Sprawozdanie NUM3

1.Wprowadzenie:

W tym zadaniu chcemy pokazać efektywny sposób wyznaczania y=A^(-1)x

$$\mathbf{A} = \begin{pmatrix} 1.2 & \frac{0.1}{1} & \frac{0.15}{1^2} & & & & & & \\ 0.2 & 1.2 & \frac{0.1}{2} & \frac{0.15}{2^2} & & & & & \\ & 0.2 & 1.2 & \frac{0.1}{3} & \frac{0.15}{3^2} & & & & \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ & & & & & 0.2 & 1.2 & \frac{0.1}{N-2} & \frac{0.15}{(N-2)^2} \\ & & & & & & 0.2 & 1.2 & \frac{0.1}{N-1} \\ & & & & & & & 0.2 & 1.2 \end{pmatrix}$$

To macierz A, możemy zauważyć, że jest macierzą rzadką, zapełnione są cztery diagonale, reszta jest wypełniona zerami

2.Uruchomienie programu:

- uruchomienie programu:

Aby wyświetlić wyniki: make run1

Aby wyświetlić wykres zależności czasu od n: make run2

3.Przechowywanie macierzy A:

Przechowywanie naszej macierzy wstęgowej będzie oszczędniejsze dla tablicy 4xN. Ma ona wymiary 4 rzędy każdy rządy to jedna nasza „diagonala" jak przedstawiono na rysunku poniżej. Będzie to lepsze rozwiązanie, ponieważ na zapisanie całej macierzy A zmarnowalibyśmy dużo pamięci. Przykładowo Dla tablicy dwuwymiarowej o wymiarach 1000x1000 i przechowującej liczby zmiennoprzecinkowe typu float, możemy obliczyć jej rozmiar w pamięci w ten sam sposób, co wcześniej: Rozmiar jednego elementu = 4 bajty (float)

Liczba elementów 1000*1000 = 1,000,000

Całkowity rozmiar tablicy = Rozmiar jednego elementu * Liczba elementów = 4 bajty * 1,000,000 = 4,000,000 bajtów

Aby przeliczyć to na megabajty:

4,000,000 bajtów / 1,048,576 = 3.8147 MB

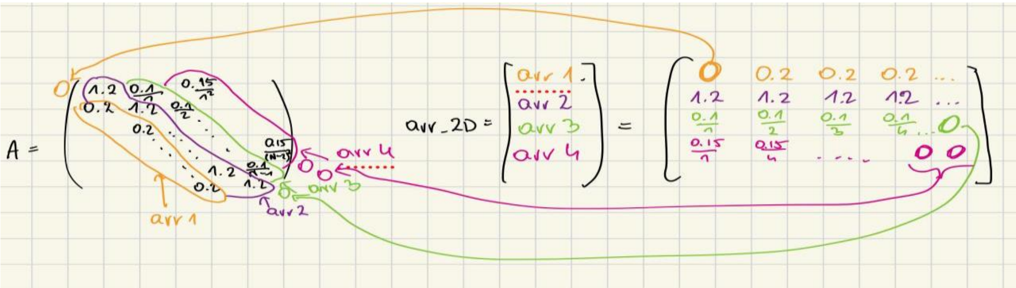
Dla tablicy dwuwymiarowej o wymiarach 4x1000:

Liczba elementów 4*1000 = 4,000

Całkowity rozmiar tablicy 4 bajty * 4,000 = 16,000 bajtów

16,000 bajtów / 1,048,576 = 0.01526 MB

Rozmiar tablicy dwuwymiarowej o wymiarach 4x1000 i przechowującej liczby zmiennoprzecinkowe typu float wynosi około 0.01526 megabajtów. Jest to bardzo niewielka tablica w porównaniu do 1000x1000.



3.Wyznaczanie macierzy odwrotnej:

Wyznaczanie macierzy odwrotnej ma zazwyczaj duża złożoność obliczeniową. Aby uniknąć tego w naszym zadaniu postarajmy się przekształcić y=(A^(-1))x. Mnożąc obustronnie przez A otrzymujemy Ay=A(A^(-1))x, wiedząc, że A*A^-1=1 otrzymujemy Ay=x. Zmniejsza to znacznie złożoność obliczeniową całego zadania.

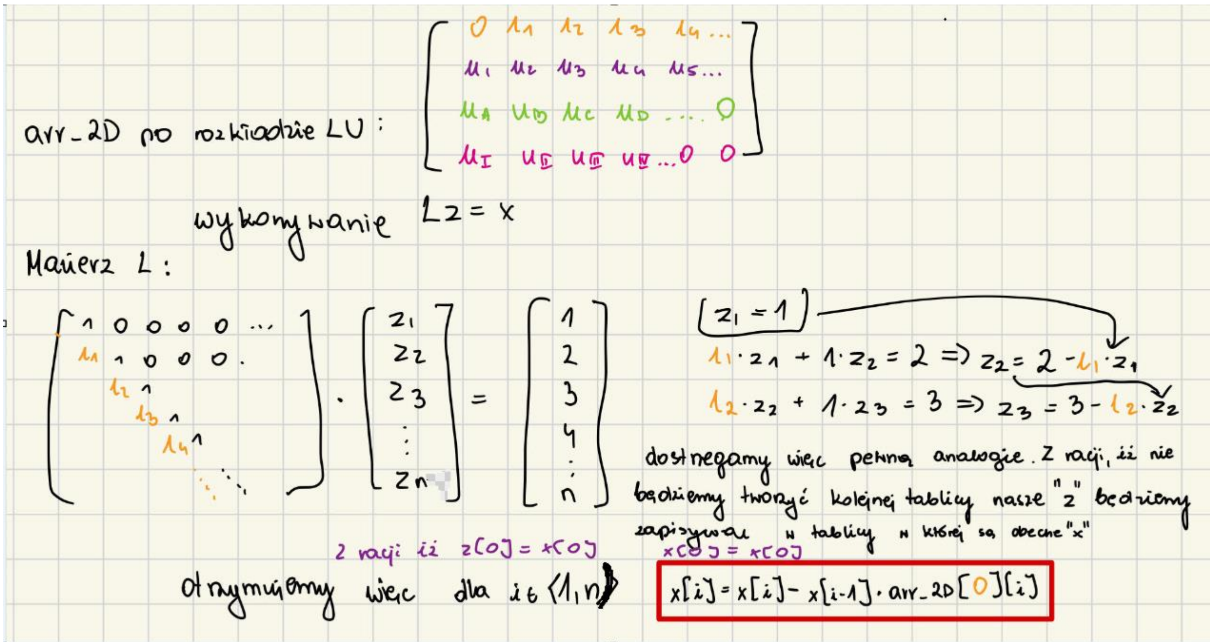
4.Rozkład macierzy A na macierze L i U:

Faktoryzacji LU macierzy, która ma na celu rozkład macierzy na dwie macierze: macierz dolną L i macierz górną U. Jest to często używane do rozwiązywania układów równań liniowych i obliczania wyznaczników macierzy. W naszym przypadku będzie się to wiązało ze złożonością O(n), związane jest to z wyglądem naszej macierzy, dla innych macierzy złożoność może wynieść O(n^3).

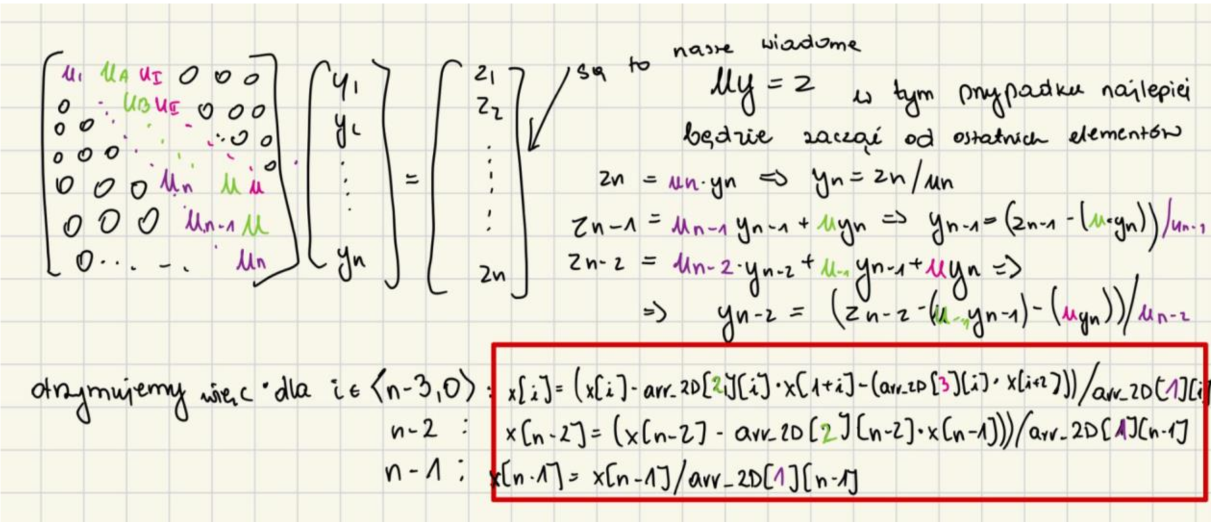
W naszym zadaniu musimy rozbić tą faktoryzacje na zakresy od 1 do n – 2, n-2 oraz n-1. Razem tworzymy pełny rozkład macierzy LU. Korzystać będę z algorytmu Dolittle'a. Przez budowę macierzy A, która jest wypełniona wieloma zerami usunie nam wiele obliczeń.

Z racji, że A=LU mamy do rozwiązania teraz dwa równania Lz=x, Uy=z

Lz=x:



Uy=z



5.Obliczanie wyznacznika:

Z własności wyznaczników wiemy, iż wyznacznik iloczynu macierzy = iloczynowi ich wyznaczników:

$$\det(A) = \det(L \times U) = (\det L) \cdot (\det U)$$

Z rozwinięcia Laplace'a wynika, że wyznacznik macierzy trójkątnej jest równy iloczynowi wyrazów znajdujących się na diagonali. L i U to nasze macierze trójkątne. Jak można zauważyć poniżej macierz trójkątna L nie ważne ile wynosi n ma na swojej diagonali same jedynki (przykład dla n=5). Co daje nam, że det(L)=1

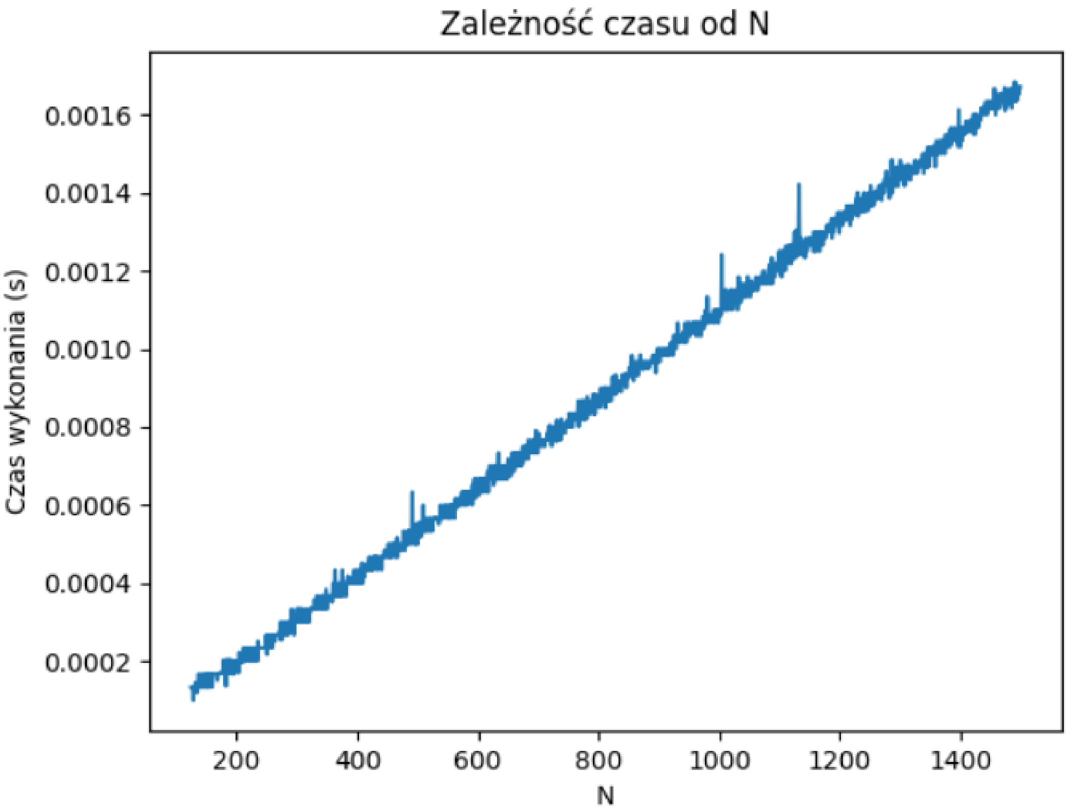
```
1.0 0.0 0.0 0.0 0.0
0 1.0 0.0 0.0 0.0
0 0.16666666666666669 1.0 0.0 0.0
0 0.16666666666666669 0.16901408450704225 1.0 0.0
0 0.16666666666666669 0.16901408450704225 0.1672555948174323 1.0
```

Wyznacznik naszej macierzy możemy więc zapisać w postaci det(L)*det(U)=

1*det(U)= det(U). Będzie więc to iloczyn elementów na diagonali macierzy U.

6. Zależność czasu:

Jak można zauważyć na wykresie wywołanym przez program zależność czasu od N rośnie liniowo. Dla każdej wartości N zostaje wykonane 30 prób pomiarowych, co pomaga w uśrednieniu wyników i zminimalizowania wpływów losowych na czas. Większe odchylenia widoczne na wykresie są spowodowane pracą komputera.



7. Wyniki:

Wyznacznik macierzy A wynosi: 6141973498.857843

Wektor rozwiązania:

y=[0.448700827728733, 1.4132732869357947, 2.1348778535462736, 2.8690132654396248, 3.5914885705595205, 4.311604959915503, 5.029827173723323, 5.747011462584994, 6.463503693914558, 7.179525964548697, 7.8952125968955915, 8.610651859797315, 9.325903619364162, 10.041009954626537, 10.756001271894783, 11.470900080737994, 12.185723394049369, 12.900484305313817, 13.615193053266005, 14.329857758065131, 15.044484941235352, 15.759079899902147, 16.473646980766127, 17.18818978377055, 17.902711315621058, 18.617214106977666, 19.331700302956037, 20.046171733762908, 20.76062997036838, 21.47507636878333, 22.189512105570603, 22.903938206548368, 23.6183555701598, 24.332764986629712, 25.047167153767735, 25.761562690082965, 26.475952145728595, 27.190336011683936, 27.904714727496145, 28.61908868783829, 29.33345824808956, 30.047823729103516, 30.762185421298863, 31.476543588182352, 32.19089846939369, 32.90525028334641, 33.61959922952588, 34.33394549049516, 35.048289233651346, 35.762630612767495, 36.4769697693505, 37.19130683383959, 37.90564192666726, 38.6199751592003, 39.33430663457682, 40.04863644845216, 40.762964689665075, 41.47729144083417, 42.19161677889273, 42.905940775569235, 43.62026349782013, 44.33458500822004, 45.04890536531427, 45.763224623938, 46.47754283550541, 47.19186004827236, 47.90617630757509, 48.62049165604775, 49.334806133820685, 50.04911977870149, 50.76343262634067, 51.47774471038327, 52.192056062607854, 52.9063667130542, 53.62067669014054, 54.33498602077156, 55.04929473043772, 55.76360284330703, 56.47791038230969, 57.192217369216245, 57.906523824710035, 58.62082976845425, 59.335135219153926, 60.049440194613666, 60.763744711791205, 61.47804878684707, 62.192352435190934, 62.90665567152471, 63.62095850988271, 64.3352609636691, 65.04956304569288, 65.76386476820053, 66.47816614290662, 67.19246718102224, 67.90676789328191, 68.62106828996849, 69.33536838093679, 70.0496681756356, 70.76396768312829, 71.47826691211242, 72.19256587093781, 72.90686456762393, 73.62116300987596, 74.33546120510012, 75.04975916041795, 75.76405688267984, 76.47835437847795, 77.19265165415811, 77.90694871583132, 78.62124556938441, 79.33554222049035, 80.04983867461782, 80.76413493704023, 81.47843101284448, 82.19272690693916, 82.90702262406211, 83.62131816878797, 84.33561354553247, 85.04990875968521, 85.76420401633567, 86.47835643833078, 87.1806171079943, 87.88133892276831, 88.68203579310355]

8. Wnioski:

Czas własnej implementacji działa o wiele krócej w porównaniu do metody użytych w bibliotece Numpy, nasz program działa w czasie O(n), własna implementacja dla macierzy A wypada korzystniej, ponieważ widzimy że poza elementami na diagonali, pod diagonalą i dwoma nad diagonalą posiadamy w tej macierzy zera. Łatwo jestem nam więc zoptymalizować program, dostrzegając eliminacje obliczeń wiążących się z tymi zerami.

Optymalizujemy również zużycie pamięci poprzez zapisanie naszej macierzy w tablicy dwuwymiarowej 4xN zamiast tablicy dwuwymiarowej NxN.