Obliczenia naukowe - lista 5.

Piotr Kołodziejczyk

Styczeń 2020

1 Wstęp

Zadanie polega na rozpatrzeniu problemu rozwiązywania układu równań liniowych $\boldsymbol{A}x = b$ dla specyficznej postaci macierzy $\boldsymbol{A}$, w szczególności z uwzględnieniem dużych rozmiarów macierzy.

2 Specyfikacja danych wejściowych

Dana jest macierz $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ i wektor prawych stron $b \in \mathbb{R}^n$, gdzie $n \geqslant 4$. Macierz $\boldsymbol{A}$ jest rzadka i blokowa, postaci:

$$\begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{C}_1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & ... & \boldsymbol{0} \\ \boldsymbol{B}_2 & \boldsymbol{A}_2 & \boldsymbol{C}_2 & \boldsymbol{0} & \boldsymbol{0} & ... & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_3 & \boldsymbol{A}_3 & \boldsymbol{C}_3 & \boldsymbol{0} & ... & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_4 & \boldsymbol{A}_4 & \boldsymbol{C}_4 & ... & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & 0 & \boldsymbol{B}_v & \boldsymbol{A}_v \end{pmatrix},$$

gdzie $v = \frac{n}{l}$, zakładając, że $l|n$ a l to wielkość pojedynczego kwadratowego bloku, zaś same bloki przedstawiają się następująco:

- Macierz $\boldsymbol{A_k} \in \mathbb{R}^{l \times l}$ jest macierzą gęstą,
- Macierz $\boldsymbol{B_k} \in \mathbb{R}^{l \times l}$:

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

- Macierz przekątniowa $\boldsymbol{C}_k \in \mathbb{R}^{l \times l}$:

$$\begin{pmatrix} c^k_1 & 0 & \cdots & 0 \\ 0 & c^k_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c^k_l \end{pmatrix}.$$

3 Przegląd algorytmów

Podstawowym algorytmem do rozwiązywania układu równań liniowych jest metoda eliminacji Gaussa, często zaimplementowana w podstawowych bibliotekach matematycznych. Jej złożoność wynosi jednak $\mathcal{O}(n^3)$, przez co dla macierzy rozmiaru rzędu kilkudziesięciu tysięcy staje się nieefektywna, a ponadto zapamiętanie macierzy tego rozmiaru explicite może być niemożliwe pamięciowo. Jednakowoż specyficzna struktura zadanej macierzy pozwala tak zmodyfikować metodę Gaussa, by osiągnąć nawet złożoność liniową.
Poniżej zaprezentuję zmodyfikowane algorytmy bazujące na metodzie eliminacji Gaussa - klasycznej oraz z wyborem elementów głównych - dostosowane do postaci macierzy $\boldsymbol{A}$, wraz z rozkładem $\boldsymbol{LU}$. Ponieważ macierz jest rzadka a jej pamiętanie w dwuwymiarowej tablicy nieefektywne, w implementacji używam struktury `SparseArray`, posiadającej informację tylko o niezerowych elementach macierzy.

4 Metoda eliminacji Gaussa

Ideą eliminacji Gaussa jest takie przekształcenie pierwotnej macierzy, aby w konsekwencji otrzymać macierz trójkątną górną, z której łatwo już wyznaczyć rozwiązanie algorytmem podstawiania wstecz. W każdym kroku algorytmu eliminacji staramy się wyzerować kolejny element pod przekątną macierzy. W pierwszym kroku wybieram pierwszy współczynnik pierwszego wiersza a_{11} i dla każdego niezerowego elementu a_{j1} poniżej w pierwszej kolumnie, wyznaczam mnożnik $\frac{a_{j1}}{a_{11}}$, by następnie od j-tego wiersza odjąć przemnożony przez ów mnożnik wiersz pierwszy. Krok ten powtarzamy dla każdej kolumny, aby w konsekwencji otrzymać macierz z niezerowymi elementami tylko ponad przekątną. Ponadto, modyfikując j-ty wiersz macierzy, należy zmodyfikować tak samo j-ty wiersz wektora prawych stron, by układy były tożsame. Poniższy pseudokod przedstawia ów algorytm.

Algorithm 1 Algorytm Gaussa

procedure GAUSS($n, (a_{ij}), b$)
  for $k = 1$ **to** $n - 1$ **do** ▷ Pętla 1
    for $i = k + 1$ **to** n **do** ▷ Pętla 2
      $multiplier \leftarrow \frac{a_{ik}}{a_{kk}}$
      $a_{ik} \leftarrow 0$
      for $j = k + 1$ **to** n **do** ▷ Pętla 3
        $a_{ij} \leftarrow a_{i_j} - multiplier \cdot a_{kj}$
      $b_i \leftarrow b_i - multiplier \cdot b_i$
  for $i = n$ **downto** 1 **do**
    $x_i \leftarrow \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}$
  return $\overline{x}$ ▷ Wektor rozwiązania

4.1 Dostosowanie metody Gaussa

Dostosowanie metody eliminacji Gaussa do naszej postaci macierzy polega na zawężeniu przedziałów iteracji w pętlach. Spójrzmy na przykładowy fragment macierzy:

$$\begin{pmatrix} a_{11}^1 & a_{12}^1 & a_{13}^1 & c_{11}^1 & 0 & 0 & 0 & 0 & \cdots \\ a_{21}^1 & a_{22}^1 & a_{23}^1 & 0 & c_{22}^1 & 0 & 0 & 0 & \cdots \\ a_{31}^1 & a_{32}^1 & a_{33}^1 & 0 & 0 & c_{33}^1 & 0 & 0 & \cdots \\ 0 & b_{12}^2 & b_{13}^2 & a_{11}^2 & a_{12}^2 & a_{13}^2 & c_{11}^2 & 0 & \cdots \\ 0 & b_{22}^2 & b_{23}^2 & a_{21}^2 & a_{22}^2 & a_{23}^2 & 0 & c_{22}^2 & \cdots \\ 0 & b_{32} & b_{33} & a_{31} & a_{32} & a_{33} & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Założeniem algorytmu jest wyzerowanie wszystkich elementów poniżej przekątnej, a więc poniżej elementów a_{ii}, gdzie jednak już znajduje się znaczna większość elementów niezerowych. Możemy więc w pętli 2 iterować tylko od $k+1$ do $\min(n, k+l+1)$ (w dół kolumny). Podobnie w pętli 3, ponieważ dalsze elementy w wierszu są zerowe, dodawanie ich do innych mija się z celem, więc pętla 3 może być od $k+1$ do $\min(n,\ k+l)$ (co najwyżej te komórki w k-tym wierszu są niezerowe). Dodatkowo, sumowanie elementów w wierszu przy obliczaniu kolejnych x-ów również można ograniczyć. Daje to następujący algorytm:

Algorithm 2 Zmieniony algorytm Gaussa

1: **procedure** ModifiedGauss$(n, (a_{ij}), b)$
2: **for** $k = 1$ **to** $n-1$ **do** ▷ Pętla 1
3: **for** $i = k+1$ **to** $\texttt{min}(n, k+l+1)$ **do** ▷ Pętla 2
4: $multiplier \leftarrow \frac{a_{ik}}{a_{kk}}$
5: $a_{ik} \leftarrow 0$
6: **for** $j = k+1$ **to** $\texttt{min}(n, k+l)$ **do** ▷ Pętla 3
7: $a_{ij} \leftarrow a_{i_j} - multiplier \cdot a_{kj}$
8: $b_i \leftarrow b_i - multiplier \cdot b_i$
9: **for** $i = n$ **downto** 1 **do**
10: $x_i \leftarrow \frac{b_i - \sum_{j=i+1}^{min(n,i+l)} a_{ij} x_j}{a_{ii}}$ ▷ Pętla 4
11: **return** $\overline{x}$ ▷ Wektor rozwiązania

Pętla 1. wykonuje się $n-1$ razy, natomiast pętle 2. i 3. co najwyżej po l(szacowanie z nadmiarem), więc jeśli l jest stałą, to algorytm wyznaczenia macierzy wykonuje się w czasie liniowym. Wyznaczenie rozwiązania to n iteracji w pętli zewnętrznej i co najwyżej l potrzebnych do obliczenia sumy (pętla 4.). Złożoność to zatem $\mathcal{O}(n)$.

5 Metoda Gaussa z wyborem elementów głównych

Metoda Gaussa potrafi zawodzić już dla prostych macierzy - wystarczy by któryś element na przekątnej był równy (lub bliski) zeru. W pierwszym przypadku zawodzi sam algorytm, w drugim mogą występować problemy numeryczne. Aby sobie z tym poradzić, stosuje się tzw. wybór elementów głównych. Polega on na odpowiednim spermutowaniu kolejności wierszy macierzy, aby po pierwsze elementy na głównej macierzy były niezerowe, oraz miały stosunkowo duże wartości. W tym przypadku zastosuję częściowy wybór elementów głównych, który polega na takim przestawieniu wierszy,

by na przekątnej znajdowała się wartość maksymalna w danej kolumnie (wyboru dokonuje się dla każdej kolumny na na bieżąco modyfikowanej macierzy, spośród wierszy poniżej przekątnej).

Poniżej przedstawiony jest pseudokod wyznaczania macierzy powyższą metodą. Znalezienie rozwiązania przebiega jak w przypadku algorytmu 2., z uwzględnieniem permutacji w kolejności wierszy macierzy i wektora prawych stron b.

Algorithm 3 Zmieniony algorytm Gaussa z wyborem elementu głównego

1: **procedure** ModifiedChoiceGauss$(n, (a_{ij}), b)$
2: $\overline{p} \leftarrow [1..n]$
3: **for** $k = 1$ **to** $n - 1$ **do** ▷ Pętle 1 i 2
4: wybierz j, że $A_{p_j k}$ maksymalne poniżej przekątnej w kolumnie k
5: $p_k \leftrightarrow p_j$
6: **for** $i = k + 1$ **to** $\texttt{min}(n, k + l + 1)$ **do** ▷ Pętla 3
7: $multiplier \leftarrow \frac{a_{p_i k}}{a_{p_k k}}$
8: $a_{p_i k} \leftarrow 0$
9: **for** $j = k + 1$ **to** $\texttt{min}(n, k + 2 * l)$ **do** ▷ Pętla 4
10: $a_{p_i j} \leftarrow a_{p_i j} - multiplier \cdot a_{p_k j}$
11: $b_{p_i} \leftarrow b_{p_i} - multiplier \cdot b_{p_i}$
12: **return** a_{ij} ▷ Macierz Gaussa

Pętla 1. wykonuje się $n - 1$ razy, pętla 2. l razy, natomiast pętle 3. i 4. co najwyżej odpowiednio l oraz $2 \cdot l$ (szacowanie z nadmiarem), więc jeśli l jest stałą, to algorytm wyznaczenia macierzy wykonuje się w czasie liniowym. Wyznaczenie rozwiązania jak w algorytmie wyżej, więc złożoność to $\mathcal{O}(n)$.

6 Rozkład LU

Przy rozwiązywaniu układów równań liniowych można się również posłużyć rozkładem $\boldsymbol{LU}$ macierzy. Polega on na przedstawieniu pierwotnej macierzy jako iloczynu dwu o specyficznej postaci - macierz $\boldsymbol{L}$ to macierz trójkątna dolna, zaś $\boldsymbol{U}$ - macierz trójkątna górna. Efektywnym sposobem dokonania owego rozkładu jest użycie eliminacji Gaussa, gdzie macierz dolną $\boldsymbol{L}$ wypełnią wyliczane w kolejnych krokach mnożniki, zaś macierzą górną $\boldsymbol{U}$ uzyskana macierz Gaussa. Przedstawia to poniższy algorytm, już z dostosowanymi zakresami pętli do specyficznej postaci macierzy wejściowej.

Złożoność powyższego algorytmu jest asymptotycznie taka sama, jak algorytmu 2.

6.1 Rozwiązanie układu na podstawie $\boldsymbol{LU}$

Jeżeli macierz $\boldsymbol{A}$ da się wyrazić jako iloczyn macierzy trójkątnej dolnej i górnej $\boldsymbol{LU}$, to rozwiązanie układu $\boldsymbol{A}x = b$ sprowadza się do rozwiązania dwu prostych układów:

- rozwiązanie $\boldsymbol{L}z = b$ względem z
- rozwiązanie $\boldsymbol{U}x = z$ względem x,

które rozwiązujemy algorytmami podstawienia odpowiednio w przód i wstecz. Poniżej pseudokod, który przedstawią ową procedurę z dostosowaniem do postaci macierzy.

Każda z dwu pętli zewnętrznych wykonuje się n razy, a pętle wewnętrzne co najwyżej l razy, więc problem pozostaje w klasie złożoności liniowej.

Algorithm 4 Rozkład LU

```
1: procedure LU(n, (a_ij))
2:     L_ij ← 0   ∀i, j ⩽ n
3:     U_ij ← (a_ij)
4:     for k = 1 to n − 1 do                                   ▷ Pętla 1
5:         L_kk ← 1.0
6:         for i = k + 1 to min(n, k + l + 1) do               ▷ Pętla 2
7:             multiplier ← a_ik / a_kk
8:             L_ik ← multiplier
9:             U_ik ← 0
10:            for j = k + 1 to min(n, k + l) do               ▷ Pętla 3
11:                U_ij ← U_i_j − multiplier · U_kj
12:    return L, U                                             ▷ Macierze L i U
```

Algorithm 5 Rozwiązanie na podstawie LU

```
1: procedure LUSolve(n, (L_ij), (U_ij), b̄)
2:     for i = 1 to n − 1 do
3:         z_i ← b_i − Σ_{j=i+1}^{min(n,i+l+1)} L_ij · z_j
4:     for i = n downto 1 do
5:         x_i ← z_i − Σ_{j=i+1}^{min(n,i+l)} U_ij · x_j
6:     return x̄
```

6.2 Rozkład *LU* z wyborem elementów głównych

Podobnie jak w przypadku zwykłej eliminacji Gaussa, by zapobiec błędom numerycznym i zwykłej niemożności rozwiązania układu przez element zerowy na przekątnej, przy rozkładzie $\boldsymbol{LU}$ stosuje się wybór elementów głównych. W tym przypadku, tak samo jak poprzednio, dokonuję takiej permutacji w kolejności wierszy macierzy, by element na głównej przekątnej był największy spośród wartości w danej kolumnie (w dół od przekątnej).

Algorytm zatem wygląda jak algorytm 4., z dodaniem wektora permutacji jak w algorytmie 2. Analogicznie, algorytm znalezienia rozwiązania to algorytm 5. z dokładnością do permutacji kolejności wierszy.

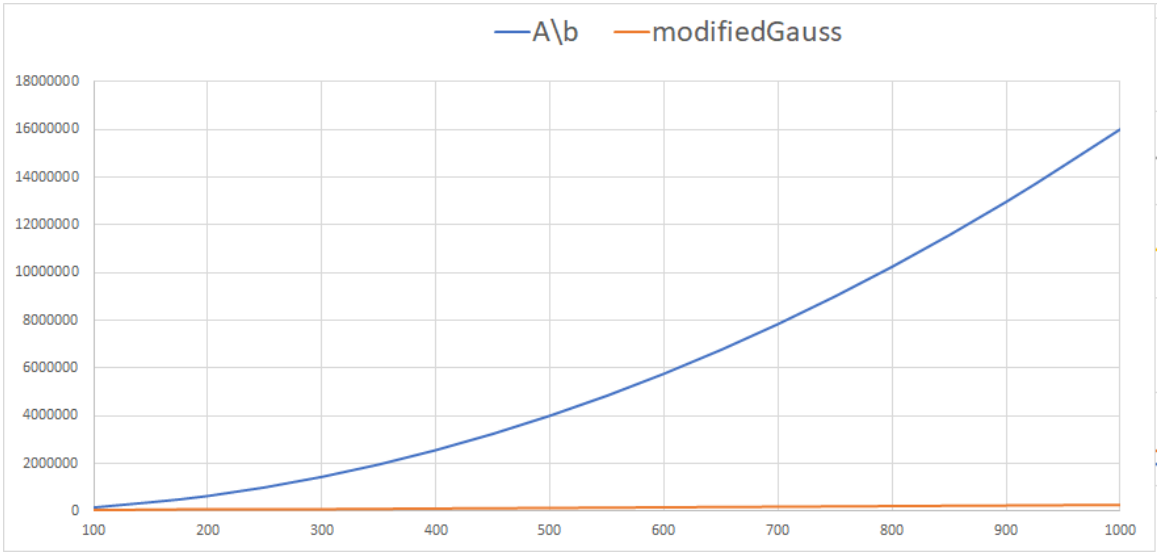
7 Przykłady i testy

Aby sprawdzić poprawność zaimplementowanych algorytmów generowana jest macierz wejściowa $\boldsymbol{A}$ oraz wektor b (równaniem $b = \boldsymbol{A} \cdot x$, gdzie x to wektor jednostkowy) - stąd wiemy, że za każdym razem rezultatem powinien być właśnie wektor jednostkowy.

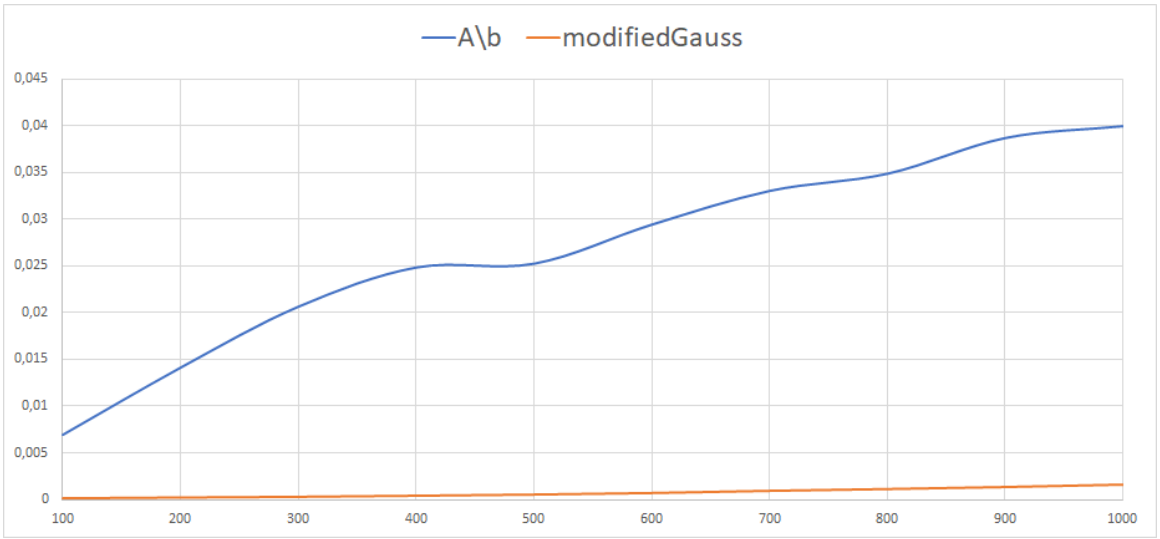
Pierwsze dwa wykresy przedstawiają porównanie czasowe i pamięciowe bibliotecznej metody Gaussa z tą przeze mnie zaimplementowaną. Ewidentnie autorskie metody radzą sobie ze specyficzną postacią macierzy wejściowej asymptotycznie lepiej.

Porównanie czasowe zaimplementowanych metod pokazuje, że wariant z wyborem wymaga więcej czasu, oraz że generowanie rozkładu$\boldsymbol{LU}$ jest nieznacznie dłuższe.

Zgodnie z intuicją, generowanie i zapamiętanie rozkładu $\boldsymbol{LU}$ jest bardziej kosztowne pamięciowo.a



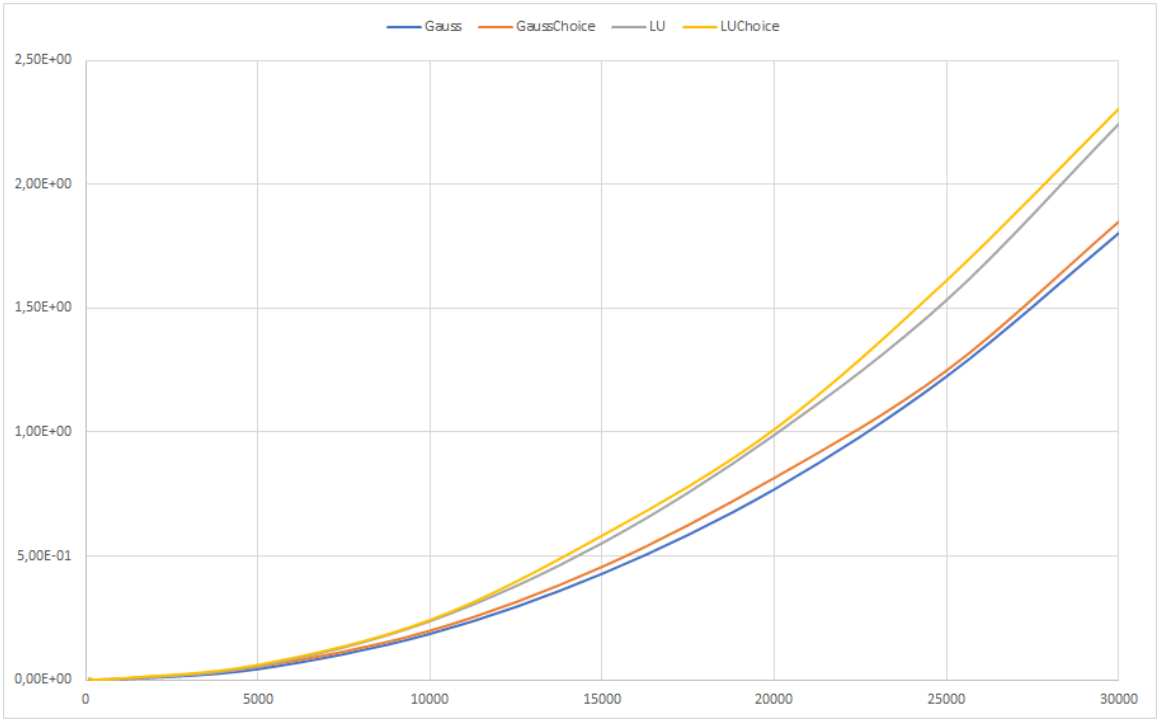
Rysunek 1: Porównanie nakładu pamięciowego metody bibliotecznej i zaimplementowanego algorytmu



Rysunek 2: Porównanie czasu wykonania metody bibliotecznej i zaimplementowanego algorytmu

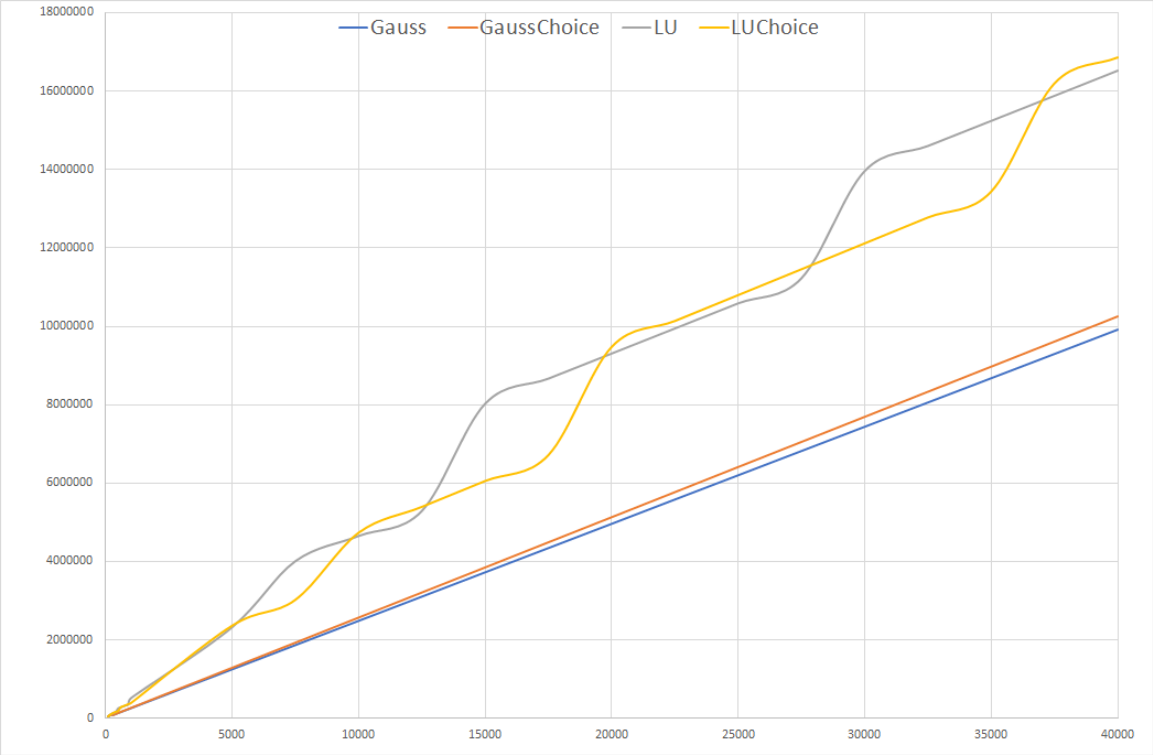
8 Wnioski

Standardowe metody znalezienia rozwiązania układu równań liniowych mogą zawieść dla danych dużego rozmiaru. Zaimplementowane algorytmy zmodyfikowanej metody eliminacji Gaussa (zwykłej oraz z częściowym wyborem elementu głównego), nie dość że radzą sobie lepiej pamięciowo (pamiętamy tylko niezerowe elementy, które stanowią znaczną większość całej macierzy), to obliczenia wykonują się efektywnie szybko. Wybór elementu głównego pozwala na uzyskanie wyniku w



Rysunek 3: Porównanie czasu wykonania zaimplementowanych algorytmów

przypadku, gdy na głównej przekątnej macierzy pojawia się element zerowy, lub na dokładniejszy wynik, gdy na przekątnej pojawi się bardzo mały element. Obliczenie rozkładu $\boldsymbol{LU}$ jest bardziej czasochłonne, jednak może być użyteczne, jeśli dla tej samej macierzy szukamy rozwiązań dla różnych wektorów prawych stron - raz znaleziony rozkład może być użyty ponownie.



Rysunek 4: Porównanie nakładu pamięciowego zaimplementowanych algorytmów