Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Inżynierii Metali i Informatyki Przemysłowej

Sprawozdanie z Laboratorium: Metoda Eliminacji Gaussa

Przedmiot: Metody Numeryczne

Kierunek: Inżynieria Obliczeniowa

Autor: Filip Rak

Prowadzący przedmiot: dr hab. inż. Marcin Hojny

Data: 26 kwietnia 2024

Numer lekcji: 8

Grupa laboratoryjna: 4

Wstęp teoretyczny

Metoda eliminacji Gaussa jest jednym z podstawowych algorytmów stosowanych w numerycznej analizie do rozwiązywania układów równań liniowych. Działa ona poprzez stopniowe przekształcanie układu równań do postaci, z której łatwiej można uzyskać rozwiązanie.

Proces eliminacji Gaussa składa się z dwóch głównych etapów:

1. **Eliminacja w przód** - Celem jest przekształcenie macierzy współczynników układu równań do postaci trójkątnej górnej. Stosuje się operacje na wierszach macierzy: zamianę wierszy, mnożenie wiersza przez skalar oraz dodawanie lub odejmowanie wielokrotności jednego wiersza do innego. Dzięki temu pod główną przekątną macierzy pojawiają się zera, co upraszcza dalsze obliczenia.
2. **Podstawienie wsteczne** - Po uzyskaniu macierzy w formie trójkątnej górnej, można rozpocząć proces rozwiązywania układu równań. Rozpoczynając od ostatniego równania (gdzie mamy najmniej niewiadomych do obliczenia), oblicza się wartości niewiadomych kolejno wstecz aż do pierwszego równania.

Eliminacje w przód dokonujemy poprzez obliczenie współczynnika

$$m_{ij} = \frac{a_{ij}^{k}}{a_{jj}^{k}} \tag{1}$$

Gdzie:

i, j są indeksami elementu, który chcemy wyzerować w macierzy

k jest indeksem obecnego kroku w eliminacji, gdzie element a_{jj}^{k} jest elementem diagonalnym, względem którego wykonujemy eliminację w i-tym wierszu.

Podstawienie wsteczne opisane jest następującym wzorem:

$$x_i = \frac{b_i^n - \sum_{k=i+1}^{n} a_{i,k}^n x_k}{a_{ii}^n} \tag{2}$$

Gdzie:

x_i – jest to i-ta niewiadoma w układzie równań, którą chcemy znaleźć.

b_i^n – to i-ty element wektora wyrazów wolnych po wykonaniu eliminacji w przód, czyli wartość wyrazu wolnego w i-tym równaniu po przekształceniu macierzy do postaci trójkątnej górnej.

a_{ii}^{n} – jest to element na przekątnej macierzy, w i-tym wierszu i i-tej kolumnie, po wykonaniu eliminacji w przód.

$\sum_{k=i+1}^{n} a_{i,k}^{n} x_k$ – suma iloczynów elementów macierzy znajdujących się w i-tym wierszu, od kolumny $i+1$ do ostatniej kolumny n, i odpowiadających im już obliczonych wartości niewiadomych x_k. Suma ta jest odjęta od wyrazu wolnego b_i^n, ponieważ w trakcie podstawienia wstecznego korzystamy z wartości niewiadomych już obliczonych w poprzednich krokach.

Implementacja

Implementacja metody została przeprowadzona w języku C++ w postaci funkcji `gaussian_elimination`.

Funkcja przyjmuje dwa argumenty:

- Wskaźnik do macierzy rozszerzonej liczb zmiennoprzecinkowych: `double** matrix`.
- Liczbę całkowitą będącą stopniem macierzy: `int size`.

Typem zwracanym jest wskaźnik do tablicy liczb zmiennoprzecinkowych będących wynikami: `double*`

Pełna definicja funkcji `gaussian_elimination`

```
double* gaussian_elimination(double** matrix, int size)
{
    int k = 0;
    while (k <= size)
    {
        for (int i = k + 1; i < size; i++)
        {
            double m = matrix[i][k] / matrix[k][k];

            for (int j = k; j < size + 1; j++)
                matrix[i][j] -= matrix[k][j] * m;
        }

        k++;
    }

    double* results = new double[size];
    for (int i = size - 1; i >= 0; i--)
    {
        double sum = matrix[i][size];
        for (int j = i + 1; j < size; j++)
            sum -= matrix[i][j] * results[j];

        results[i] = sum / matrix[i][i];
    }

    return results;
}
```

Omówienie elementów funkcji `gaussian_elimination`

Poniższe część funkcji zajmują się eliminacją w przód. Pętla, w której zmienna **k** oznacza obecny krok eliminacji. Iteruje ona do momentu, aż **k** przekroczy rozmiar macierzy.

```
int k = 0; while (k <= size)
```

Pętla `for` przetwarzająca wiersze macierzy, zaczynając od wiersza bezpośrednio pod obecnym elementem diagonalnym. Celem jest wyzerowanie elementów poniżej obecnego elementu diagonalnego w **k**-tej kolumnie.

```
for (int i = k + 1; i < size; i++)
```

Obliczanie współczynnika **m**, który będzie używany do wyzerowania elementów w kolumnie poniżej elementu diagonalnego. Jest to stosunek elementu obecnego wiersza i kolumny **k** do elementu diagonalnego w kolumnie **k**. Obliczany jest wzorem (1).

```
double m = matrix[i][k] / matrix[k][k];
```

Wewnętrzna pętla `for`, która iteruje przez kolumny w **i**-tym wierszu, zaczynając od **k**-tej kolumny. Pętla ta służy do aktualizacji wartości w wierszu po wyzerowaniu elementów w **k**-tej kolumnie. Wartości przed **k**-tą kolumną są już zerami.

```
for (int j = k; j < size + 1; j++)
```

Aktualizacja elementów wiersza, które znajdują się w **k** -tej kolumnie i po prawej stronie od niej, przez odjęcie odpowiednio pomnożonych elementów wiersza **k**.

```
matrix[i][j] -= matrix[k][j] * m
```

Inkrementacja zmiennej **k**, aby przejść do następnego kroku eliminacji.

```
k++;
```

Następująca cześć funkcji zajmuje się podstawieniem wstecznym, realizując wzór (2). Alokacja pamięci dla tablicy, która będzie przechowywać wyniki układu równań (niewiadome).

```
double* results = new double[size];
```

Pętla **for** wykonująca podstawienie wsteczne, rozpoczynająca od ostatniego równania i idąca w kierunku pierwszego równania.

```
for (int i = size - 1; i >= 0; i--)
```

Inicjalizacja zmiennej **sum** wartością wyrazu wolnego równania **i**-tego po wykonaniu eliminacji.

```
double sum = matrix[i][size];
```

Wewnętrzna pętla **for**, która służy do obliczenia sumy iloczynów już wyznaczonych niewiadomych i ich współczynników z **i**-tego wiersza.

```
for (int j = i + 1; j < size; j++)
```

Aktualizacja sumy przez odjęcie iloczynu elementu macierzy i już obliczonej niewiadomej.

```
sum -= matrix[i][j] * results[j];
```

Obliczanie **i**-tej niewiadomej przez podzielenie zaktualizowanej sumy przez element diagonalny macierzy.

```
results[i] = sum / matrix[i][i];
```

Zwrócenie wskaźnika do tablicy z wynikami.

```
return results;
```

Testy na wybranych przykładach

Zaimplementowana metoda została przetestowana z losowo wygenerowanymi macierzami, oraz wynikami dla tych macierzy obliczonymi z wykorzystaniem biblioteki NumPy w języku Python. Testy skupiły się na sprawdzeniu poprawności implementacji w pracy z coraz to większymi macierzami.

Macierz 2x2

Rozszerzona macierz współczynników:

0.67619564 0.89823447 0.86030956

0.58450535 0.80142182 0.50400993

Otrzymane wyniki (NumPy):

14.01305603 -9.59132399

Otrzymane wyniki (`gaussian_elimination`):

14.0131 -9.59132

Wyniki są zgodne

Czas trwania algorytmu (mikrosekundy): 0

Macierz 4x4

Rozszerzona macierz współczynników:

0.20587593 0.93161931 0.69587167 0.16859267 0.23534964

0.80281588 0.70175646 0.01460726 0.33707521 0.80101967

0.61731012 0.42710096 0.13360009 0.73728234 0.36243149

0.08404864 0.66120946 0.42353606 0.12259852 0.94887664

Otrzymane wyniki (NumPy):

-5.02595496 5.8170708 -6.57341869 2.52106565

Otrzymane wyniki (`gaussian_elimination`):

-5.02596 5.81707 -6.57342 2.52107

Wyniki są zgodne

Czas trwania algorytmu (mikrosekundy): 0

Macierz 8x8

Rozszerzona macierz współczynników:

0.01676396 0.38258981 0.17282758 0.68775071 0.08306159 0.96526532 0.14177672 0.60240019 0.88621938

0.68862798 0.96724735 0.31244262 0.47932431 0.29721004 0.10719107 0.89254373 0.93886634 0.84664112

0.29024781 0.36871551 0.92736684 0.95971082 0.49312072 0.5146685 0.34333808 0.16359809 0.11292148

0.15251429 0.93554863 0.94020341 0.97749877 0.81792651 0.18833384 0.88340251 0.05336312 0.12188208

0.6952123 0.1108061 0.12848211 0.09716657 0.90106663 0.38908892 0.49241251 0.7865106 0.03390049

0.18707689 0.49825476 0.20622045 0.30510209 0.53229858 0.7667906 0.74296845 0.60054992 0.10089406

0.69976166 0.90982982 0.55210373 0.7360065 0.17723295 0.30181829 0.3274657 0.62622764 0.27262373

0.51150262 0.63146511 0.29125906 0.79174283 0.04305162 0.38190924 0.19934132 0.15883857 0.15834575

Otrzymane wyniki (NumPy):

-1.1145283 -0.73085712 -0.94647617 1.97517113 -0.26813817 -1.09249985 0.30472292 1.69869665

Otrzymane wyniki (`gaussian_elimination`):

-1.11453 -0.730857 -0.946476 1.97517 -0.268138 -1.0925 0.304723 1.6987

Wyniki są zgodne

Czas trwania algorytmu (mikrosekundy): 1

Macierz 16x16

Rozszerzona macierz współczynników:

Począwszy od tego testu, macierze wejściowe nie będą już umieszczane w sprawozdaniu ze względu na rozmiar.

Otrzymane wyniki (NumPy):

1.60331208 0.49680829 0.56796858 0.25380671 2.28774823 1.15878003 0.93738856 0.75100236 0.50647879 1.02798473 -1.48053595 -5.28952895 -4.47005135 0.39260368 -0.55526194 0.90675795

Otrzymane wyniki (`gaussian_elimination`):

1.60328 0.496784 0.567968 0.253808 2.28774 1.15878 0.937389 0.750996 0.506505 1.02797 -1.48055 -5.28947 -4.47004 0.392598 -0.555233 0.90674

Wyniki są zgodne

Czas trwania algorytmu (mikrosekundy): 6

Macierz 32x32

Otrzymane wyniki (NumPy):

0.96281969 -1.54300631 1.94809592 -2.32827322 -0.4979606 1.70962515 2.63102928 1.42096578 3.95320907 0.70333797 3.12036687 -4.06429205 0.93434385 3.63478842 -1.29178442 -6.36742875 -1.96588092 2.02826807 0.60996684 -1.02741469 -4.22231854 -1.83873014 -2.30543661 0.02649001 0.75313032 2.10474675 5.66995114 -5.71856334 -1.18920164 -4.57804557 3.51190594 3.95291237

Otrzymane wyniki (`gaussian_elimination`):

0.962825 -1.54301 1.9481 -2.32828 -0.497958 1.70963 2.63103 1.42097 3.95323 0.703329 3.12037 -4.06431 0.934339 3.6348 -1.29178 -6.36746 -1.96589 2.02828 0.609975 -1.02742 -4.22232 -1.83873 -2.30545 0.0264914 0.753136 2.10475 5.66997 -5.71859 -1.1892 -4.57807 3.51191 3.95292

Wyniki są zgodne

Czas trwania algorytmu (mikrosekundy): 35

Macierz 64x64

Otrzymane wyniki (NumPy):

1.248573 -1.928746 1.157442 -0.964319 0.094309 0.559454 0.069018 -0.416958 -0.719083 0.025619 0.640205 -1.531770 -1.561333 -0.269903 0.744233 0.101859 -0.703221 -0.365404 0.428693 -1.173555 2.129390 -0.286656 -1.339347 0.084512 0.338797 -0.092868 0.834537 1.263599 -0.751522 -1.182485 -0.526990 0.081071 0.069479 -0.273221 1.158217 -0.583634 -0.462652 -1.283065 0.523252 0.994322 1.991177 -0.249904 -1.094371 -1.121756 0.188321 -0.006784 0.156822 1.247656 0.754959 -0.903396 -0.860709 0.572576 -0.570189 1.887302 1.084766 0.443076 1.617242 0.486274 -1.092686 0.937793 0.380691 0.016282 -0.164332 -0.596956

Otrzymane wyniki (`gaussian_elimination`):

1.24856 -1.92873 1.15743 -0.964334 0.0943109 0.559448 0.069018 -0.416953 -0.719096 0.0256138 0.640205 -1.53176 -1.56133 -0.269907 0.74424 0.101838 -0.703218 -0.365417 0.428677 -1.17354 2.12939 -0.28665 -1.33935 0.0845292 0.338784 -0.0928671 0.834534 1.2636 -0.751514 -1.18249 -0.526986 0.0810834 0.0694648 -0.273226 1.15822 -0.583636 -0.462639 -1.28305 0.523261 0.994301 1.99116 -0.249905 -1.09437 -1.12176 0.188321 -0.00676839 0.156831 1.24765 0.754949 -0.903404 -0.860696 0.572576 -0.570195 1.8873 1.08478 0.443084 1.61725 0.486277 -1.09267 0.937793 0.3807 0.0162886 -0.164344 -0.596942

Wyniki są zgodne

Czas trwania algorytmu (mikrosekundy): 246

Macierz 128x128

Otrzymane wyniki (NumPy):

-0.757699 -0.456700 0.459905 1.058870 0.134573 0.018629 0.366659 -0.064250 -1.612084 -0.219182 -1.179670 -0.043774 0.509063 1.153054 0.736767 0.669584 -1.957031 -0.702050 -0.604217 0.043235 1.350841 -1.274269 0.452623 -0.521633 -1.645344 -0.575895 1.013635 0.190032 -0.268042 1.214798 2.260093 -0.945704 -0.419249 0.901754 -0.473747 -0.024530 -0.230769 2.290147 0.576909 -0.774488 0.674816 -0.021472 -0.855158 -0.305193 1.153889 0.154882 0.588985 0.471520 -0.522028 -1.243996 -0.202734 0.477552 -0.059322 0.560343 1.150465 0.649374 1.192158 -0.248003 0.271856 -0.117218 1.416539 0.292258 0.688281 -1.025858 0.406564 0.786624 -0.633497 -1.755142 -0.619782 -0.274446 0.242921 -0.096585 -0.063711 0.783185 -0.924602 -0.648731 -2.058018 0.278822 -0.405251 1.872755 -1.855941 -0.543675 0.063249 1.117268 -0.022572 -0.976977 -0.781755 0.146024 0.759302 -0.385753 -1.284597 -0.920824 -0.855757 -1.384691 -1.662075 0.490608 0.273609 0.143464 1.248993 -0.833571 -0.854374 -0.250523 -0.753738 0.023117 0.046136 -1.214695 2.285316 0.109239 0.131541 0.091827 0.253494 -0.197870 0.033987 -0.380350 -0.084072 0.704925 1.018437 0.936986 0.399319 -0.150506 -0.530046 0.605636 0.282127 0.471689 0.342898 0.925133 0.795187 -0.666641

Otrzymane wyniki (`gaussian_elimination`):

-0.757706 -0.456699 0.459919 1.05888 0.134569 0.0186283 0.366655 -0.064253 -1.61212 -0.219184 -1.17969 -0.0437625 0.509086 1.15306 0.736781 0.669598 -1.95706 -0.702068 -0.604228 0.043242 1.35086 -1.27428 0.452632 -0.52164 -1.64537 -0.575897 1.01365 0.190041 -0.268045 1.21481 2.26014 -0.945706 -0.419262 0.90176 -0.473763 -0.0245309 -0.230774 2.29017 0.576911 -0.774488 0.674809 -0.0214717 -0.85517 -0.305183 1.15391 0.154884 0.588991 0.471525 -0.522041 -1.24402 -0.202734 0.477557 -0.0593257 0.560344 1.15047 0.649373 1.19216 -0.24801 0.271864 -0.117215 1.41655 0.292259 0.68829 -1.02587 0.406565 0.786636 -0.633502 -1.75516 -0.619788 -0.274443 0.242936 -0.0965807 -0.0637205 0.78319 -0.92463 -0.64875 -2.05805 0.278837 -0.405255 1.87278 -1.85597 -0.543688 0.0632624 1.11727 -0.0225702 -0.976987 -0.781765 0.146033 0.759319 -0.385765 -1.2846 -0.92084 -0.855764 -1.38471 -1.66209 0.490616 0.27361 0.143466 1.24901 -0.833571 -0.854388 -0.250519 -0.753742 0.0231007 0.0461499 -1.2147 2.28534 0.109243 0.131539 0.0918331 0.253501 -0.197879 0.0339913 -0.380369 -0.0840807 0.704932 1.01845 0.936986 0.399318 -0.150494 -0.530067 0.605646 0.282131 0.471688 0.342912 0.925146 0.795201 -0.666646

Wyniki są zgodne

Czas trwania algorytmu (mikrosekundy): 1838

Macierz 256x256

Otrzymane wyniki (NumPy):

3.576870 -4.128263 -20.724744 -11.388634 -8.211510 -7.606752 21.398302 -2.911441 -5.753866 -2.878670 3.930341 3.436177 -0.894313 -0.948769 32.096361 -17.624233 -8.660989 -8.384920 -21.889254 -3.352512 5.714616 -10.992725 15.479435 -0.036836 -13.955542 19.063497 -20.789271 -15.973526 29.504176 26.539057 21.310612 9.048692 -2.327647 12.522302 -26.277246 6.285793 27.994265 -41.906496 10.010970 4.374295 -7.727059 -33.969825 5.193384 -8.625181 2.584167 -6.147876 -12.654108 -5.578382 16.031858 7.754183 -23.078161 -4.413083 -6.681851 -17.423813 -4.924090 27.428517 17.216973 22.041098 6.894445 -5.223708 -8.890039 -4.747730 2.375067 4.584285 32.258497 15.352910 -2.879758 16.790974 4.157940 -23.499168 -22.414647 -3.463139 -12.105036 -13.594319 16.114527 -19.649855 0.707176 -14.567423 -17.995476 -10.949259 6.632231 -0.842941 10.608946 10.175415 15.071181 -10.703843 -14.965983 -9.132674 -11.700874 6.296208 10.520305 12.218952 12.055279 9.619131 -17.418709 -7.685125 -29.023847 4.535321 -8.136516 -7.622965 13.918673 -4.111713 17.440081 10.188876 6.143499 -15.743532 9.802082 -4.965152 -20.336933 12.380548 34.566855 5.144638 8.154481 -3.962425 3.665296 -26.760167 16.345305 -0.093101 -22.063337 -30.992191 3.174750 6.179596 3.409792 24.359174 -10.265110 -16.539573 31.766869 2.807524 -4.507722 4.159444 22.521528 -3.035307 -0.656787 -7.310350 1.742789 -4.451431 -19.691234 24.205724 -9.022537 1.547000 19.282662 18.592367 6.088759 1.201291 12.981696 4.692939 8.654696 -11.660810 2.880640 4.539927 -12.462778 22.331865 9.928757 5.947365 -8.425480 2.605177 -7.019584 11.644904 16.927262 34.111705 -4.599935 -7.794602 -1.870921 -7.546162 13.305199 -4.706827 -17.100486 11.636120 1.753968 44.628394 0.201818 5.551493 -12.067175 27.300335 16.020820 -14.687718 10.942578 -3.966559 -10.399366 -2.482172 -8.122283 4.987930 2.417756 21.545805 -9.284182 -5.191351 -11.302500 3.277846 8.983361 4.251834 -24.531874 20.303286 23.034242 -5.247609 -48.298407 11.595909 -7.119832 -1.060213 2.151461 -6.915824 15.535939 -9.790842 -2.626586 -21.640480 4.832669 -1.825259 19.766997 15.290898 1.047104 4.710372 3.702477 20.201924 -1.379199 8.462344 -7.097329 -31.319510 -22.571797 6.467543 -10.455949 -9.619846 -23.053671 -16.064423 8.052007 8.934108 -2.962821 -23.899603 -1.864926 0.040704 -14.515896 28.577679 32.281697 9.692068 41.253259 5.803553 -15.957730 -15.064440 -18.475518 11.655556 1.168589 -1.351519 -22.335697 -1.089029 -7.427407 -31.640697 3.757130 -2.818766 10.606582 -24.761210 -13.641595 10.962202 -0.962526 2.579507 -6.321951 1.101785 -13.383192 -1.923216

Otrzymane wyniki (`gaussian_elimination`):

3.5765 -4.12784 -20.7232 -11.3878 -8.21066 -7.60656 21.3969 -2.91171 -5.75292 -2.87833 3.92982 3.4362 -0.894101 -0.948536 32.0939 -17.6228 -8.66022 -8.38402 -21.8875 -3.35237 5.71424 -10.9922 15.4781 -0.0364781 -13.9543 19.0624 -20.7878 -15.972 29.5016 26.5365 21.3083 9.04768 -2.32713 12.5209 -26.2748 6.28488 27.992 -41.9028 10.0101 4.37434 -7.7262 -33.9667 5.1929 -8.62458 2.58387 -6.14753 -12.6528 -5.57806 16.0306 7.75355 -23.0765 -4.41262 -6.68142 -17.4227 -4.9236 27.426 17.2155 22.0394 6.89349 -5.2234 -8.88943 -4.74721 2.37509 4.58379 32.2555 15.3517 -2.87957 16.7893 4.15781 -23.4973 -22.4131 -3.46297 -12.1041 -13.5936 16.1134 -19.6483 0.70707 -14.5664 -17.9936 -10.9486 6.6319 -0.842725 10.6082 10.1744 15.0699 -10.7027 -14.9645 -9.1318 -11.7 6.29544 10.5198 12.2179 12.0542 9.61831 -17.4176 -7.68437 -29.0214 4.53496 -8.13575 -7.62214 13.9174 -4.11138 17.4382 10.1878 6.14265 -15.7418 9.80134 -4.96488 -20.3353 12.3799 34.5643 5.14413 8.15377 -3.96172 3.66484 -26.7583 16.3439 -0.0931328 -22.0617 -30.9897 3.17459 6.1791 3.40931 24.3573 -10.2643 -16.5383 31.7643 2.80753 -4.50741 4.15917 22.5198 -3.03532 -0.656826 -7.30957 1.74261 -4.45128 -19.6897 24.2039 -9.02197 1.54654 19.2813 18.591 6.08828 1.20102 12.9807 4.69259 8.65388 -11.6601 2.88058 4.53962 -12.4616 22.33 9.92793 5.94712 -8.42433 2.60511 -7.01924 11.6444 16.926 34.1091 -4.59945 -7.79382 -1.87007 -7.54533 13.3041 -4.70644 -17.0991 11.6351 1.75361 44.625 0.202292 5.55162 -12.0659 27.2981 16.0198 -14.6867 10.9416 -3.96622 -10.3983 -2.4822 -8.1219 4.98753 2.41746 21.544 -9.28329 -5.19117 -11.3016 3.27765 8.98282 4.25182 -24.5302 20.3017 23.0323 -5.24712 -48.2947 11.595 -7.11864 -1.05993 2.15113 -6.91536 15.5347 -9.79009 -2.62586 -21.6385 4.8324 -1.82492 19.7651 15.2898 1.04683 4.70984 3.7026 20.2003 -1.37965 8.4617 -7.09698 -31.3168 -22.5699 6.4668 -10.4555 -9.61874 -23.0521 -16.0633 8.05161 8.93357 -2.96274 -23.8975 -1.86486 0.0407086 -14.515 28.5756 32.2787 9.69107 41.2496 5.80312 -15.9563 -15.0633 -18.4744 11.6544 1.16835 -1.35151 -22.334 -1.08858 -7.4268 -31.6383 3.75714 -2.81867 10.6057 -24.7593 -13.6403 10.9612 -0.962472 2.5794 -6.32129 1.10169 -13.3821 -1.92346

Wyniki są zgodne

Czas trwania algorytmu (mikrosekundy): 13891

Kolejne testy skupiły się wyłącznie na przebadaniu czasu trwania algorytmu

Macierz 512x512

Czas trwania algorytmu (mikrosekundy): 108412

Macierz 1024x1024

Czas trwania algorytmu (mikrosekundy): 888141

Macierz 2048x2048

Czas trwania algorytmu (mikrosekundy): 6973636

Macierz 4096x4096

Czas trwania algorytmu (mikrosekundy): 58637271

Analiza wyników

Wyniki testów wykazały, że implementacja algorytmu eliminacji Gaussa jest skuteczna i precyzyjna w rozwiązywaniu układów równań liniowych. Rozwiązania uzyskane za pomocą funkcji `gaussian_elimination` są zgodne z wynikami uzyskanymi przy użyciu biblioteki NumPy, co potwierdza poprawność obliczeń nawet dla bardzo dużych macierzy.

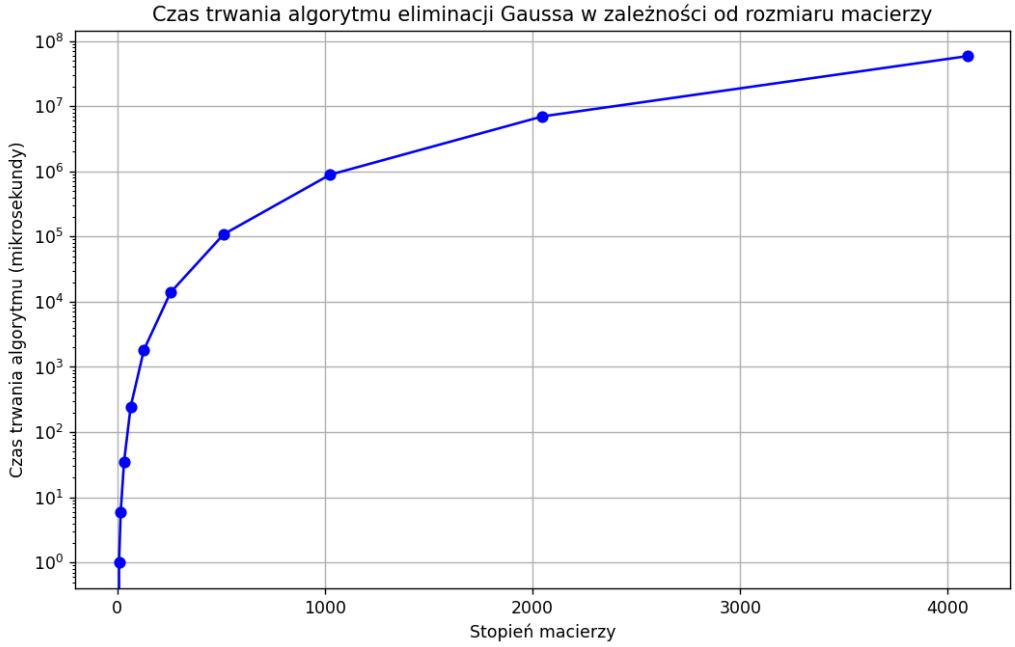
Czas trwania algorytmu

Obserwowane czasy wykonania algorytmu rosną w sposób niemal eksponencjalny wraz ze wzrostem rozmiaru macierzy:

- **Małe macierze (do 16x16)**: Czas trwania jest niezwykle krótki, wynosząc kilka mikrosekund, co wskazuje na bardzo szybką efektywność obliczeniową na małą skalę.
- **Średnie macierze (32x32 do 256x256)**: Wzrost czasu trwania algorytmu jest wyraźniejszy, od 35 mikrosekund do około 14 milisekund, co wynika z rosnącej złożoność obliczeniowej.
- **Duże macierze (512x512 do 4096x4096)**: Czas trwania znacząco się zwiększa, osiągając od 108 milisekund do 58 sekund dla największych testowanych macierzy. Znaczny wzrost czasu trwania pokazuje ograniczenia algorytmu przy bardzo dużych rozmiarach danych.

Stopień macierzy	2	4	8	16	32	64	128	256	512	1024	2048	4096
Czas	0	0	1	6	35	246	1838	13891	108412	888141	6973636	58637271

Tabela 1



Wykres 1

Podsumowanie

Implementacja metody eliminacji Gaussa działa poprawnie i jest zgodna z oczekiwaniami dla różnorodnych rozmiarów macierzy.

Czas trwania algorytmu jest akceptowalny dla małych i średnich macierzy, ale dla bardzo dużych macierzy rośnie eksponencjalnie.

Wyniki testów potwierdzają nie tylko poprawność implementacji, ale także pokazują jej ograniczenia wydajnościowe przy rozmiarach macierzy przekraczających 2048x2048, co może wymagać optymalizacji lub użycia bardziej zaawansowanych technik dla tak dużych danych.