

Instrukcja do programu:

1. Zainstalować python: <https://www.python.org/downloads/>
2. Zainstalować biblioteki NumPy i Scipy: `pip install numpy` i `pip install scipy`
3. Wpisać do terminalu: `python "FileName"`

Wstęp:

Miałem podane macierz `A` rozmiarem `N x N`

$$A = \begin{pmatrix} 1.2 & \frac{0.1}{1} & \frac{0.4}{1^2} & & & & & & \\ 0.2 & 1.2 & \frac{0.1}{2} & \frac{0.4}{2^2} & & & & & \\ & 0.2 & 1.2 & \frac{0.1}{3} & \frac{0.4}{3^2} & & & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \\ & & & & 0.2 & 1.2 & \frac{0.1}{N-2} & \frac{0.4}{(N-2)^2} & \\ & & & & & 0.2 & 1.2 & \frac{0.1}{N-1} & \\ & & & & & & 0.2 & 1.2 & \end{pmatrix}$$

oraz $x = (1, 2, \dots, N)^T$. Rozmiar `N = 100`.

Musiałem wyznaczyć równanie $y = A^{-1}x$ dla macierzy `A`, również obliczyć wyznacznik macierzy `A`. Zrobić to trzeba było właściwą metodą i wykorzystać strukturę macierzy.

Aby nie liczyć macierzy odwrotnej A^{-1} przekształciłem równanie do postaci: $Ay = x$

Dla sprawdzania wyników używałem bibliotek `numpy` i `scipy`.

Prezentacja wyników:

$\det(A) = 78240161.00959387$

$y =$

```
[3.28713349e-02  1.33962280e+00  2.06648030e+00  2.82554361e+00
 3.55757172e+00  4.28449287e+00  5.00721018e+00  5.72766400e+00
 6.44661558e+00  7.16455440e+00  7.88177388e+00  8.59846587e+00
 9.31475980e+00  1.00307462e+01  1.07464903e+01  1.14620401e+01
 1.21774318e+01  1.28926932e+01  1.36078460e+01  1.43229071e+01
 1.50378905e+01  1.57528071e+01  1.64676661e+01  1.71824750e+01
 1.78972401e+01  1.86119666e+01  1.93266590e+01  2.00413212e+01
 2.07559563e+01  2.14705671e+01  2.21851562e+01  2.28997256e+01
 2.36142771e+01  2.43288125e+01  2.50433330e+01  2.57578401e+01
 2.64723348e+01  2.71868182e+01  2.79012910e+01  2.86157543e+01
 2.93302086e+01  3.00446546e+01  3.07590929e+01  3.14735241e+01
 3.21879487e+01  3.29023671e+01  3.36167796e+01  3.43311868e+01
 3.50455888e+01  3.57599861e+01  3.64743789e+01  3.71887674e+01
 3.79031520e+01  3.86175328e+01  3.93319101e+01  4.00462841e+01
 4.07606549e+01  4.14750227e+01  4.21893876e+01  4.29037498e+01
 4.36181095e+01  4.43324668e+01  4.50468217e+01  4.57611744e+01
 4.64755250e+01  4.71898736e+01  4.79042203e+01  4.86185652e+01
 4.93329083e+01  5.00472497e+01  5.07615896e+01  5.14759279e+01
 5.21902647e+01  5.29046002e+01  5.36189343e+01  5.43332671e+01
 5.50475986e+01  5.57619290e+01  5.64762582e+01  5.71905863e+01
 5.79049133e+01  5.86192393e+01  5.93335643e+01  6.00478884e+01
 6.07622116e+01  6.14765338e+01  6.21908553e+01  6.29051759e+01
 6.36194957e+01  6.43338147e+01  6.50481330e+01  6.57624506e+01
 6.64767674e+01  6.71910836e+01  6.79053992e+01  6.86197141e+01
 6.93340283e+01  7.00483379e+01  7.07650589e+01  7.15391569e+01]
```

Przedyskutowanie wyników:

Dla tych wyników używałem faktoryzację `LU`. Żeby nie przechowywać macierz `A` rozmiarem `N x N` stworzyłem 4 listy, w których przychowywałem elementy z przekątnych. Dalej musiałem zrobić rozkład `LU`. Elementy macierz liczyłem przez wzory:

$$L_{i+1\ i} = A_{i+1\ i} / U_{i\ i}$$

$$U_{i\ i} = A_{i\ i} - L_{i\ i-1} * U_{i-1\ i}$$

$$U_{i\ i+1} = A_{i\ i+1} - L_{i\ i-1} * U_{i-1\ i+1}$$

$$U_{i\ i+2} = A_{i\ i+2}$$

Przez to, że mam taką macierz i przechowuję elementy w taki sposób złożoność obliczeniowa wyszła $O(N)$. Dla wyliczenia wektora `y` używałem metod `forward substitution` i `back substitution`, które miały złożoność $O(2N)$. W końcu cała faktoryzacja i wyliczenia miały złożoność $O(3N)$ czyli $O(N)$.

Wyznacznik macierzy liczyłem w taki sposób: $\det(A) = \det(LU) = \det(L) * \det(U) = \prod_i U_{ii}$

Wszystkie wyniki porównałem z wynikami uzyskanymi przy użyciu bibliotek algebraicznych.