Metody numeryczne – Projekt nr 2

Układy równań liniowych

Jakub Stachowicz 188888 gr.5 Informatyka 23.04.2023r.

1. **Wstęp**

Głównym celem projektu jest implementacja algorytmów iteracyjnych (Jacobiego i Gaussa-Seidla) i bezpośrednich (faktoryzacja LU) do rozwiązywania układu równań liniowych. Macierze używane w tych algorytmach zostały zaprojektowane zgodnie z niżej opisanymi wymaganiami. W ramach projektu została utworzona i zaimplementowana klasa „Matrix”, która ma wiele podobnych funkcjonalności co wbudowana biblioteka „numpy”.

1. **Analiza zadania**

* Zadanie A:

Dla indeksu 188888 została utworzona macierz kwadratowa **A** o:

* wymiarach 988x988,
* a1 = 13,
* a2=a3=-1,

Oraz wektor **b** o:

* długości 988,
* n-ty element ma wartość sin(988\*9) = 0.4121184852417566
* Zadanie B:

Dla ww. Macierzy A i wektora b została przeprowadzona metoda Jacobiego, oraz Gaussa-Seidla. Algorytmy wykonywały się do momentu otrzymania normy z wektora residuum równą 10−9. Poniżej przedstawione są wyniki (ilość iteracji, oraz czas wykonywania) dla poszczególnych metod:

Text

Description automatically generated

Metoda Gaussa-Seidla wykonała się szybciej w mniejszej ilości iteracji.

* Zadanie C:

Została utworzona macierz kwadratowa **A** o:

* wymiarach 988x988,
* a1 = 3,
* a2=a3=1,

oraz wektor **b** z zadania A.

1. Rezultaty dla metody Jacobiego:

Początkowa norma:

Text

Description automatically generated

Norma po 200 iteracjach:

Text

Description automatically generated

1. Rezultaty dla metody Gaussa-Seidla:

Początkowa norma:

Text

Description automatically generated

Norma po 200 iteracjach:

Text

Description automatically generated

Jak można zauważyć dla obu metod norma rośnie cały czas w górę co oznacza, że ww. metody nie zbiegają się. Powyższe wartości wynikają z powodu, gdyż wartość na przekątnej jest mniejsza od sumy pozostałych wartości w wierszach 3<4, więc jest to macierz o słabej dominacji diagonalnej (weakly diagonally dominant matrix).

* Zadanie D:

Została utworzona macierz kwadratowa **A** i wektor **b** z zadania C. Dla nich została wykonana faktoryzacja LU w celu obliczenia normy residuum.

Rezultat:



Norma residuum jest bardzo niska (bliska zero), co oznacza, że algorytm posiada wysoką dokładność obliczeniową.

* Zadanie E:

Chart, line chart

Description automatically generatedPoniższy wykres przedstawia porównanie wykonania algorytmów na macierzy o wymiarach N = {100, 500, 1000, 2000, 3000 }. Zamiast faktoryzacji LU został użyty algorytm Choleskiego, gdyż jego czas obliczeniowy jest około 2 razy krótszy.

* Zadanie F:

Podsumowując powyższe zadania można zauważyć ogromną różnice w czasie wykonywania algorytmów iteracyjnych i bezpośrednich. Algorytmy iteracyjne zbliżają się do prawdziwego wyniku w kolejnych iteracjach i obliczenia kończą się w momencie przekroczenia thresholdu „progu wyjściowego”. Pomimo, że nie mamy dokładnego wyniku, ta metoda jest o wiele bardziej praktyczna, gdyż ich złożoność obliczeniowa to O(n^2), natomiast w metodzie bezpośredniej jest to już O(n^3).

1. **Opis plików**

GitHub: <https://github.com/mikitomi21/System_Of_Linear_Equations>

1. Main.py -> Główny plik to rozwiązywania zadań i przedstawiania ich wyników
2. Matrix.py -> Klasa przedstawiająca macierz i jej operacje
3. Methods.py -> Algorytmy rozwiązywania liniowych układów równań
4. **Zakończenie**

W projekcie zostały przedstawione różnice pomiędzy metodami iteracyjnymi i bezpośrednimi z przewagą obliczeniową dla pierwszych. Dla dużych macierzy złożoność O(n^3) jest zdecydowanie za duża i często może trwać parę godzin, dni, miesięcy, … . Algorytmy iteracyjne przyśpieszają zdecydowanie moc obliczeniową obliczania rozwiązania liniowych układów równań.