**Raport 2**

Rekurencyjne wyznaczanie odwrotności, LU i wyznacznika

*Michał Kobiera, Maciej Pięta*

# Opis ćwiczenia

Celem ćwiczenia było zaimplementowanie trzech algorytmów:

* Rekurencyjnego odwracania macierzy
* Rekurencyjnej faktoryzacji LU macierzy
* Rekurencyjnego obliczania wyznacznika

Oraz następnie zmierzenie czasu działania i ilości operacji zmiennoprzecinkowych dla macierzy o rozmiarach wypełnionych liczbami z przedziału (0,1) i oszacowanie złożoności obliczeniowej zaimplementowanych algorytmów

# Opis algorytmów

1. Rekurencyjne odwracanie macierzy

Inverse(A)

Jeśli macierz A ma tylko 1 element:

Zwróć 1/A

W przeciwnym wypadku:

Podziel A na 4 macierze A11, A12, A21, A22:

Oblicz rekurencyjnie

Oblicz

Oblicz rekurencyjnie

Oblicz:

Zwróć

1. Rekurencyjna faktoryzacja LU

LU(A):

Jeśli macierz A ma tylko 1 element:

Zwróć L, U

W przeciwnym wypadku:

Podziel A na 4 macierze A11, A12, A21, A22:

Oblicz rekurencyjnie

Oblicz:

,

Zwróć

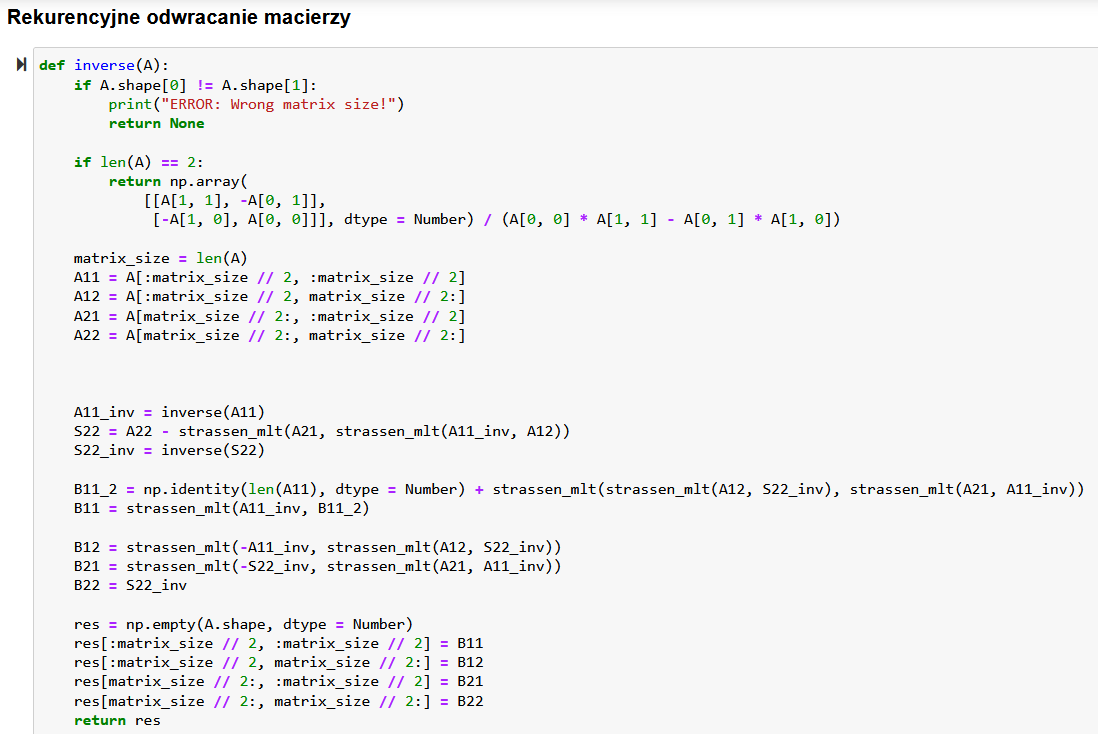
1. Rekurencyjne obliczanie wyznacznika

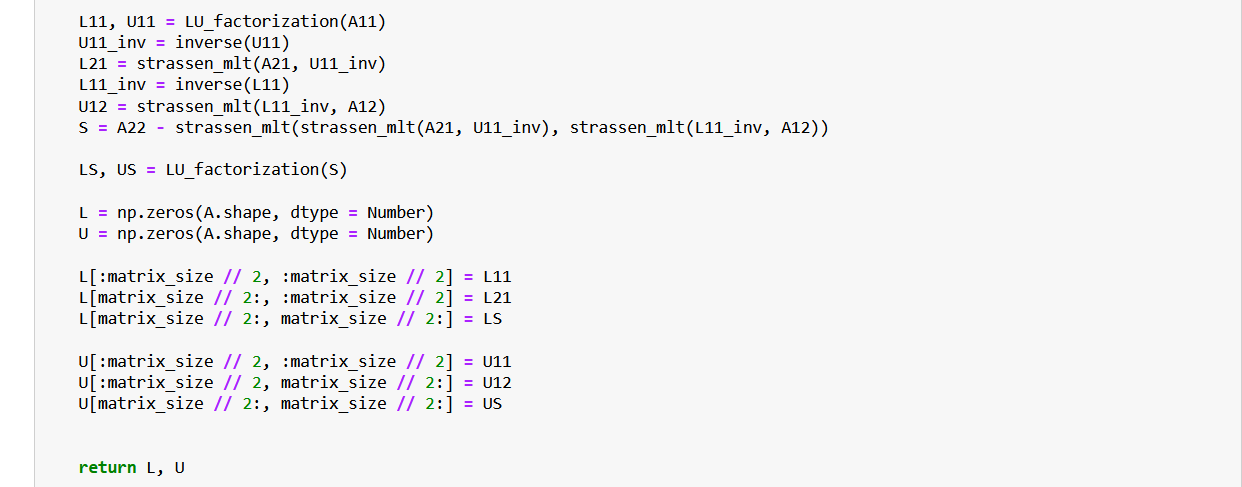
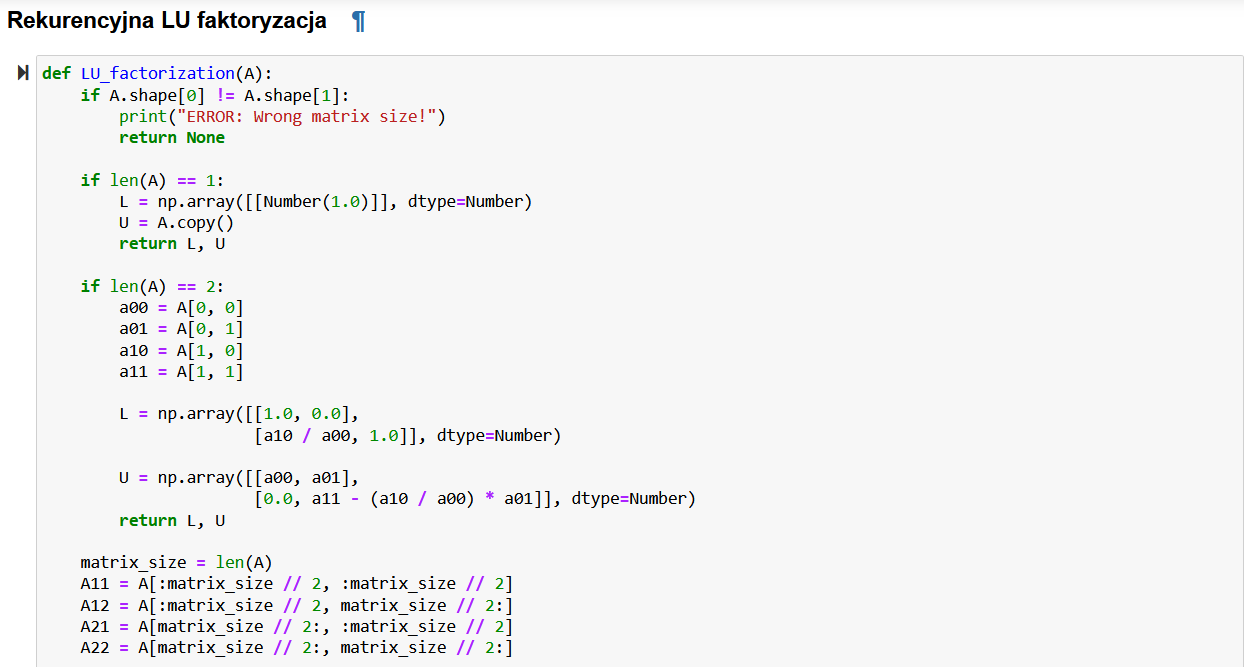
Det(A):

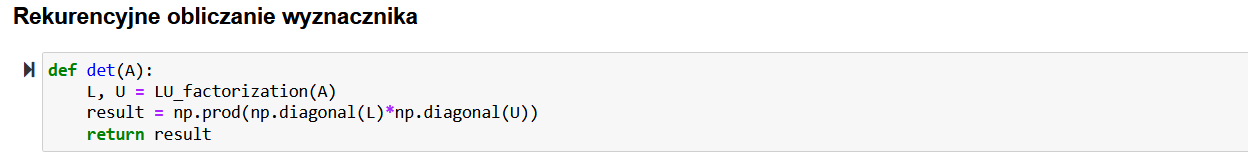
Oblicz

Zwróć

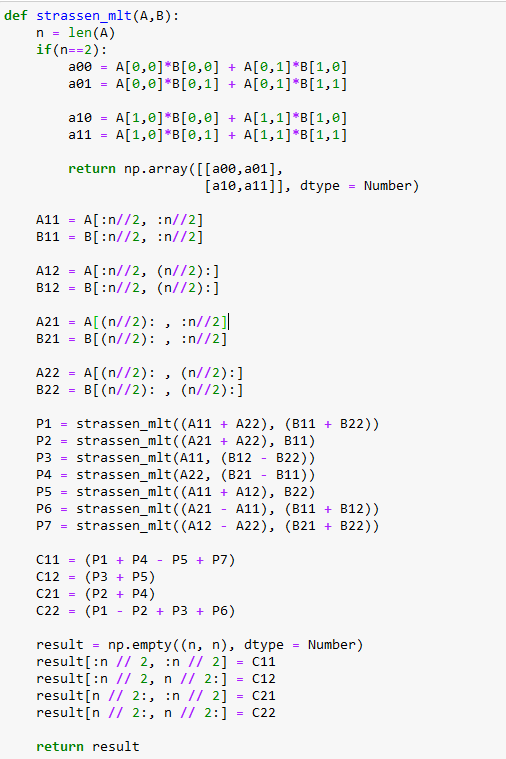
# Implementacje algorytmów







Do mnożenia macierzy wykorzystaliśmy zaimplementowaną przez nas funkcję *strassen\_mlt(A, B):*

**

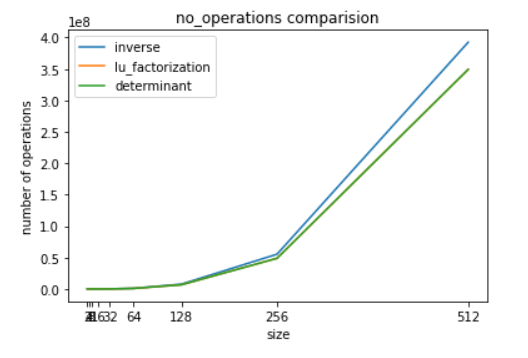
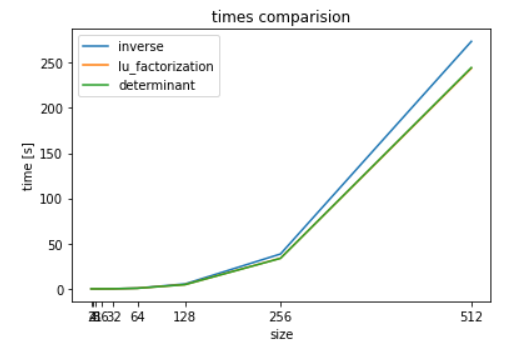
# Wykresy czasu działania i ilości operacji zmiennoprzecinkowych

W celu zliczania ilości operacji zmiennoprzecinkowych zaimplementowaliśmy własny typ Number, który dziedziczy po domyślnym typie float i nadpisuje operatory +, -, \*, / by po każdym wykonaniu danej operacji zwiększać zmienną statyczną *operation\_counter* :

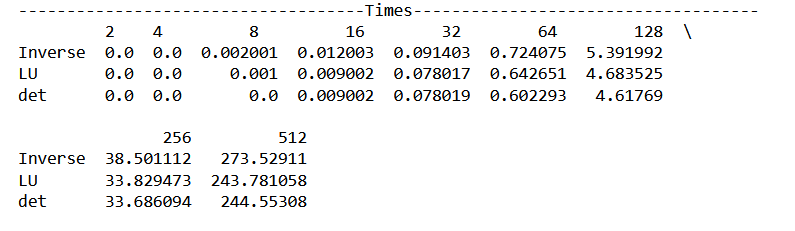
A screenshot of a computer program

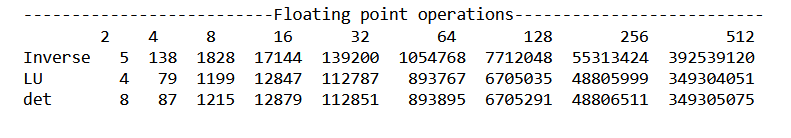
Description automatically generated

Porównanie czasów wykonania poszczególnych algorytmów:



Dane z wykresów:





# Szacowanie złożoności

Do oszacowania złożoności wykorzystaliśmy dane z wykresów oraz funkcję *curve\_fit* z biblioteki*scipy* która aproksymuje parametry funkcji korzystając z metody najmniejszych kwadratów.

Jako model przyjęliśmy funkcję typu i otrzymaliśmy następujące wyniki:

1. Odwrotność macierzy:

Bazując na otrzymanej ilości operacji zmiennoprzecinkowych otrzymaliśmy:

Na podstawie czasu:

1. Faktoryzacja LU:

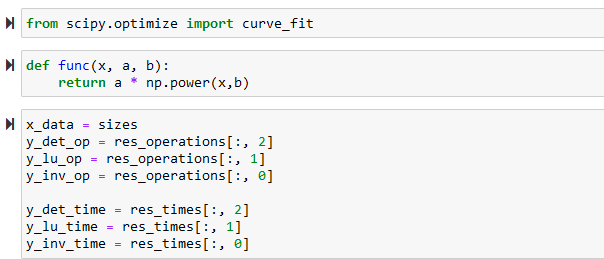
Na podstawie ilości operacji zmiennoprzecinkowych:

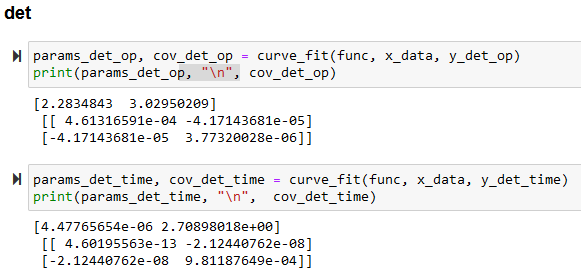
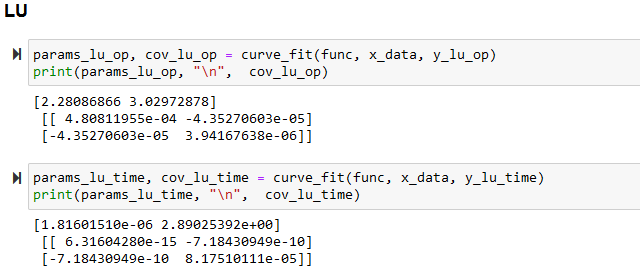
Na podstawie czasu:

1. Wyznacznik macierzy:

Na podstawie ilości operacji zmiennoprzecinkowych:

Na podstawie czasu:

A screenshot of a computer program

Description automatically generated

# Sprawdzenie poprawności zaimplementowanych algorytmów

Do sprawdzenia poprawności implementacji funkcji *inverse(A)* oraz *det(A)* wykorzystaliśmy istniejące funkcję biblioteki *numpy: numpy.linalg.inv(A), numpy.linalg.det(A)*

Poprawność implementacji LU\_factorization(A) była sprawdzana poprzez sprawdzenie bliskości wyniku mnożenia obliczonych macierzy *L, U* z wejściową macierzą *A*

Odwrotność:

A computer screen shot of a computer code

Description automatically generated

Faktoryzacja LU:

A white rectangular object with black text

Description automatically generated

Wyznacznik:

A white rectangular object with black text

Description automatically generated