstęp

Celem zadania jest wyznaczenie wartości własnych zadanej macierzy M różnymi metodami numerycznymi, a także analiza ich zbieżności oraz ewentualnych problemów natury numerycznej. W części (a) wykorzystujemy **metodę potęgową** (ang. *power method*) do odnalezienia największej (w sensie wartości bezwzględnej) wartości własnej oraz odpowiadającego jej wektora własnego. Metoda potęgowa jest jedną z najprostszych iteracyjnych metod tego typu i często stanowi punkt wyjścia do badań nad metodami iteracyjnymi.

W części (b) stosujemy **iteracyjny algorytm QR** (bez przesunięć). Pozwala on w ogólności wyznaczać wszystkie wartości własne macierzy. Badamy przy tym, w jaki sposób w kolejnych krokach iteracji sama macierz „upodabnia się" do postaci trójkątnej górnej.

W części (c) mamy zastanowić się nad zbieżnością obu algorytmów i ewentualnymi usprawnieniami (np. zastosowanie przesunięć w algorytmie QR, wybór lepszego wektora startowego w metodzie potęgowej, itp.).

3 Opis metod

W niniejszej części omówimy **metodę potęgową**, **iteracyjny algorytm QR bez przesunięć** oraz **algorytm QR z przesunięciem Wilkinsona**, które posłużyły do wyznaczania wartości własnych macierzy.

3.1 Metoda potęgowa (ang. *power method*)

Metoda potęgowa służy do przybliżonego wyznaczania *największej (co do modułu) wartości własnej* macierzy. Przyjmuje się wektor początkowy $x^{(0)}$ i w każdym kroku wykonuje mnożenie przez macierz M oraz normalizację:

$$x^{(k+1)} = \frac{Mx^{(k)}}{\|Mx^{(k)}\|}.$$

Wartość własną w kroku iteracji można szacować jako

$$\lambda^{(k+1)} \approx (x^{(k)})^T \cdot \Big(M\, x^{(k)}\Big).$$

Jeśli istnieje wyraźna dominująca wartość własna (tj. taka, która w sensie bezwzględnym jest większa od pozostałych), metoda potęgowa zbiega do wektora własnego odpowiadającego tej wartości. Zaletą metody jest jej prostota implementacyjna; wadą — ograniczenie do jednej, największej (w sensie modułu) wartości własnej.

3.2 Algorytm QR bez przesunięć

Iteracyjny algorytm QR pozwala wyznaczać wszystkie wartości własne danej macierzy. Podstawowe kroki to:

$$A_k = Q_k\, R_k,$$
$$A_{k+1} = R_k\, Q_k,$$

gdzie $A_0 = M$, a w każdej iteracji dokonujemy rozkładu $A_k = Q_k\, R_k$ (np. metodą Gram–Schmidta lub Householdera). Wraz z postępem iteracji kolejne macierze A_k przybliżają macierz górnotrójkątną, a wartości własne są odczytywane z przekątnej uzyskanej macierzy.

3.3 Algorytm QR z przesunięciem Wilkinsona

Aby przyspieszyć zbieżność iteracji QR, stosuje się tzw. **przesunięcia**, dzięki którym algorytm unika powolnej redukcji w pobliżu wyróżnionych wartości własnych. **Przesunięcie Wilkinsona** (ang. *Wilkinson shift*) opiera się na analizie ostatnich dwóch elementów diagonalnych macierzy górnego bloku:

shift = obliczone m.in. z tzw. „dwóch ostatnich" elementów przekątnej i sąsiedniego poddiagonalnego, a następnie w danym kroku zamiast rozkładać A_k bezpośrednio, rozkładamy $\Big(A_k - \mu I\Big)$. Potem μ dodajemy z powrotem na przekątną po zakończeniu dekompozycji:

$$A_{k+1} = \Big(R_k\, Q_k\Big) + \mu I.$$

Dzięki tej modyfikacji liczba iteracji wymagana do osiągnięcia zadanej dokładności jest zazwyczaj zdecydowanie mniejsza, niż w wersji bez przesunięć.

Metody z przesunięciami np. Wilkinsona należą do najskuteczniejszych technik znajdowania wartości własnych za pomocą iteracji QR, ponieważ *znacznie* poprawiają szybkość zbieżności zwłaszcza wtedy, gdy różnice między wartościami własnymi nie są duże.

4 Wyniki

W tej sekcji prezentujemy rezultaty działania zaimplementowanych metod dla macierzy:

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

gdzie wszystkie obliczenia prowadzono z dokładnością rzędu 10^{-12}.

4.1 Metoda potęgowa

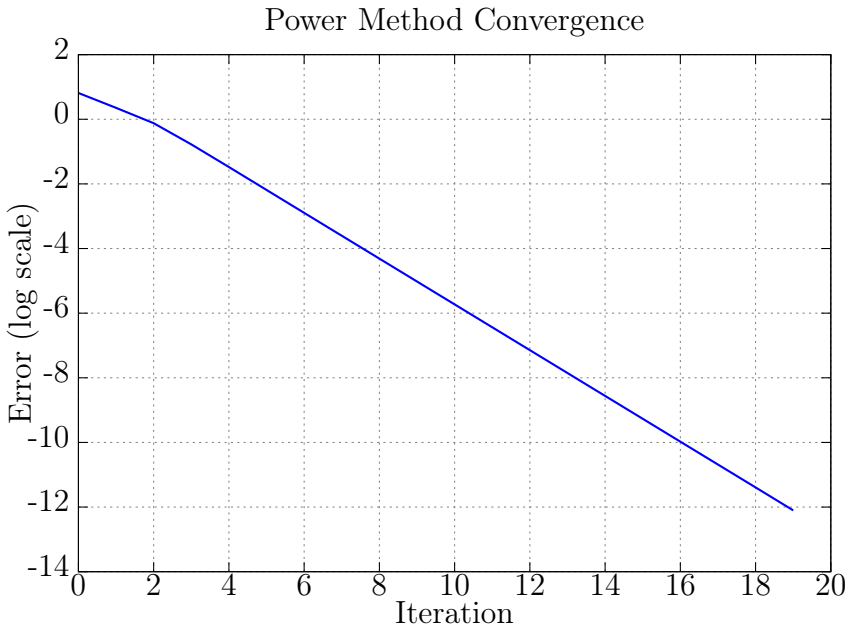
Po zastosowaniu *metody potęgowej* (ang. *power method*) do wyznaczenia **największej (co do modułu) wartości własnej** otrzymano:

- **Największa (co do modułu) wartość własna**:

$$\lambda_{\max} \approx 9.71855,$$

- **Wektor własny (znormalizowany)**:

$$\begin{pmatrix} 0.939848 \\ 0.337663 \\ 0.0512466 \\ 0.00663943 \end{pmatrix}.$$



Zbieżność metody potęgowej (skala logarytmiczna).

4.2 Algorytm QR bez przesunięć

Przy wykorzystaniu klasycznego **iteracyjnego algorytmu QR** (bez przesunięć) w kolejnych krokach otrzymywano macierze $\{A_k\}$, które *zbiegały do postaci górnotrójkątnej*. Po osiągnięciu zbieżności (tj. gdy elementy poddiagonalne spadły poniżej 10^{-12}) uzyskano macierz:

$$A_\infty = \begin{pmatrix} 9.71855 & -3.77734 \times 10^{-16} & 5.88785 \times 10^{-16} & -9.44747 \times 10^{-16} \\ 7.40723 \times 10^{-23} & 4.3017 & 7.7636 \times 10^{-13} & -9.35123 \times 10^{-16} \\ 0 & 7.76384 \times 10^{-13} & 2.74019 & -5.7915 \times 10^{-16} \\ 0 & 0 & 2.18954 \times 10^{-22} & 1.23955 \end{pmatrix},$$

z której można odczytać wartości własne:

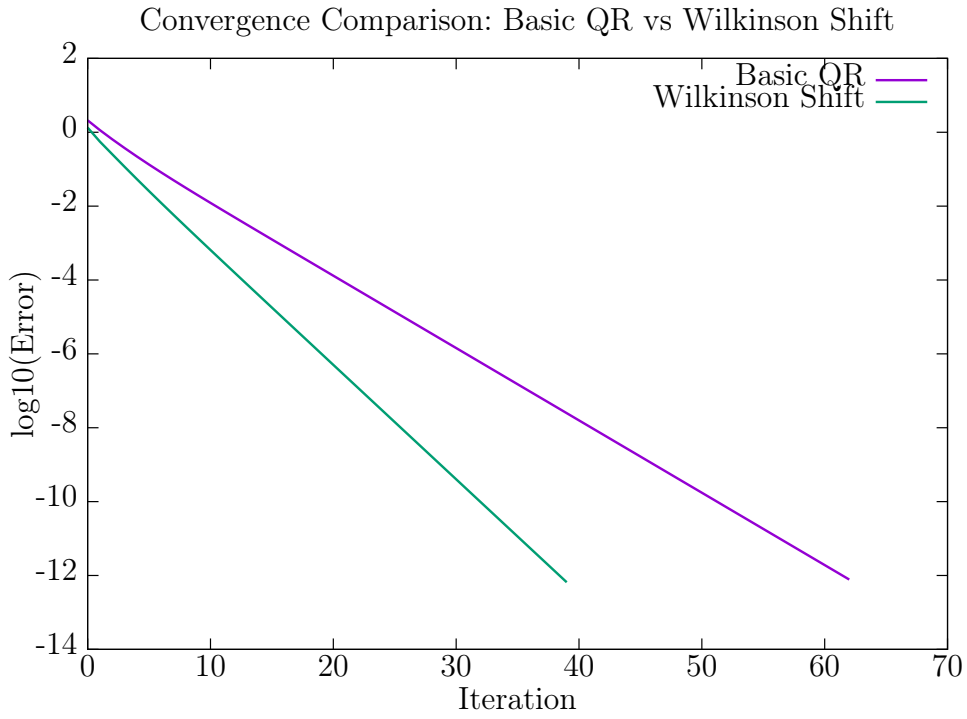
$$\lambda_0 \approx 9.71855, \quad \lambda_1 \approx 4.3017, \quad \lambda_2 \approx 2.74019, \quad \lambda_3 \approx 1.23955.$$

4.3 Algorytm QR z przesunięciem Wilkinsona

Zastosowanie *przesunięcia Wilkinsona* przyspieszyło proces redukcji elementów poddiagonalnych i już po mniejszej liczbie iteracji uzyskano macierz:

$$\begin{pmatrix} 9.71855 & 3.00875 \times 10^{-16} & -1.91496 \times 10^{-16} & -8.48545 \\ 2.90456 \times 10^{-18} & 4.3017 & 6.55562 \times 10^{-13} & 0.00200732 \\ 0 & 6.55356 \times 10^{-13} & 2.74019 & -4.88961 \times 10^{-6} \\ 0 & 0 & 0 & 1.23955 \end{pmatrix},$$

dając te same wartości własne na przekątnej. Algorytm ten jest zatem *znacznie szybszy* w zbieżności niż wariant bez przesunięć.



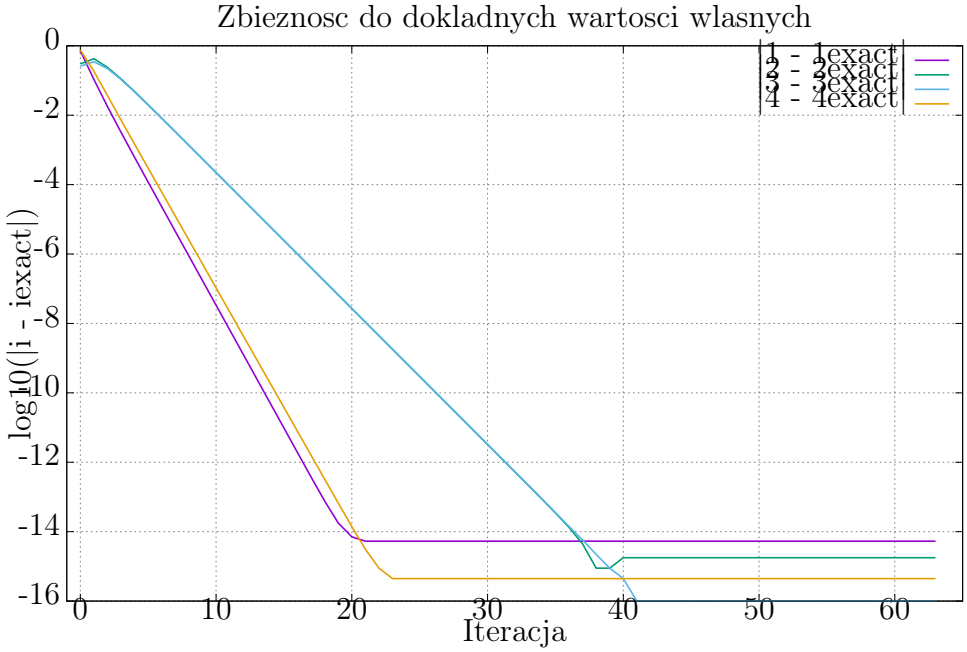
Porównanie szybkości zbieżności algorytmu QR bez przesunięcia i z przesunięciem Wilkinsona.

4.4 Weryfikacja wyników z biblioteką `Eigen` oraz porównanie wartości własnych

W celu weryfikacji skorzystano z biblioteki `Eigen`, która również zwróciła cztery wartości własne:

$$\lambda_0 \approx 9.71855, \quad \lambda_1 \approx 4.3017, \quad \lambda_2 \approx 2.74019, \quad \lambda_3 \approx 1.23955,$$

co *pokrywa się* z rezultatami otrzymanymi w naszych implementacjach. Dla zobrazowania różnic pomiędzy wartościami własnymi wyznaczonymi przez algorytm a bibliotekę `Eigen` (mierzonej jako $|\lambda_{\text{metoda}} - \lambda_{\text{eigen}}|$), przygotowano wykres w pliku:



Rożnica wartości własnych.

Na osi pionowej (w skali logarytmicznej) przedstawiono różnice względem dokładnych (z punktu widzenia `Eigen`) wartości własnych w kolejnych iteracjach.

4.5 Ewolucja macierzy w iteracjach (QR bez przesunięć)

Dla poglądowego przykładu prześledzono wygląd macierzy $\{A_k\}$ w kilku wybranych iteracjach:

$$\textbf{Iteracja } i = 0: \quad \begin{pmatrix} 9 & 2 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix},$$

$$\textbf{Iteracja } i = 32: \quad \begin{pmatrix} 9.71855 & 6.95691 \times 10^{-12} & 5.88785 \times 10^{-16} & -9.44747 \times 10^{-16} \\ 6.95729 \times 10^{-12} & 4.3017 & 9.15618 \times 10^{-7} & -9.35123 \times 10^{-16} \\ 0 & 9.15618 \times 10^{-7} & 2.74019 & 1.04794 \times 10^{-11} \\ 0 & 0 & 1.048 \times 10^{-11} & 1.23955 \end{pmatrix},$$

$$\textbf{Iteracja } i = 63: \quad \begin{pmatrix} 9.71855 & -3.77734 \times 10^{-16} & 5.88785 \times 10^{-16} & -9.44747 \times 10^{-16} \\ 7.40723 \times 10^{-23} & 4.3017 & 7.7636 \times 10^{-13} & -9.35123 \times 10^{-16} \\ 0 & 7.76384 \times 10^{-13} & 2.74019 & -5.7915 \times 10^{-16} \\ 0 & 0 & 2.18954 \times 10^{-22} & 1.23955 \end{pmatrix}.$$

Jak widać, elementy poddiagonalne dążą do zera w miarę postępu iteracji, aż w końcu macierz staje się (w granicach błędu rzędu 10^{-12}) górnotrójkątna.

5 Dyskusja wyników

Otrzymane wyniki pokazują, że:

- **Metoda potęgowa** (wyznaczająca największą co do modułu wartość własną) daje wynik $\lambda_{\max} \approx 9.71855$, co jest zbieżne z wartościami zwracanymi przez algorytm QR oraz bibliotekę `Eigen`. Metoda ta sprawdza się bardzo dobrze, gdy interesuje nas wyłącznie *dominująca* wartość własna.
- **Algorytm QR bez przesunięć** pozwala wyznaczyć wszystkie wartości własne, ale dla bardziej „zbliżonych" wartości własnych może wymagać wielu iteracji, zanim elementy poddiagonalne osiągną zadany poziom tolerancji (10^{-12}).
- **Algorytm QR z przesunięciem Wilkinsona** znacząco *przyspiesza* zbieżność, ponieważ odpowiednio dobrane przesunięcie μ (analizujące ostatni blok macierzy) przyspiesza „oddzielanie się" kolejnych wartości własnych. Liczba iteracji potrzebnych do uzyskania macierzy w postaci trójkątnej górnej jest z reguły *mniejsza* niż w wersji bez przesunięć.
- Porównanie z **biblioteką** `Eigen` potwierdza poprawność wyników – wszystkie otrzymane wartości własne (zarówno z metody potęgowej, jak i algorytmów QR) są zgodne z tym, co zwraca wyspecjalizowana biblioteka. Niewielkie różnice mogą pojawić się w dalszych cyfrach po przecinku w zależności od dokładności obliczeń i przyjętego kryterium stopu.
- **Ewolucja macierzy** w kolejnych iteracjach QR dobrze ilustruje zjawisko „zerowania się" elementów poddiagonalnych, co finalnie prowadzi do postaci górnotrójkątnej. Zastosowanie przesunięć (np. Wilkinsona) przyspiesza to „zerowanie" i tym samym – obniża całkowity koszt obliczeniowy.

Podsumowując, jeśli zależy nam tylko na największej (modułowo) wartości własnej, wystarczy metoda potęgowa. Natomiast do wyznaczenia pełnego spektrum lepiej użyć algorytmu QR. Wersja z przesunięciami Wilkinsona jest bardziej efektywna od podstawowej. W praktyce zaawansowane biblioteki (np. `Eigen`) implementują różne warianty algorytmu QR z przesunięciami i technikami poprawy stabilności (np. transformacje Householdera zamiast klasycznego Gram–Schmidta).

6 Zakończenie

W ramach zadania dokonano analizy wyznaczania wartości własnych macierzy metodą potęgową oraz iteracyjnym algorytmem QR (zarówno w wersji bez przesunięć, jak i z przesunięciem Wilkinsona). Przeprowadzone obliczenia oraz porównania z biblioteką `Eigen` wykazały następujące wnioski:

- **Zbieżność metody potęgowej** jest zazwyczaj bardzo dobra *dla największej (co do modułu) wartości własnej*, szczególnie gdy ta wartość własna jest wyraźnie oddzielona (dominuje) nad pozostałymi.

- **Iteracja QR bez przesunięć** umożliwia wyznaczenie całego spektrum wartości własnych, jednak przybliżanie kolejnych wartości własnych może wymagać dość dużej liczby iteracji, szczególnie w przypadku, gdy macierz ma wartości własne bliskie sobie.

- **Zastosowanie przesunięć (np. Wilkinsona)** w algorytmie QR *znacznie przyspiesza* proces zerowania elementów poddiagonalnych i pozwala w mniejszej liczbie iteracji osiągnąć macierz niemal górnotrójkątną. Z tego powodu zbieżność algorytmu QR z przesunięciem można uznać za *bardziej zadowalającą* niż w przypadku wersji bez przesunięć.

- **Usprawnienia algorytmów**:

  – *Algorytm QR* warto wzbogacić o przesunięcia Wilkinsona oraz stosować stabilniejsze metody rozkładu (np. Householdera zamiast klasycznego Gram–Schmidta).

- **Weryfikacja wyników** przy użyciu `Eigen` wykazuje zgodność wszystkich otrzymanych wartości własnych, co potwierdza poprawność implementacji i uzyskanych rezultatów.

Odpowiadając na postawione w zadaniu pytania:

1. *„Czy zbieżność algorytmu z pkt. (a) i (b) jest zadowalająca?"* Tak, obie metody zbiegały do prawidłowych rozwiązań. Metoda potęgowa sprawdziła się przy wyznaczaniu największej wartości własnej, a iteracja QR – przy wszystkich wartościach własnych. Warto jednak zauważyć, że iteracja QR bez przesunięć potrafi być stosunkowo wolna, jeśli wartości własne są do siebie zbliżone.

2. *„Jak można usprawnić te algorytmy?"* Metodę potęgową można usprawnić, zmieniając wektor początkowy lub stosując wariant z przybliżeniem Wilkinsona. Natomiast algorytm QR zdecydowanie przyspiesza, gdy wprowadzi się przesunięcia Wilkinsona oraz użyje bardziej stabilnych metod rozkładu (np. transformacje Householdera).

3. *„Wyniki sprawdź używając wybranego pakietu algebry komputerowej lub biblioteki numerycznej."* Porównanie z biblioteką `Eigen` potwierdziło, że otrzymane wartości własne są zbieżne z wynikami uzyskanymi z powszechnie uznawanego pakietu numerycznego, co stanowi dodatkowe potwierdzenie poprawności implementacji.