

Algorytmy numeryczne

Zadanie 2

Aleksander Kosma Tomasz Adamczyk
grupa 1 tester-programista

8 Listopad 2017

1 Dodawanie i Mnożenie

Opracowanie prezentuje obliczenia na macierzach, przy użyciu języka java i wsparciu biblioteki eigen wykorzystanego w C++. Pierwsza część opisuje czas działania i błędy operacji dodawania i mnożenia. Obliczenia wygenerowane w javie porównywane były z wynikami wygenerowanymi z eigena. Testy zostały przeprowadzone na 3 wzorach:

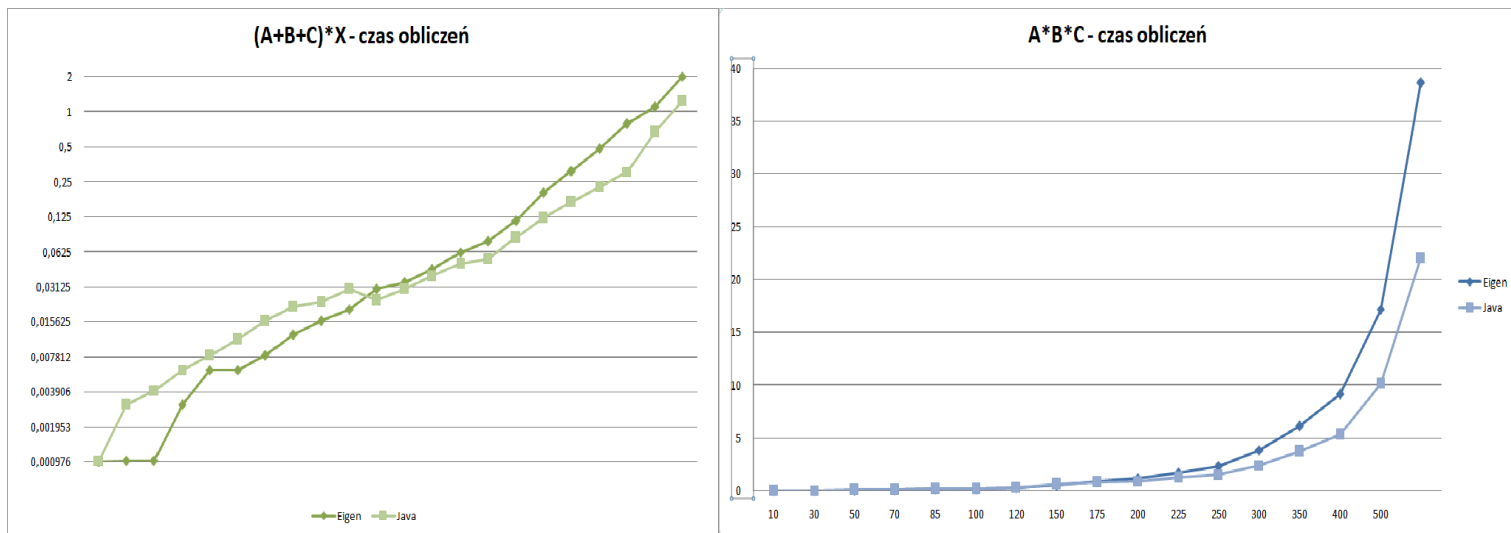
$$A * X$$

$$(A + B + C) * X$$

$$A * B * C$$

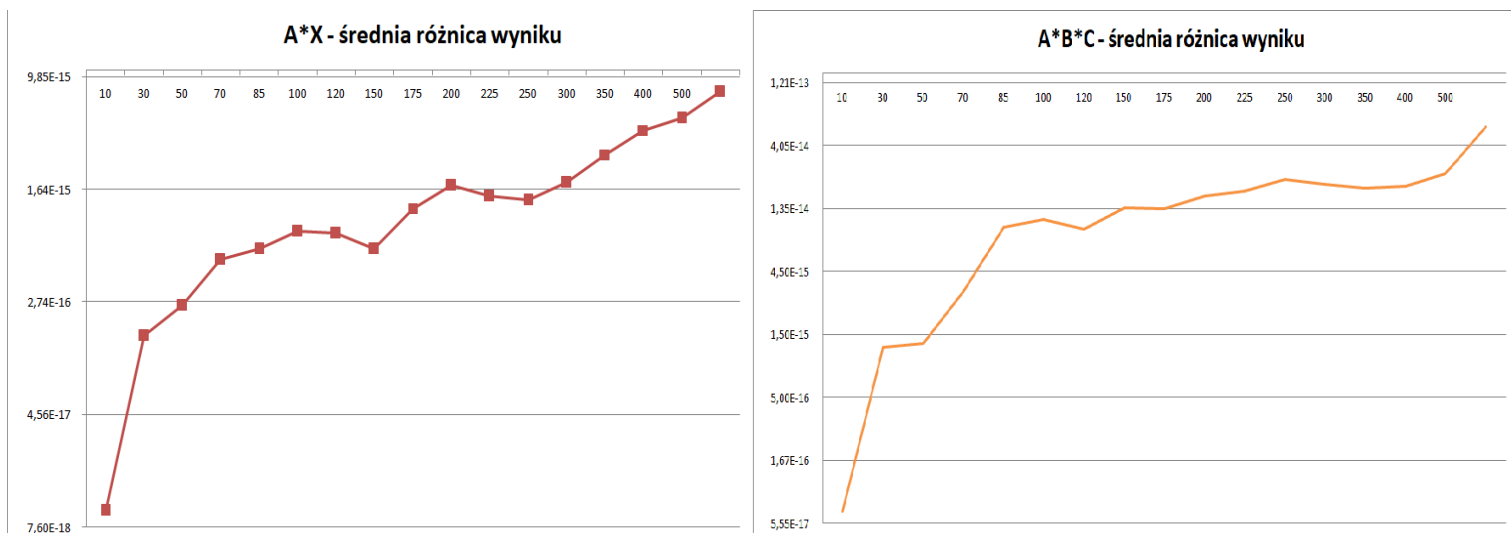
Gdzie litery A,B,C to macierze, X to wektor. Testy zostały przeprowadzone na 3 typach zmiennych. Double, Float i na własnym typie, który przechowuje liczbę w postaci ułamka. Typ ten ma bezstratną precyzję.

Pierwszy wykres prezentuje czas potrzebny do obliczenia podanego wzoru. Oś pionowa reprezentuje czas w sekundach, oś pozioma rozmiar macierzy. Pod koniec widać czasową przewagę własnych obliczeń nad Eigen. Podane wykresy utworzone na zmiennej double.



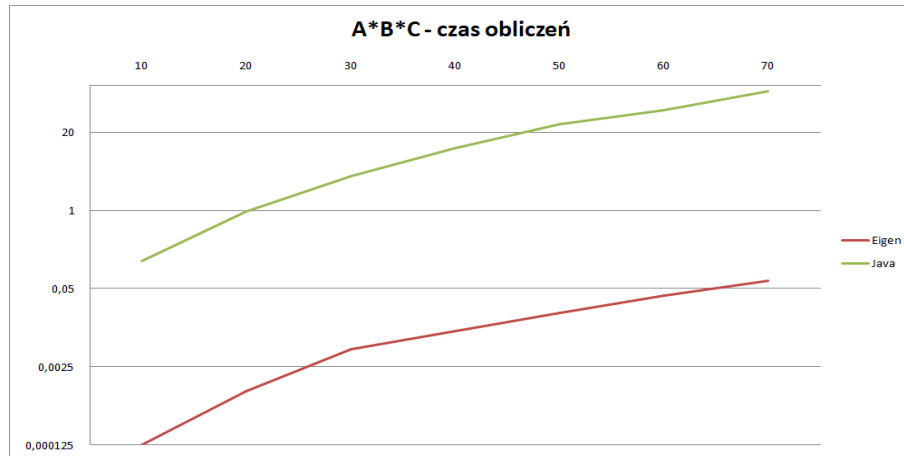
Drugi wykres prezentuje dla innego wzoru czas obliczeń. W pewnym momencie obliczenia javy stają się szybsze i tendencja ta się utrzymuje.

Trzeci wykres prezentuje te same dane, tylko w kontekście precyzji. Widać że wraz ze wzrostem wielkości macierzy, błędy przybierają na wartości.



Czwarty wykres prezentuje podobne wnioski jak w przypadku trzeciego. Tym większa macierz tym większy błąd ostateczny. W przypadku mnożenia wynika to z faktu obliczania kolejnych miejsc po przecinku, które nie miejszczą już się w zakresie danej liczby zmiennoprzecinkowej.

Obliczenia na Float wykazywały podobną tendencję, tyle że w mniejszej skali i wolniejszym tempie, więc pominę prezentację dla tej zmiennoprzecinkowej. W przypadku własnej precyzji, w zakresie porównania do Double obliczonego przez Eigen wyniki pokrywały się idealnie. Główna różnica to dłuższy czas oczekiwania na wynik.



W przypadku jakiegokolwiek sumowania nie odnotowano jakiegokolwiek utraty wartości względem Eigen. Wiąże się to prawdopodobnie z faktem działania operacji dodawania. W momencie dodania dwóch liczb ich suma będzie potrzebować w najgorszym przypadku jednego miejsca więcej do zapisu. W przypadku mnożenia, ilość potrzebnych bitów rośnie zdecydowanie szybciej. W konsekwencji traci się precyzję która już się nie mieści w zakresie danej zmiennoprzecinkowej.

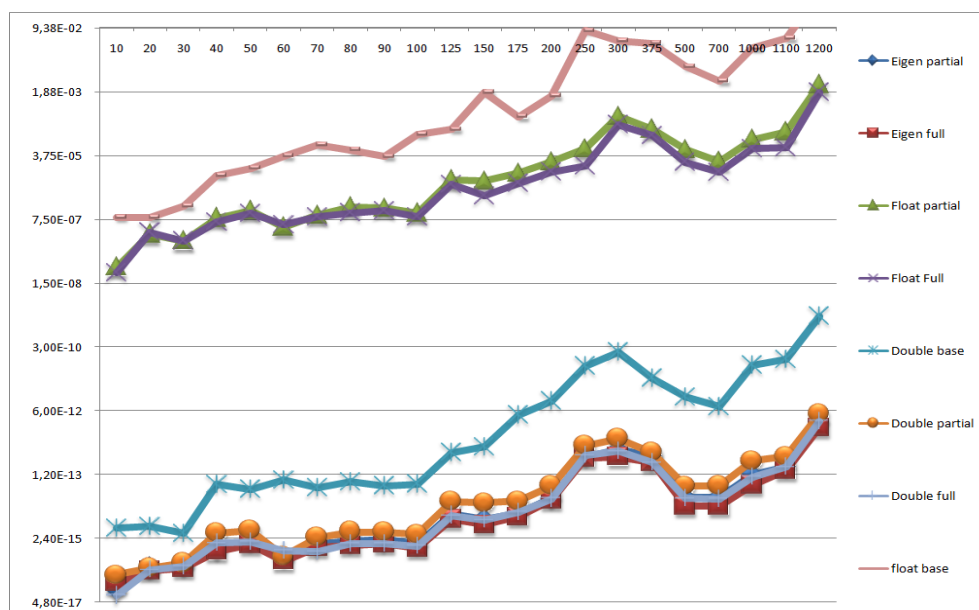
2 Rozwiązywanie układów równań liniowych

Druga część opisuje rozwiązywanie układu równań liniowych macierzy A i wektora B. W javie wykorzystane zostały 3 wersje metody eliminacji Gaussa:

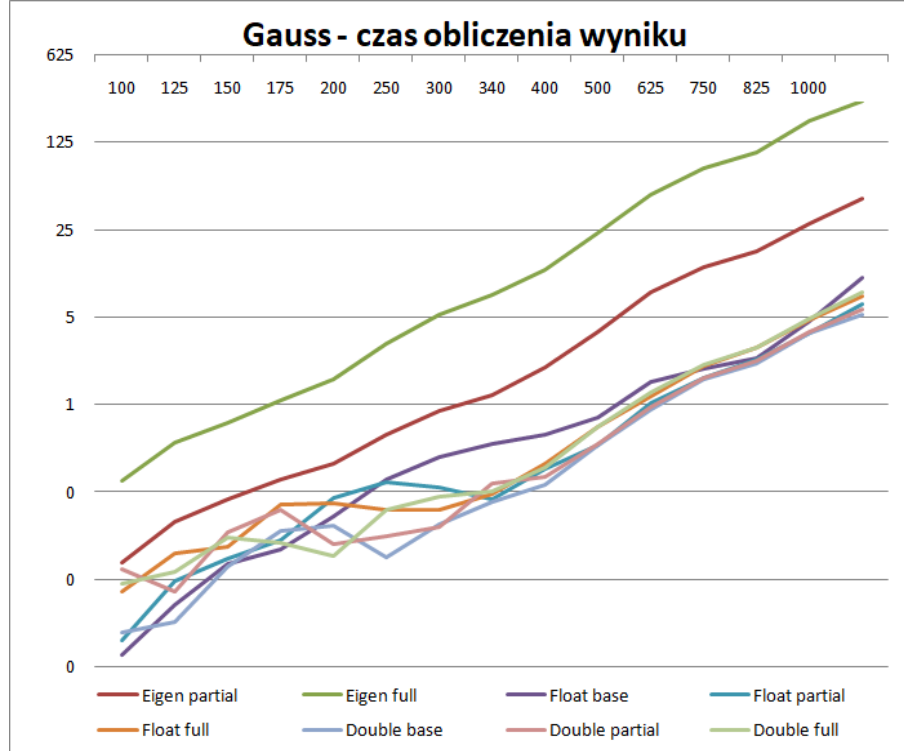
- bez wyboru elementu podstawowego
- z częściowym wyborem elementu podstawowego
- z pełnym wyborem elementu podstawowego

Odnosnikiem jako poprawny wynik było podstawienie pod układ obliczonych niewiadomych. W eigen skorzystano z wyliczenia poprzez częściowy i pełny wybór.

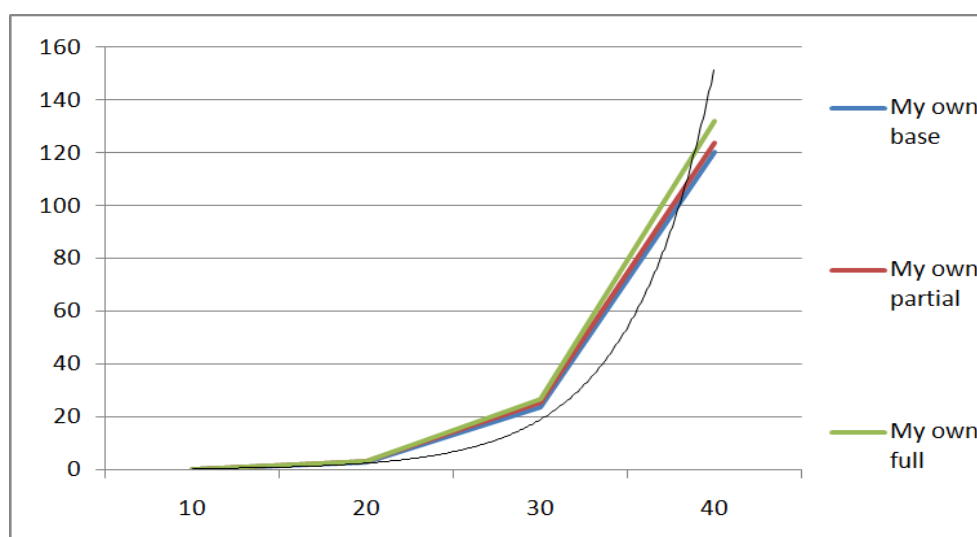
Pierwszy wykres przedstawia różnice obliczonego układu dla wyliczonych niewiadomych i oryginalnego wektora. Tym niższa wartość w danym punkcie tym precyzyjniejszy wynik. Najlepszy okazuje się wynik wyliczony przez Eigen. Niewiele gorzej wypada Double dla pełnego wyboru elementu podstawowego.



Drugi wykres przedstawia czas potrzebny do osiągnięcia rozwiązania. Chaotyczny rozkład i przenikanie przez siebie dolnych linii wynika z sytuacji, gdzie dana macierz może mieć odpowiedni układ nie wymagający jakiegokolwiek sortowania. Stąd tak zbliżone wyniki.



Ostatni wykres przedstawia czas potrzebny do wyliczenia wyników dla idealnej precyzji. W istocie wynik wychodzi za każdym razem idealny. Wraz ze wzrostem macierzy, czas oczekiwania na wynik bardzo szybko rośnie co widać na dorysowanej na szaro funkcji.



Większość obliczeń została uśredniona z 5-15 próbek na jeden rozmiar macierzy. Zazwyczaj tyle wystarczyło by zauważyć tendencje wzrostowe.

Podział obowiązków	
Aleksander Kosma	Tomasz Adamczyk
-Klasa MyOwnPrecision	-Implementacja Gaussa do Klasy MyMatrix
-Szkielet klasy i zapewnienie generyczności	-Klasa do generowania losowych macierzy
-Implementacja metod dodawania i mnożenia	-Generowanie wyniku poprzez bibliotekę Eigen w C++
-Testy i generowanie wyników	-Zautomatyzowanie generowania wyników
-Obróbka wyników, wykresy i opracowanie	-