Obliczenia naukowe
Lista 5

Joanna Szołomicka

6 stycznia 2020

1 Opis problemu

Zadanie polegało na rozwiązaniu układu równań postaci:

$$\mathbf{Ax} = \mathbf{b}$$

dla danej macierzy współczynników $\mathbf{A} \in \mathbb{R}^{n \times n}$ i wektora prawych stron $\mathbf{b} \in \mathbb{R}^n$. Macierz $\mathbf{A}$ jest macierzą rzadką i blokową o następującej strukturze:

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

gdzie $v = \frac{n}{l}$, zakładając, że n jest podzielne przez l, gdzie l jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych (bloków): $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$. Mianowicie, $\mathbf{A}_k \in \mathbb{R}^{l \times l}$, $k = 1, \dots, v$ jest macierzą gęstą, $\mathbf{0}$ jest kwadratową macierzą zerową stopnia l, macierz $\mathbf{B}_k \in \mathbb{R}^{l \times l}$, $k = 2, \dots, v$ jest następującej postaci:

$$\mathbf{B}_k = \begin{pmatrix} 0 & \dots & 0 & b^k_{1l-1} & b^k_{1l} \\ 0 & \dots & 0 & b^k_{2l-1} & b^k_{2l} \\ \vdots & \ddots & \vdots & \vdots & \\ 0 & \dots & 0 & b^k_{ll-1} & b^k_{ll} \end{pmatrix},$$

$\mathbf{B}_k$ ma tylko dwie ostatnie kolumny niezerowe. Natomiast macierz $\mathbf{C}_k \in \mathbb{R}^{l \times l}$, $k = 1, \dots, v-1$ jest macierzą diagonalną:

$$\mathbf{C}_k = \begin{pmatrix} c^k_1 & 0 & 0 & \dots & 0 \\ 0 & c^k_2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & c^k_{l-1} & 0 \\ 0 & \dots & 0 & 0 & c^k_l \end{pmatrix}.$$

2 Algorytm eliminacji Gaussa

2.1 Opis działania

Algorytm eliminacji Gaussa polega na sprowadzeniu wejściowego układu równań do równoważnego układu z macierzą trójkątną górną przy wykorzystaniu operacji elementarnych (dodawanie, odejmowanie i mnożenie przez czynnik $z \neq 0$) na wierszach i kolumnach. Otrzymany układ z macierzą trójkątną można rozwiązać przy pomocy algorytmu podstawiania wstecz.

Algorytm polega na zerowaniu kolejnych elementów macierzy pod diagonalą. W pierwszym kroku zostaje wyeliminowana niewiadoma x_1 z $n-1$ równań przez odjęcie odpowiedniej krotności pierwszego równania od i-tego równania dla $i = 2, \cdots, n$. Proces jest powtarzany dla kolejnych niewiadomych x_k, gdzie dla $i = k+1, \cdots, n$ od i-tego równania odejmowana jest odpowiednia krotność k-tego równania. W celu wyzerowania elementu a_{ik}, od wiersza i-tego odejmowany jest wiersz k-ty pomnożony przez *mnożnik:* $z = a_{ik}/a_{kk}$. Opisana powyżej wersja algorytmu nie zadziała jeżeli na diagonali macierzy występuje element zerowy, gdyż przy wyznaczaniu mnożnika wystąpi dzielenie przez 0. Rozwiązaniem tego problemu w tej wersji algorytmu jest zamiana miejscami wierszy lub kolumn. Przy dokonywaniu eliminacji należy także dokonać zmian w wektorze prawych stron **b**. Po otrzymaniu macierzy trójkątnej należy zastosować algorytm podstawiania wstecz, który polega na obliczeniu:

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

dla wierszy $i = n \ldots 1$.
Złożoność obliczeniowa przedstawionego algorytmu wynosi $\mathcal{O}(n^3)$.

2.2 Dostosowanie do postaci macierzy rzadkiej A

Z opisu problemu wiadomo, że macierz **A** jest macierzą rzadką o rozmiarze $n \times n$ i składa się z podmacierzy o rozmiarach $l \times l$, tak że $l \mid n$. Poniżej przedstawiono fragment macierzy **A** dla $n = lk$, $k \in \mathbb{N}$ i $l = 4$:

$$\begin{pmatrix}
a_{11}^1 & a_{12}^1 & a_{13}^1 & a_{14}^1 & c_{11}^1 & 0 & 0 & 0 & \ldots \\
a_{21}^1 & a_{22}^1 & a_{23}^1 & a_{24}^1 & 0 & c_{22}^1 & 0 & 0 & \ldots \\
a_{31}^1 & a_{32}^1 & a_{33}^1 & a_{34}^1 & 0 & 0 & c_{33}^1 & 0 & \ldots \\
a_{41}^1 & a_{42}^1 & a_{43}^1 & a_{44}^1 & 0 & 0 & 0 & c_{44}^1 & \ldots \\
0 & 0 & b_{11}^2 & b_{12}^2 & a_{11}^2 & a_{12}^2 & a_{13}^2 & a_{14}^2 & \ldots \\
0 & 0 & b_{21}^2 & b_{22}^2 & a_{21}^2 & a_{22}^2 & a_{23}^2 & a_{24}^2 & \ldots \\
0 & 0 & b_{31}^2 & b_{32}^2 & a_{31}^2 & a_{32}^2 & a_{33}^2 & a_{34}^2 & \ldots \\
0 & 0 & b_{41}^2 & b_{42}^2 & a_{41}^2 & a_{42}^2 & a_{43}^2 & a_{34}^2 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}$$

Specyficzna postać macierzy **A** umożliwia znaczne zredukowanie wykonywanych operacji w stosunku do tradycyjnego algorytmu eliminacji Gaussa. Wiele elementów znajdujących się pod diagonalą jest zerami i nie będzie konieczna ich eliminacja.

W pierwszych $l-2$ kolumnach niezerowe elementy mogą się znajdować jedynie w bloku $\mathbf{A_1}$, czyli w pierwszych l rzędach. Dla kolejnych l kolumn niezerowe elementy znajdują się w pierwszych $2l$ rzędach, w kolejnych l kolumnach niezerowe elementy znajdują się w pierwszych $3l$ rzędach. Schemat ten powtarza się w następnych kolumnach, dzięki czemu można wyprowadzić wzór na numer wiersza ostatniego niezerowego elementu r_{max} w danej kolumnie c:

$$r_{\max}(c) = \min\left\{\ell + \ell \cdot \left\lfloor \frac{c+1}{\ell} \right\rfloor, n\right\}. \tag{1}$$

W każdym wierszu (poza l ostatnimi) ostatni niezerowy element należy do bloku C_i i jest odległy o l od elementów diagonalnych całej macierzy. W ostatnich l rzędach leżącym najbardziej po prawej stronie blokiem jest A_v, ostatnie niezerowe elementy leżą w kolumnie n-tej. Wzór na indeks kolumny $c_{\max}$, w której znajduje się ostatni niezerowy element w rzędzie r:

$$c_{\max}(r) = \min\{r + \ell, n\}. \tag{2}$$

Można zauważyć, że jeżeli w danym kroku eliminacji Gaussa odejmowany jest r-ty rząd od rzędów leżących poniżej, nie trzeba modyfikować elementów w kolumnach o indeksach większych niż $c_{\max}(r)$. Po wykonaniu eliminacji Gaussa należy rozwiązać układ przy pomocy algorytmu podstawiania wstecz, w którym także można ograniczyć liczbę wykonywanych operacji. Eliminacja Gaussa nie dodaje nowych elementów poza tymi znajdującymi się pod diagonalą macierzy $\mathbf{C_i}$. W każdym wierszu wystarczy sumować elementy od kolumny $c_{\max}(r)$ (2).

Przyjmując l jako stałą, złożoność obliczeniowa zmodyfikowanej metody eliminacji Gaussa wynosi $\mathcal{O}(n)$. W algorytmie eliminacji Gaussa zewnętrzna pętla wykonuje $n-1$ przebiegów, środkowa maksymalnie $2l$, a wewnętrzna maksymalnie l. W algorytmie podstawiania wstecz zewnętrzna pętla wykonuje n przebiegów, natomiast wewnętrzna maksymalnie l. Przebieg algorytmu został przedstawiony poniżej Algorithm 1:

Algorithm 1: Eliminacja Gaussa

Dane wejściowe:

A – dana w zadaniu macierz postaci,
b – wektor prawych stron,
n – rozmiar macierzy A,
l – rozmiar bloku macierzy A.

Dane wyjściowe:

x – wektor zawierający rozwiązania układu $Ax = b$.

function `eliminacja_gaussa`(A, b, n, l)

```
for k ← 1 to n − 1 do
    r_max ← in(l + l · ⌊(k+1)/ℓ⌋, n);
    c_max ← in(k + l, n);
    for i ← k + 1 to r_max do
        if A[k][k] = 0 then
            error współczynnik na przekątnej równy zeru
        end
        z ← A[i][k]/A[k][k];
        A[i][k] ← 0 ;
        for j ← k + 1 to c_max do
            A[i][j] ← A[i][j] − z · A[k][j];
        end
        b[i] ← b[i] − z · b[k];
    end
end
for i ← n downto 1 do
    c_max ← in(i + l, n);
    for j ← k + 1 to c_max do
        suma ← suma + x[i] · A[i][j];
    end
    x[i] ← (b[i] − suma)/A[i][i];
end
return x;
```

3 Eliminacja Gaussa z częściowym wyborem elementu głównego

3.1 Opis działania

Jest to modyfikacja algorytmu Gaussa, która rozwiązuje problem zer oraz numerycznie małych liczb na diagonali macierzy. Polega na znalezieniu największego elementu w kolumnie z dokładnością do wartości bezwzględnej i zastąpieniu nim elementu na diagonali poprzez przestawienie wierszy:

$$|a_{kk}| = |a_{s(k),k}| = max\{|a_{ik}| : i = k, \ldots, n\}$$

gdzie $s(k)$ jest wektorem permutacji, w którym pamiętana jest kolejność przestawień.

3.2 Dostosowanie do postaci macierzy rzadkiej A

Liczbę wykonywanych operacji można także zmniejszyć w algorytmie eliminacji Gaussa z częściowym wyborem elementu głównego. Ze względu na kosztowność zamiany wierszy wprowadzony został wektor permutacji wierszy p, w którym pamiętana jest aktualna pozycja danego wiersza. Oprócz tego wartość (2) $c_{\max}(r)$ ulega zmianie, gdyż odejmowanie wierszy w innej kolejności, może doprowadzić do powstania nowych elementów niezerowych.

Można zauważyć, że i-ty wiersz w najgorszym przypadku będzie zamieniony z wierszem o indeksie $r_{\max}(i)$, w którym ostatni niezerowy element ma indeks $c_{\max}(r_{\max}(i)) = \min\{r_{\max}(i) + l, n\}$. Z tego wynika wzór:

$$c_{\max}(i) = \min\left\{2l + l \cdot \left\lfloor \frac{i+1}{l} \right\rfloor, n\right\}, \tag{3}$$

jest to największy możliwy indeks kolumny, w której znajduje się ostatni niezerowy element w rzędzie i po zastosowaniu permutacji.

Przebieg zaimplementowanej metody przedstawia Algorithm 2. Algorytm ma większą złożoność obliczeniową niż wersja bez wyboru elementu głównego, jednak przy założeniu że l jest stałą ogólna złożoność pozostaje liniowa $\mathcal{O}(n)$.

Algorithm 2: Eliminacja Gaussa z częściowym wyborem elementu głównego

Dane wejściowe:

A – dana w zadaniu macierz postaci
b – wektor prawych stron.
n – rozmiar macierzy A,
l – rozmiar bloku macierzy A.

Dane wyjściowe:

x — wektor zawierający rozwiązania układu $Ax = b$.

function `eliminacja_gaussa_z_elementem_głównym`(A, b, n, l)

```
p ← {i : i ∈ {1, ..., n}};
for k ← 1 to n − 1 do
    r_max ← min(l + l · ⌊(k+1)/l⌋, n);
    c_max ← min(2l + l · ⌊(k+1)/l⌋, n);
    for i ← k + 1 to r_max do
        i_max ← m takie, że: A[p[m]][k] = max(|A[p[q]][k]| : q ∈ {i, ..., r_max});
        if p[i_max] = 0 then
            error macierz osobliwa
        end
        swap (p[k], p[i_max]);
        z ← A[p[i]][k]/A[p[k]][k];
        A[p[i]][k] ← 0 ;
        for j ← k + 1 to c_max do
            A[p[i]][j] ← A[p[i]][j] − z · A[p[k]][j];
        end
        b[p[i]] ← b[p[i]] − z · b[p[k]];
    end
end
for i ← n downto 1 do
    c_max ← min(2l + l · ⌊(p[i]+1)/ℓ⌋, n);
    for j ← k + 1 to c_max do
        suma ← suma + x[j] · A[p[i]][j];
    end
    x[i] ← (b[p[i]] − suma)/A[p[i]][i];
end
return x;
```

4 Rozkład LU

4.1 Opis działania

Metoda rozkładu **LU** jest powiązana z metodą eliminacji Gaussa. Polega na wyznaczeniu dwóch macierzy trójkątnych macierzy **A**: dolnej (**L**) i górnej (**U**), gdzie:

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots 0 & \\ \vdots & \vdots & \ddots 0 & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

Początkowy układ równań $\mathbf{Ax} = \mathbf{b}$ może zostać przedstawiony, jako: $\mathbf{LUx} = \mathbf{b}$. Jego rozwiązanie to rozwiązanie poniższego układu równań:

$$\mathbf{Ux} = \mathbf{b}^{'}$$
$$\mathbf{Lb}^{'} = \mathbf{b}$$

Rozkład można uzyskać za pomocą eliminacji Gaussa. Macierz $\mathbf{A}$ jest przekształcona do macierzy trójkątnej $\mathbf{U}$, a $\mathbf{L}$ jest tworzona poprzez zapamiętanie mnożników wykorzystanych do eliminacji współczynników macierzy $\mathbf{A}$. Mnożnik użyty do wyzerowania elementu a_{ij} staje się elementem macierzy $\mathbf{L}$ w wierszu i i kolumnie j. Złożoność obliczeniowa rozkładu $\mathbf{LU}$ wynosi $\mathcal{O}(n^3)$, a rozwiązanie układu równań $\mathcal{O}(n^2)$.

4.2 Dostosowanie algorytmu do postaci macierzy rzadkiej A

W celu wyznaczenia rozkładu $\mathbf{LU}$ wykorzystywany jest algorytm eliminacji Gaussa z modyfikacją (z częściowym wyborem elementu głównego oraz bez). Różnica polega na tym, że w miejsce wyeliminowanych elementów podstawiany jest mnożnik $z = a_{ik}/a_{kk}$ zamiast 0 - są to elementy macierzy $\mathbf{L}$. Układu równań z macierzą dolną i górną można rozwiązać za pomocą algorytmów podstawiania w przód i wstecz. Rozwiązanie równania $\mathbf{Ux} = \mathbf{b}^{'}$ za pomocą algorytmu podstawienia wstecz jest takie samo, jak przy eliminacji Gaussa. Równanie $\mathbf{Lb}^{'} = \mathbf{b}$ należy rozwiązać metodą podstawienia wprzód, który polega na sumowaniu elementów od kolumny wcześniejszej do kolumn dalszych. W macierzy $\mathbf{L}$ niepotrzebne jest sumowanie od pierwszej kolumny dla każdego wiersza, ze względu na zerowe elementy. Pierwszy niezerowy indeks kolumny w r-tym wierszu można przedstawić za pomocą poniższego wzoru:

$$c_{\min}(r) = \min\left\{\ell \cdot \left\lfloor \frac{r-1}{l} - 1 \right\rfloor, n\right\}, \tag{4}$$

Złożoność obliczeniowa rozkładu $\mathbf{LU}$ przy założeniu, że l jest stałą wynosi $\mathcal{O}(n)$, a rozwiązywanie układu równań z rozkładu LU wynosi także $\mathcal{O}(n)$. Przebieg algorytmu rozkładu $\mathbf{LU}$ przedstawia Algorithm 3, a jego wersję z wyborem elementu głównego Algorithm 4.

Algorithm 3: Rozwiązanie układu równań przy użyciu rozkładu LU.

Dane wejściowe:

A	–	macierz po rozkładzie LU,
b	–	wektor prawych stron.
n	–	rozmiar macierzy A,
l	–	rozmiar bloku macierzy A.

Dane wyjściowe:

x	–	wektor zawierający rozwiązania układu $Ax = b$.

function `rozwiązanie_LU(`A, b, n, l`)`

```
for i ← 1 to n do
    suma ← 0;
    c_min ← min(l · ⌊(i−1)/l⌋, n);
    for j ← c_min to i − 1 do
        suma ← suma + z[j] · A[i][j];
    end
    z[i] = b[i] − suma;
end
for i ← n downto 1 do
    suma ← 0;
    c_max ← min(i + l, n);
    for j ← i + 1 to c_max do
        suma ← suma + x[j] · A[i][j];
    end
    x[i] ← (z[i] − suma)/A[i][i];
end
return x;
```

Algorithm 4: Rozwiązanie układu równań rozkładu LU macierzy z częściowym wyborem elementu głównego.

Input :

A — macierz po rozkładzie LU,
b — wektor prawych stron,
n — rozmiar macierzy A,
l — rozmiar bloku macierzy A,
p — wektor permutacji wierszy macierzy A.

Output :

x — wektor długości n zawierający pierwiastki układu $Ax = b$.

Function `rozwiazanie_LU_z_wyborem`(A, b, n, l, p)
  for i $\leftarrow 1$ **to** n **do**
    suma $\leftarrow 0$;
    $c_{\min} \leftarrow \min\left(l \cdot \left\lfloor \frac{\mathsf{i}-1}{\ell} \right\rfloor - 1, n\right)$;
    for j $\leftarrow c_{\min}$ **to** i $- 1$ **do**
      suma $\leftarrow$ suma $+$ z[j] $\cdot A[p[\mathsf{i}]][\mathsf{j}]$;
    end
    z[i] $= b[p[\mathsf{i}]] -$ suma;
  end
  for i $\leftarrow n$ **downto** 1 **do**
    suma $\leftarrow 0$;
    $c_{\max} \leftarrow \min\left(2l + l \cdot \left\lfloor \frac{\mathsf{i}+1}{\ell} \right\rfloor, n\right)$;
    for j $\leftarrow$ i $+ 1$ **to** $c_{\max}$ **do**
      suma $\leftarrow$ suma $+$ x[j] $\cdot A[p[\mathsf{i}]][\mathsf{j}]$;
    end
    x[i] $\leftarrow$ (z[i] $-$ suma)$/A[p[\mathsf{i}]][\mathsf{i}]$;
  end
  return x;

5 Przechowywanie macierzy w pamięci

Ze względu na to, że macierz **A** jest macierzą rzadką, przechowywanie jej w dwuwymiarowej tablicy $n \times n$ jest nieefektywne. W języku `Julia` jest dostępna struktura macierzy rzadkich `SparseMatrixCSC`. Pamięta ona tylko niezerowe elementy zadanej macierzy, dzięki czemu jest oszczędzana pamięć. Macierze przechowywane są w porządku kolumnowym, a nie wierszowym, przez co dostęp do elementów macierzy jest szybszy.

6 Wyniki

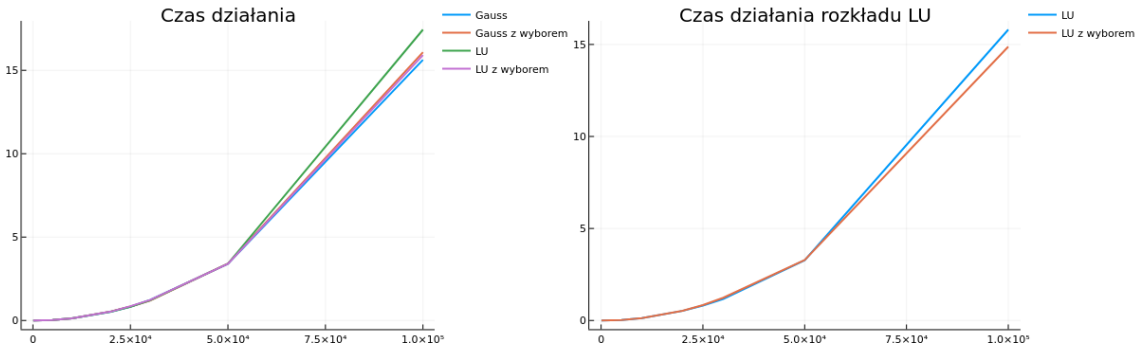
W tabelach Table 1 i Table 2 przedstawiono błędy względne poszczególnych algorytmów dla różnych rozmiarów macierzy $\mathbf{A}$. Figure 1 przedstawia wykresy rzeczywistego czasu działania oraz zużytej pamięci. Dane zostały wygenerowane dla macierzy $\mathbf{A}$ o $l = 4$.

	Eliminacja Gaussa	
n	bez wyboru	z wyborem
16	$1.389\,720\,312\,952\,042 \times 10^{-15}$	$2.513\,374\,269\,302\,153\,6 \times 10^{-16}$
100	$2.562\,908\,432\,060\,991\,8 \times 10^{-15}$	$2.719\,479\,911\,021\,036\,5 \times 10^{-16}$
500	$5.025\,488\,015\,314\,575\,5 \times 10^{-15}$	$2.548\,197\,662\,290\,045\,3 \times 10^{-16}$
2000	$6.499\,263\,982\,377\,220\,4 \times 10^{-15}$	$2.562\,066\,655\,661\,773 \times 10^{-16}$
50000	$5.450\,401\,667\,596\,608\,5 \times 10^{-14}$	$2.435\,698\,757\,587\,772\,6 \times 10^{-16}$

Tabela 1: Wartości błędu względnego rozwiązań układów równań uzyskanych przy pomocy metod Gaussa

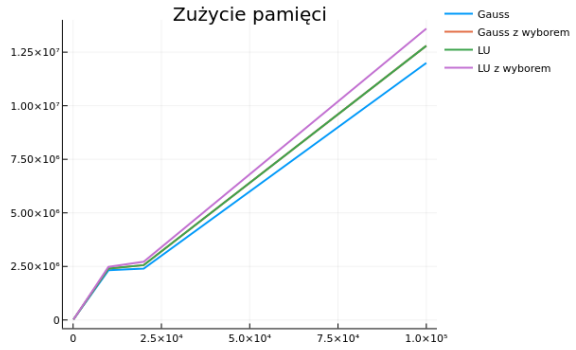
	Rozkład LU	
n	bez wyboru	z wyborem
16	$1.135\,606\,750\,051\,131\,5 \times 10^{-15}$	$2.131\,946\,216\,196\,336\,7 \times 10^{-16}$
100	$1.134\,656\,621\,724\,819\,3 \times 10^{-15}$	$2.399\,212\,249\,423\,666\,5 \times 10^{-16}$
500	$1.094\,432\,269\,765\,163\,5 \times 10^{-14}$	$2.283\,391\,675\,872\,784\,7 \times 10^{-16}$
2000	$1.723\,262\,307\,757\,433\,5 \times 10^{-14}$	$2.279\,880\,217\,973\,703\,6 \times 10^{-16}$
50000	$4.151\,765\,558\,483\,575\,4 \times 10^{-14}$	$2.271\,647\,621\,289\,407\,4 \times 10^{-16}$

Tabela 2: Wartości błędu względnego rozwiązań układów równań uzyskanych przy pomocy metod rozkładu LU.



(a)

(b)



(c)

Rysunek 1: Wykresy rzeczywistego czasu działania i zużycia pamięci dla poszczególnych algorytmów

6.1 Wnioski

Można zauważyć, że błędy względne wynikające z metod Gaussa z wyborem elementu głównego oraz LU z wyborem elementu głównego są mniejsze o co najmniej rząd wielkości niż przy metodach bez wyboru elementu głównego. Metody Gaussa i LU bez wyboru elementu głównego są szybsze niż te z wyborem. Na rysunku (b) przedstawiono czas samego rozkładu LU, na rysunku (a) pokazany jest czas rozkładu i rozwiązania układu. Widać, że samo rozwiązanie z posiadanego rozkładu zajmuje niewielką ilość czasu. Rozwiązywanie pojedynczych układów równań metodą LU i mniej opłacalne niż metodą Gaussa, jednak przy rozwiązywaniu układów równań dla jednej macierzy i wielu wektorów prawych stron metoda LU jest lepsza. Zastosowane modyfikacje pozwalają na rozwiązanie układu równań dla z dużą liczbą niewiadomych, co byłoby niemożliwe bez optymalizacji.