Sprawozdanie z Listy nr 5

Rozwiązywanie układów równań liniowych o strukturze blokowej

Sara Żyndul
279686

5 stycznia 2026

Spis treści

1 Przedstawienie problemu

Zadanie polega na rozwiązaniu układu równań liniowych postaci $Ax = b$, gdzie $A \in \mathbb{R}^{n \times n}$ jest macierzą rzadką o specyficznej strukturze blokowej. Macierz A zdefiniowana jest następująco:

$$A = \begin{pmatrix} A_1 & C_1 & 0 & \dots & 0 \\ B_2 & A_2 & C_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & B_{v-1} & A_{v-1} & C_{v-1} \\ 0 & \dots & 0 & B_v & A_v \end{pmatrix} \tag{1}$$

gdzie n jest podzielne przez rozmiar bloku l, a $v = n/l$ oznacza liczbę bloków wzdłuż głównej diagonali. Poszczególne bloki mają następującą charakterystykę:

- $A_k \in \mathbb{R}^{l \times l}$ – macierze gęstej.
- $B_k \in \mathbb{R}^{l \times l}$ – macierze rzadkie, posiadające elementy niezerowe jedynie w pierwszym wierszu i ostatniej kolumnie,
- $C_k \in \mathbb{R}^{l \times l}$ – macierze diagonalne.

Celem jest implementacja efektywnych algorytmów numerycznych uwzględniających rzadkość macierzy, tak aby zredukować złożoność obliczeniową ze standardowego $O(n^3)$ do $O(n)$.

2 Struktura pamiętająca macierz

Dla dużych wartości n (rzędu 10^6) przechowywanie macierzy w postaci pełnej tablicy dwuwymiarowej jest nieefektywne i niemożliwe ze względu na ograniczenia pamięciowe komputera. Zaprojektowano strukturę `BlockMatrix`, która przechowuje wyłącznie niezerowe bloki macierzy. Struktura ta zawiera:

- Tablicę trójwymiarową dla bloków A_k o rozmiarze $l \times l \times v$.
- Tablicę trójwymiarową dla bloków B_k o rozmiarze $l \times l \times v$.
- Tablicę trójwymiarową dla bloków C_k o rozmiarze $l \times l \times v$.

Całkowita złożoność pamięciowa wynosi:

$$M(n) \approx v \cdot l^2 + v \cdot l^2 + v \cdot l^2 = 3 \cdot \frac{n}{l} \cdot l^2 = 3nl$$

Złożoność pamięciowa wynosi zatem $O(n)$, co jest optymalną wartością dla macierzy rzadkich tego typu.

3 Metoda eliminacji Gaussa

3.1 Algorytm bazowy dla macierzy gęstych

Klasyczna metoda eliminacji Gaussa dla macierzy gęstej $A \in \mathbb{R}^{n \times n}$ polega na sprowadzeniu jej do postaci górnotrójkątnej U przy pomocy operacji elementarnych na wierszach.

W k-tym kroku algorytmu (dla $k = 1, \ldots, n-1$) zerowane są elementy pod główną przekątną w k-tej kolumnie. Współczynnik zerujący (mnożnik) definiuje się jako $m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$, a następnie aktualizuje się wiersze: $w_i \leftarrow w_i - m_{ik} \cdot w_k$ dla $i > k$. Standardowa złożoność tego procesu wynosi $O(n^3)$.

3.2 Adaptacja do struktury blokowej

W przypadku omawianej macierzy blokowej, eliminacja Gaussa została zmodyfikowana, aby wykorzystać fakt występowania licznych bloków zerowych. Proces przebiega następująco:

1. Iterujemy po blokach diagonalnych k od 1 do $v - 1$.
2. Wewnątrz bloku A_k wykonujemy lokalną eliminację Gaussa, sprowadzając go do postaci górnotrójkątnej. Operacje na wierszach wewnątrz A_k są powielane na odpowiadających im wierszach bloku C_k oraz wektora b.
3. Następnie zerujemy blok B_{k+1} (znajdujący się pod A_k), wykorzystując wiersze z przetworzonego bloku A_k. Ponieważ B_{k+1} znajduje się w tym samym wierszu blokowym co A_{k+1}, a A_k w tym samym co C_k, operacja $Wiersz(B_{k+1}) - m \cdot Wiersz(A_k)$ powoduje modyfikację bloku A_{k+1} o wartości wynikające z C_k. Struktura bloków zerowych gwarantuje, że zmniejszenie rzadkości bloków nie propaguje się dalej niż do sąsiedniego bloku.

3.2.1 Warianty wyboru elementu głównego

- **Bez wyboru (GNP):** Pivotem jest zawsze element diagonalny a_{ii}.
- **Z częściowym wyborem (GPP):** W celu zmiejszenia błędu względnego, przed eliminacją i-tej kolumny w bloku A_k, wyszukiwany jest element o największym module w tej kolumnie (w zakresie wierszy należących do bieżącego bloku). Następuje zamiana wierszy w A_k, C_k, B_k (jeśli istnieje) oraz w wektorze b.

3.3 Analiza złożoności czasowej

Niech l będzie stałym rozmiarem bloku.

- Liczba kroków głównych pętli (liczba bloków) wynosi $v = n/l$.
- W każdym kroku wykonujemy operacje na macierzach rozmiaru $l \times l$. Koszt operacji wewnątrz bloku (mnożenia, dodawania, zamiany wierszy) zależy tylko od l i w pesymistycznym przypadku wynosi $O(l^3)$.
- Całkowity czas $T(n)$ można oszacować jako:

$$T(n) \approx \frac{n}{l} \cdot l^3 = n \cdot l^2$$

Dla ustalonego, małego l (w zadaniu $l \geq 2$), czynnik l^2 jest stałą. Zatem całkowita złożoność czasowa wynosi $O(n)$.

4 Rozkład LU

4.1 Teoria i związek z eliminacją Gaussa

Rozkład LU jest metodą faktoryzacji macierzy A na iloczyn macierzy dolnotrójkątnej L (z jedynkami na przekątnej) i górnotrójkątnej U, tj. $A = LU$. Proces ten jest ściśle związany z eliminacją Gaussa. Przejście od macierzy $A^{(k)}$ do $A^{(k+1)}$ (kolejny krok eliminacji) można zapisać jako mnożenie przez macierz elementarną $L^{(k)}$:

$$A^{(k+1)} = L^{(k)} A^{(k)}$$

Proces sprowadzania macierzy $A^{(1)}$ do macierzy górnotrójkątnej $A^{(n)}$ (którą oznaczamy jako U) w $n-1$ krokach można zapisać:

$$U = L^{(n-1)} \dots L^{(2)} L^{(1)} A$$

Stąd otrzymujemy:

$$A = (L^{(n-1)} \dots L^{(2)} L^{(1)})^{-1} U$$

$$A = L^{(1)-1} L^{(2)-1} \dots L^{(n-1)-1} U$$

Przyjmując $L = L^{(1)-1} \dots L^{(n-1)-1}$, otrzymujemy $A = LU$. W praktyce macierz L zawiera mnożniki używane podczas eliminacji Gaussa.

4.2 Adaptacja do struktury blokowej

Zaimplementowano algorytm rozkładu LU działający "w miejscu"(in-place). Elementy macierzy L i U są przechowywane w strukturze ‘BlockMatrix‘ w następujący sposób:

- Macierz U: elementy na i nad główną przekątną bloków A_k oraz całe bloki C_k.
- Macierz L: elementy pod główną przekątną bloków A_k oraz całe bloki B_k (z domyślnymi jedynkami na głównej diagonali całego układu).

Podczas eliminacji bloku B_{k+1} przy użyciu A_k, wyliczone mnożniki są zapisywane w miejscu zerowanego bloku B_{k+1} (stając się fragmentem macierzy L). Należy przy tym pamiętać o aktualizacji pozostałej części bloku B_{k+1} oraz modyfikacji A_{k+1} o wpływ C_k.

Dla wariantu z częściowym wyborem elementu głównego (LUPP), algorytm wyznacza rozkład $PA = LU$, gdzie P jest macierzą permutacji. Implementacja zwraca w tym przypadku wektor permutacji p.

4.3 Rozwiązywanie układu przy pomocy LU

Po wyznaczeniu rozkładu, rozwiązanie układu $Ax = b$ (lub $PAx = Pb$) sprowadza się do rozwiązania dwóch układów trójkątnych:

1. $Ly = b$ (dla LUPP: $Ly = b[p]$) – podstawienie w przód. Wykorzystuje bloki B_k i dolne części A_k.
2. $Ux = y$ – podstawienie wstecz. Wykorzystuje górne części A_k i bloki C_k.

4.4 Analiza złożoności czasowej

- **Faktoryzacja:** Przechodzi przez wszystkie n/l bloków raz. Złożoność $O(n)$.
- **Solver:** Wykonuje dwa przejścia przez bloki (w przód i w tył). Każde przejście to operacje na małych macierzach $l \times l$. Złożoność $O(n)$.

Łączna złożoność wynosi $O(n)$.

5 Wyniki eksperymentów

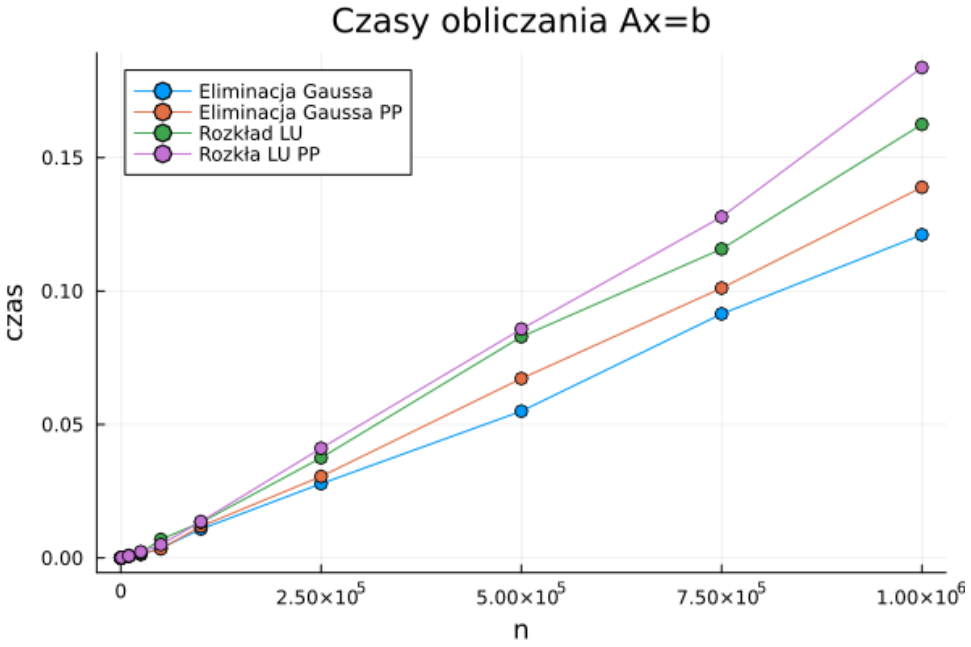
Testy wydajnościowe i dokładnościowe przeprowadzono dla macierzy dobrze uwarunkowanych o rozmiarach $n \in \{10^3, 10^4 \ldots, 10^6\}$ generowanych funkcją 'blockmat'.

5.1 Analiza czasu wykonania

Poniższa tabela oraz wykres (Rysunek 1) prezentują czasy działania algorytmów.

Tabela 1: Czas wykonania algorytmów [s] w zależności od rozmiaru macierzy

n	**GNP**	**GPP**	**LUNP**	**LUPP**
16	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$
1000	$9.1 \cdot 10^{-5}$	$1.03 \cdot 10^{-4}$	$1.22 \cdot 10^{-4}$	$1.39 \cdot 10^{-4}$
10000	$9.04 \cdot 10^{-4}$	$5.21 \cdot 10^{-4}$	$5.76 \cdot 10^{-4}$	$6.78 \cdot 10^{-4}$
50000	$2.65 \cdot 10^{-3}$	$3.05 \cdot 10^{-3}$	$3.36 \cdot 10^{-3}$	$3.90 \cdot 10^{-3}$
100000	$5.48 \cdot 10^{-3}$	$6.23 \cdot 10^{-3}$	$6.88 \cdot 10^{-3}$	$8.01 \cdot 10^{-3}$
500000	$2.92 \cdot 10^{-2}$	$3.34 \cdot 10^{-2}$	$3.67 \cdot 10^{-2}$	$4.17 \cdot 10^{-2}$
750000	$4.35 \cdot 10^{-2}$	$4.99 \cdot 10^{-2}$	$5.40 \cdot 10^{-2}$	$6.29 \cdot 10^{-2}$
1000000	$5.64 \cdot 10^{-2}$	$6.66 \cdot 10^{-2}$	$7.18 \cdot 10^{-2}$	$8.38 \cdot 10^{-2}$



Rysunek 1: Zależność czasu obliczeń od rozmiaru macierzy n.

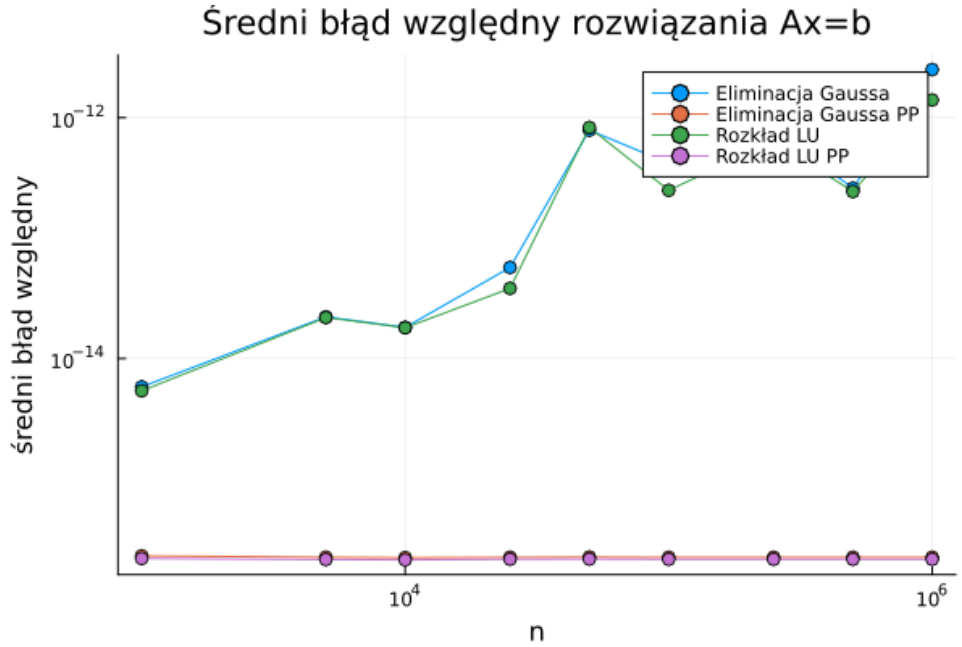
Interpretacja: Wyniki potwierdzają teoretyczną złożoność $O(n)$ – czas rośnie liniowo wraz z rozmiarem problemu. Dla $n = 10^6$ czas wynosi około 0.15s. Algorytmy z pivotingiem (GPP, LUPP) są nieznacznie wolniejsze od ich odpowiedników bez pivotingu (narzut na wyszukiwanie maksimum i permutacje), a rozkład LU jest nieco wolniejszy od eliminacji Gaussa (narzut strukturalny i dwuetapowość), lecz różnice te są rzędu stałego czynnika.

5.2 Analiza błędu względnego

Tabela 2 i wykres (Rysunek 2) przedstawiają błąd względny rozwiązania $\frac{||x_{obl}-x_{dokl}||}{||x_{dokl}||}$.

Tabela 2: Błędy względne algorytmów w zależności od rozmiaru macierzy

n	GNP	GPP	LUNP	LUPP
16	$6.64 \cdot 10^{-16}$	$2.76 \cdot 10^{-16}$	$5.12 \cdot 10^{-16}$	$4.86 \cdot 10^{-16}$
1000	$1.23 \cdot 10^{-14}$	$2.40 \cdot 10^{-16}$	$1.13 \cdot 10^{-14}$	$2.31 \cdot 10^{-16}$
10000	$1.67 \cdot 10^{-14}$	$5.24 \cdot 10^{-16}$	$2.32 \cdot 10^{-14}$	$5.17 \cdot 10^{-16}$
50000	$1.02 \cdot 10^{-13}$	$5.00 \cdot 10^{-16}$	$1.02 \cdot 10^{-13}$	$4.92 \cdot 10^{-16}$
100000	$2.54 \cdot 10^{-13}$	$4.66 \cdot 10^{-16}$	$1.70 \cdot 10^{-13}$	$4.58 \cdot 10^{-16}$
500000	$6.07 \cdot 10^{-13}$	$4.32 \cdot 10^{-16}$	$6.29 \cdot 10^{-13}$	$4.25 \cdot 10^{-16}$
750000	$5.30 \cdot 10^{-13}$	$4.12 \cdot 10^{-16}$	$3.78 \cdot 10^{-13}$	$4.05 \cdot 10^{-16}$
1000000	$7.06 \cdot 10^{-14}$	$4.11 \cdot 10^{-16}$	$8.64 \cdot 10^{-14}$	$4.03 \cdot 10^{-16}$

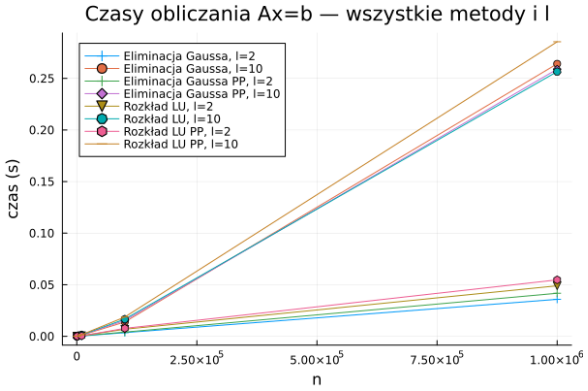


Rysunek 2: Zależność błędu względnego od rozmiaru macierzy n (skala logarytmiczna).

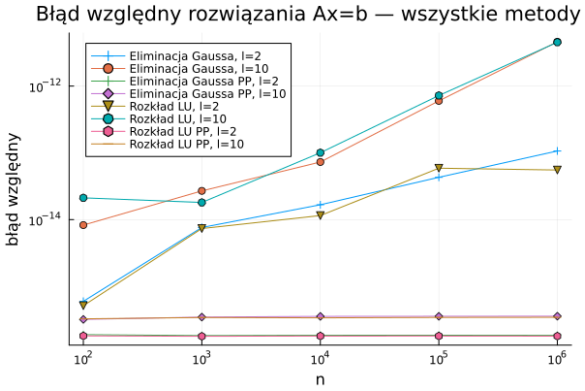
Interpretacja: Metody bez pivotingu (GNP, LUNP) wykazują wyraźny wzrost błędu wraz ze wzrostem n, osiągając poziom 10^{-13}, co świadczy o błędach zaokrągleń. Metody z częściowym wyborem elementu głównego (GPP, LUPP) utrzymują błąd na poziomie precyzji maszynowej (10^{-16}), co dowodzi ich wysokiej stabilności numerycznej.

5.3 Wpływ rozmiaru bloku l na wyniki

W celu zbadania wpływu wielkości podmacierzy na działanie algorytmów, przeprowadzono dodatkowe testy dla rozmiarów bloku $l = 2$ oraz $l = 10$. Poniższe wykresy prezentują porównanie czasów wykonania oraz błędów względnych dla wszystkich badanych metod.



Rysunek 3: Porównanie czasów obliczeń dla $l = 2$ i $l = 10$.



Rysunek 4: Porównanie błędów względnych dla $l = 2$ i $l = 10$.

Na podstawie wykresu 4 można stwierdzić, że rozmiar bloku l nie ma istotnego wpływu na stabilność numeryczną algorytmów. Błędy względne dla $l = 10$ pozostają na tym samym poziomie rzędów wielkości, co dla $l = 2$ (odpowiednio $\approx 10^{-16}$ dla metod z pivotingiem i wzrost dla metod bez pivotingu).

Wykres 3 pokazuje natomiast wyraźną korelację między rozmiarem bloku a czasem wykonania. Wzrost parametru l powoduje wydłużenie czasu obliczeń, co wynika ze zwiększonej liczby operacji wewnątrz bloków (złożoność operacji wewnątrzblokowych rośnie kwadratowo lub sześciennie względem l). Przykładowo dla największej badanej macierzy $n = 10^6$:

- dla $l = 2$ czas wynosi około $0.05\,s$,
- dla $l = 10$ czas wzrasta do około $0.25\,s$.

Mimo wzrostu czasu, zależność od głównego rozmiaru macierzy n pozostaje liniowa.

6 Wnioski

1. Zastosowanie struktury blokowej pozwoliło na zredukowanie złożoności czasowej i pamięciowej do $O(n)$, co umożliwia efektywne rozwiązywanie bardzo dużych układów równań.
2. Częściowy wybór elementu głównego jest niezbędny dla uniknięcia błędów wynikających z błędów zaokrągleń algorytmów, przy czym jego koszt obliczeniowy jest pomijalny.
3. Eliminacja Gaussa i rozkład LU dają bardzo zbliżone wyniki czasowe i jakościowe. Rozkład LU jest bardziej efektywny w przypadku konieczności wielokrotnego rozwiązywania układu z tą samą macierzą A (dopiero przy rozwiązywaniu układu $LUx = b$ potrzebujemy wektora b).