Sprawozdanie z zadania numerycznego 3

Problematyka: Rozwiązanie układu równań liniowych macierzowych za pomocą rozkładu LU i wpływ wartości pól macierzy na czas wyliczenia programu. Chcemy rozwiązać równanie nie tylko z poprawnym wynikiem, ale też liniową zależnością czasu.

Rozwiązanie: Zanim podjąłem się rozwiązania problemu postanowiłem przekształcić polecenie, aby nie musieć wyliczać macierzy odwrotnej. Dokonujemy tego poprzez obustronne pomnożenie równania $y=A^{-1}x$ lewostronnie o A, w wyniku czego dostajemy $Ay=x$.

Następnie przechodzimy do wyboru metody generacji macierzy A. Zdecydowałem, że ponieważ macierz jest wypełniona zerami poza diagonalą i trzema innymi „diagonalami" sąsiadującymi ze sobą. Zgodnie z naszą dyskusją na ćwiczeniach lokowanie pamięci na macierz n * n byłoby okropnie nieefektywne i całkowicie zbędne, dlatego optowałem w przyporządkowanie tym wartościom pamięci w postaci macierzy 4 * n, gdzie każda kolejna kolumna jest zajęta przez ten sam rodzaj wartości, który w przypadku dwóch z nich zależy od numeru wiersza w macierzy. Prezentuje się to w ten sposób:

0	1.2	0.1/1	$0.15/1^2$
0.2	1.2	0.1/2	$0.15/2^2$
…	…	…	…
0.2	1.2	0.1/n-2	$0.15/(n-2)^2$
0.2	1.2	0.1/n-1	0
0.2	1.2	0	0

Jest to odwzorowanie tabeli w poleceniu zadania NUM3. Algorytm generujący tabelę- zwykła pętla for i metoda append() powodują, że działa to dla wielu rozmiarów, nie tylko podanego w zadaniu N=124.

Po wygenerowaniu macierzy rozpoczynam pomiar czasu.

Do rozwiązania zadania posłużyłem się algorytmem Doolittle'a, ze względu na to że pomimo jego ogólnej złożoności n^3 w naszym przypadku dostaniemy rozwiązanie liniowe. Musimy rozbić naszą macierz na L(macierz Lower- dolną) i U (Upper- górną). Zgodnie z naszym algorytmem:

k<i k<j

$U_{ij}=A_{ij}-\Sigma L_{ik}U_{kj}$ oraz $L_{ij}=(A_{ij}- \Sigma L_{ik}U_{kj})/U_{jj}$

Z tym, że w naszym przykładzie wszystko poza pozycją wiersz nad obliczanym polem na tej samej kolumnie(oryginalnej macierzy) będzie zerem, więc będziemy odejmować tylko A[iteracja][pozycja] -A[iteracja][0] * A[iteracja-1][pozycja+1]. (pozycja w rzędzie nad elementem, który chcemy obliczyć).

Po rozbiciu równania do postaci LUy=x. Przyjmuję, że Uy=z, i rozwiązuje Lz=x przy użyciu backward i forward substitution(źródło:http://mathforcollege.com/ma/book2021/gaussianelimination-method-for-solving-simultaneous-linear-equations.html).

W tym momencie kończę też mierzyć czas.

Wyniki: Do sprawdzenia wyników użyłem numpy.linalg.solve() dla N=124 zgodnie z poleceniem.

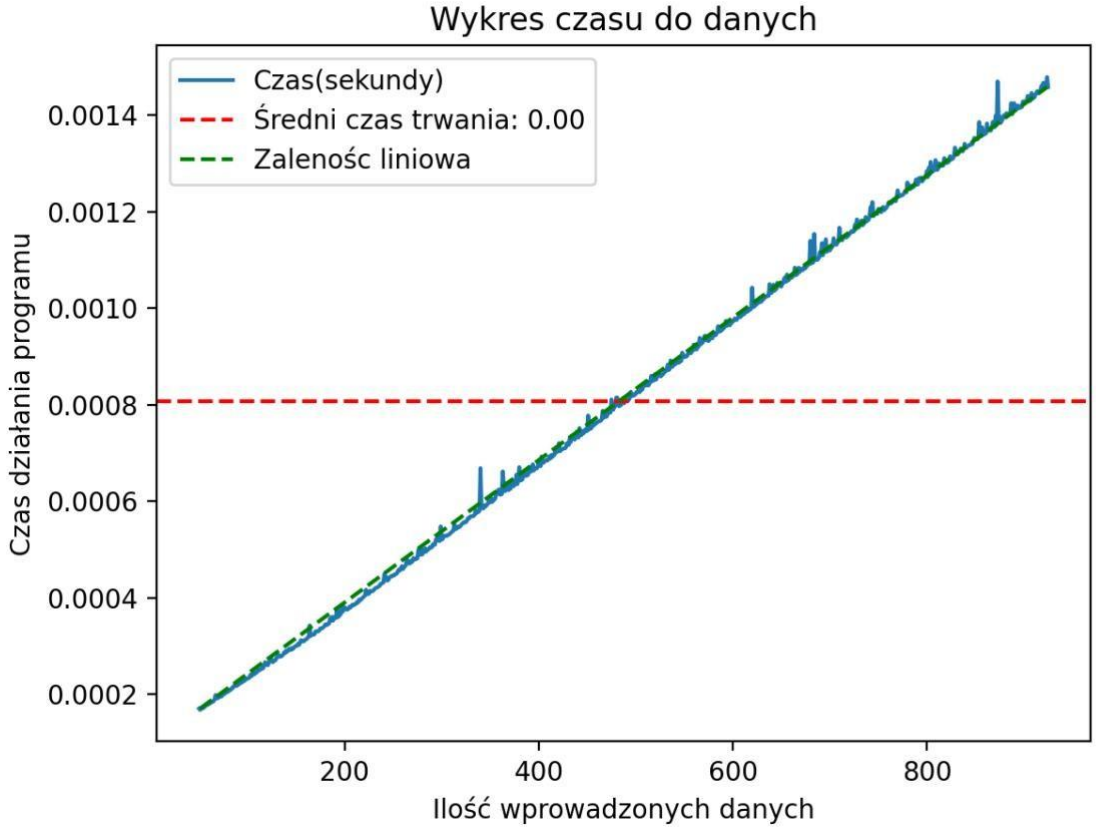
```
Szukane rozwiązanie to:  [0.448700827728733, 1.4132732869357947, 2.1348778535462736, 2.8690132654396248, 3.
5914885705595205, 4.311604959915503, 5.029827173723323, 5.747011462584994, 6.463503693914558, 7.179525964548697, 7.
8952125968955915, 8.610651859797315, 9.325903619364162, 10.041009954626537, 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.
33513519153926, 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, 83.62131816878798, 84.33561354553508, 85.
04990875857204, 85.76420381203626, 86.47849870460276, 87.19279247126808, 87.90778524137869, 88.68203579310355]
Numpy mowi:  [ 0.44870083  1.41327329  2.13487785  2.86901327  3.59148857  4.31160496
  5.02982717  5.74701146  6.46350369  7.17952596  7.8952126   8.61065186
  9.32590362 10.04100995 10.75600127 11.47090008 12.18572339 12.90048431
 13.61519305 14.32985776 15.04448494 15.7590799  16.47364698 17.18818978
 17.90271132 18.61721411 19.3317003  20.04617173 20.76062997 21.47507637
 22.18951211 22.90393821 23.61835557 24.33276499 25.04716715 25.76156269
 26.47595215 27.19033601 27.90471473 28.61908869 29.33345825 30.04782373
 30.76218542 31.47654359 32.19089847 32.90525028 33.61959923 34.33394549
 35.04828923 35.76263061 36.47696977 37.19130683 37.90564193 38.61997516
 39.33430663 40.04863645 40.76296469 41.47729144 42.19161678 42.90594078
 43.6202635  44.33458501 45.04890537 45.76322462 46.47754284 47.19186005
 47.90617631 48.62049166 49.33480613 50.04911978 50.76343263 51.47774471
 52.19205606 52.90636671 53.62067669 54.33498602 55.04929473 55.76360284
 56.47791038 57.19221737 57.90652382 58.62082977 59.33513522 60.04944019
 60.76374471 61.47804879 62.19235244 62.90665567 63.62095851 64.33526096
 65.04956305 65.76386477 66.47816614 67.19246718 67.90676789 68.62106829
 69.33536838 70.04966818 70.76396768 71.47826691 72.19256587 72.90686457
 73.62116301 74.33546121 75.04975916 75.76405688 76.47835438 77.19265165
 77.90694872 78.62124557 79.33554222 80.04983867 80.76413494 81.47843101
 82.19272691 82.90702262 83.62131817 84.33561355 85.04990876 85.76420381
 86.4784987  87.19279247 87.90778524 88.68203579]
```

Jak widać wyniki różnią się jedynie precyzją wyniku i notacją w której są podane. Poza tym obliczyłem zgodnie z poleceniem wyznacznik macierzy, jako iloczyn elementów na diagonali macierzy U (bo na L wszystkie sa jedynkami) (funkcja determinant).

Wyznacznik macierzy A = 6141973498.857843, sprawdziłem go w WolframAlpha jako log1.2(6141973498.857843) i otrzymałem 123.619 czyli prawie nasze N z polecenia.

Czas wykonywania programu: Rozwiązując równanie zakończyłem pierwszą część zadania, ale nie sprawdziłem jeszcze czy to równanie jest efektywne- czy czas jego wykonywania jest linowy. W tym celu użyłem pętli for która wykonuje pomiar czasu działania programu, który zdefiniowałem jako

funkcja program(data_size) zwracającą różnicę czasu początkowego i tego po wyliczeniu macierzy y, rozwiązania naszego równania.



Jak widać linia czasu wykonywania programu(niebieska) i linia łącząca czas wykonywania programu dla najmniejszej i największej ilości danych (zielona) pokrywają się dość dokładnie, a sam wykres bardzo przypomina wykres funkcji liniowej. Ewentualne „górki” na wykresie wynikają z tego, że komputer wykonuje wiele operacji jednocześnie i nie zawsze proces programu zostanie wykonany od razu, stąd opóźnienia i w efekcie wzrosty czasu niespowodowane zwiększeniem ilości danych. Sama Ilość jest wymiarem macierzy, przy 50 jest to 4x50, przy 100 4x100 itd. Średni czas trwania niestety nie wypisał się dokładnie, ale został zaznaczony czerwoną liną przerywaną nieco poniżej 0.00014 sekund(w legendzie obok niebieskiej linii Czas(sekundy)).

Wnioski: Wyniki programu są bardzo podobne do tych otrzymanych z funkcji biblioteki numpy. Jedyne różnice wynikają z zaokrągleń no i prawdopodobnie błędem precyzji. Jednocześnie wiadomo jest, że linalg.solve() używa rozkładu LU do rozwiązywania równań, co oznacza, że metoda jest poprawna, a sam program działa wedle wymagań zadania. Ponadto udowodniliśmy, że przy macierzy o

konkretnej budowie i wartościach jesteśmy w stanie przeprowadzić wyliczanie wyniku równania liniowego w czasie linowym.