lab2_paweł_skierkowski

May 27, 2025

1 Sprawozdanie z lab 2

1.1 Paweł Skierkowski

1.2 METODA JACOBIEGO

Metoda Jacobiego służy do rozwiązywania układów równań liniowych w postaci:

$$A\mathbf{x} = \mathbf{b}$$

1.3 Kroki:

1. Inicjalizacja przybliżenia początkowego x0
2. Obliczanie nowej wartości przy użyciu wzoru iteracyjnego
3. Sprawdzenie warunku zbieżności

Zalety: - Łatwa do zaimplementowania - Równoległa

Wady: - Może nie zbiegać dla dowolnej macierzy

Aby algorytm Jacobiego był zbieżny, macierz A musi mieć silnie dominującą diagonalę ### KOD:

[53]:
```
import numpy as np
import matplotlib.pyplot as plt
import time

def jacobi(A, b, x0, tol=1e-10, max_i=1000):
    n = len(A)
    x = x0.copy()
    x_new = x0.copy()
    iteracje = 0
    for k in range(max_i):
        for i in range(n):
            s = sum(A[i][j]*x[j] for j in range(n) if j != i)
            x_new[i] = (b[i] - s) / A[i][i]
        if np.linalg.norm(x_new - x, np.inf) < tol:
            break
        x = x_new.copy()
        iteracje += 1
    return x, iteracje
```

1.4 METODA GAUSSA-SEIDLA

Metoda Gaussa-Seidla różni się od metody Jacobiego tym, że podczas każdej iteracji wykorzystuje natychmiast najnowsze dostępne wartości zmiennych.

Podobnie do metody Jacobiego warunkiem zbieżności metody Gaussa-Seidela jest macierz diagonalnie dominująca lub symetryczna dodatnio określona, jednak może działać w szerszym zakresie i jest szybsza wykorzystując nadpisywanie wektora i korzystania z nowszych wyników.

1.5 Kroki:

1. zaczynamy od przybliżenia x0
2. obliczanie nowej wartości wykorzystując dane z bierzącej i poprzedniej iteracji
3. sprawdzenie warunku zbieżności

Zalety: - Szybsza zbieżność niż Jacobiego w wielu przypadkach - Mniejsza potrzeba pamięci (nie trzeba przechowywać osobnego x_new) - Prostsza logika dla macierzy, które są dobrze uwarunkowane

Wady: - Trudna do zrównoleglenia (ponieważ kolejność ma znaczenie) - Może nie zbiegać się dla niektórych macierzy

1.5.1 KOD:

[56]:
```
def gauss_seidel(A, b, x0, tol=1e-10, max_iter=1000):
    n = len(A)
    x = x0.copy()
    iteracje = 0
    for k in range(max_iter):
        x_old = x.copy()
        for i in range(n):
            s1 = sum(A[i][j]*x[j] for j in range(i))
            s2 = sum(A[i][j]*x_old[j] for j in range(i+1, n))
            x[i] = (b[i] - s1 - s2) / A[i][i]
        if np.linalg.norm(x - x_old, np.inf) < tol:
            break
        iteracje += 1
    return x, iteracje
```

2 Wyniki działania

[75]:
```
A = np.array([[4.0, -1.0, 0.0, 0.0],
              [-1.0, 4.0, -1.0, 0.0],
              [0.0, -1.0, 4.0, -1.0],
              [0.0, 0.0, -1.0, 3.0]])
b = np.array([15, 10, 10, 10])
x0 = np.zeros_like(b)

start = time.time()
```

```
x_jacobi, iter_jacobi = jacobi(A, b, x0)
time_jacobi = time.time() - start

start = time.time()
x_gs, iter_gs = gauss_seidel(A, b, x0)
time_gs = time.time() - start

print("Jacobi:", iter_jacobi, "iteracji, czas:", time_jacobi)
print("Gauss-Seidel:", iter_gs, "iteracji, czas:", time_gs)
```

```
Jacobi: 3 iteracji, czas: 0.001016378402709961
Gauss-Seidel: 2 iteracji, czas: 0.0
```

[71]:
```
#macierz dobrze uwarunkowana
A2 = np.array([[10, -1, 2],
               [-1, 11, -1],
               [2, -1, 10]], dtype=float)

b2 = np.array([6, 25, -11], dtype=float)
x02 = np.zeros_like(b2)
#macierz słabo uwarunkowana
A = np.array([[1.0, 1/2, 1/3],
              [1/2, 1/3, 1/4],
              [1/3, 1/4, 1/5]])

b3 = np.array([1, 1, 1], dtype=float)
x03 = np.zeros_like(b3)

start = time.time()
x_jacobi2, iter_jacobi2 = jacobi(A2, b2, x02)
time_jacobi2 = time.time() - start

start = time.time()
x_gs2, iter_gs2 = gauss_seidel(A2, b2, x02)
time_gs2 = time.time() - start

start = time.time()
x_jacobi3, iter_jacobi3 = jacobi(A3, b3, x03)
time_jacobi3 = time.time() - start

start = time.time()
x_gs3, iter_gs3 = gauss_seidel(A3, b3, x03)
time_gs3 = time.time() - start

print("macierz dobrze uwarunkowana")
print("Jacobi:", iter_jacobi2, "iteracji, czas:", time_jacobi2)
print("Gauss-Seidel:", iter_gs2, "iteracji, czas:", time_gs2)
```

```
print("macierz słabo uwarunkowana")
print("Jacobi:", iter_jacobi3, "iteracji, czas:", time_jacobi3)
print("Gauss-Seidel:", iter_gs3, "iteracji, czas:", time_gs3)
```

```
macierz dobrze uwarunkowana
Jacobi: 18 iteracji, czas: 0.0
Gauss-Seidel: 9 iteracji, czas: 0.0
macierz słabo uwarunkowana
Jacobi: 1000 iteracji, czas: 0.017139434814453125
Gauss-Seidel: 1000 iteracji, czas: 0.021492481231689453
```

2.1 Podsumowanie

Jak widać metoda Gaussa-Seidla działa szybciej, lub w podobnym czasie wykonując przy tym mniej iteracji. Równoległość w metodzie Jacobiego pozwala na jednoczesne obliczanie wszystkich nowych przybliżeń niewiadomych. Metoda Gaussa-Seidela wykorzystuje najnowsze dostępne wartości w trakcie jednej iteracji, co zwykle przyspiesza zbieżność.