Obliczenia Naukowe - Lista nr 5

Maksymilian Piotrowski

1 Zadanie 1

Opis problemu: Należy napisać funkcję rozwiązującą układ $Ax = b$ metodą eliminacji Gaussa uwzględniając specyficzną postać macierzy **A** opisaną na liście zadań. Zadanie wykonać dla dwóch wariantów:
(a) bez wyboru elementu głównego
(b) z częściowym wyborem elementu głównego

Rozwiązanie:

$$\begin{pmatrix} A_1 & C_1 & 0 & 0 & 0 & 0 & ... & 0 \\ B_2 & A_2 & C_2 & 0 & 0 & 0 & ... & 0 \\ 0 & B_3 & A_3 & C_3 & 0 & 0 & ... & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & ... & \vdots \\ 0 & \cdots & 0 & B_{v-3} & A_{v-3} & C_{v-3} & 0 & 0 \\ 0 & \cdots & 0 & 0 & B_{v-2} & A_{v-2} & C_{v-2} & 0 \\ 0 & \cdots & 0 & 0 & 0 & B_{v-1} & A_{v-1} & C_{v-1} \\ 0 & \cdots & 0 & 0 & 0 & 0 & B_v & A_v \end{pmatrix}$$

Rysunek 1: Struktura macierzy **A**.

Adresując specyficzną postać macierzy **A** w kontekście złożoności pamięciowej, przechowuje ją w następującej strukturze:

SpecialMatrix {
n - rozmiar macierzy **A**
l - rozmiar macierzy A_i, B_i, C_i
v - $\frac{n}{l}$
A - tablica zawierająca macierze A_i, i = 1,..,v
B - tablica zawierająca macierze B_i, i = 2,..,v
C - tablica zawierająca macierze C_i, i = 1,..,v-1
}

Aby pobrać ze struktury SpecialMatrix element $\mathbf{A}_{ij}$ macierzy $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$ sprawdzam w której macierzy wewnętrznej $A_k, B_k, C_k, 0$ znajduje się $\mathbf{A}_{ij}$, obliczając $i' = \lfloor \frac{i-1}{l} + 1 \rfloor$ oraz $j' = \lfloor \frac{j-1}{l} + 1 \rfloor$.

Jeśli $i' = j'$, to znaczy że macierz wewnętrzna leży na przekątnej, czyli to macierz $A_{i'}$. Jeśli $i' = j' + 1$, to leży pod przekątną, jest to $B_{j'}$. Analogicznie jeśli $i' + 1 = j'$, to macierz wewnętrzna jest macierzą $C_{i'}$. W pozostałych przypadkach macierz wewnętrzna jest jedną z macierzy zerowych.

Aby sprawdzić który konkretnie element macierzy wewnętrznej, znanej z obliczonych i', j', o wymiarach $l \times l$ jest pobierany, można obliczyć $i'' = (i-1) \mod l + 1$ oraz $j'' = (j-1) \mod l + 1$. Dla macierzy zerowych nie ma to znaczenia, po prostu zwracamy 0.

Dane:
A - macierz zawarta w strukturze SpecialMatrix
i,j - koordynaty elementu do pozyskania
Wynik:
$\mathbf{A}_{ij}$ - pożądany element macierzy **A** .

Algorytm 1 Algorytm pobierający element $\mathbf{A}_{ij}$ macierzy **A** zapisanej w strukturze SpecialMatrix

$i' \leftarrow \lfloor \frac{i-1}{l} + 1 \rfloor$
$j' \leftarrow \lfloor \frac{j-1}{l} + 1 \rfloor$
$i'' \leftarrow (i-1) \mod l + 1$
$j'' \leftarrow (j-1) \mod l + 1$
if $i' = j'$ **then**
  return **A**.A$[i'][i'', j'']$
else if $i' = j' + 1$ **then**
  return **A**.B$[j'][i'', j'']$
else if $i' + 1 = j'$ **then**
  return **A**.C$[i'][i'', j'']$
else
  return 0
end if

Analogicznie wygląda proces dostępu do elementów przy modyfikacji macierzy **A**. Kiedy w kolejnych Algorytmach będę pisał "$\mathbf{A}_{ij} \leftarrow x$" lub "$x \leftarrow \mathbf{A}_{ij}$" mam na myśli pobranie/modyfikacje elementu $\mathbf{A}_{ij}$ struktury SpecialMatrix **A** w opisany wyżej sposób (Algorytm 1).

Struktura SpecialMatrix daje nam złożoność pamięciową $O(v * l^2)$, czyli $O(n)$ jeżeli l jest stałą. Ponadto dostęp do elementów struktury jest możliwy w czasie liniowym (Algorytm 1). Możemy więc wykonać na niej opisany na wykładzie algorytm implementujący eliminację Gaussa ze złożonością czasową $O(n^3)$ (Algorytm 2).

Metoda eliminacji Gaussa polega na zerowaniu elementów macierzy znajdujących się pod przekątną wykonując na macierzy operacje elementarne. W k-tym kroku algorytmu dodajemy do rzędów $\mathbf{i} > \mathbf{k}$ taką wielokrotność rzędu **k**, żeby wartość A_{ik} zmieniła się na 0 (mnożymy rząd k-ty przez $\frac{\mathbf{A}_{ik}}{\mathbf{A}_{kk}}$). W k-tym kroku zerujemy więc k-tą kolumnę pod przekątną. W ten sposób otrzymujemy macierz górno trójkątną.

Dane:
A - macierz współczynników równań pamiętana w strukturze SpecialMatrix
b - wektor prawych stron równań
Wynik:
A',b' - równoważny układ równań z **A'** w formie macierzy trójkątnej.

Algorytm 2 Standardowy algorytm wyprowadzający macierz górną trójkątną **A'** przy pomocy metody eliminacji Gaussa - Złożoność czasowa $O(n^3)$

for $k \leftarrow 1$ to $n-1$ **do**
  for $i \leftarrow k+1$ to n **do**
    $z \leftarrow \frac{\mathbf{A}_{ik}}{\mathbf{A}_{kk}}$
    for $j \leftarrow k+1$ to n **do**
      $\mathbf{A}_{ij} \leftarrow \mathbf{A}_{ij} - z\mathbf{A}_{kj}$
    end for
    $\mathbf{b}_i \leftarrow \mathbf{b}_i - z\mathbf{b}_k$
  end for
end for
return **A**,**b**

Następnie mając macierz **A'** górną trójkątną możemy rozwiązać układ równań **A'x = b'** zaczynając

od obliczenia $x_n = \frac{b'_n}{\mathbf{A'}_{nn}}$. Następnie dla $k = (n-1), ..., 1$. $x_k = \frac{b'_k - \sum_{j=k+1}^{n} \mathbf{A'}_{kj} x_j}{\mathbf{A'}_{kk}}$ Algorytm ma złożoność O(n^2).

Możemy zmniejszyć złożoność czasową obu tych algorytmów potrzebnych do obliczenia $\mathbf{x}$ biorąc pod uwagę specyficzną postać macierzy $\mathbf{A}$.

Zacznijmy od algorytmu eliminacji Gaussa. Algorytm polega na zerowaniu kolejnych kolumn macierzy pod przekątną. Zauważmy, że ze względu na strukturę macierzy $\mathbf{A}$ pod przekątną znajduje się co najwyżej l niezerowych wartości. Zatem wystarczy wyzerować kolumnę w co najwyżej l (rozmiar macierzy wewnętrznych) kolejnych rzędach pod wartością $\mathbf{A}_{kk}$.

Podczas modyfikacji tych co najwyżej l rzędów, elementy $\mathbf{A}_{ij}$, dla których $\mathbf{A}_{kj} = 0$, nie zmienią swoich wartości. Ze względu na specyficzną postać $\mathbf{A}$, wszystkie $\mathbf{A}_{kh}$, takie że $h > k + l$ mają wartość 0. Możemy więc modyfikować tylko wartości $\mathbf{A}_{ij}$, takie że $j \leqslant k + l$.

Te dwie optymalizacje dają nam łącznie czas $O(nl^2)$, czyli $O(n)$ dla stałego l (Algorytm 3).

Algorytm 3 Zmodyfikowany algorytm wyprowadzający macierz górną trójkątną **A'** przy pomocy metody eliminacji Gaussa - Złożoność czasowa $O(n)$

```
for k ← 1 to n − 1 do
    for i ← k + 1 to min(k + l, n) do
        z ← A_ik / A_kk
        for j ← k + 1 to min(k + l, n) do
            A_ij ← A_ij − z A_kj
        end for
        b_i ← b_i − z b_k
    end for
end for
return A,b
```

Możemy też poprawić złożoność czasową rozwiązywania układu równań $\mathbf{A'x = b'}$. Ze względu na strukturę $\mathbf{A}$, otrzymana w Algorytmie 3 macierz trójkątna **A'** ma co najwyżej l niezerowych wartości w każdym rzędzie. Wzór na x_k dla $k = (n-1), ..., 1$ możemy więc zapisać jako $x_k = \frac{b'_k - \sum_{j=k+1}^{\min(k+l,n)} \mathbf{A'}_{kj} x_j}{\mathbf{A'}_{kk}}$. Czyli rozwiązanie układu równań można obliczyć w czasie $O(nl) = O(n)$.

Łącząc te dwa algorytmy otrzymujemy algorytm, który oblicza $\mathbf{x}$ z układu równań $\mathbf{Ax = b}$ w czasie $O(n + n) = O(n)$.

W wariancie algorytmu eliminacji Gaussa z częściowym wyborem przed zerowaniem k-tej kolumny przestawiamy k-ty wiersz macierzy $\mathbf{A}$ i wektora $\mathbf{b}$ z p-tym, gdzie p: wiersz macierzy $\mathbf{A}$ z maksymalną wartością bezwzględną w k-tej kolumnie oraz $k \leqslant i \leqslant n$. Unikamy wtedy dzielenia przez małe co do wartości bezwzględnej $\mathbf{A}_{kk}$. Wykonujemy te przestawienia, bo dzielenie przez bardzo małe co do wartości bezwzględnej liczby wpływa negatywnie na dokładność obliczeń.

W dotychczasowej strukturze SpecialMatrix operacja zamiany miejscami rzędów okazała się problematyczna - cześć rzędu p należąca do macierzy wewnętrznej C powinna zostać przeniesiona wyższego rzędu k. Jednak w rzędzie k miejsce to nie jest pamiętane jeśli rzędy nie dzielą tych samych macierzy wewnętrznych - macierz nad C to macierz zerowa (Rysunek 1). Stąd dodałem do struktury macierze wewnętrzne D, dzięki którym pamiętane są wartości z ewentualnego przestawienia (Rysunek 2).

$$\begin{pmatrix} A_1 & C_1 & D_1 & 0 & 0 & 0 & ... & 0 \\ B_2 & A_2 & C_2 & D_2 & 0 & 0 & ... & 0 \\ 0 & B_3 & A_3 & C_3 & D_3 & 0 & ... & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & ... & \vdots \\ 0 & \cdots & 0 & B_{v-3} & A_{v-3} & C_{v-3} & D_{v-3} & 0 \\ 0 & \cdots & 0 & 0 & B_{v-2} & A_{v-2} & C_{v-2} & D_{v-2} \\ 0 & \cdots & 0 & 0 & 0 & B_{v-1} & A_{v-1} & C_{v-1} \\ 0 & \cdots & 0 & 0 & 0 & 0 & B_v & A_v \end{pmatrix}$$

Rysunek 2: Struktura macierzy **A** po dodaniu macierzy wewnętrznych D.

Zmiany w strukturze macierzy wpływają na warunki końcowe pętli algorytmu. W Algorytmie 3 przy zerowaniu kolumny pod $\mathbf{A}_{kk}$ elementy $\mathbf{A}_{kj}$ dla $k+1 \leqslant j \leqslant k+l$ były niezerowe i modyfikowały $\mathbf{A}_{ij}$. Po dodaniu macierzy wewnętrznej D, elementy niezerowe mogą znajdować się na przedziale $k+1 \leqslant j \leqslant k+2l$.

Algorytm 4 Zmodyfikowany algorytm wyprowadzający macierz górną trójkątną **A'** przy pomocy metody eliminacji Gaussa z częściowym wyborem elementu głównego - Złożoność czasowa $O(n)$

for $k \leftarrow 1$ to $n-1$ **do**
  $p \leftarrow k$
  for $i \leftarrow k+1$ to $\min(k+l, n)$ **do**
    if $|\mathbf{A}_{ik}| > |\mathbf{A}_{pk}|$ **then**
      $p \leftarrow i$
    end if
    Przestaw wierzsze p,k w **A**
    Przestaw $\mathbf{b}_p, \mathbf{b}_k$
  end for
  for $i \leftarrow k+1$ to $\min(k+l, n)$ **do**
    $z \leftarrow \frac{\mathbf{A}_{ik}}{\mathbf{A}_{kk}}$
    for $j \leftarrow k+1$ to $\min(k+2l, n)$ **do**
      $\mathbf{A}_{ij} \leftarrow \mathbf{A}_{ij} - z\mathbf{A}_{kj}$
    end for
    $\mathbf{b}_i \leftarrow \mathbf{b}_i - z\mathbf{b}_k$
  end for
end for
return A,**b**

Analogicznie we wzorze na elementy x_k musimy liczyć $\sum_{j=k+1}^{\min(k+2l,n)}$ zamiast $\sum_{j=k+1}^{\min(k+l,n)}$.

Implementacja powyższych algorytmów znajduje się w pliku L5_module.jl jako część modułu **L5**.

Algorytmy 3 i 4 zostały zaimplementowane w funkcji:
gauss(m_in::SpecialMatrix, b_in::Vector{Float64}, pivoting::Bool).

Algorytm rozwiązujący **A'x = b'** dla macierzy górno trójkątnej **A'** został zaimplementowany w funkcji:
calculate_x(m::SpecialMatrix, b::Vector{Float64})

Funkcja ***solve_equations***(A::SpecialMatrix, b::Vector{Float64}, pivoting::Bool, LU::Bool), wywołana dla **LU = false**, uruchamia ***gauss*** oraz ***calculate_x*** i zwraca wynik **x**, taki że **Ax = b**.

Macierz **A** i wektor prawych stron **b** można wczytać funkcjami ***readA***(filename::String), ***readb***(filename::String) lub ***readA_and_generate_b***(filename::String).

Funkcja ***read_solve_save***(A_file::String, b_file::String, pivoting::Bool, LU::Bool, out_file::String) korzystając z wyżej wymienionych funkcji wczytuje z plików A_file, b_file macierz **A** i wektor **b**. Rozwiązuje $\mathbf{Ax = b}$, a następnie zapisuje **x** do pliku out_file.
Dodatkowo jeśli nie podano b_file, funkcja ta generuje **b**, takie że **x** = (1,..,1) i zapisuje błąd względny do pliku.

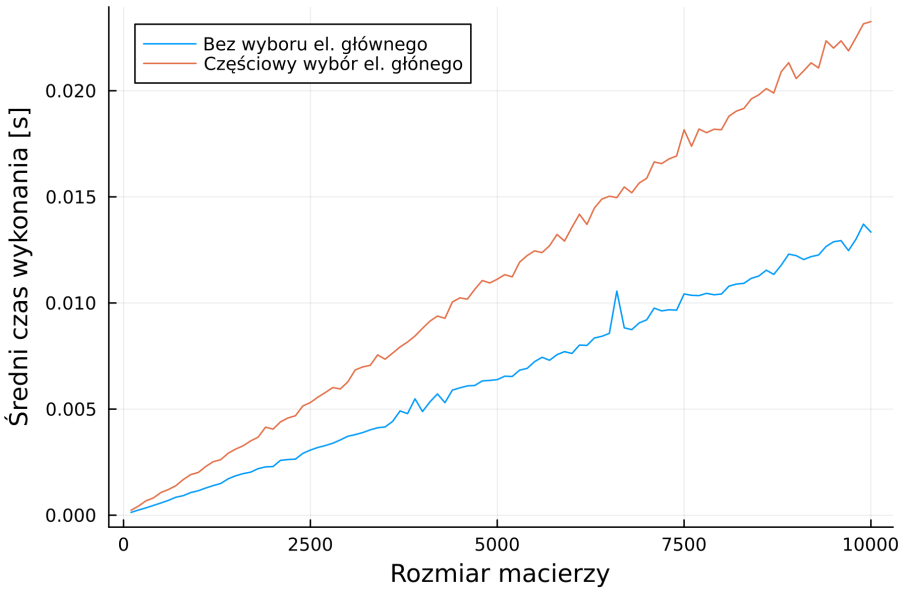
Kod testujący działanie funkcji dla przykładowych danych od prof. Zielińskiego znajduje się w pliku L5_testowe.jl.
Kod testujący działanie funkcji dla losowo wygenerowanych macierzy znajduje się w pliku L5_losowe.jl.
Kod testujący funkcjonalność zapisu wyników do pliku znajduje się w L5_zapisywanie.jl.

Wyniki i interpretacja:

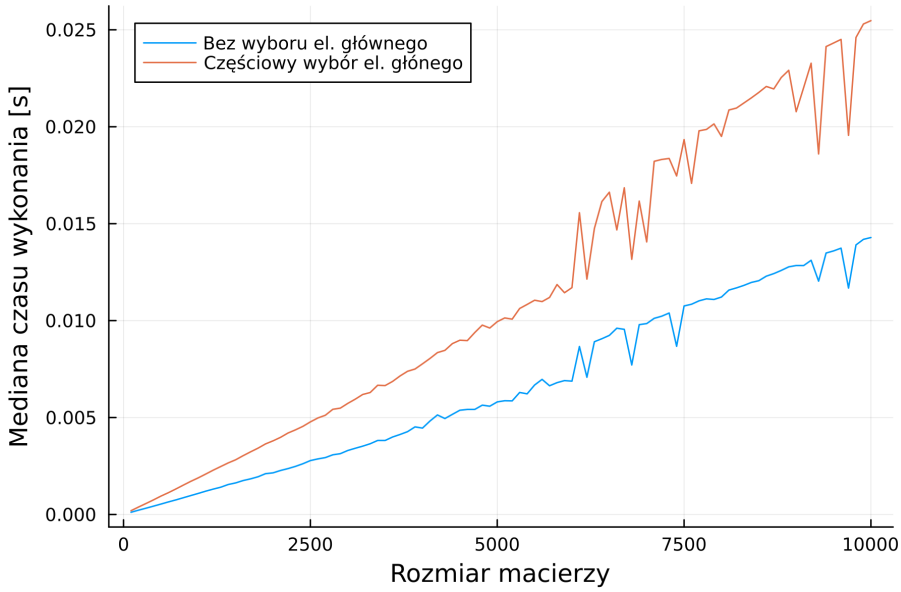
Wariant Algorytmu	Rozmiar Macierzy	$\frac{\|\|x-x'\|\|}{\|\|x\|\|}$
Bez wyboru el. głównego	16	1.456043846448818e-15
Bez wyboru el. głównego	10000	2.486670541464147e-14
Bez wyboru el. głównego	50000	7.061143576494811e-14
Bez wyboru el. głównego	100000	7.435717705787765e-13
Bez wyboru el. głównego	300000	8.481445977654631e-14
Bez wyboru el. głównego	500000	8.422020129360188e-14
Częściowy wybór el. głównego	16	3.090730095650652e-16
Częściowy wybór el. głównego	10000	4.897556061990869e-16
Częściowy wybór el. głównego	50000	5.320431531829447e-16
Częściowy wybór el. głównego	100000	5.116066229467421e-16
Częściowy wybór el. głównego	300000	4.477225396084591e-16
Częściowy wybór el. głównego	500000	4.427737010372654e-16

Tabela 1: Błąd względny obliczonego rozwiązania **Ax' = b** względem **x**=[1,...,1] dla danych testowych od prof. Zielińskiego

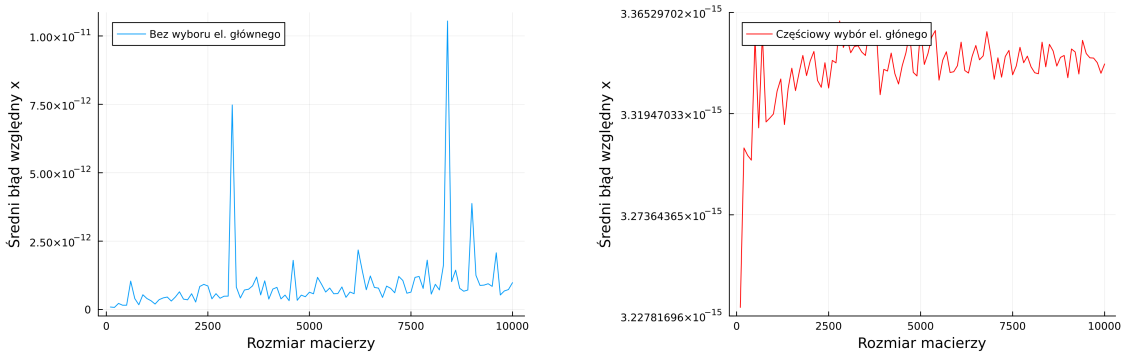
Algorytm używający częściowego wyboru elementu głównego dał dokładniejsze wyniki niż standardowy. (Tabela 1)



Rysunek 3: Średni czas działania algorytmów rozwiązujących $\mathbf{Ax} = \mathbf{b}$ w zależności od rozmiaru macierzy. Średnia ze 100 prób.

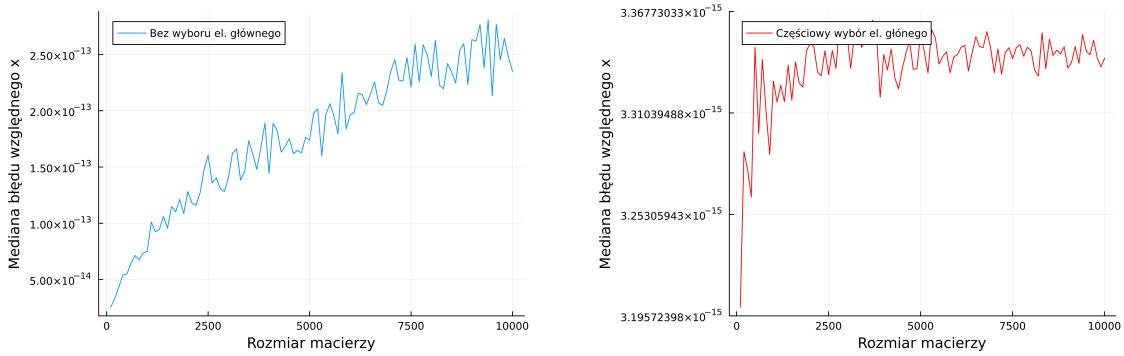


Rysunek 4: Mediana czasu działania algorytmów rozwiązujących $\mathbf{Ax} = \mathbf{b}$ w zależności od rozmiaru macierzy. Mediana ze 100 prób.



Bez wyboru el. głównego | Częściowy wybór el. głównego

Rysunek 5: Średni błąd względny rozwiązań dla generowanych macierzy $\mathbf{A}$ i wektorów $\mathbf{b}$ takich, że $\mathbf{A(1,...,1) = b}$ w zależności od rozmiarów macierzy. Średnia ze 100 prób. Wskaźnik uwarunkowania 100.



Bez wyboru el. głównego | Częściowy wybór el. głównego

Rysunek 6: Mediana błędu względnego rozwiązań dla generowanych macierzy $\mathbf{A}$ i wektorów $\mathbf{b}$ takich, że $\mathbf{A(1,...,1) = b}$ w zależności od rozmiarów macierzy. Mediana ze 100 prób. Wskaźnik uwarunkowania 100.

Z wykresów (Rysunek 3, 4) widać że złożoność czasowa algorytmów to $O(n)$, zgodnie z teoretyczną analizą. Wykresy złożoności pamięciowej oraz ich analiza znajdują się w Zadaniu 3.

Przy braku wyboru elementu głównego, wraz ze zwiększaniem rozmiaru macierzy błąd względny rośnie (Rysunek 5,6). Mediana błędu względnego rośnie logarytmicznie względem rozmiaru (Rysunek 6). Dzieje się tak prawdopodobnie przez zwiększone prawdopodobieństwo wykonania dzielenia przez małą co do wartości bezwzględnej liczbę przy obliczeniach wykonywanych na większej liczbie rzędów.

W porównaniu do wariantu z brakiem wyboru, zmiany mediany i średniej na wykresach dla algorytmu korzystającego z częściowego wyboru el. głównego są małe (Rysunek 5, 6). Nieco lepsze wyniki otrzymano dla macierzy o niewielkim rozmiarze.

Wnioski:

Znając strukturę macierzy $\mathbf{A}$ możemy czasem znacznie zmniejszyć złożoność obliczeniową rozwiązywania $\mathbf{Ab = x}$ używając metody eliminacji Gaussa. W Zadaniu 1 udało się algorytm $O(n^3)$ przyspieszyć do złożoności liniowej. Częściowy wybór elementu głównego znacznie poprawia dokładność algorytmu.

2 Zadanie 2

Opis problemu: Należy napisać funkcję wyznaczającą rozkład **LU** macierzy **A** metodą eliminacji Gaussa uwzględaniającą specyficzną postać macierzy **A** dla wariantów:
(a) bez wyboru elementu głównego,
(b) z częściowym wyborem elementu głównego

Rozwiązanie:

Rozkład **LU** polega na wyznaczeniu macierzy górno trójkątnej **U**, którą wcześniej oznaczałem przez **A'** i którą wyprowadzaliśmy w Algorytmach 2,3,4, oraz jednocześnie macierzy dolno trójkątnej **L** spełniającej $\mathbf{U} = \mathbf{L}^{-1}\mathbf{A}$. Macierz tą możemy odtworzyć zauważając, że krok zewnętrznej pętli w Algorytmie 2 można zapisać jako $\mathbf{A}^{(k+1)} = \mathbf{L}^{(k)}\mathbf{A}^{(k)}$, gdzie $\mathbf{L}^{(k)}$ ma postać:

$$\begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{(k+1)k} & & & \\ & & -l_{(k+2)k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -l_{nk} & & & 1 \end{pmatrix}$$

Rysunek 7: Struktura macierzy $\mathbf{L}^{(k)}$. $l_{ik} = \frac{\mathbf{A}_{ik}}{\mathbf{A}_{kk}}$

Stąd mamy, że $\mathbf{A'} = \mathbf{U} = \mathbf{L}^{(n-1)} \cdots \mathbf{L}^{(2)}\mathbf{L}^{(1)}\mathbf{A}$. $\mathbf{U} = \mathbf{L}^{-1}\mathbf{A}$, czyli $\mathbf{L}^{-1} = \mathbf{L}^{(n-1)} \cdots \mathbf{L}^{(2)}\mathbf{L}^{(1)}$. Stąd $\mathbf{L} = {\mathbf{L}^{(1)}}^{-1}{\mathbf{L}^{(2)}}^{-1} \cdots {\mathbf{L}^{(n-1)}}^{-1}$.

$$\begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ l_{41} & l_{42} & l_{43} & & \\ \vdots & & & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{n(n-1)} & 1 \end{pmatrix}$$

Rysunek 8: Struktura macierzy **L**. $l_{ik} = \frac{\mathbf{A}_{ik}}{\mathbf{A}_{kk}}$

Zauważmy, że dla każdego l_{ik} zerujemy jeden element **A** podczas eliminacji Gaussa ($\mathbf{A}_{ik}$). Zatem nie musimy tworzyć osobnej tablicy do zapamiętywania **L**. Możemy przetrzymywać wartości l_{ik} w miejscu zerowanych elementów. Przekątnej 1 ... 1 macierzy **L** nie musimy pamiętać, bo zawsze wygląda ona tak samo. Zatem wyznaczamy macierz **L** w strukturze w której jednocześnie wyznaczamy **U**, zapamiętując l_{ik} w $\mathbf{A}_{ik}$ (Algorytm 5).

Algorytm 5 Algorytm wyznaczający macierz zawierającą informacje o $\mathbf{L}$ oraz $\mathbf{U}$ - $O(n)$

for $k \leftarrow 1$ to $n-1$ **do**
  for $i \leftarrow k+1$ to $\min(k+l, n)$ **do**
    $z \leftarrow \frac{\mathbf{A}_{ik}}{\mathbf{A}_{kk}}$
    $\mathbf{A}_{ik} \leftarrow z$
    for $j \leftarrow k+1$ to $\min(k+l, n)$ **do**
      $\mathbf{A}_{ij} \leftarrow \mathbf{A}_{ij} - z\mathbf{A}_{kj}$
    end for
  end for
end for
return A

Otrzymana przy pomocy Algorytmu 5 macierz ma postać (Rysunek 9):

$$\begin{pmatrix} \mathbf{A'}_{11} & \mathbf{A'}_{12} & \mathbf{A'}_{13} & \cdots & \mathbf{A'}_{1n} \\ l_{21} & \mathbf{A'}_{22} & \mathbf{A'}_{23} & \cdots & \mathbf{A'}_{2n} \\ l_{31} & l_{32} & \mathbf{A'}_{33} & \cdots & \mathbf{A'}_{3n} \\ l_{41} & l_{42} & l_{43} & \cdots & \mathbf{A'}_{4n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{n(n-1)} & \mathbf{A'}_{nn} \end{pmatrix}$$

Rysunek 9: Struktura macierzy zawierającej informacje o macierzach $\mathbf{L}$ oraz $\mathbf{U}$. $l_{ik} = \frac{\mathbf{A}_{ik}}{\mathbf{A}_{kk}}$

Analogicznie jak w Zadaniu 1 możemy zwiększyć dokładność obliczeń upewniając się, że wykonujemy dzielenie przez największą możliwą liczbę (Algorytm 6). Dokonujemy częściowego wyboru elementu głównego i przestawiamy wiersze w $\mathbf{A}$ oraz $\mathbf{b}$. Tak jak w poprzednim Zadaniu, konieczne jest wprowadzenie macierzy wewnętrznej D, oraz rozszerzenie rozpatrywanych kolumn j dla $\mathbf{A}_{ij}$ do takich, że $k+1 \leqslant j \leqslant k+2l$. Musimy w tym wariancie zapamiętać także jakie przestawienia wykonywaliśmy w wektorze $\mathbf{b}$. Będzie to potrzebne do rozwiązania układu równań w Zadaniu 3.

Algorytm 6 Algorytm wyznaczający macierz zawierającą informacje o **L** oraz **U** z częściowym wyborem elementu głównego - $O(n)$

for $k \leftarrow 1$ to $n-1$ **do**
  $p \leftarrow k$
  for $i \leftarrow k+1$ to $\min(k+l, n)$ **do**
    if $|\mathbf{A}_{ik}| > |\mathbf{A}_{pk}|$ **then**
      $p \leftarrow i$
    end if
    Przestaw wierzsze p,k w **A**
    Przestaw $\mathbf{b}_p, \mathbf{b}_k$
  end for
  for $i \leftarrow k+1$ to $\min(k+l, n)$ **do**
    $z \leftarrow \frac{\mathbf{A}_{ik}}{\mathbf{A}_{kk}}$
    $\mathbf{A}_{ik} \leftarrow z$
    for $j \leftarrow k+1$ to $\min(k+2l, n)$ **do**
      $\mathbf{A}_{ij} \leftarrow \mathbf{A}_{ij} - z\mathbf{A}_{kj}$
    end for
  end for
end for
return A,**b**

Algorytmy 5 i 6 zostały zaimplementowane w funkcji:
gauss_LU(m_in::SpecialMatrix, b_in::Vector{Float64}, pivoting::Bool).

3 Zadanie 3

Opis problemu: Należy napisać funkcję rozwiązującą układ równań $\mathbf{Ax} = \mathbf{b}$ przy pomocy rozkładu **LU** wyznaczonego w Zadaniu 2.

Rozwiązanie:

Mając równanie $\mathbf{LUx} = \mathbf{b}$, możemy podzielić jego rozwiązanie na dwa pod problemy: $\mathbf{Ly} = \mathbf{b}$, a następnie $\mathbf{Ux} = \mathbf{y}$.

Aby rozwiązać $\mathbf{Ly} = \mathbf{b}$ przekształcamy równanie do $\mathbf{y} = \mathbf{L}^{-1}\mathbf{b}$. Następnie wyznaczamy **y**, przy pomocy tablicy **A'** wyznaczonej w Algorytmie 5, która zawiera informacje o **L**, oraz wektora prawych stron **b**. $\mathbf{L}^{-1} = \mathbf{L}^{(n-1)} \cdots \mathbf{L}^{(2)}\mathbf{L}^{(1)}$. Czyli rozwiązujemy $\mathbf{y} = \mathbf{L}^{(\mathbf{n-1})}\mathbf{L}^{(\mathbf{n-2})}...\mathbf{L}^{(\mathbf{1})}\mathbf{b}$. Algorytm rozwiązujący równanie będzie zatem wykonywał kolejne mnożenia $\mathbf{L}^{(\mathbf{i})}$ przez **b**, czytając dolne współczynniki kolejnych kolumn macierzy **A'**. Tym samym będzie on naśladował zachowanie Algorytmu 3, podczas modyfikacji $\mathbf{b}_i$.

Algorytm 7 Algorytm rozwiązujący równanie $\mathbf{Ly} = \mathbf{b}$ - czas $O(n)$

for $k \leftarrow 1$ to $n-1$ **do**
  $\mathbf{y} \leftarrow \mathbf{b}$
  for $i \leftarrow k+1$ to $\min(k+l, n)$ **do**
    $\mathbf{y}_i \leftarrow \mathbf{y}_i - \mathbf{y}_k\mathbf{A}_{ik}$
  end for
end for
return y

Mając **y** pozostaje rozwiązać równanie $\mathbf{Ux} = \mathbf{y}$, gdzie **U** to macierz górna trójkątna. Rozwiązujemy je tak samo jak rozwiązywaliśmy **A'x = b'** w Zadaniu 1 (funkcja ***calculate_x***).

W wariancie z częściowym wyborem elementu głównego musimy rozwiązać **Ly = Pb** zamiast **Ly = b**, gdzie **P** jest macierzą zmieniającą kolejność elementów wektora. Dlatego w Algorytmie 6 zwracamy także zmodyfikowane **b**.

Algorytm 7 rozwiązujący **Ly = b** został zaimplementowany w funkcji:
calculate_y_LU _x(m::SpecialMatrix, b::Vector{Float64})

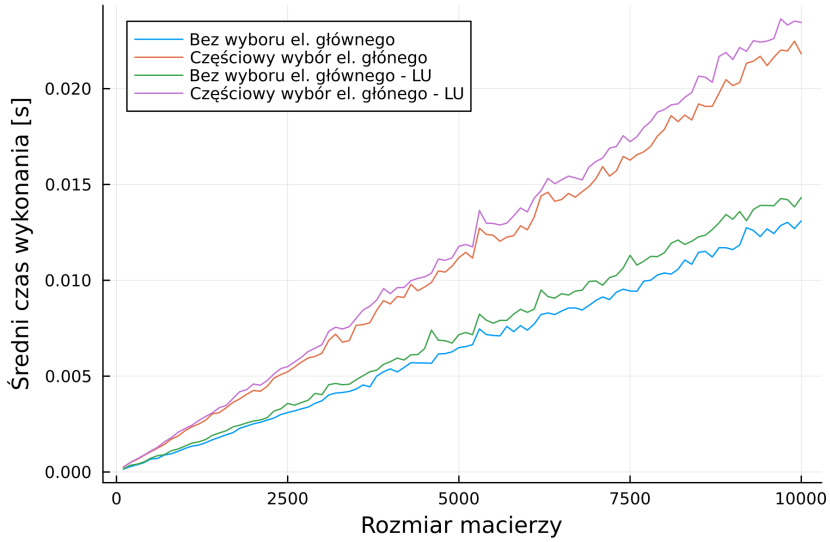
Funkcje ***solve_equations*** oraz ***read_solve_save*** opisane w Zadaniu 1 używają rozkładu LU, jeśli wywołamy je z argumentem **LU = true**.
Korzystają wtedy z ***gauss_LU***, ***calculate_y_LU***, ***calculate_x*** zamiast ***gauss***, ***calculate_x***.

Wyniki i interpretacja:

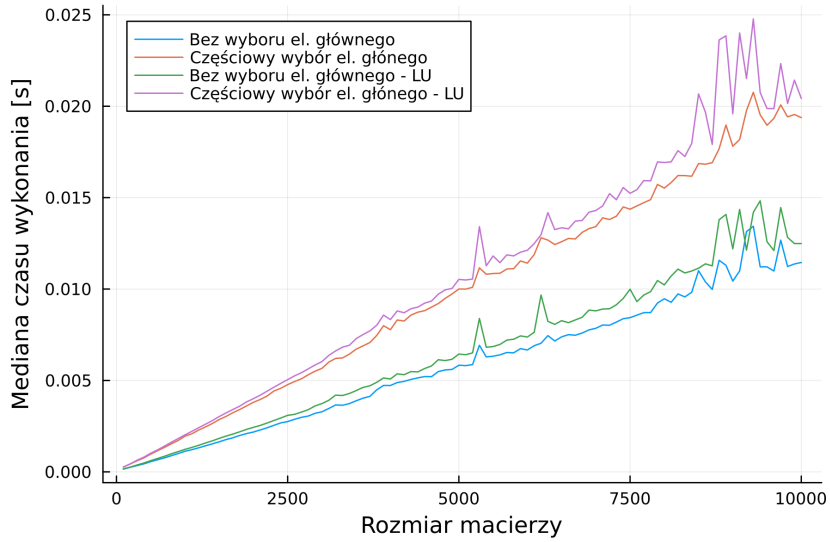
Wariant Algorytmu	Rozmiar Macierzy	$\frac{\|\|x-x'\|\|}{\|\|x\|\|}$
Bez wyboru el. głównego - LU	16	1.456043846448818e-15
Bez wyboru el. głównego - LU	10000	2.486670541464147e-14
Bez wyboru el. głównego - LU	50000	7.061143576494811e-14
Bez wyboru el. głównego - LU	100000	7.435717705787765e-13
Bez wyboru el. głównego - LU	300000	8.481445977654631e-14
Bez wyboru el. głównego - LU	500000	8.422020129360188e-14
Częściowy wybór el. głównego - LU	16	3.090730095650652e-16
Częściowy wybór el. głównego - LU	10000	4.897556061990869e-16
Częściowy wybór el. głównego - LU	50000	5.320431531829447e-16
Częściowy wybór el. głównego - LU	100000	5.116066229467421e-16
Częściowy wybór el. głównego - LU	300000	4.477225396084591e-16
Częściowy wybór el. głównego -LU	500000	4.427737010372654e-16

Tabela 2: Błąd względny obliczonego rozwiązania **Ax' = b** względem **x**=[1,...,1] przy użyciu Algorytmów 5,6,7 **LU** dla danych testowych od prof. Zielińskiego

Otrzymane dokładności rozwiązań **Ax' = b** są identyczne dla algorytmów używających LU i tych z Zadania 1. (Tabela 2)

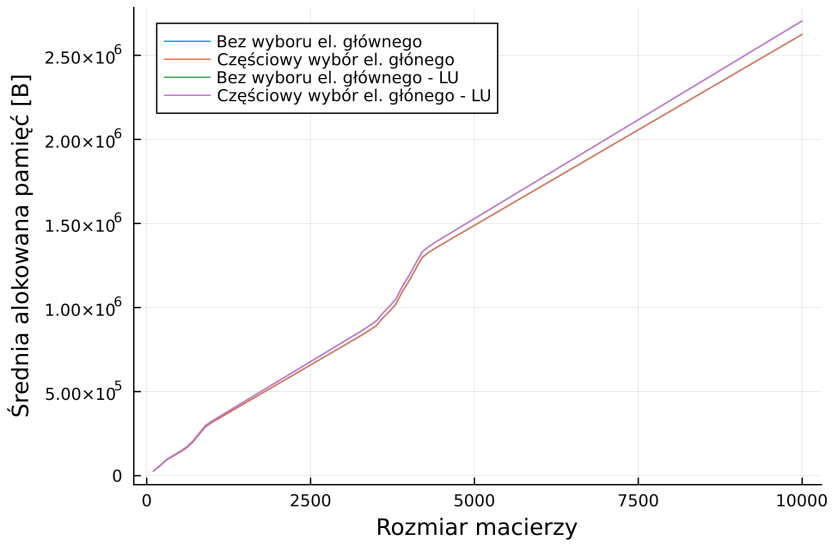


Rysunek 10: Średni czas działania algorytmów rozwiązujących $\mathbf{Ax} = \mathbf{b}$ w zależności od rozmiaru macierzy. Średnia ze 100 prób.

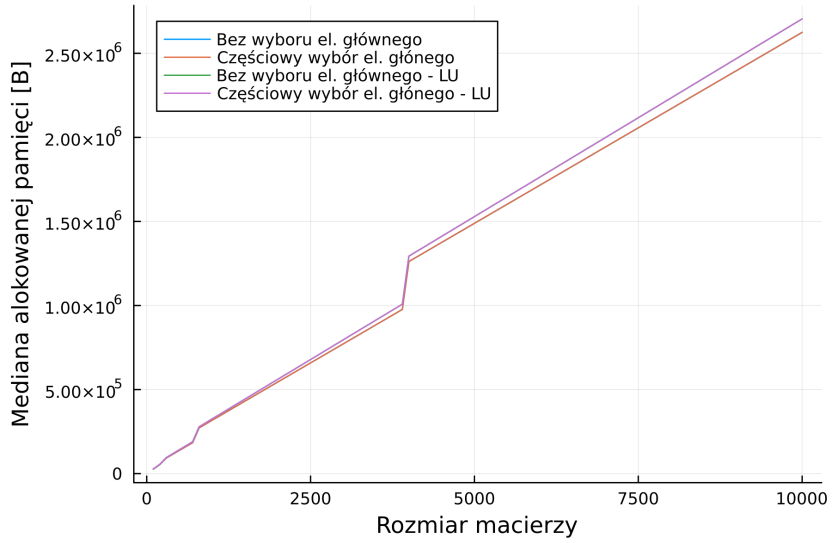


Rysunek 11: Mediana czasu działania algorytmów rozwiązujących $\mathbf{Ax} = \mathbf{b}$ w zależności od rozmiaru macierzy. Mediana ze 100 prób.

Czas wykonywania algorytmów korzystających z rozkładu LU jest zbliżony do odpowiadających algorytmów z Zadania 1. (Rysunek 10,11)



Rysunek 12: Średnia alokowana pamięć algorytmów rozwiązujących $\mathbf{Ax} = \mathbf{b}$ w zależności od rozmiaru macierzy. Średnia ze 100 prób.



Rysunek 13: Mediana alokowanej pamięci algorytmów rozwiązujących $\mathbf{Ax} = \mathbf{b}$ w zależności od rozmiaru macierzy. Mediana ze 100 prób.

Algorytmy korzystające z rozkładu **LU** alokują więcej pamięci niż ich odpowiedniki z Zadania 1. Dzieje się tak z uwagi na alokowanie pamięci dla wektora $\mathbf{y}$ przy rozwiązywaniu $\mathbf{Ly} = \mathbf{b}$. Brak wyboru i częściowy wybór elementu głównego nie wpływają na złożoność pamięciową. Wszystkie algorytmy zachowują złożoność pamięciową $O(n)$. (Rysunek 10,11)

Wnioski:

Algorytmy wyznaczające $\mathbf{x}$ korzystające z rozkładu **LU** dają identyczne wyniki w podobnym czasie (też $O(n)$) co algorytmy z Zadania 1. Algorytmy korzystające z **LU** alokują więcej pamięci, zachowując jednak złożoność pamięciową $O(n)$ algorytmów z Zadania 1.