Metody Obliczeniowe w Nauce i Technice
Sprawozdanie 1 - Algebra liniowa
Jakub Synowiec

31.03.2015r.

Cele ćwiczenia:

Implementacja algorytmu rozwiązującego układ równań zapisany przy pomocy macierzy trójdiagonalnej metodą dekompozycji LU. Porównanie wydajnościowe czasu pracy zaimplementowanego algorytmu z funkcją gsl_linalg_solve_tridiag z biblioteki Gnu Scientific Library. Badanie czasu obliczenia operacji BLAS poziomu 1, 2 I 3.

Wstęp:

Algorytmy zostały zaimplementowane w języku C, kompilowane przy użyciu GCC w wersji 4.8.2, użyta została biblioteka Gnu Scientific Library. Testy zostały przeprowadzone na komputerze z systemem operacyjnym Ubuntu Linux 64 bit. Procesor w tym komputerze taktowany jest na 2.4 GHz. Do utworzenia wykresów użyty został proram gnuplot.

Przebieg ćwiczenia:

1. Funkcja generująca testowe macierze NxN oraz macierze trójdiagonalne o rozmiarze N.

W programie macierze reprezentowane są przez tablice danych o rozmiarze N*N. Element o pozycji i, j został zapisany na miejscu i*N + j. Generowanie liczby losowej zostało zaimplementowane przy pomocy funkcji rand() jako podzielenie przez siebie zmiennoprzecinkowo dwóch liczb całkowitych. Do reprezentacji macierzy trójdiagonalnej użyta została tablica o rozmiarze 3*N, gdzie odpowiednio miejsca i * N, 1 + i * N, 2 + i * N reprezentują dane w macierzy znajdujące się na lewo od przekątnej, na przekątnej i na prawo od przekątnej w i-tym wierszu.
Przy wykonywaniu funkcji z biblioteki GSL użyte zostały odpowiedno zawarte w tej bibliotece wektory gsl_vector oraz macierze gsl_matrix, wypełnione w taki sam sposób losowymi danymi.

2. Program rozwiązujący układ równań metodą dekompozycji LU oraz przy użyciu funkcji z biblioteki GSL.

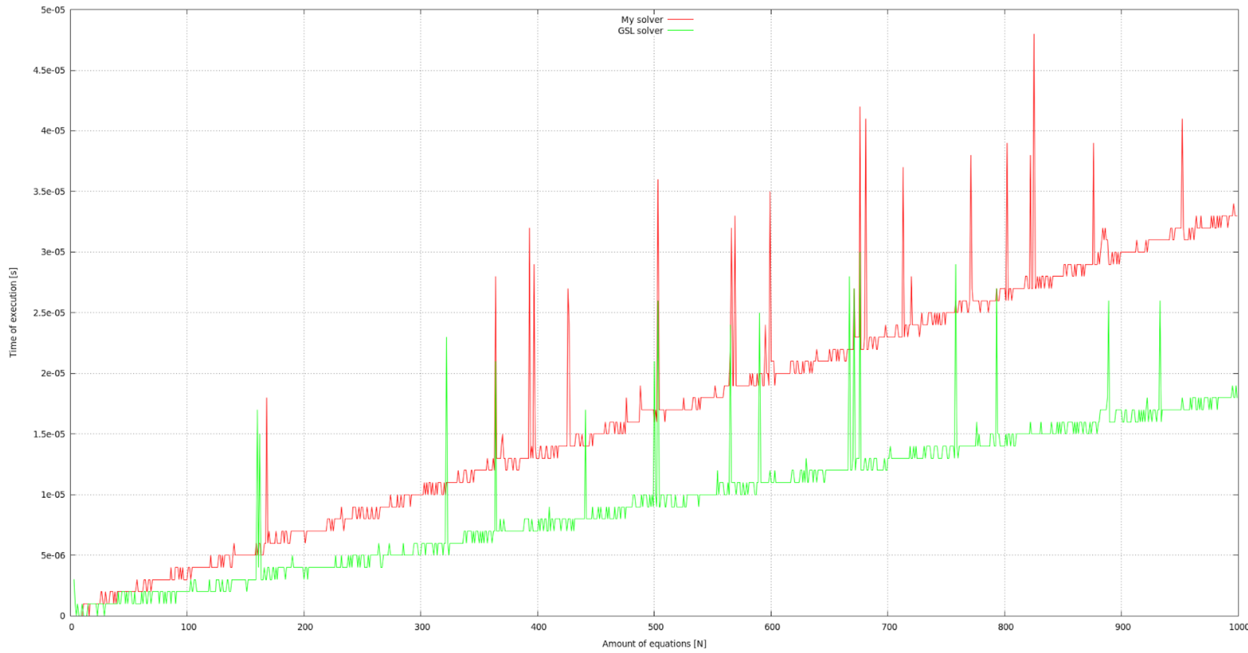
Algorytm rozwiązania układu równań trójdiagonalnych został zaimplementowany w dwóch krokach:

- Dekompozycja macierzy A układu równań na macierze L oraz U.
- Uzyskanie wektora wynikowego X przez operacje na macierzy L, a później U.

Zaimplementowany algorytm dla serii losowych danych zwracał identyczne rezultaty jak funkcja gsl_lialg_solve. Przed użyciem funkcji dane zostały wygenerowane, a później skopiowane, aby obydwie metody mogły działać na identycznym zestawie danych.

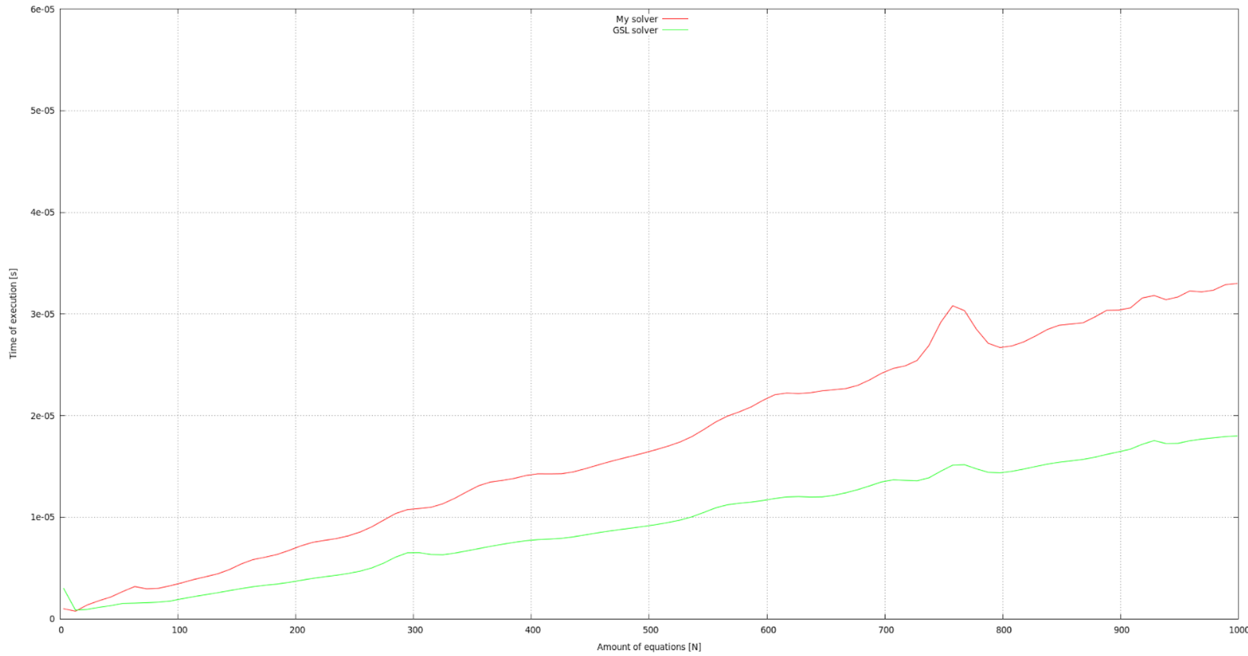
3. Porównanie wydajności powyższych rozwiązań.

Zmierzone zostały czasy wykonywania algorytmów, nie włączając w to alokację pamięci, ani generację danych. Pomiar został wykonany dla obydwóch metod liczących układy od 3 do 1000 równań. Czasy te prezentują się następująco:



Według legendy: My solver - algorytm implementowany, GSL solver - funkcja gsl_linalg_solve.

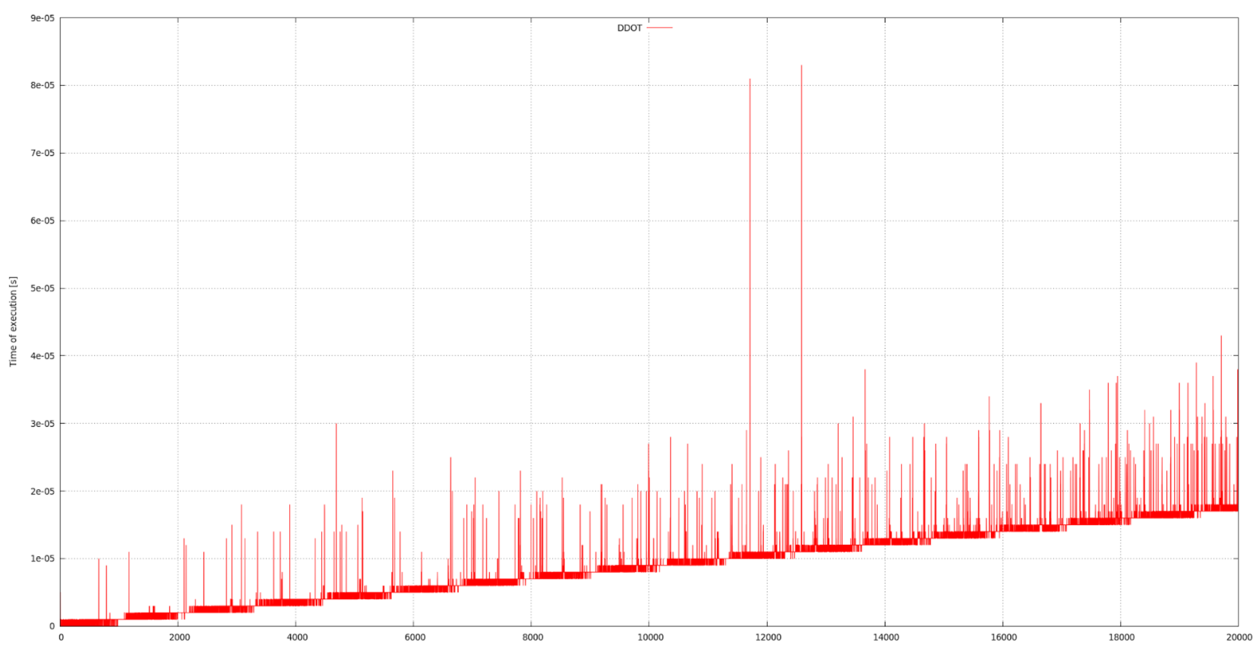
Na wykresie widać pojedyńcze “szpice”, najczęściej zdarzające się zarówno przy algorytmie implementowanym jak i funkcji z biblioteki. Wskazuje to na powstanie wyjątkowo “trudnego” dla obu algorytmu wygenerowanego zestawu danych. Jednak w większości przypadków widać tendencje do liniowego czasu wynonania. Po zastosowaniu operacji wygładzenia wykresu, którą umożliwia program gnuplot uzyskany został następujący efekt:



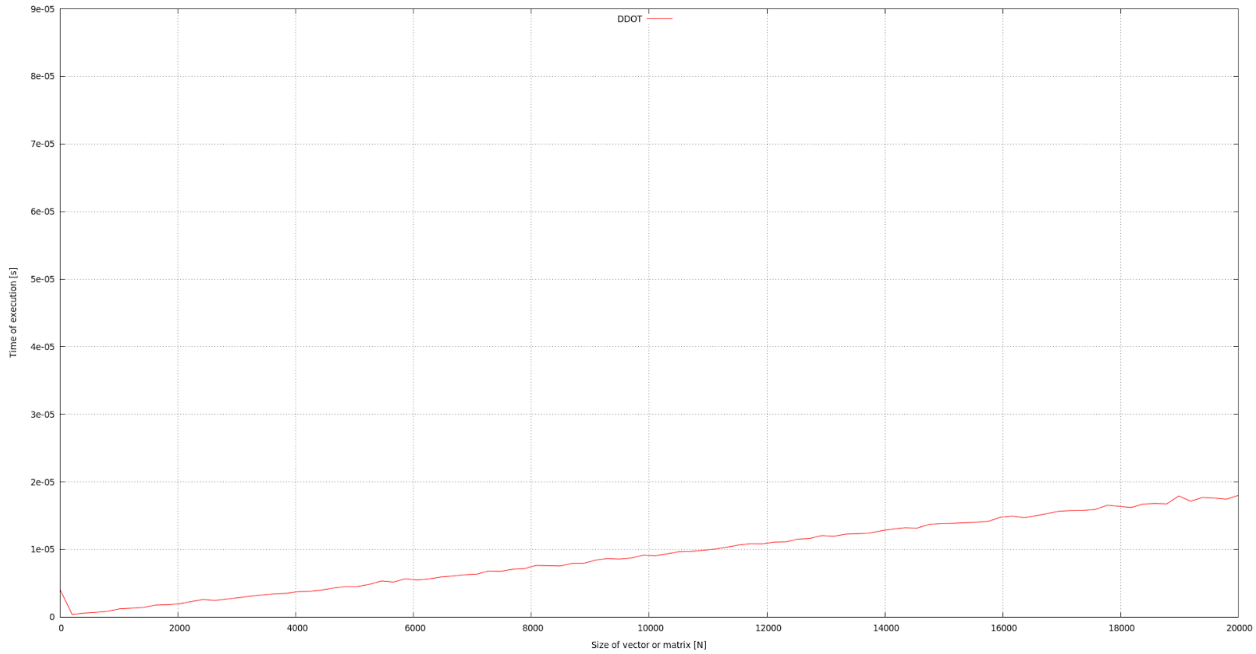
Na powyższym wykresie widać tendencje do liniowego przyrostu czasu wraz z większą ilością danych. Dwukrotny przyrost ilości danych, na przykładzie 500 i 1000 operacji powoduje dwukrotnie dłuższy czas wykonywania tej operacji: wzrost z około 0.9 * 10e-5 do 1.8 * 10e-5 oraz z 1.6 * 10-5 do 3.2 * 10^-5 dla odpowiednio algorytmu implementowanego i algorytmu bibliotecznego.

4. Wykresy i analiza czasu obliczenia operacji BLAS poziomu 1,2,3.

- Operacje poziomu pierwszego:Wykonana została funkcja gsl_blas_ddot obliczająca iloczyn skalarny dwóch wektorów. Otrzymany wykres czasu zamieszczono poniżej:



Tutaj również widać wspomniane w poprzednim punkcie “szpice”. Interpretacja ich powstania jest dokładnie analogiczna. Po “wygładzeniu” wykresu uzyskujemy następujący efekt:



Wnioskując z dwóch powyższych wykresów można stwierdzić, że złożoność czasowa algorytmu jest stała. Widać, że dla dwukrotnego zwiększenia się ilości danych; z 10000 na 20000 czas wzrósł z niecałej 1e-5 do niecałej 2e-5.

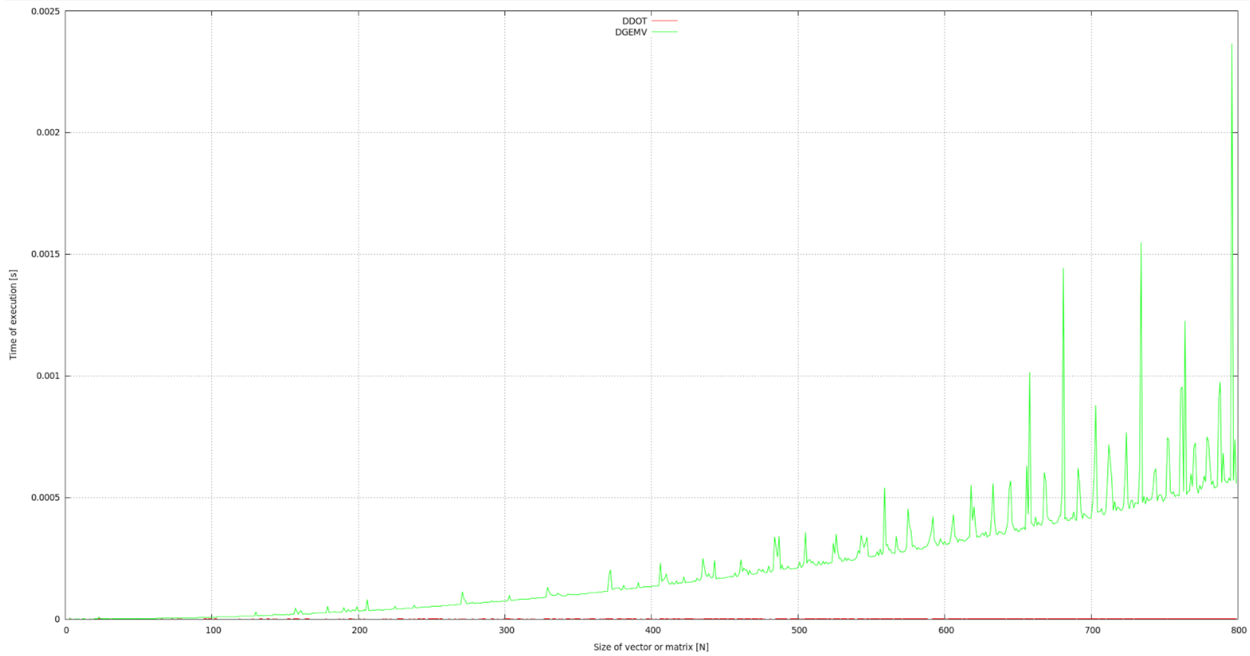
- Operacje poziomu drugiego:

Do operacji poziomu drugiego została wybrana funkcja gsl_blas_dgemv wykonująca operacje:

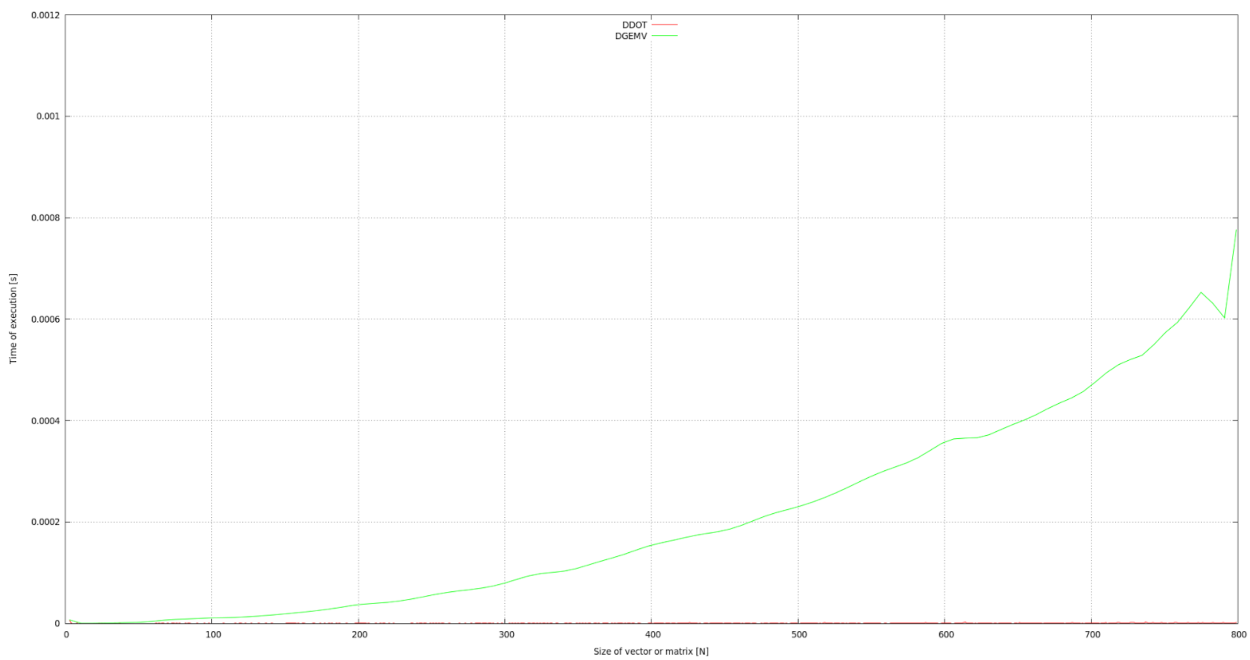
y = alfa * A * x + beta * y,

gdzie: y, x - wektory długości N; A - macierz NxN, alfa, beta - stałe rzeczywiste

Poniższy wykres reprezentuje złożoność czasową tej funkcji, dodatkowo na wykresie umieszczono wcześniejszą funkcje poziomu 1:



Przeprowadzając analogiczny tok myślenia odnośnie wyglądu wykresu, oraz po wygładzeniu otrzymujemy wykres:

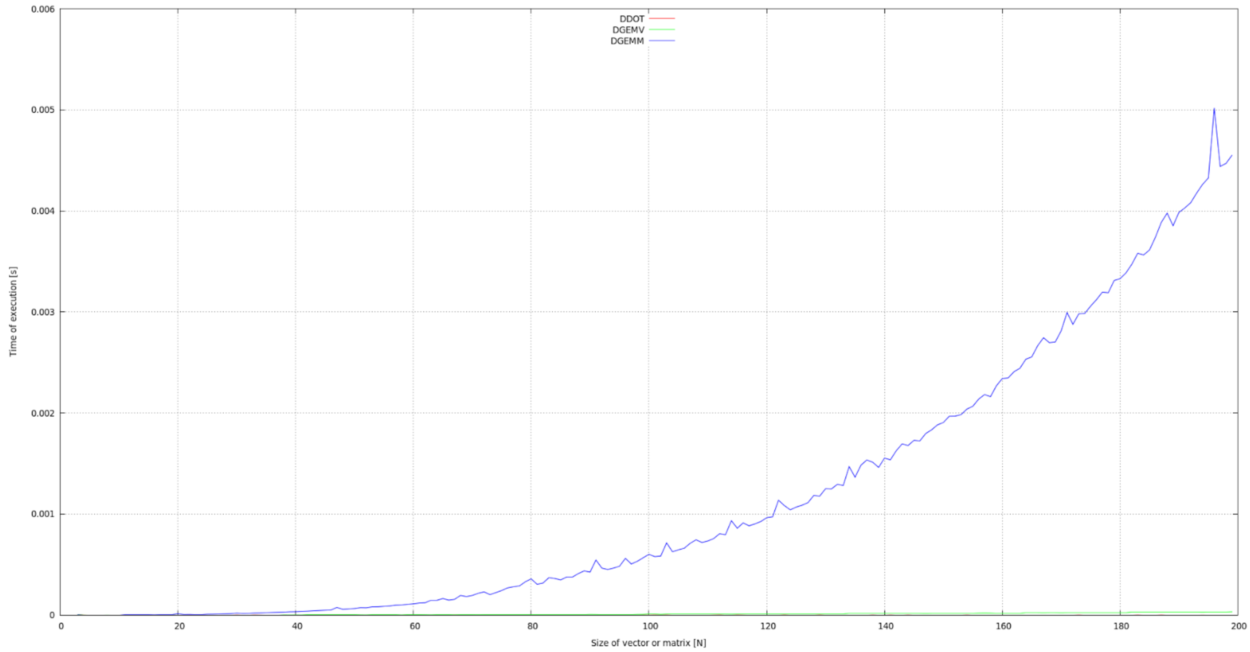


W tym przypadku nie mamy już do czynienia z liniowym wzrostem czasu. Po zwiększeniu ilości danych z 400 na 800 czas wzrósł z ok 0.00016 na niecałe 0.0004. Mamy więc do czynienia ze złożonością kwadratową. Funkcja poziomu 1 którą analizowaliśmy w poprzednim podpunkcie znajduje się bardzo blisko osi X.

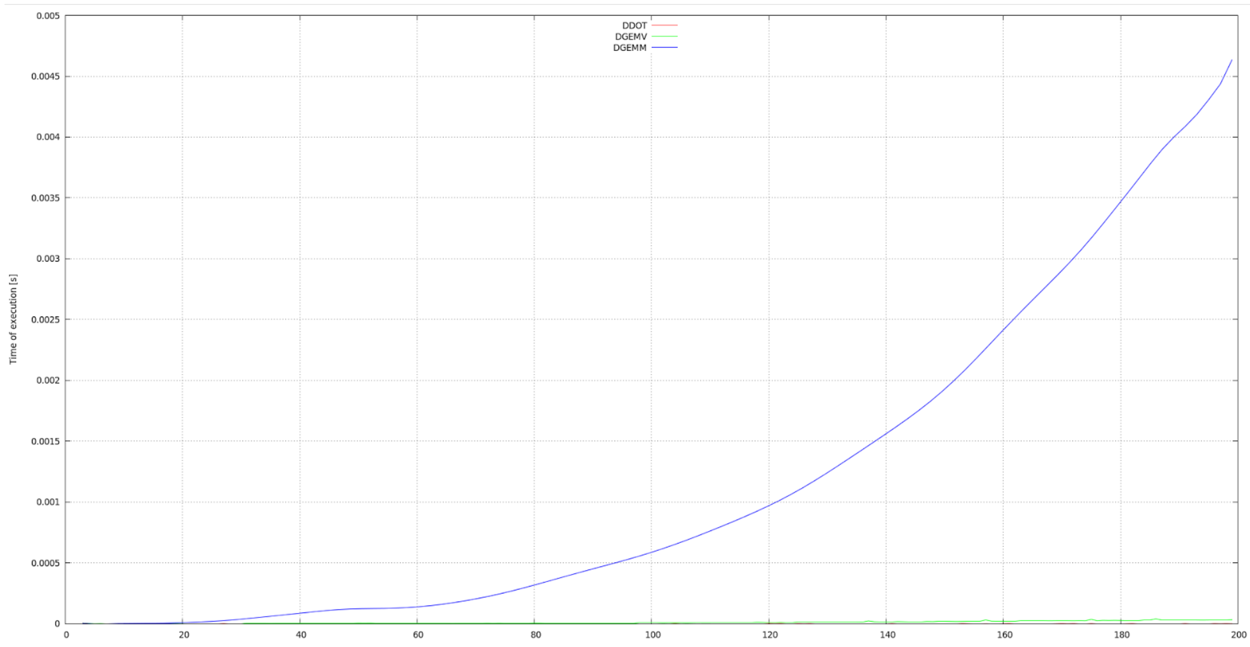
- Operacje poziomu trzeciego:

Funkcja użyta w ćwiczeniu to gsl_blas_dgemm, która wykonuje następującą operacje:

C = alfa * A * B + beta * C, gdzie A, B, C - macierze kwadratowe o rozmiarze NxN, alfa, beta - stałe rzeczywiste. Wykres złożoności czasowej w zależności od ilości operacji wygląda następująco:



Natomiast po “wygładzeniu”:



Na wykresach zostały zawarte wcześniejsze funkcje, poziomu 1 i 2, jednak ich czas wykonywania w porównaniu do funkcji z poziomu 3 jest na tyle mały, że wykresy tych funkcji znajdują się przy osi X.

Patrząc na przytost czasu dla ilości operacji wynoszących 90 i 180 można stwierdzić wzrost czasu z ok 0.0005 do 0.0035, co oznacza w przybliżeniu trzykrotne wydłużenie czasu algorytmu przy dwukrotnie większych danych. Tak więc algorytm ma złożoność N^3.

5. Wnioski z ćwiczeń:

- Czas mierzenia algorytmów dla losowo wygenerowanych danych posiada charakterystyczne "szpice", ponieważ od czasu do czasu powstaje taki zestaw danych, że komputer potrzebuje większej ilości operacji, na przykład w celu zachowania stabilności numerycznej, bądź trudności wykonania operacji elementarnych aby uzyskać prawidłowy wynik.
- Algorytm rozwiązywania układu równań dla macierzy trójdiagonalnych ma złożoność liniową.
- Implementacja tego algorytmu jest wolniejsza od algorytmu z biblioteki o stałą. Jest to spowodowane zapewne mniejszą optymalizacją tego algorytmu.
- Po obserwacji wybranych przykładów z zestawu operacji BLAS zawartych w bibliotece GSL można się domyślać, że algorytmy BLAS poziomu 1, 2 i 3 posiadają odpowiednio złośoności: liniową, kwadratową i sześcienną.

Jakub Synowiec