Układy równań liniowych – metody bezpośrednie.

Bartłomiej Kozera

Bartłomiej Kozera

1. Informacje techniczne

Obliczenia zostały wykonane na 64 bitowej wersji systemu Windows 10 Pro, z procesorem Ryzen7 3750H oraz z 16 GB pamięci RAM. Biblioteka Pandas odpowiada za wypisywanie danych na standardowe wyjście oraz zapis danych do pliku formatu .xlsx. Biblioteka numpy odpowiada za działania na macierzach, typowania zmiennych w tym języku.

# Metoda Gaussa

Obraz zawierający tekst, Czcionka, pismo odręczne, diagram

Opis wygenerowany automatycznie

Obraz zawierający tekst, Czcionka, zrzut ekranu, diagram

Opis wygenerowany automatycznie

Obraz zawierający tekst, Czcionka, zrzut ekranu, pismo odręczne

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Złożoność obliczeniowa tego algorytmu to O().

# Zadanie 1

Elementy macierzy A o wymiarze n x n są określone wzorem:

Przyjmij wektor x jako dowolną n−elementową permutację ze zbioru { 1, -1 } i oblicz wektor b. Następnie metodą eliminacji Gaussa rozwiąż układ równań liniowych Ax=b (przyjmując jako niewiadomą wektor x). Przyjmij różną precyzję dla znanych wartości macierzy A i wektora b. Sprawdź, jak błędy zaokrągleń zaburzają rozwiązanie dla różnych rozmiarów układu (porównaj – zgodnie z wybraną normą – wektory x obliczony z x zadany). Przeprowadź eksperymenty dla różnych rozmiarów układu.

Rozmiary układu które zostały przetestowane: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 30, 50 ,100. Przyjęta precyzja to float32, double (odpowiadająca float64), longdouble (odpowiadająca dloat128) Z biblioteki numpy. Błędy obliczane były jako norma Frobeniusa, która wyraża się wzorem:

Tabela 1. Błędy powstałe przy obliczaniu wektora X, dla różnych precyzji

|  |  |  |  |
| --- | --- | --- | --- |
| **Rozmiar macierzy** | **float32** | **float64** | **float128** |
| 2 | 0,000000000000 | 0,000000000000 | 0,000000000000 |
| 3 | 0,000001115104 | 0,000000000000 | 0,000000000000 |
| 4 | 0,000013340415 | 0,000000000000 | 0,000000000000 |
| 5 | 0,000608187644 | 0,000000000009 | 0,000000000004 |
| 6 | 0,050900953357 | 0,000000000364 | 0,000000000049 |
| 7 | 0,856667861209 | 0,000000013609 | 0,000000002908 |
| 8 | 3,642708638239 | 0,000000120335 | 0,000000080933 |
| 9 | 2,923197641240 | 0,000000540051 | 0,000002930454 |
| 10 | 6,417048863860 | 0,000166203405 | 0,000016906836 |
| 11 | 5,751244994527 | 0,012234080758 | 0,000468217286 |
| 12 | 10,347249015588 | 1,213975651780 | 0,063597590976 |
| 13 | 113,677878177354 | 21,215106178492 | 0,692350424898 |
| 14 | 2,071366598910 | 21,115595078870 | 0,493772558251 |
| 15 | 16,312666424731 | 15,048517395802 | 1,225054409577 |
| 20 | 3,668081115407 | 871,401137695759 | 99,682551149197 |
| 30 | 27,856895232617 | 189,245712824525 | 10,818550016853 |
| 50 | 67035,367072224300 | 246,085913885037 | 22,344965632748 |
| 100 | 182,321454351423 | 3777,059093080890 | 76,142672130272 |
| 200 | 154,393709010716 | 3428,305964496290 | 239,700505571498 |

Tabela 2. Czasy wykonania algorytmu w sekundach dla różnych dokładności.

|  |  |  |  |
| --- | --- | --- | --- |
| **Rozmiar macierzy** | **czas dla float32** | **czas dla float64** | **czas dla float128** |
| 2 | 0,00 | 0,00 | 0,00 |
| 3 | 0,00 | 0,00 | 0,00 |
| 4 | 0,00 | 0,00 | 0,00 |
| 5 | 0,00 | 0,00 | 0,00 |
| 6 | 0,00 | 0,00 | 0,00 |
| 7 | 0,00 | 0,00 | 0,00 |
| 8 | 0,00 | 0,00 | 0,00 |
| 9 | 0,00 | 0,00 | 0,00 |
| 10 | 0,00 | 0,00 | 0,00 |
| 11 | 0,00 | 0,00 | 0,00 |
| 12 | 0,00 | 0,00 | 0,00 |
| 13 | 0,00 | 0,00 | 0,00 |
| 14 | 0,00 | 0,00 | 0,00 |
| 15 | 0,00 | 0,00 | 0,00 |
| 20 | 0,00 | 0,00 | 0,00 |
| 30 | 0,01 | 0,01 | 0,01 |
| 50 | 0,04 | 0,04 | 0,03 |
| 100 | 0,27 | 0,26 | 0,26 |
| 200 | 2,09 | 2,04 | 2,27 |

Możemy zauważyć że dla zwiększających się wymiarów macierzy, dokładność obliczeń zmniejsza się. Dokładność obliczeń zmienia się także dla różnej architektury liczby zmiennoprzecinkowej. Im większą dokładność oferuje nam dana architektura liczby, tym większą dokładność otrzymujemy, z wyjątkiem macierzy 100 x 100 oraz 200 x 200, gdzie float32 uzyskuje lepszą dokładność niż float64. Czasy działania algorytmów praktycznie takie same dla każdej precyzji. Różnice zauważalne dopiero dla n = 50, 100, 200.