Metody Numeryczne - Zadanie VI

Mateusz Kamiński

November 2025

1 Wprowadzenie

Celem ćwiczenia było zaimplementowanie i przetestowanie metody iteracji odwrotnej do wyznaczenia wektora własnego dla zadanej przybliżonej wartości własnej

2 Opis funkcji

2.1 Zamiana macierzy

W zadaniu rozważano macierz M postaci:

$$M = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} + uv^T,$$

przy czym wektory u i v są następujące:

$$u = [1, 0, 0, 0, 1]^T,$$
$$v = [1, 0, 0, 0, 1]^T.$$

Dzięki zapisowi $M = T + uv^T$ możemy efektywnie korzystać z faktoryzacji macierzy trójdiagonalnej T i zastosować formułę Shermana-Morrisona.

2.2 Faktoryzacja Thomasa

Wykorzystano funkcję `thomas_factor`, która dla zadanych wektorów macierzy trójdiagonalnej wykonuje faktoryzację LU. Algorytm pracuje w czasie liniowym $O(n)$ i przygotowuje dane potrzebne do późniejszego efektywnego rozwiązywania układów.

2.3 Rozwiązywanie układu równania trójprzekątnego

Funkcja `thomas_solve` realizuje etap podstawiania w przód i wstecz, także w czasie liniowym. Na jej podstawie można szybko obliczać $T^{-1}x$ dla różnych wektorów x.

2.4 Formuła Shermana–Morrisona

Zaimplementowano zależność

$$(T + uv^T)^{-1}z = T^{-1}z - \frac{v^T T^{-1} z}{1 + v^T T^{-1} u} T^{-1}u, \tag{1}$$

co pozwala uniknąć kosztownej faktoryzacji zmodyfikowanej macierzy.

2.5 Iteracja odwrotna z poprawką Shermana–Morrisona

Funkcja `inverse_iteration_with_sherman` wykonuje kolejne kroki iteracji odwrotnej:

1. Normalizacja wektora startowego.
2. Obliczenie raz $T^{-1}u$
3. Obliczenie $T^{-1}y$.
4. Zastosowanie formuły Shermana-Morrisona.
5. Renormalizacja wektora.

3 Implementacja

```
import numpy as np
from numpy.typing import NDArray

vector = NDArray[np.float64]
matrix = NDArray[np.float64]

def thomas_factor(subdiagonal: vector, diagonal: vector) -> tuple[vector,
    vector]:
    n = len(diagonal)
    L = np.zeros(n - 1)

    for i in range(1, n):
        L[i - 1] = subdiagonal[i - 1] / diagonal[i - 1]
        diagonal[i] -= L[i - 1] * subdiagonal[i - 1]

    return L, diagonal
```

```
def thomas_solve(L: vector, U: vector, b: vector, superdiagonal: vector) ->
    vector:
    n = len(U)

    # forward
    for i in range(1, n):
        b[i] -= L[i - 1] * b[i - 1]

    # back
    x = np.zeros(n)
    x[-1] = b[-1] / U[-1]
    for i in range(n - 2, -1, -1):
        x[i] = (b[i] - superdiagonal[i] * x[i + 1]) / U[i]

    return x


def sherman_morrison_formula(z: vector, q: vector, v: vector) -> vector:
    """
    wzór Shermana Morrisona:
        w = z - (v^T z)/(1 + v^T q) * q
    """

    vTz = np.dot(v, z)
    vTq = np.dot(v, q)
    alpha = float(vTz / (1.0 + vTq))

    return z - alpha * q

def inverse_iteration_with_sherman(subdiagonal: vector, diagonal: vector, u
    : vector, iterations: int = 1000) -> vector:
    n = len(diagonal)
    L, U = thomas_factor(subdiagonal, diagonal)
    superdiagonal = subdiagonal

    y = np.random.rand(n)
    y /= np.linalg.norm(y)

    T_inverse_u = thomas_solve(L, U, u, superdiagonal)
    for k in range(iterations):
        T_inverse_y = thomas_solve(L, U, y, superdiagonal)

        z = sherman_morrison_formula(T_inverse_y, T_inverse_u, u)

        y = z / np.linalg.norm(z)

    return y

def main():
    # u = v
```

```
    u: vector = np.array([1, 0, 0, 0, 1], dtype=np.float64)

    # M = T + u*vT
    diagonal = np.array([1, 2, 1, 2, 1], dtype=np.float64)
    subdiagonal = np.array([-1, 1, 1, -1], dtype=np.float64)

    tau = 0.38197
    diagonal_T = diagonal - tau

    np.set_printoptions(linewidth=np.inf)
    eigenvector = inverse_iteration_with_sherman(subdiagonal, diagonal_T, u
        )
    print("Przybliżony wektor własny dla ż      0.38197:")
    print(eigenvector)

if __name__ == '__main__':
    main()
```

4 Wyniki

```
Przybliżony wektor własny dla λ ≈ 0.38197:
[ 6.01492908e-01  3.71753008e-01 -1.60944503e-05 -3.71743061e-01 -6.01509002e-01]
```

Przybliżony wektor własny uzyskany w wyniku iteracji odwrotnej z poprawką Shermana–Morrisona:

$$x \approx \begin{bmatrix} 0.60149 \\ 0.37175 \\ 0.00000 \\ -0.3717 \\ -0.6015 \end{bmatrix}$$