Metoda potęgowa oraz rozkład SVD macierzy 3x3

Jakub Płowiec, Filip Dziurdzia

Zadanie

W wybranym języku programowania (**Python**) napisać program, który:

1. Implementuje metodę potęgową dla macierzy 3x3 z warunkiem początkowym:
   - losujemy wektor $z_0 \in (0,1)^3$,
   - obliczamy $w_0 = A \cdot z_0$,
   - liczymy błąd: $error = ||Az_0 - max(w_{0i}) \cdot z_0||_p$,
   - jeśli $error < 1e-8$, losujemy nowy z_0,
   - iterujemy aż $||Az - \lambda z||_p < epsilon = 0.0001$ dla różnych $p = 1, 2, 3, 4, \infty$.
2. Oblicza SVD macierzy $A = UDV^T$:
   - najpierw poprzez własne wartości i wektory $A \cdot A^T \to U, D$,
   - następnie $V = A^T \cdot U \cdot inv(D)$, gdzie $inv(D)$ to macierz diagonalna z odwrotnościami wartości własnych.
3. Porównuje wykresy zbieżności metody potęgowej dla różnych $p \in 1, 2, 3, 4, \infty$ i 3 losowych wektorów startowych — w sumie 15 wykresów (iteracja vs error).
4. Sprawdza dokładność rekonstrukcji: $||UDV^T - A||_p$ dla $p \in 1, 2, 3, 4, \infty$ oraz porównuje z wynikami bibliotecznego SVD.

Pseudokod algorytmu metody potęgowej

1. Wylosuj $z_0 \in (0,1)^3$
2. Oblicz $w_0 = A \cdot z_0$
3. Oblicz $error = ||Az_0 - max(w_{0i}) \cdot z_0||_p$
4. Jeśli $error < 1e-8$, wróć do 1
5. Inaczej:
   - iteruj aż $error < epsilon$:
     - $w = A \cdot z$
     - $\lambda = max(w)$
     - $z = \frac{w}{||w||_p}$
     - $error = ||Az - \lambda z||_p$

Implementacja

Implementacja została wykonana w języku `Python` z wykorzystaniem biblioteki `numpy`. Do wygenerowania wykresów wykorzystaliśmy bibliotekę `matplotlib`.

Fragmenty kodu:

Metoda potęgowa:

```
def power_method(A, p=2, epsilon=1e-4, max_iter=15, z=None):
    if z is None:
        z = np.random.rand(3)
    z = z / np.linalg.norm(z, ord=p)

    errors = []
    for _ in range(max_iter):
        w = A @ z
        lam = np.max(w)
        z = w / np.linalg.norm(w, ord=p)
        error = np.linalg.norm(A @ z - lam * z, ord=p)
        errors.append(error)
        if error < epsilon:
            break
    return errors
```

Obliczanie SVD ręcznie:

```
AAT = A @ A.T
eigvals, U = np.linalg.eigh(AAT)
D = np.diag(np.sqrt(eigvals[::-1]))
U = U[:, ::-1]
V = A.T @ U @ np.linalg.inv(D)
S = U @ D @ V.T
```

Wylosowana macierz A

```
[[0.09182508 0.0098887  0.90247511]
 [0.74022275 0.95098443 0.04265521]
 [0.61992262 0.16091681 0.3074236 ]]
```

Wyniki obliczeń

Macierz U

```
[[-0.25247388  0.90462637 -0.34337746]
 [-0.84895771 -0.37736453 -0.36995514]
 [-0.46424965  0.19810893  0.86326422]]
```

Macierz D (wartości osobliwe)

```
[[1.35291223 0.         0.        ]
 [0.         0.92099344 0.        ]
 [0.         0.         0.32127661]]
```

Macierz V

```
[[-0.69435406 -0.07975542  0.71520033]
 [-0.65381017 -0.3453262  -0.67326226]
 [-0.30067372  0.93508764 -0.18763375]]
```

Błędy rekonstrukcji $ ||UDV - A||__p $

Porównano rekonstrukcję macierzy z oryginalną A dla różnych norm:

- $p = 1$: $1.176 \cdot 10^{-15}$
- $p = 2$: $4.892 \cdot 10^{-16}$
- $p = 3$: $3.887 \cdot 10^{-16}$
- $p = 4$: $3.572 \cdot 10^{-16}$
- $p = \infty$: $3.330 \cdot 10^{-16}$

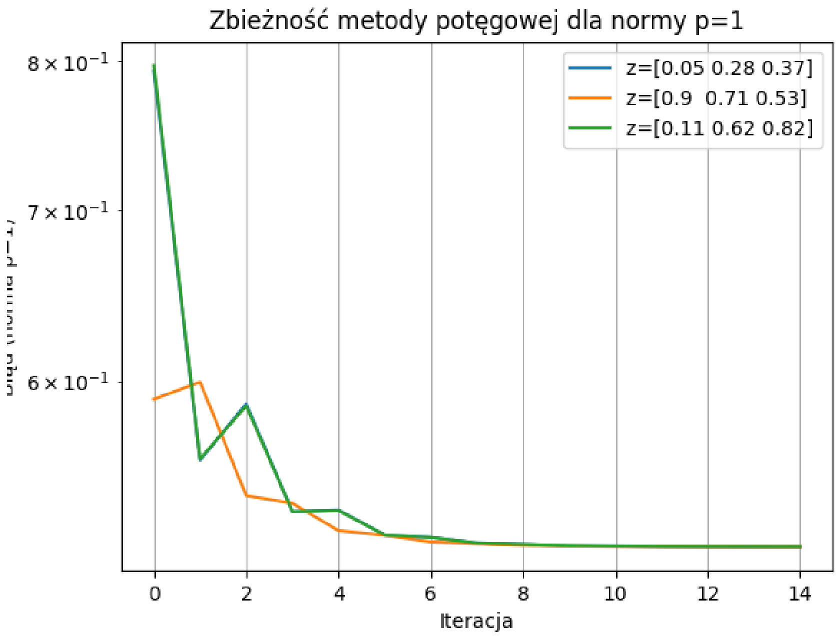
W każdym przypadku błędy są bliskie zeru, co świadczy o poprawności obliczeń.

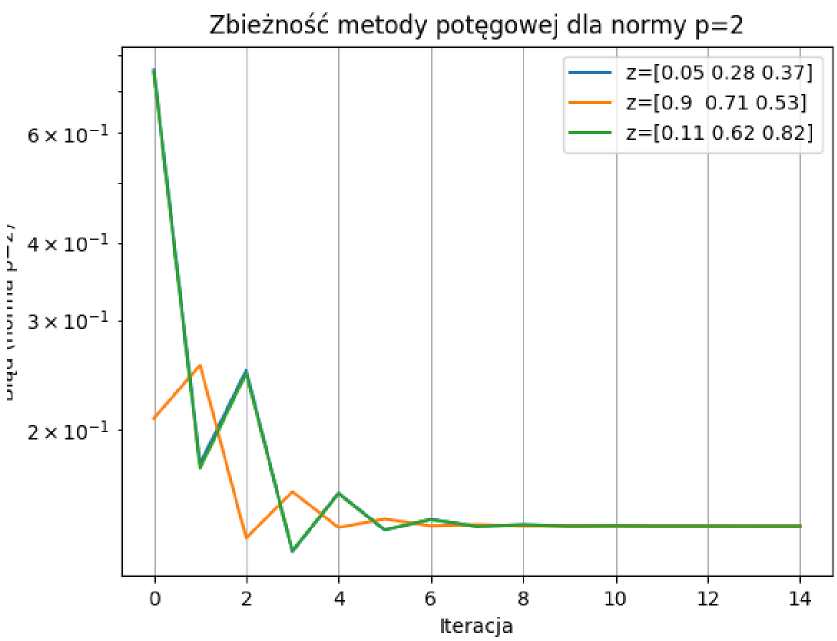
Wykresy zbieżności metody potęgowej

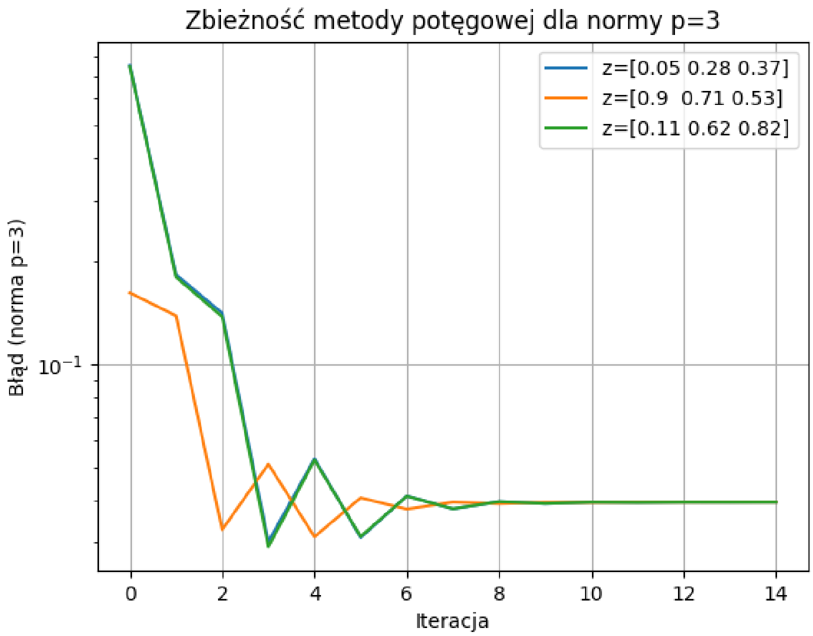
Dla każdej normy $p \in 1, 2, 3, 4, \infty$ wygenerowano 3 wykresy (dla 3 losowych z_0), przedstawiające:

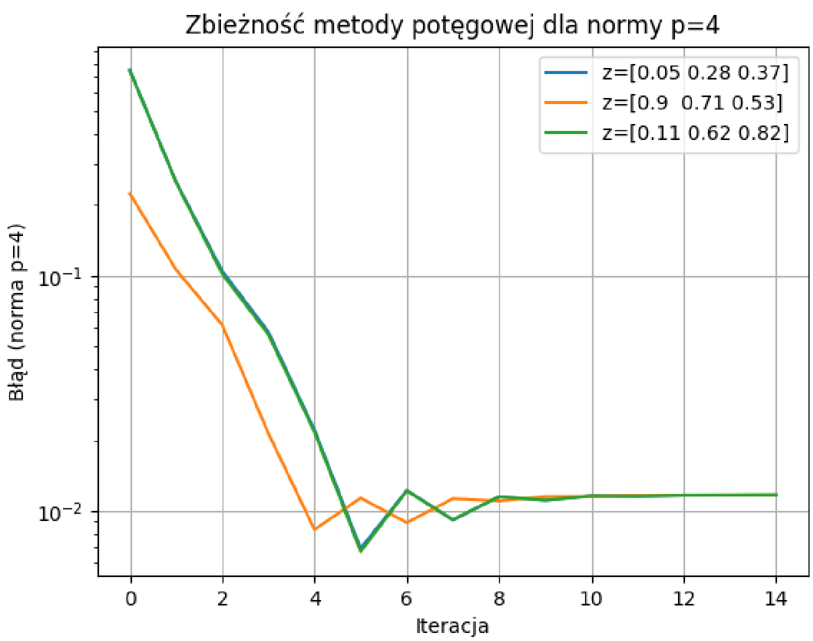
- Oś X: numer iteracji
- Oś Y: błąd $||Az - \lambda z||_p$

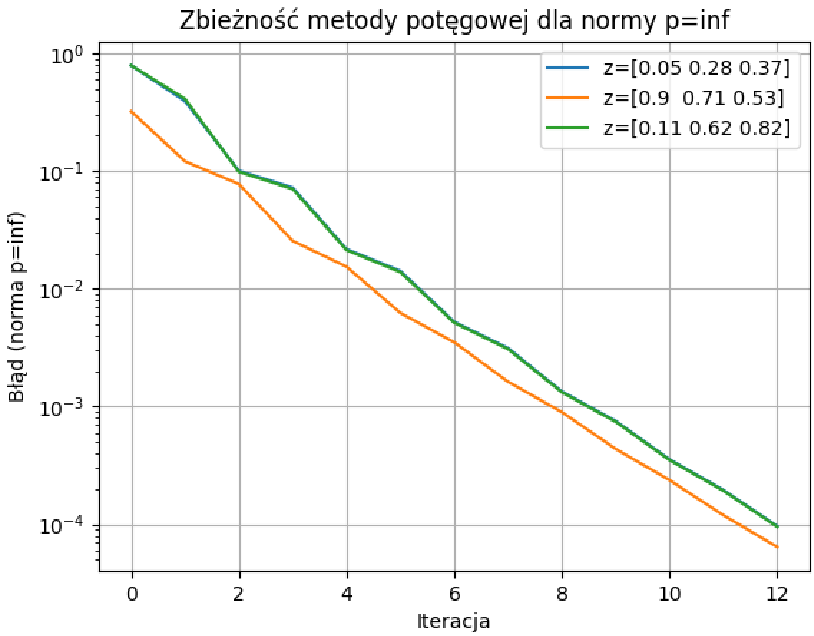
Łącznie 15 wykresów ilustrujących szybkość zbieżności.


Zbieżność metody potęgowej dla normy p=1
z=[0.05 0.28 0.37]
z=[0.9 0.71 0.53]
z=[0.11 0.62 0.82]
Iteracja


Zbieżność metody potęgowej dla normy p=2
z=[0.05 0.28 0.37]
z=[0.9 0.71 0.53]
z=[0.11 0.62 0.82]
Iteracja


Zbieżność metody potęgowej dla normy p=3
z=[0.05 0.28 0.37]
z=[0.9 0.71 0.53]
z=[0.11 0.62 0.82]
Błąd (norma p=3)
10^{-1}
0
2
4
6
8
10
12
14
Iteracja


Zbieżność metody potęgowej dla normy p=4
z=[0.05 0.28 0.37]
z=[0.9 0.71 0.53]
z=[0.11 0.62 0.82]
Błąd (norma p=4)
10^{-1}
10^{-2}
0
2
4
6
8
10
12
14
Iteracja



Wnioski

- Metoda potęgowa skutecznie aproksymuje dominującą wartość i wektor własny macierzy A.
- Obliczenia SVD z $A \cdot A^T$ są zgodne z wynikami bibliotecznymi `numpy.linalg.svd`.
- Błędy rekonstrukcji UDV - A są bardzo małe — potwierdza to poprawność własnej implementacji.
- Metoda potęgowa może być przydatna przy przybliżonym znajdowaniu największej wartości własnej.