MOwNiT - Rozwiązywanie układów równań liniowych metodami bezpośrednimi

Jakub Frączek

9 czerwca 2024

1 Wstęp

Tematem ćwiczenia było zaimplementowanie algorytmu Gaussa oraz Thomasa do rozwiązywania układów równań liniowych, a następnie rozwiązanie 3 ćwiczeń polegających na przeanalizowaniu wyników otrzymanych przy użyciu tych algorytmów w zależności od zadancyh układów oraz przyjętych precyzji oraz przygotowanie sprawozdania.

1.1 Ćwiczenie 1

Dla macierzy zadanej wzorem:

$$\begin{cases} a_{ij} = 1, & \text{i != 1.} \\ a_{ij} = 1/(i+j-1), & \text{otherwise.} \end{cases} \tag{1}$$

Gdzie $i, j = 1, 2, ...n$
Przyjąć wektor X jako dowolną n-elementowąpermutacją ze zbioru -1, 1 i obliczyć wektor B ze wzoru $A \cdot X = B$. Następnie dla różnych precyzji i rozmiarów macierzy A i wektora B wyliczyć rozwiązanie ukłądu metodą Gaussa i porównać błędy zaokrągleń.

1.2 Ćwiczenie 2

Dla macierzy danej wzorem:

$$\left\{ \begin{array}{ll} a_{ij} = \frac{2i}{j} & \text{dla } j \geq i \\ a_{ij} = a_{ji} & \text{dla } j < i \end{array} \right. \qquad \text{dla } i, j = 1, \ldots, n$$

Powtórzyć eksperyment z ćwiczenia 1, porównać różnice w wynikach i policzyć uwarunkowanie obu układów.

1.3 Ćwiczenie 3

Dla macierzy danej wzorem:

$$\left\{ \begin{array}{ll} a_{i,j} = k & \\ a_{i,j+1} = \frac{1}{i+m} & \\ a_{i+1,j} = \frac{k}{i+m+1} & \text{dla } i > 1 \\ a_{i,j} = 0 & \text{dla } j < i-1 \quad \text{oraz} \quad j > i+1 \end{array} \right. \qquad i, j = 1, \ldots, n$$

Powtórzyć eksperyment z ćwiczenia 1 i 2, porównać różnice w wynikach oraz policzyć uwarunkowanie obu układów. Dodatkowo przeprowadzić analizę dla algorytmu `Thomasa` dla macierzy trójdiagonalnej.

Parametry k oraz m przydzielone w zadaniu indywidualnym wynosiły:

$$k = 6, \ m = 3$$

2 Dane techniczne

2.1 Hardware

Do przeprowadzenia testów wykorzystany został laptop o następującej specyfikacji:

- Procesor `Intel Core i5-9300H 2.4GHz`
- `32 GB` pamięci `RAM`.

2.2 Software

Wykorzystany został system `Windows 11 x64` oraz język `Python` w wersji `3.11.8` wraz z bibliotekami:

- `numpy`
- `random`
- `time`
- `functools`
- `typing`
- `warning`

Do stworzenia wykresów wykorzystane zostało narzędzie `Google Sheets`.

3 Wyznaczanie błędu obliczeń

Błąd obliczony został jako norma euklidesowa różnicy oczekiwanego i otrzymanego wektora. Zakładając, że w_1 to wynik wzorcowy, a w_2 to wynik otrzymany za pomocą algorytmu `Gaussa` lub `Thomasam` różnicę otrzymujemy w następujący sposób:

$$v = w_1 - w_1$$

A następnie liczona jest z niej norma euklidesowa, którą przyjmuję jako błąd obliczeń.

$$\|\mathbf{v}\|_2 = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$$

Funkcja wyliczająca normę z wektora została zaimplementowana przeze mnie.

4 Wyznaczanie uwarunkowania układu

Współczynnik uwarunkowania został obliczony według poniższego wzoru:

$$k(A) = ||A^{-1}|| \cdot |A|$$

gdzie A - macierz zadana wzorem z ćwiczenia 1, 2 oraz 3

Ponownie funkcja wyliczająca normę z wektora została zaimplementowana przeze mnie, natomiast do wyznaczania odwrotności macierzy wykorzystałem funkcję `np.linalg.inv()` z biblioteki `numpy`.

5 Sposób implementacji

Algorytm Thomasa został zaimplementowany w wersji z oszczędnością pamięci. Tj. operuje na trzech listach zamiast na całej macierzy.

Implementacja obu algorytmów została zaadoptowana ze źródeł podanych w ostatnim paragrafie.

6 Wyniki

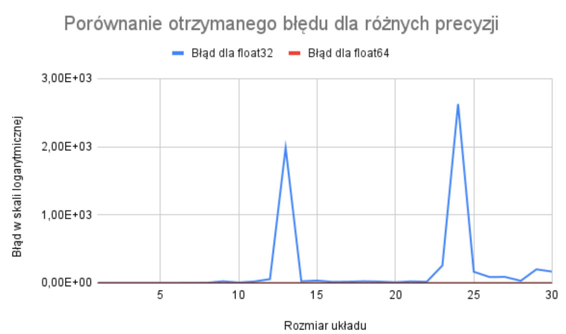
Podczas przeprowadzania testów wykorzystałem dwie różne precyzje liczby zmiennoprzecinkowej udostępnione przez bibliotekę `numpy`:

- `np.float32`
- `np.float64`

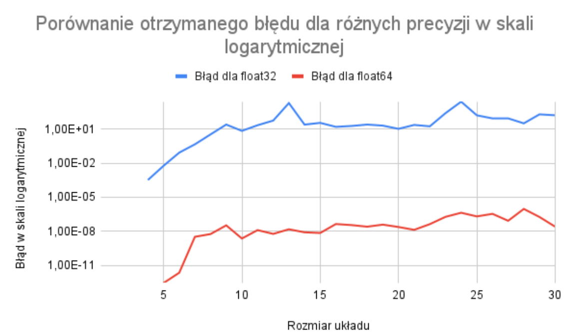
W miejscach, gdzie wykres w skali liniowej nie obrazował dobrze badanej zależności, wykorzystałem wykres w skali logarytminczej.

6.1 Zadanie 1

Jak widać na `wykresie 1` oraz `wykresie 2` błąd przybliżenia bardzo szybko rośnie wraz ze zwiększeniem rozmiaru układum dodatkowo błąd jest znacznie większy dla 64 bitowej precyzji. Generalnie przybliżenie jest złe.



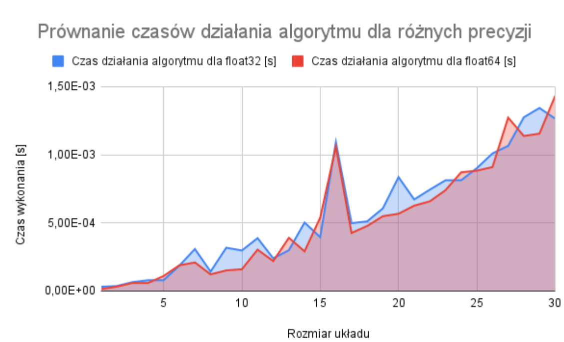
Wykres 1: Wykres błędu w zależności od przyjętej precyzji



Wykres 2: Wykres błędu w zależności od przyjętej precyzji w skali logarytmicznej

Po przeanalizowaniu `Wykreu 3` okazało się, że uwarunkowanie układu jest bardzo złe, a współczynnik uwarunkowania rośnie niezwykle szybko wraz z wzrostem rozmiaru układu. Na `wykresie 4` zaprezentowany jest czas działania algorymu. Jak można było się spodziewać algorytm działa w czasie sześciennym. Ciekawą rzeczą jest natomiast to, że algorytm wykonuje się nieznacznie szybciej dla wersji 64 bitowej. Dzieje się tak dlatego, że dla 32 bitowego float'a tracimy część informacji i wyznaczenie wyniku może zająć dłużej.

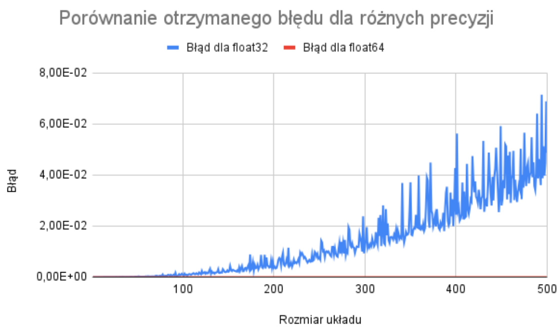
Wykres 3: Wykres uwarunkowania układu w skali logarytmicznej w zależności od przyjętej precyzji



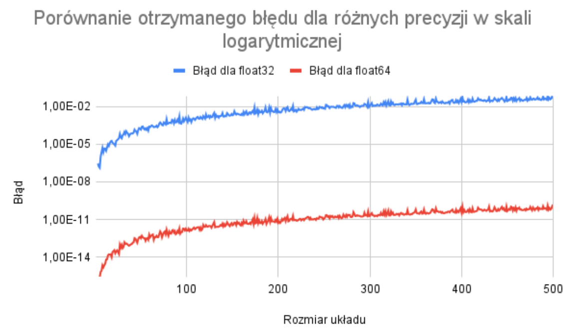
Wykres 4: Wykres czasów działania algorytmu w zależności od przyjętej precyzji

6.2 Zadanie 2

Jak widać na `Wykresie 5` i `Wykresie 6` błąd otrzymany dla 64 bitowej precyzji jest minimalny i rośnie bardzo powoli wraz z wzrostem rozmiaru układu. Błąd otrzymany dla 32 bitowej precyzji jest znacznie większy i rośńie dużo szybciej, jednak dla układu o rozmiarze 500 nadal jest on akceptowalny.

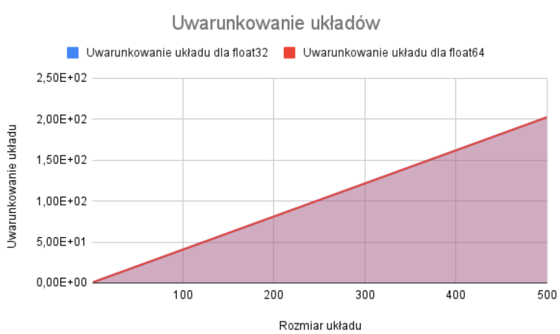


Wykres 5: Wykres błędu w zależności od przyjętej precyzji



Wykres 6: Wykres błędu w zależności od przyjętej precyzji w skali logarytmicznej

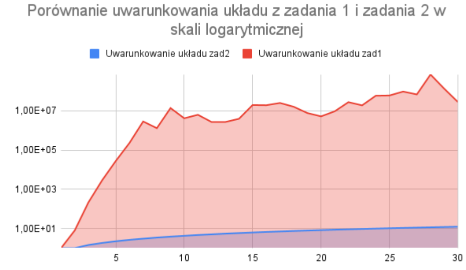
Po spojrzeniu na `wykres 7` widać, że uwarunkowanie układu jest znacznie lepsze niż w przypadku zadania 1 i rośnie liniowo wraz z wzrostem rozmiaru układu. `Wykres 8` ponownie dowodzi sześciennej złożoności algorytmu, choć tym razem nie ma jasnej granicy pomiędzy czasami otrzymanymi dla różnych precyzji.



Wykres 7: Wykres uwarunkowania układu w skali logarytmicznej w zależności od przyjętej precyzji

Wykres 8: Wykres czasów działania algorytmu w zależności od przyjętej precyzji

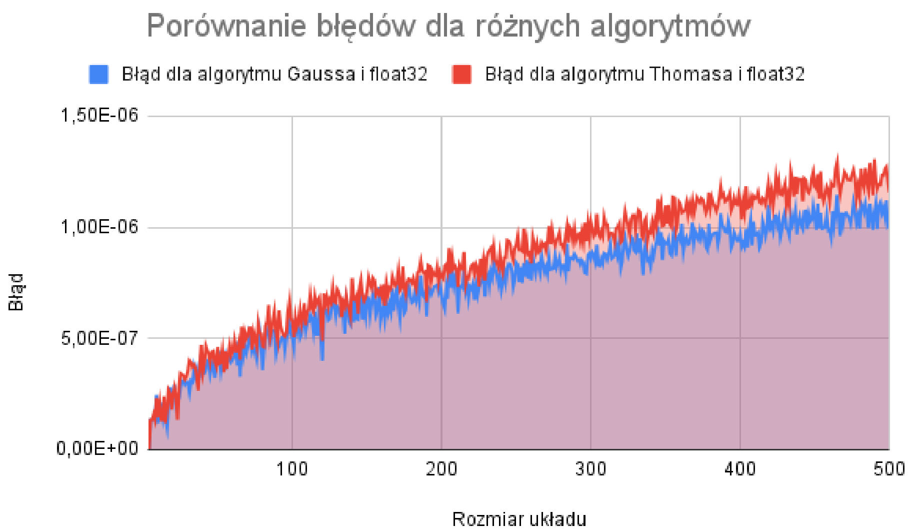
Na `wykresie 9` pokazane zostało porównanie pomiędzy uwarunkowaniem układu z zadania 1, a uwarunmkowaniem układu z zadania 2 dla rozmiarów układu od 1 do 30. Jak widać różnica jest kolosalna co potwierdza, że uwarunkowanie w zadaniu 2 jest znacznie lepsze.



Wykres 9: Porównanie uwarunkowania układu z zadania 1 i zadania 2 w skali logarytmicznej)

6.3 Zadanie 3a - float32

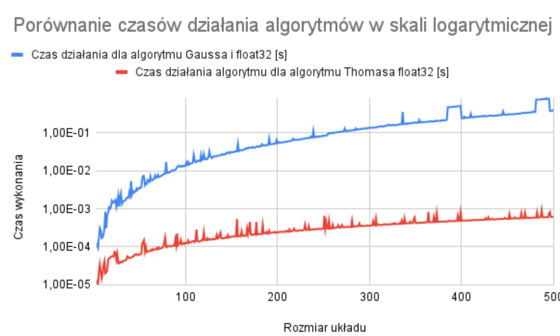
`Wykres 10` porównuje błąd otrzymany dla algrytmu `Thomasa` i algorytmu `Gaussa`. Jak widać, wartości błędów są bardzo zbliżone do siebie, jednak można zauważyć małą różnicę, która prawdopodobnie wynika ze sposobu implementacji obu algorytmów i różnicy w kolejności wykonanych operacji. Jak widać wraz ze wzrosem rozmiaru układu wartości nie odbiegają znacznie od siebie. Otrzymane błędy są bardzo małe i przybliżenie jest dobre.



Wykres 10: Wykres błędów dla algorytmu Gaussa i Thomasa oraz 32 bitowej precyzji

`Wykres 11` i `wykres 12` potwierdzają przewidywane złożoności tj. sześcienna dla algorytmu `Gaussa` i liniowa dla `Thomasa`. Dla dużych rozmiarów układu wykonanie algorytmu `Gaussa` zajmuje aż do 1 sekundy.

Wykres 11: Wykres czasów działania w zależności od przyjętej precyzji



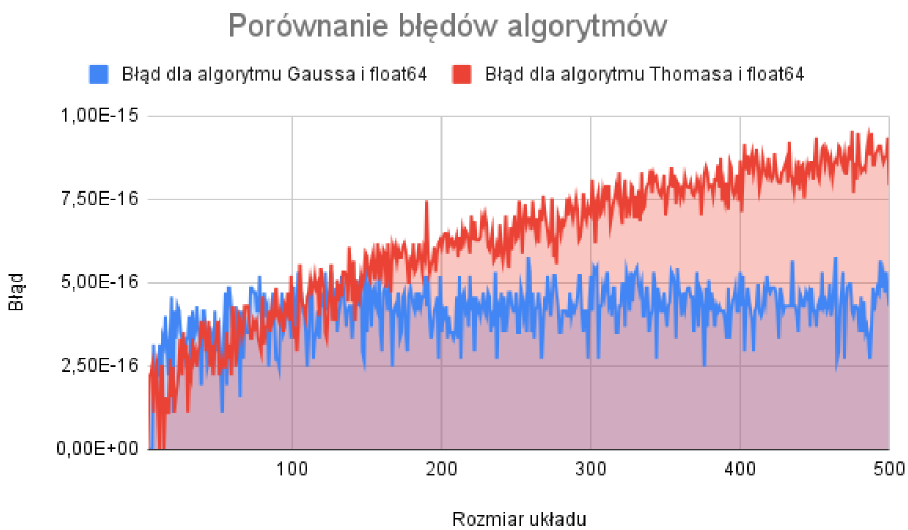
Wykres 12: Wykres czasów działania w skali logarytmicznej w zależności od przyjętej precyzji

Uuwarunkowanie układu jest bardzo dobre i nie zależy od jego rozmiaru.
Współczynnik uwarunkowania dla zadania 3 wynosi:

$$k(A) = 1.16$$

6.4 Zadanie 3a - float64

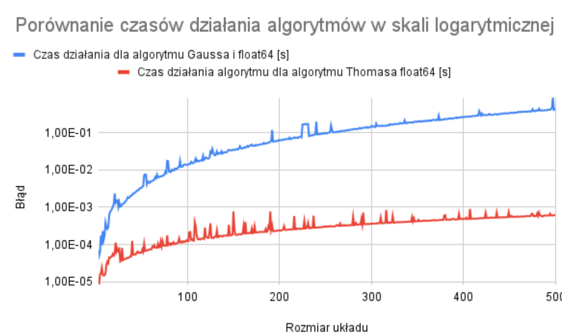
`Wykres 13` porównuje błąd otrzymany dla algrytmu `Thomasa` i algorytmu `Gaussa`. Jak widać, wartości błędów są trochę mniej zbliżone do siebie niż w przypadku precyzji 32 bitowej (wykres 10). Ponownie różnica prawdopodobnie wynika ze sposobu implementacji obu algorytmów i różnicy w kolejności wykonanych operacji. Wraz ze wzrostem rozmiaru układu wartość błędów dla algorytmu Thomasa zwiększa się nieznacznie, jednak nadal jest ona bardzo mała, a otrzymany wynik jak najbardziej akceptowalny.



Wykres 13: Wykres błędów dla algorytmu Gaussa i Thomasa oraz 64 bitowej precyzji

Ponownie `Wykres 14` i `wykres 15` potwierdzają przewidywane złożoności obliczeniowe algorytmu `Gaussa` i `Thomasa`. Wykonanie algorytmu Thomasa nawet dla bardzo dużych rozmiarów układu zajmuje ułamek sekundy.

Wykres 14: Wykres czasów działania w zależności od przyjętej precyzji



Wykres 15: Wykres czasów działania w skali logarytmicznej w zależności od przyjętej precyzji

Zmiana precyzji nie wpłynęła na współczynnik uwarunkowania układu, który ponownie wynosi:

$$k(A) = 1.16$$

7 Wnioski

1. Na podstawie przeprowadzonych testów można zauważyć, że uwarunkowanie układu ma duży wpływa na dokładnośc otrzymanego przybliżenia.
2. W niektórych przypadkach wyniki dla 64 bitowej precyzji są "o niebo" lepsze od wyników dla 32 bitowej precyzji
3. Algorytm Thomasa jest znacznie szybszy od algorytmu Gaussa, jednek ten drugi jest bardziej uniwersalny.

8 Źródła

1. Wykład z przedmiotu MOwNiT prowadzony przez Panią dr. Katarzynę Rycerz
2. https://eduinf.waw.pl/inf/alg/001_search/0076.php
3. https://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm