11.11.2019

Algorytmy numeryczne
Zadanie 2

Wojciech Rosiński
Nr indeksu: 240425

1. Opis zadania

Zadanie polegało na napisaniu algorytmów realizujących działanie trzech wersji metod eliminacji Gauss'a, które pozwalają na rozwiązywanie układów równań za pomocą operacji na macierzach oraz przetestowaniu ich działania na trzech typach danych. Z założenia dwa z nich miały być typami wbudowanymi: jeden o pojedynczej precyzji, drugi o podwójnej precyzji. Typem trzecim miał być samodzielnie przygotowany typ ułamek, w której pola licznik i mianownik są typami BigInteger.

2. Poprawność implementacji

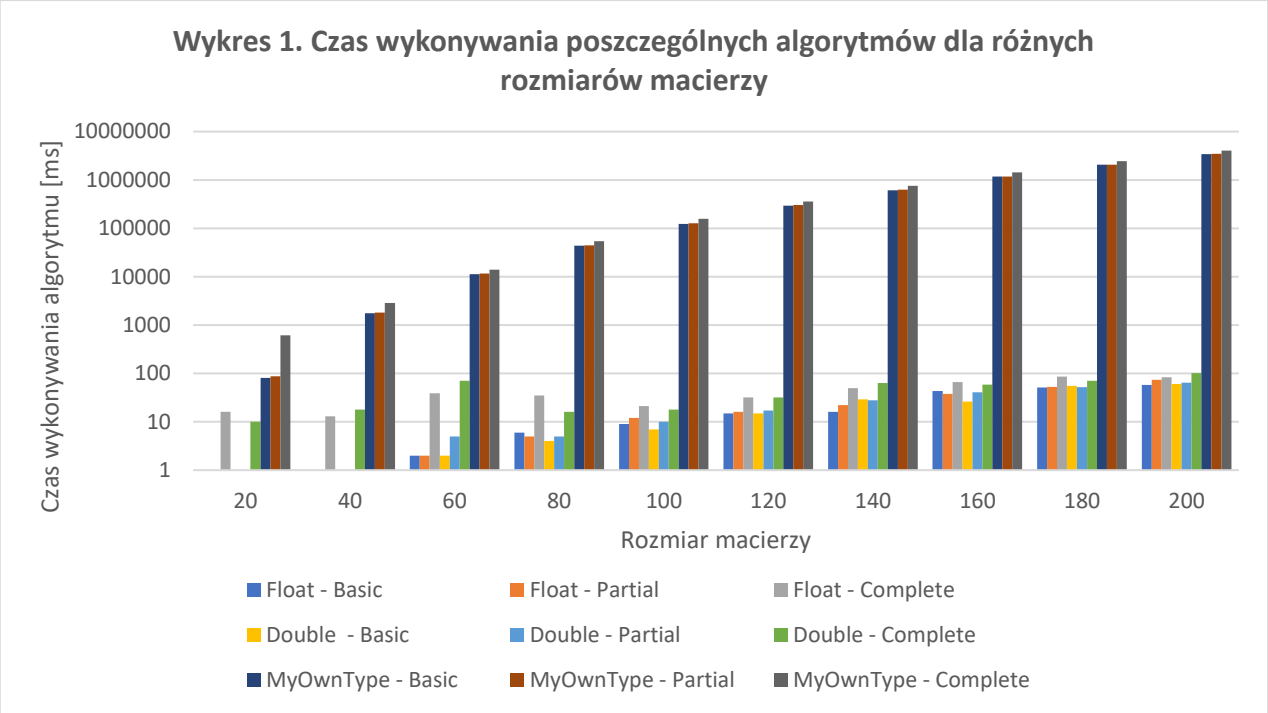
Funkcje liczące algorytmy zostały napisane w języku Java. W związku z tym, że metody eliminacji Gauss'a operują na macierzach, konieczna była implementacja klasy reprezentującej macierz z odpowiednimi metodami do wykonywania następujących operacji: mnożenia dwóch macierzy, dodawania pojedynczej wartości do jej konkretnego elementu, rozszerzenia macierzy o wektor, zamieniania wierszy lub kolumn w macierzy, odejmowania pomnożonego o wartość wiersza od innego wiersza w macierzy, odszukiwania bezwzględnie największego elementu w danej kolumnie macierzy oraz w podmacierzy. W klasie tej zaimplementowano również opisane wyżej metody eliminacji Gauss'a. Dodatkowo, w projekcie wykorzystano samodzielnie napisany generator liczb dla wszystkich trzech typów danych, wymienionych w założeniach projektu.

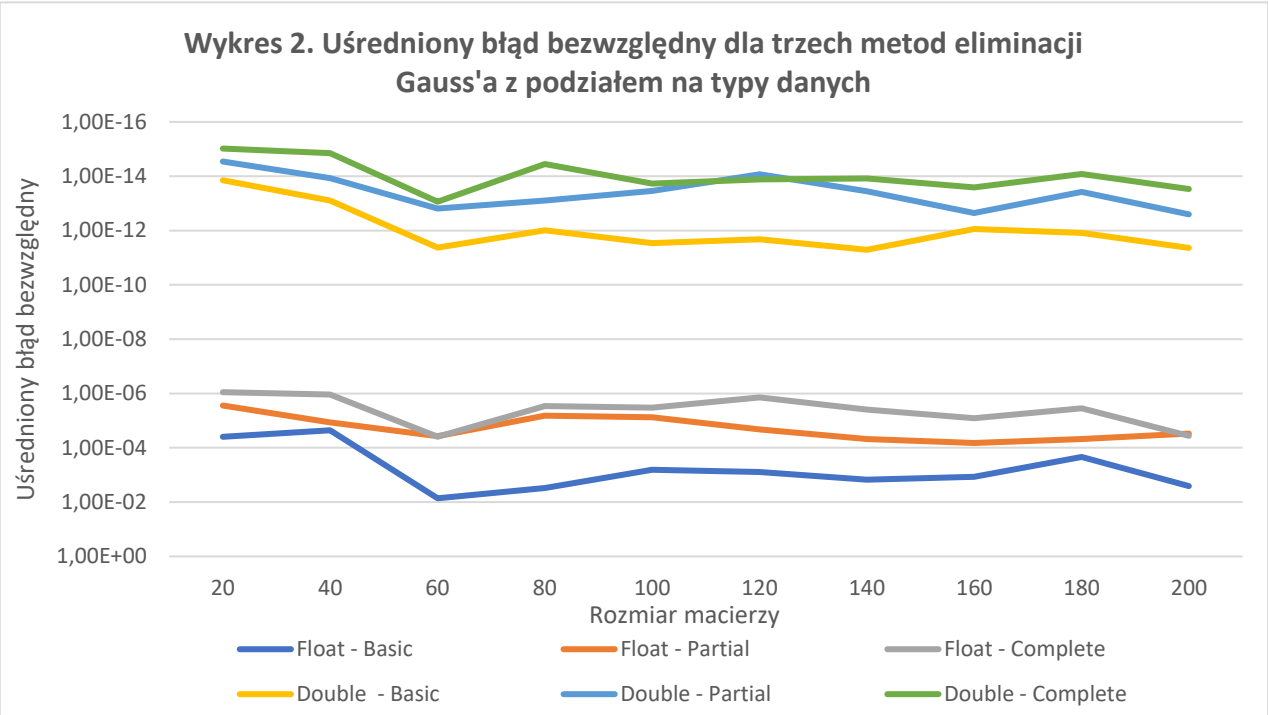
Celem sprawdzenia poprawności implementacji program generował losowe wartości dla macierzy A i wektora X. Wektor B wyliczany był poprzez przemnożenie macierzy A przez wektor X. Po rozszerzeniu macierzy A o wektor B i uruchomieniu algorytmu, program porównywał wejściowy wektor X z nowym wektorem X otrzymanym w wyniku jego działania. Jeśli wektory są identyczne, to znaczy, że algorytm poprawnie rozwiązuje problem. Dla macierzy o typie zmiennoprzecinkowym, wbudowanym występowały różnice, jednak nie wynikały one z błędnego działania algorytmu, a raczej z utraty precyzji typów przy wszelkich działaniach arytmetycznych. W przypadku macierzy o typie zaimplementowanym samodzielnie, gdzie wartości przechowywane są w formie ułamka na obiektach typu BigInteger, takie różnice nie występują, co dowodzi na to, że algorytmy są poprawnie zaimplementowane, jak i również świadczy o bezbłędnym działaniu operacji na macierzach.

*Poprawność operacji na macierzach (w tym mnożenia macierzy przez wektor) została przedstawiona w pliku Main.java.

3. Testy i wykresy

Testy wykonano na macierzach o różnych rozmiarach i z podziałem na 3 typy określone w założeniach projektu. Rozmiar testowanych macierzy zaczyna się od 20x20 i kolejno zwiększany jest o wartość 20 wierszy i 20 kolumn (do rozmiaru 200x200), co daje łącznie 10 macierzy różnych rozmiarów. Na takich macierzach uruchomiono każdy z trzech algorytmów w podziale na trzy typy danych, co daje łącznie 9 wariantów. Jak wcześniej wspomniano, program porównywał wejściowy wektor X z wektorem wynikowym algorytmu i na ich podstawie liczył uśredniony błąd bezwzględny z całego wektora różnic. Ponadto liczony był czas wykonywania się każdego z algorytmów w milisekundach.





**Uśredniony błąd bezwzględny dla własnego typu (ułamek) wynosi 0 dla wszystkich metod i przedziałów.*

4. Hipotezy

H1: Dla dowolnego ustalonego rozmiaru macierzy czas działania metody Gauss'a w kolejnych wersjach (G, PG, FG) rośnie.

Zgodnie z wykresem nr 1 czas działania metody Gauss'a w kolejnych wersjach algorytmu dla większości testowanych rozmiarów rośnie niezależnie od typu danych w macierzy. Hipoteza jest prawdziwa.

H2: Dla dowolnego ustalonego rozmiaru macierzy błąd uzyskanego wyniku metody Gauss'a w kolejnych wersjach (G, PG, FG) maleje.

Na wykresie nr 2 widać, że dla dowolnego rozmiaru macierzy uśredniony błąd wyniku w kolejnych wersjach metody Gauss'a maleje dla typów wbudowanych. Natomiast dla własnego typu (ułamek) błąd nie występuje w żadnej z metod. Hipoteza jest prawdziwa.

H3: Użycie własnej arytmetyki na ułamkach zapewnia bezbłędne wyniki niezależnie od wariantu metody Gauss'a i rozmiaru macierzy.

Dla typu własnego i wszystkich testowanych rozmiarów macierzy oraz metod błąd nie wystąpił. Hipoteza jest prawdziwa.

5. Pytania

Q1: 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)?

Dla typu Double i algorytmów PG oraz FG błąd na wykresie zmienia się o 2 - 3 rzędy wielkości. Na wykresie widać, że dla początkowych, coraz większych rozmiarów macierzy błąd dla obu metod zwiększa się. Od rozmiaru 60 następuje zmniejszenie błędu dla obu metod, a później znowu wzrost. Takie zmiany występują naprzemiennie. Nie da się tutaj zauważyć konkretnej zależności pomiędzy rozmiarem macierzy a błędem obliczeń. Błąd jest mniejszy dla algorytmu FG.

Q2: Jak przy wybranym przez Ciebie wariancie metody Gauss'a zależy czas działania algorytmu od rozmiaru macierzy i różnych typów?

Dla algorytmu Gauss'a w wariancie PG czas działania algorytmu zwiększa się wraz ze wzrostem rozmiaru macierzy dla wszystkich typów.

E1: Wydajność implementacji

Czas działania algorytmów dla macierzy rozmiaru 500 [ms]								
Gauss			PartialGauss			FullGaus		
Float	Double	MyOwnType	Float	Double	MyOwnType	Float	Double	MyOwnType
692	**1621**	**>3438384**	**757**	**1632**	**>3455990**	**1630**	**2545**	**>4069739**

Konfiguracja sprzętowa:
Procesor: Intel Core i5 4200M
Pamięć RAM: 12 GB