Adam Przemysław Chojecki, 298814, grupa 1, projekt 2, zadanie 55

Wstęp

Zadanie polegało na zaimplementowaniu algorytmu QR do szukania wartości własnych zespolonych macierzy Hessenberga. Rozkład QR wyznaczono stosując obroty Givensa.

Cała implementacja i testy przeprowadzone zostały za pomocą oprogramowania MATLAB, a kod źródłowy dostępny jest publicznie na repozytorium GitHub [1].

Krótki opis zaimplementowanej metody

Metoda na swoim wejściu dostaje macierz A Hessenberga. Iteracyjnie wykonuje jej rozkład $A_1 = A = QR$, gdzie macierz Q jest macierzą ortogonalną, a macierz R jest górnotrójkątna. Następnie obliczana jest kolejna $A_2 = RQ$, która również jest Hessenberga. Iteracje te wykonywane są do zbieżności.

Dokładniejszy opis algorytmu można znaleźć w [2] w rozdziale 5.5.

Zbieżność jest przyśpieszana poprzez sprawdzanie, czy odpowiednie elementy poddiagonali są pomijalnie małe. Jeśli pierwsza od dołu wartość z poddiagonali jest pomijalnie mała, to ostatnia wartość na diagonali uznano za wartość własną macierzy A i kontynuowano obliczenia dla obciętej macierzy bez ostatniego wiersza i ostatniej kolumny. Podobnie gdy druga od dołu wartość z poddiagonali jest pomijalnie mała, to policzono wartości własne końcowej macierz 2x2 i uznano je za wartości własne macierzy A, i kontynuowano obliczenia dla obciętej macierzy bez ostatnich dwóch wierszy i ostatnich dwóch kolumn.

Inne potencjalne metody przyśpieszania działania algorytmu można znaleźć w [4].

Macierz Q nie była zapisywana w pamięci komputera w całości, a jedynie wektory wartości c i s potrzebne do opisania obrotów Givensa.

Warto zauważyć, że rozkład dowolnej $A = QR \in \mathbb{C}^{n\times n}$ ma złożoność $o(n^3)$ (licząc jako ilość mnożeń), natomiast gdy macierz A jest Hessenberga, $o(n^2)$.

Warto zwrócić uwagę, że każdą macierz $B \in \mathbb{C}^{n\times n}$ można sprowadzić do postaci Hessenberga. Oznacza to, że algorytm zaimplementowany w ramach tego projektu może być z powodzeniem stosowany do znajdowania wartości własnych dowolnej macierzy, po odpowiednim jej przygotowaniu.

Eksperymenty numeryczne

Liczba iteracji

Na rysunku 1 pokazano histogram liczby iteracji QR, które algorytm potrzebował, aby znaleźć kolejną wartość własną. Warto zwrócić uwagę na logarytmiczną skalę na osi Y.

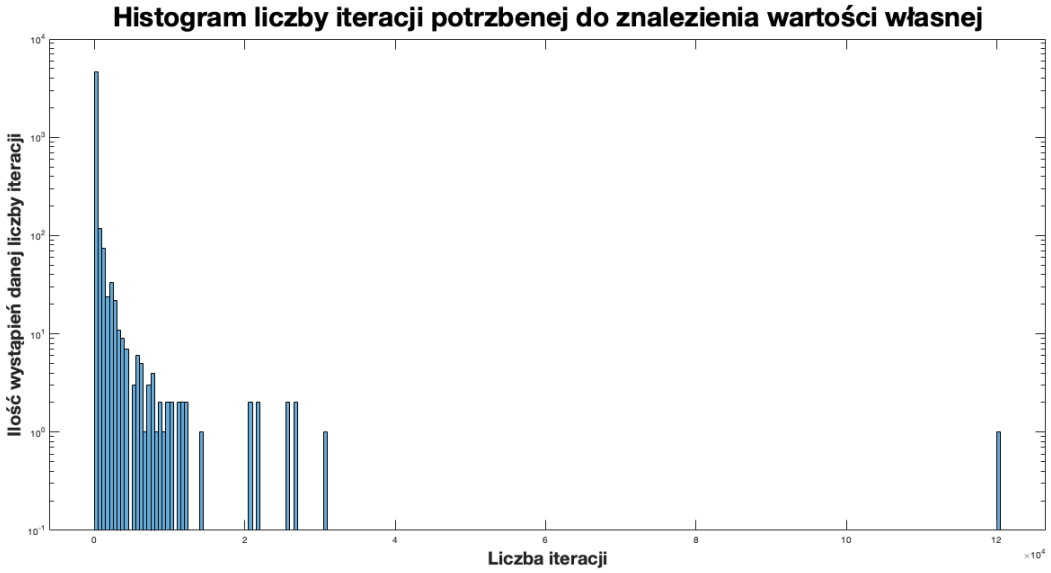
Na wykresie widać np. że bardzo często wystarczyło mało (poniżej 500) iteracji. Okazuje się, że 75% razy wystarczyła jedna iteracja. Prawdopodobnie kolejna wartość własna już była mniejsza niż aktualnie sprawdzana, bądź bliska niej.

Poza tym widoczna jest pojedyncza wartość własna, która została znaleziona dopiero po ponad 120000 iteracji. Stało się tak z powodu nieszczęśliwego wylosowania się wartości własnych blisko siebie. Widzimy więc, że rozkład czasu oczekiwania na kolejną wartość własną ma ciężki ogon i czasem będzie wymagać dużej liczby iteracji.

Ostatnia wartość z poddiagonali

Doktor Keller w swoim skrypcie [3] w podrozdziale 7.4 twierdzi, że ostatni element poddiagonali ($a^{(k)}_{n,n-1}$) zazwyczaj najszybciej staje się pomijalnie mały. W tym teście sprawdzono empirycznie prawdziwość tego stwierdzenia.

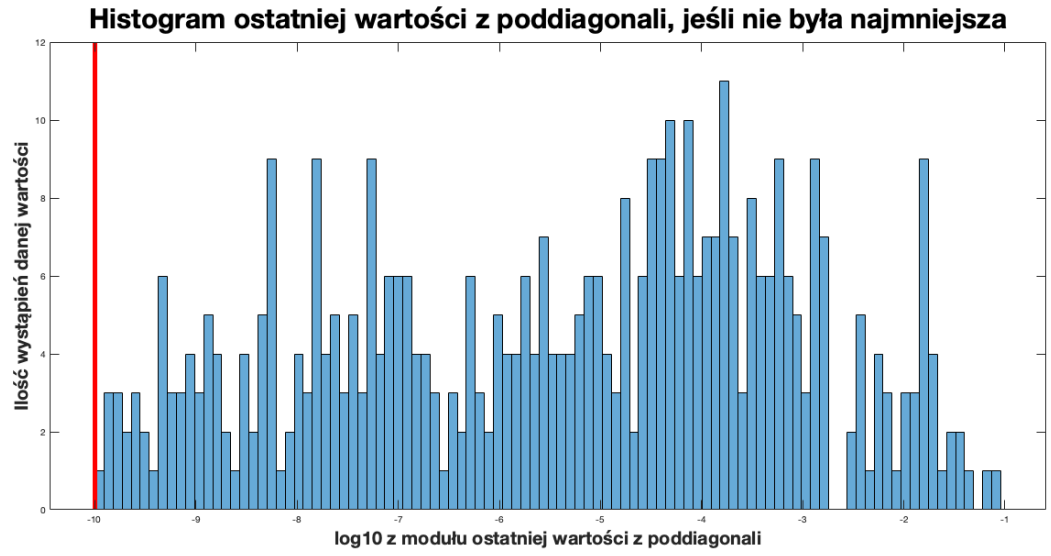
Macierze $A \in \mathbb{R}^{50\times 50}$ do tego eksperymentu wylosowano tak, aby ich wartości własne były niezależnie i jednostajnie wylosowane z rozkładu $U(-1, 1)$. Wylosowano 1000 takich macierzy. Na każdej z nich wykonywano iteracje algorytmu QR tak długo aż któraś z wartości na poddia-



Rysunek 1: Histogram pokazujący ile iteracji algorytmu QR było potrzeba, by znaleźć kolejną wartość własną.

gonali była co do modułu mniejsza niż 10^{-10}. Gdy tak się stało to zapisywano wartość modułu ostatniej z liczb z poddiagonali.

Okazało się, że w 567 z 1000 przypadków, to właśnie ta ostatnia liczba była mniejsza niż 10^{-10}. Można więc stwierdzić, że zazwyczaj tak się dzieje. Wartości pozostałych przypadków pokazano na wykresie 2. Czerwoną linią na nim zaznaczono moment obcięcia. Nie widzimy na nim żadnej większej zależności wśród tych danych.



Rysunek 2: Histogram pokazujący rozłożenie wartości 433 przypadków, gdy to nie ostatnia wartość z poddiagonali była najmniejsza. Czerwoną kreską zaznaczono moment obcięcia - na lewo od niego znajduje się 567 obserwacji, gdy to ta wartość była najmniejsza.

Bibliografia

[1] Przemysław Chojecki. *Strona GitHub zawierająca kod do tego projektu.* https://github.com/PrzeChoj/numerki. [Online; 09-01-2023].

[2] David R. Kincaid i E. Ward Cheney. *Numerical Analysis: Mathematics of Scientific Computing.* Pacific Grove, Calif.: Brooks/Cole, 1991. ISBN: 0534130143 9780534130145.

[3] dr Paweł Keller. *Skrypt Metody Numeryczne 2.* http://pages.mini.pw.edu.pl/~wrobeli/PWMN2_zima_2022-23/mn2_pliki/PKeller_Skrypt_MN2.pdf. [Online; 09-01-2023], Język Angielski.

[4] Gorka Eraña Robles. *Implementing the QR algorithm for efficiently computing matrix eigenvalues and eigenvectors.* https://addi.ehu.es/bitstream/handle/10810/26427/TFG_Erana_Robles_Gorka.pdf?sequence=1. [Online; 09-01-2023], Język Angielski.

Uwaga

W kodzie źródłowym brak jest polskich znaków. Spowodowane jest to negatywnym doświadczeniem autora z problemami jakie sprawia używanie ich w kodach źródłowych.