

Rozwiązywanie układów równań liniowych metodami bezpośrednimi

Łukasz Wala

*AGH, Wydział Informatyki, Elektroniki i Telekomunikacji
Metody Obliczeniowe w Nauce i Technice 2021/2022*

Kraków, 25 maja 2022

1 Problem 1

1.1 Opis problemu

Dany jest układ równań liniowych $\mathbf{Ax}=\mathbf{b}$. Elementy macierzy \mathbf{A} o wymiarze $n \times n$ są określone wzorem:

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

Za wektor \mathbf{x} przyjęta zostanie dowolna n -elementowa permutacja ze zbioru $\{1, -1\}$ i obliczony zostanie wektor \mathbf{b} . Następnie metodą eliminacji Gaussa rozwiązany zostanie układ równań liniowych $\mathbf{Ax}=\mathbf{b}$ (przyjmując jako niewiadomą wektor \mathbf{x}). Przyjęte zostaną różne precyzje dla znanych wartości macierzy \mathbf{A} i wektora \mathbf{b} . Sprawdzane zostanie to, jak błędy zaokrągleń zaburzają rozwiązanie dla różnych rozmiarów układu (porównany \mathbf{x} obliczony z \mathbf{x} zadany). Eksperyment przeprowadzony zostanie dla różnych rozmiarów układu.

1.2 Opracowanie problemu

Program użyty do rozwiązania układu został napisany w języku Python z użyciem pakietu numpy.

Zakres użytego n wynosi 3-100, natomiast, w celu uzyskania różnych precyzji, zostaną użyte typy float128, float64 oraz float32 (128 bitów, 64 bity oraz 32 bitów, typy zmiennoprzecinkowe). Przyjęty wektor \mathbf{x} składa się naprzemiennie z 1 oraz -1, czyli $\mathbf{x} = [1, -1, 1, -1, \dots]$.

Poniżej wyniki dla wartości 3-18 n z użyciem float64:

n	Wynik (float64)
3	[1. -1. 1.]
4	[1. -1. 1. -1.]
5	[1. -1. 1. -1. 1.]
6	[1. -1. 1. -1. 1. -1.]
7	[1. -1. 1. -0.99999999 0.99999999 -0.99999999 1.]
8	[1. -1. 1.00000001 -1.00000003 1.00000007 -1.00000008 1.00000005 -1.00000001]
9	[1. -1. 1.00000002 -1.00000008 1.00000022 -1.00000034 1.00000031 -1.00000015 1.00000003]
10	[1. -0.99999992 0.99999887 -0.999992 0.99996863 -0.99992652 0.99989471 -0.99990954 0.9999572 -0.99999143]
11	[0.99999997 -0.99999848 0.99997297 -0.99976817 0.99887436 -0.99664975 0.99366282 -0.99235204 0.99429537 -0.99760317 0.99956614]
12	[0.99999941 -0.99996221 0.99919827 -0.99172934 0.95111984 -0.81988143 0.56753005 -0.31190968 0.28013607 -0.52390499 0.81951835 -0.97011435]
13	[0.99999778 -0.99983381 0.99588177 -0.95002628 0.6492147 0.55558381 -3.57758546 8.16116812 -11.53222833 10.53875693 -5.83889397 1.35691714 0.6410476]
14	[0.99999839 -0.99987809 0.99693457 -0.96208463 0.72726558 0.24775735 -2.82147498 7.05683998 -10.81750725 10.99258477 -7.21472848 2.59779619 0.10075086 -0.90425428]
15	[1.00000032 -1.00002608 1.00068109 -1.0084821 1.05859506 -1.23642266 1.52425818 -1.31770342 -0.63773969 4.60311654 -8.06911186 7.84695378 -4.30073459 0.80508738 0.73152805]
16	[1.00000035 -1.00002873 1.00077369 -1.0099144 1.07044155 -1.2924074 1.66629056 -1.39635283 -1.36182391 7.41476733 -13.5730076 14.63721333 -9.78616429 3.63410525 -0.11663269 -0.88726018]
17	[0.99999998 -1.00000181 1.00012819 -1.00256848 1.02437038 -1.12426363 1.32037494 -1.09632243 -1.12225943 6.55223303 -12.60591686 13.4139281 -7.23211203 -0.09782497 2.92766818 -2.18587769 1.22844453]
18	[1.00000009 -1.00000875 1.00027248 -1.00388918 1.029806 -1.12729424 1.25609572 -0.81278282 -1.66686582 6.95901598 -12.29735709 12.50041866 -6.48195112 -0.38410709 3.12164386 -2.45497509 1.40118285 -1.03920443]

Tabela 1: Wyniki dla wartości 3-18 n

Już dla $n = 12$ widać istotne błędy, które wraz ze wzrastającym b będą stały się coraz bardziej poważne. Poniżej tabele błędów dla wszystkich badanych wartości i precyzji (błąd - norma euklidesowa wektora prawdziwych wyników i tego uzyskanego za pomocą eliminacji Gaussa):

n	float128	float64	float32	n	float128	float64	float32
3	0.00000e+00	0.00000e+00	0.00000e+00	52	1.16113e-07	1.61072e+04	9.53346e+02
4	6.64652e-15	3.01871e-13	0.00000e+00	53	8.12417e-08	2.58046e+02	1.80353e+01
5	7.28264e-13	9.22938e-12	4.06851e-12	54	1.35778e-07	5.96323e+02	1.99145e+01
6	5.44100e-11	3.63798e-10	1.08591e-11	55	9.55140e-08	5.90284e+02	1.11223e+01
7	3.69517e-09	1.36093e-08	1.80863e-09	56	1.02527e-07	1.24163e+02	1.10942e+02
8	6.74760e-09	1.20335e-07	6.03675e-09	57	1.31040e-07	1.01673e+02	1.37209e+01
9	4.03806e-08	5.40051e-07	1.62591e-06	58	1.07265e-07	1.20849e+03	5.19018e+01
10	6.73052e-09	1.66203e-04	2.23585e-05	59	3.79130e-08	9.46613e+02	4.18553e+01
11	8.55678e-09	1.22341e-02	1.51898e-03	60	3.90234e-08	2.75634e+02	2.14181e+01
12	1.10052e-08	1.21398e+00	5.23169e-02	61	9.38645e-08	2.68919e+02	6.71575e+01
13	2.39652e-08	2.12151e+01	5.62548e-01	62	1.34922e-07	1.09973e+03	8.14264e+01
14	1.81406e-08	2.11156e+01	3.72763e-01	63	8.84059e-07	4.24254e+03	3.31771e+02
15	2.39101e-08	1.50485e+01	6.50496e-01	64	6.89870e-08	3.73795e+02	2.33192e+02
16	3.24268e-08	2.59432e+01	1.17698e+00	65	6.60796e-08	3.79244e+02	8.67555e+01
17	7.13559e-08	2.29849e+01	2.33258e+00	66	7.23018e-08	6.00618e+02	1.01882e+02
18	2.67034e-08	2.21990e+01	1.50270e+00	67	1.03184e-07	6.39984e+02	5.68154e+01
19	3.79751e-08	8.53660e+01	2.24009e+01	68	7.58521e-08	5.05802e+02	5.60500e+01
20	1.77932e-08	8.71401e+02	8.78090e+00	69	7.60955e-08	1.28675e+03	1.43560e+02
21	2.25166e-08	5.86659e+01	8.24051e+00	70	8.48296e-08	9.68115e+02	6.86839e+01
22	1.56222e-08	5.29518e+01	4.86432e+00	71	7.66839e-08	2.09254e+02	3.09189e+01
23	4.74004e-08	4.70098e+01	4.75505e+00	72	2.93301e-07	4.05753e+02	4.63669e+01
24	1.85685e-07	1.23774e+02	9.36016e+00	73	8.10593e-08	4.11489e+02	3.42964e+01
25	1.83021e-07	2.69255e+02	1.50937e+01	74	8.26313e-08	2.38284e+02	8.09815e+01
26	1.24864e-07	1.11393e+02	1.35080e+01	75	8.04852e-08	7.10297e+02	1.62780e+02
27	1.23479e-07	1.24288e+02	3.16106e+01	76	9.18682e-08	4.12513e+02	2.99406e+01
28	1.80821e-06	1.57776e+03	7.67399e+01	77	1.08387e-07	1.87440e+02	2.79378e+01
29	9.48392e-08	4.02865e+02	2.22652e+01	78	1.03110e-07	2.03857e+02	2.57488e+01
30	1.57316e-08	1.89246e+02	6.82841e+00	79	1.37492e-07	6.78755e+02	3.58451e+01
31	1.17393e-08	2.24508e+02	3.83671e+00	80	9.16911e-08	5.08628e+02	3.48783e+01
32	1.48993e-08	6.72210e+01	6.23303e+00	81	5.00448e-07	6.06770e+02	4.10914e+01
33	4.08596e-08	7.39117e+01	1.14283e+01	82	2.15135e-07	7.73650e+02	5.92223e+01
34	5.13099e-08	5.53441e+02	4.80933e+01	83	8.16408e-07	1.17550e+03	1.05222e+02
35	2.78104e-08	5.19503e+02	4.10183e+01	84	8.50474e-08	1.16940e+04	3.85955e+02
36	4.51019e-08	3.53450e+02	3.31293e+01	85	1.25238e-06	1.03478e+04	1.20177e+02
37	2.31974e-08	3.91273e+02	1.13820e+01	86	2.46212e-07	1.24309e+04	2.93697e+02
38	5.33618e-07	3.77185e+02	2.47144e+01	87	1.34919e-07	1.50364e+04	1.45779e+02
39	4.03595e-08	1.87523e+02	1.70904e+01	88	1.77220e-07	6.09119e+04	7.15888e+01
40	1.00401e-07	2.70230e+02	1.74852e+01	89	2.10088e-07	1.41214e+03	1.00808e+02
41	6.78265e-08	5.55432e+02	3.25176e+01	90	1.11207e-07	1.76003e+03	6.43347e+01
42	3.89531e-08	3.19532e+02	1.45562e+01	91	1.40491e-07	9.43939e+02	4.96421e+01
43	7.07625e-08	3.53136e+02	1.21748e+01	92	3.61188e-07	1.68947e+03	5.39511e+01
44	4.28090e-08	3.75632e+02	1.37852e+01	93	7.47588e-07	1.23000e+03	3.33221e+01
45	4.61908e-08	4.01859e+02	9.50281e+00	94	8.67451e-07	1.16441e+03	4.16179e+01
46	4.47556e-08	1.08220e+03	2.68951e+01	95	1.40393e-06	1.11327e+03	4.31873e+01
47	1.06853e-07	4.67922e+03	4.85818e+01	96	1.02786e-06	1.93032e+04	5.98646e+02
48	1.01608e-07	1.75233e+03	9.09009e+00	97	1.94037e-07	9.62075e+02	3.08990e+01
49	5.73074e-07	5.92046e+03	1.69193e+02	98	2.01763e-07	1.33434e+03	6.69423e+01
50	1.47107e-07	2.46086e+02	7.41959e+00	99	9.39441e-06	1.52907e+03	9.06394e+01
51	9.47365e-08	8.57508e+02	3.73456e+01	100	2.77768e-06	3.77706e+03	1.63594e+02

Tabela 2: Błędy

Błędy powiększają się wraz ze wzrastającym rozmiarem układu jak również z malejącą precyzją użytego typu, przy czym należy zauważyć, że błąd wynikający z precyzji jest znacznie większy dla typów float32 oraz float 64 niż dla float128. Następnym krokiem będzie porównanie wyników w problemie drugim z tymi uzyskanymi tutaj.

2 Problem 2

2.1 Opis problemu

Eksperyment analogiczny do problemu 1 zostanie przeprowadzony dla macierzy zadanej wzorem:

$$\begin{cases} a_{1j} = \frac{2i}{j} & \text{dla } j \geq i \\ a_{ij} = a_{ij} & \text{dla } j \leq i \end{cases} \quad i, j = 1, \dots, n$$

Wyniki zostaną porównane z tymi w problemie 1. Zostanie sprawdzone uwarunkowanie obu układów.

2.2 Opracowanie problemu

Zakres badanych n , precyzji oraz użyty wektor \mathbf{x} są identyczne jak w problemie 1.

n	Wynik (float64)
3	[1. -1. 1.]
4	[1. -1. 1. -1.]
5	[1. -1. 1. -1. 1.]
6	[1. -1. 1. -1. 1. -1.]
7	[1. -1. 1. -1. 1. -1. 1.]
8	[1. -1. 1. -1. 1. -1. 1. -1.]
9	[1. -1. 1. -1. 1. -1. 1. -1. 1.]
10	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]
11	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]
12	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]
13	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]
14	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]
15	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]
16	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]
17	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]
18	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]

Tabela 3: Wyniki dla wartości 3-18 n

Wszystkie uzyskane wyniki w tym zakresie są na tyle dobre (tzn. mają na tyle dużo zer po przecinku), że numpy nie pokazuje ich rozwinięcia dziesiętnego, w odróżnieniu od problemu 1, gdzie widać było ewidentne niedokładności.

n	float128	float64	float32
3	0.00000e+00	3.14018e-16	3.14018e-16
4	0.00000e+00	2.48253e-16	5.66105e-16
5	2.48253e-16	4.15407e-16	4.96507e-16
6	3.14018e-16	9.74217e-16	6.47366e-16
7	2.48253e-16	1.69468e-15	7.77156e-16
8	6.08094e-16	4.67218e-15	2.29416e-15
9	1.48122e-15	3.31025e-15	1.42611e-15
10	2.99143e-15	3.08274e-15	5.30473e-15
11	2.98318e-15	4.42142e-15	6.66689e-15
12	3.63842e-15	1.98040e-14	1.08353e-14
13	4.76491e-15	2.20079e-14	9.17665e-15
14	4.02599e-15	2.27677e-14	1.17500e-14
15	3.61292e-15	2.83611e-14	1.40438e-14
16	3.52485e-15	3.80102e-14	1.42723e-14
17	1.00449e-14	3.71618e-14	1.41234e-14
18	1.00792e-14	3.65198e-14	1.23634e-14
19	1.52780e-14	3.88935e-14	1.32065e-14
20	1.97189e-14	3.80900e-14	1.55973e-14
21	2.00517e-14	3.65030e-14	1.69886e-14
22	2.61846e-14	4.75961e-14	1.57937e-14
23	2.92058e-14	4.25638e-14	1.89406e-14
24	3.11027e-14	3.68189e-14	2.35490e-14
25	4.01138e-14	3.90851e-14	2.60590e-14
26	4.78317e-14	3.97499e-14	3.78837e-14
27	4.94930e-14	4.30520e-14	4.32887e-14
28	5.64670e-14	1.01729e-13	4.83767e-14
29	6.79743e-14	1.22073e-13	6.59737e-14
30	7.74665e-14	9.93568e-14	6.37943e-14
31	7.20535e-14	1.23333e-13	8.47986e-14
32	7.49745e-14	1.22503e-13	1.03770e-13
33	8.34764e-14	1.20980e-13	1.00354e-13
34	9.65992e-14	1.27744e-13	8.96049e-14
35	1.03406e-13	1.25167e-13	8.98755e-14
36	1.05284e-13	1.81115e-13	9.92374e-14
37	9.74833e-14	1.78416e-13	8.58149e-14
38	8.99382e-14	1.91015e-13	8.43823e-14
39	9.39602e-14	1.75834e-13	8.63363e-14
40	1.06354e-13	2.56295e-13	9.07217e-14
41	9.88085e-14	2.55501e-13	8.50089e-14
42	1.18753e-13	2.56017e-13	9.76102e-14
43	1.23169e-13	2.59007e-13	9.94367e-14
44	1.65062e-13	2.20919e-13	1.02333e-13
45	1.80798e-13	2.21368e-13	1.19101e-13
46	1.42598e-13	2.43778e-13	1.46323e-13
47	1.37076e-13	3.02347e-13	1.48848e-13
48	1.49469e-13	3.16843e-13	1.58636e-13
49	1.38619e-13	3.23690e-13	1.72794e-13
50	1.36274e-13	3.46057e-13	1.67102e-13
51	1.43897e-13	3.58384e-13	1.85238e-13

n	float128	float64	float32
52	1.39772e-13	3.67188e-13	2.47585e-13
53	1.65545e-13	3.71622e-13	2.44301e-13
54	1.83550e-13	3.95670e-13	2.80490e-13
55	1.50536e-13	4.47478e-13	3.02780e-13
56	1.47575e-13	5.07994e-13	3.29484e-13
57	1.63723e-13	5.18223e-13	3.51037e-13
58	1.77244e-13	5.38007e-13	3.61192e-13
59	1.88617e-13	5.32264e-13	3.53027e-13
60	2.01146e-13	8.55487e-13	3.90212e-13
61	1.81530e-13	8.64971e-13	3.94910e-13
62	1.87015e-13	8.52088e-13	4.01676e-13
63	1.92385e-13	8.53760e-13	4.09343e-13
64	2.34847e-13	1.02174e-12	3.75456e-13
65	2.29063e-13	1.08640e-12	5.19077e-13
66	2.41997e-13	1.20444e-12	5.83299e-13
67	2.62325e-13	1.27659e-12	6.21968e-13
68	2.83034e-13	8.29059e-13	6.88447e-13
69	2.75237e-13	8.79906e-13	7.48081e-13
70	3.04656e-13	9.25090e-13	8.04045e-13
71	3.06542e-13	9.76631e-13	8.66095e-13
72	3.02185e-13	1.20634e-12	8.94100e-13
73	3.29546e-13	1.20252e-12	9.29830e-13
74	3.27333e-13	1.20104e-12	9.24748e-13
75	3.19322e-13	1.20343e-12	9.47747e-13
76	3.34962e-13	1.89808e-12	9.03211e-13
77	3.73973e-13	1.88248e-12	8.68586e-13
78	4.06460e-13	1.96682e-12	8.98199e-13
79	4.22540e-13	1.96288e-12	9.04314e-13
80	4.27885e-13	1.57121e-12	8.69675e-13
81	5.00671e-13	1.66451e-12	9.31137e-13
82	5.15757e-13	1.97529e-12	9.47249e-13
83	5.69078e-13	2.05867e-12	9.50470e-13
84	6.33495e-13	2.30233e-12	9.70425e-13
85	6.50080e-13	2.28787e-12	1.01391e-12
86	6.48275e-13	2.28022e-12	1.00284e-12
87	7.06903e-13	2.29377e-12	1.09560e-12
88	7.49646e-13	2.21554e-12	1.14357e-12
89	8.26162e-13	2.24017e-12	1.16569e-12
90	8.57090e-13	2.39984e-12	1.19415e-12
91	8.54227e-13	2.69002e-12	1.24766e-12
92	8.90434e-13	2.24959e-12	1.23540e-12
93	9.01505e-13	2.26938e-12	1.27993e-12
94	9.70675e-13	2.29189e-12	1.32937e-12
95	9.36551e-13	2.27211e-12	1.41987e-12
96	9.29838e-13	2.46800e-12	1.39482e-12
97	9.26189e-13	2.46843e-12	1.42324e-12
98	9.77432e-13	2.51404e-12	1.48752e-12
99	1.16504e-12	2.54730e-12	1.53327e-12
100	1.21042e-12	2.28769e-12	1.55604e-12

Tabela 4: Błędy

Nawet dla największej badanej wartości n , niezależnie od precyzji użytego typu, pojawiający się błąd jest rzędu 10^{-12} . Precyzja uzyskanych wyników w problemie 2 w takich samych warunkach jest znacznie lepsza niż w problemie 1.

Tutaj należałoby się zastanowić nad powodem powyższej różnicy wyników pomiędzy macierzami w problemie pierwszym i drugim. Poniżej części macierzy \mathbf{A} dla $n = 100$ dla obu problemów:

$$A_1 = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ 0.5 & 0.33333333 & 0.25 & \dots & 0.01010101 & .01 & 0.00990099 \\ 0.33333333 & 0.25 & 0.2 & \dots & 0.01 & 0.00990099 & 0.00980392 \\ \vdots & & & \ddots & & & \vdots \\ 0.01020408 & 0.01010101 & 0.01 & \dots & 0.00512821 & 0.00510204 & 0.00507614 \\ 0.01010101 & 0.01 & 0.00990099 & \dots & 0.00510204 & 0.00507614 & 0.00505051 \\ 0.01 & 0.00990099 & 0.00980392 & \dots & 0.00507614 & 0.00505051 & 0.00502513 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 2 & 1. & 0.66666667 & \dots & 0.02040816 & 0.02020202 & 0.02 \\ 1. & 2. & 1.33333333 & \dots & 0.04081633 & 0.04040404 & 0.04 \\ 0.66666667 & 1.33333333 & 2. & \dots & 0.06122449 & 0.06060606 & 0.06 \\ \vdots & & & \ddots & & & \vdots \\ 0.02040816 & 0.04081633 & 0.06122449 & \dots & 2. & 1.97979798 & 1.96 \\ 0.02020202 & 0.04040404 & 0.06060606 & \dots & 1.97979798 & 2. & 1.98 \\ 0.02 & 0.04 & 0.06 & \dots & 1.96 & 1.98 & 2. \end{bmatrix}$$

Jedną z przyczyn, dla których układ równań z problemu pierwszego daje tak złe wyniki jest metoda wybierania elementu wiodącego (pivotu). W tej implementacji algorytmu eliminacja Gaussa jako pivot wybierane są zawsze kolejne elementy na przekątnej macierzy. W przypadku macierzy z problemu drugiego jest to zawsze 2 (po kolejnych redukcjach pivot osiąga wartości rzędu 10^{-2}), w przypadku macierzy z problemu pierwszego elementy na przekątnej redukują się na tyle mocno, że osiągają wartości rzędu nawet 10^{-18} . Stanowi to problem, ponieważ poszczególne wiersze w metodzie eliminacji Gaussa są dzielone przez pivot, to znaczy mnożone przez jego odwrotność. Jeżeli pivot jest mały (< 1), wówczas wiersze mnożone są przez dużą wartość, więc błędy zaokrągleń stają się duże w stosunku do współczynników oryginalnej macierzy.

Oprócz tego można również obliczyć wskaźnik uwarunkowania macierzy κ . Wartość ta jest miarą tego, jak bardzo zmieni się rozwiązanie \mathbf{x} układu równań w stosunku do zmiany \mathbf{b} . Jeżeli wskaźnik uwarunkowania macierzy jest duży, nawet mały błąd w \mathbf{b} może powodować duże błędy w \mathbf{x} .

$$\kappa = \|A^{-1}\| \|A\|$$

Można zastosować dowolną normę zwartą, tutaj zostanie użyta norma “nieskończoność”, tj. $\|A\|_\infty = \max_i \sum_j |a_{ij}|$.

Poniżej tabela z wybranymi wskaźnikami uwarunkowania macierzy dla różnych n :

n	κ_1 (problem 1)	κ_2 (problem 2)
3	8.64000e+02	8.66667e+00
6	5.63447e+07	3.96667e+01
9	2.84317e+12	9.23810e+01
12	1.36454e+17	1.66822e+02
15	1.04335e+19	2.63294e+02
18	2.21394e+21	3.81736e+02
21	1.89813e+19	5.21917e+02
24	1.51520e+19	6.83833e+02
27	2.25108e+20	8.67483e+02
30	5.08850e+19	1.07287e+03
33	6.34109e+19	1.30055e+03
36	3.60448e+20	1.54997e+03
39	2.50037e+20	1.82111e+03
42	6.68682e+19	2.11399e+03
45	3.72911e+21	2.42860e+03
48	3.75331e+20	2.76521e+03
51	1.67838e+21	3.12386e+03
54	4.26339e+20	3.50424e+03
57	6.19806e+20	3.90635e+03
60	9.06089e+20	4.33019e+03
63	2.47455e+20	4.77576e+03
66	2.37977e+20	5.24360e+03
69	1.79781e+21	5.73321e+03
72	7.20453e+20	6.24455e+03
75	5.53474e+21	6.77762e+03
78	1.18038e+21	7.33242e+03
81	8.58646e+21	7.90918e+03
84	4.81463e+20	8.50802e+03
87	1.01666e+21	9.12860e+03
90	9.19833e+20	9.77090e+03
93	5.60807e+20	1.04349e+04
96	1.08158e+21	1.11207e+04
99	2.33922e+21	1.18287e+04

Tabela 5: Wskaźniki uwarunkowania macierzy

Wskaźniki uwarunkowania dla macierzy z problemu 1 są znacznie większe niż dla problemu 2. Oznacza to, że niewielki błąd (wynikający np. z przybliżeń) znacznie wpływa na wynik.

3 Problem 3

Ekspetyment z dwóch poprzednich problemów zostanie powtórzony dla macierzy zadanej wzorem:

$$\begin{cases} a_{ii} = k \\ a_{i,i+1} = \frac{1}{i+m} \\ a_{i,i-1} = \frac{k}{i+m+1} \text{ dla } i > j \\ a_{i,j} = 0 \text{ dla } j < i-1 \text{ oraz } j > i+1 \end{cases} \quad i, j = 1, \dots, n$$

Gdzie $k = 6$ oraz $m = 5$.

Następnie układ rozwiązany zostanie metodą przeznaczoną do rozwiązywania układów z macierzą trójdziagonalną. Te dwie metody zostaną porównane (czas, dokładność obliczeń i zajętość pamięci) dla różnych rozmiarów układu (z pominięciem czasu tworzenia układu). Opisane zostanie to, jak w metodzie dla układów z macierzą trójdziagonalną przechowywano i wykorzystano macierz **A**.

3.1 Opracowanie problemu

Zastosowane parametry są takie same jak w poprzednich problemach, $n \in \{3, \dots, 100\}$, 3 rodzaje precyzji floata oraz $\mathbf{x} = [1, -1, 1, -1, \dots]$. Zastosowaną rutyną do rozwiązywania układów z macierzą trójdziagonalną będzie algorytm Thomasa. Algorytm ten działa analogicznie do eliminacji Gaussa, jednak wykorzystuje fakt, że poza trzema środkowymi przekątnymi macierzy są w niej same zera, tak więc tam, gdzie metoda Gaussa itarowałaby przez wszystkie wiersze, odejmując wiersz z obecnym pivotem, algorytm Thomasa działa tylko na wierszu z pivotem oraz kolejnym (bo wszystkie inne zawierają same zera). Algorytm ten przeważa nad eliminacją Gaussa w kilku aspektach: jego złożoność obliczeniowa to $O(n)$ (eliminacja Gaussa - $O(n^3)$), złożoność pamięciowa (jeżeli założyć, że macierze wejściowe muszą pozostać niemodyfikowane) to $O(n)$ (eliminacja Gaussa - $O(n^2)$, trzeba skopiować całą macierz).

Poniżej kilka wyników dla początkowych n :

n	Wynik (Gaussian elimination)	Wynik (Thomas)
3	[1. -1. 1.]	[1. -1. 1.]
4	[1. -1. 1. -1.]	[1. -1. 1. -1.]
5	[1. -1. 1. -1. 1.]	[1. -1. 1. -1. 1.]
6	[1. -1. 1. -1. 1. -1.]	[1. -1. 1. -1. 1. -1.]
7	[1. -1. 1. -1. 1. -1. 1.]	[1. -1. 1. -1. 1. -1. 1.]
8	[1. -1. 1. -1. 1. -1. 1. -1.]	[1. -1. 1. -1. 1. -1. 1. -1.]
9	[1. -1. 1. -1. 1. -1. 1. -1. 1.]	[1. -1. 1. -1. 1. -1. 1. -1. 1.]
10	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]

Tabela 6: Wyniki (float64)

Dla pierwszych n wyniki algorytmu Thomasa tak samo dokładne, co jest zgodne z intuicją, bo algorytm Thomasa to uproszczona wersja eliminacji Gaussa. Poniżej prównanie dokładności dla wszystkich badanych n (tylko z wykorzystaniem float64, wpływ precyzji typu został zbadany już wcześniej).

n	Gauss	Thomas	n	Gauss	Thomas
3	1.11022e-16	1.11022e-16	52	6.37775e-16	6.37775e-16
4	3.14018e-16	3.14018e-16	53	6.37775e-16	6.37775e-16
5	3.14018e-16	3.14018e-16	54	6.75322e-16	6.75322e-16
6	3.14018e-16	3.14018e-16	55	7.10890e-16	7.10890e-16
7	3.14018e-16	3.14018e-16	56	7.10890e-16	7.10890e-16
8	2.93737e-16	2.93737e-16	57	7.10890e-16	7.10890e-16
9	2.93737e-16	2.93737e-16	58	7.10890e-16	7.10890e-16
10	2.93737e-16	2.93737e-16	59	7.10890e-16	7.10890e-16
11	2.93737e-16	2.93737e-16	60	7.10890e-16	7.10890e-16
12	3.68219e-16	3.68219e-16	61	7.10890e-16	7.10890e-16
13	3.68219e-16	3.68219e-16	62	7.44760e-16	7.44760e-16
14	3.68219e-16	3.68219e-16	63	7.52990e-16	7.52990e-16
15	3.68219e-16	3.68219e-16	64	7.52990e-16	7.52990e-16
16	3.84593e-16	3.84593e-16	65	7.52990e-16	7.52990e-16
17	4.00297e-16	4.00297e-16	66	7.52990e-16	7.52990e-16
18	4.00297e-16	4.00297e-16	67	7.52990e-16	7.52990e-16
19	4.00297e-16	4.00297e-16	68	8.15844e-16	8.15844e-16
20	4.57757e-16	4.57757e-16	69	8.15844e-16	8.15844e-16
21	4.57757e-16	4.57757e-16	70	8.45521e-16	8.45521e-16
22	4.71028e-16	4.71028e-16	71	8.45521e-16	8.45521e-16
23	4.71028e-16	4.71028e-16	72	8.45521e-16	8.45521e-16
24	4.71028e-16	4.71028e-16	73	8.45521e-16	8.45521e-16
25	4.71028e-16	4.71028e-16	74	8.45521e-16	8.45521e-16
26	4.71028e-16	4.71028e-16	75	8.45521e-16	8.45521e-16
27	4.71028e-16	4.71028e-16	76	8.74190e-16	8.74190e-16
28	4.71028e-16	4.71028e-16	77	8.74190e-16	8.74190e-16
29	4.83935e-16	4.83935e-16	78	8.74190e-16	8.74190e-16
30	4.83935e-16	4.83935e-16	79	8.74190e-16	8.74190e-16
31	4.83935e-16	4.83935e-16	80	8.74190e-16	8.74190e-16
32	4.83935e-16	4.83935e-16	81	9.01949e-16	9.01949e-16
33	4.96507e-16	4.96507e-16	82	9.01949e-16	9.01949e-16
34	4.96507e-16	4.96507e-16	83	9.01949e-16	9.01949e-16
35	4.96507e-16	4.96507e-16	84	9.01949e-16	9.01949e-16
36	5.43896e-16	5.43896e-16	85	9.01949e-16	9.01949e-16
37	5.43896e-16	5.43896e-16	86	9.01949e-16	9.01949e-16
38	5.43896e-16	5.43896e-16	87	9.01949e-16	9.01949e-16
39	5.43896e-16	5.43896e-16	88	9.01949e-16	9.01949e-16
40	5.43896e-16	5.43896e-16	89	9.08757e-16	9.08757e-16
41	5.43896e-16	5.43896e-16	90	9.08757e-16	9.08757e-16
42	5.43896e-16	5.43896e-16	91	9.08757e-16	9.08757e-16
43	5.87475e-16	5.87475e-16	92	9.08757e-16	9.08757e-16
44	5.55112e-16	5.55112e-16	93	9.08757e-16	9.08757e-16
45	5.97873e-16	5.97873e-16	94	9.35491e-16	9.35491e-16
46	6.37775e-16	6.37775e-16	95	9.35491e-16	9.35491e-16
47	6.37775e-16	6.37775e-16	96	9.35491e-16	9.35491e-16
48	5.97873e-16	5.97873e-16	97	9.35491e-16	9.35491e-16
49	5.97873e-16	5.97873e-16	98	9.35491e-16	9.35491e-16
50	5.97873e-16	5.97873e-16	99	9.35491e-16	9.35491e-16
51	5.97873e-16	5.97873e-16	100	9.35491e-16	9.35491e-16

Tabela 7: Błędy dla metod Gaussa i Thomasa

Błędy są dokładnie takie same niezależnie od metody, jak zostało wspomniane wcześniej, jest przewidywalny wynikiem, bo metody Gaussa i Thomasa są u podstaw takie same, tylko metoda Thomasa ogranicza się do działania na elementach trójdziagonalnych.

Poniżej porównanie czasów działania:

n	Gauss	Thomas
3	0.0000619888	0.0000097752
4	0.0000605583	0.0000104904
5	0.0000586510	0.0000112057
6	0.0001332760	0.0000202656
7	0.0001723766	0.0000269413
8	0.0001933575	0.0000269413
9	0.0001623631	0.0000243187
10	0.0002050400	0.0000312328
11	0.0002467632	0.0000228882
12	0.0002651215	0.0000255108
13	0.0002963543	0.0000422001
14	0.0003426075	0.0000267029
15	0.0005545616	0.0000529289
16	0.0005488396	0.0000472069
17	0.0006279945	0.0000414848
18	0.0007517338	0.0000555515
19	0.0008482933	0.0000672340
20	0.0009593964	0.0000400543
21	0.0007262230	0.0000364780
22	0.0006790161	0.0000369549
23	0.0007746220	0.0000381470
24	0.0010666847	0.0000436306
25	0.0009360313	0.0000662804
26	0.0011546612	0.0000457764
27	0.0009803772	0.0000619888
28	0.0010209084	0.0000708103
29	0.0010361671	0.0000452995
30	0.0010993481	0.0000452995
31	0.0012125969	0.0000746250
32	0.0022766590	0.0000927448
33	0.0023801327	0.0000879765
34	0.0023727417	0.0000596046
35	0.0015447140	0.0000562668
36	0.0016992092	0.0000882149
37	0.0015561581	0.0000529289
38	0.0019032955	0.0000991821
39	0.0028629303	0.0000839233
40	0.0019550323	0.0000615120
41	0.0019819736	0.0000629425
42	0.0020363331	0.0000622272
43	0.0040171146	0.0001215935
44	0.0043051243	0.0000996590
45	0.0022635460	0.0000650883
46	0.0023288727	0.0000636578
47	0.0029795170	0.0000715256
48	0.0026295185	0.0000736713
49	0.0038347244	0.0001218319
50	0.0029559135	0.0000746250
51	0.0035314560	0.0000882149

n	Gauss	Thomas
52	0.0032415390	0.0000820160
53	0.0053093433	0.0000941753
54	0.0035138130	0.0000839233
55	0.0038602352	0.0000853539
56	0.0055491924	0.0001106262
57	0.0039956570	0.0000896454
58	0.0039348602	0.0000896454
59	0.0063655376	0.0001385212
60	0.0067534447	0.0001616478
61	0.0047607422	0.0000948906
62	0.0045101643	0.0000922680
63	0.0043694973	0.0001051426
64	0.0059125423	0.0001597404
65	0.0049693584	0.0000951290
66	0.0047736168	0.0000939369
67	0.0064024925	0.0000991821
68	0.0050079823	0.0000927448
69	0.0053901672	0.0001285076
70	0.0065517426	0.0001037121
71	0.0056121349	0.0001022816
72	0.0068676472	0.0001111031
73	0.0059673786	0.0001046658
74	0.0073094368	0.0001091957
75	0.0062007904	0.0001070499
76	0.0088074207	0.0001599789
77	0.0066230297	0.0001106262
78	0.0078511238	0.0001115799
79	0.0068879128	0.0001113415
80	0.0087833405	0.0001146793
81	0.0073301792	0.0001149178
82	0.0085895061	0.0001168251
83	0.0080413818	0.0001251698
84	0.0079605579	0.0001177788
85	0.0092260838	0.0001583099
86	0.0082290173	0.0001211166
87	0.0094833374	0.0001237392
88	0.0088658333	0.0001304150
89	0.0087473392	0.0001268387
90	0.0106673241	0.0001313686
91	0.0092604160	0.0001437664
92	0.0093936920	0.0001323223
93	0.0108370781	0.0001347065
94	0.0098938942	0.0001597404
95	0.0105099678	0.0001373291
96	0.0181446075	0.0002424717
97	0.0181982517	0.0002415180
98	0.0185339451	0.0002431870
99	0.0193188190	0.0002481937
100	0.0204241276	0.0002508163

Tabela 8: Czas działania metod Gaussa i Thomasa

W metodzie Thomasa czasy działania są znacznie krótsze, co również nie zaskakuje, wykonuje ona bowiem znacznie mniej operacji, ograniczając się tylko do trójdziagonalności macierzy.

Algorytm Thomasa wygrywa również w kwestii zajętości pamięci, ponieważ, przy założeniu, że oryginalne macierze nie mogą być modyfikowane, wykorzystana tu implementacja używa dwóch dodatkowych tablic o długości n , natomiast eliminacja Gaussa potrzebuje dodatkowej macierzy o rozmiarze $n \times (n + 1)$.

3.2 Wnioski

Metoda Gaussa jest skutecznym i prostym sposobem na rozwiązywanie układu równań liniowych, jednak czasami cierpi z powodu błędów w dokładności spowodowanych słabym uwarunkowaniem układów wejściowych lub sposobu wybierania elementu wiodącego (czemu można zapobiec stosując inne elementy wiodące). Jeżeli macierz jest trójdziagonalna, warto zamiast eliminacji Gaussa wykorzystać algorytm Thomasa, który jest uproszczoną wersją eliminacji, daje wyniki o takiej samej dokładności, ale działa znacznie szybciej.