MOWNIT – Laboratorium 3
Metoda Eliminacji Gaussa

Mikołaj Wróblewski

1. Zgodnie z poleceniem ćwiczenie rozpocząłem od zaimplementowania najbardziej podstawowej metody eliminacji Gaussa. Następnie używając napisanych przeze mnie funkcji testujących na losowo generowanych macierzach skonfrontowałem bazową metodę eliminacji Gaussa z funkcją biblioteczną, wszystkie wyniki zostały porównane z tolerancją do epsilon = 1e-10. Poniżej zamieszczam zrzuty ekranu kilka możliwych zachowań podczas testów najbardziej podstawowej metody:

```
[mikolaj@mikolaj-pc ~]$ python /home/mikolaj/Pulpit/MOWNIT2/lab3/gauss.py
Attempted to divide by zero, aborting...
```

```
[mikolaj@mikolaj-pc ~]$ python /home/mikolaj/Pulpit/MOWNIT2/lab3/gauss.py
Tests passed
```

```
[mikolaj@mikolaj-pc ~]$ python /home/mikolaj/Pulpit/MOWNIT2/lab3/gauss.py
The matrix test failed
```

Wszystkie te możliwe stany wyjściowe są dosyć proste do opisania:

Tests passed – mieliśmy szczęście, wygenerowane macierze nie były źle uwarunkowane dla naszego algorytmu.

Attempted to divide by zero – jest to typowy problem dla tej implementacji. W podstawowej implementacji tego algorytmu w niektórych sytuacjach możemy dojść do sytuacji, w której będziemy próbowali dzielić przez zero.

The matrix test failed – kolejna typowa sytuacja, przy porównywaniu wyniku z wynikiem funkcji bibliotecznej uzyskaliśmy błąd z uwagi na błędne wyniki implementacji bazowej.

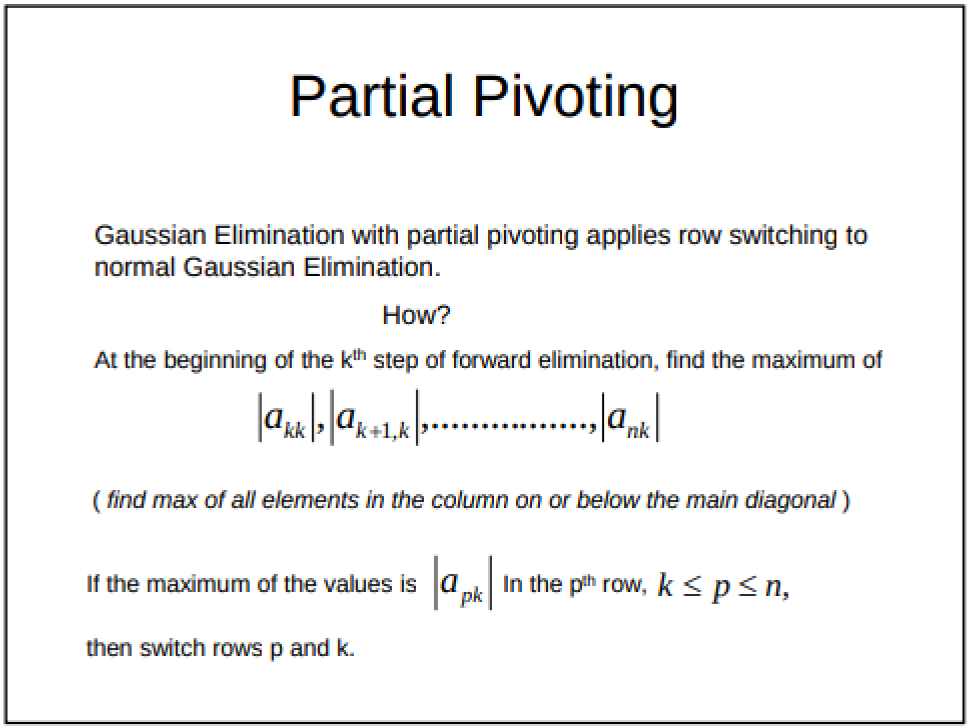
Wnioski z użycia najbardziej podstawowej metody są oczywiste, nasza metoda ma dwie zasadnicze wady:

- może wystąpić w niej dzielenie przez zero
- jest podatna na błędy zaokrąglania

Należy zatem naszą metodę poprawić. Jest na to wiele podejść, możemy:

- skorzystać z piwotowania
- użyć dekompozycji
- użyć faktoryzacji blokowej

2. W dalszej części ćwiczenia zmodyfikowałem bazowy algorytm w taki sposób, by korzystał z częściowego piwotowania. Metoda dodaje do bazowej metody zamiany wierszy w sposób następujący:



Metoda ta uodparnia algorytm na błąd związany z dzieleniem przez zero oraz redukuje możliwe błędy związane z zaokrąglaniem.
Przy każdym z wykonanych testów na losowych macierzach zwracany jest tylko jeden komunikat:

```
[mikolaj@mikolaj-pc ~]$ python /home/mikolaj/Pulpit/MOWNIT2/lab3/gauss.py
Tests passed with usage of <function guass_w_swaps at 0x7fd8e0793d90>
```

Aczkolwiek nadal możliwe jest „złamanie” algorytmu .
Dla macierzy:

$$\left[\begin{array}{cc|c} 2 & 2c & 2c \\ 1 & 1 & 2 \end{array}\right]$$

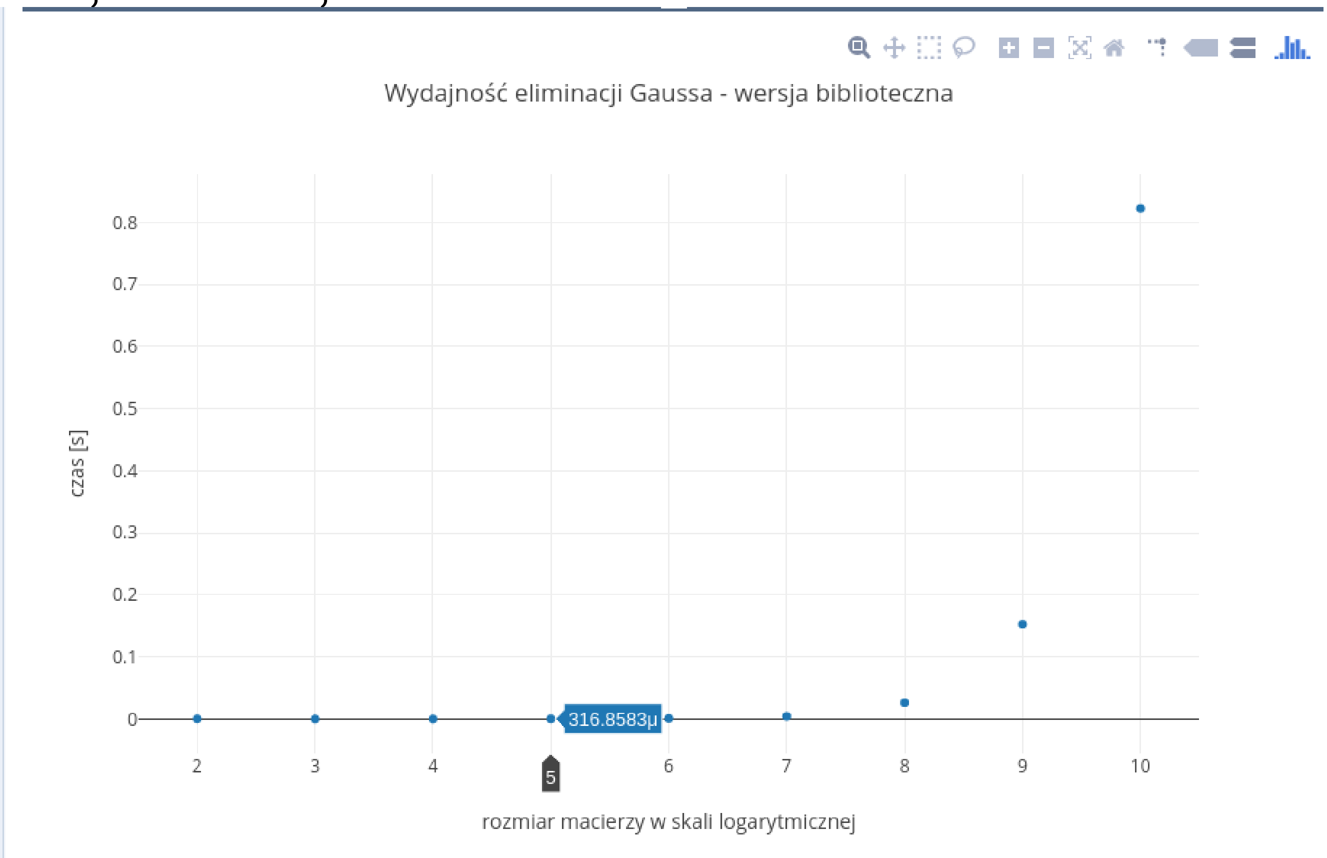
Gdzie c przyjmuje duże wartości numeryczne uzyskamy błędne wyniki, jedynym rozwiązaniem danego układu jest: x = 1, y = 1. Dla c = 5e20:

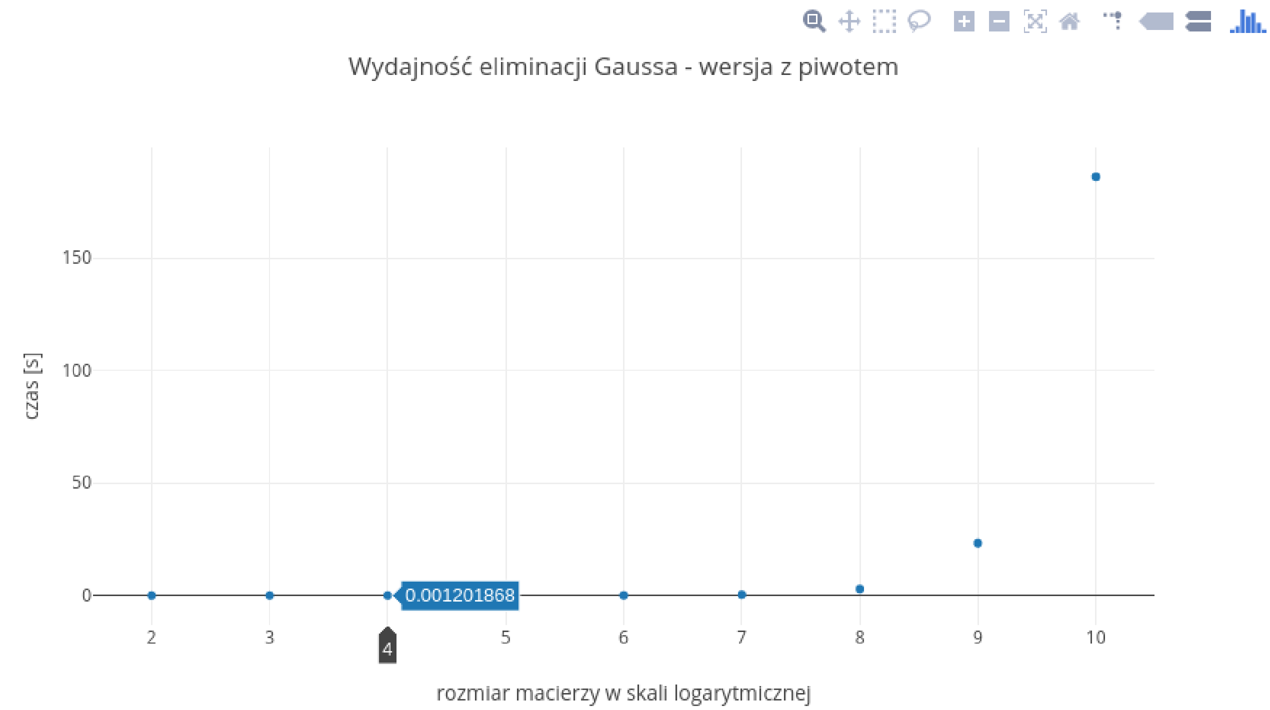
```
Gauss with pivoting:  [0.0, 1.0]
Gauss with library:  [0. 1.]
```

Widzimy zatem, że zarówno algorytm z piwotowaniem, jak i algorytm z funkcji bibliotecznej jest obarczony błędem.

3. Testy wydajnościowe

Na koniec wykonałem testy wydajnościowe rozwiązania z piwotowaniem jak i funkcji bibliotecznej.





Wykresy dobrze przedstawiają jak ma się złożoność algorytmów. Generalnie algorytm Gaussa ma złożoność O(n^3). Jest to widoczne na zamieszczonych wykresach. Na wykresach skorzystałem ze skali logarytmicznej dla lepszej czytelności wykresu.