MOwNiT Lab6 - Równania liniowe

Jakub Płowiec

31 Maja 2023

1 Cel zadania

Rozwiązywanie układów równań liniowych postaci Ax=b metodami bezpośrednimi.

2 Sprzęt

W zadaniu obliczeniowym posłużono się językiem Python 3.9.0 na systemie Windows 10 z procesorem Intel core i5-9600KF

3 Zadanie 1

3.1 Wprowadzenie

Elementy macierzy A o wymiarze n x n dla $i, j = 1, \ldots, n$ są określone wzorem:

$$\begin{cases} a_{1j} = 1 \\ a_{ij} = \frac{1}{i+j-1} \quad \text{dla } i \neq 1 \end{cases}$$

Jako wektor x została przyjęta permutacja postaci $[-1, 1, 1, -1, 1, 1, -1, 1, 1, \ldots]$.

Następnie metodą eliminacji Gaussa przyjmując jako niewiadomą wektor x, rozwiązujemy układ $Ax = b$ dla precyzji float64 oraz float32 badając jak błędy zaokrągleń zaburzają rozwiązanie dla różnych rozmiarów układu.

Do łatwiejszej i lepszej analizy wyników w każdym z zadań posłużymy się w poniższych przykładach normą maksimum wyliczaną w następujący sposób:

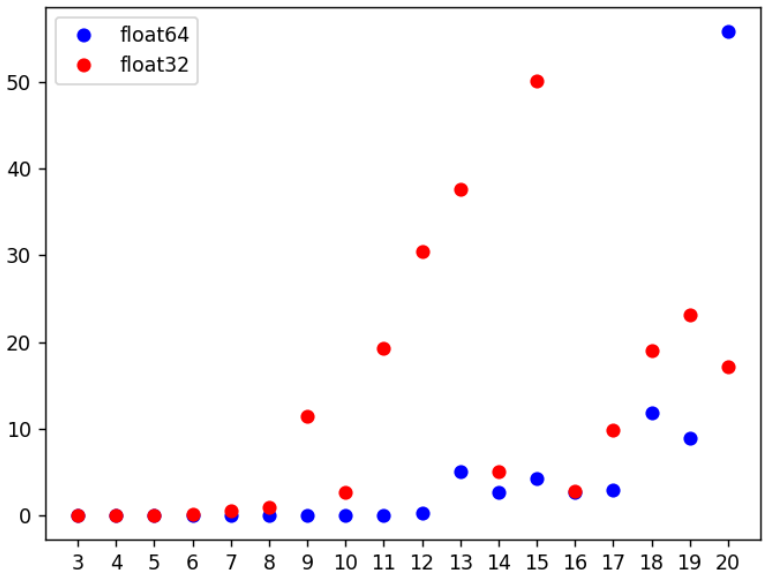
$$max|x_i - k_i|,$$

gdzie x_i to zadany ustawiony wektor x, a k_i wektor wyliczony w algorytmie.

3.2 Wyniki

n | f	float64	float32
3	2.33146e-15	0.0000e+00
4	1.33226e-15	1.6510e-05
5	4.58966e-13	1.8000e-04
6	1.03355e-10	6.7595e-02
7	2.18121e-10	5.2437e-01
8	4.85541e-08	8.7247e-01
9	2.73258e-08	1.1483e+01
10	1.10337e-04	2.6668e+00
11	5.55430e-03	1.9252e+01
12	2.20058e-01	3.0431e+01
13	5.05322e+00	3.7551e+01
14	2.70030e+00	5.1126e+00
15	4.22135e+00	5.0167e+01
16	2.61694e+00	2.8322e+00
17	2.98908e+00	9.8189e+00
18	1.18222e+01	1.8960e+01
19	8.83883e+00	2.3134e+01
20	5.58194e+01	1.7201e+01

Tabela 1: Norma maksimum dla dwóch precyzji w zad1



Wykres 1: Norma maksimum dla dwóch precyzji w zad1

Korzystając z wizualizacji wyników jesteśmy w stanie zauważyć, że do $n = 5$ wyniki dla obu dokładności były mniej więcej poprawne. Od $n = 6$ zaczynamy zauważać odbiegające wyniki dla float32, które wraz ze zwiększaniem wielkości równania nie maleją.
Dla $n = 9$ zaczynamy już otrzymywać dość absurdalne wyniki np. dla float32, gdzie otrzymujemy normę maksimum równą 11.
Poczynając od $n = 13$ również i float64 zaczyna zawodzić i generowane wyniki już są błędne.
Zwiększając coraz bardziej liczbę niewiadomych dokładność za bardzo już nie odgrywa żadnej roli, gdyż dla obu dokładności wyniki są absurdalne a równanie nie jest poprawnie rozwiązywane.

4 Zadanie 2

4.1 Wprowadzenie

Zostanie powtórzony eksperyment z poprzedniego zadania dla następującej macierzy:

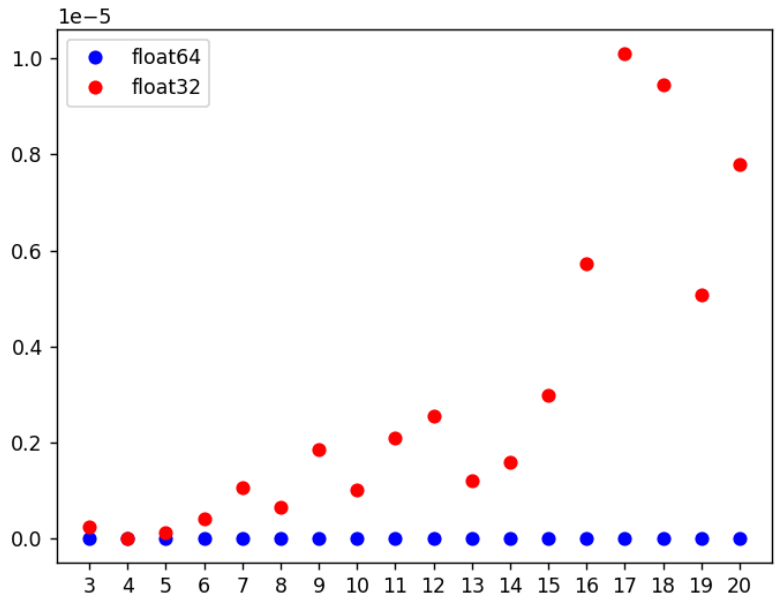
$$\begin{cases} a_{ij} = \frac{2i}{j}, & \text{dla } j \geq i \\ a_{ij} = a_{ji}, & \text{dla } j < i \end{cases}$$

Następnie zostaną porównane wyniki z tym co otrzymano w zadaniu 1.

4.2 Wyniki

n | f	float64	float32
3	4.44089e-16	2.3841e-07
4	4.44089e-16	0.0000e+00
5	7.77156e-16	1.1523e-07
6	1.11022e-15	4.1392e-07
7	1.77635e-15	1.0656e-06
8	1.88737e-15	6.4240e-07
9	4.55191e-15	1.84277e-06
10	6.32827e-15	1.0184e-06
11	9.43689e-15	2.08473e-06
12	1.21014e-14	2.55017e-06
13	1.08801e-14	1.19999e-06
14	8.88178e-15	1.58337e-06
15	1.25455e-14	2.98023e-06
16	9.99200e-15	5.72546e-06
17	1.39888e-14	1.01018e-05
18	2.02060e-14	9.45336e-06
19	1.37667e-14	5.08327e-06
20	2.14273e-14	7.78959e-06
50	1.90292e-13	1.35689e-04
100	1.43351e-12	8,65706e-04
150	4.16688e-12	1.45382e-03
200	1.29696e-11	4.24649e-03

Tabela 2: Norma maksimum dla dwóch precyzji w zad2



Wykres 2: Norma maksimum dla dwóch precyzji w zad2 do n=20

Najważniejszą obserwacją powinno być zanotowanie, iż wszystkie wyniki na wykresie reprezentowane są w postaci $1e-5$. Wszystkie wartości otrzymane dla dowolnej wielkości równania są bardzo bliskie zera, co w przypadku równania w pierwszym zadaniu było tak naprawdę nieosiągalne.

Ważną kwestią tutaj odgrywa tak zwany współczynnik uwarunkowania układu, który określa w jakim stopniu błąd reprezentacji danych wejściowych wpływa na błąd wyniku. Jeśli wskaźnik osiąga dla danego równania dużą wartość, wówczas nawet niewielki błąd danych może spowodować błędny wynik.
Wskaźnik jest wyliczany w następujący sposób:

$$||A^{-1}|| * ||A||$$

Odwracanie macierzy było wykonywane za pomocą biblioteki numpy.

Niżej zostanie przedstawiona wizualizacja porównania współczynnika uwarunkowania układu z pierwszego jak i drugiego zadania.

n | zad	zad1	zad2
3	4.85350e+02	7.84232e+00
4	1.71448e+04	1.50381e+01
5	5.96485e+05	2.53255e+01
6	2.05563e+07	3.90617e+01
7	7.04645e+08	5.65711e+01
8	2.40786e+10	7.81523e+01
9	8.21197e+11	1.04084e+02
10	2.79733e+13	1.34628e+02
11	9.52925e+14	1.70031e+02
12	3.22899e+16	2.10530e+02
13	6.06325e+17	2.56350e+02
14	5.82665e+17	3.07706e+02
15	2.10155e+18	3.64807e+02
16	6.50779e+17	4.27853e+02
17	2.85612e+18	4.97039e+02
18	3.26235e+20	5.72552e+02
19	4.52205e+18	6.54576e+02
20	1.88208e+18	7.43287e+02

Tabela 3: Uwarunkowanie układu kolejno dla zad1, zad2 w zależności od jego wielkości

Tak jak możemy zauważyć na Tabeli 3 - różnica jest ogromna. Wszystkie wartości w zadaniu 2 posiadały niemalże wykładnik równy 2, kiedy to wartości w zadaniu 1 prędko osiągają dwucyfrową wartość. Jest to idealne porównanie ukazujące jak układ wpływa na jego błędy obliczeniowe.

5 Zadanie 3

5.1 Wprowadzenie

Powtórzymy ponownie eksperyment z poprzednich zadań dla nowej macierzy, a następnie skorzystamy z metody przeznaczonej do rozwiązywania układów z macierzą trójdiagonalną - w celu porównania obu tych sposobów i ich rezultatów. Macierz wyglądaja następująco:

$$\begin{cases} a_{i,i} = -2i - 4 \\ a_{i,i+1} = i \\ a_{i,i-1} = \frac{2}{i}, \quad \text{dla } i > 1 \\ a_{i,j} = 0, \quad \text{dla } j < i - 1 \quad \text{oraz } j > i + 1 \end{cases}$$

W przypadku algorytmu Thomasa jesteśmy w stanie wykorzystać macierz 3 x n, natomiast w tym przypadku wykorzystana została x x n.

5.2 Wyniki

n | zad	zad3
3	3.61475e+00
4	5.06536e+00
5	6.64487e+00
6	8.35021e+00
7	1.01758e+01
8	1.21157e+01
9	1.41647e+01
10	1.63177e+01
11	1.85703e+01
12	2.09186e+01
13	2.33590e+01
14	2.58883e+01
15	2.85034e+01
16	3.12018e+01
17	3.39810e+01
18	3.68387e+01
19	3.97729e+01
20	4.27817e+01
50	1.61598e+02
100	4.50336e+02
150	8.23285e+02
200	1.26445e+03

Tabela 4: Uwarunkowanie układu kolejno dla zad3 w zależności od jego wielkości

Jak widać na powyższej tabeli, uwarunkowanie układu w tym przykładzie jest jeszcze lepsze niż w zadaniu drugim.

n | algorytm	Gauss64	Gauss32	Thomas64/32
3	0.0	0.0	0.0
4	0.0	0.0	0.0
5	0.0	0.0	0.0
6	0.0	0.0	0.0
7	0.0	0.0	0.0
8	0.0009973	0.0	0.0
9	0.0	0.0	0.0
10	0.0	0.0	0.0
11	0.0009973	0.0009973	0.0
12	0.0009970	0.0	0.0
13	0.0	0.0	0.0
14	0.0009970	0.0009975	0.0
15	0.0009970	0.0009970	0.0
16	0.0009977	0.0009973	0.0
17	0.0009970	0.0009973	0.0
18	0.0009982	0.0009975	0.0
19	0.0019946	0.0009975	0.0
20	0.0019943	0.0019950	0.0
50	0.0392923	0.0429420	0.0
100	0.3319818	0.3083395	0.0
150	1.0389521	1.0431900	0.0
200	2.3723649	2.4464857	0.0
300	8.0248115	8.1435306	0.0
1000	dużo	dużo	0.0115611

Tabela 5: Czasy wykonywania programu w [s] dla zad3 korzystając z dwóch algorytmów i dwóch dokładności

Powyższe wyniki zostały obliczone za pomocą biblioteki time. Jak jesteśmy w stanie zauważyć na Tabeli 5, to czas osiągany przez algorytm Thomasa jest dla rozsądnej wielkości równania równy zeru, co wpływa na jego korzyść w przypadku pozostałej metody niezależnie od dokładności jaką przyjmiemy.
Dodatkowo jeszcze możemy zerknąć na wyniki dla $n = 1000$, gdzie algorytm Thomasa radzi sobie nawet w 0.01 sekundy, kiedy to za pomocą Gaussa algorytm wykonuje się dobre parę minut.

n | f	Gauss64	Gauss32	Thomas64	Thomas32
3	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
4	1.11022e-16	1.19209e-07	1.11022e-16	2.52676e-08
5	3.33066e-16	0.00000e+00	3.33066e-16	5.62704e-08
6	3.33066e-16	0.00000e+00	3.33066e-16	5.92725e-08
7	3.33066e-16	1.19209e-07	3.33066e-16	5.76691e-08
8	3.33066e-16	1.06711e-07	3.33066e-16	5.83953e-08
9	3.33066e-16	1.25491e-07	3.33066e-16	5.75233e-08
10	3.33066e-16	1.21632e-07	3.33066e-16	5.76300e-08
11	3.33066e-16	1.29636e-07	3.33066e-16	5.76672e-08
12	3.33066e-16	1.19329e-07	3.33066e-16	5.76243e-08
13	3.33066e-16	1.19329e-07	3.33066e-16	5.75928e-08
14	3.33066e-16	1.19644e-07	3.33066e-16	7.69279e-08
15	3.33066e-16	1.19644e-07	3.33066e-16	7.55922e-08
16	3.33066e-16	1.19694e-07	3.33066e-16	7.73746e-08
17	3.33066e-16	1.19688e-07	3.33066e-16	7.73773e-08
18	3.33066e-16	1.28515e-07	3.33066e-16	7.70530e-08
19	3.33066e-16	1.19693e-07	3.33066e-16	7.70197e-08
20	3.33066e-16	1.20342e-07	3.33066e-16	7.70331e-08
50	3.33066e-16	1.19694e-07	3.33066-16	7.83795e-08
100	4.44089e-16	1.97723e-07	4.44089e-16	9.89738e-08
150	4.44089e-16	1.97723e-07	4.44089e-16	9.89738e-08
200	4.44089e-16	1.97723e-07	4.44089e-16	9.89738e-08

Tabela 6: Norma maksimum dla zad3 przedstawiającą dwa algorytmy z dwoma różnymi dokładnościami

Zauważmy jak algorytm Thomasa zwraca identyczne rezultaty jak w przypadku Gaussa dla dokładności float64. Norma maksimum w obu przypadkach zwraca prawie identyczną wartość niezależnie od wielkości układu. Można więc wywnioskować, iż obie metody zwracają poprawne wyniki, natomiast kwestia prędkości obliczeń leży znacząco na plus po stronie algorytun Thomasa.
Różnica jednak występuje dla dokładności float32 w której to wyniki są lekko na korzyść dla algorytmu Thomasa.

6 Ogólne wnioski

Współczynnik uwarunkowania układu odgrywa bardzo ważną rolę w algorytmach, których celem jest wydobycie żądanych rozwiązań. Dla poszczególnych dokładności zauważamy, jak wyniki bardzo zaczynają wzajemnie od siebie odbiegać w przypadku wysokiego współczynnika - gdzie dla małego różnice między float64, a float32 są bardzo małe.
Metoda Gaussa oraz Thomasa zwraca identyczne rezultaty rozwiązywania układu, natomiast algorytm Thomasa jest dużo szybszy jeżeli jesteśmy w stanie go wykorzystać.