# Metody Numeryczne – Zadanie NUM6

Bartosz Bochniak

## Wstęp

### 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?

Wyniki sprawdź używając wybranego pakietu algebry komputerowej lub biblioteki numerycznej.

## Rozwiązanie

### Metoda potęgowa

Macierz M jest macierzą symetryczną ($M \in \mathbb{R}^{4\times4},\ M = M^T$). Jej wartości własne oznaczone są jako liczby $\lambda_i,\ i \in \{1,2,3,4\}$. Dodatkowo, kolejnym z założeń jest to, że wartości własne są dodatnie, oraz uporządkowane: $\lambda_1 > \lambda_2 > \lambda_3 > \lambda_4 > 0$.

Dla powyższych warunków dobrany zostaje wektor v, taki, że:

$$v = \sum_{i=1}^{4} \beta_i e_i$$

W powyższym zapisie $e_i$ to wersor w i-tym wymiarze. (Wszystkie wersory $e_i$ są liniowo niezależne oraz ich normy wynoszą 1.)

Następnym krokiem metody potęgowej jest obliczenie M * v:

$$Mv = M\sum_{i=1}^{4} \beta_i e_i = \sum_{i=1}^{4} \beta_i M e_i = \sum_{i=1}^{4} \beta_i \lambda_i e_i$$

Następną iteracją metody potęgowej będzie $M^2 * v$, itd.:

$$M^2 v = M * Mv = M \sum_{i=1}^{4} \beta_i \lambda_i e_i = \sum_{i=1}^{4} \beta_i \lambda_i^2 e_i$$

$$\vdots$$

$$M^k v = \cdots = \sum_{i=1}^{4} \beta_i \lambda_i^k e_i$$

Dla dostatecznie wysokiej iteracji k, można zauważyć, że powyższy wzór będzie posiadać znacznie dominującą wartość $\lambda_i^k$, wobec której inne czynniki będą zaniedbywalnie małe. Oznacza to, że suma po prawej stronie tego wzoru będzie dążyć do wektora własnego dla największej wartości własnej $\lambda_1$.

Metoda ta posiada niestety wady, takie jak:

1. Dla macierzy które nie są dodatnio określone (czyli ich wartości własne nie są dodatnie), wartością dominującą zostanie wartość własna o największej wartości modułowej.
2. Ta metoda staje się bardziej kosztowna dla wyszukiwania innych wartości własnych niż największej, ponieważ należy przeprowadzić proces reortogonalizacji.
3. Osiągnięcie zbieżności metody potęgowej nastąpi po bardzo długim czasie dla wartości własnych zbliżonych modułem sobie (np.: 6.0 i 5.999999, -8.5 i 8.499999).
4. Metoda ta nie działa dla macierzy niesymetrycznych $(A \neq A^T)$.

# Algorytm QR

Algorytm QR korzysta z iteracyjnego rozkładu następujących macierzy $M_i$ na ich odpowiadające macierze $Q_iR_i$, wedle następujących wzorów:

$$M = M_1 = Q_1 R_1 \qquad M_2 = R_1 Q_1 = Q_1^T M_1 Q_1$$

$$\vdots$$

$$M_i = R_{i-1} Q_{i-1} = Q_{i-1}^T M_{i-1} Q_{i-1} = Q_i R_i$$

Wymnożenie czynników faktoryzacji QR w odwrotnej kolejności stanowi ortogonalną transformację macierzy podobnej do macierzy M. Po wystarczającej ilości iteracji (n), macierz $M_n$ jest ortogonalnie podobna do macierzy M. Jeżeli wartości własne należą do zbioru liczb rzeczywistych oraz są parami różne od siebie, n-ta iteracja będzie zbieżna do macierzy trójkątnej górnej, na której przekątnej będą kolejne wartości własne macierzy M.

Problemem algorytmu QR jest jego kosztowność: jeden rozkład QR macierzy posiada złożoność obliczeniową $O(n^3)$, co dla k iteracji, czyni złożoność całkowitego algorytmu $O(n^3 * k)$. Nie jest to problemem dla macierzy rzadkich, gdyż rozkład QR dla takowych jest znacznie prostszy.

# Wynik

## Wynik dla metody potęgowej

```
Zbieznosc algorytmu metody potegowej osiagnieta po 18 iteracjach.

Najwieksza wartosc wlasna (metoda potegowa): 9.71855

Odpowiadajacy jej wektor wlasny (metoda potegowa):
(0.939848, 0.337663, 0.0512467, 0.00663946)
```

## Wynik dla algorytmu QR

```
Zbieznosc algorytmu QR osiagnieta po 33 iteracjach.

Finalna macierz Ai:
(9.71855, 3.07894e-12, 3.35715e-16, 3.47079e-17
3.07949e-12, 4.3017, 5.8325e-07, 9.73802e-17
0, 5.8325e-07, 2.74019, -4.74033e-12
0, 0, -4.74071e-12, 1.23955)
```

Wartości własne macierzy M leżą na przekątnej macierzy końcowej:

$$\lambda_1 = 9.71855 \qquad \lambda_2 = 4.3017 \qquad \lambda_3 = 2.74019 \qquad \lambda_4 = 1.23955$$

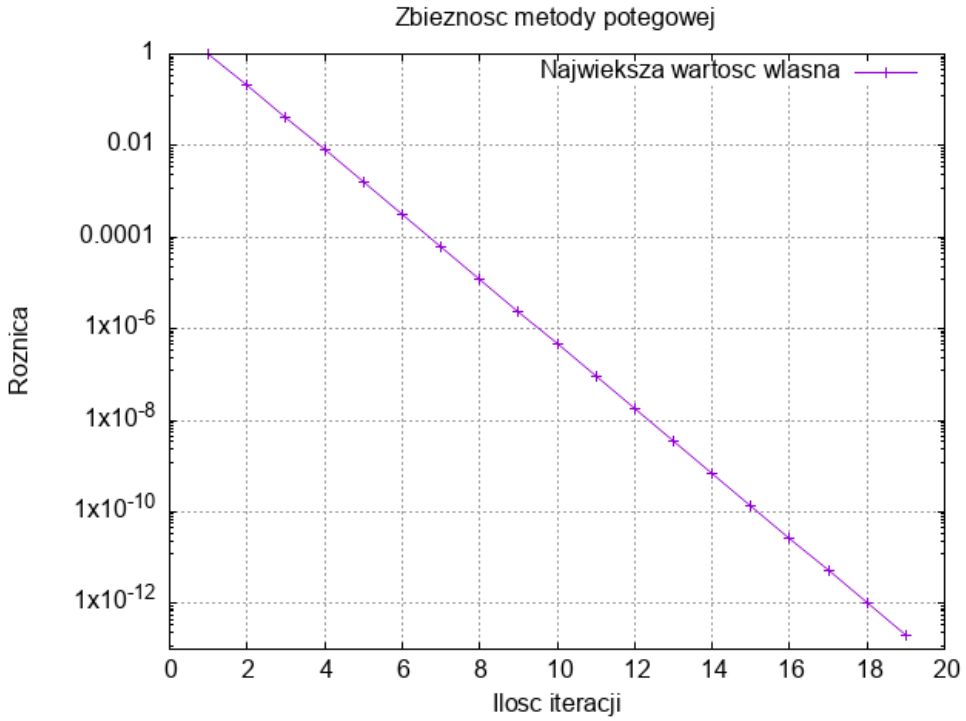
## Wynik dla biblioteki Eigen

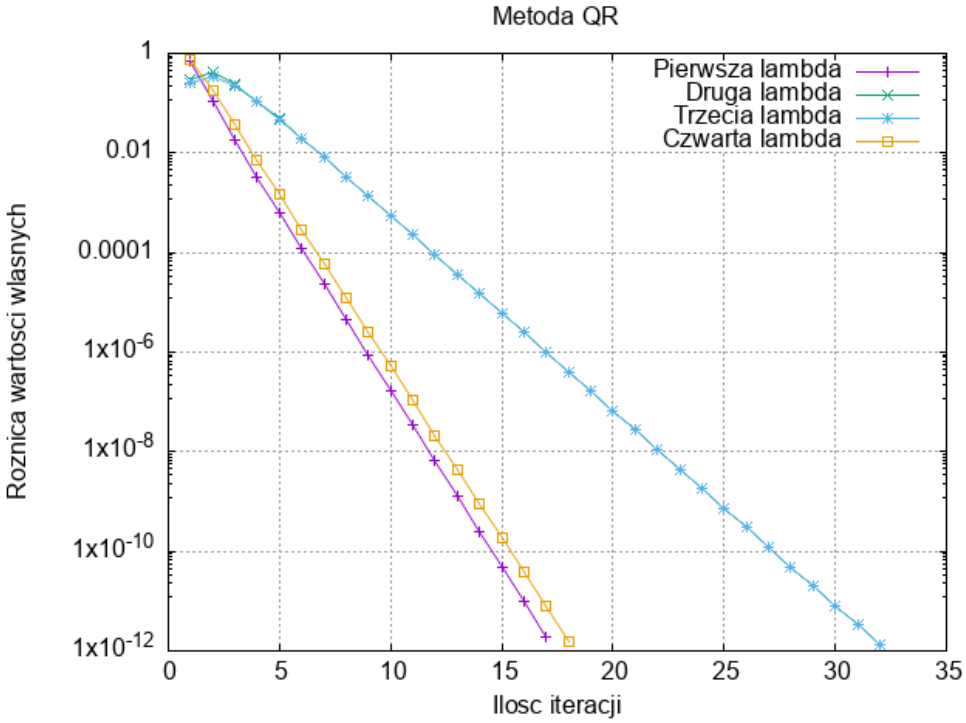
```
Wartosci wlasne obliczone za pomoca biblioteki Eigen:
(9.71855, 4.3017, 2.74019, 1.23955)

Znormalizowane wektory wlasne obliczone za pomoca biblioteki Eigen zlozone w macierz:
(-0.939848, 0.256151, 0.215852, -0.0669403
-0.337663, -0.601737, -0.675596, 0.259743
-0.0512465, -0.693849, 0.419416, -0.583127
-0.0066394, -0.30145, 0.566629, 0.766821)
```

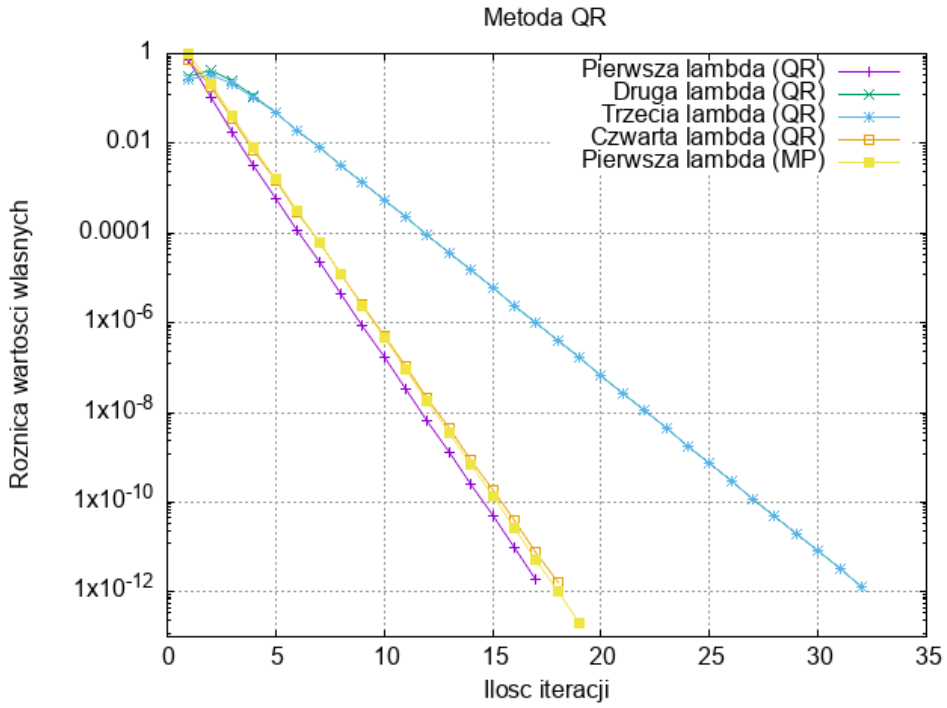
Można zauważyć, że największa wartość własna obliczona za pomocą metody potęgowej, oraz wartości własne wyliczone za pomocą algorytmu QR są takie same jak wynik otrzymany za pomocą biblioteki Eigen, co udowadnia ich skuteczność.

Metoda potęgowa osiągnęła zbieżność `tolerance = 1e-12` po 18 iteracjach, natomiast algorytm QR – po 33 iteracjach. Poniżej, pokazane są wykresy ich zbieżności:


Zbieznosc metody potegowej
Najwieksza wartosc wlasna
Roznica
1
0.01
0.0001
1x10-6
1x10-8
1x10-10
1x10-12
0
2
4
6
8
10
12
14
16
18
20
Ilosc iteracji


Metoda QR
Pierwsza lambda
Druga lambda
Trzecia lambda
Czwarta lambda
Roznica wartosci wlasnych
1
0.01
0.0001
1x10-6
1x10-8
1x10-10
1x10-12
0
5
10
15
20
25
30
35
Ilosc iteracji

Łącząc oba wykresy otrzymujemy:




Liczenie skrajnych wartości własnych macierzy M (największej oraz najmniejszej) zajęło algorytmowi QR około tyle samo iteracji co metodzie potęgowej, natomiast reszta wartości własnych osiągnęła zadowalającą zbieżność po 30+ iteracjach.

Aby usprawnić metodę algorytmu QR należy wziąć pod uwagę strukturę macierzy M – jest ona macierzą trójdiagonalną oraz symetryczną – w takim wypadku, można skorzystać z metod obrotów Givensa.

## Macierze $M_i$

Poniżej podany zostanie ciąg macierzy $M_i$, w celu zbadania jego własności.

$$M_1 = M = \begin{pmatrix} 9 & 2 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 9.61176 & 0.783568 & -5.55112 * 10^{-17} & -1.38778 * 10^{-17} \\ 0.783568 & 3.87787 & 0.77444 & -5.55112 * 10^{-17} \\ 0 & 0.77444 & 3.08677 & -0.544922 \\ 0 & 0 & -0.544922 & 1.4236 \end{pmatrix}$$

$$\vdots$$

$$M_{15} = \begin{pmatrix} 9.71855 & 1.63582 * 10^{-5} & 3.34632 * 10^{-16} & -3.47039 * 10^{-17} \\ 1.63582 * 10^{-5} & 4.3017 & 0.00307029 & -9.66456 * 10^{-17} \\ 0 & 0.00307029 & 2.7402 & 1.66592 * 10^{-5} \\ 0 & 0 & 1.66592 * 10^{-5} & 1.23955 \end{pmatrix}$$

$$\vdots$$

$$\vdots$$

$$M_{33} = M_{końcowa} = \begin{pmatrix} 9.71855 & 6.95674 * 10^{-12} & 3.35715 * 10^{-16} & -3.47079 * 10^{-17} \\ 6.95729 * 10^{-12} & 4.3017 & 9.15618 * 10^{-7} & -9.73801 * 10^{-17} \\ 0 & 9.15618 * 10^{-7} & 2.74019 & 1.04796 * 10^{-11} \\ 0 & 0 & 1.048 * 10^{-11} & 1.23955 \end{pmatrix}$$

Reszta iteracji jest zapisana w pliku „`Matrices Ai.txt`”.

Macierz $M_i$ dąży do macierzy diagonalnej ortogonalnie podobnej do macierzy M.