Wyznaczanie wartości własnych macierzy z wykorzystaniem oprogramowania Julia

Autorzy: Jarosław Królik (284363), Mateusz Derszniak (284293)

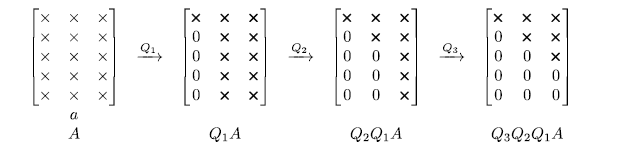
# Streszczenie

Wyznaczanie wartości własnych macierzy to jeden z podstawowych i najbardziej czasochłonnych etapów projektowania nowoczesnych struktur informatycznych. Z problemem tym możemy się spotkać także realizując i wprowadzając w życie innowacyjne pomysły inżynierskie. Dobranie skutecznej metody obliczania wartości własnych macierzy to gwarancja pozytywnego wyniku i optymalnego czasu przetwarzania programu. W raporcie zostaną rozważone różne algorytmy obliczania wartości własnych oraz wektorów własnych macierzy. Algorytmy te zostaną porównane pod względem czasu działania oraz zajętości pamięciowej.

Z definicji, jeżeli przekształcenie **A** przekształca prostą w siebie, to mówimy, że **v** jest wektorem własnym przekształcenia **A**. Oznacza to:

dla pewnej liczby rzeczywistej **λ**, zwanej wartością własną związaną z wektorem własnym **v**. Innymi słowami wektory i własności własne są to wielkości opisujące endomorfizm (przekształcenie liniowe) danej przestrzeni liniowej. Wektor własny można rozumieć jako wektor, którego kierunek nie zmienia się po przekształceniu go endomorfizmem, zmianie ulega jedynie jego długość. Wartość własna może być rozumiana jako skala podobieństwa wektora przed przekształceniem do wektora będącego wynikiem endomorfizmu. Przekształcenia liniowe to jedno z podstawowych zastosowań tych właśnie wielkości. Kolejnym zastosowaniem jest diagonalizacja macierzy [1], pozwala ona na łatwe potęgowanie i pierwiastkowanie macierzy, co otwiera nowe możliwości do rozwiązywania wielu problemów inżynierskich. Zastosowanie możemy znaleźć również w algorytmach PCA [1], stosowanych do kompresji sygnałów i danych. Algorytm ten jest stosowany między innymi w uczeniu maszynowym i sieciach neuronowych, w których pokłada się coraz większe nadzieje oraz stawia się im coraz bardziej wymagające zadania. W tej właśnie dziedzinie możemy doszukać się kolejnego algorytmu wykorzystującego wartości własne. Jest to algorytm ICA [2], stosowany do rozkładu mieszaniny sygnałów na sygnały oryginalne. Algorytm ten możemy znaleźć w codziennym życiu, ponieważ podstawowym zastosowaniem tego algorytmu jest rozdzielanie sygnałów mowy w telefonii komórkowej. Wychodząc z dziedzin informatycznych a przechodząc do automatyki, odnajdziemy własności własne w wielu przykładach rozwiązywania równań różniczkowych, których wykorzystywanie jest używane w regulatorach PID [1], stanowiących integralną cześć sterowników PLC. Głównym założeniem tych układów sterowania jest wydajność, dlatego implementacja w nich najlepszych algorytmów wyznaczania wartości własnych jest tak ważna.

Dotychczasowe badania dotyczące wyznaczania wartości własnych macierzy sprowadzają się do: **metod potęgowych** [5] które są wykorzystywane dla macierzy o wartościach rzeczywistych. Metoda ta, do wyprowadzenia wartości i wektora własnego macierzy, wykorzystuje szereg mnożeń na dowolnym wektorze z bazy w celu wyznaczenia głównej składowej szukanego wektora, **metod Lanczos** [6] które są zoptymalizowanym algorytmem bazującym na metodzie potęgowej. Działanie algorytmu polega na szukaniu nowego wektora bazowego w każdej iteracji, **metod QR** [7] która wywodzi się z metody LR, ale w celu zapewnienia stabilności numerycznej macierz L została zamieniona na macierz ortogonalną. Działanie metody polega na otrzymaniu ciągu podobnych do siebie macierzy, m**etod Householder** [7] wykorzystującej do wyznaczenia wartości własnych transformacje geometryczne lustrzanego odbicia, **metod gradientu sprzężonego** [7]będącą kolejnym przykładem metody iteracyjnej**, metod Jacobiego** [8]bazującej na przekształceniu przez podobieństwo, **metod Hilberta** [9] polegająca na iteracyjnym wykorzystaniu macierzy Hilberta do wyznaczanie wartości własnych macierz.

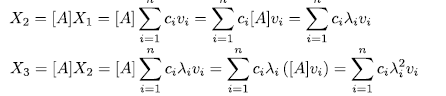


Jako główne założenie raportu zostało postawione porównanie działania algorytmów wyznaczania wartości własnej. Porównywane algorytmy muszą różnić się od siebie specyfiką działania, dlatego zostały wybrane 3 algorytmy przedstawiające odmienne drogi do wyznaczenia wartości własnej macierzy. Tymi metodami są: **metoda potęgowa** opierająca się na prostych metodach iteracyjnych, **metoda Householdera**, której głównym celem jest redukcja macierzy do postaci trójdiagonalnej oraz **metoda QR Grama -Schmidta.**

**Metoda potęgowa** wykorzystuje własność:



oraz własność, że każdy wektor może być wyrażony za pomocą kombinacji liniowej bazy zbudowanej z wektorów własnych:

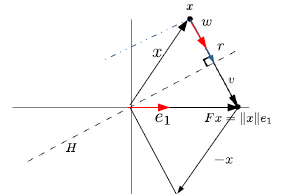


Co pozwala zapisać wyrażenie rekurencyjnie wektora własnego a następnie przekształcić do postaci:



**Houseldera** aby przekształcić macierz do postaci trójdiagonalnej wykorzystuje mnożenie macierzy **A** przez macierze unitarne. W tej metodzie macierzami unitarnymi stasowane są specjalnie dobrane odbicia prowadzące do eliminacji wartości poniżej diagonali.

Kluczem tej metody jest dobór odpowiednich odbić takich, aby przekształcany wektor x był równoległy do pierwszego wersora e1.



Wersor e1 będzie specjalnie dobierany tak, że będzie miał tylko pierwszy element niezerowy. Aby tego dokonać wykorzystamy odbicie względem płaszczyzny zdefiniowanej przez wektor:



Zakładając, że aby wykonać odbicie względem hiperpłaszczyzny ortogonalnej do wektora v, musimy wymnożyć go przez macierz unitarną:

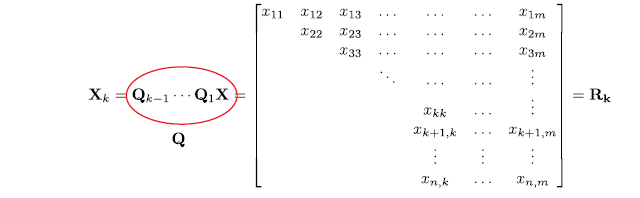


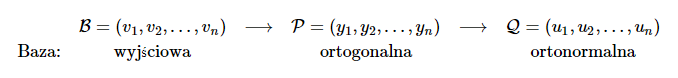
Możemy zauważyć, że:



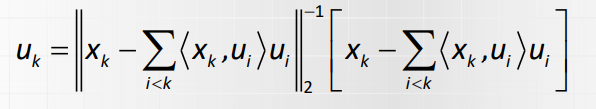
I wyprowadzić transformację Householdera:  

Która pomoże nam ostatecznie skonstruować ciąg odbić :

**Metoda QR** zakłada zbudowanie bazy ortonormalnej z dowolnej podprzestrzeni liniowej za pomocą procedury ortogonizacji Grama-Schmidta, zgodnie ze schematem:



Procedura ta pozwala zbudować ciąg ortonormalnych złożonych z kombinacji liniowych wektorów, który można wyrazić wzorem:



Dla liniowo niezależnych wektorów x1, x2, …, xn

Następnym etapem prowadzącym do wyznaczenia wartości własnych macierzy jest faktoryzacja, z użyciem macierzy wynikowej procedury G-S. Faktoryzacja polega na przedstawieniu danych w postaci:

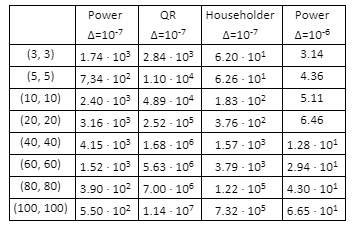
Dane w tej postaci można wyliczyć i w efekcie wyznaczyć wartości własne macierzy.

Po przedstawieniu badanych algorytmów należy zdefiniować zagadnienia jakie będą badane w raporcie. Aby otrzymać jasną i rzetelną odpowiedź który z badanych algorytmów jest najlepszy. Zdecydowano się wyznaczyć, jak zmieniają się czasy trwania oraz zajętość pamięciowa poszczególnych algorytmów w funkcji stopnia macierzy danych wejściowych.

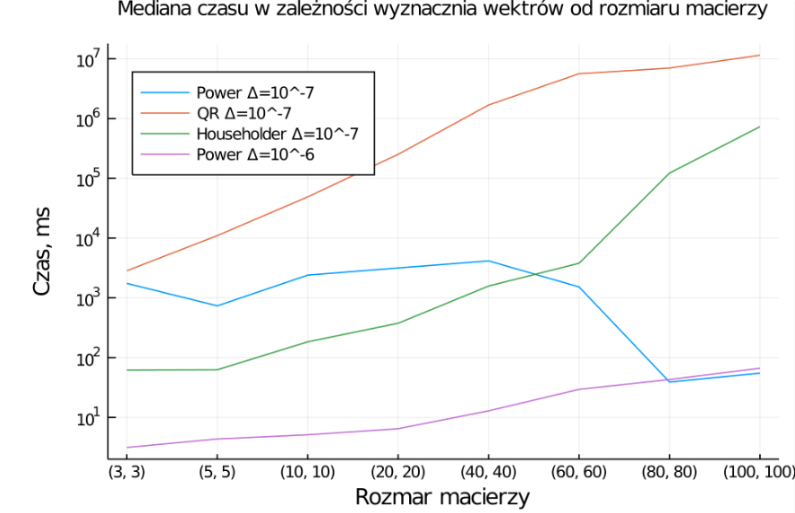
# Faza testów

Do przeprowadzenia testów zdecydowano się na wykorzystanie języka programowania Julia. Jest to stosunkowo nowy język programowania, który został stworzony głównie do rozwiązywania problemów naukowej natury. Język pozwala na szybkie tworzenie nowych bibliotek i posiada składnie przyjazną obliczeniom matematycznym. Ponadto, posiada porównywalne wyniki pod względem szybkości obliczeniowej porównując do języka C. W naszym przypadku został on połączony ze środowiskiem Atom z zaadaptowanymi bibliotekami: LinearAlgebra, Plots oraz BenchmarkTools, które pozwoliło na uzyskanie w pełni funkcjonalnego stanowiska badawczego. Pierwszym etapem przeprowadzonych badań było przygotowanie danych testujących. Dane te będą reprezentowane przez macierze kwadratowe o stopniach: 3, 5, 10, 20, 40, 60, 80, 100. Dane wejściowe, aby były czytelne i uporządkowane zostały umieszczone w specjalnie przygotowanej macierzy. Aby zwiększyć wiarygodność rezultatów dla każdego rozmiaru macierz, wartości zostały wygenerowane 21 razy. Do losowania liczb wykorzystano funkcję *rand,* która losuje wartości z przecidzału [0,1). Po przygotowaniu danych testowy, została utworzona grupa BenchmarkGroup w której zawarte są testów wydajności dla różnych kombinacji algorytmów wyznaczania wartości własnych macierzy oraz rozmiaru macierzy. Podczas testów zwrócono uwagę na czas jaki potrzebował algorytm do obliczeń oraz na zajętość pamięciową. Dla danej kombinacji (metoda wyznaczania oraz jedna wylosowana macierz) funkcja wykonująca benchmark w zależności od rozmiaru macierzy wykonywała od kilkunastu do kilkuset wyznaczeni wartości własnej. Spośród wszystkich uzyskanych rezultatów dla jednej kombinacji brano pod uwagę wartość środkową (mediana). Nieznaczna część wyników (prawdopodobnie ze względu na czas dostępu do dysku), wymagała znacznie dłuższego czasu na wyznaczenie wartości własnych. Z tego powodu uwzględniano medianę zamiast wartości średnie. Wartości środkowe zawarto w Tabeli 1 oraz Tabeli 2

Tabela 1 Czas trwania algorytmów w ms.



W kolumnach tabeli zostały przedstawione analizowane algorytmy wyznaczania wartości własnej macierzy, a w kolejnych wierszach można znaleźć coraz to większe stopnie macierzy wejściowych.

Język Julia został wyposażony w bibliotekę, której użycie pozwala na wizualizację uzyskanych danych. Dane z tabeli zostały zwizualizowane z użyciem biblioteki Plots oprogramowania Julia.

Na wykresie możemy zobaczyć 4 łamane obrazujące medianę czasu przetwarzania w funkcji rozmiaru macierzy. Możemy zauważyć, że metoda potęgowa została zawarta na wykresie dwa razy. Jest to metoda iteracyjna w której czas trwania jest zależny od ilości iteracji. Ilość iteracji w naszym przypadku jest uzależniona od zmiany wartości własnych macierzy w kolejnych 2 iteracjach. Wykres zawiera 2 przypadki tej metody, dla których różnice w kształcie wykresu były największe.

Powtarzając metodę wyznaczania danych została wygenerowana Tabela 2. zawierająca zajętość pamięciową badanych algorytmów. Dane potrzebne do stworzenia tabeli odpowiadają przedstawionym powyżej czasom trwania algorytmów.

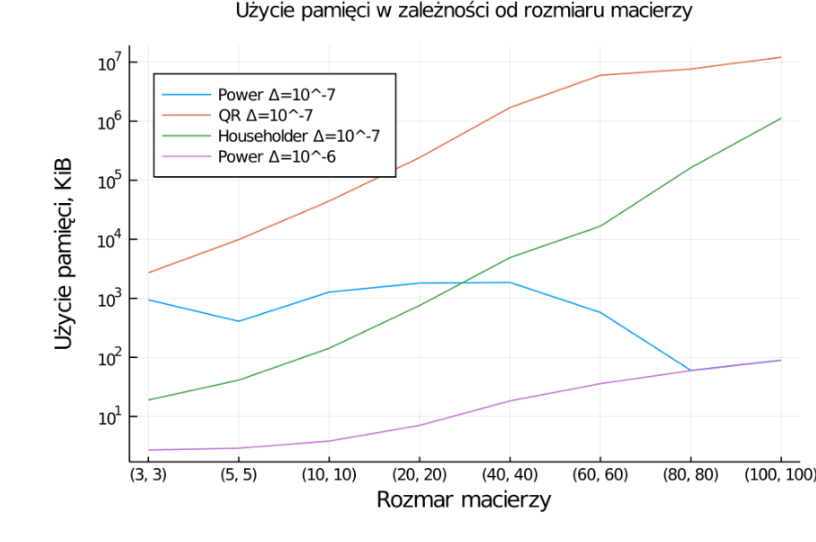
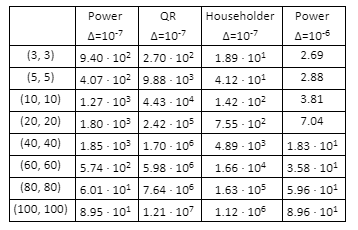
Ponownie dane zostały zwizualizowane w języku programowania Julia i przedstawione na poniższym wykresie.

Tabela 2

# Wnioski

Analizując sporządzone wykresy możemy stwierdzić, że algorytmem, który najszybciej wyznaczył wartości własne we wszystkich rodzajach macierzy był algorytm potęgowy. Ta metoda wyznaczania, potrzebowała również najmniejszej ilości pamięci. Szukając zależności pomiędzy czasem i zajętością pamięciową możemy zauważyć związek, że czas przetwarzania algorytmu jest proporcjonalny do zajmowanej przez algorytm pamięci. Zwycięskiemu algorytmowi nie można jednoznacznie przypisać miana najlepszego algorytmu. Dokonując dogłębnej analizy możemy zaważyć, że metoda potęgowa potrafi wyznaczyć jedynie główną wartość własną macierzy, kiedy metody Househldera i QR wszystkie. Kolejnym spostrzeżeniem może być znaczne wydłużenie czasu pracy przy zmianie parametru akceptowalnej wartości własnych. Przebieg wyznaczonej krzywej dla teoretycznej większej dokładności jest dość kontrowersyjny, co podważa poprawność wykonania próby.

Podsumowując, metoda potęgowa pozwala na bardzo szybkie wyznaczenie tylko wybranych wartości własnych. Jeżeli zależy nam na wyznaczeniu wszystkich wartości własnych z dużą precyzją, należy skorzystać z metody Hauseholdera, która okazała się szybszą metodą niż QR.

Programy sporządzone podczas testów zostały dołączone do raportu. Źródła zawierające sprawdzane algorytmy: Householder QR [10], QR algorithm for eigenvalues [11], Power Method with Inverse & Rayleigh [12]. Ponadto dodadkowe informacje o wartościach własnych macierzy były zaczerpnięte ze strony studia informatycznego [13] oraz materiałów „Algorytmy w inżynierii danych” [14]

# Bibliografia

|  |  |
| --- | --- |
| [1] | A. Pieper, M. Kreutzer, A. Alvermann i M. Galgon, „High-performance implementation of Chebyshev filter diagonalization for interior eigenvalue computations,” 2016. |
| [2] | M. Zhai, „The PLC Signals’ Noise Mitigating Algorithm with PCA,” 2017. |
| [3] | M. N. Patil, B. Iyer i R. Arya, „Performance Evaluation of PCA and ICA Algorithm for Facial Expression Recognition Application,” 2016. |
| [4] | K. Rajagopal i A. S. Guessas Laarem: Anitha Karthikeyan, „FPGA implementation of adaptive sliding mode control and genetically optimized PID control for fractional-order induction motor system with uncertain load,” 2017. |
| [5] | M. Tammen, I. Kodrasi i S. Doclo, „COMPLEXITY REDUCTION OF EIGENVALUE DECOMPOSITION-BASED DIFFUSE POWER SPECTRAL DENSITY ESTIMATORS USING THE POWER METHOD,” 2018. |
| [6] | A. R. d. Faria, „Adaptation of the Lanczos Algorithm for the Solution of Buckling Eigenvalue Problems,” 2018. |
| [7] | T. Lyche, „Numerical Linear Algebra and Matrix Factorizations,” 2020. |
| [8] | Z.-W. Sun, „Generalized inverse eigenvalue problems for augmented periodic Jacobi Matrices,” 2019. |
| [9] | R. R. Sharma i R. B. Pachori, „A New Method for Non-stationary Signal Analysis using Eigenvalue Decomposition of the Hankel Matrix and Hilbert Transform,” 2017. |
| [10] | [Online]. Available: https://www.youtube.com/watch?v=d-yPM-bxREs. |
| [11] | [Online]. Available: https://www.youtube.com/watch?v=\_neGVEBjLJA. |
| [12] | [Online]. Available: https://www.youtube.com/watch?v=LHlg\_lfihiA. |
| [13] | [Online]. Available: http://wazniak.mimuw.edu.pl/index.php?title=MN13. |
| [14] | [Online]. Available: https://isod.ee.pw.edu.pl/isod-stud/?wicket:bookmarkablePage=:isod.app.courseinfo.CourseInfoPage&idCourseDef=3846. |