Sprawozdanie

Układy równań liniowych – metody bezpośrednie

Autor: Adrian Żerebiec

1. Dane techniczne

Zadanie zostało zrealizowane na laptopie z procesorem AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz z systemem Windows 10, a do tego 8 GB pamięci RAM. Całość została napisana w języku Python3.

2. Wstęp teoretyczny do zadań

Symbol ‘n’, który został wykorzystany we wzorach, opisach, tabelach oraz wykresach, rozumiany jest jako wymiar układu.

2.1 Pomiar błędu

Jako kryterium, służące do obliczania wartości błędów, skorzystałem z normy maksimum. Błąd był wyznaczany jako maksymalna wartość bezwzględnych różnic kolejnych współrzędnych wektorów:

$$\max_{i=1,\dots,n} \{|x_i - \overline{x_i}|\}$$

gdzie:

$x_i - i$. współrzędna wyznaczonego wektora x,

$\overline{x_i} - i$. współrzędna zadanego wektora x.

(1)

2.2 Uwarunkowanie układu

Współczynnik uwarunkowania określa, w jakim stopniu błąd reprezentacji danych wejściowych wpływa na błąd wyniku. Jeżeli wskaźnik uwarunkowania dla danego problemu jest duży, wówczas nawet niewielki błąd danych może spowodować, że wynik będzie znacznie odbiegał od przewidywanego. Wskaźnik uwarunkowania oblicza się ze wzoru:

$$\kappa(A) = \|A^{-1}\| \cdot \|A\|$$

(2)

Norma jest określona poniższym wzorem:

$$\|A\| = \max_{1 < i < n} \sum_{j=1}^{n} |a_{i,j}|$$

(3)

2.3 Algorytm Gaussa

Algorytm eliminacji Gaussa polega zredukowaniu macierzy do postaci macierzy górnotrójkątnej za pomocą operacji, a następnie obliczeniu poszukiwanych wartości od końca. Elementy zapisujemy w macierzy kwadratowej o wymiarach n na n. Algorytm przechodzi przez każdy wiersz macierzy i wykonuje operacje w celu uzyskania postaci macierzy trójkątnej górnej. Po wykonaniu eliminacji Gaussa, obliczane są wartości zmiennych nieznanych X. Najpierw obliczane jest ostatnia zmienna. Metoda ta ma złożoność $O(n^3)$.

2.4 Algorytm Thomasa

Algorytm Thomasa, który działa podobnie do algorytmu Gaussa. Wykorzystuje fakt, iż poza trzema głównymi przekątnymi, pozostałe wartości w macierzy są zerami. Nie musimy iterować po wszystkich wierszach pod główną przekątną, gdyż zamiast tego operujemy jedynie na wartości głównej przekątnej i wartości z przekątnej pod nią. Dzięki temu, dla każdego pivota wykonane zostanie O (1) operacji, a ponieważ przekątna główna ma n elementów, ostatecznie złożoność obliczeniowa algorytmu Thomasa jest rzędu O(n).

3. Zadanie 1.

Dany jest układ równań liniowych $\mathbf{A}\boldsymbol{x}=\mathbf{b}$.

1) Elementy macierzy **A** o wymiarze $n \times n$ są określone wzorem:

$$\begin{cases} a_{1j} = 1 \\ a_{ij} = \dfrac{1}{i+j-1} \quad dla \;\; i \neq 1 \end{cases} \qquad i, j = 1, \ldots, n$$

Przyjmij wektor $\boldsymbol{x}$ jako dowolną n–elementową permutację ze zbioru { 1, -1 } i oblicz wektor **b**.

Następnie metodą eliminacji Gaussa rozwiąż układ równań liniowych $\mathbf{A}\boldsymbol{x}=\mathbf{b}$ (przyjmując jako niewiadomą wektor $\boldsymbol{x}$). Przyjmij różną precyzję dla znanych wartości macierzy **A** i wektora **b**. Sprawdź, jak błędy zaokrągleń zaburzają rozwiązanie dla różnych rozmiarów układu (porównaj – zgodnie z wybraną normą – wektory $\boldsymbol{x}$ *obliczony* z $\boldsymbol{x}$ *zadany*). Przeprowadź eksperymenty dla różnych rozmiarów układu.

3.1 Opracowanie zadania

W celu uzyskania błędów wykorzystamy algorytm Gaussa do rozwiązania układów, a następnie korzystając z (1) obliczmy wartości błędów. Możemy zakładać, iż float64 poradzi sobie lepiej dla zadanego układu ze względu na większą dokładność.

Otrzymane wartości błędów

Z tabeli 1 wynika, że, błędy zwiększają się wraz ze wzrostem rozmiaru naszego układu. Dla różnych precyzji wartości błędów również od siebie odbiegają.

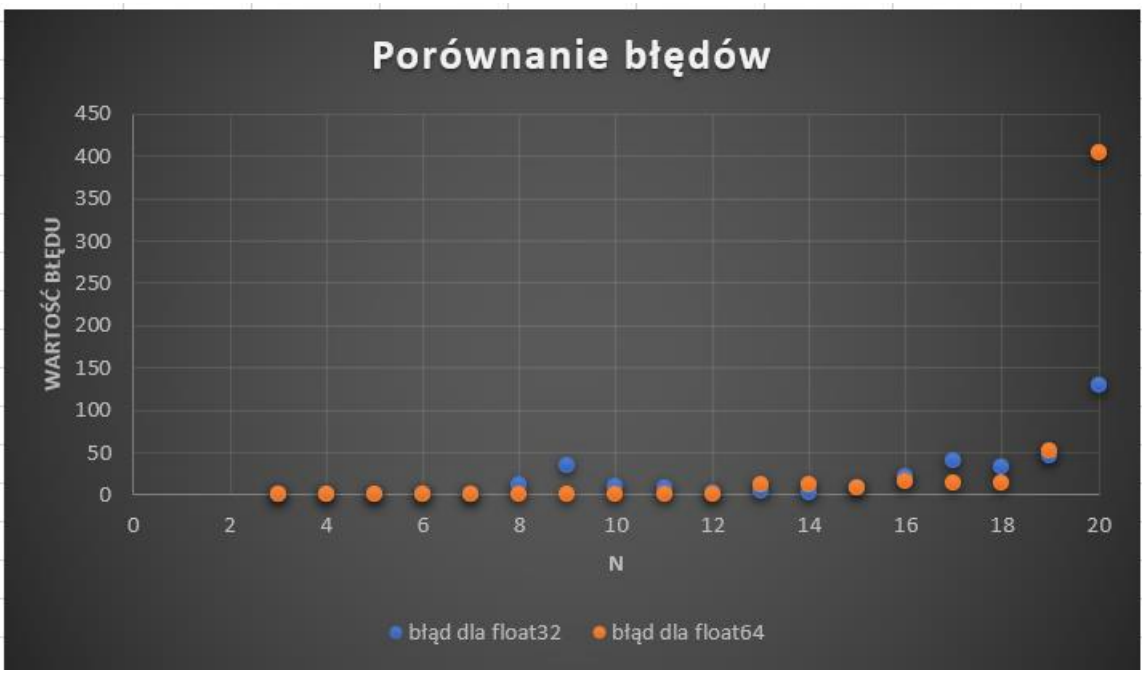
Początkowo zdecydowanie lepiej radzi sobie float64, gdyż wynik powyżej jeden uzyskujemy dopiero dla n równego 13. Jego wyniki rosną systematycznie tego wymiaru. Następnie jednak wyniki są wyraźnie gorsze i zaczynają być losowe, co wynika ze złego uwarunkowania naszego układu.

Float32 wyniki powyżej jeden uzyskał już dla n=7, zatem bardzo szybko. Analogicznie jak dla float64 od tego momentu wyniki nie są w żaden sposób systematyczne, a wręcz wydają się losowe.

n	błąd dla float32	błąd dla float64
3	2,68E-06	0
4	5,329071E-15	2,364775E-13
5	2,254701E-04	6,513567E-12
6	3,714318E-11	2,486656E-10
7	1,53901	9,436947E-09
8	12,73622	7,987404E-08
9	35,23430	3,423918E-07
10	10,10908	1,052929E-04
11	7,42139	7,647956E-03
12	2,22514	0,71986
13	4,01387	12,53223
14	3,24557	11,99258
15	7,71166	9,06911
16	20,90875	15,63721
17	40,91737	14,41393
18	32,50551	13,50042
19	46,78297	52,15635
20	128,88750	403,45554

Tabela 1: tabela z wartościami błędów dla różnych precyzji w zależności od wymiarów

Na wykresie 1 lepiej widać początkowy powolny wzrost a następnie losowość wyników dla zadania 1.



Wykres 1: wykres błędów na podstawie tabeli 1

4. Zadanie 2.

2) Powtórz eksperyment dla macierzy zadanej wzorem:

$$\begin{cases} a_{ij} = \dfrac{2i}{j} & dla \;\; j \geq i \\ a_{ij} = a_{ji} & dla \;\; j < i \end{cases} \qquad i, j = 1, \ldots, n$$

Porównaj wyniki z tym, co otrzymano w przypadku układu z punktu 1). Spróbuj uzasadnić, skąd biorą się różnice w wynikach. Sprawdź uwarunkowanie obu układów.

4.1 Opracowanie zadania

Podobnie jak w zadaniu 1 do uzyskania błędów wykorzystujemy algorytm Gaussa oraz sposób liczenia błędu opisany w punkcie 2.1 tego sprawozdania. Również, analogicznie jak w poprzednim zadaniu zakładamy, że wyniki dla float64 będą zdecydowanie bardziej dokładne niż dla float32.

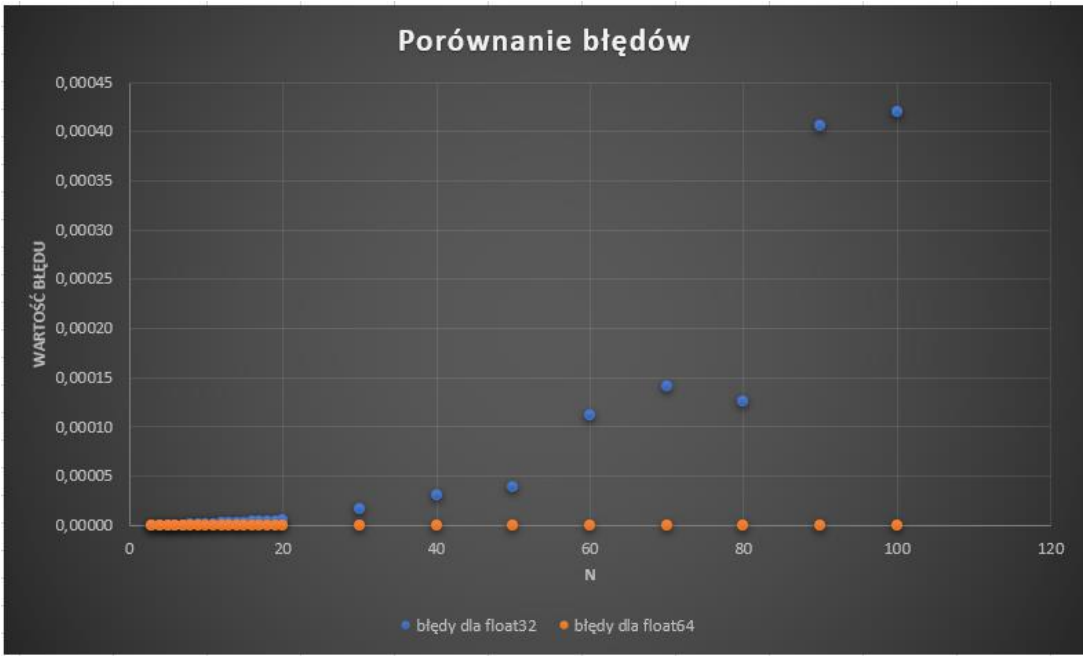
Otrzymane wartości błędów

Z tabeli 2, tak jak zakładaliśmy, wynika, że błędy uzyskane dla float64 są zdecydowanie mniejsze niż dla float32. Jednak warto zauważyć, iż nawet dla float32 nie przekroczyliśmy w żadnym miejscu wartości jeden. Łatwo stwierdzić, więc że układ ten jest zdecydowanie lepiej uwarunkowany niż ten w zadaniu 1. Błędy uzyskane dla float64 są zdecydowanie mniejsze od float32, co wynika z zakresu precyzji obu typów.

Poniżej znajduje się także wykres 2. Ilustruje on wyniki z tabeli 1. Dla lepszej widoczności jako maksymalne n przyjęto 100.

n	błędy dla float32	błędy dla float64
3	5,960465E-08	2,220446E-16
4	0	2,220446E-16
5	1,192093E-07	3,330669E-16
6	1,192093E-07	6,661338E-16
7	1,192093E-07	1,332268E-15
8	8,344650E-07	3,774758E-15
9	8,344650E-07	2,664535E-15
10	8,344650E-07	1,665335E-15
11	9,536743E-07	2,442491E-15
12	2,861023E-06	1,176836E-14
13	2,861023E-06	1,243450E-14
14	2,861023E-06	1,287859E-14
15	2,861023E-06	1,620926E-14
16	3,933907E-06	2,020606E-14
17	3,933907E-06	2,020606E-14
18	3,933907E-06	2,020606E-14
19	3,933907E-06	2,153833E-14
20	6,020069E-06	2,464695E-14
30	1,704693E-05	4,862777E-14
40	3,027916E-05	1,145750E-13
50	3,874302E-05	1,587619E-13
60	1,123548E-04	2,822187E-13
70	1,406670E-04	2,950973E-13
80	1,259446E-04	6,054046E-13
90	4,053116E-04	7,649437E-13
100	4,203320E-04	9,219292E-13
200	1,667738E-03	7,401413E-12
500	3,117538E-02	8,360179E-11

Tabela 2: tabela z wartościami błędów dla różnych precyzji w zależności od wymiarów



Wykres 2: wykres błędów na podstawie tabeli 2 z n maksymalnie równym 100

Jak widać na wykresie 2, float32 przyjmuje większe błędy niż float64. Gdy dla n = 100 float64 nadal jest bardzo bliski zera, float32 osiąga już wartości sięgające około 0,0004.

4. Uwarunkowanie układów w zadaniach 1 i 2

Wskaźnik uwarunkowania macierzy jest miarą jak bardzo zmieni się rozwiązanie x układu równań w stosunku do zmiany b. Jeżeli wskaźnik macierzy jest duży, to nawet mały błąd b może spowodować duże błędy w x. Jak widać w tabeli 3 uwarunkowanie w zadaniu 1 jest o wiele gorsze niż w zadaniu 2. Stąd wynika, iż w zadaniu 1 nawet mały błąd może powodować, iż uzyskane wyniki będą bardzo różnić się od przewidywanych. Wyniki przestawiono tylko dla n z zakresu od 3 do 20, gdyż już tutaj widać ogromne rozbieżności. Warto również zauważyć, że wraz ze wzrostem rozmiaru rośnie nam wskaźnik uwarunkowania, więc szanse otrzymania błędnych wyników dla dużych macierzy są większe niż dla małych.

n	Wskaźnik uwarunkowania dla zadania 1	Wskaźnik uwarunkowania dla zadania 2
3	2,160000E+02	1,444444
4	2,880000E+03	1,833333
5	2,800000E+04	2,233333
6	2,268000E+05	2,644444
7	1,629936E+06	3,031746
8	1,286208E+07	3,448413
9	1,120002E+08	3,849206
10	8,841438E+08	4,249206
11	6,473792E+09	4,659428
12	4,407939E+10	5,055219
13	1,347671E+11	5,465475
14	2,459224E+11	5,868898
15	1,733309E+11	6,268898
16	1,084972E+11	6,678405
17	3,045485E+11	7,077618
18	1,187493E+14	7,485025
19	3,750396E+11	7,889565
20	4,003893E+11	8,289565

Tabela 3: wskaźniki uwarunkowania dla zadań 1 oraz 2 w zależności od wielkości n

Poniżej znajduje się dodatkowa tabela 4 porównująca wartości błędów. Widać, że wskaźnik uwarunkowania ma ogromny wpływ na wartości uzyskiwanych błędów nawet dla stosunkowo małych n.

	Float32		Float64	
n	Zadanie 1	Zadanie 2	Zadanie 1	Zadanie 2
3	2,68E-06	5,960465E-08	0	2,220446E-16
4	5,329071E-15	0	2,364775E-13	2,220446E-16
5	2,254701E-04	1,192093E-07	6,513567E-12	3,330669E-16
6	3,714318E-11	1,192093E-07	2,486656E-10	6,661338E-16
7	1,53901	1,192093E-07	9,436947E-09	1,332268E-15
8	12,73622	8,344650E-07	7,987404E-08	3,774758E-15
9	35,23430	8,344650E-07	3,423918E-07	2,664535E-15
10	10,10908	8,344650E-07	1,052929E-04	1,665335E-15
11	7,42139	9,536743E-07	7,647956E-03	2,442491E-15
12	2,22514	2,861023E-06	0,71986	1,176836E-14
13	4,01387	2,861023E-06	12,53223	1,243450E-14
14	3,24557	2,861023E-06	11,99258	1,287859E-14
15	7,71166	2,861023E-06	9,06911	1,620926E-14
16	20,90875	3,933907E-06	15,63721	2,020606E-14
17	40,91737	3,933907E-06	14,41393	2,020606E-14
18	32,50551	3,933907E-06	13,50042	2,020606E-14
19	46,78297	3,933907E-06	52,15635	2,153833E-14
20	128,88750	6,020069E-06	403,45554	2,464695E-14

Tabela 4: tabela porównująca wartości błędów uzyskane dla takich samych n w zadaniach 1 oraz 2

5. Zadanie 3.

Powtórz eksperyment dla jednej z macierzy zadanej wzorem poniżej (macierz i parametry podane w zadaniu indywidualnym). Następnie rozwiąż układ metodą przeznaczoną do rozwiązywania układów z macierzą trójdiagonalną. Porównaj wyniki otrzymane dwoma metodami (czas, dokładność obliczeń i zajętość pamięci) dla różnych rozmiarów układu. Przy porównywaniu czasów należy pominąć czas tworzenia układu. Opisz, jak w metodzie dla układów z macierzą trójdiagonalną przechowywano i wykorzystywano macierz A.

Jako parametry przyjmujemy m = 2 oraz k = 9.

a) $(m,k$ - parametry zadania):

$$\begin{cases} a_{i,i} = k \\ a_{i,i+1} = \dfrac{1}{i+m} \\ a_{i,i-1} = \dfrac{k}{i+m+1} \quad dla \quad i>1 \\ a_{i,j} = 0 \qquad dla \quad j<i-1 \quad oraz \quad j>i+1 \end{cases} \qquad i,j = 1,\ldots,n$$

5.1 Wykorzystanie i zapis macierzy A

W układzie z macierzą trójdiagonalną postanowiłem wykorzystać macierz o wymiarach n na n. Można byłoby także przedstawić ją w postaci tablicy n na 3, gdzie górny wiersz to elementy pod przekątną główną, trzecia elementom nad przekątną główną oraz środkowa jako przekątna główna. Macierz A wykorzystujemy najpierw do stworzenia macierzy B, uzyskiwanej jako iloczyn macierzy A oraz wektora X. Następnie dane macierzy A oraz macierzy B w algorytmie Thomasa są wykorzystywane do uzyskania odpowiednich wyników.

5.2 Opracowanie zadania

Wartości błędów obliczałem z wykorzystaniem (1) dla dwóch typów: float32 oraz float64. Dodatkowo mierzyłem czasy wykonań dla poszczególnych algorytmów, czyli dla metody Gaussa oraz dla metody Thomasa. Patrząc na złożoność algorytmów lepsze wyniki pod względem czasu powinniśmy uzyskać dla algorytmu Thomasa. Błędy otrzymywane w obydwu przypadkach powinny być zbliżone do siebie, najlepiej takie same.

Błędy i czasy uzyskane dla float32

n	błąd dla Gaussa	błąd dla Thomasa	czas dla Gaussa [s]	czas dla Thomasa [s]
3	0	0	7,8600016E-05	2,6300084E-05
4	0	0	6,6499924E-05	3,0700001E-05
5	0	1,1920929E-07	6,1400002E-05	3,2800017E-05
6	5,9604645E-08	1,1920929E-07	7,0699956E-05	3,5699923E-05
7	5,9604645E-08	1,1920929E-07	9,8100048E-05	4,1499967E-05
8	5,9604645E-08	1,1920929E-07	1,1829997E-04	4,4999993E-05
9	5,9604645E-08	1,1920929E-07	1,4289992E-04	5,0100032E-05
10	5,9604645E-08	1,1920929E-07	1,6599998E-04	5,3600059E-05
11	1,1920929E-07	1,1920929E-07	1,8410000E-04	5,7399971E-05
12	1,1920929E-07	1,1920929E-07	2,0900008E-04	6,4800028E-05
13	1,1920929E-07	1,1920929E-07	2,5260006E-04	6,7499932E-05
14	1,1920929E-07	1,1920929E-07	2,7660001E-04	7,2099967E-05
15	1,1920929E-07	1,1920929E-07	3,1679997E-04	7,6800003E-05
20	1,1920929E-07	1,1920929E-07	5,3460000E-04	1,0299997E-04
30	1,1920929E-07	1,1920929E-07	1,2700999E-03	1,5429989E-04
40	1,1920929E-07	1,1920929E-07	2,0339000E-03	2,0090002E-04
50	1,1920929E-07	1,1920929E-07	3,1181000E-03	2,4500000E-04
60	1,1920929E-07	1,7881393E-07	4,5015001E-03	2,9829994E-04
70	1,1920929E-07	1,7881393E-07	6,0928001E-03	3,4939998E-04
80	1,1920929E-07	1,7881393E-07	7,9454000E-03	4,1160011E-04
90	1,1920929E-07	1,7881393E-07	9,8105001E-03	4,3869996E-04
100	1,1920929E-07	1,7881393E-07	1,2400400E-02	4,8490008E-04
150	1,1920929E-07	1,7881393E-07	3,3480600E-02	7,4249995E-04
200	1,1920929E-07	1,7881393E-07	5,0657700E-02	9,8710007E-04
300	1,1920929E-07	1,7881393E-07	1,1443900E-01	1,4698999E-03
500	1,1920929E-07	1,7881393E-07	3,4889330E-01	2,4274000E-03

Tabela 5: błędy i czasy uzyskane dla float32 dla algorytmów Thomasa oraz Gaussa

Według tabeli 5, nasza hipoteza o tym, że czasy dla algorytmu Gaussa będą większe się potwierdziła. Jak widać w tabeli 5 dla n=500 różnica sięga dwóch rzędów wielkości. Zyskujemy zatem wiele czasu wykorzystując algorytm Thomasa. Warto także spojrzeć na wartości błędów. W obydwu przypadkach błędy są bardzo małe, jednak dla około n = 60 lepsze, ale tylko odrobinę, uzyskuje algorytm Gaussa.

Błędy i czasy uzyskane dla float64

n	błąd dla Gaussa	błąd dla Thomasa	czas dla Gaussa [s]	czas dla Thomasa [s]
3	0	0	5,4799952E-05	1,8200022E-05
4	2,2204460E-16	2,2204460E-16	4,2199972E-05	1,8800027E-05
5	2,2204460E-16	2,2204460E-16	4,9900031E-05	2,1299929E-05
6	2,2204460E-16	2,2204460E-16	6,5400032E-05	2,3399945E-05
7	2,2204460E-16	2,2204460E-16	8,2700048E-05	2,6499969E-05
8	2,2204460E-16	2,2204460E-16	1,0140007E-04	2,8699986E-05
9	2,2204460E-16	2,2204460E-16	1,2330001E-04	3,1600008E-05
10	2,2204460E-16	2,2204460E-16	1,4890009E-04	3,5200035E-05
11	2,2204460E-16	2,2204460E-16	1,7530005E-04	3,8099941E-05
12	2,2204460E-16	2,2204460E-16	2,0470005E-04	4,0599960E-05
13	2,2204460E-16	2,2204460E-16	2,4049997E-04	4,3699984E-05
14	2,2204460E-16	2,2204460E-16	2,7299998E-04	4,6700006E-05
15	2,2204460E-16	2,2204460E-16	3,1270005E-04	4,8900023E-05
20	2,2204460E-16	2,2204460E-16	5,2390003E-04	6,4799911E-05
30	2,2204460E-16	2,2204460E-16	1,1404000E-03	9,4399904E-05
40	2,2204460E-16	2,2204460E-16	1,9689000E-03	1,2270000E-04
50	2,2204460E-16	2,2204460E-16	3,0433000E-03	1,5580002E-04
60	2,2204460E-16	2,2204460E-16	4,3743000E-03	1,8290000E-04
70	2,2204460E-16	2,2204460E-16	5,9697001E-03	2,1979993E-04
80	2,2204460E-16	2,2204460E-16	7,6968001E-03	2,4630001E-04
90	2,2204460E-16	2,2204460E-16	1,1822400E-02	3,1619996E-04
100	2,2204460E-16	2,2204460E-16	1,2060600E-02	3,1880010E-04
150	2,2204460E-16	2,2204460E-16	2,9286700E-02	4,6830007E-04
200	2,2204460E-16	2,2204460E-16	4,9797000E-02	6,1440002E-04
300	2,2204460E-16	2,2204460E-16	1,1280720E-01	9,2570006E-04
500	2,2204460E-16	2,2204460E-16	3,7490710E-01	1,5739000E-03

Tabela 6: błędy i czasy uzyskane dla float64 dla algorytmów Thomasa oraz Gaussa

W tabeli 6 widzimy, że analogicznie jak dla float32, algorytm Thomasa uzyskał lepsze czasy. Tak samo jak w tamtym przypadku dla n=500 są to czasy dwa rzędy niższe. Warto zauważyć również, że w tym przypadku uzyskiwane błędy są takie same, co utwierdza nas w przekonaniu, iż wyniki są poprawne.

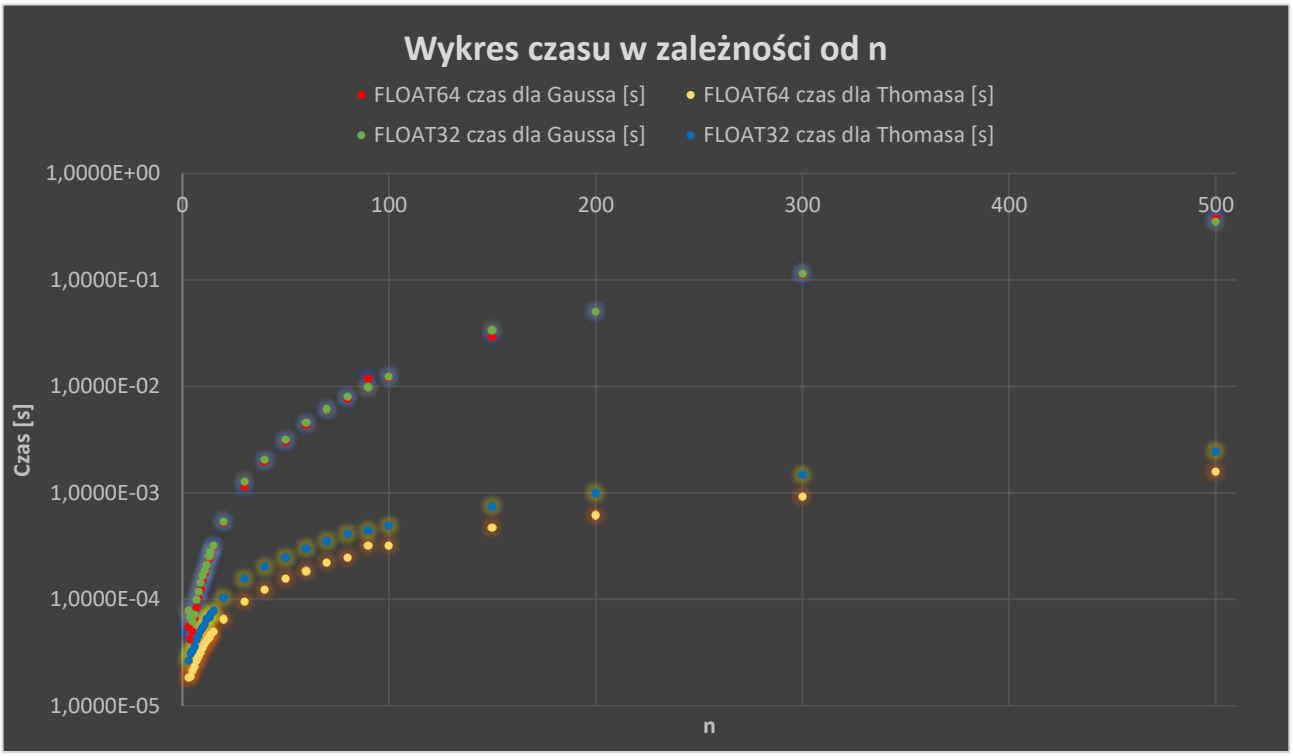
Dodatkowa tabela do porównania czasów

W tabeli 7 widać, iż dla mniejszej precyzji otrzymaliśmy nieco gorsze czasy.

	FLOAT64		FLOAT32	
n	czas dla Gaussa [s]	czas dla Thomasa [s]	czas dla Gaussa [s]	czas dla Thomasa [s]
3	5,4799952E-05	1,8200022E-05	7,8600016E-05	2,6300084E-05
4	4,2199972E-05	1,8800027E-05	6,6499924E-05	3,0700001E-05
5	4,9900031E-05	2,1299929E-05	6,1400002E-05	3,2800017E-05
6	6,5400032E-05	2,3399945E-05	7,0699956E-05	3,5699923E-05
7	8,2700048E-05	2,6499969E-05	9,8100048E-05	4,1499967E-05
8	1,0140007E-04	2,8699986E-05	1,1829997E-04	4,4999993E-05
9	1,2330001E-04	3,1600008E-05	1,4289992E-04	5,0100032E-05
10	1,4890009E-04	3,5200035E-05	1,6599998E-04	5,3600059E-05
11	1,7530005E-04	3,8099941E-05	1,8410000E-04	5,7399971E-05
12	2,0470005E-04	4,0599960E-05	2,0900008E-04	6,4800028E-05
13	2,4049997E-04	4,3699984E-05	2,5260006E-04	6,7499932E-05
14	2,7299998E-04	4,6700006E-05	2,7660001E-04	7,2099967E-05
15	3,1270005E-04	4,8900023E-05	3,1679997E-04	7,6800003E-05
20	5,2390003E-04	6,4799911E-05	5,3460000E-04	1,0299997E-04
30	1,1404000E-03	9,4399904E-05	1,2700999E-03	1,5429989E-04
40	1,9689000E-03	1,2270000E-04	2,0339000E-03	2,0090002E-04
50	3,0433000E-03	1,5580002E-04	3,1181000E-03	2,4500000E-04
60	4,3743000E-03	1,8290000E-04	4,5015001E-03	2,9829994E-04
70	5,9697001E-03	2,1979993E-04	6,0928001E-03	3,4939998E-04
80	7,6968001E-03	2,4630001E-04	7,9454000E-03	4,1160011E-04
90	1,1822400E-02	3,1619996E-04	9,8105001E-03	4,3869996E-04
100	1,2060600E-02	3,1880010E-04	1,2400400E-02	4,8490008E-04
150	2,9286700E-02	4,6830007E-04	3,3480600E-02	7,4249995E-04
200	4,9797000E-02	6,1440002E-04	5,0657700E-02	9,8710007E-04
300	1,1280720E-01	9,2570006E-04	1,1443900E-01	1,4698999E-03
500	3,7490710E-01	1,5739000E-03	3,4889330E-01	2,4274000E-03

Tabela 7: zbiorcza tabela z czasami dla algorytmów Thomasa oraz Gaussa dla różnych n i precyzji

Postanowiłem także stworzyć wykres 3 na podstawie tabeli 7, aby lepiej zwizualizować różnice w czasach dla poszczególnych wartości n.



Wykres 3: wykres zależności czasu od n dla różnych typów dla algorytmów Gaussa oraz Thomasa

Na wykresie 3 możemy zauważyć, że dla metody Gaussa otrzymujemy zdecydowanie gorsze czasy. Najgorzej wypada tak dla Float64, ze względu na precyzje obliczeń oraz złożoność algorytmu. Dla algorytmu Thomasa widzimy, że lepsze czasy uzyskał Float32.

6. Wnioski

Zadania 1-2

Wskaźnik uwarunkowania jest bardzo istotny, gdyż jeśli jest on duży to nawet mały błąd może powodować, że uzyskiwane wyniki będą bardzo odbiegać od rzeczywistych.

Warto stosować typy o większej precyzji, gdyż dzięki swojemu zakresowi powodują mniejsze błędy w obliczeniach.

Jeśli mamy do czynienia z źle uwarunkowanym zadaniem to wyniki od pewnego wymiaru n staną się losowe.

Zadanie 3

Dzięki lepszej złożoności czasowej algorytm Thomasa zdecydowanie szybciej radzi sobie z zadaniem niż algorytm Gaussa. Dodatkowo, jeśli wdrożymy pewne poprawki, zyskamy także na złożoności pamięciowej.

W zadaniu 3 widać, że dzięki dobremu uwarunkowaniu otrzymujemy dla float64 wszędzie takie same, bardzo małe wartości błędów.

7. Źródła

1)Wykład doktor Katarzyny Rycerz z przedmiotu Metody obliczeniowe w nauce i technice

2) Wikipedia na temat algorytmu Gaussa oraz algorytmu Thomasa