Obliczenia naukowe
Sprawozdanie 5

Józef Piechaczek

29 grudnia 2019

Opis problemu

W zadaniu mamy do czynienia z problemem rozwiązania układu równań liniowych

$$Ax = b$$

dla danej macierzy współczynników $\mathbf{A} \in \mathbb{R}^{n \times n}$ i wektora prawych stron $b \in \mathbb{R}^n, n \geqslant 4$ Macierz $\mathbf{A}$ jest rzadką i blokową o następującej strukturze:

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

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

$$\mathbf{B}_k = \begin{pmatrix} 0 & \cdots & 0 & b^k_{1\ell-1} & b^k_{1\ell} \\ 0 & \cdots & 0 & b^k_{2\ell-1} & b^k_{2\ell} \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & b^k_{\ell\ell-1} & b^k_{\ell\ell} \end{pmatrix} \tag{2}$$

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

$$\mathbf{C}_k = \begin{pmatrix} c^k_1 & 0 & 0 & \cdots & 0 \\ 0 & c^k_2 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c^k_{\ell-1} & 0 \\ 0 & \cdots & 0 & 0 & c^k_\ell \end{pmatrix} \tag{3}$$

Macierz $\mathbf{A}$ jest bardzo dużego rozmiaru (n jest bardzo duże) co wyklucza pamiętanie macierzy jako dwuwymiarowej tablicy oraz użycie standardowych (bibliotecznych) algorytmów dla macierzy gęstych. Należy zatem użyć specjalnej struktury pamiętającej efektywnie macierz $\mathbf{A}$, która pamięta tylko elementy niezerowe oraz zaadaptować algorytmy tak, aby uwzględniały specyficzną postać macierzy. Zakładając, że ℓ jest stałe, czas rozwiązywania układu powinien zostać zredukowany do $O(n)$.

1 Zadanie 1

Zadanie 1 polega na napisaniu funkcji rozwiązującej układ $Ax = b$ metodą eliminacji Gaussa z uwzględnieniem specyficznej postaci macierzy $\mathbf{A}$ dla dwóch wariantów:

(a) bez wyboru elementu głównego,

(b) z częściowym wyborem elementu głównego.

Metoda eliminacji Gaussa

Metoda eliminacji Gaussa to algorytm rozwiązywania układów liniowych i wyznaczania rozkładu **LU** wykorzystujący trzy typy operacji elementarnych:

(a) Zamiana dwóch wierszy

(b) Mnożenie wiersza przez element niezerowy

(c) Dodawanie wielokrotności jednego rzędu do innego

Za pomocą tych operacji jesteśmy w stanie zamienić macierz do postaci trójkątnej górnej. Algorytm polega na używaniu operacji elementarnych w ten sposób, aby w danej iteracji pod wartością na diagonali znajdowały się tylko wartości 0. W tym celu od wierszy znajdujących się poniżej rozpatrywanego odejmujemy taką wielokrotność rozpatrywanego rzędu, aby wyzerować wartość w rozpatrywanej kolumnie. Gdy macierz znajduje się w postaci trójkątnej górnej możemy użyć algorytmu *podstawienia wstecz* do łatwego rozwiązania układu. Algorytm *podstawienia wstecz* polega na rozpoczęciu rozwiązywania układu od dołu układu, gdzie uzyskujemy równanie $a_n x_n = b_n$, zatem $x_n = b_n/a_n$, następnie poprzednio uzyskane rozwiązana używamy do rozwiązania równania znajdującego się wyżej, gdzie obliczamy kolejną wartość x_i. Algorytm kończy się, gdy przejdziemy po wszystkich wierszach. Standardowy algorytm zamiany do postaci trójkątnej górnej ma złożoność czasową $O(n^3)$, a algorytm *podstawienia wstecz* $O(n^2)$, co daje łączny czas $O(n^3)$.

Algorytm 1 Metoda Gaussa

Require: $n, (a_{ij}), b_i$
  {**Pętla 1**}
  for $k = 1$ **to** $n - 1$ **do**
    {**Pętla 2**}
    for $i = k + 1$ **to** n **do**
      $z \leftarrow a_{ik}/a_{kk}$
      $a_{ik} \leftarrow 0$
      {**Pętla 3**}
      for $j = k + 1$ **to** n **do**
        $a_{ij} \leftarrow a_{ij} - z a_{kj}$
      end for
      $b_i \leftarrow b_i - z b_k$
    end for
  end for
  {**Pętla 4**}
  for $i = n$ **to** 1 **step** -1 **do**
    $x_i \leftarrow (b_i - \sum_{j=i+1}^{n} a_{ij} x_j)/a_{ii}$
  end for

Metoda eliminacji Gaussa z częściowym wyborem elementu głównego

Metoda eliminacji Gaussa z częściowym wyborem elementu głównego zawiera jedną modyfikację w porównaniu do tradycyjnego algorytmu. Na początku każdej iteracji zewnętrznej pętli algorytmu szukamy wiersza, który zaczyna się od bezwzględnie największej wartości, a następnie zamieniamy

ten wiersz z aktualnie rozpatrywanym. Umożliwia to rozwiązanie układu, gdy na diagonali pojawią się zera. Zamiana wierszy w praktyce może okazać się kosztowną operacją, dlatego w algorytmie stosujemy tablicę **p**, początkowo zawierającą wartości od 1 do n. W momencie, gdy powinniśmy zamienić wiersze, zamieniamy odpowiadające im wartości w tabeli **p**. Za każdym razem gdy odnosimy się do określonego wiersza używamy wartości z tabeli **p**.

Algorytm 2 Metoda Gaussa z częściowym wyborem elementu głównego

Require: $n, a_{ij}), b_i$
  $p \leftarrow [n]$
  {**Pętla 1**}
  for $k = 1$ **to** $n - 1$ **do**
    wybór takiego $j \geqslant k$, że $|a_{p_j k}| \geqslant |a_{p_k k}|$
    $p_j \leftrightarrow p_k$
    {**Pętla 2**}
    for $i = k + 1$ **to** n **do**
      $z \leftarrow a_{p_i k}/a_{p_k k}$
      $a_{p_i k} \leftarrow 0$
      {**Pętla 3**}
      for $j = k + 1$ **to** n **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
  {**Pętla 4**}
  for $i = n$ **to** 1 **step** -1 **do**
    $x_i \leftarrow (b_{p_i} - \sum_{j=i+1}^{n} a_{p_i j} x_j)/a_{p_i i}$
  end for

Zmodyfikowany algorytm

Analizując specyficzną budowę macierzy **A** możemy dostrzec, że nie jest konieczne iterowanie po wszystkich wierszach dla **pętli 2**, ponieważ poniżej pewnego wiersza, dla danej kolumny, będą się znajdować wyłącznie elementy zerowe. Możemy dostrzec, że indeks wiersza ostatniego elementu niezerowego w danej kolumnie uzależniony jest od wielkości bloków w macierzy **A** oraz od aktualnie rozpatrywanego elementu. Dla pierwszych $\ell - 2$ kolumn indeksem najniżej położonego niezerowego elementu jest ostatni wiersz macierzy $\mathbf{A}_1$. Dla kolejnych ℓ kolumn indeksem najniżej położonego niezerowego elementu są ostatnie wiersze macierzy $\mathbf{B}_1$ i $\mathbf{A}_2$, dla kolejnych ℓ elementów wiersze macierzy $\mathbf{B}_2$ i $\mathbf{A}_3$ itd. Ponieważ schemat będzie się powtarzał, możemy wyprowadzić wzór na indeks wiersza ostatniego elementu niezerowego dla danej kolumny:

$$r_k = min\{n; (k + (\ell + 1) - (k + 1)\%\ell)\} \tag{4}$$

Podobną zależność można zauważyć dla iteracji po kolumnach dla danego wiersza w **pętli 3** oraz przy wyznaczaniu **sumy w pętli 4**. Na prawo od pewnej kolumny dla danego wiersza będą znajdować się tylko elementy zerowe. Ostatnim niezerowym elementem dla danego wiersza, jest znajdujący się na diagonali macierzy $\mathbf{C}_k$, zatem w odległości ℓ od rozpatrywanego elementu. Możemy zatem

wyprowadzić wzór na indeks kolumny ostatniego elementu niezerowego dla danego wiersza:

$$c_k = min\{n; k + \ell\} \tag{5}$$

W przypadku algorytmu z częściowym wyborem elementu głównego indeks kolumny ostatniego elementu niezerowego dla danego wiersza, może być większy, gdyż uległ zmianie w wyniku zamiany wierszy. Dany wiersz mógł zostać zamieniony z jednym z maksymalnie $\ell + 1$ wierszy znajdujących się poniżej, zatem musimy rozpatrzyć wartość dla ostatniego wiersza, z którym mogliśmy zamienić aktualnie rozpatrywany wiersz. We wzorze na c_k zamiast k musimy zatem podstawić wartość $k+\ell+1$ co daje nam następujący wzór:

$$c_k = min\{n; k + 2\ell + 1\} \tag{6}$$

Wzór r_k należy również użyć przy **wyborze elementu głównego na początku pętli 1**.

Otrzymane algorytmy

Algorytm 3 Zmodyfikowana metoda Gaussa

Require: $n, l, (a_{ij}), b_i$
  {**Pętla 1**}
  for $k = 1$ **to** $n - 1$ **do**
    {**Pętla 2**}
    for $i = k + 1$ **to** r_k **do**
      $z \leftarrow a_{ik}/a_{kk}$
      $a_{ik} \leftarrow 0$
      {**Pętla 3**}
      for $j = k + 1$ **to** c_k **do**
        $a_{ij} \leftarrow a_{ij} - za_{kj}$
      end for
      $b_i \leftarrow b_i - zb_k$
    end for
  end for
  {**Pętla 4**}
  for $i = n$ **to** 1 **step** -1 **do**
    $x_i \leftarrow (b_i - \sum_{j=i+1}^{c_i} a_{ij}x_j)/a_{ii}$
  end for

Algorytm 4 Zmodyfikowana metoda Gaussa z częściowym wyborem elementu głównego

Require: $n, l, (a_{ij}), b_i$
  $p \leftarrow [n]$
  {**Pętla 1**}
  for $k = 1$ **to** $n - 1$ **do**
    wybór takiego j, że $r_k \geqslant j \geqslant k$, że $|a_{p_j k}| \geqslant |a_{p_k k}|$
    $p_j \leftrightarrow p_k$
    {**Pętla 2**}
    for $i = k + 1$ **to** r_k **do**
      $z \leftarrow a_{p_i k}/a_{p_k k}$
      $a_{p_i k} \leftarrow 0$
      {**Pętla 3**}
      for $j = k + 1$ **to** c_k **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
  {**Pętla 4**}
  for $i = n$ **to** 1 **step** -1 **do**
    $x_i \leftarrow (b_{p_i} - \sum_{j=i+1}^{c_i} a_{p_i j} x_j)/a_{p_i i}$
  end for

Analizując wyżej podany pseudokod, możemy zauważyć, że jedynie **Pętla 1** jest zależna od n, reszta pętli (i sumowań) wykonuje się maksymalnie 2ℓ razy, a zatem w czasie stałym. Zatem złożoność obu algorytmów wynosi $O(n)$

2 Zadanie 2

Zadanie drugie polega na napisaniu funkcji wyznaczającej 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.

Rozkład LU

Rozkład **LU** polega na zastąpieniu macierzy **A** iloczynem macierzy trójkątnej dolnej oraz trójkątnej górnej $\mathbf{A} = \mathbf{LU}$. Rozkład można policzyć korzystając z metody eliminacji Gaussa. Macierz **L** przechowuje wartości mnożników obliczanych w metodzie Gaussa, a macierz **U** zredukowaną macierz **A**. Aby zaadaptować **Algorytm 3** do obliczania rozkładu **LU** należy dokonać trzech modyfikacji: za wartość a_{ik} w pętli 1 przyjąć wartość mnożnika z, nie modyfikować wektora b oraz nie wykonywać *podstawienia wstecz*. Obliczanie rozkładu **LU** umożliwia nam ograniczenie liczby obliczeń w przypadku zmiany wektora b, gdyż po zmianie wektora będzie można wykorzystać obliczony już rozkład. Macierze **L** i **U** będę przechowywać w wejściowej macierzy **A**. W tym wypadku macierz **A** będzie

wyglądać następująco:

$$\mathbf{A} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1_n} \\ l_{21} & u_{22} & u_{23} & \cdots & u_{2n} \\ l_{31} & l_{32} & u_{33} & \cdots & u_{3n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & u_{nn} \end{pmatrix} \tag{7}$$

przy domniemaniu, że $\forall i \in [n], l_{ii} = 1$.

Rozkład **LU** ma taką samą złożoność obliczeniową jak **Algorytm 3** wynoszącą $O(n)$ przy założeniu, że ℓ jest stałe.

Algorytm 5 Rozkład LU

Require: $n, l, (a_{ij}), b_i$
  {**Pętla 1**}
  for $k = 1$ **to** $n - 1$ **do**
    {**Pętla 2**}
    for $i = k + 1$ **to** r_k **do**
      $z \leftarrow a_{ik}/a_{kk}$
      $a_{ik} \leftarrow z$
      {**Pętla 3**}
      for $j = k + 1$ **to** c_k **do**
        $a_{ij} \leftarrow a_{ij} - za_{kj}$
      end for
    end for
  end for

Rozkład LU z częściowym wyborem elementów głównych

Rozkład **LU** z częściowym wyborem elementów głównych wygląda analogicznie do **Algorytmu 4**, z modyfikacjami opisanymi wyżej.

Algorytm 6 Rozkład LU z częściowym wyborem elementów głównych

Require: $n, l, (a_{ij}), b_i$
  $p \leftarrow [n]$
  {**Pętla 1**}
  for $k = 1$ **to** $n - 1$ **do**
    wybór takiego j, że $r_k \geqslant j \geqslant k$, że $|a_{p_j k}| \geqslant |a_{p_k k}|$
    $p_j \leftrightarrow p_k$
    {**Pętla 2**}
    for $i = k + 1$ **to** r_k **do**
      $z \leftarrow a_{p_i k}/a_{p_k k}$
      $a_{p_i k} \leftarrow z$
      {**Pętla 3**}
      for $j = k + 1$ **to** c_k **do**
        $a_{p_i j} \leftarrow a_{p_i j} - z a_{p_k j}$
      end for
    end for
  end for

3 Zadanie 3

Zadanie 3 polega na rozwiązaniu układu $\mathbf{Ax} = \mathbf{b}$, jeśli wcześniej został policzony rozkład **LU**. Zatem obliczenie układu sprowadza się do rozwiązania wyrażenia $\mathbf{LUx} = \mathbf{b}$. Ponieważ macierze **L** i **U** są schodkowe, do rozwiązania użyjemy algorytmu *podstawienia wstecz/w przód*. Zatem rozwiązanie dzieli się na dwa etapy.

$$\begin{cases} \mathbf{Ly} = \mathbf{b} \\ \mathbf{Ux} = \mathbf{y} \end{cases} \tag{8}$$

Zmodyfikowany algorytm

Standardowy algorytm obliczania wartości **x** przy obliczonym już rozkładzie **LU** ma złożoność $O(n^2)$. Aby zmniejszyć złożoność musimy zredukować liczbę iteracji podczas sumowania. W wypadku *podstawienia wstecz*, które ma miejsce w drugiej pętli algorytmu użyjemy wcześniej opisanej wartości c_i określającej ostatni element niezerowy w danym wierszu. W wypadku *podstawienia w przód* musimy określić od której kolumny należy zacząć sumowanie. W wypadku pierwszych ℓ wierszy sumowanie zaczynamy od pierwszej kolumny, dla następnych ℓ wierszy sumowanie zaczynamy od pierwszej niezerowej kolumny macierzy $\mathbf{B}_1$, dla następnych ℓ od pierwszej niezerowej kolumny macierzy $\mathbf{B}_2$ itd. Zatem otrzymujemy następujący wzór na indeks kolumny pierwszego niezerowego elementu w danym wierszu:

$$g_k = min\{1, k - (2 + (k - 1)\%l)\} \tag{9}$$

Przy założeniu, że ℓ jest stałe, złożoność czasowa algorytmu zostaje zredukowana do $O(n)$

Algorytm 7 Rozwiązywanie układu równań z obliczonym rozkładem **LU**

Require: $n, l, (a_{ij}), b_i$
  {**Pętla 1**}
  for $i = 1$ **to** n **do**
    $y_i \leftarrow b_i - \sum_{j=g_i}^{i-1} a_{ij} y_j$
  end for
  {**Pętla 2**}
  for $i = n$ **to** 1 **step** -1 **do**
    $x_i \leftarrow (y_i - \sum_{j=(i+1)}^{c_i} a_{ij} x_j)/a_{ii}$
  end for

Algorytm 8 Rozwiązywanie układu równań z obliczonym rozkładem **LU** z cz. wyborem el. głównego

Require: $n, l, (a_{ij}), b_i, p_i$
  {**Pętla 1**}
  for $i = 1$ **to** n **do**
    $y_i \leftarrow b_{p_i} - \sum_{j=g_i}^{i-1} a_{p_i j} y_j$
  end for
  {**Pętla 2**}
  for $i = n$ **to** 1 **step** -1 **do**
    $x_i \leftarrow (y_i - \sum_{j=(i+1)}^{c_i} a_{p_i j} x_j)/a_{p_i i}$
  end for

Obliczanie wektora b

Wektor **b** obliczmy poprzez równanie $\mathbf{b} = \mathbf{Ax}$, gdzie $x = (1, \ldots, 1)^T$. Ze względu na elementy zerowe nie jest konieczna iteracja po wszystkich wartościach, dlatego dla danego wiersza zaczynamy od wcześniej opisanej wartości g_k, a kończymy na c_k, co pozwala na policzenie wektora **b** w czasie liniowym.

4 Wyniki

Test poprawności

W celu poprawności metod wykonano test, polegający na

1. utworzeniu macierzy za pomocą funkcji `blockmat`
2. wczytaniu macierzy z pliku
3. obliczeniu wektora **b**
4. rozwiązaniu układu równań utworzonymi metodami, dla różnych wielkości macierzy **A**
5. sprawdzeniu czy składowe wektora x są bliskie jeden dla danej delty

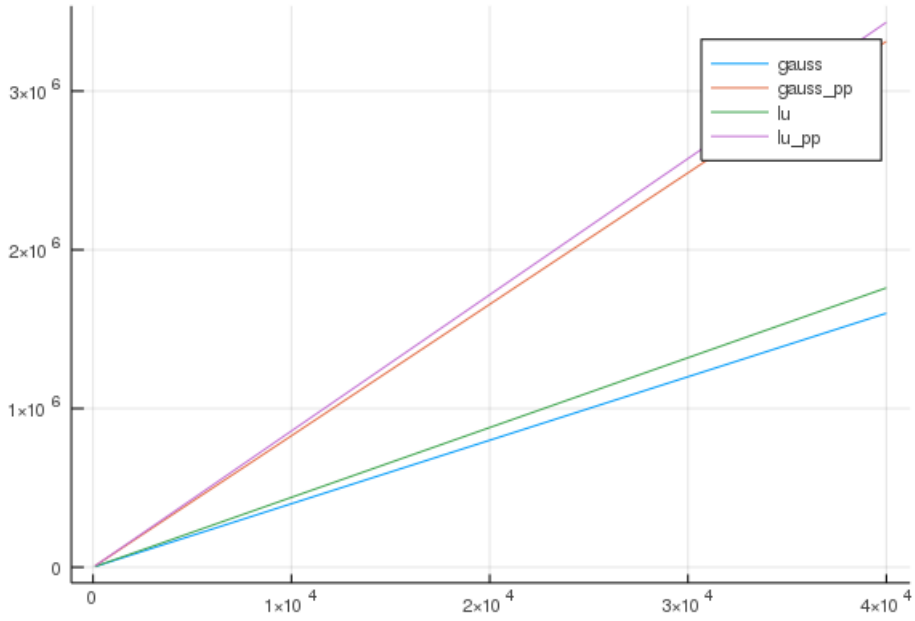
Test nie zwrócił błędu, zatem obliczone wartości są poprawne.

Test złożoności

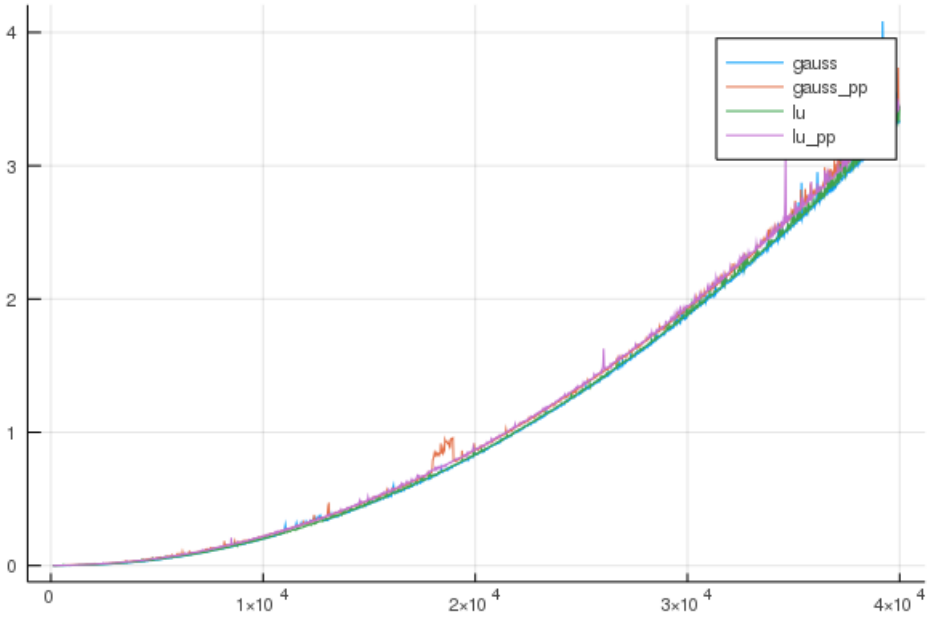
W celu sprawdzania szybkości metod wykonano test polegający na

1. utworzeniu macierzy za pomocą funkcji `blockmat`
2. wczytaniu macierzy z pliku
3. obliczeniu wektora **b**
4. rozwiązaniu układu równań utworzonymi metodami, dla rosnących wielkości macierzy **A**, mierząc czas rozwiązywania i zaalokowaną pamięć za pomocą makra `@timed` oraz liczbę operacji
5. umieszczeniu zmierzonych wartości na wykresie

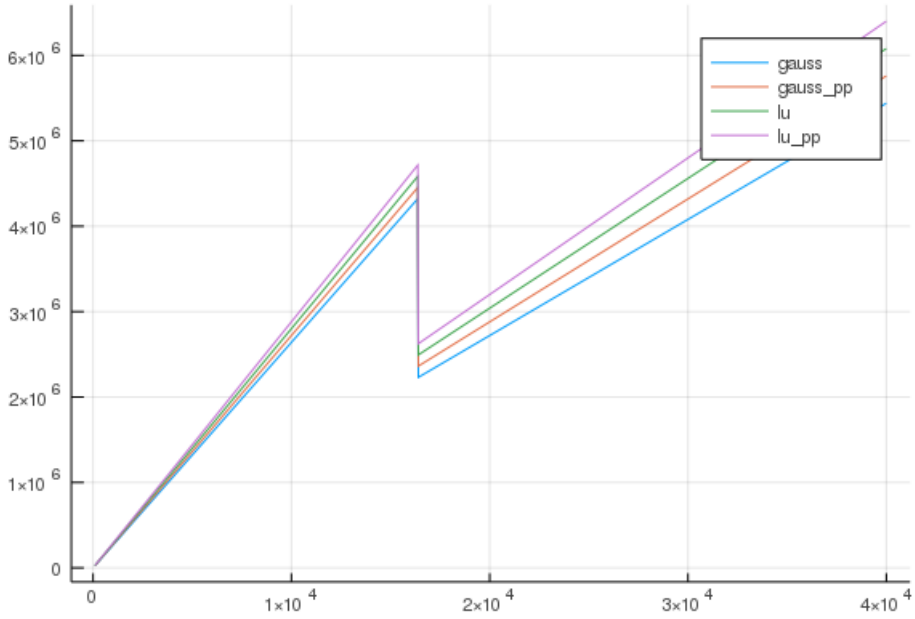
Wyniki widoczne są na następujących wykresach:



Rysunek 1: Wykres liczby operacji w zależności od n



Rysunek 2: Wykres czasu rozwiązywania układu w zależności od n



Rysunek 3: Wykres zaalokowanej pamięci w zależności od n

Obserwacje i wnioski

Na wykresie liczby operacji można zaobserwować liniową złożoność wszystkich algorytmów, co potwierdza wymaganą złożoność algorytmu $O(n)$. Zgodnie z oczekiwaniami, metody z wyborem elementu głównego wymagały większej liczby operacji. Również metody z wcześniej obliczonym rozkładem **LU** wymagały większej liczby operacji niż eliminacja bez rozkładu.

Patrząc na wykres czasu od n możemy zauważyć, że wzrasta on kwadratowo względem n. Wynika to z czasu dostępu do elementu w macierzy, gdyby czas ten był stały, czas wzrastałby liniowo względem n.

Patrząc na wykres zaalokowanej pamięci dla poszczególnych wykresów, możemy zauważyć, ze metody z wyborem elementu głównego zajmują więcej pamięci, ze względu na konieczność pamiętania tablicy p. Metody z rozkładem **LU** również wymagają większej ilości pamięci, ze względu na konieczność pamiętania przejściowego wektora y.

Obserwując wynikowe wektory x, które podczas testów były zapisane do plików, można było dostrzec, że składowe dla algorytmów z wyborem elementów głównych były nieznacznie bliższe wartości 1, co wskazuje na większą dokładność algorytmu. Zatem jeśli dokładność algorytmu jest dla nas ważniejsza niż szybkość, należy użyć algorytmów z wyborem elementów głównych. Algorytm ten również należy wybrać jeśli na diagonali znajdują się zera.

Głównym wnioskiem płynącym z eksperymentu jest fakt, że znając specyfikację danych wejściowych często możemy w łatwy sposób zmodyfikować algorytm, znacznie zwiększając szybkość jego działania.