

Rozwiązanie układu równań liniowych iteracyjną metodą Richardsona Sprawozdanie, Etap I

Kacper Górka, Krzysztof Rudnicki, Aleksandra Sobala

9 grudnia 2024

1 Zadanie

Metoda Richardsona Metoda Richardsona służy do iteracyjnego rozwiązywania systemów równań liniowych postaci $Ax = b$.
Pojedyncza iteracja wygląda następująco:

$$x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)})$$

Gdzie ω to skalar wybrany tak by $x^{(k)}$ zbiegało

Wymagania Mieliśmy za zadanie stworzyć program rozwiązujący układ równań dla wygenerowanych macierzy gęstych oraz dla macierzy rzadkich: [nemeth12](#)
i [poli3](#)

2 Baza

Generowanie i zapisywanie macierzy Macierze gęste są przez nas generowane przy użyciu biblioteki **numpy**, aby przyspieszyć obliczenia zapewniamy **bewzględna dominację wierszową głównej przekątnej** i upewniamy się że wygenerowana macierz jest **symetryczna i dodatnio określona**

Macierze są potem zapisywane do pliku w formacie .npz, łącznie z ich wartościami własnymi, tak by skrócić działanie programu i ujednolicić testy

Macierze nemeth12 i poli3 są pobierane ze strony podanej wyżej, dla macierzy nemeth12 aby spełnić warunki stosowalności metody musieliśmy przemnożyć ją przez -1

Testy Do testów wykorzystujemy biblioteki **numpy** oraz **pytest** oraz wbudowane w Pythona narzędzia do mierzenia czasu.

Sprawdzamy poprawność naszych algorytmów poprzez porównanie naszych wyników z wynikami policzonymi przy wykorzystaniu funkcji `np.linalg.norm` z biblioteki `numpy`. Jeżeli nasze rozwiązanie różni się od rozwiązania `numpy` o mniej niż 8×10^{-3} akceptujemy je jako poprawne

Zarówno wielkość macierzy, jej typ i typ metody użytej do zrównoleglenia Richardsona jest podawana jako parametr testów, pozwala nam to łatwo dodawać nowe metody zrównoleglenia bez zmiany kodu testów.

Funkcje pomocnicze Wszelkie podstawowe metody operacji na macierzach takie jak mnożenie wektorów, macierzy itp, napisaliśmy od zera, bez użycia zewnętrznych bibliotek, funkcje są zdefiniowane w pliku **linear_algebra_utils.py**

Metoda Richardsona Metoda Richardsona jest zaimplementowana w pliku **richardson_method.py**, sprowadza się ona do pętli:

```
1     for iteration in range(self.max_iterations):
2         Ax = self.LinAlg.matrix_vector_multiply(self.A, x)
3         residual = self.LinAlg.vector_vector_subtraction(
4             self.b, Ax)
5         x = self.LinAlg.vector_vector_addition(
6             x,
7             self.LinAlg.scalar_vector_multiply(self.omega,
8                 residual)
9         )
10    if self.LinAlg.SequentialLinearAlgebraUtils.
11        vector_norm(residual) < self.tol:
12        break
```

Listing 1: Python Code for Iterative Solver

Dla różnych metod zrównoleglenia stosujemy różne implementacje podstawowych funkcji odpowiedzialnych za mnożenie macierzy przez wektor, odejmowanie wektorów itp. Ponownie, dzięki temu możemy łatwo dodawać nowe metody zrównoleglenia bez zmiany podstawowego kodu Richardsona

3 Zrównoleglenie