Dawid Lipiński
Informatyka III
238186

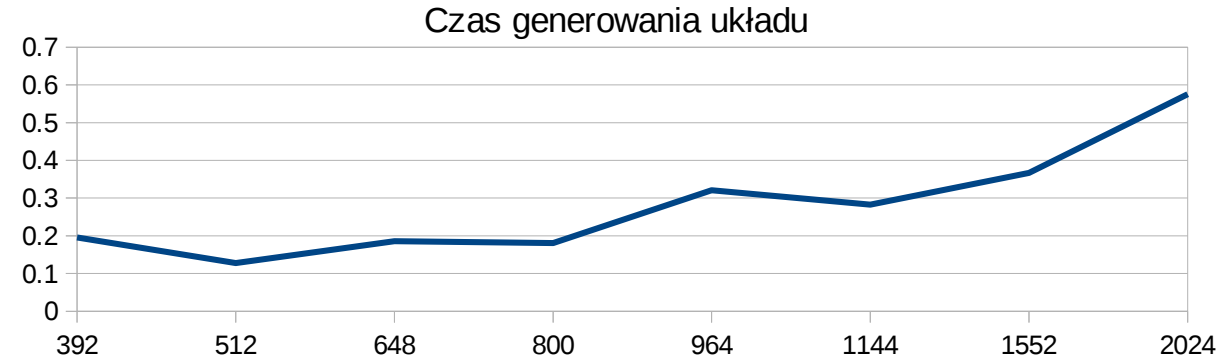
19.01.2018

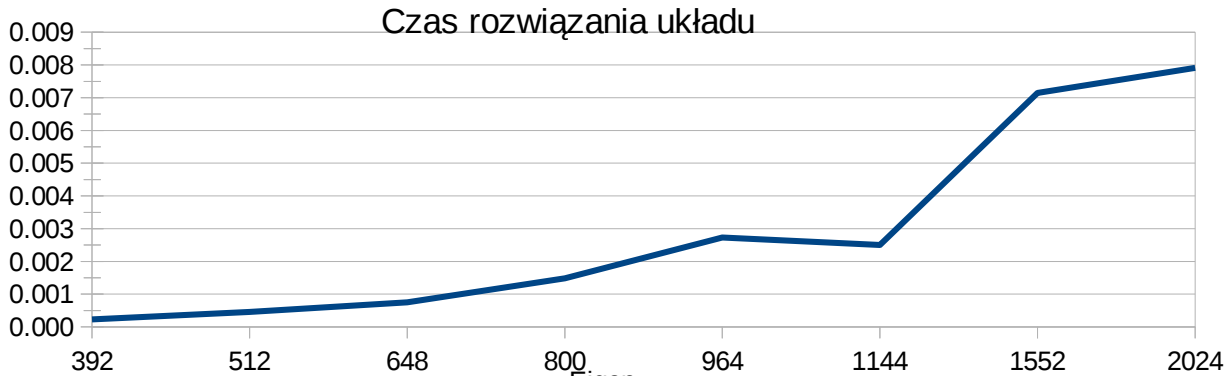
Aproksymacja

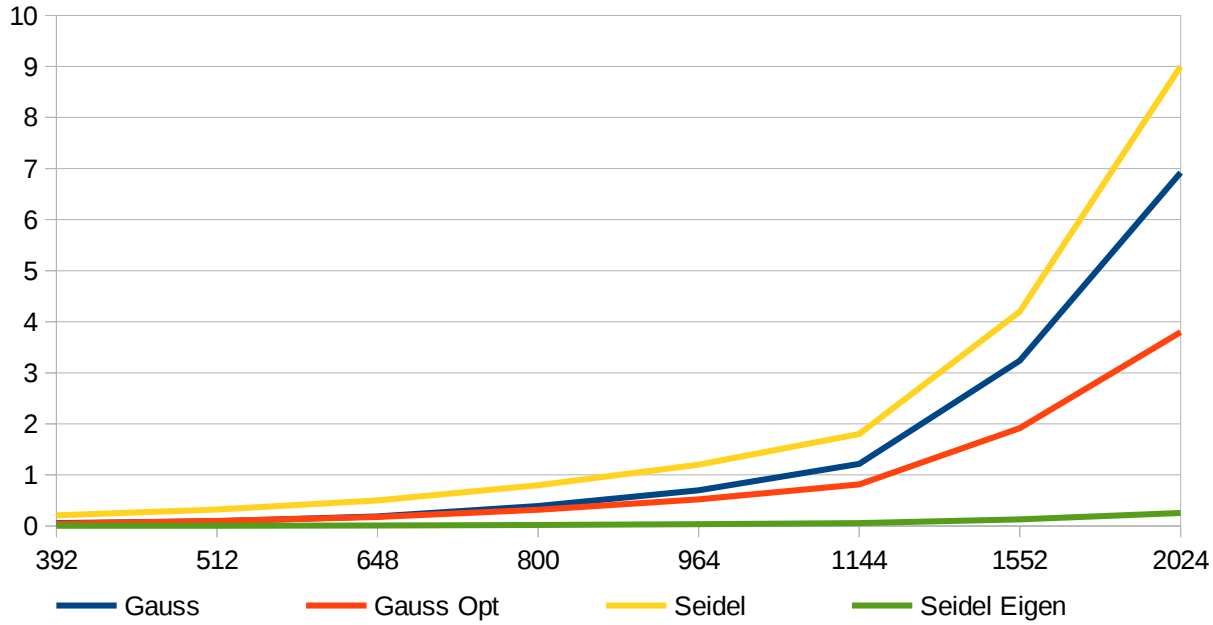
Algorymty 4

Czasy działania

Przeprowadzone zostały pomiary czasu stopniowo zwiększajac rozmiar planszy otrzymując układ od 392 do 2024 równań, stosując metody obliczeniowe. Czas budowania układu został zmierzony oddzielnie od czasu jego rozwiązania.







Wielomiany aproksymacyjne

Znając charakter macierzy opisującej układ równań dla poszczególnych, stosowanych wcześniej metod, stosując aproksymację średniokwadratową dyskretną znalezione zostały wielomiany aproksymacyjne dla każdej z serii pomiarów.[1]

1.Wielomian 3-go stopnia dla metody Gaussa.

$$y(x)=(9.32695\text{e-}10)x^3+(-9.02387\text{e-}07)x^2+(0.00138398)x^1+(-0.489533)$$

2. Wielomian 2-go stopnia dla metody Gaussa z drobną optymalizacją dla macierzy rzadkich.

$$y(x)=(2.71952\text{e-}06)x^2+(-0.00275423)x^1+(0.809337)$$

3. Wielomian 1-go stopnia dla metody LU z wykorzystaniem specjalizowanych struktur danych z bibliotegi Eigen3.

$$y(x)=(5.48418\text{e-}06)x^1+(-0.00273905)$$

4.Wielomian 2-go stopnia dla metody iteracyjnej Gaussa-Seidela przy założonej dokładności 1e-10 (właśna implementacja)

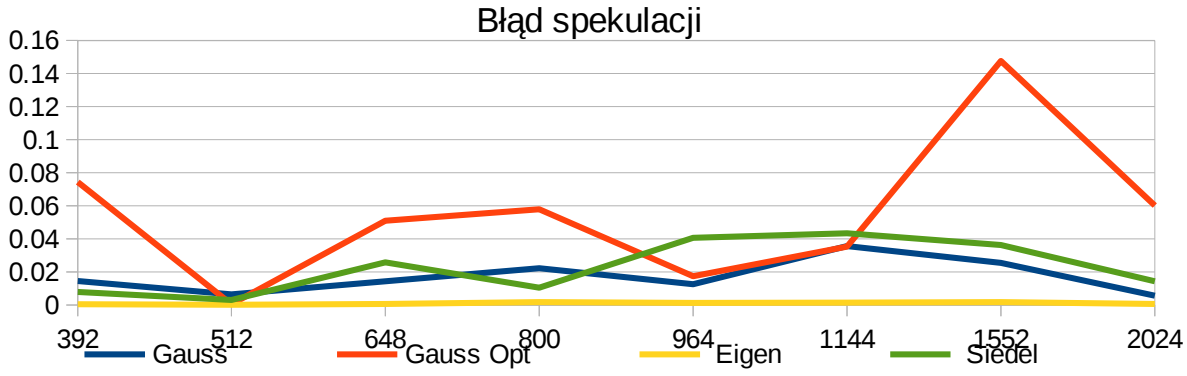
$$y(x)=(4.59203\text{e-}06)x^2+(-0.00456714)x^1+(1.60142)$$

5.Wielomian 1-go stopnia dla metody iteracyjnej Gaussa-Seidela przy założonej dokładności 1e-10 (właśna implementacja z użyciem specjalnych struktur z biblioteki Eigen)

$$y(x)=(0.000746427)x^1+(-0.429188)$$

Poprawność implementacji

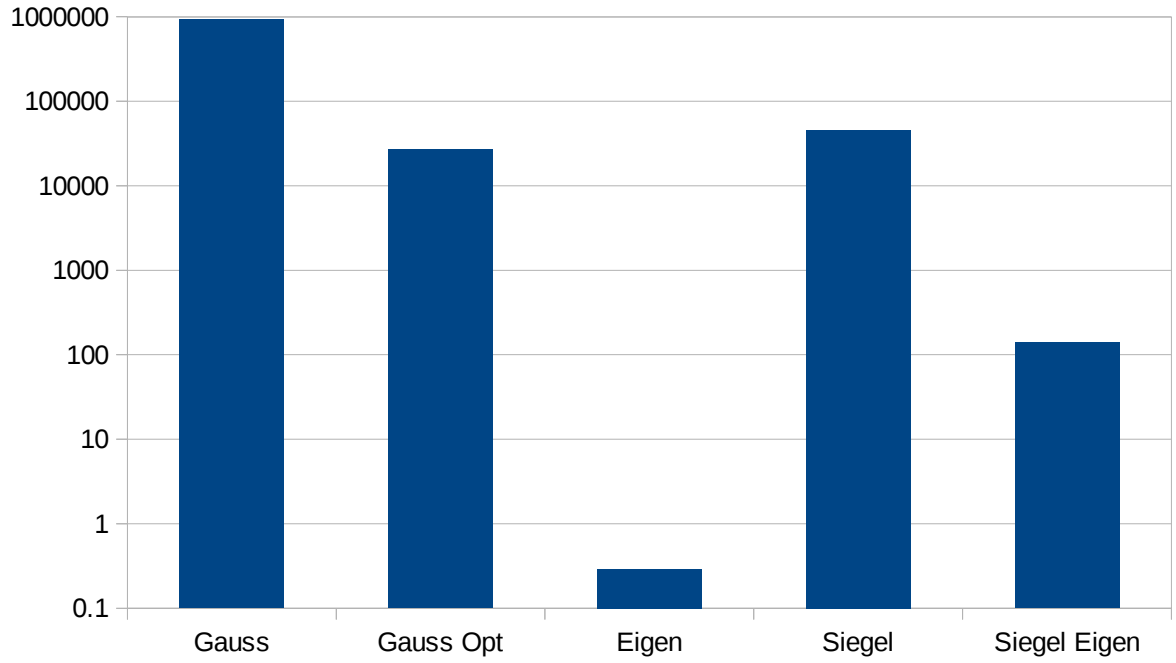
Znając wyniki pomiarów czasów oraz wielomiany aproksymacyjne, możemy zbadać poprawność implementacji. Zauważyć jednak trzeba, iż pomiar czasu obarczony jest pewnym błędem, spowodowanym działaniem wielu innych procesów na komputerze w trakcie działania programu, co z kolei, potęgowane przez bardzo krótki czas wykonywania metod, prowadzi do powstania błędów między wynikiem obliczeń a spekulacją.



Rozwiązanie dużego układu

Mając wielomiany aproksymacyjne, możliwe się stało oszacowanie ile czasu zajęłoby obliczenie układu o wielkości 100 000 równań dla każdej z metod.

Jak widać, zgodnie z oczekiwaniami, najszybszą metodą okazała się implementacja eigen. Zatem tą metodą policzymy wielką macierz 100000x100000



Zaimplementowana została SparseMatrix<double> o wyżej wymienionym rozmiarze oraz SparseVector<double> i wypełnione zostały na wzór poprzednich generowań. Następnie przeprowadzona zostało obliczenie używając metod Sparse (SparseLU, analyzePattern, factorize).

Obliczenia zajęły 0.00745 sekundy, czyli ok. 8 razy szybciej, niż spekulowano.

Podsumowanie

Podsumowując wszystkie zebrane programem infomacje, oraz wiedzę na temat architektury komputerów i idące za tym skutki w postaci błędów w pomiarze czasów, dodatkowo mając wiedzę, iż aproksymacja została przetestowana z wynikiem pozytywnym na innych danych, stwierdzić można, że błąd pomiędzy szacowanym czasem wykonania, a realnym wynika z wyżej wymienionych czynników, niezależnych od twórcy kodu.

1. Implementacja powstała na podstawie wykładów oraz skryptu https://eti.pg.edu.pl/documents/176593/26763380/Wykl_AlgorOblicz_3.pdf , z racji możliwości kontroli działania na przykładzie drugiego na bieżąco w trakcie pisania programu.