Eliza Zawisza, 244967

SPRAWOZDANIE 5

DANE DO ZADAŃ 1-3:

Dana macierz współczynników $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ i wektor prawych stron $\boldsymbol{b} \in \mathbb{R}^{n}$, $n \geq 4$. Macierz $\boldsymbol{A}$ jest rzadką, tj. mającą dużo elementów zerowych i blokową o następującej strukturze:

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

$v = n/\ell$, zakładając, że n jest podzielne przez ℓ, gdzie $\ell \geq 2$ jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych (bloków): $\boldsymbol{A_k}, \boldsymbol{B_k}$ *i* $\boldsymbol{C_k}$.

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

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

- $\boldsymbol{C_k} \in \mathbb{R}^{\ell \times \ell}, k = 1, \dots, v-1$ jest macierzą diagonalną postaci:

$$\begin{pmatrix} c^k_1 & 0 & 0 & \cdots & 0 \\ 0 & c^k_2 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c^k_{\ell-1} & 0 \\ 0 & \cdots & 0 & 0 & c^k_{\ell} \end{pmatrix}$$

Sposób przechowywania macierzy:

Ponieważ $\boldsymbol{A}$ jest macierzą rzadką przechowywanie jej w sposób standardowy, czyli w tablicy dwuwymiarowej $n \times n$ byłoby nieefektywne, do przechowywania macierzy została użyta struktura *SparseMatrixCSC* dostępna w bibliotece Julii. Dzięki niej macierze przechowywane są w skompresowanym porządku kolumnowym i mamy szybki dostęp do przechowywanych elementów.

ZADANIE 1

Opis problemu:

Celem zadania pierwszego było napisać funkcję rozwiązującą układ $\boldsymbol{Ax} = \boldsymbol{b}$ metodą Gaussa uwzględniającą specyficzną postać macierzy $\boldsymbol{A}$ dla dwóch wariantów:

a) bez wyboru elementu głównego
b) z częściowym wyborem elementu głównego

Opis metody:

Metoda eliminacji Gaussa służy do rozwiązywania równań pierwszego stopnia, obliczania wartości wyznacznika macierzy, obliczania jej rzędu oraz rozkładu LU danej macierzy.

Metoda składa się z dwóch głównych etapów. Etap pierwszy (*Redukcja macierzy górnej)* służy do sprowadzenia układu równań do układu równoważnego z macierzą trójkątną górną. Polega on na zerowaniu elementów, które znajdują się pod elementami głównymi występującymi na diagonali macierzy. W pierwszym kroku zerowane są wszystkie poniżej pierwszego wiersza w pierwszej kolumnę. W celu wyzerowania elementu a_{k1} od k-tego wiersza zostanie odjęty pierwszy wiersz pomnożony przez $z = a_{k1}/a_{11}$ (mnożnik). Krok ten jest powtarzany dla drugiej kolumny, trzeciej, itd. Aby można było wykonać wcześniej opisaną procedurę żaden z elementów na diagonali nie może być równy 0. W przypadku, gdy mamy do czynienia z 0 na diagonali algorytm wymaga wprowadzenia drobnych modyfikacji, np. zamiany wierszy. Drugim etapem rozwiązywania równań przy użyciu metody eliminacji Gaussa jest zastosowanie algorytmu podstawiania wstecz. Polega on na zastosowaniu poniższego wzoru:

$x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{kj}}{a_{kk}}$ dla kolejnych $k -$ wierszy od n do 1.

Algorytm eliminacji Gaussa ma złożoność obliczeniową rzędu $O(n^3)$, podstawianie wstecz z etapu drugiego cechuje złożoność obliczeniowa $O(n^2)$, zatem by obliczyć układ równań metodą eliminacji Gaussa należy wykonać $O(n^3)$ operacji.

Modyfikacja algorytmu dla specyficznych danych wejściowych:

Ponieważ macierz A ma specyficzną trójdiagonalno blokową postać można zauważyć, że nie potrzebujemy zerować wszystkich elementów będących w danej kolumnie pod diagonalą. Ze względu na budowę podmacierzy A_i oraz B_i znacznie zmniejsza się ilość zerowanych wartości pod diagonalą. Dla pierwszych $\ell - 2$ kolumn niezerowe elementy mogą znajdować się jedynie w pierwszych ℓ rzędach (elementy bloku $\boldsymbol{A_1}$), dla kolejnych ℓ kolumn niezerowe elementy mogą znajdować się jedynie w pierwszych 2ℓ rzędach (dwie ostatnie kolumny bloku $\boldsymbol{B_2}$ oraz elementy bloku $\boldsymbol{A_3}$) itd.

Dzięki wcześniejszym obserwacjom możemy wyznaczyć wzór na maksymalny indeks kolumny w danym rzędzie, w której znajduje się element niebędący zerem:

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

Warto również dostrzec, iż podczas odejmowania od siebie rzędów nie jest konieczne modyfikowanie wszystkich wartości w wierszu. Wystarczy jedynie odjąć wartości niezerowe. Czyli wszystkie elementy, których indeks kolumny jest mniejszy bądź równy wyliczonemu w $lastColumn(row)$.

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

Stosując zmodyfikowaną wersję algorytmu rozwiązywania równań liniowych eliminacją Gaussa dla macierzy A otrzymujemy algorytm, którego złożoność obliczeniowa jest rzędu $O(n)$. Zewnętrzna pętla wykona się $n-1$ razy, środkowa maksymalnie 2ℓ, wewnętrzna maksymalnie ℓ. Dodatkowo zewnętrzna pętla przebiegu wstecz wykona się maksymalnie n razy, wewnętrzna maksymalnie ℓ. Stąd mamy do czynienia ze złożonością liniową. Przyjmując ℓ jako stałą.

Dokładny przebieg powyżej opisanego algorytmu można znaleźć w sekcji ***Rozwiązanie*** w ZADANIU 1.

Metoda eliminacji Gaussa z wyborem elementu głównego:

Ze względu na specyficzną postać macierzy $\boldsymbol{A}$ w celu obliczenia równani liniowego można zastosować metodę eliminacji Gaussa z częściowym wyborem elementu głównego. Jest to wariant zwykłej metody eliminacji Gaussa wykorzystywany w momencie, gdy na diagonali znajdują się elementy zerowe (dokładnie tak jak jest to w rozpatrywanej przez nas macierzy A). W przypadku zastosowania tego algorytmu wybieramy wiersz, dla którego element w eliminowanej kolumnie k ma największą, co do modułu wartość i dokonujemy zamiany z k-tym wierszem. Po dokonaniu tej zamiany dalsza eliminacja przeprowadzana jest już w zwykły sposób (opisany we wcześniejszej sekcji).

Ponieważ zamiana wierszy jest bardzo kosztowna, szczególnie dla macierzy o większych rozmiarach w algorytmie został stworzony nowy wektor permutacji wierszy (p), w którym zapisywana jest informacja, o aktualnej pozycji danego wiersza w macierzy. Dzięki temu zamiast odwoływać się do konkretnego wiersza można odwołać się do jego pozycji w wektorze permutacji, co znacznie ulepsza algorytm.

Ze względu na wybór elementu głównego należy na nowo wyprowadzić wzór na maksymalny indeks kolumny. Dotychczasowy wzór na obliczanie indeksu może prowadzić do powstania większej ilości elementów niezerowych. Warto więc ponownie przyjrzeć się budowie naszej macierzy. Podczas eliminowania współczynników w pierwszych ℓ kolumnach najdalszym niezerowym elementem jest element będący w kolumnie o indeksie 2ℓ. Gdy eliminujemy kolejne ℓ kolumn, najdalszym niezerowym elementem jest element w 3ℓ kolumnie etc. Dzięki temu powstaje nam nowy wzór na indeks ostatniej kolumny, która w danym rzędzie zawiera niezerowy element.

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

zmiana to dotyczy również algorytmu podstawiania wstecz. Wybór elementu głównego ma niewielki wpływ na złożoność, dzięki czemu nadal otrzymujemy złożoność całego algorytmu rzędu $O(n)$.

Dokładny przebieg powyżej opisanego algorytmu można znaleźć w sekcji poniżej.

Rozwiązanie:

- Dane wejściowe:
  - A – podana w treści zadania macierz
  - b – wektor prawych stron
  - n – rozmiar macierzy A
  - l – rozmiar bloku macierzy A
- Dane wyjściowe:
  - x – wektor zawierający rozwiązanie równania $Ax = b$

Algorytm metody eliminacji Gaussa:

```
function calculateGaussElimination(A, n, l, b)
   for k ← 1 to n-1 do
         lastRow ← min{n, l + l * ⌊(k+1)/l⌋}          //zakres indeksu
         lastColumn ← min{n, k + l}                   //zakres indeksu
         for i ← k + 1 to lastRow do
               if A[k,k] = 0 then
                     error błąd w obliczaniu
               z ← A[k, i] / A[k, k]                  //mnożnik
               A[k, i] ← 0
               for j ← k + 1 to lastColumn do
                     A[j, i] ← A[j, i] - z · A[j, k]
               b[i] ← b[i] - z · b[k]                 //zmiana wektora prawych stron
   x ← backwardSubstitution(A, l, n, b)               //podstawienie wstecz
return x                                              //wektor wynikowy
```

Pseudokod funkcji `backwardSubstitution(A, l, n, b)` wraz z opisem danych wejściowych znajduje się w sekcji ***Rozwiązanie*** w Zadaniu 3.

- Dane wejściowe:
  - A – podana w treści zadania macierz
  - b – wektor prawych stron
  - n – rozmiar macierzy A
  - l – rozmiar bloku macierzy A
- Dane wyjściowe:
  - x – wektor zawierający rozwiązanie równania $Ax = b$

Algorytm metody eliminacji Gaussa z wyborem elementu głównego

```
function calculateGaussWithChoice(A, n, l, b)
   p ← {i: i ∈ {1, ... , n}}
   for k ← to n – 1 do
         lastRow ← min{n, l + l * ⌊(k+1)/l⌋}          //zakres indeksu
         lastColumn ← min{n, 2l + l * ⌊(k+1)/l⌋}     //zakres indeksu
         for i ← k + 1 to lastRow do
               maxRow ← wiersz, dla którego element w eliminowanej kolumnie k ma największą, wartość bezwgl.
                     if p[maxRow] = 0 then
                     error błąd w obliczaniu
               swap(p[k], p[maxRow])
               z ← A[k,p[i]] / A[k,p[k]]  //mnożnik
               A[k,p[i]] ← 0
               for j ← k + 1 to lastColumn do
                     A[j,p[i]] ← A[j,p[i]] - z * A[j,p[k]]
               b[p[i]] ← b[p[i]] - z * b[p[k]]  //zamiana wektora prawych stron
    x ← backwardInChoiceSubstitution(A, p, l, n, b) //podstawienie wstecz
   return x                                    //wektor wynikowy
```

Pseudokod funkcji `backwardInChoiceSubstitution (A, p, l, n, b)` wraz z opisem danych wejściowych znajduje się w sekcji ***Rozwiązanie*** w Zadaniu 3.

ZADANIE 2

Opis problemu:

Celem zadania drugiego było napisać funkcję wyznaczającą rozkład ***LU*** macierzy ***A*** metodą eliminacji Gaussa uwzględniającą specyficzną postać macierzy ***A*** dla dwóch wariantów:

a) bez wyboru elementu głównego
b) z częściowym wyborem elementu głównego

Opis metody:

Rozkład LU to kolejna metoda rozwiązywania równań liniowych. Zakłada ona rozkład danej macierzy A na macierz dolnotrójkątną L (*ang. lower*) oraz górnotrójkątną U (*ang. upper*), taki, że A=LU.

Dla przykładu daną macierz A można przedstawić w następujący sposób:

$$[\boldsymbol{A}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{12} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}$$

W celu otrzymania powyższego rozkładu stosuje się metodę eliminacji Gaussa. Metoda ta przekształca daną macierz **A** do macierzy trójkątnej górnej (szczegółowy opis metody Gaussa znajduje się kilka stron wcześniej), **U**. Macierz dolną **L** otrzymuje się poprzez zapamiętanie mnożników, które zostały użyte do eliminacji kolejnych współczynników macierzy. (mnożnik służący do wyzerowania elementu a_{ij} zostanie zapisany w polu o indeksie [i, j] w macierzy **L**.

Algorytm wyznaczania rozkładu **LU** macierzy ma złożoność podobną do algorytmu eliminacji Gaussa, gdyż w głównej mierze na nim opiera się stworzenie rozkładu i jest rzędu $O(n^3)$. Rozkład **LU** w pewnych specyficznych przypadkach pozwala na szybsze rozwiązywanie układów równań niż metoda eliminacji Gaussa. W przypadku, gdy mamy do czynienia z tą samą macierzą etap, w którym dokonujemy eliminacji Gaussa jest wykonywany tylko raz, natomiast rozwiązanie układów równań zostaje sprowadzone wyłącznie do dwóch etapów:

$$\begin{cases} L * z = y \\ U * x = z \end{cases}$$

ponieważ macierze **L** i **U** są trójkątne to złożoność algorytmu jest rzędu $O(n^2)$.

Modyfikacja algorytmu dla specyficznych danych wejściowych:

Rozkład **LU** dla specyficznej postaci macierzy ***A*** jest wyznaczany w bardzo podobny sposób do algorytmu eliminacji Gaussa w obu wariantach (bez wyboru elementu głównego oraz z częściowym wyborem elementu głównego). Z tym wyjątkiem, że nie zerujemy elementów a_{ij} tylko w ich miejsce podstawiamy odpowiednie mnożniki $z = a_{ij}/a_{jj}$, które są elementami dolnotrójkątnej macierzy ***L***.

Złożoność obliczeniowa tego algorytmu jest taka sama jak algorytmu eliminacji Gaussa zarówno z częściowym wyborem, jak i bez wyboru elementu głównego i przy założeniu, że ℓ jest stałą złożoność jest rzędu $O(n)$.

Rozwiązanie:

- Dane wejściowe:
  - A – podana w treści zadania macierz
  - b – wektor prawych stron
  - n – rozmiar macierzy A
  - l – rozmiar bloku macierzy A
- Dane wyjściowe:
  - x – wektor zawierający rozwiązanie równania $Ax = b$

Algorytm rozwiązywania równań liniowych, przy użyciu rozkładu LU:

```
function calculateLU (A, n, l, b)
   for k ← 1 to n-1 do
         lastRow ← min{n, l + l * ⌊(k+1)/l⌋}          //zakres indeksu
         lastColumn ← min{n, k + l}                   //zakres indeksu
         for i ← k + 1 to lastRow do
               if A[k,k] = 0 then
                     error błąd w obliczaniu
               z ← A[k, i] / A[k, k]                  //mnożnik
               A[k, i] ← z                            //tworzenie macierzy L
               for j ← k + 1 to lastColumn do
                     A[j, i] ← A[j, i] - z · A[j, k]
               b[i] ← b[i] - z · b[k]                 //zmiana wektora prawych stron
   x ← backwardSubstitution(A, l, n, b)               //podstawienie wstecz
   return x                                           //wektor wynikowy
```

Pseudokod funkcji `backwardSubstitution(A, l, n, b)` wraz z opisem danych wejściowych znajduje się w sekcji ***Rozwiązanie*** w Zadaniu 3.

- Dane wejściowe:
  - A – podana w treści zadania macierz
  - b – wektor prawych stron
  - n – rozmiar macierzy A
  - l – rozmiar bloku macierzy A
- Dane wyjściowe:
  - x – wektor zawierający rozwiązanie równania $Ax = b$

Algorytm rozwiązywania równań liniowych, przy użyciu rozkładu LU z częściowym wyborem elementu głównego:

```
function calculateLUWithChoice(A, n, l, b)
   p ← {i: i ∈ {1, ... , n}}
   for k ← to n – 1 do
         lastRow ← min{n, l + l * ⌊(k+1)/l⌋}          //zakres indeksu
         lastColumn ← min{n, 2l + l * ⌊(k+1)/l⌋}      //zakres indeksu
         for i ← k + 1 to lastRow do
               maxRow ← wiersz, dla którego element w eliminowanej kolumnie k ma największą, wartość bezwgl.
                     if p[maxRow] = 0 then
                     error błąd w obliczaniu
               swap(p[k], p[maxRow])
               z ← A[k,p[i]] / A[k,p[k]]  //mnożnik
               A[k,p[i]] ← z
               for j ← k + 1 to lastColumn do
                     A[j,p[i]] ← A[j,p[i]] - z * A[j,p[k]]
               b[p[i]] ← b[p[i]] - z * b[p[k]]  //zamiana wektora prawych stron
    x ← backwardInChoiceSubstitution(A, p, l, n, b) //podstawienie wstecz
   return x                                  //wektor wynikowy
```

Pseudokod funkcji `backwardInChoiceSubstitution (A, p, l, n, b)` wraz z opisem danych wejściowych znajduje się w sekcji ***Rozwiązanie*** w Zadaniu 3.

ZADANIE 3

Opis problemu:

Celem zadania trzeciego było napisać funkcję rozwiązującą układ równań $\boldsymbol{Ax = b}$ (uwzględniającą specyficzną postać macierzy $\boldsymbol{A}$) jeśli wcześniej został już wyznaczony rozkład $\boldsymbol{LU}$ przez funkcję z zadania 2.

Opis metody:

Algorytm rozwiązywania układu równań liniowych przy pomocą rozkładu LU sprowadza się tak naprawdę do zastosowania metody eliminacji Gaussa z tym wyjątkiem, że zamiast w miejsce zerowanych elementów zostają wstawiane odpowiednie mnożniki.

Rozwiązanie:

- Dane wejściowe:
  - A - zadana macierz po przekształceniu
  - l - rozmiar bloków macierzy
  - n - rozmiar macierzy
  - b - wektor prawych stron
- Dane wyjściowe:
  - x – wektor zawierający rozwiązanie równania $Ax = b$

Funkcja wykonująca podstawienie wstecz i obliczająca rozwiązanie $Ax = b$

```
function backwardSubstitution (A, l, n, b)
    for i ← n downto 1 do
        sum ← 0.0
        lastColumn ← min{n, 2l + l * ⌊(p[i]+1)/l⌋}
        for j ← i + 1 to lastColumn do
            sum ← sum + A[j,p[i]] * x[j]
        x[i] ← (b[p[i]] - sum) / A[i, p[i]]
   return x                                  //wektor wynikowy
```

- Dane wejściowe:
  - A - zadana macierz po przekształceniu
  - p - wektor permutacji
  - l - rozmiar bloków macierzy
  - n - rozmiar macierzy
  - b - wektor prawych stron
- Dane wyjściowe:
  - x – wektor zawierający rozwiązanie równania $Ax = b$

Funkcja wykonująca podstawienie wstecz i obliczająca rozwiązanie $Ax = b$ dla metod z częściowym wyborem elementu głównego:

```
function backwardInChoiceSubstitution (A, p, l, n, b)
    for i ← n downto 1 do
        sum ← 0.0
        lastColumn ← min{n, 2l + l * ⌊(p[i]+1)/l⌋}
        for j ← i + 1 to lastColumn do
            sum ← sum + A[j,p[i]] * x[j]
        x[i] ← (b[p[i]] - sum) / A[i, p[i]]
   return x                                  //wektor wynikowy
```

Pseudokody algorytmu rozkładu LU znajdują się w sekcji ***Rozwiązanie*** w Zadaniu .

Wszystkie opisane wcześniej funkcje dostępne są w module *blocksys.jl.* Dodatkowo w module znajdują się funkcje:

- `loadMatrixFromFile(filePath)` – funkcja pobiera z pliku macierz i zapisuje ją do struktury typu SparseMatrixCSC.
- `loadRightSideVectorFromFile(filePath)` – funkcja pobiera z pliku wektor prawych stron I zapisuje go do tablicy
- `exportSolutionToFile(filePath)` – funkcja zapisuje rozwiązania układu równań $Ax = b$ wraz z błędem względnym obliczeń do pliku

Wyniki i ich interpretacja:

Dla powyżej zaimplementowanych funkcji zostały napisane programy testujące, których dane wejściowe znajdują się odpowiednio w katalogach „Dane16_1_1", „Dane10000_1_1" i „Dane50000_1_1".

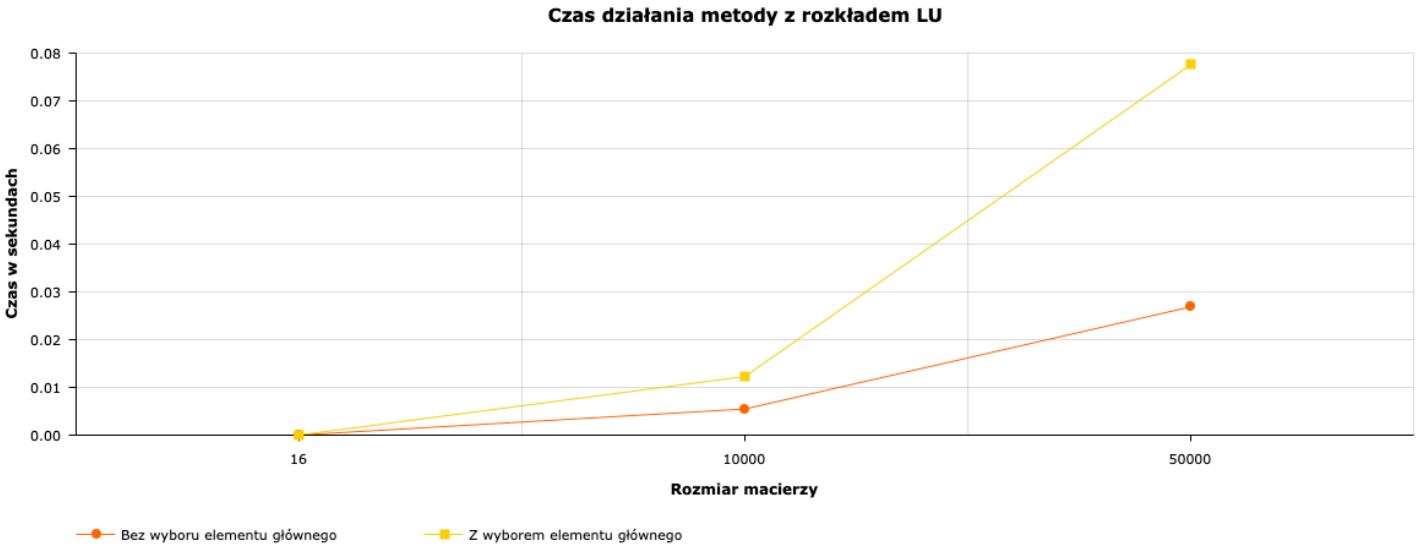
Rozmiar macierzy	Błąd względny obliczeń	
	Bez wyboru elementu głównego	Z wyborem elementu głównego
16	5.45310538513047e-16	5.0037075531084e-16
10000	3.6738202190459565e-14	4.478865903518945e-16
50000	9.447293155699398e-14	5.427796231409846e-16

Ponieważ wyniki zwracające rozwiązanie układu równań przy użyciu metody eliminacji Gaussa i metody z rozkładem LU są takie same. W powyższej tabeli zostały umieszczone pojedyncze wyniki odpowiednio dla metod bez wyboru elementu głównego oraz z wyborem elementu głównego, dla poszczególnych rozmiarów macierzy.

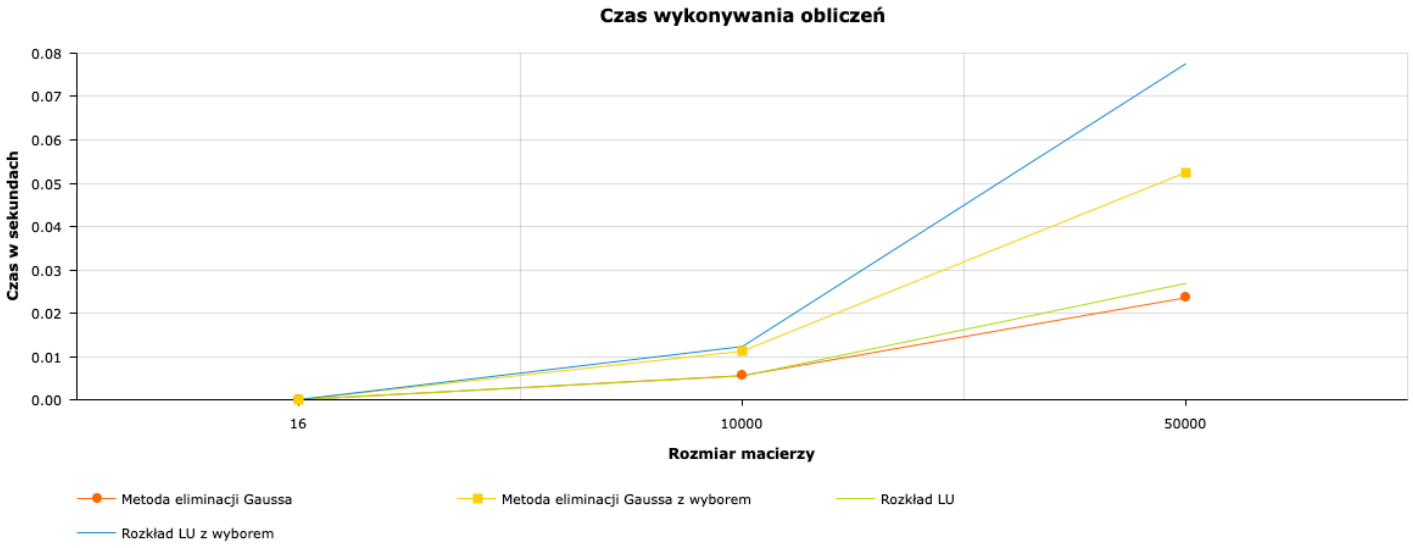
	Czas wykonywania obliczeń w sekundach			
	Metoda eliminacji Gaussa		Metoda z rozkładem LU	
	Bez wyboru elementu głównego	Z wyborem elementu głównego	Bez wyboru elementu głównego	Z wyborem elementu głównego
16	0.000018	0.000033	0.000025	0.000030
10000	0.005574	0.011216	0.005482	0.012282
50000	0.023554	0.052476	0.026870	0.077584

Powyższa tabela przedstawia zestawienie czasu potrzebnego do obliczenia układu równań dla macierzy oo rozmiarach 16, 10000, 50000 wszystkich 4 przypadków. Łatwo można zauważyć, że wraz ze wzrostem rozmiaru macierzy wzrasta ilość czasu potrzebna na wykonanie wszystkich obliczeń. Ponadto widać duże różnice pomiędzy czasem potrzebnym do obliczenia układu równań metodą bez wyboru elementu głównego i z wyborem takiego elementu.

Na następnej stronie znajdują się wykresy obrazujące powyższe dane w tabeli. Osobno został przedstawiony wykres ze wszystkim 4 przypadkami obliczeń oraz wykres przedstawiający różnice w czasie rozwiązywania równania liniowego za pomocą metody rozkładu LU dla opcji z częściowym wyborem elementu głównego i brakiem jego wyboru.



Wykres porównujący czas wykonania programu dla metody rozkładu LU obliczania układu równań liniowych.

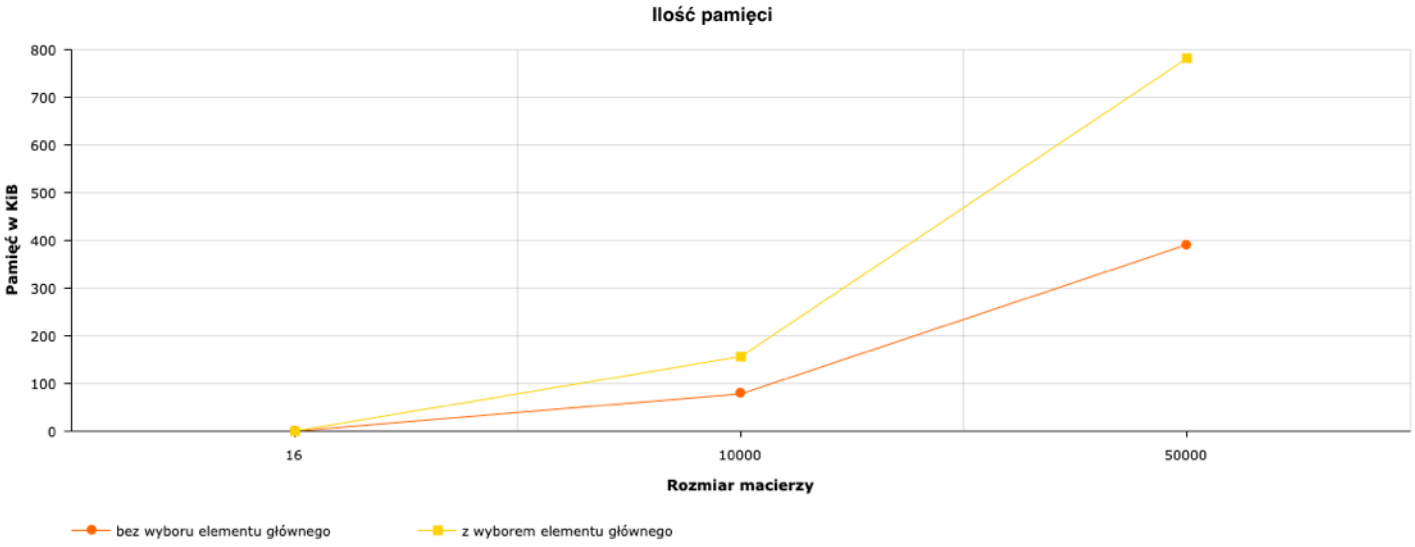


Wykres porównujący czas wykonania programu dla wszystkich 4 przypadków rozwiązywania równań liniowych.

Rozmiar macierzy	Zużycie pamięci dla metod obliczania równań liniowych	
	Bez wyboru elementu głównego	Z wyborem elementu głównego
16	0.358 KiB	0.563 KiB
10000	78.359 KiB	156.563 KiB
50000	390.859 KiB	781.563 KiB

Podobnie jak w przypadku błędu względnego obliczeń zarówno dla obliczania układu równań liniowych metodą z eliminacją Gaussa i z wykorzystaniem rozkładu LU otrzymujemy takie same wyniki dla odpowiednich przypadków z częściowym wyborem elementu głównego i z jego brakiem.

Jak łatwo zaobserwować wraz ze wzrostem rozmiaru macierzy wzrasta nam ilość pamięci potrzebnej do wykonania programów. Ponadto widać znaczącą różnicę pomiędzy algorytmem, w którym nie dokonujemy częściowego wyboru elementu głównego, a algorytmem, w którym dokonujemy wyboru. Ilość pamięci potrzebnej do wykonania algorytmu z częściowym wyborem jest ponad 2-krotnie większa.



Wnioski:

Tabela przedstawiająca błędy względne obliczeń pokazuje, że zaimplementowane metody są prawidłowe, gdyż błędy są bardzo małej wielkości, często wynikają z dokładności arytmetyki. Analizując dokładnie tabelę można zauważyć, że dla algorytmów z częściowym wyborem elementu głównego dokładność obliczeń jest jeszcze większa (błąd∂ względny jest mniejszy). Wynika to z tego, że algorytm ten cechuje się lepszą poprawnością numeryczną.

Dokonując analizy czasu wykonywania poszczególnych programów można zauważyć, że algorytmy, w których dokonujemy wyboru elementu głównego są znacznie wolniejsze od algorytmów bez wyboru. Niestety jak zostało wcześniej wspomniane w niektórych przypadkach niemożliwe jest korzystanie z algorytmu bez wyboru elementu głównego. Takim przypadkiem, w którym jest to niemożliwe jest sytuacja, w której chcemy rozwiązać układ równań liniowych z macierzą diagonalną, której jakikolwiek element na diagonali jest zerowy.

Dzięki dokonanym modyfikacjom przy algorytmach eliminacji Gaussa i rozkładu LU dostosowującym algorytm do specyficznej postaci macierzy osiągnięta została złożoność liniowa. Widać to dokładnie na

wykresie obrazującym ilość pamięci potrzebną do obliczenia poszczególnych algorytmów. W przypadku algorytmu, w którym nie dokonujemy wyboru elementu głównego udało się osiągnąć złożoność pamięciową rzędu $O(n)$. Natomiast w momencie, gdy stosujemy algorytm z częściowym wyborem elementu głównego złożoność pamięciowa jest już nieco gorsza i wynosi $O(n^2)$ niewątpliwie wpływa na to wypełnianie macierzy wykorzystujące permutowanie jej rzędów.

Ogólny wniosek jaki można wysnuć po dokonaniu powyższych eksperymentów jest taki, że poprze dostosowanie algorytmów do postaci danych wejściowych możemy ingerować z złożoność obliczeniową i dokonywać znaczących optymalizacji w dokonywanych przez nas obliczeniach.