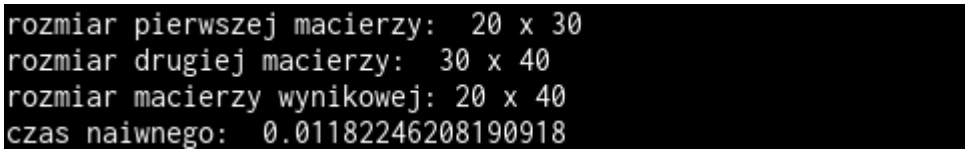
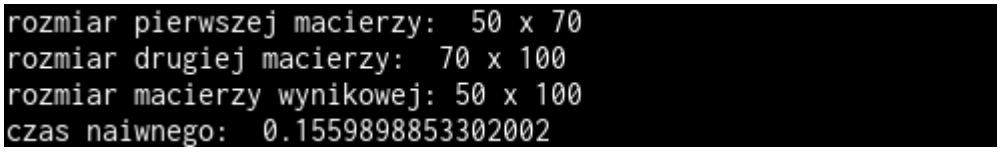
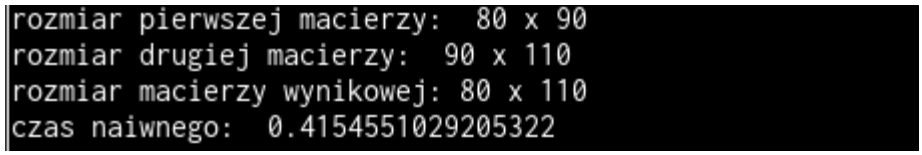
MOWNIT – Laboratorium 2
Mnożenie Macierzy

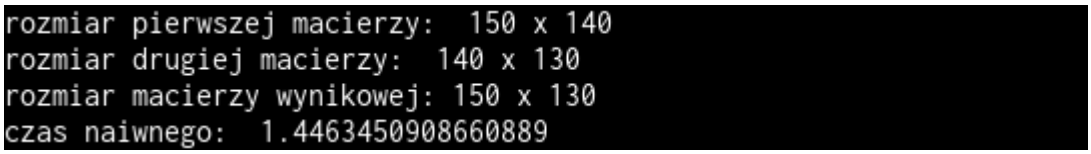
Mikołaj Wróblewski

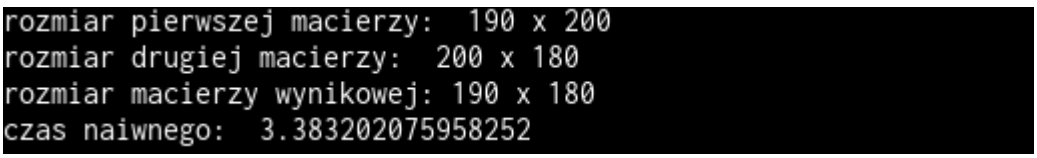
1. Narzędziem z jakiego korzystałem podczas wykonywania ćwiczenia jest język Python. Ćwiczenie rozpocząłem od napisania naiwnej funkcji mnożenia macierzy w pętlach, po przetestowaniu poprawności działania funkcji zabrałem się za wykonywanie pomiarów czasowych (w sekundach), w zależności od rozmiarów mnożonych macierzy, które wypełniłem losowymi liczbami całkowitymi.
Pomiary zamieszczam w formie zrzutów ekranu wraz z rozmiarami macierzy.









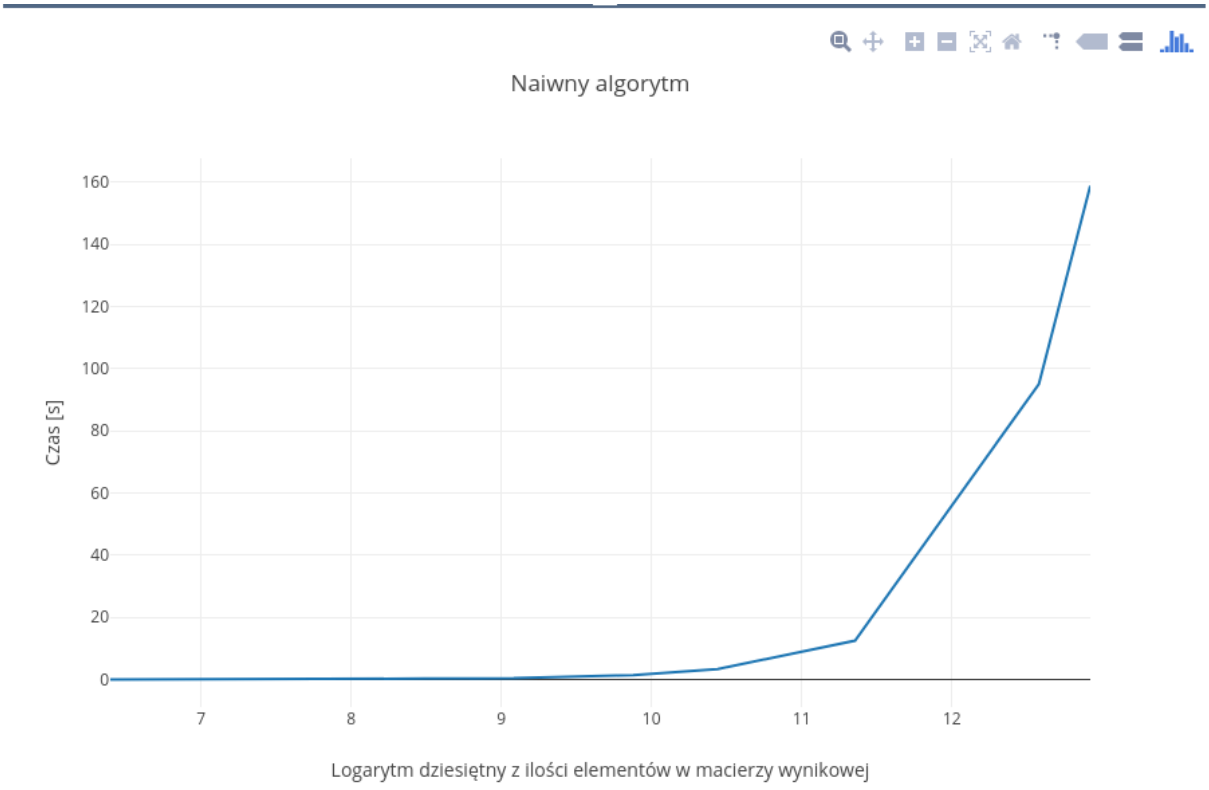


```
rozmiar pierwszej macierzy:  280 x 290
rozmiar drugiej macierzy:  290 x 305
rozmiar macierzy wynikowej: 280 x 305
czas naiwnego:  12.525611400604248
```

```
rozmiar pierwszej macierzy:  480 x 590
rozmiar drugiej macierzy:  590 x 605
rozmiar macierzy wynikowej: 480 x 605
czas naiwnego:  95.08216118812561
```

```
rozmiar pierwszej macierzy:  580 x 690
rozmiar drugiej macierzy:  690 x 705
rozmiar macierzy wynikowej: 580 x 705
czas naiwnego:  158.86759233474731
```

Z pomiarów widać, iż zwiększenie rozmiarów macierzy, o ok. 600 w każdym z wymiarów skutkuje znaczym wydłużeniem wykonywanego mnożenia. W pierwszym pomiarze czas rzędu setnych sekund, w ostatnim czas rzędu setek sekund. Podczas wykonywania pomiarów spróbowałem także znacznie większych rozmiarów, jednakowoż musiałem zatrzymywać procesy – z uwagi na czas jaki zabierały.
Poniżej zamieszczam wyniki opracowane w formie wykresu.



Widzimy iż wykres przypomina wykres funkcji y = x^3.

2. Następie zmodyfikowałem algorytm tak, by wykorzystać możliwości narzędzia z jakiego skorzystałem. W czystym pythonie macierze są reprezentowane jako lista list, zatem za oczywiste uznałem wykorzystanie list comprehensions, czyli de facto „cukru syntaktycznego” jaki oferuje język Python. Dzięki temu udało mi się uzyskać bardziej wydajne mnożenie macierzy aniżeli naiwny sposób. Następnie zamieniłem lekko dany sposób, by działał na odwrotnej reprezentacji. Poniżej zamieszczam wyniki pomiarów czasowych wraz z rozmiarami macierzy.

```
rozmiar pierwszej macierzy:  20 x 30
rozmiar drugiej macierzy:  30 x 40
rozmiar macierzy wynikowej: 20 x 40
czas zoptymalizowanego:  0.007485628128051758
czas zoptymalizowanego - odwrotna reprezentacja:  0.0073833465576171875
```

```
rozmiar pierwszej macierzy:  50 x 70
rozmiar drugiej macierzy:  70 x 100
rozmiar macierzy wynikowej: 50 x 100
czas zoptymalizowanego:  0.09038257598876953
czas zoptymalizowanego - odwrotna reprezentacja:  0.09582686424255371
```

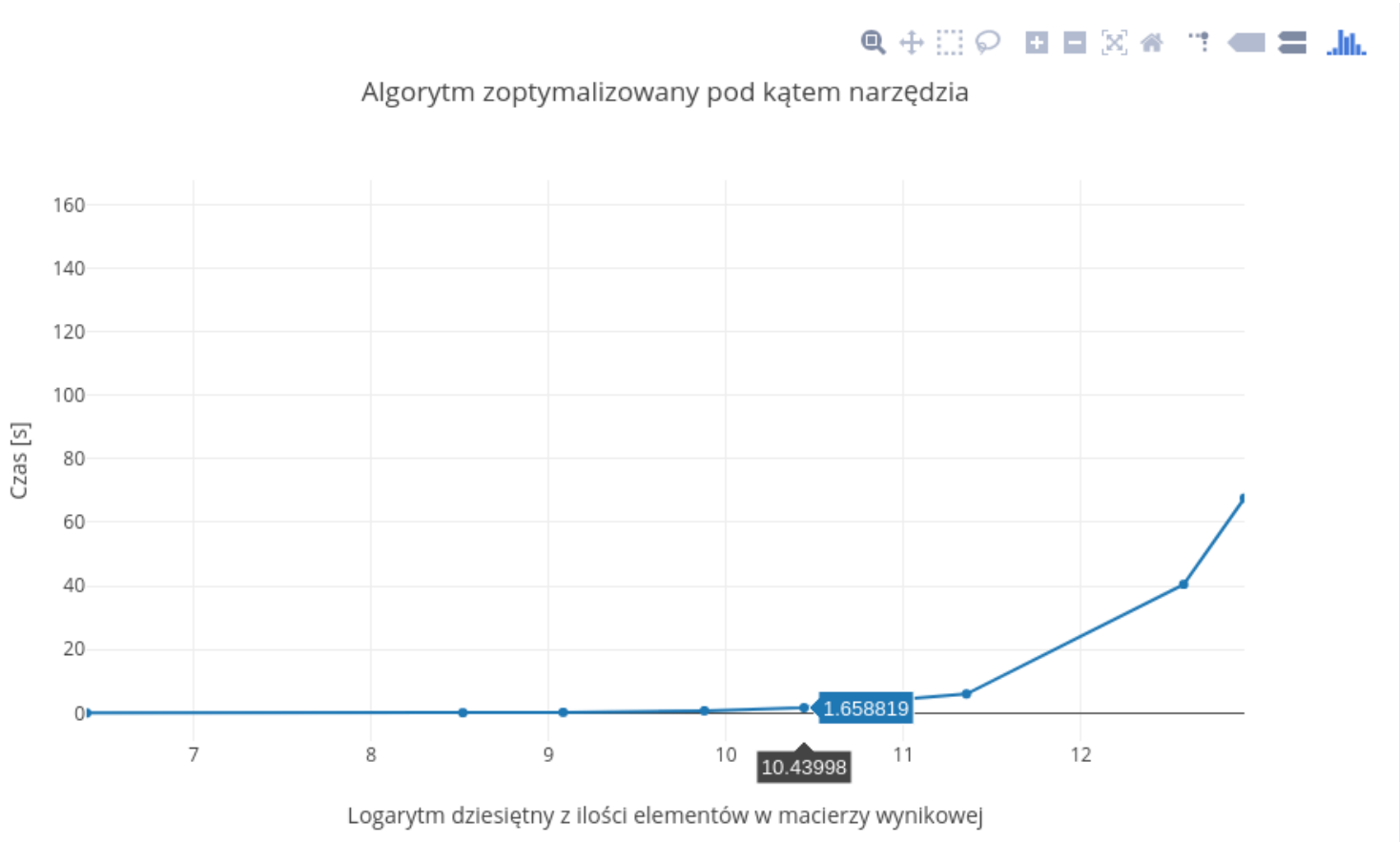
```
rozmiar pierwszej macierzy:  80 x 90
rozmiar drugiej macierzy:  90 x 110
rozmiar macierzy wynikowej: 80 x 110
czas zoptymalizowanego:  0.20204567909240723
czas zoptymalizowanego - odwrotna reprezentacja:  0.2024974822998047
```

```
rozmiar pierwszej macierzy:  150 x 140
rozmiar drugiej macierzy:  140 x 130
rozmiar macierzy wynikowej: 150 x 130
czas zoptymalizowanego:  0.6850364208221436
czas zoptymalizowanego - odwrotna reprezentacja:  0.6763091087341309
rozmiar pierwszej macierzy:  190 x 200
rozmiar drugiej macierzy:  200 x 180
rozmiar macierzy wynikowej: 190 x 180
czas zoptymalizowanego:  1.6588194370269775
czas zoptymalizowanego - odwrotna reprezentacja:  1.6735925674438477
rozmiar pierwszej macierzy:  280 x 290
rozmiar drugiej macierzy:  290 x 305
rozmiar macierzy wynikowej: 280 x 305
czas zoptymalizowanego:  5.736278533935547
czas zoptymalizowanego - odwrotna reprezentacja:  5.80521559715271
```

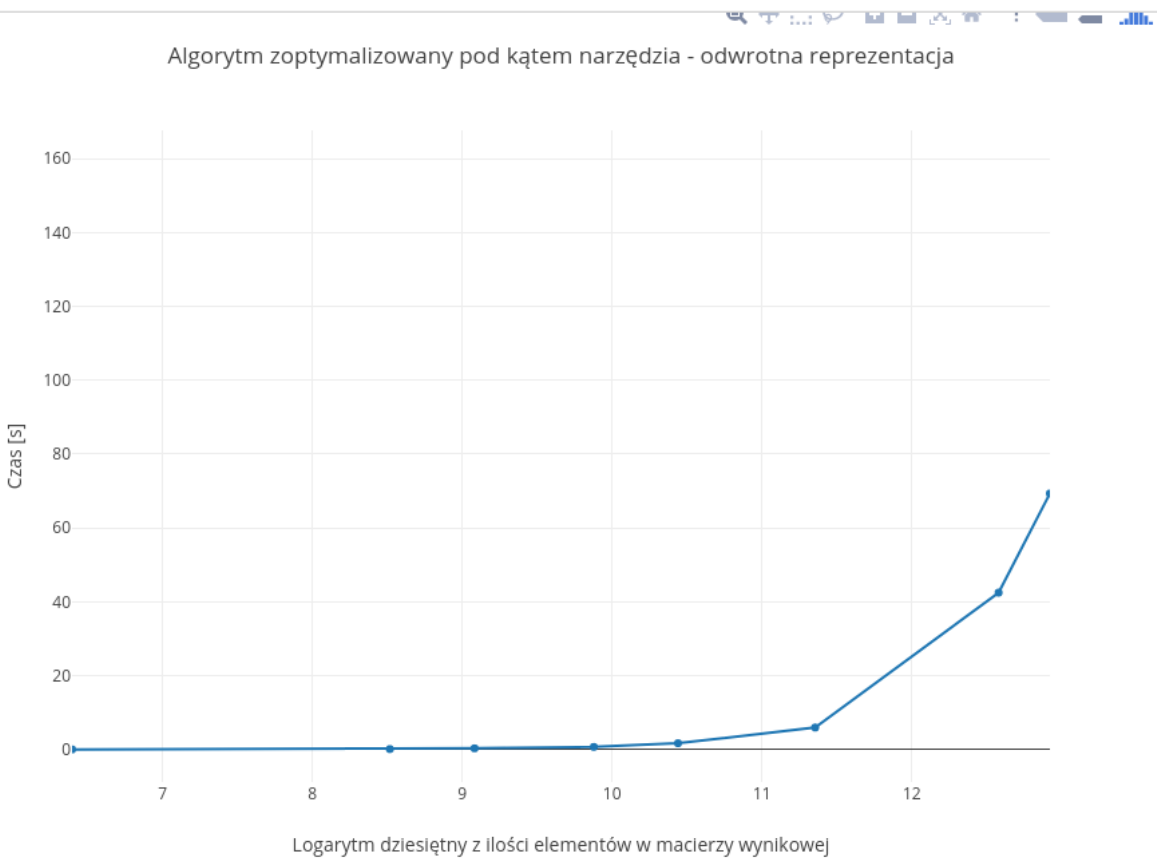
```
rozmiar pierwszej macierzy:  480 x 590
rozmiar drugiej macierzy:  590 x 605
rozmiar macierzy wynikowej: 480 x 605
czas zoptymalizowanego:  43.17749261856079
czas zoptymalizowanego - odwrotna reprezentacja:  41.75796937942505
```

```
rozmiar pierwszej macierzy:  580 x 690
rozmiar drugiej macierzy:  690 x 705
rozmiar macierzy wynikowej: 580 x 705
czas zoptymalizowanego:  67.6450366973877
czas zoptymalizowanego - odwrotna reprezentacja:  68.33976101875305
```

W przypadku obu reprezentacji trudno o jakiekolwiek dalekoidące wnioski, gdyż wyniki są do siebie zbliżone. Jednakowoż widzimy sporą poprawę względem algorytmu naiwnego.
Poniżej wyniki przedstawione w formie wykresu.



Poniżej wykres dla odwrotnej reprezentacji.



Wciąż zauważamy znaczne podobieństwo do wykresu poprzedniego.

3. Użycie biblioteki do obliczeń numerycznych – numpy, poskutkowało znacznym przyspieszeniem wszystkich obliczeń. Nie umieszczam tutaj zrzutów ekranu dla poprzednich rozmiarów – obrazuję je jedynie w formie wykresu - postanowiłem użyć większych macierzy, takich dla których musiałem „zabijać” procesy przy użyciu algorytmu naiwnego, je również umieszczam na jednym wykresie.

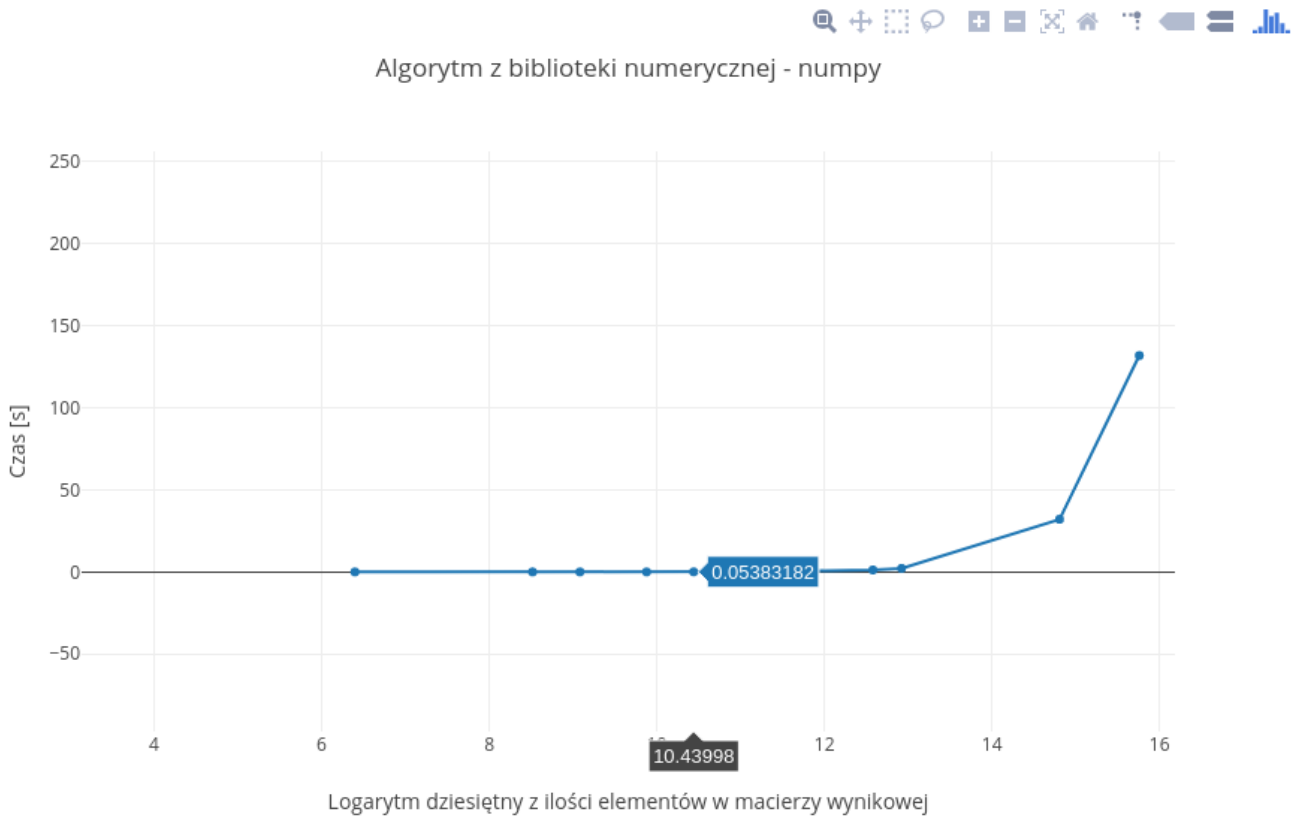
```
rozmiar pierwszej macierzy:  1580 x 1690
rozmiar drugiej macierzy:  1690 x 1705
rozmiar macierzy wynikowej: 1580 x 1705
Czas z wykorzystaniem biblioteki numpy:  32.05855178833008
```

```
rozmiar pierwszej macierzy:  2580 x 2690
rozmiar drugiej macierzy:  2690 x 2705
rozmiar macierzy wynikowej: 2580 x 2705
Czas z wykorzystaniem biblioteki numpy:  131.69645500183105
```

Przy algorytmie naiwnym czasy rzędu ~150 sekund były obecne już przy macierzach o rozmiarach mniejszych o 2000, widać zatem znaczną wyższość funkcji bibliotecznych. Dodatkowo dla porównania zamieszczam zrzut ekranu przy rozmiarach macierzy, dla których algorytm zoptymalizowany pod kątem używanego narzędzia generował czasy rzędu 40 sekund:

```
rozmiar pierwszej macierzy:  480 x 590
rozmiar drugiej macierzy:  590 x 605
rozmiar macierzy wynikowej: 480 x 605
Czas z wykorzystaniem biblioteki numpy:  1.1884160041809082
```

Poniżej zamieszczam wykres dla biblioteki numpy:



Wnioski nasuwają się same. Ze wszystkich wykresów możemy zaobserwować podobieństwo do początku wykresu y = x^3. Wynika to z tego, iż złożoność algorytmu mnożenia macierzy w najgorszym przypadku to O(n^3). Przy aplikacjach dla których czas działania jest bardzo ważny (na przykład w systemach czasu

rzeczywistego) użycie dwóch wcześniejszych metod można śmiało uznać za kategoryczny błąd.Gdybym jednak sam został postawiony przed zadaniem napisania własnej implementacji wydajnego mnożenia macierzy, najprawdopodobniej zrównolegliłbym kod. Wydaje mi się iż takie rozwiązanie byłoby najbardziej optymalne.