Rozwiązywanie układów równań liniowych metodami bezpośrednimi

Martyna Olszewska

Treść zadania

Elementy macierzy **A** o wymiarze *n x n* są określone wzorem:

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

Przyjmij wektor x jako dowolną n−elementową permutację ze zbioru { 1, -1 } i oblicz wektor b. Następnie metodą eliminacji Gaussa rozwiąż układ równań liniowych ***Ax=b*** (przyjmując jako niewiadomą wektor 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 x obliczony z x zadany). Przeprowadź eksperymenty dla różnych rozmiarów układu.
Następnie powtórzyć dla macierzy A danej wzorem:

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

SPECYFIKACJE

Do obliczeń użyłam języka python, na systemie operacyjnym Ubuntu 20.04.4 LTS. Procesor komputera to Intel Core i3-4030U CPU @ 1.90GHz × 4 , RAM: 8GB. Do generowania wykresów użyłam biblioteki matplotlib, a dokładniej narzędzia pyplot. Do wyznaczenia równoodległych punktów użyłam narzędzia linspace z biblioteki numpy. Korzystam również z biblioteki math.

WYNIKI

Aby uzyskać wyniki stworzyłam programy, które rozwiązywały równanie dla dwóch typów: float i Decimal, gdzie pierwsza z nich ma precyzje 18, zaś druga 28. Wartości wektora X są z zakresu {-1, 1} i są one w macierzy podniesione do odpowiedniej potęgi, która jest indeksem jej miejsca w tablicy. Następnie wykonywałam eksperymenty, gdzie w każdej iteracji obliczyłam wyniki dla $n = 2, \ldots, 20$, a wyniki zapisałam w tabeli. Obliczyłam błąd dla znalezionego rozwiązania dla każdej wartości niewiadomej jaka maksymalna bezwzględna różnica z obliczonej wartości i początkowej wektora X.

ZADANIE PIERWSZE

Poniższa tabela zawiera wyniki obliczonego błędu dla każdej z wielkosci macierzy A oraz dwóch typów- floata z mniejszą precyzją i Decimala z większą.
Wartość błędu to maximum z modułu różnicy X oczekiwanego z X obliczonym.

	Błąd	
Wielkość macierzy	**Float**	**Decimal**
2	0.0	0
3	0.0	5.4E-27
4	0.0	0.0
5	2.84150480e-12	1.386E-24
6	3.39519523e-11	8.9821E-23
7	2.01685093e-09	2.8816662E-21
8	5.35907513e-08	9.75087790E-20
9	1.80870425e-06	2.51341002E-18
10	1.07398276e-05	1.05700296E-17
11	0.00029340	1.55547965E-15
12	0.03773808	6.02947743E-14
13	0.40971738	2.09046449E-12
14	0.24474964	1.49092408E-10
15	0.72967091	3.09814676E-9
16	1.15102499	6.10628740E-8
17	1.09646785	0.00000106
18	1.87580086	0.00004057
19	11.8604430	0.00173038
20	47.7423561	0.05437183
50	2459.1006	3.751664665

Tabela 1. Obliczone błędy dla różnych wielkości macierzy A

Analizując powyższa tabele możemy zauważyć, że dla typu Decimal błędy są zdecydowanie mniejsze. Wynika to z tego, że dla typu z mniejszą precyzją błędy związane z arytmetyką komputera pojawiają się wcześniej zatem wyniki o wiele wcześniej będą odbiegały od satysfakcjonujących. Widać, że błędy się powiększają wraz ze wzrastającym rozmiarem układu oraz z malejącą precyzją użytego typu.

ZADANIE DRUGIE

Poniższa tabela zawiera wyniki obliczonego błędu dla każdej z wielkosci macierzy A oraz dwóch typów- floata z mniejszą precyzją i Decimala z większą.
Wartość błędu to maximum z modułu różnicy X oczekiwanego z X obliczonym.

	Błąd	
Wielkość macierzy	**Float**	**Decimal**
2	0.0	0
3	2.22044634e-16	0
4	2.22044604e-16	0
5	2.22044604e-16	0
6	2.22044604e-16	1,00E-27
7	3.33066907e-16	1,00E-27
8	1.88737918e-15	1,00E-27
9	1.554312239e-15	1.2E-27
10	4.440892098e-15	2.1E-27
11	6.217248939e-15	2.6E-27
12	6.217248937e-15	1.9E-27
13	4.884981308e-15	5.9E-27
14	7.771561176e-15	3,00E-27
15	8.659739592e-15	2.096E-25
16	7.993605777e-15	9.3E-27
17	7.771561172e-15	3.46E-26
18	7.993605777e-15	8.52E-26
19	7.549511064e-15	6.4E-26
20	7.993605727e-15	1.07E-25
50	8.815170894-14	1.41689E-23
200	1.64235292e-12	1.60918E-22
500	1.14537268e-11	6.47283E-22

Tabela 2. Obliczone błędy dla różnych wielkości macierzy A

Analizując powyższa tabele można zauważyć, że błędy są dużo mniejsze niż w poprzednim zadaniu. Dla typu Decimal są one praktycznie wszystkie rzędu -27, zatem jest to bardzo dobry wynik. Dla typu float wyniki są rzędu -15 zatem wciąż nie są najgorsze, ale i tak drugi typ radzi sobie lepiej.Jak widać błędy związane z arytmetyką komputera są o wiele mniejsze niż w poprzednim przypadku. Próbując obliczyć wynik dla obydwu typów przy macierzy wielkości 500 float daje błąd rzędu -11, a Decimal -22. Więc nawet przy takich dużych macierzach wyniki są wciąż dość satysfakcjonujące.

Widać, że prezycja uzyskanych wyników w problemie drugim w takich samych warunkach jest dużo większa niż w poprzednim. Może to wynikać z tego jak wyglądają macierze.
Macierz w pierwszym problemie na przekątnej ma coraz mniejsze wartości natomiast macierz w drugim problemie ma zawsze 2. Zatem jeżeli popatrzymy na implementację metody Gaussa widzimy, że zawsze dzielimy poszczególne wiersze przez tą wartość, więc w metodzie pierwszej ta wartość osiąga bardzo małe wartości z którymi komputer już sobie nie radzi - błędy zaokrągleń stają się bardzo duże.
Macierz A z przykładu drugiego dla n = 10:

```
[2.0, 1.0, 0.67, 0.5, 0.4, 0.33, 0.29, 0.25, 0.22, 0.2]
[1.0, 2.0, 1.33, 1.0, 0.8, 0.67, 0.57, 0.5, 0.44, 0.4]
[0.67, 1.33, 2.0, 1.5, 1.2, 1.0, 0.86, 0.75, 0.67, 0.6]
[0.5, 1.0, 1.5, 2.0, 1.6, 1.33, 1.14, 1.0, 0.89, 0.8]
[0.4, 0.8, 1.2, 1.6, 2.0, 1.67, 1.43, 1.25, 1.11, 1.0]
[0.33, 0.67, 1.0, 1.33, 1.67, 2.0, 1.71, 1.5, 1.33, 1.2]
[0.29, 0.57, 0.86, 1.14, 1.43, 1.71, 2.0, 1.75, 1.56, 1.4]
[0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 1.78, 1.6]
[0.22, 0.44, 0.67, 0.89, 1.11, 1.33, 1.56, 1.78, 2.0, 1.8]
[0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0]
```

Macierz A z przykładu pierwszego dla n = 10

```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[0.5, 0.33, 0.25, 0.2, 0.17, 0.14, 0.12, 0.11, 0.1, 0.09]
[0.33, 0.25, 0.2, 0.17, 0.14, 0.12, 0.11, 0.1, 0.09, 0.08]
[0.25, 0.2, 0.17, 0.14, 0.12, 0.11, 0.1, 0.09, 0.08, 0.08]
[0.2, 0.17, 0.14, 0.12, 0.11, 0.1, 0.09, 0.08, 0.08, 0.07]
[0.17, 0.14, 0.12, 0.11, 0.1, 0.09, 0.08, 0.08, 0.07, 0.07]
[0.14, 0.12, 0.11, 0.1, 0.09, 0.08, 0.08, 0.07, 0.07, 0.06]
[0.12, 0.11, 0.1, 0.09, 0.08, 0.08, 0.07, 0.07, 0.06, 0.06]
[0.11, 0.1, 0.09, 0.08, 0.08, 0.07, 0.07, 0.06, 0.06, 0.06]
[0.1, 0.09, 0.08, 0.08, 0.07, 0.07, 0.06, 0.06, 0.06, 0.05]
```

Można również obliczyć wskaźnik uwarunkowania macierzy dany wzorem:

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

Wartość ta pokazuje jak zmienia się rozwiązanie układu w stosunku do zmiany b. Im większy wskaźnik uwarunkowania tym możemy spodziewać się gorszych wyników.

Poniższa tabela przedstawia wartości wskaźnika uwarunkowania dla macierzy z obydwu zadań.

Wielkość macierzy	Zad 1	Zad 2
3	567.857	8.666
5	1525.975	26.800
10	19427.999	114.728
50	356606.430	3001.8149
60	588138.46	4330.188

Tabela 3. Wartości współczynnika uwarunkowania dla obydwu macierzy.

Jak widać wartości te są dużo większe dla macierzy z zadania pierwszego, zatem ten układ jest źle uwarunkowany oznacza to, że nawet mały błąd wynikający z przybliżeń znacznie wpływa na ostateczny wynik.

ZADANIE TRZECIE

Powtórzyć eksperyment dla macierzy:

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

Gdzie m = 4 i k = 4.
Następnie rozwiązać układ metodą przeznaczoną do rozwiązywania układów z macierzą trójdiagonalną.

WYNIKI

Wykonanie zadania jest analogiczne do poprzednich dwóch z tym, że rozwiązuje układ również metodą Thomasa. Metodę tą stosuje się do układów trójprzekątnych:

$$\begin{bmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & a_3 & b_3 & c_3 & & & \\ & & . & . & . & & \\ & & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & & a_n & b_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ . \\ d_{n-1} \\ d_n \end{bmatrix}$$

gdzie

$$a_1 = 0$$

$$c_n = 0$$

$$i = 1, 2, \cdots, n$$

Układ taki można przedstawić w postaci: $a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$

Do zalet tego algorytmu należą jego prostota w obliczeniach dzięki czemu można w łatwy sposób obliczyć równaniach posiadające dużą liczbę niewiadomych. Posiada małą złożoność obliczeniową oraz możliwość oszczędnego korzystania z pamięci operacyjnej komputera.

W programie macierz A jest przechowywana w postaci 4 Tablic jednowymiarowych a, b, c i d. Każda z nich jest wielkości n, czyli liczby niewiadomych. W metodzie Gaussa była to Tablica dwuwymiarowa wielkości n x n. Zatem metoda Thomasa jest rzeczywiście o wiele bardziej wydajniejsza pod względem pamięciowym niż Gaussa. Dla n = 10 przy metodzie Gaussa będziemy przetrzymywać 100 wartości, gdzie przy metodzie Thomasa tych wartości będzie już tylko 40, więc jest to różnica ponad połowy.
Algorytm ten działa analogicznie do Metody Gaussa, jednak on wykorzystuje fakt, że poza przekątną i dwoma wartościami wokół niej w macierzy są same zera. Więc tam gdzie Gauss przechodziłby po każdej wartości w macierzy w tym każdym z zer, które nie zmieniły by wyniki, algorytm Thomasa całkiem opuszcza te wartości. Jego złożoność obliczeniowa to O(n) gdzie metoda Gaussa to O(n^3).
W tym zadaniu również porównywałam czasy obliczeń dla poszczególnych metod.
Postanowiłam wykonywać eksperymenty na macierz rozmiaru od 100 do 200 oraz z typem float.

Poniższa tabela zawiera wartości błędu dla tych dwóch metod i dwóch typów dla których były obliczane wartości. Były one przybliżane do 18 miejsca po przecinku.

	Thomas		Gauss	
Wielkość macierzy	Float	Decimal	Float	Decimal
100	1.42611e-15	0	1.42611e-15	0
105	1.46869e-15	0	1.46869e-15	0
110	1.51821e-15	0	1.51821e-15	0
115	1.55827e-15	0	1.55827e-15	0
120	1.61269e-15	0	1.61269e-15	0
125	1.62412e-15	0	1.62412e-15	0
130	1.67272e-15	0	1.67272e-15	0
135	1.7271e-15	0	1.7271e-15	0
140	1.78673e-15	0	1.78673e-15	0
145	1.82428e-15	0	1.82428e-15	0
150	1.86768e-15	0	1.86768e-15	0
155	1.88411e-15	0	1.88411e-15	0
160	1.92616e-15	0	1.92616e-15	0
165	1.9421e-15	0	1.9421e-15	0
170	1.97045e-15	0	1.97045e-15	0
175	2.02292e-15	0	2.02292e-15	0
180	2.04715e-15	0	2.04715e-15	0
185	2.09476e-15	0	2.09476e-15	0
190	2.11234e-15	0	2.11234e-15	0
195	2.12978e-15	0	2.12978e-15	0
200	2.14994e-15	0	2.14994e-15	0
300	2.0568e-15	0	2.0568e-15	0
400	1.952e-15	0	1.952e-15	0
500	0.32578e-15	0	0.32578e-15	0

Tabela 4. Obliczone błędy dla różnych wielkości macierzy A, przy dwóch różnych typach

Wyniki są dokładnie takie same, wynika to z tego, że algorytm Thomasa to uproszczona wersja algorytmu Gaussa.
Jak można zauważyć w każdym przypadków wartości te były bardzo małe. Dla typu float w obu metodach są dość satysfakcjonującym wynikiem. Dla typu Decimal który ma większą precyzje obliczeń błąd w obu metodach jest równy zero.
Dla wielkości macierzy ponad 500 błędy są tak samo małe dla obydwu metod.

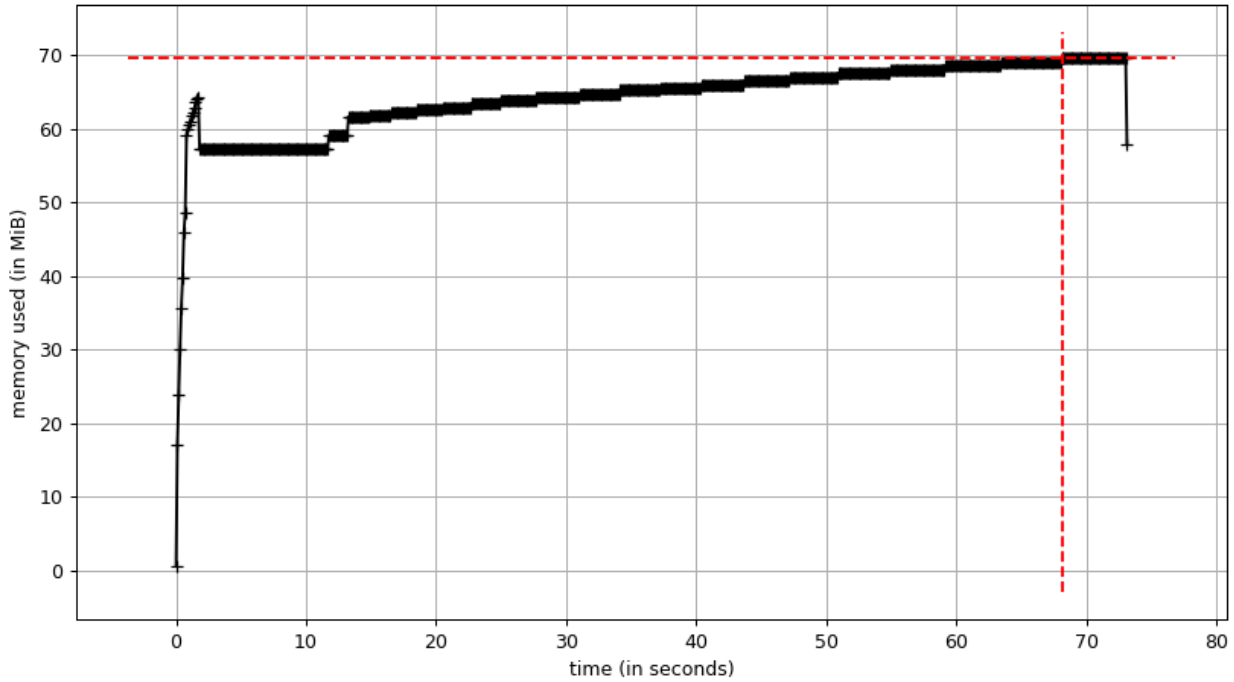
Poniższa tabela zawiera porównanie czasów rozwiązywania układu dla różnych typów oraz metod. Wynik jest podawany w sekundach oraz w jego wartość nie liczy się czas tworzenia macierzy.

	Float		Decimal	
Wielkość macierzy	Thomas	Gauss	Thomas	Gauss
100	0.00035	0.100658	0.001287	0.167837
105	0.000417	0.119548	0.001338	0.195698
110	0.000379	0.120863	0.001432	0.219505
115	0.000373	0.126112	0.001487	0.26366
120	0.000761	0.149131	0.00156	0.325934
125	0.000424	0.176687	0.001681	0.367887
130	0.000422	0.186545	0.001741	0.433925
135	0.000445	0.230187	0.00177	0.459724
140	0.00052	0.227882	0.001799	0.519806
145	0.000467	0.277514	0.001852	0.533257
150	0.000487	0.294298	0.002013	0.593289
155	0.000505	0.324437	0.002033	0.606515
160	0.000522	0.348178	0.002051	0.690134
165	0.00056	0.385581	0.002208	0.782302
170	0.000601	0.419587	0.002273	0.923846
175	0.001267	0.455103	0.002271	0.937356
180	0.000606	0.493136	0.002426	0.979977
185	0.000612	0.546487	0.002558	1.150819
190	0.000635	0.584837	0.002894	1.118412
195	0.000624	0.634347	0.00247	1.217498
200	0.000645	0.675575	0.003695	1.3567
400	0.001435	6.105335	0.005367	0.006676
500	0.001755	13.490941	0.007423	23.54654

Tabela 5.Porównanie czasów w sekundach dla dwóch różnych typów i metod

Można zauważyć, że metoda Thomasa dla dwóch różnych typów daje o wiele lepsze wyniki czasowe, które są lepsze od metody Gaussa nawet o trzy rzędy wielkości dla typu Float.Oczywistym jest, że dla typu Decimal ten czas jest w porównaniu dla typu Float(który ma mniejszą precyzje) większy. Wraz z wielkością macierzy rośnie również czas potrzebny na dokonanie obliczeń dla obydwu typów.
Postanowiłam również sprawdzić jak obydwie metody będą się zachowywać kiedy, będę próbować obliczyć wyniki dla macierzy wiele większych. Dla rozmiaru 500 metoda Gaussa potrzebowała już 23 sekund, zaś metoda Thomasa 0.007 sekund. Dla rozmiaru 1000 Metoda Thomasa 0.0136 sekundy, a metoda Gaussa obliczała wynik już ponad 3 minuty.

Poniższy wykres powstał poprzez wykonanie eksperymentu najpierw z metodą Thomasa dla wielkości macierzy od 200 do 300. Następnie była 10s przerwa w obliczeniach, po której ponownie został przeprowadzony eksperyment tym razem z metodą Gaussa na takich samych wielkościach macierzy.



Wykres 1.Przedstawienie w czasie zużycia pamięci dla obydwu metod przy wielkości macierzy od 200 do 300.

Można zaobserwować na powyższym wykresie, że zużycie pamięci przy metodzie Thomasa jest niższe niż przy metodzie Gaussa, gdzie ono stale rosło wraz z powiększaniem się macierzy.

WNIOSKI

- Metody Thomasa oraz Gaussa dają dokładnie takie same wyniki, jednak pierwsza metoda jest o wiele bardziej wydajniejsza pod względem czasowym jak i pamięciowym.
- Metoda Gaussa z powodu złego uwarunkowania układów, może dawać złe wyniki ze względu na problemy związane z arytmetyką komputera.
- Problemy z złym uwarunkowaniem układu można rozwiązać na przykład poprzez wybranie innego elementu przez który będziemy dzielić wiersze.
- Dla macierzy trójdiagonalnej o wiele lepszą metodą jest Metoda Thomasa.
- Dla typu z większą dokładności wyniki są lepsze, ponieważ później pojawiają się błędy związane z arytmetyką komputera.