Rozwiązywanie równań liniowych metodami bezpośrednimi

Krystian Madej, 05.06.2024

1. Treść zadania

Przyjmij wektor x jako dowolną n-elementową permutację ze zbioru $\{-1, 1\}$.

1. Elementy macierzy A o wymiarze nxn są określone wzorami:
$$\begin{aligned} & \boldsymbol{a_{1j} = 1} \\ & \boldsymbol{a_{ij} = \frac{1}{i+j-1}, \quad dla\ i \neq 1} \\ & \boldsymbol{i, j = 1, \dots, n} \end{aligned}$$
Oblicz wektor b zadany wzorem $b = Ax$. Następnie metodą eliminacji Gaussa rozwiąż układ równań $Ax = b$. Przyjmij różną precyzję obliczeń dla wartości macierzy A i wektora b. Sprawdź w jaki sposób błędy zaokrągleń wpływają na rozwiązania różnych rozmiarów układu, porównując zgodnie z wybraną normą wektor x zadany i wektor x obliczony.
2. Powtórz obliczenia dla macierzy zadanej wzorem:
$$\begin{aligned} & \boldsymbol{a_{ij} = \frac{2i}{j} \quad \text{dla j} \geq i} \\ & \boldsymbol{a_{ij} = a_{ji} \quad \text{dla j} < \text{i}} \\ & \boldsymbol{i, j = 1, \dots, n} \end{aligned}$$
Porównaj z wynikami punktu **1.** Uzasadnij, skąd biorą się różnice w wynikach oraz sprawdź uwarunkowanie obu układów.
3. Powtórz eksperyment dla macierzy zadanej wzorem:
$$\begin{aligned} & \boldsymbol{a_{i,i} = -m \cdot i - k} \\ & \boldsymbol{a_{i,i+1} = i} \\ & \boldsymbol{a_{i,i-1} = \frac{m}{i} \quad dla\ i > 1} \\ & \boldsymbol{a_{i,j} = 0 \quad dla\ j < i - 1 \quad oraz\ j > i + 1} \\ & \boldsymbol{i, j = 1, \dots, n} \\ & \boldsymbol{m = 2,\ k = 6} \end{aligned}$$
Następnie ponownie rozwiąż układ równań, tym razem algorytmem Thomasa, dla macierzy trójdiagonalnych. Porównaj wyniki metod Gaussa i Thomasa (dokładność obliczeń, czas) dla różnych

rozmiarów układu. Opisz sposób przechowywania macierzy trójdiagonalnej.

2. Środowisko obliczeń

Obliczenia zostały wykonane przy pomocy języka **C++20** na systemie **Windows 11**, kompilacja 22631.3593, na **12-wątkowym** procesorze **64-bitowym** Intel Core i5-11400H 2.70GHz, z najwyższym możliwym priorytetem, kod kompilowany kompilatorem **MSVC** (wersja 19.39). Obliczenia wykonane na typach wbudowanych: float (32-bitowy) i double (64-bitowy).

3. Użyte biblioteki i programy pomocnicze

Wykresy rysowano programem Excel z pakietu Microsoft Office.

Do instalacji bibliotek **C++** użyto programu **conan**, wersja 2.3.2.

Najważniejsze użyte biblioteki:

- <format> - łatwe formatowanie
- <numbers> - stałe matematyczne
- <future> - obiekty std::future oraz std::async
- <thread> - wielowątkowość
- <fstream> - zapisywanie do plików
- <chrono> - zegary i operacje na jednostkach czasu
- <random> - algorytm Mersenne Twister (std::mt19937)

4. Sposób obliczeń

4.1 Norma porównywania błędów

Za normę do porównywania błędów przyjęto błąd średni:

$$\frac{\sum_{i=1}^{n}|X_i - x_i|}{n}$$

X – wektor zadany
x – wektor obliczony
n – rozmiar obu wektorów

4.2 Uwarunkowanie układu

Uwarunkowanie układów jest obliczane wzorem:

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

gdzie:

A – macierz współczynników układu

$|| \cdot || = \max_{i=1,\dots,n} \sum_{j=1}^{n} |a_{ij}|$ – norma macierzy (maksymalna suma wartości bezwzględnych wierszy)
A^{-1} – jest obliczana metodą Gaussa, sprowadzając macierz blokową $[A|I]$ do postaci $[I|B]$, gdzie B to macierz odwrotna A

4.3 Metoda Gaussa

Do obliczeń użyto metody Gaussa z Partial Pivotingiem.

5. Implementacja obliczeń

Na początku zaimplementowano funkcje operacji na macierzach i wektorach:

- add_rows – funkcja dodająca jeden wiersz macierzy do innego, złożoność czasowa: $O(n)$
- subtract_rows – funkcja odejmująca jeden wiersz od innego, złożoność czasowa: $O(n)$ lub $O(k)$, gdzie k to długość zadanego przedziału
- multiply_rows – funkcja mnożąca wiersz przez skalar, złożoność czasowa: $O(n)$
- divide_rows – funkcja dzieląca wiersz przez skalar, złożoność czasowa: $O(n)$
- swap_rows – funkcja zamieniająca kolejnośc wierszy w macierzy, złożoność czasowa: O(1)

Funkcje te przyjmują macierz, na której operacja ma zostać wykonana, drugą macierz lub wektor na których operacje także zostanie wynonana. Funkcje add_rows i subtract_rows przyjmują też numery wiersza do którego dodać/od którego odjąć inny wiersz oraz numer wiersza dodawanego/odejmowanego oraz skalar, przez który wiersz dodawany/odejmowany ma być przemnożony przed operacją. Dodatkowo subtract_rows przyjmuje opcjonalne parametry określające przedział odejmowania. Funkcje multiply_rows i divide_rows przyjmują dodatkowo numer wiersza mnożonego/dzielonego oraz skalar. Funkcja swap_rows przymuje dodatkowo numery zamienianych wierszy.

Przy użyciu tych funkcji zaimplementowano funkcje solve, oraz solve_tridiag, rozwiązujące zadane układy równań odpowiednio metodą Gaussa i Thomasa. Zakładając postąć układu $Ax = b$, obie funkcje przyjmują macierz A oraz wektor b, a zwracają trójkę: obliczony wektor x, macierz A po przekształceniach oraz wektor b po przekształceniach. Implementacja algorytmu Thomasa przyjmuje macierz tylko w postaci kwadratowej nxn.

Zaimplementowano też funkcję gauss_inv, obliczająca macierz odwrotną zadanej macierzy metodą Gaussa opisaną w punkcie 4.2. Przy jej pomocy napisano funkcję condition obliczającą wskaźnik uwarunkowana zadanej macierzy, wedługo metody w punkcie 4.2.

Zaimplementowana też funkcję avg_error, obliczającą błąd średni wg. wzoru w punkcie 4.1.

Opisane wyżej funkcje są szablonowe, podczas kompilacji dostosują się do zadanego im typu zmiennoprzecinkowego.

Początkowy wektor x jest wyznaczany osobno dla typu float i double, przy pomocy osobnych instancji algorytmu pseudolosowego Mersenne Twister. Obie instancje mają identyczne ustawienia początkowe, co gwarantuje identyczność obu wariantów wektora x.

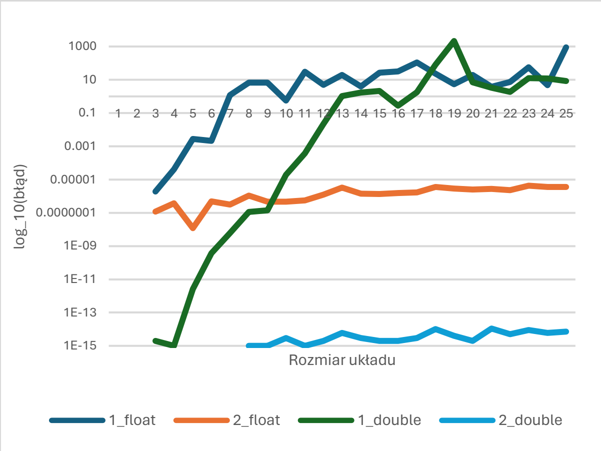
Pliki z wynikami obliczeń zostaną zapisane w folderze z plikiem wykonywalnym.

6. Porównanie wyników

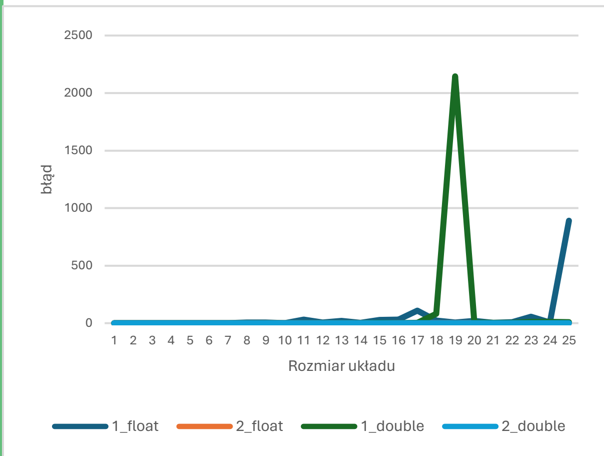
6.1 Błędy obliczeń w podpunktach 1. i 2.

	float, 32-bit		double, 64-bit	
n\punkt	1	2	1	2
1	0	0	0	0
2	0	0	0	0
3	1.91E-06	1.19E-07	2E-15	0
4	4.16E-05	3.73E-07	1E-15	0
5	0.002816	1.19E-08	2.56E-12	0
6	0.002086	4.87E-07	3.77E-10	1E-15
7	1.171676	3.15E-07	5.96E-09	0
8	6.663738	1.09E-06	1.15E-07	1E-15
9	6.6756	4.64E-07	1.43E-07	1E-15
10	0.556096	4.59E-07	1.93E-05	3E-15
11	30.3578	5.47E-07	0.000383	1E-15
12	5.036991	1.25E-06	0.022582	2E-15
13	19.30298	3.32E-06	1.026384	6E-15
14	3.995234	1.44E-06	1.707251	3E-15
15	26.86791	1.37E-06	2.118683	2E-15
16	32.11998	1.55E-06	0.277626	2E-15
17	108.256	1.67E-06	1.796039	3E-15
18	23.11374	3.62E-06	80.93642	1E-14
19	5.518388	2.85E-06	2145.408	4E-15
20	19.73093	2.52E-06	7.003007	2E-15
21	3.750141	2.74E-06	3.381475	1.1E-14
22	7.45298	2.29E-06	1.852299	5E-15
23	57.35627	4.28E-06	12.73779	9E-15
24	4.666538	3.53E-06	12.18685	6E-15
25	890.5539	3.53E-06	8.266211	7E-15

Tabela 1. Średnie błędy obliczeń w podpunktach 1. i 2.



Wykres 1. Średnie błędy obliczeń w podpunktach 1. i 2., skala logarytmiczna



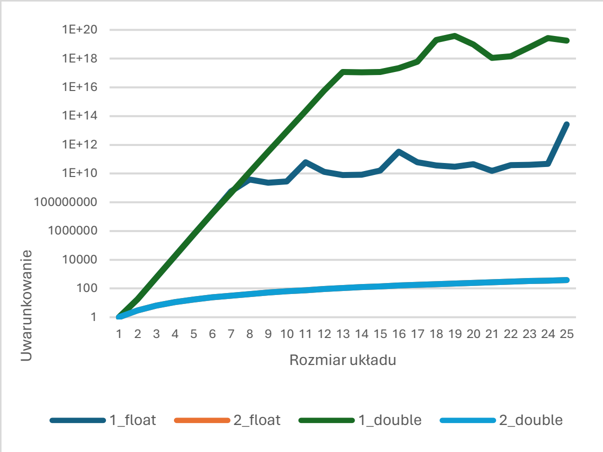
Wykres 2. Średnie błędy obliczeń w podpunktach 1. i 2.

Na wykresach 1. i 2. oraz tabeli 1. widać, że obliczenia dla typu float mają zwykle większy błąd niż dla typu double. Wyraźnie widoczny jest też trend wzrostowy błędu średniego wraz ze wzrostem rozmiaru układu. Błąd dla punktu 2. jest o rzędy wielkości mniejszy od błędu dla punktu 1.

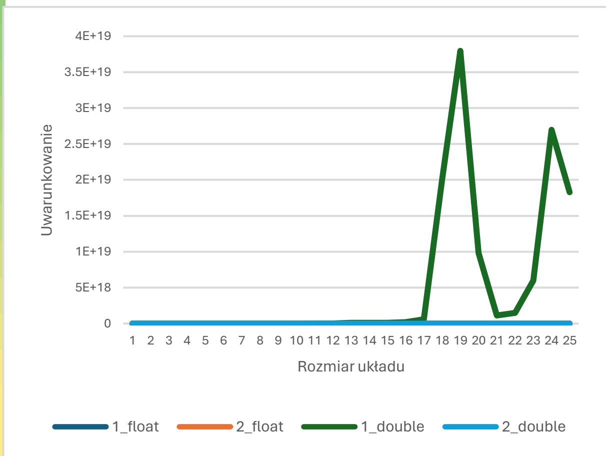
6.2 Uwarunkowanie w podpunktach 1. i 2.

	float, 32-bit		double, 64-bit	
n\punkt	1	2	1	2
1	1	1	1	1
2	18	3	18	3
3	570.0016	6.5	570	6.5
4	18620.34	11	18620	11
5	567951.8	16.75	567630	16.75
6	16648311	23.79999	16385292	23.8
7	5.74E+08	31.83333	4.56E+08	31.83333
8	3.8E+09	41.38099	1.23E+10	41.38095
9	2.33E+09	51.96429	3.28E+11	51.96429
10	2.8E+09	63.73804	8.57E+12	63.7381
11	6.1E+10	76.88062	2.22E+14	76.88056
12	1.28E+10	90.99403	5.91E+15	90.99394
13	7.8E+09	106.5768	1.2E+17	106.5768
14	8.07E+09	123.2469	1.13E+17	123.2469
15	1.6E+10	141.0502	1.19E+17	141.0502
16	3.3E+11	160.2821	2.13E+17	160.2817
17	6.01E+10	180.4794	6.1E+17	180.4793
18	3.61E+10	202.0968	2E+19	202.0957
19	2.99E+10	224.8536	3.8E+19	224.8526
20	4.61E+10	248.6871	9.83E+18	248.687
21	1.52E+10	274.0071	1.11E+18	274.0064
22	3.86E+10	300.291	1.48E+18	300.2893
23	4.03E+10	327.9407	6E+18	327.9388
24	4.75E+10	356.7848	2.7E+19	356.7825
25	2.64E+13	386.6537	1.83E+19	386.6484

Tabela 2. Uwarunkowanie układów w podpunktach 1. i 2.



Wykres 3. Uwarunkowanie układów w podpunktach 1. i 2., skala logarytmiczna



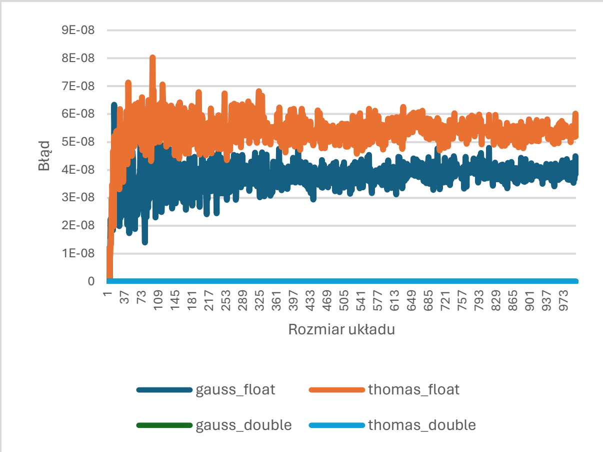
Wykres 4. Uwarunkowanie układów w podpunktach 1. i 2.

Jak widać na tabeli 2. i wykresach 3. i 4. współczynnik uwarunkowania rośnie wraz z rozmiarem układu. Uwarunkowanie jest o rzędu wielości większe w podpunkcie 2. w porównaniu do podpunktu 1. To dlatego wartości błędów były mniejsze w podpunkcie 1. w porównaniu do podpunktu 2. Miejsca nagłych przyrostów uwarunkowania pokrywają się z miejscami nagłych przyrostów błędów, np. n=25 i n=19 dla punktu pierwszego, odpowiednio dla typów float i double. łatwo także zauważyć, iż uwarunkowanie jest zbliżone w podpunkcie 2., niezależnie od typu zmiennoprzecinkowego.

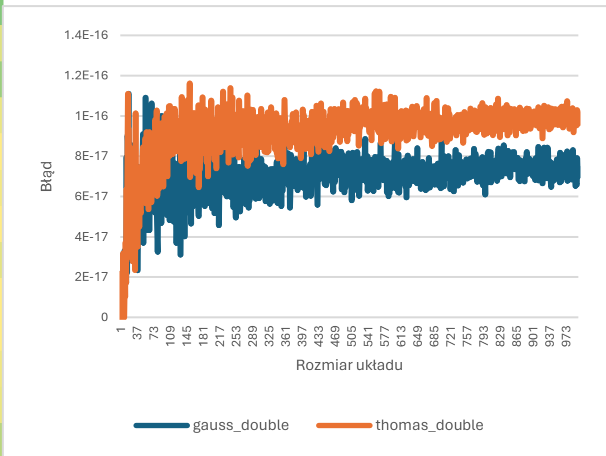
6.3 Błędy obliczeń w podpunkcie 3.

	float, 32-bit		double, 64-bit	
n\punkt	3 gauss	3 thomas	3 gauss	3 thomas
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	1.19E-08	1.19E-08	2.22E-17	2.22E-17
6	0	0	0	0
7	8.51E-09	8.51E-09	3.17E-17	3.17E-17
8	2.24E-08	1.49E-08	1.39E-17	0
9	1.99E-08	1.32E-08	1.23E-17	0
10	2.38E-08	2.98E-08	3.33E-17	2.22E-17
11	2.71E-08	2.71E-08	2.02E-17	1.01E-17
12	2.48E-08	3.48E-08	2.78E-17	3.7E-17
13	1.83E-08	2.75E-08	1.71E-17	1.71E-17
14	3.41E-08	4.68E-08	7.93E-17	6.34E-17
15	1.99E-08	1.99E-08	2.22E-17	2.96E-17
16	6.33E-08	5.22E-08	9.02E-17	6.94E-17
17	4.91E-08	4.56E-08	7.84E-17	1.11E-16
18	3.64E-08	3.31E-08	7.4E-17	7.4E-17
19	4.39E-08	4.71E-08	1.11E-16	9.35E-17
20	5.96E-08	5.36E-08	5E-17	6.66E-17
21	2.84E-08	5.39E-08	7.4E-17	8.46E-17
22	3.79E-08	3.52E-08	8.58E-17	8.07E-17
23	2.85E-08	3.63E-08	7.24E-17	7.24E-17
24	2.73E-08	3.48E-08	6.01E-17	6.94E-17
25	3.58E-08	3.81E-08	5.33E-17	4.44E-17

Tabela 3. Średnie błędy obliczeń w podpunkcie 3.



Wykres 5. Średnie błędy obliczeń w podpunkcie 3.



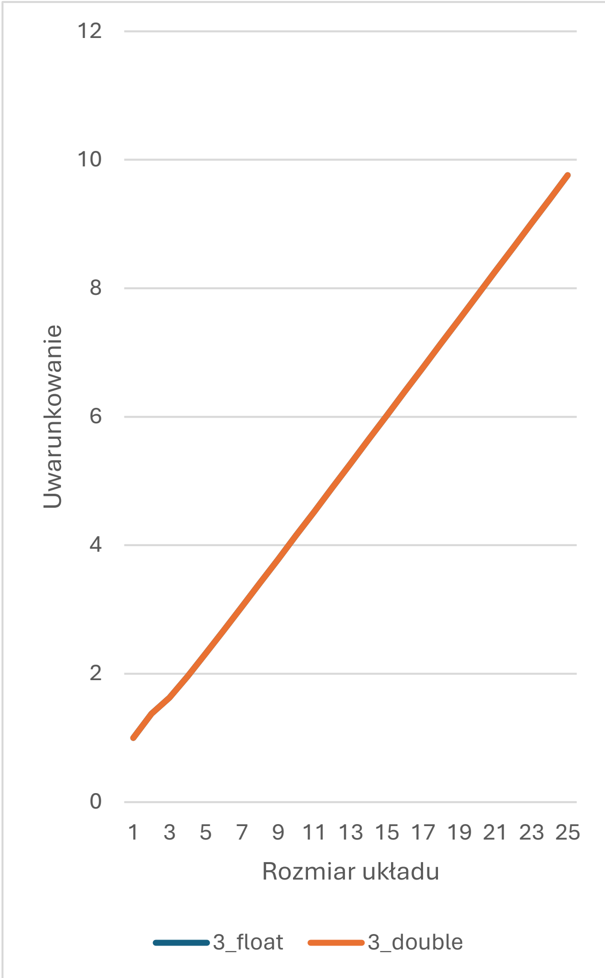
Wykres 6. Średnie błędy obliczeń w podpunkcie 3. dla typu double

Na wykresach 5. i 5. oraz tabeli 3. widać, że dla podpunktu 3. obliczenia są wyraźnie dokładniejsze od tych w podpunkcie 1. Występują różnice pomiędzy metodą Gaussa a algorytmem Thomasa. Metoda Gaussa jest nieco dokładniejsza od algorytmu Thomasa. Po przekroczeniu pewnego rozmiaru układu, błąd utrzymuje się na względnie stałym poziomie.

6.4 Uwarunkowanie w podpunkcie 3.

	float, 32-bit	double, 64-bit
n\punkt	3	3
1	1	1
2	1.375	1.375
3	1.625	1.625
4	1.958333373	1.958333333
5	2.3125	2.3125
6	2.674999952	2.675
7	3.041666746	3.041666667
8	3.410714388	3.410714286
9	3.78125	3.78125
10	4.152777672	4.152777778
11	4.525000095	4.525
12	4.897727013	4.897727273
13	5.270833015	5.270833333
14	5.644230843	5.644230769
15	6.017857075	6.017857143
16	6.391666889	6.391666667
17	6.765625	6.765625
18	7.139705658	7.139705882
19	7.513888836	7.513888889
20	7.888157845	7.888157895
21	8.262499809	8.2625
22	8.636904716	8.636904762
23	9.011363983	9.011363636
24	9.38586998	9.385869565
25	9.760416031	9.760416667

Tabela 4. Uwarunkowanie układów w podpunkcie 3.



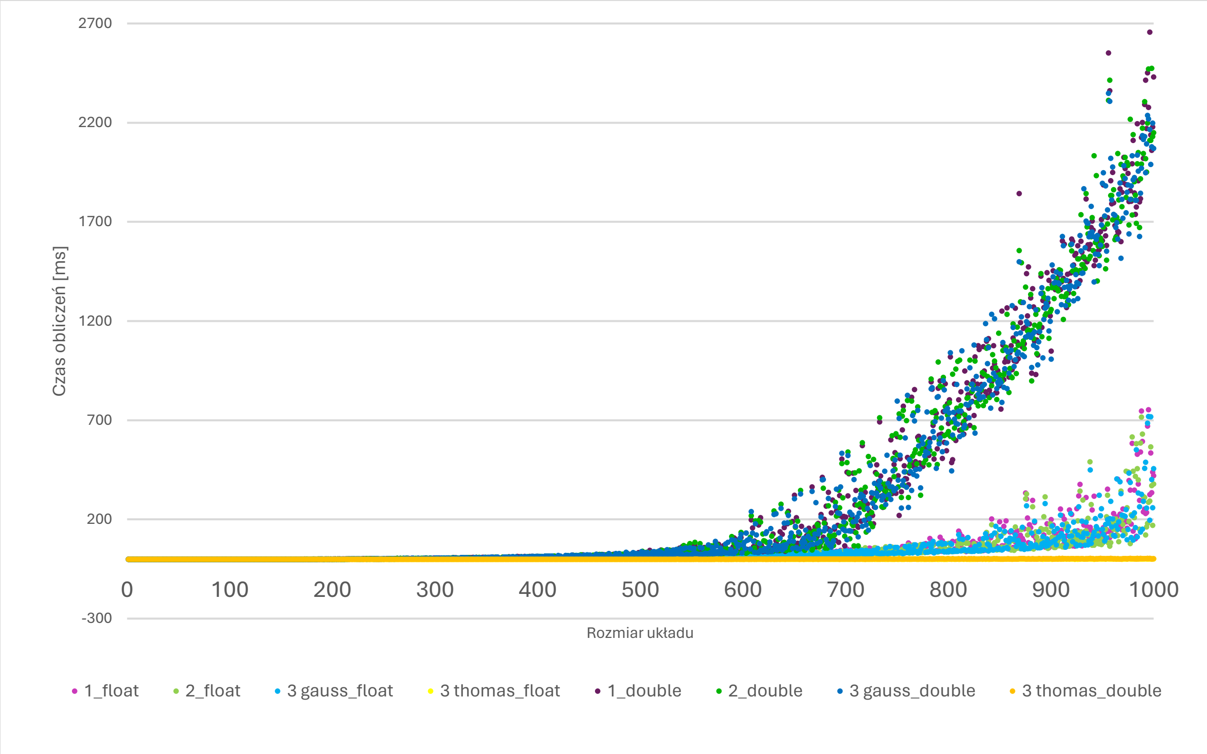
Wykres 7. Uwarunkowanie układów w podpunkcie 3.

Jak widać na tabeli 4. i wykresie 7. współczynnik uwarunkowania rośnie liniowo wraz z rozmiarem układu. Uwarunkowanie jest o rzędy wielkości mniejsze niż w podpunkcie 2. i 1. To tłumaczy niewielkie wartości błędów. Różnica pomiędzy float a double jest prawie żadna.

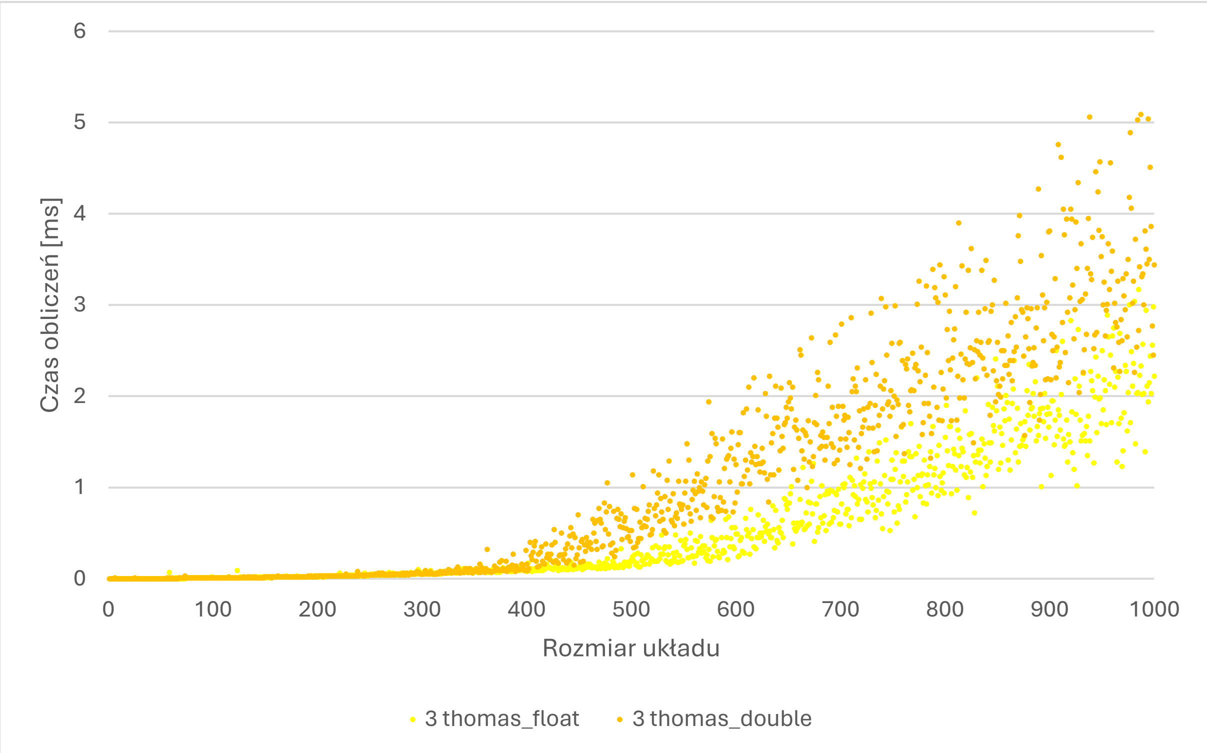
6.5 Porównanie czasów obliczeń

	float, 32-bit				double, 64-bit			
n\punkt	1	2	3 gauss	3 thomas	1	2	3 gauss	3 thomas
	czas obliczeń [ms]							
1	0	0	0	0	0	0	0	0
25	0.02	0.01	0.01	0	0.04	0.02	0.01	0.01
50	0.07	0.05	0.02	0	0.09	0.07	0.03	0
75	0.12	0.1	0.08	0.01	0.14	0.15	0.1	0.01
100	0.17	0.19	0.16	0.01	0.23	0.25	0.14	0.01
125	0.38	0.35	0.32	0.01	0.34	0.46	0.42	0.01
150	0.55	0.51	0.3	0.01	0.84	0.52	0.7	0.02
175	0.89	0.9	0.48	0.02	0.82	1.28	0.71	0.02
200	0.68	0.65	0.61	0.02	1.08	1.16	0.96	0.02
225	1.08	1.45	1.38	0.03	1.55	1.46	1.4	0.03
250	2.66	1.19	1.19	0.04	2.32	2.59	3.37	0.04
275	3.18	3.05	1.55	0.05	2.59	2.67	2.49	0.05
300	5.03	3.34	4.71	0.07	3.73	3.52	6.23	0.05
325	5.15	4.93	4.72	0.06	8.55	4.7	8.24	0.1
350	5.87	3.15	3.37	0.08	5.85	5.34	5.61	0.08
375	7.21	3.87	3.62	0.08	7.15	12.22	12.61	0.09
400	4.16	7.46	3.69	0.08	14.92	9.5	14.23	0.16
425	5.66	9.92	5.17	0.1	9.52	9.59	9.17	0.18
450	11.01	10.89	10.25	0.12	21.36	11.89	20.69	0.34
475	14.3	7.52	7.34	0.11	14.57	13.5	14.18	0.54
500	13.25	10.06	15.46	0.21	28.8	17.66	18.68	0.53
525	18.58	17.8	17.54	0.19	23.8	23.37	24.43	0.94
550	11.46	20.28	10.99	0.36	30.89	36.35	55.42	0.6
575	24.82	24.48	24.32	0.19	33.11	72.88	39.48	1.34
600	29.02	25.39	27.94	0.49	38.29	69.84	72.2	1.25
625	19.06	30.92	30.48	0.5	41.14	60.98	65.73	1.43
650	43.26	33.32	23.39	0.83	322.81	56.58	305.24	1.94
675	22.55	23.56	21.53	0.41	64.62	135.12	89.65	1.71
700	48.59	25.19	45.54	0.6	213.64	180.64	143.61	1.53
725	29.01	28.2	28.95	0.84	311.25	236.69	335.98	1.31
750	33.01	31.43	59.86	1.09	618.23	527.06	797.21	1.88
775	37.96	53.4	74.27	1.45	469.26	549.22	583.26	3.26
800	40.36	80.47	38.25	1.55	684.82	644.23	707.02	3.11
825	48.41	78.77	41.08	1.59	725.62	664.54	1080.69	3.62
850	189.58	48.46	150.41	2.11	875.15	931.5	904.9	2.59
875	335.21	306.26	58.09	1.55	1170.63	1372.46	1120.39	2.44
900	200.3	162.77	84.34	1.81	1049.6	1259.31	1010.12	3.81
925	255.46	100.98	68.44	1.8	1375.2	1443.07	1394.75	3.91
950	138.08	82.57	152.88	2.5	1607.43	1577.14	1895.26	3.75
975	188.73	441.26	151.12	2.04	1946.44	1986.25	1887.13	3.5
1000	421.37	379.04	457.07	2.22	2431.18	2150.62	2071.86	3.44

Tabela 5. Czasy obliczeń



Wykres 8. Czasy obliczeń



Wykres 9. Czasy obliczeń algorytmu Thomasa

Jak widać na tabeli 5. oraz wykresach 8 i 9. czas obliczeń rośnie wraz z rozmiarem układu. Poza nielicznymi odstępstwami, obliczenia metodą Gaussa zajmują podobną ilość czasu. Obliczenia na typie double zajmują więcej czasu niż na typie float, a różnica pomiędzy nimi zwiększa się wraz z rozmiarem układu.

7. Wnioski

Eksperymenty pokazały, że wyniki obliczeń na układach o dużej wartości wskaźnika uwarunkowania miały duże wartości średniego błędu, natomiast te dobrze uwarunkowane miały mniejsze, a czasami znikome błędy.

Innym istotnym czynnikiem wpływającym na dokładność obliczeń jest precyzja typu zmiennoprzecinkowego.

Porównanie czasów obliczeń metody Gaussa i algorytmu Thomasa pokazuje, że dobór uproszczonego algorytmu znacznie zmniejsza czas obliczeń, przy praktycznie żadnej stracie na dokładności.