Obliczenia naukowe

Prowadzący: dr hab. Paweł Zieliński

wtorek TN 7^{30}

Sprawozdanie 5

Weronika Jasiak

236733

1. Opis problemu

Rozwiązanie układu równań liniowych:

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

dla danej macierzy współczynników $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{n}}$ i wektora prawych stron $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{n}}$. Macierz $\boldsymbol{A}$ jest rzadką, tj. mającą dużo elementów zerowych, i blokową o następującej strukturze:

$$\boldsymbol{A} = \begin{pmatrix} A_1 & C_1 & 0 & 0 & 0 & \cdots & 0 \\ B_2 & A_2 & C_2 & 0 & 0 & \cdots & 0 \\ 0 & B_3 & A_3 & C_3 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & B_{v-2} & A_{v-2} & C_{v-2} & 0 \\ 0 & \cdots & 0 & 0 & B_{v-1} & A_{v-1} & C_{v-1} \\ 0 & \cdots & 0 & 0 & 0 & B_v & A_v \end{pmatrix}, \tag{1}$$

$v = \frac{n}{l}$, zakładając, że n jest podzielne przez l, gdzie l jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych (bloków): $\boldsymbol{A_k}, \boldsymbol{B_k}, \boldsymbol{C_k}$. Mianowicie:

1. $\boldsymbol{0}$ jest kwadratową macierzą zerową stopnia l,
2. $\boldsymbol{A_k} \boldsymbol{\epsilon} \mathbb{R}^{\boldsymbol{l}\times\boldsymbol{l}}$, $k = 1, \dots, v$ jest macierzą gęstą,
3. $\boldsymbol{B_k} \boldsymbol{\epsilon} \mathbb{R}^{\boldsymbol{l}\times\boldsymbol{l}}$, $k = 2, \dots, v$ jest następującej postaci:

$$\boldsymbol{B_k} = \begin{pmatrix} 0 & \cdots & 0 & b_1^k \\ 0 & \cdots & 0 & b_2^k \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & b_l^k \end{pmatrix},$$

4. $\boldsymbol{C_k} \boldsymbol{\epsilon} \mathbb{R}^{\boldsymbol{l}\times\boldsymbol{l}}$, $k = 1, \dots, v-1$ jest macierzą diagonalną:

5. $$\boldsymbol{C_k} = \begin{pmatrix} c_1^k & 0 & 0 & \cdots & 0 \\ 0 & c_2^k & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c_{l-1}^k & 0 \\ 0 & \cdots & 0 & 0 & c_l^k \end{pmatrix}.$$

2. Rozwiązanie

Efektywne rozwiązanie problemu $\boldsymbol{Ax} = \boldsymbol{b}$, gdzie $\boldsymbol{A}$ jest postaci (1) wiąże się z odpowiednim sposobem przechowywania elementów macierzy rzadkiej. Przyglądając się bliżej jej budowie można zauważyć, że zaledwie $nl + 2(n-l)$ jest niezerowych, co więcej, aż $n^2 - (nl + 2(n-l))$ elementów przyjmuje wartość równą 0. Przedstawienie tej dysproporcji wygląda następująco:

$$n^2 > n^2 - \big(nl + 2(n-l)\big) > nl + 2(n-l),$$

zatem pamiętanie macierzy $\boldsymbol{A}$ jako tablicy dwuwymiarowej rozmiaru $n \times n$ jest najmniej efektywnym rozwiązaniem.

W bibliotece `SparseArrays` dostępnej w języku `Julia` znajduje się struktura `SparseMatrixCSC`, z której skorzystano przy implementacji rozwiązania powyższego problemu. Pozwala ona na skompresowane przechowywanie elementów w porządku kolumnowym, co umożliwia łatwy i szybki dostęp do elementów, w przeciwieństwie do porządku wierszowego.

Jednak algorytm eliminacji Gaussa wykonuje kolejno operacje elementarne na wierszach, zatem należy wykonać modyfikację indeksowania kolumn i wierszy poprzez ich zamianę miejscami.

Kolejnym ważnym aspektem jest adaptacja standardowych algorytmów dostępnych w bibliotece języka `Julia`, do specyficznych wymagań macierzy trójprzekątniowej. Ograniczając w odpowiedni sposób zakres obliczeń można zredukować zarówno czas z $O(n^3)$ do $O(n)$ dla stałej l, jaki i wykorzystanie pamięci, przy założeniu, że dostęp do elementu macierzy jest w czasie stałym (powszechnie wiadomo, że tak nie jest).

2.1. Metoda Eliminacji Gaussa

Metoda eliminacji Gaussa to algorytm pozwalający rozwiązywać układy równań liniowych, lecz nie jest to jedyna rzecz, którą można wyznaczyć dzięki tej metodzie. Pozwala ona również znaleźć rząd macierzy, wyznacznik macierzy, obliczyć macierz odwrotną oraz wyznaczyć rozkład LU.

Wykonując redukcję wiersza macierzy wykorzystuje się sekwencję elementarnych operacji, aby zmodyfikować macierz w taki sposób, żeby dolny lewy róg został wypełniony zerami. Istnieją trzy typy podstawowych operacji na wierszach:

1. zamiana dwóch rzędów,
2. mnożenie wiersza przez niezerową wartość,
3. dodawanie wielokrotności jednego wiersza do innego wiersza.

Rozwiązanie układu równań metodą eliminacji Gaussa opiera się na dwóch etapach. Pierwszy z nich polega na przekształceniu macierzy współczynników $\boldsymbol{A}$ do postaci macierzy trójkątnej górnej. Następuje to poprzez przypisywanie 0 kolejnym wartościom znajdującym się poniżej głównej przekątnej macierzy. Elementy przekształcamy zgodnie z zależnością:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\left(a_{kj}^{(k)}\right),$$

gdzie:
$k = 1, 2, \dots, n-1$ oznacza bieżący krok eliminowanego układu,
$i = k+1, k+2, \dots, n,\ \ j = k+1, k+2, \dots, n$ oznaczają indeksy wierszy i kolumn.
Warto zauważyć, że gdy $a_{kk}^{(k)} = 0$ należy przestawić wiersze, tak aby dla elementów głównej przekątnej eliminowanego równania zachodziła zależność $a_{kk}^{(k)} \neq 0$. Z kolei drugi etap rozwiązania polega na znalezieniu wartości niewiadomych x_i. W tym celu dla przekształconej macierzy współczynników zostaje zastosowany *algorytm podstawiania wstecz*:

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j\right),$$

gdzie:
$i = n, n-1, \dots, 1$ oznacza kolejne wierze macierzy.

Liczba operacji arytmetycznych niezbędnych do rozwiązania układu n równań metodą eliminacji Gaussa wynosi $O(n^3)$.

2.1.1. Rozwiązanie układu równań $\boldsymbol{Ax = b}$

Macierz $\boldsymbol{A}$ jest macierzą trójprzekątniową, oznacza to, że posiada szczególną strukturę, więc należy zastosować algorytm, który ją uwzględnia. Pozwoli to zoptymalizować liczbę operacji arytmetycznych potrzebnych do uzyskania prawidłowego wyniku, w przeciwieństwie do zastosowania podstawowej wersji algorytmu eliminacji Gaussa. Zakłada ona, wyzerowanie wszystkich elementów

znajdujących się poniżej głównej przekątnej, co w tym przypadku jest to całkowicie zbędne.

Algorytm 1: Eliminacja Gaussa

Input: (A, b, n, l), gdzie:
- A - rzadka macierz
- b - wektor prawych stron $\boldsymbol{b}$
- n - rozmiar macierzy $\boldsymbol{A}$
- l - rozmiar macierzy $\boldsymbol{A_k}, \boldsymbol{B_k}, \boldsymbol{C_k}$

Output: x, gdzie:
- x - wektor x o długości n będący rozwiązaniem równania $\boldsymbol{Ax = b}$

GAUSSIANELIMINATION (A, b, n, l)
for $k \leftarrow 1$ **to** $n - 1$ **do**
  $endRow \leftarrow l + l * \left\lfloor \frac{k}{l} \right\rfloor$
  if $endRow > n$
    $endRow \leftarrow n$
  end if
  $endColumn \leftarrow k + l$
  if $endColumn \geq n$
    $endColumn \leftarrow n$
  end if
  for $i \leftarrow k + 1$ **to** $endRow$ **do**
    if $|A_{kk}| < \varepsilon$
      error $Błąd: znaleziony\ element\ jest\ zerem$
    end if
    $z \leftarrow \frac{A_{ik}}{A_{kk}}$
    for $j \leftarrow k + 1$ **to** $endColumn$ **do**
      $A_{ij} \leftarrow A_{ij} - z * A_{kj}$
    end for
    $b_i \leftarrow b_i - z * b_k$
  end for
end for
for $i \leftarrow n$ **to** 1 **step** -1 **do**
  $endColumn \leftarrow i + l$
  if $endColumn > n$
    $endColumn \leftarrow n$
  end if
  $z \leftarrow b_i$
  for $j \leftarrow i + 1$ **to** $endColumn$ **do**
    $z \leftarrow z - A_{ij} * x_j$
  end for
  if $|A_{ii}| < \varepsilon$
    error $Błąd: znaleziony\ element\ jest\ zerem$
  end if
  $x_i \leftarrow \frac{z}{A_{ii}}$
end for
return x

W pierwszych $l - 3$ kolumnach elementy znajdujące się w pierwszych l wierszach przyjmują wartości różne od 0, odpowiadają one trzem kolumnom kwadratowej macierzy wewnętrznego bloku A_1. W następnych l kolumnach elementy znajdujące się w pierwszych $2l$ wierszach przyjmują również wartości różne od 0, które odpowiadają następująco ostatniej kolumnie B_2 oraz trzem kolumnom kwadratowej macierzy wewnętrznego bloku A_2. Analizując kolejne przejścia, można zauważyć schemat, mianowicie dla kolejnych l kolumn elementy znajdujące się w pierwszych xl wierszach przyjmują wartości różne od 0, dla $x \in \left(1, \frac{n}{l}\right)$. W ostatniej n-tej kolumnie ostatnie l

wierszy należy do kwadratowej macierzy wewnętrznego bloku $\boldsymbol{A_v}$. Zatem można ograniczyć przebieg iteracji do maksymalnego indeksu kolumny, która przyjmuje wartość różną od 0 dla danego wiersza, czyli :

$$ostatniaKolumna(wiersz) = min\{wiersz + l, n\}.$$

Z kolei indeks ostatniego elementu przyjmującego wartość różną od 0 danej kolumny można wyznaczyć ze wzoru:

$$wiersz(kolumna) = min\left\{ l * \left\lfloor \frac{kolumna}{l} \right\rfloor, n \right\}.$$

Należy pamiętać o tym, że podczas odejmowania wiersza x od wiersza $x + 1$, dla $x \in (1, n)$ następuje zmiana wyłącznie kolumn spełniających zależność $column > endColumn(x)$. W każdym wierszu, nie uwzględniając l ostatnich elementów, ostatnia wartość różna od 0 zawiera się w głównej przekątnej kwadratowej macierzy wewnętrznego bloku $\boldsymbol{C_i}$,a elementy te są stale odległe od elementów głównej przekątnej macierzy $\boldsymbol{A}$. Otrzymana w ten sposób macierz trójkątna górna umożliwia skorzystanie z wyżej opisanego *algorytm podstawiania wstecz*.

Dla stałej l liczba operacji arytmetycznych niezbędnych do rozwiązania układu n równań metodą eliminacji Gaussa wynosi $O(n)$, wynika to ze złożoności obliczeniowej metody, której realizację przedstawia Algorytm 1.

2.1.2. Rozwiązanie układu równań $\boldsymbol{Ax = b}$ z częściowym wyborem elementu głównego

Algorytm eliminacji Gaussa opisany powyżej nie uwzględnia kilku rodzajów układów, które w prosty sposób można rozwiązać:

1. gdy pierwszym elementem głównym jest 0,
2. gdy pierwszym elementem głównym jest ε – dostatecznie mała liczba różna od 0, dla której wartości obu różnic nie są identyczne z dokładnym rozwiązaniem, co w konsekwencji prowadzi do otrzymania fałszywego wyniku,
3. gdy współczynnik a_{ii} jest mały w porównaniu z innymi elementami pierwszego wiersza, co również w konsekwencji prowadzi do otrzymania fałszywego wyniku.

Prowadzi to do wniosku, że algorytm powinien również uwzględniać przestawianie równań w zadanym układzie, gdy okoliczności wymuszają taką sytuację. Pierwszym sposobem, który nasuwa się na myśl jest przestawianie wierszy w pamięci komputera, lecz nie jest to najbardziej efektywna metoda, ponieważ wydłuża ona czas obliczeń. Zatem należy znaleźć bardziej optymalną metodę, którą jest wybieranie kolejnych wierszy o wskaźnikach $p_1, p_2, \dots, p_{n-1}$, gdzie $(p_1, p_2, \dots, p_n)$ jest pewną permutacją zbioru $(1,2, \dots, n)$. Następnie odejmując wielokrotności wiersza p_2 od wierszy o wskaźnikach $p_3, p_4, \dots, p_n$ powtarzamy wykonywaną operację zwiększając indeks wskaźnika.

Algorytm eliminacji Gaussa z częściowym wyborem elementu głównego jest bardzo podobny do Algorytmu 1. z tym, że odwołanie do konkretnego wiersza zostało zastąpione odwołaniem do odpowiadającej mu pozycji w wektorze permutacji.

Wykonując eliminację współczynników z pierwszych $l - 3$ kolumn można utworzyć element różny od 0 jedynie w kolumnie z indeksem $2l$, następuje to poprzez odejmowanie wiersza l, który w danej kolumnie zawiera element różny od 0. Analizując kolejne kroki można zauważyć schemat, eliminując współczynniki z kolejnych l kolumn element różny od 0 znajdujący się najdalej można utworzyć jedynie w kolumnie z indeksem $3l$ przez odjęcie wiersza $2l$, który w tej kolumnie zawiera element różny od 0.

Algorytm 2: Eliminacja Gaussa z częściowym wyborem elementu głównego

Input: (A, b, n, l), gdzie:

A - rzadka macierz
b - wektor prawych stron $\boldsymbol{b}$
n - rozmiar macierzy $\boldsymbol{A}$
l - rozmiar macierzy $\boldsymbol{A_k}, \boldsymbol{B_k}, \boldsymbol{C_k}$

Output: x, gdzie:

x - wektor x o długości n będący rozwiązaniem równania $\boldsymbol{Ax = b}$

GAUSSIANELIMINATIONWITHPARTIALPIVOTING (A, b, n, l)
$p \leftarrow \{1, \dots, n\}$
for $k \leftarrow 1$ **to** $n - 1$ **do**
  $endRow \leftarrow l + l * \left\lfloor \frac{k}{l} \right\rfloor$
  if $endRow > n$
    $endRow \leftarrow n$
  end if
  $endColumn \leftarrow 2 * endRow$
  if $endColumn > n$
    $endColumn \leftarrow n$
  end if
  for $i \leftarrow k + 1$ **to** $endRow$ **do**
    $m = \left|A_{p_k k}\right|$
    $r = k$
    for $w = i$ **to** $endRow$ **do**
      if $\left|A_{p_w k}\right| > m$
        $m = A_{p_w k}$
        $r = w$
      end if
    end for
    if $\left|A_{pkk}\right| < \varepsilon$
      error $Błąd: znaleziony\ element\ jest\ zerem$
    end if
    $temp = p_k$
    $p_k = p_r$
    $p_r = temp$
    $z \leftarrow \frac{A_{p_i k}}{A_{p_k k}}$
    for $j \leftarrow k + 1$ **to** $endColumn$ **do**
      $A_{p_i j} \leftarrow A_{p_i j} - z * A_{p_k j}$
    end for
    $b_{p_i} \leftarrow b_{p_i} - z * b_{p_k}$
  end for
end for
for $i \leftarrow n$ **to** 1 **step** -1 **do**
  $endColumn \leftarrow 2 * l + l * \left\lfloor \frac{p_i}{l} \right\rfloor$
  if $endColumn \geq n$
    $endColumn \leftarrow n$
  end if
  $z \leftarrow b_{p_i}$
  for $j \leftarrow i + 1$ **to** $endColumn$ **do**
    $z \leftarrow z - A_{p_i j} * x_j$
  end for
  if $\left|A_{p_i i}\right| < \varepsilon$
    error $Błąd: znaleziony\ element\ jest\ zerem$
  end if
  $x_i \leftarrow \frac{z}{A_{p_i i}}$
end for
return x

Zatem można ograniczyć przebieg iteracji do maksymalnego indeksu kolumny, która przyjmuje wartość różną od 0 dla danego wiersza, czyli :

$$ostatniaKolumna(wiersz) = min\left\{2l + l * \left\lfloor\frac{wiersz}{l}\right\rfloor + l, n\right\}.$$

Z kolei indeks ostatniego elementu przyjmującego wartość różną od 0 danej kolumny można wyznaczyć ze wzoru:

$$wiersz(kolumna) = min\left\{ l + l * \left\lfloor\frac{kolumna}{l}\right\rfloor, n\right\}.$$

Dla stałej l liczba operacji arytmetycznych niezbędnych do rozwiązania układu n równań metodą eliminacji Gaussa z częściowym wyborem elementu głównego wynosi $O(n)$, wynika to ze złożoności obliczeniowej metody, której realizację przedstawia Algorytm 2.

2.2. Rozkład LU

Metoda LU to metoda rozwiązywania układu równań liniowych $\boldsymbol{A} = \boldsymbol{LU}$. Polega ona na utworzeniu dwóch macierzy trójkątnych, tj. macierzy trójkątnej dolnej – $\boldsymbol{L}$ oraz macierzy trójkątnej górnej – $\boldsymbol{U}$. Poniżej znajdują się przykłady takich macierzy:

$$\boldsymbol{L} = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix}, \qquad \boldsymbol{U} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}.$$

Wyznaczenie rozkładu $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ metodą eliminacji Gaussa opiera się na przekształceniu macierzy $\boldsymbol{A}$ do postaci trójkątnej górniej - $\boldsymbol{U}$ i zapamiętaniu mnożników wyliczanych ze wzoru:

$$z_{ij} = \frac{a_{ij}}{a_{jj}},$$

gdzie i to to wiersze, a j to kolumny macierzy $\boldsymbol{L}$. W następnym kroku obliczone mnożnik z_{ij} posłużą do utworzenia macierzy trójkątnej dolnej – $\boldsymbol{L}$.

Liczba operacji arytmetycznych niezbędnych do rozwiązania układu n równań poprzez wyznaczenie rozkładu $\boldsymbol{LU}$ metodą eliminacji Gaussa wynosi $O(n^3)$. Jednak wyznaczając rozwiązanie przy pomocy układu:

$$Ly = b,$$
$$Ux = y,$$

koszt wykonywania powyższych operacji maleje i wynosi jedynie $O(n^2)$.

2.2.1. Rozwiązanie układu równań $\boldsymbol{A} = \boldsymbol{LU}$

Wyznaczenie rozkładu $\boldsymbol{LU}$ dla danej macierzy $\boldsymbol{A}$ opiera się na wykorzystaniu metody eliminacji Gaussa opisanej w Algorytmie 1, następnie odpowiednio zmodyfikowanej, tak aby zamiast zerować elementy a_{ij} przypisywała im wartość:

$$z = \frac{a_{ij}}{a_{jj}}.$$

Dla stałej l złożoność algorytmu wynosi $O(n)$ tak jak metody eliminacji Gaussa.

2.2.2. Rozwiązanie układu równań $\boldsymbol{A = LU}$ z częściowym wyborem elementu głównego

Częściowy wybór elementu głównego podczas wyznaczania rozkładu $\boldsymbol{LU}$ przebiega podobnie do metody eliminacji Gaussa z częściowym wyborem elementu głównego opisanej w Algorytmie 2. Jednak metoda ta zwraca wektor permutacji p otrzymany w wyniku przestawień wierszy jakie zostały wykonane podczas działania algorytmu.

Dla stałej l liczba operacji arytmetycznych niezbędnych do rozwiązania układu n równań z częściowym wyborem elementu głównego wynosi $O(n)$.

2.2.3. Rozwiązanie układu równań $\boldsymbol{LUx = b}$

Rozwiązanie układu równań:

$$Ly = b,$$
$$Ux = y,$$

opiera się na wyznaczeniu macierzy trójkątnej górnej i dolnej spełniającej założenia. Podczas implementacji metody skorzystano z *algorytmu podstawiania w przód* oraz *algorytmu podstawiania wstecz* przy jednoczesnej optymalizacji wykonywanych operacji, na co pozwoliły zastosowane ograniczenia dla kolumn w obu algorytmach.

Zatem można ograniczyć przebieg iteracji podczas *podstawiania wstecz* do maksymalnego indeksu kolumny, która przyjmuje wartość różną od 0 dla danego wiersza, czyli :

$$ostatniaKolumna(wiersz) = min\{wiersz + l, n\},$$

z kolei podczas *podstawiania w przód* liczba operacji zostaje zminimalizowana do:

$$ostatniaKolumna(wiersz) = min\left\{ l * \left\lfloor \frac{wiersz-1}{l} \right\rfloor, n \right\}.$$

Algorytmy wchodzące w skład powyższej metody wykonują $O(l)$ operacji dla zadanego wiersza, zatem koszt rozwiązania układu równań $\boldsymbol{LUx = b}$ dla znanego rozkładu $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ wynosi $O(n)$. Szczegółowa realizacja przebiegu algorytmu rozwiązującego układ równań $\boldsymbol{Ax = b}$, która uwzględnia specyficzną postać macierzy $\boldsymbol{A}$, gdy wcześniej został wyznaczony już rozkład $\boldsymbol{LU}$ została przedstawiona w Algorytmie 3.

2.2.4. Rozwiązanie układu równań $\boldsymbol{LUx = b}$ z częściowym wyborem elementu głównego

Opisane powyżej działania wyglądają analogicznie w przypadku rozwiązania układu równań $\boldsymbol{LUx = b}$ z częściowym wyborem elementu głównego.

Jak wcześniej miało to już miejsce algorytmy korzystające z przestawiania wierszy posiadają inne ograniczenia spowodowane wykorzystaniem wskaźnika p_i pewnej permutacji zbioru $(1,2,\dots,n)$. Dla *algorytmu podstawiania wstecz* liczba operacji zostaje ograniczona do:

$$ostatniaKolumna(wiersz) = min\left\{2l + l * \left\lfloor \frac{wiersz}{l} \right\rfloor, n\right\},$$

zaś dla *algorytmu podstawiania w przód* :

$$ostatniaKolumna(wiersz) = min\left\{l * \left\lfloor \frac{wiersz-1}{l} \right\rfloor, n\right\}.$$

3. Wyniki

Dla wcześniej zaimplementowanej funkcji eliminacji Gaussa otrzymano następujące wyniki (Tabela 1.) .

rozmiar macierzy	eliminacja Gaussa	eliminacja Gaussa z częściowym wyborem elementu głównego
16 × 16	3.6912828624410226e-15	5.10280049072227e-16
10000 × 10000	7.997607911218936e-14	3.905334752145677e-16
5000 × 5000	2.6390293711611795e-14	4.2015187204731675e-16

Tabela 1. Wartości błędów względnych otrzymane podczas wywołania funkcji `gaussianElimination()` dla macierzy o zadanych rozmiarach.

Dla wcześniej zaimplementowanej funkcji wyznaczającej rozkład LU metodą eliminacji Gaussa, następnie rozwiązującej układ $\boldsymbol{Ax = b}$ otrzymano następujące wyniki (Tabela 2.) .

rozmiar macierzy	rozkład LU	rozkład LU z częściowym wyborem elementu głównego
16 × 16	3.6912828624410226e-15	5.10280049072227e-16
10000 × 10000	7.997607911218936e-14	3.905334752145677e-16
5000 × 5000	2.6390293711611795e-14	4.2015187204731675e-16

Tabela 2. Wartości błędów względnych otrzymane podczas wywołania funkcji `gaussianEliminationLU()`, a następnie funkcji `decompositionOfLU()` dla macierzy o zadanych rozmiarach.

Dla wcześniej zaimplementowanej funkcji eliminacji Gaussa oraz generowanych macierzy otrzymano następujące wyniki (Tabela 3.) .

rozmiar macierzy	eliminacja Gaussa		eliminacja Gaussa z częściowym wyborem elementu głównego	
	Czas[s]	Pamięć	Czas[s]	Pamięć
1000 × 1000	0.004429	7.938 KiB	0.003450	15.906 KiB
2000 × 2000	0.007831	15.750 KiB	0.007818	31.531 KiB
3000 × 3000	0.015525	23.516 KiB	0.018403	47.063 KiB
4000 × 4000	0.027620	31.328 KiB	0.031406	62.688 KiB
5000 × 5000	0.043609	1.038 MiB	0.049104	1.077 MiB
6000 × 6000	0.060471	46.953 KiB	0.065968	93.938 KiB
7000 × 7000	0.102507	54.766 KiB	0.094176	109.563 KiB
8000 × 8000	0.115072	62.578 KiB	0.116955	125.188 KiB
9000 × 9000	0.167728	2.069 MiB	0.185064	2.138 MiB
10000 × 10000	0.182877	2.076 MiB	0.188434	2.153 MiB

Tabela 3. Wyniki otrzymane podczas wywołania funkcji `gaussianElimination()` dla macierzy generowanych przy pomocy funkcji `blockmat()` o zadanych rozmiarach.

4. Wnioski

Wyniki znajdujące się w Tabeli 1. jednoznacznie wskazują na to, że metoda eliminacji Gaussa z częściowym wyborem elementu głównego generuje mniejsze błędy względne niż podstawowa wersja eliminacji Gaussa, jednak w obu przypadkach są one nieznaczne. Porównując wyniki uzyskane w Tabeli 1. i Tabeli 2. można śmiało stwierdzić, że są one identyczne, a więc wyznaczając wcześniej rozkład LU, a następnie rozwiązując rozkład LU otrzymujemy identyczne wartości. Metoda ta jest zatem najbardziej optymalna, gdy posiadamy jedną macierz, a wiele wektorów prawych stron, ponieważ w tym przypadku wyznaczanie rozkładu macierzy wykonywane jest tylko raz, co znacznie obniża czas jej wykonywania. Pozwala to jednoznacznie stwierdzić, że metody

zostały zaimplementowane w sposób poprawny, a wartości otrzymane są niemalże identyczne względem rzeczywistego rozwiązania.

Wykonując eksperyment, który polegał na generowaniu macierzy przy pomocy funkcji `blockmat()` uzyskano wyniki, które pokazują różnice pomiędzy podstawową wersją algorytmu, a tą, która wymaga częściowego wyboru elementu głównego. Korzystanie z rozbudowanej wersji algorytmu jest obciążone jego dłuższym czasem wykonywania oraz większym zużyciem pamięci. Jednak wolniejsze nie zawsze znaczy gorsze, ponieważ ta wersja algorytmu pozwala uzyskać wynik, gdy na głównej przekątnej znajdują się 0, w przeciwieństwie do książkowej wersji algorytmu.

Modyfikując odpowiednio ilość iteracji można zmniejszyć złożoność algorytmu z sześciennej do zaledwie $O(n)$. Zatem odpowiednie ograniczenia zastosowane w metodach obliczających układy równań pozwalają dopasować się do specyficznej struktury macierzy, co pozwala uzyskać lepszą złożoność obliczeniową.