Algorytmy numeryczne

Zadanie 2
Dawid Bińkuś & Oskar Bir & Mateusz Małecki
grupa 1 tester-programista

11 Listopad 2018

1 Operacje na macierzach

Sprawozdanie prezentuje analizę wydajności i poprawności implementacji algorytmu eliminacji Gaussa, dla losowej macierzy kwadratowej A i wektora B w układzie liniowym $A \cdot X = B$.
Zaimplementowano następujące warianty algorytmu:

G: bez wyboru elementu podstawowego,

PG: z częściowym wyborem elementu podstawowego,

FG: z peanym wyborem elementu podstawowego.

Dodatkowo, obliczenia zostały wykonane, używając trzech różny typów reprezentujących liczbę rzeczywistą:

TF: typ pojedynczej precyzji: **float**

TD: typ podwójnej precyzji: **double**

TC: własna implementacja, przechowująca liczbę w postaci ułamka liczb całkowitych: **fraction**

Jako współczynniki macierzy A oraz wektora X zostały wylosowane liczby zmiennoprzecinkowe z przedziału: $[\frac{-2^{16}}{2^{16}}, \frac{2^{16}-1}{2^{16}}]$ Następnie wektor B został wyliczony według wzoru $B = A \cdot X$. Macierz A i wektor B zostają podane jako parametry do rozwiązania układu równań, wektor X zaś pozostawiamy jako rozwiązanie wzorcowe, za pomocą którego obliczamy błąd wykonanego algorytmu.

Program do realizacji testów został wykonany w języku *Java*. Typ danych *TC* został zaimplementowany za pomocą wbudowanego typu całkowitego *BigInteger*. Testy zostały wykonane na macierzach o rozmiarze $\{10, 20, ..., 500\}$ (**float**,**double**) $\{10, 20..., 150\}$ (**fraction**) w ilości prób malejącej, wraz z wykonywaniem testów na coraz to większych macierzach. Wyniki zostały zagregowane za pomocą średniej arytmetycznej. Wykonywanie testów zajęło około godziny, w zależności od komputera na którym był uruchamiany (50-70 minut)

2 Analiza hipotez

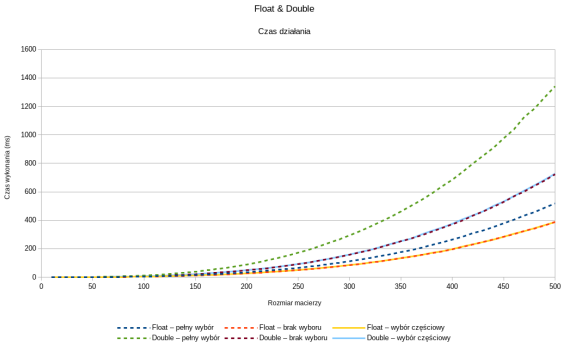
2.1 Związek czasu wykonywania z wariantem algorytmu eliminacji Gaussa

Hipoteza 1 *Dla dowolnego ustalone rozmiaru macierzy czas działania metody Gaussa w kolejnych wersjach (G,PG,FG) rośnie.*

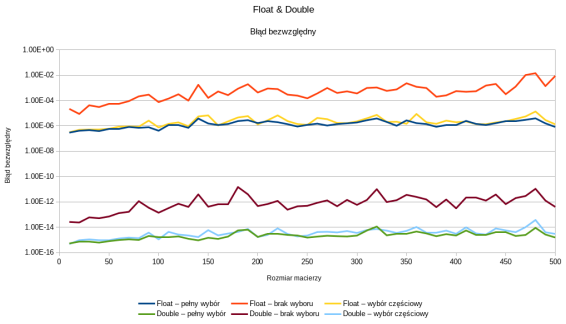
Przeanalizujmy wykresy 1a oraz 1d. Zaprezentowano na nich porównanie czasu wykonania w zależności od wybranego typu danych oraz wariantu algorytmu. Niezależnie od typu danych można zauważyć pewną tendencję. Po pierwsze, wraz z wzrostem rozmiaru macierzy użytej do rozwiązania układu równań, można zauważyć znaczący wzrost czasu wykonywania dla wariantu FG względem wariantów G i PG. Wykonując operacje na małych macierzach proces rozwiązywania nie jest na tyle skomplikowany by w jakikolwiek sposób wpływać na wydajność. Drugą interesującą rzeczą natomiast jest, brak znaczących różnic między wariantem G i PG. By bardziej uwidocznić to zjawisko przeanalizujmy wykres 1c, który prezentuje różnicę między cześciowym wyborem elementu podstawowego a brakiem wyboru w przypadku typu danych float. Wynikają z niego dwie rzeczy: różnica między wykonywaniem tych dwóch argumentów jest praktycznie nieznacząca (zaledwie kilka milisekund). Drugą rzeczą jest natomiast fakt, że wariant PG zdaje się być nieznacznie wolniejszy od wariantu G (Wraz z wzrostem rozmiaru macierzy okazuje sie w niektorych wypadkach nawet wydajniejszy niz standardowy algorytm Gaussa)

Wniosek 1 *Hipoteza okazuje się być cześciowo prawidziwa. Wariant FG powoduje znaczny wzrost czasu wykonania dla wszystkich wykorzystanych typów danych. Interesujący natomiast jest brak znaczących różnic między wariantami PG i G, który jest spowodowany niewielkim wpływem na czas wykonania (w wariancie PG sprawdzamy tylko od n wartości w przeciwieństwie do $n * n$ w wariancie FG).*

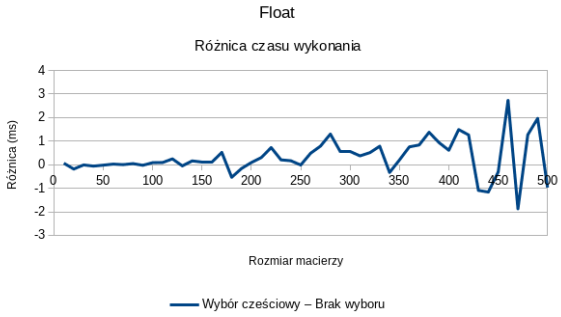
Rysunek 1: Wykresy reprezentujące błąd bezwzględny oraz czas wykonania dla różnych typów danych oraz różnych wariantów algorytmu



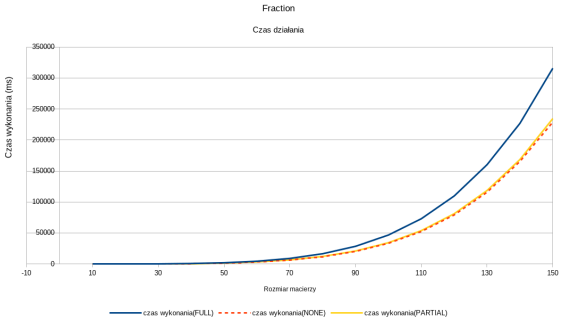
(a)



(b)



(c)



(d)

Fractial - porównanie							
n	Błąd bezwzględny			Czas wykonania			Ilość prób
Wariant ->	G	PG	FG	G	PG	FG	
10	0	0	0	2.39034797	2.70733242	2.62618003	100
20	0	0	0	26.08455804	28.1878754	37.95241776	25
50	0	0	0	1353.7187555	1418.31329225	1935.74870875	4
100	0	0	0	33360.40558	34285.387315	46668.033684	1
130	0	0	0	115763.860973	118493.372512	160113.679247	1
150	0	0	0	228794.993101	234503.568338	315739.697225	1

Rysunek 2: Tabela prezentująca porównanie wyników dla typu Fractial

2.2 Związek błędu obliczeń z wariantem algorytmu eliminacji Gaussa

Hipoteza 2 *Dla dowolnego ustalonego rozmiaru macierzy błąd uzyskanego wyniku metody Gaussa w kolejnych wersjach (G,PG,FG) maleje.*

Przeanalizujmy wykres 1b. Prezentuje on zestawienie błędu bezwzględnego rozwiązania układu równań względem wektora X dla typów danych float i double. W przypadku obu typów danych najmniej dokładny jest wariant G (różnice między nimi są efektem zastosowanej precyzji w tych typach). W przypadku wariantów PG i FG można zauważyć iż mają one bardzo zbliżoną dokładność z małą przewagą wariantu PG - jego dodatkowym atutem jest większa stabliność wyników. W sytuacjach gdy wariant G i PG mają problemy z bardziej skomplikowanymi równaniami (charakterystyczne wzrosty na wykresie) - PG radzi sobie zdecydowanie lepiej.

Wniosek 2 *Hipoteza okazuje się być prawdziwa. Wariant PG zapewnia największą dokładność i stabilność wyników względem nieco gorszego FG oraz G, w którym różnica błędu sięga nawet dwóch rzędów wielkości. Jednakże biorąc pod uwagę wniosek 1 najbardziej opłacalnym w kategorii dokładność/wydajność okazuje się być wariant PG jako iż różnice błędu między nim a pełnym wyborem zdają się być akceptowalne gdy weźmiemy pod uwagę znacznie szybszy czas działania.*

2.3 Poprawność własnej arytmetyki

Hipoteza 3 *Użycie własnej arytmetyki na ułamkach zapewnia bezbłędne wyniki niezależnie od wariantu metody Gaussa i rozmiaru macierzy.*

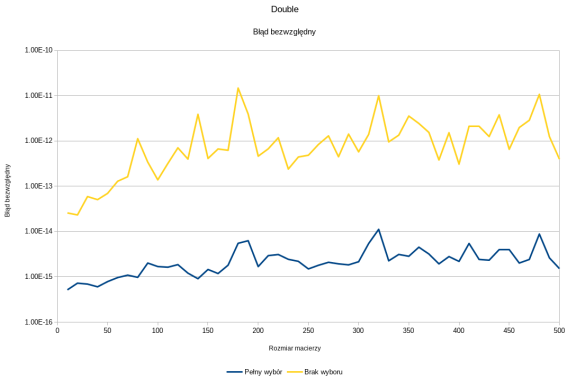
Przeanalizujmy tabelę 2 Prezentuje ona wyniki z użyciem własnej arytmetyki dla wszystkich wariantów algorytmu. Obliczenia zostały wykonane na macierzach o rozmiarze n w przedziale $\{10, 20, ..., 150\}$ z racji ograniczeń sprzętowych. Rozpatrując wyniki testów, można dość do wniosku, że użycie własnej artymetyki całkowicie eliminuje błąd obliczeń. Jest to spowodowane charakterystyką typu *BigInteger*. Teoretycznie, typ ten nie ma żadnego limitu przechowywanych danych - w zależności od potrzeb

zajmuje potrzebną ilość pamięci w komputerze. Jedynym jego ogranicznikiem jest pamięć naszego komputera. Dodając do tego sposób przechowywania ułamków ($\frac{licznik}{mianownik}$) jesteśmy w stanie wykonywać bardzo skomplikowane równania bez utraty precyzji.

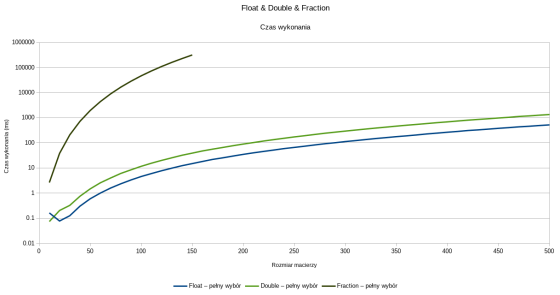
Wniosek 3 *Hipoteza 3 okazuje się być prawdziwa. Użycie własnej arytmetyki zapewnia teoretycznie największą możliwą dokładność przechowywanych danych w porównaniu do zwykłych typów zmiennoprzecinkowych. Dzięki temu, wszelkie operacje wykonywane na typie Fraction są wykonywane z dokładnością równą 0*

3 Pytania

Rysunek 3: Wykresy reprezentujące błąd bezwzględny dla różnych sposobów sumowania



(a)



(b)

3.1 Dokładność obliczeń (typ podwójnej precyzji)

Pytanie 1 *Jak zależy dokładność obliczeń (błąd) od rozmiaru macierzy dla dwóch wybranych przez Ciebie wariantów metody Gaussa gdy obliczenia prowadzone są na typie podwójnej precyzji (TD)?*

Do odpowiedzi na te pytanie, wybrany został wariant G oraz FG. W tym celu, przeanalizujmy wykres 3a. Wynika z niego, że wraz z wzrostem rozmiaru macierzy, średni błąd bezwzględny rośnie. Jest to spowodowane większą ilością obliczeń potrzebnych do osiągnięcia rozwiązania (przy dużej ilości obliczeń na liczbach wymagających większej precyzji, możliwa jest utrata precyzji). Nagłe wzrosty i spadku błędu są spowodowane wylosowaniem takiego zestawu liczb, które znacząco utrudniały rozwiązanie równania.

3.2 Zależność czasu działania algorytmu od rozmiaru macierzy oraz typu

Pytanie 2 *Jak przy wybranym przez Ciebie wariancie metody Gaussa zależy czas działania algorytmu od rozmiaru macierzy i różnych typów?*

Do odpowiedzi na te pytanie, wybrany został wariant FG dla wszystkich zdefiniowanych wcześniej typów. Na potrzeby analizy czasu wykorzystajmy wykres 3b. Przedstawia on w skali logarytmicznej średni czas wykonywania w zależności od rozmiaru tablicy. Wywnioskować z niego można, iż czas wykonywania algorytmu Gaussa wraz ze wzrostem rozmiaru tablicy rośnie w sposób wykładniczy.

4 Wydajność implementacji

Zadanie 1 *Podaj czasy rozwiązania układu równań uzyskane dla macierzy o rozmiarze* 500 *dla* 9 *testowanych wariantów.*

Z powodu ograniczeń sprzętowych, testowanie własnej arytmetyki odbyła się tylko do wielkości macierzy 150. By obliczyć kolejne wartości, wykorzystany został wzór wyznaczony za pomocą aproksymacji wielomianowej danej następująco:
G: $t = 1.012030867 \cdot 10^{-3}n^4 - 1.270979539 \cdot 10^{-1}n^3 + 7.671220016n^2 - 190.6473966n + 1419.501062$
PG: $t = 1.074913757 \cdot 10^{-3}n^4 - 1.425823098 \cdot 10^{-1}n^3 + 9.125439696n^2 - 239.6679003n + 1865.170906$
FG: $t = 1.426104474 \cdot 10^{-3}n^4 - 0.187431069n^3 + 12.05517905n^2 - 319.4153988n + 2506.335399$
Źródło: `http://www.xuru.org/rt/PR.asp`
Czasy wykonania dla rozmiaru macierzy 500 znajdują się w tabeli poniżej:

Czasy rozwiązania dla n = 500			
Typ danych	Wariant	Czas wykonania (ms)	Czas wykonania (h)
Float	G	388.13750325	N/A
	PG	387.1670819	N/A
	FG	518.17535205	N/A
Double	G	723.1210641	N/A
	PG	727.3498923	N/A
	FG	1340.82832105	N/A
Fraction	G	49188585.756762	13.663496043545
	PG	51522712.232256	14.31186450896
	FG	68559239.398499	19.04423316624972

5 Podział pracy

Dawid Bińkuś	**Oskar Bir**	**Mateusz Małecki**
Praca nad strukturą projektu.	Analiza algorytmu Gaussa oraz implementacja wariantu G	Implementacja typu własnej precyzji
Przygotowanie sprawozdania	Przygotowanie testów i ich uruchomienie	Operacje na macierzach
Implementacja algorytmu Gaussa w wariantach PG i FG	Analiza danych oraz określenie czasu pracy typu Fraction	Praca nad strukturą projektu
Implementacja generycznej klasy MyMatrix	Przygotowanie wykresów końcowych	Implementacja generycznej klasy MyMatrix