Agata Jasionowska 229726

Laboratorium – Lista 5

1. Zadanie 1

1.1. Opis problemu

Zadanie dotyczyło rozwiązania układu równań liniowych:

$$\boldsymbol{Ax} = \boldsymbol{b},$$

dla danej macierzy współczynników $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ i wektora prawych stron $\boldsymbol{b} \in \mathbb{R}^n$, $n \geq 4$ metodą eliminacji Gaussa (bez wyboru elementu głównego oraz z częściowym wyborem elementu głównego) oraz obliczeniu rozkładu $\boldsymbol{LU}$ macierzy i rozwiązaniu z jego wykorzystaniem układu $\boldsymbol{Ax} = \boldsymbol{b}$.

Macierz $\boldsymbol{A}$ jest rzadka i blokowa o następującej strukturze:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{C}_1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B}_2 & \boldsymbol{A}_2 & \boldsymbol{C}_2 & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_3 & \boldsymbol{A}_3 & \boldsymbol{C}_3 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{B}_{v-2} & \boldsymbol{A}_{v-2} & \boldsymbol{C}_{v-2} & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_{v-1} & \boldsymbol{A}_{v-1} & \boldsymbol{C}_{v-1} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_v & \boldsymbol{A}_v \end{pmatrix},$$

gdzie $v = \frac{n}{\ell}$, $l|n$, a $\ell \geq 2$ jest rozmiarem wszystkich bloków $\boldsymbol{A}_k$, $\boldsymbol{B}_k$, $\boldsymbol{C}_k$ macierzy $\boldsymbol{A}$.

Struktura wewnętrznych bloków $\boldsymbol{A}_k$, $\boldsymbol{B}_k$, $\boldsymbol{C}_k$ macierzy $\boldsymbol{A}$ jest następująca:

(a) $\boldsymbol{A}_k \in \mathbb{R}^{\ell\times\ell}$, $k = 1, \ldots, v$ są macierzami gęstymi,
(b) $\boldsymbol{B}_k \in \mathbb{R}^{\ell\times\ell}$, $k = 2, \ldots, v$ są postaci:

$$\boldsymbol{B}_k = \begin{pmatrix} 0 & \cdots & 0 & b^k_{1\,\ell-1} & b^k_{1\,\ell} \\ 0 & \cdots & 0 & b^k_{2\,\ell-1} & b^k_{2\,\ell} \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & b^k_{\ell\,\ell-1} & b^k_{\ell\,\ell} \end{pmatrix},$$

(c) $\boldsymbol{C}_k \in \mathbb{R}^{\ell\times\ell}$, $k = 1, \ldots, v-1$ są diagonalne:

$$\boldsymbol{C}_k = \begin{pmatrix} c_1^k & 0 & 0 & \cdots & 0 \\ 0 & c_2^k & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c_{\ell-1}^k & 0 \\ 0 & \cdots & 0 & 0 & c_\ell^k \end{pmatrix},$$

2. Sposób rozwiązania

2.1. Wykorzystanie pamięci

W konstruowaniu rozwiązania problemu niezwykle istotny był odpowiedni sposób przechowywania macierzy w pamięci.

Zważywszy na to, iż $\boldsymbol{A}$ jest macierzą rzadką (tylko $n\ell + 3(n-\ell)$ elementów niezerowych — ℓ^2 w każdym z v bloków $\boldsymbol{A}_k$, 2ℓ w każdym z $v-1$ bloków $\boldsymbol{B}_k$ i ℓ w każdym z $v-1$ bloków $\boldsymbol{C}_k$), to istnieje sposób na zapamiętanie jej znacznie efektywniej, niżby to miało miejsce w przypadku dwuwymiarowej tablicy $n \times n$.

W implementacji skorzystano z dostępnej w języku `Julia` struktury `SparseMatrixCSC`, gdzie macierze przechowywane są w skompresowanym porządku kolumnowym. Pozwala to na jeszcze jedną optymalizację, bazującą na fakcie, że czas dostępu do elementów jest krótszy, jeśli poruszamy się wzdłuż kolumn. Skoro opisywany algorytm eliminacji Gaussa ma przebieg wierszowy, więc przechowywane w pamięci macierze są zawsze transponowane. W dalszych rozważaniach jednak, dla ułatwienia, macierze indeksowane są w sposób standardowy, gdyż różnica polega wyłącznie na zamianie miejscami indeksów wiersza i kolumny.

2.2. Eliminacja Gaussa

2.2.1. Podstawy teoretyczne

Metoda eliminacji Gaussa to algorytm rozwiązywania układów równań liniowych, obliczania rzędu macierzy i wartości wyznacznika oraz wyznaczania rozkładu LU, wykorzystując operacje elementarne.

W sposobie rozwiązywania układu równań tą metodą wyróżnić można dwa główne etapy. Pierwszym z nich jest sprowadzenie układu do układu równoważnego z macierzą trójkątną górną. Algorytm będzie polegał na zerowaniu kolejnych elementów znajdujących się pod diagonalą. Przykładowo w celu wyzerowania elementu a_{i1} od i-tego wiersza zostanie odjęty pierwszy wiersz pomnożony przez $z = a_{i1}/a_{11}$ (liczba ta nazywana dalej będzie *mnożnikiem*). W ten sposób w pierwszym kroku wyzerowane zostają wszystkie elementu poniżej pierwszego wiersza w pierwszej kolumnie, dalej powtarzane jest to dla drugiej kolumny, itd. Warto zwrócić uwagę na fakt, iż procedura zawiedzie w sytuacji, gdy na diagonali pojawi się zero. Algorytm wymaga wtedy do poprawnego działania pewnej modyfikacji, np. zamiany miejscami wierszy.

Drugim i ostatnim etapem rozwiązywania układu równań jest zastosowanie algorytmu zwanego *algorytmem podstawiania wstecz*. Opiera się on w głównej mierze na zastosowaniu wzoru

$$n_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}}{a_{ii}}$$

do kolejnych i-tych wierszy macierzy, rozpoczynając od ostatniego z nich.

W tym miejscu warto także wspomnieć o złożoności obliczeniowej. Otóż algorytm eliminacji Gaussa cechuje złożoność rzędu $O(n^3)$, zaś podstawianie wstecz z kroku drugiego — $O(n^2)$. Prowadzi to do konkluzji, że w celu rozwiązania układu równań należy wykonać $O(n^3)$ operacji.

2.2.2. Rozwiązywanie układów równań

W sposobie rozwiązania uwzględniona została charakterystyczna, trójdiagonalno blokowa postać macierzy $\boldsymbol{A}$, co pozwoliło na znaczne zredukowanie liczby potrzebnych do wykonania operacji w stosunku do rozwiązania z zastosowaniem standardowego algorytmu eliminacji Gaussa. Otóż należy zauważyć, że nie jest konieczne wyzerowanie wszystkich elementów znajdujących się w danej kolumnie pod diagonalą. Dla pierwszych $\ell - 2$ kolumn niezerowe elementy mogą się znajdować jedynie w pierwszych ℓ rzędach (są to tylko elementy bloku $\boldsymbol{A}_1$), dla kolejnych ℓ kolumn niezerowe elementy mogą się znajdować w pierwszych 2ℓ rzędach (dwie ostatnie kolumny bloku $\boldsymbol{B}_2$ oraz elementy bloku $\boldsymbol{A}_3$), w kolejnych ℓ kolumnach niezerowe elementy będą się znajdowały w pierwszych 3ℓ rzędach (niezerowe elementy bloku $\boldsymbol{B}_3$ oraz elementy bloku $\boldsymbol{A}_4$) itd.

W ostatnich ℓ rzędach to $\boldsymbol{A}_v$ jest leżącym najbardziej z prawej blokiem, a więc ostatnie niezerowe elementy leżą w kolumnie n-tej. Pozwala to na zbudowanie wzoru na maksymalny indeks kolumny, w której w danym rzędzie znajduje się niezerowy element:

$$last_c(row) = \min\{n,\ row + \ell\}.$$

Ponadto, jeśli w którymś z etapów metody eliminacji Gaussa następuje odjęcie podanego rzędu od znajdujących się poniżej to nie następuje modyfikacja elementów w kolumnach o indeksach większych niż $last_c(row)$.

Bazując na spostrzeżeniach przedstawionych powyżej, indeks ostatniego niezerowego elementu danej kolumny można wyliczyć jako:

$$last_r(column) = \min\left\{n,\ \ell + \ell \cdot \left\lfloor \frac{column + 1}{\ell} \right\rfloor\right\}$$

Warte zauważenia jest także to, iż w każdym wierszu (poza ℓ ostatnimi elementami) ostatni niezerowy element należy do diagonali bloku $\boldsymbol{C}_i$. Elementy te są zawsze odległe o ℓ od elementów diagonalnych całej macierzy. Finalnie eliminacja Gaussa doprowadziła do otrzymania układu z macierzą trójkątną górną, na którego rozwiązanie pozwala już algorytm podstawiania wstecz.

Złożoność obliczeniowa całej metody, przyjmując ℓ jako stałą, wynosi $O(n)$ (liczba przebiegów wszystkich pętli to: $(n-1) \cdot 2\ell \cdot \ell + n \cdot l$).

Szczegółowy przebieg opisywanej metody przedstawia Algorytm 1.

2.2.3. Wariant z wyborem elementu głównego

Powyżej została przedstawiona metoda eliminacji Gaussa bez wyboru elementu głównego. W pewnych sytuacjach przydatny bywa także jej wariant z częściowym wyborem, umożliwiając rozwiązywanie układów z zerami na diagonali. W praktyce stosuje się następująco: wybrany zostaje wiersz, dla którego element z eliminowanej obecnie kolumny k ma największą wartość (co do modułu), a następnie zamieniony z k-tym wierszem; dalsze kroki pozostają bez zmian.

Takie działanie może być bardzo kosztowne dla macierzy o dużych rozmiarach, stąd właśnie informacja, na której pozycji w macierzy znajduje się obecnie dany wiersz przechowywana jest w dodatkowym wektorze (p). Zmiana w samym algorytmie jest niewielka - odwołanie do konkretnego wiersza zastąpione zostało odwołaniem do odpowiadającej pozycji w wektorze permutacji.

Kolejnym z problemów jest prawidłowe określenie granicznego $last_c$. Zaobserwować można, że podczas eliminowania współczynników z $\ell - 2$ pierwszych kolumn najdalszy niezerowy element można stworzyć w kolumnie z indeksem 2ℓ — poprzez odejmowanie ℓ-tego wiersza, który w tej kolumnie posiada niezerowy element. Podczas eliminowania współczynników z kolejnych ℓ kolumn najdalszy niezerowy element można stworzyć w kolumnie z indeksem 3ℓ, analogicznie poprzez odejmowanie 2ℓ-tego wiersza, który w tej kolumnie posiada niezerowy element. Zgodnie z tym rozumowaniem nowy wzór będzie miał następującą postać:

$$last_c(row) = \min\ \left\{n,\ 2\ell + l \cdot \left\lfloor \frac{row + 1}{\ell} \right\rfloor\right\}.$$

Algorytm 1: Metoda eliminacji Gaussa.

Input :
- $\boldsymbol{A}$ — macierz,
- $\boldsymbol{b}$ — wektor prawych stron,
- n — rozmiar macierzy $\boldsymbol{A}$,
- ℓ — rozmiar bloku macierzy $\boldsymbol{A}$.

Output :
- $\boldsymbol{x}$ — wektor długości n zawierający pierwiastki układu $\boldsymbol{Ax} = \boldsymbol{b}$.

```
1  Function gauss(A, b, n, ℓ)
2      for k ← 1 to n − 1 do
3          lastC ← min(k + ℓ, n)
4          lastR ← min(ℓ + ℓ · ⌊(k+1)/ℓ⌋, n)
5          for i ← k + 1 to lastR do
6              if A[k][k] = 0 then
7                  error Diagonal coefficient is equal zero.
8              end
9              z ← A[i][k]/A[k][k]
10             A[i][k] ← 0
11             for j ← k + 1 to lastC do
12                 A[i][j] ← A[i][j] − z · A[k][j]
13             end
14             b[i] ← b[i] − z · b[k]
15         end
16     end
17     for i ← n downto 1 do
18         sum ← 0
19         lastC ← min(i + ℓ, n)
20         for j ← k + 1 to lastC do
21             sum ← sum + x[i] · A[i][j]
22         end
23         x[i] ← (b[i] − sum)/A[i][i]
24     end
25     return x
```

Identycznie ograniczenie zostało nałożone dla wykonywania algorytmu podstawiania wstecz. Brak nowych niezerowych elementów poza tymi już uwzględnionymi, a jedyną zmianą jest więcie pod uwagę permutacji wiersza.

Złożoność obliczeniowa wariantu z częściowym wyborem elementu głównego jest gorsza niż poprzednio przedstawiona standardowa eliminacja Gaussa (ograniczenia nałożone na $last_c$). Ostatecznie jednak, przy założeniu, że ℓ jest stałe, nie wpływa to na osiągnięcie złożoności większej niż $O(n)$.

Szczegóły implementacji metody eliminacji Gaussa z wyborem elementu głównego przedstawia Algorytm 2.

2.3. Rozkład LU

2.3.1. Podstawy teoretyczne

Rozkład $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ to przedstawienie zadanej macierzy w postaci iloczynu $\boldsymbol{A} = \boldsymbol{LU}$, gdzie $\boldsymbol{L}$ jest macierzą trójkątną dolną, zaś $\boldsymbol{U}$ — macierzą trójkątną górną. Dodatkowym założeniem jest, że wszystkie diagonalne elementy $\boldsymbol{L}$ to 1.

Na uzyskanie rozkładu $\boldsymbol{LU}$ pozwala wspomniany wcześniej algorytm eliminacji Gaussa. Macierz $\boldsymbol{A}$ zostanie przekształcona do postaci górnotrójkątnej (będzie to finalnie szukana macierz $\boldsymbol{U}$), zaś macierz $\boldsymbol{L}$ uzyskana zostanie poprzez zapamiętywanie mnożników użytych do przekształceń (mnożnik z_{ij} służący wyzerowaniu elementu a_{ij} zapisane zostanie w i-tym wierszu i j-tej kolumnie $\boldsymbol{L}$).

Algorytm 2: Metoda eliminacji Gaussa z częściowym wyborem elementu głównego.

Input :
- $\boldsymbol{A}$ — macierz,
- $\boldsymbol{b}$ — wektor prawych stron,
- n — rozmiar macierzy $\boldsymbol{A}$,
- ℓ — rozmiar bloku macierzy $\boldsymbol{A}$.

Output :
- $\boldsymbol{x}$ — wektor długości n zawierający pierwiastki układu $\boldsymbol{Ax} = \boldsymbol{b}$.

```
1  Function gauss_with_pivot(A, b, n, ℓ)
2      p ← {i : i ∈ {1, ..., n}}
3      for k ← 1 to n − 1 do
4          lastR ← min(ℓ + ℓ · ⌊(k+1)/ℓ⌋, n)
5          lastC ← min(2ℓ + ℓ · ⌊(k+1)/ℓ⌋, n)
6          for i ← k + 1 to lastR do
7              idx_max ← m takie, że: A[p[m]][k] = max(|A[p[q]][k]| : q ∈ {i, ..., lastR})
8              if p[idx_max] = 0 then
9                  error Singular matrix.
10             end
11             swap (p[k], p[idx_max])
12             z ← A[p[i]][k]/A[p[k]][k]
13             A[p[i]][k] ← 0
14             for j ← k + 1 to lastC do
15                 A[p[i]][j] ← A[p[i]][j] − z · A[p[k]][j]
16             end
17             b[p[i]] ← b[p[i]] − z · b[p[k]]
18         end
19     end
20     for i ← n downto 1 do
21         sum ← 0
22         lastC ← min(2ℓ + ℓ · ⌊(i+1)/ℓ⌋, n)
23         for j ← k + 1 to lastC do
24             sum ← sum + x[j] · A[p[i]][j]
25         end
26         x[i] ← (b[p[i]] − sum)/A[p[i]][i]
27     end
28     return x
```

Przeprowadzenie rozkładu $\boldsymbol{LU}$ ma złożoność obliczeniową $O(n^3)$. Warto w tym miejscu wspomnieć też, że rozkład $\boldsymbol{LU}$ pozwala na szybsze rozwiązywanie układów równań w przypadku, gdy występuje ta sama macierz — etap z eliminacją Gaussa wykonywany jest wtedy wyłącznie raz (co jest fortunne z uwagi na jego kosztowność), zaś rozwiązanie układów równań sprowadza się do dwóch kroków: $\boldsymbol{Lz} = \boldsymbol{b}$ oraz $\boldsymbol{Ux} = \boldsymbol{z}$. Skoro macierze $\boldsymbol{L}$ oraz $\boldsymbol{U}$ są trójkątne, to koszt wykonania powyższych operacji nie przekracza $O(n^2)$.

2.3.2. Rozwiązywanie układów równań

Pierwszym krokiem jest przeprowadzenie rozkładu LU dla danej macierzy. Metoda ta przebiega w sposób identyczny jak metoda eliminacji Gaussa opisana w Algorytmie 1, z jedną różnicą — w miejsce zerowanych elementów a_{ij} podstawiane są mnożniki $z = a_{ij}/a_{jj}$ (interpretowanie jako elementy macierzy $\boldsymbol{L}$).

Dla przypadku rozkładu $\boldsymbol{LU}$ macierzy z częściowym wyborem elementu głównego utworzono osobną metodę. Zwraca ona wektor zastosowanych permutacji p, służący dalej do odtworzenia pierwotnego porządku wierszy macierzy. Jej działanie jest analogiczne do działania metody eliminacji Gaussa z częściowym wyborem elementu głównego (ponownie, jedyną różnicą jest wstawianie w miejsce zerowanych elementów mnożników).

Złożoność obliczeniowa rozkładu $\boldsymbol{LU}$ jest równa złożoności obliczeniowej metody eliminacji Gaussa, czyli wynosi $O(n)$ (znów przy założeniu, że ℓ jest stałą).

Drugim etapem jest już rozwiązanie układu równań, które to sprowadza się do znalezienia rozwiązania spełniającego układ ($\boldsymbol{Lz} = \boldsymbol{b}$ oraz $\boldsymbol{Ux} = \boldsymbol{z}$) z macierzą trójkątną dolną oraz trójkątną górną. W tym celu zastosowano odpowiednio algorytm podstawiania w przód i podstawiania wstecz oraz dodatkowo zredukowano liczbę operacji (umożliwia to postać rozważanych macierzy). Indeks ostatniej niezerowej kolumny w danym wierszu dla algorytmu podstawiania wstecz ponownie wyraża się wzorem:

$$last_c(row) = \min\{n,\ row + \ell\},$$

zaś dla algorytmu podstawiania w przód:

$$last_c(row) = \min\left\{n,\ \ell \cdot \left\lfloor \frac{row - 1}{\ell} \right\rfloor\right\},$$

Oba wspomniane algorytmy wykonują $O(\ell)$ operacji dla jednego wiersza. Podsumowując, rozwiązanie układu równań z podanym rozkładem $\boldsymbol{LU}$ macierzy ma koszt wykonania $O(n)$.

Poniżej przedstawiono szczegółowy przebieg metody rozwiązującej układ równań na podstawie rozkładu $\boldsymbol{LU}$ macierzy (Algorytm 3) oraz jego wersję z wyborem elementu głównego (Algorytm 4).

Algorytm 3: Rozwiązywanie układu równań z rozkładu $\boldsymbol{LU}$ macierzy.

Input :
- $\boldsymbol{A}$ — macierz przekształcona do postaci, w której elementy na i ponad diagonalą odpowiadają elementom macierzy $\boldsymbol{U}$, zaś elementy pod diagonalą — elementom macierzy $\boldsymbol{U}$,
- $\boldsymbol{b}$ — wektor prawych stron.
- n — rozmiar macierzy $\boldsymbol{A}$,
- ℓ — rozmiar bloku macierzy $\boldsymbol{A}$.

Output :
- $\boldsymbol{x}$ — wektor długości n zawierający pierwiastki układu $\boldsymbol{Ax} = \boldsymbol{b}$.

```
1  Function solveLU(A, b, n, ℓ)
2      for i ← 1 to n do
3          sum ← 0
4          column ← min(ℓ · ⌊(i−1)/ℓ⌋, n)
5          for j ← column to i − 1 do
6              sum ← sum + z[j] · A[i][j]
7          end
8          z[i] = b[i] − sum
9      end
10     for i ← n downto 1 do
11         sum ← 0
12         lastC ← min(i + ℓ, n)
13         for j ← i + 1 to lastC do
14             sum ← sum + x[j] · A[i][j]
15         end
16         x[i] ← (z[i] − sum)/A[i][i]
17     end
18     return x
```

3. Wyniki eksperymentów

Celem potwierdzenia poprawności implementowanych metod wygenerowano takie prawe strony, aby rozwiązaniem układu stał się wektor $(1, \ldots, 1)^{\mathrm{T}}$. Tabela 1 prezentuje błędy względne wyników, uzyskane dla tak przygotowanych układów równań.

Utworzone metody zostały poddane dalszym testom - uśredniono wyniki czasowe oraz pamięciowe 100 prób dla losowych macierzy identycznego rozmiaru, zaś jako wielkość bloku przyjęto

Algorytm 4: Rozwiązywanie układu równań z rozkładu $\boldsymbol{LU}$ macierzy z wyborem elementu głównego.

Input :

- $\boldsymbol{A}$ — macierz przekształcona do postaci, w której elementy na i ponad diagonalą odpowiadają elementom macierzy $\boldsymbol{U}$, zaś elementy pod diagonalą — elementom macierzy $\boldsymbol{U}$,
- $\boldsymbol{b}$ — wektor prawych stron,
- n — rozmiar macierzy $\boldsymbol{A}$,
- ℓ — rozmiar bloku macierzy $\boldsymbol{A}$,
- $\boldsymbol{p}$ — wektor permutacji wierszy macierzy $\boldsymbol{A}$.

Output :

- $\boldsymbol{x}$ — wektor długości n zawierający pierwiastki układu $\boldsymbol{Ax} = \boldsymbol{b}$.

```
1  Function solveLU_with_pivot(A, b, n, ℓ)
2      for i ← 1 to n do
3          sum ← 0
4          column ← min(ℓ · ⌊(i−1)/ℓ⌋ − 1, n)
5          for j ← column to i − 1 do
6              sum ← sum + z[j] · A[p[i]][j]
7          end
8          z[i] = b[p[i]] − sum
9      end
10     for i ← n downto 1 do
11         sum ← 0
12         lastC ← min(2ℓ + ℓ · ⌊(i+1)/ℓ⌋, n)
13         for j ← i + 1 to lastC do
14             sum ← sum + x[j] · A[p[i]][j]
15         end
16         x[i] ← (z[i] − sum)/A[p[i]][i]
17     end
18     return x
```

$\ell = 4$. Rezultaty widoczne są na wykresach (Rysunek 2 i 3). Sprawdzono także, jak prezentują się czasy rozwiązywania układów równań dla tej samej macierzy i różnych prawych stron z użyciem metody Gaussa i rozkładu $\boldsymbol{LU}$ (Rysunek 1 (b)), a także porównano czasy obliczeń metody eliminacji Gaussa z jej standardową wersją (Rysunek 1 (a)).

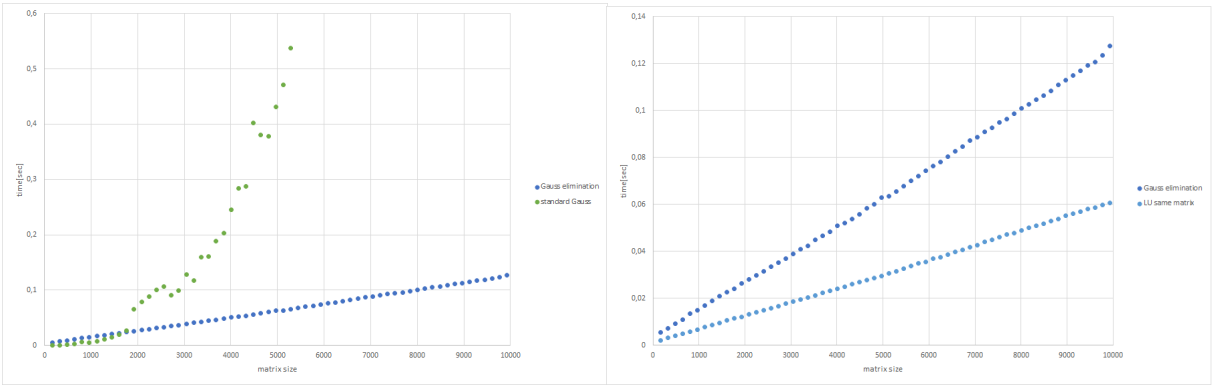
	Rozmiar macierzy		
metoda	$n = 16$	$n = 10000$	$n = 50000$
Gauss	$6.383\,782\,391\,594\,657 \times 10^{-16}$	$1.040\,205\,776\,165\,185 \times 10^{-13}$	$1.932\,760\,415\,264\,822 \times 10^{-13}$
Gauss z wyborem	$3.065\,703\,529\,144\,609 \times 10^{-16}$	$5.320\,922\,653\,862\,027 \times 10^{-16}$	$5.145\,451\,947\,624\,984 \times 10^{-16}$
$\boldsymbol{LU}$	$8.794\,620\,319\,638\,098 \times 10^{-16}$	$8.968\,695\,564\,214\,933 \times 10^{-15}$	$1.860\,501\,163\,975\,474\,5 \times 10^{-13}$
$\boldsymbol{LU}$ z wyborem	$2.288\,783\,399\,261\,118 \times 10^{-16}$	$5.186\,001\,368\,086\,655 \times 10^{-16}$	$5.031\,405\,319\,253\,065 \times 10^{-16}$

Tabela 1: Wartość błędu względnego rozwiązania układu równań z macierzami różnych wielkości dla implementowanych metod.

4. Wnioski

Poprawność działania zaimplementowanych metod widoczna jest w Tabeli 1 — uzyskano bardzo niewielkie wartości błędów względnych, przy czym są one zauważalnie mniejsze są dla algorytmów z częściowym wyborem elementu głównego.

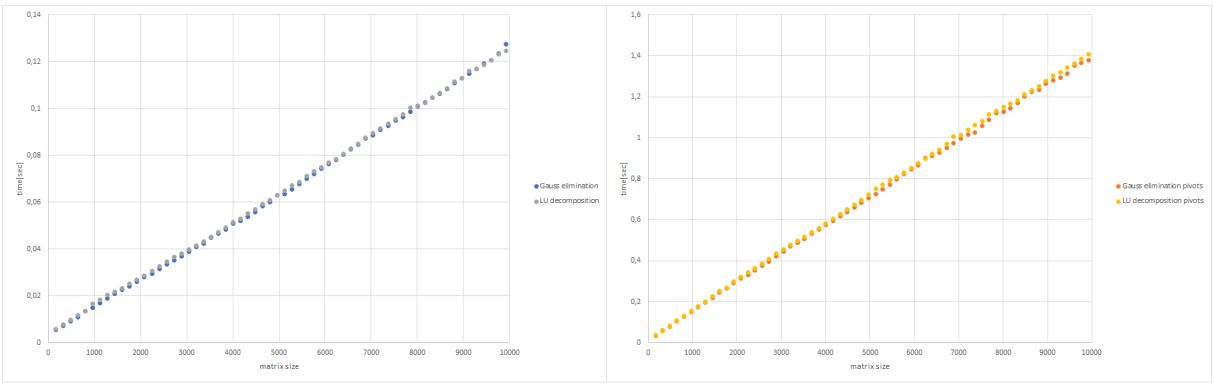
Analiza Rysunku 1 (a) pozwala potwierdzić, że przez dostosowanie algorytmu do charakterystycznej postaci macierzy można uzyskać znacznie lepszą złożoność obliczeniową (w tym przypadku udało się uzyskać $O(n)$ z metody o złożoności sześciennej).



(a) Standardowa eliminacja Gaussa a metoda dostosowana do postaci macierzy

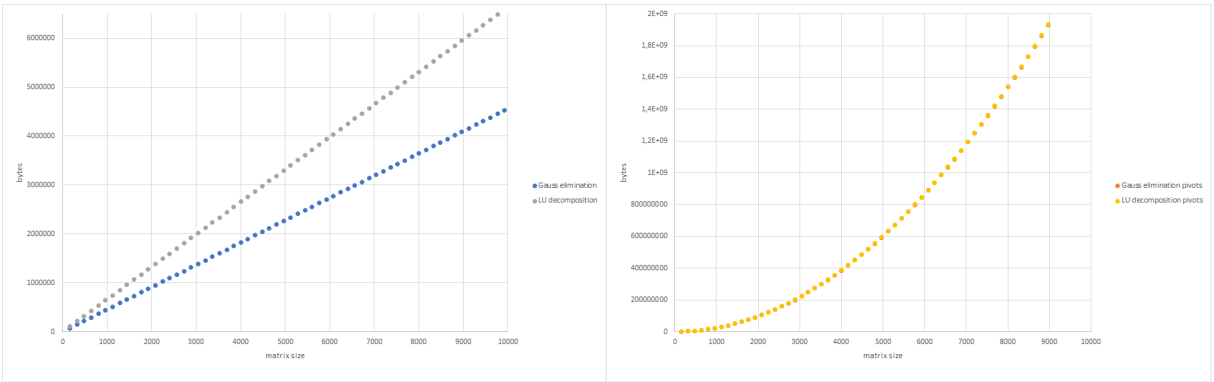
(b) Czas rozwiązywania jednego z układów równań dla tej samej macierzy (100 prób)

Rysunek 1



(a) (b)

Rysunek 2: Złożoność czasowa eliminacji Gaussa i rozkładu LU bez (a) oraz z (b) wyborem elementu głównego.



(a) (b)

Rysunek 3: Złożoność pamięciowa eliminacji Gaussa i rozkładu LU bez (a) oraz z (b) wyborem elementu głównego.

Wykresy na Rysunku 2 pokazują, że implementacja rozwiązania zadanego problemu ma wymaganą liniową złożoność obliczeniową. Uzyskane czasy dla metody eliminacji Gaussa pokrywają się z rezultatami przy wykorzystaniu rozkładu ***LU*** oraz analogicznie ma to miejsce dla odpowiadających metod z częściowym wyborem elementu głównego.

Zauważalny jest niemal 12-krotnie dłuższy średni czas rozwiązywania metodami z wyborem elementu głównego niż w przypadku ich standardowych wersji. Jednak pomimo tak drastycznej różnicy warto mieć na uwadze, że wykorzystanie tych wolniejszych wariantów niekiedy może być nieuniknione (np. przy próbie rozwiązania układów z zerami na diagonali).

Złożoność pamięciowa widoczna na Rysunku 3 jest liniowa dla funkcji bez wyboru elementu głównego oraz kwadratowa dla ich wolniejszych wersji — wynika to najprawdopodobniej z konieczności wypełniania macierzy powodowanego permutacjami wierszy.

Rysunek 1 (b) jasno przedstawia, że wykorzystanie rozkładu $\boldsymbol{LU}$ jest o wiele efektywniejsze w przypadku rozwiązywania wielu układów równań z tą sama macierzą. Znajduje to swoje teoretyczne potwierdzenie — etap rozkładu macierzy (eliminacja Gaussa) jest wtedy wykonywany jedynie raz, pozwalając na znaczne obniżenie całkowitego czasu rozwiązywania.

Problem w trafny sposób pokazuje, że algorytmy rozwiązywania układów równań można zmodyfikować w taki sposób, aby, dostosowane do struktury zadanej macierzy, osiągały o wiele lepszą złożoność obliczeniową.