# Rozdział 10

# Zadanie własne

Numeryczne zadanie własne polega na znalezieniu niektórych lub wszystkich wartości oraz wektorów własnych macierzy $A$ wymiaru $N \times N$ rzeczywistej lub zespolonej. Na wykładzie z Matematyki Obliczeniowej omawia się kilka najprostszych metod dla symetrycznego zadania własnego, tzn. dla zadania własnego z macierzą rzeczywistą symetryczną.

Funkcją octave'a służącą do znajdowania wartości i wektorów własnych dowolnej macierzy jest **eig**().

Najprostsze jej wywołanie to:

```
w=eig(A)
```

po którym funkcja powinna zwrócić wszystkie wartości własne w wektorze $w$.

Jeśli potrzebujemy poznać wektory własne, to musimy wywołać tę funkcję z dwoma zwracanymi wartościami:

```
[V,D]=eig(A)
```

W tym przypadku $D$ będzie macierzą diagonalną, na której diagonali będą wartości własne macierzy $A$, natomiast $V$ będzie macierzą ortogonalną, w której kolumnach będą wektory własne dla odpowiednich wartości na diagonali $D$, czyli

$$AV = VD.$$

Sprawdźmy na losowej macierzy diagonalnej jak działa funkcja **eig**(), najpierw przy prostszym wywołaniu:

```
A=rand(5,5);
A=A+A'; # symetryzujemy macierz
w=eig(A)
det(A-w(1)*eye(size(A)))
```

Wyznacznik $A - w(1) * I$ jest na poziomie $10^{-14}$, czyli de facto jest równy zero.

Teraz sprawdźmy drugą formę wywołania funkcji **eig**():

```
[V,D]=eig(A)
norm(V*D-A*V,1)
```

Sprawdźmy, jak działa funkcja dla macierzy większego wymiaru:

```
N=10;
for k=1:3,
  A=rand(N,N);
  A=A+A'; # symetryzujemy macierz
  [V,D]=eig(A);
  norm(V*D-A*V,1)
  N*=10
endfor
```

Wniosek: funkcja działa dobrze nawet dla większych $N$.

Przetestujmy ją dla macierzy ogromnego wymiaru o dużej ilości zer, np. macierzy trójdiagonalnej. Wykorzystamy też funkcje **sparse**() służące utworzeniu macierzy w formacie rzadkim:

```
N=1000;
x=ones(N,1);
A=sparse(diag(x))-sparse(diag(x(1:N-1),1));
A=A+A';
tic;[V,D]=eig(A);toc
norm(V*D-A*V,1)
```

Sprawdźmy, jak działa funkcja dla macierzy niesymetrycznej:

```
A=[2,-1;1,2]
w-eig(A)
[V,D]=eig(A)
norm(V*D-A*V,1)
```

Istnieje też funkcja eigs(), która oblicza tylko kilka wartości własnych macierzy, np. największych co do modułu wartości.

Przetestujmy funkcję dla losowej macierzy wymiaru $10 \times 10$:

```
A=rand(10,10);
A=A+A'; # symetryzujemy macierz
wa=eig(A)
w=eigs(A)
```

Funkcja domyślnie znajduje sześć największych co do modułu wartości własnych, ale może znaleźć mniej lub więcej:

```
w3=eigs(A,3)
w9=eigs(A,8)
```

lub bazując na innym kryterium, np. najmniejsze wartości własne co do modułu:

```
wm=eigs(A,3,'sm')
wd=eigs(A,3)
```

Możliwe opcje to:

- 'lm' największe wartości co do modułu (default).
- 'sm' najmniejsze wartości co do modułu
- 'la' największe wartości (tylko o ile $A$ jest symetryczna)
- 'sa' najmniejsze wartości (tylko o ile $A$ jest symetryczna)
- 'be' wartości z końców widma macierzy, z jedną więcej z górnej części widma, o ile chcemy znaleźć nieparzystą liczbę wartości własnych (tylko o ile $A$ jest symetryczna)
- 'lr' największe części rzeczywistych wartości własnych (tylko o ile $A$ jest niesymetryczna lub zespolona).
- 'sr' najmniejsze części rzeczywistych wartości własnych (tylko o ile $A$ jest niesymetryczna lub zespolona).
- 'li' największe wartości części urojonych wartości własnych (tylko o ile $A$ jest niesymetryczna lub zespolona).
- 'si' najmniejsze wartości części urojonych wartości własnych (tylko o ile $A$ jest niesymetryczna lub zespolona).

Spróbujmy policzyć dwie największe i dwie najmniejsze co do wartości własne macierzy trójdiagonalnej przy pomocy tej funkcji:

```
N=1000;
x=ones(N,1);
A=sparse(diag(x))-sparse(diag(x(1:N-1),1));
A=A+A';
tic;w=eigs(A,4,'be');toc
```

Niestety metoda zawiodła, a funkcja **eig**() - choć wolna, zwróciła wszystkie wartości i wektory tej macierzy.

Zaimplementujmy najprostszą wersję metody potęgowej:

Niech $\hat{x}_0 = x_0$ będzie wektorem o normie drugiej równej jeden. Wtedy ciąg iteracji metody potęgowej jest zdefiniowany następująco:

$$\begin{array}{rcll} \hat{x}_k & = & Ax_{k-1} & \text{iteracja} \\ x_k & = & \frac{\hat{x}_k}{\|\hat{x}_k\|_2} & \text{normowanie} \qquad k > 0 \\ r_k & = & x_k^T A x_k & \text{iloraz Rayleigha} \end{array} \tag{10.1}$$

W octavie kod może wyglądać następująco:

```
x=rand(N,1); #losujemy wektor
x=x/norm(x,1); #normowanie x0
y=A*x;
r=x'*y; rp=r+2*TOL;
it=0;
while(abs(r-rp)>TOL)
 x=y/norm(y,2);
 y=A*x;
 rp=r;
 r=x'*y;
 it++;
endwhile
x=y/norm(y,2)
r
```

Za warunek stopu przyjęliśmy $|r_k - r_{k-1}| < TOL$ zadanej tolerancji.

Przetestujmy ten algorytm dla macierzy symetrycznej A=[4,1;1,4], która ma wartości własne $5, 3$.

Dla TOL=1e−12 otrzymaliśmy, że po 24 iteracjach iloraz Rayleigha wynosi $r = 4.99999999999963$, czyli błąd $3.71e{-}13$. Z kolei przybliżenie wektora własnego to

```
x =

   0.707106628756140
   0.707106933616922
```

Możemy policzyć $\|Ax - 5 * x\|_2$ w octave'ie, czyli **norm**(A∗x−5∗x,2), która wynosi $4.31e-07$.

Po prostych rachunkach otrzymujemy, że wektorem własnym dla wartości własnej 5 jest wektor postaci $x = (x_1, x_1)^T \neq 0$.

Sprawdźmy, dla naszego wektora:

```
octave:111> abs(x(1)-x(2))
ans =  3.048607817301 98e-07
octave:112>
```

Oczywiście ilość iteracji i błędy zależą też od wektora $x_0$.

Powtórzmy obliczenia drukując na ekranie błąd $|5 - r_k|$ i $|(1, -1)^T x_k|$ dla $x_k$ iteracji metody potęgowej i $r_k$ ilorazu Rayleigha:

```
A =

   4   1
   1   4

| it ||   Rayleigh   |    |x-y|    |
-------------------------------------
|  1] |  2.5008722  |  0.0241133  |
|  2] |  4.9995814  |  0.0204587  |
|  3] |  4.9998493  |  0.0122760  |
|  4] |  4.9999457  |  0.0073658  |
|  5] |  4.9999805  |  0.0044195  |
|  6] |  4.9999930  |  0.0026517  |
|  7] |  4.9999975  |  0.0015910  |
|  8] |  4.9999991  |  0.0009546  |
|  9] |  4.9999997  |  0.0005728  |
| 10] |  4.9999999  |  0.0003437  |
| 11] |  5.0000000  |  0.0002062  |
| 12] |  5.0000000  |  0.0001237  |
| 13] |  5.0000000  |  0.0000742  |
| 14] |  5.0000000  |  0.0000445  |
| 15] |  5.0000000  |  0.0000267  |
| 16] |  5.0000000  |  0.0000160  |
| 17] |  5.0000000  |  0.0000096  |
| 18] |  5.0000000  |  0.0000058  |
| 19] |  5.0000000  |  0.0000035  |
| 20] |  5.0000000  |  0.0000021  |
| 21] |  5.0000000  |  0.0000012  |
-------------------------------------

x =

   7.071065567627 92e-01
```

```
    7.071070056102 32e-01

Wartosc wlasna:     4.999999999999440
```

Powtarzając eksperyment dla macierzy $A - 2 * I$, czyli macierzy o wartościach własnych 3 i 1, otrzymujemy po 14 iteracjach iloraz Rayleigha taki, że błąd $|3 - r|$ wynosi $3.99e - 14$ przy tym samym warunku stopu:

```
p =   2
A =

    2    1
    1    2

[it]|  Rayleigh+p  |    |x-y|    |
-------------------------------------
[ 1] |  3.5005623  |  0.0335338  |
[ 2] |  4.9997501  |  0.0158070  |
[ 3] |  4.9999722  |  0.0052693  |
[ 4] |  4.9999969  |  0.0017564  |
[ 5] |  4.9999997  |  0.0005855  |
[ 6] |  5.0000000  |  0.0001952  |
[ 7] |  5.0000000  |  0.0000651  |
[ 8] |  5.0000000  |  0.0000217  |
[ 9] |  5.0000000  |  0.0000072  |
[10] |  5.0000000  |  0.0000024  |
[11] |  5.0000000  |  0.0000008  |
-------------------------------------
x =

    7.071067365683 27e-01
    7.071068258047 65e-01

Wartosc wlasna:     4.999999999999928
```

Widać już z tych dwóch eksperymentów, że ilość iteracji zależy od stosunku $\lambda_2/\lambda_1$ ( gdzie $\lambda_1$ to wartość własna o maksymalnym module, $\lambda_2$ to druga z kolei wartość własna ze względu na moduł) i że szybkość zbieżności ilorazu Rayleigha jest dużo większa niż wektorów iteracji do odpowiedniego wektora własnego. Potwierdza to oszacowania teoretyczne, w których można

pokazać, że

$$v_1 = x_k + O((\lambda_2/\lambda_1)^k), \qquad \lambda_1 = r_k + O((\lambda_2/\lambda_1)^{2k}).$$

Widać, że iloraz Rayleigha zbiega do wartości własnej dużo szybciej w tym przypadku.

Jeśli $(\lambda_k, v_k)$ to para własna dla macierzy $A$, to $((\lambda_k - p)^{-1}, v_k)$ jest parą własną dla macierzy $(A - pI)^{-1}$. Ta obserwacja pozwala nam skonstruować tzw. odwrotną metodę potęgową, która jest zdefiniowana jako metoda potęgowa zastosowana do macierzy $(A-pI)^{-1}$ dla $p$ parametru będącego *dobrym* przybliżeniem $\lambda_k$ dowolnej wartości własnej $A$. Warunkiem zbieżności iteracji odwrotnej metody potęgowej do $v_k$ wektora własnego dla $\lambda_k$, przy założeniu, że wektor startowy nie jest ortogonalny do $v_k$ jest to, aby

$$|\lambda_k - p| > |\lambda_j - p| \qquad \forall j \neq k.$$

Szybkość zbieżności wektorów iteracji dla odwrotnej metody potęgowej zależy wtedy od $\max_{j \neq k} \frac{|\lambda_k - p|}{|\lambda_j - p|}$.

W octave'ie łatwo zaimplementować odwrotną metodę potęgową: wystarczy mnożenie przez macierz $A$ w metodzie potęgowej zastąpić przez mnożenie wektora przez macierz odwrotną, które jest równoważne rozwiązywaniu układu równań. W octave kod odwrotnej metody potęgowej może wyglądać następująco:

```
A=A-p*eye(size(A)); # tworzymy macierz A-p*I

x=rand(N,1); #losujemy wektor
x=x/norm(x,1); #normowanie x0
y=A\x;
r=x'*y;
rp=r+2*TOL;
it=1;

while(abs(r-rp)>TOL)
  x=y/norm(y,2);
  y=A\x;
  rp=r;
  r=x'*y;
  it++;
endwhile
```

```
x=y/norm(y,2) #przyblizenie wektora wlasnego A
1/r+p #przyblizenie wartosci wlasnej A
```

Zobaczmy jak szybko działa metoda dla naszej testowej macierzy $A$ i $p = 5.01$:

```
p =   5.0100
A =

   -1.0100    1.0000
    1.0000   -1.0100

[it]||     1/r+p       |     |x-y|     |
-------------------------------------------
[  1]  |  4.9900001  |  0.0273613  |
[  2]  |  5.0000000  |  0.0001925  |
[  3]  |  5.0000000  |  0.0000010  |
[  4]  |  5.0000000  |  0.0000000  |
-------------------------------------------

x =

   -7.07106781186489e-01
   -7.07106781186606e-01

Wartosc wlasna:    5.000000000000000
```

A teraz dla $p = 2.8$ - tu wektor własny ma postać $a * (1, -1)^T$:

```
p =   2.8000
A =

    1.2000    1.0000
    1.0000    1.2000

[it]||     1/r+p       |     |x+y|     |
-------------------------------------------
[  1]  |  7.1576796  |  1.0000000  |
[  2]  |  3.9194920  |  1.3442338  |
[  3]  |  3.0139676  |  0.3789639  |
[  4]  |  3.0001162  |  0.0357476  |
[  5]  |  3.0000010  |  0.0032508  |
[  6]  |  3.0000000  |  0.0002955  |
```

```
[ 7] | 3.0000000 | 0.0000269 |
[ 8] | 3.0000000 | 0.0000024 |
[ 9] | 3.0000000 | 0.0000002 |
```

```
x =

   7.07106782104050e-01
  -7.07106780269045e-01

Wartosc wlasna:     3.000000000000000
```

Przesuwanie widma macierzy poprzez dodanie do niej $p * I$ jest bardzo ważne w różnych uogólnieniach metody potęgowej takich jak np. metoda $QR$ z przesunięciami, por. np. [11], [18], [3].

# Bibliografia


[1] Åke Björck i Germund Dahlquist. *Metody numeryczne*. Państwowe Wydawnictwo Naukowe (PWN), Warszawa, 1983. Przetłumaczone ze szwedzkiego przez Stefana Paszkowskiego.

[2] James W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[3] Maksymilian Dryja, Janina Jankowska, i Michał Jankowski. *Metody numeryczne*, tom 2. Wydawnictwo Naukowo Techniczne (WNT), Warszawa, 1982.

[4] John W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002.

[5] Gene Golub i James M. Ortega. *Scientific computing*. Academic Press Inc., Boston, MA, 1993. An introduction with parallel computing.

[6] Gene H. Golub i James M. Ortega. *Scientific computing and differential equations*. Academic Press Inc., Boston, MA, 1992. An introduction to numerical methods.

[7] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*, tom 95, *Applied Mathematical Sciences*. Springer-Verlag, New York, 1994. Translated and revised from the 1991 German original.

[8] Janina Jankowska i Michał Jankowski. *Metody numeryczne*, tom 1. Wydawnictwo Naukowo Techniczne (WNT), Warszawa, 1981.

[9] C. T. Kelley. *Solving nonlinear equations with Newton's method*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003.

[10] A. Kiełbasiński i H. Schwetlick. *Numeryczna algebra liniowa: wprowadzenie do obliczeń zautomatyzowanych*. Wydawnictwo Naukowo Techniczne (WNT), Warszawa, 1992.

[11] David Kincaid i Ward Cheney. *Analiza numeryczna.* Wydawnictwo Naukowo-Techniczne (WNT), 2006.

[12] J. M. Ortega i W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*, tom 30, *Classics in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Reprint of the 1970 original.

[13] James M. Ortega. *Numerical analysis*, tom 3, *Classics in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 1990. A second course.

[14] Alfio Quarteroni, Riccardo Sacco, i Fausto Saleri. *Numerical mathematics*, tom 37, *Texts in Applied Mathematics.* Springer-Verlag, New York, 2000.

[15] Alfio Quarteroni i Fausto Saleri. *Scientific computing with MATLAB*, tom 2, *Texts in Computational Science and Engineering.* Springer-Verlag, Berlin, 2003.

[16] Anthony Ralston. *Wstęp do analizy numerycznej.* Państwowe Wydawnictwo Naukowe (PWN), Warszawa, 1983.

[17] J. Stoer i R. Bulirsch. *Wstęp do metod numerycznych. Tom drugi.* Państwowe Wydawnictwo Naukowe (PWN), Warszawa, 1980. Przetłumaczone z niemieckiego przez Małgorzatę Mikulską.

[18] Lloyd N. Trefethen i David Bau, III. *Numerical linear algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.