Sprawozdanie

Nie interesuj się

15 kwietnia 2020

Opis Problemu

Zadanie dotyczyło rozwiązania układu równań liniowych następującej postaci:

$$\boldsymbol{Ax} = \boldsymbol{b}$$

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 strukturze:

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

gdzie $v = \frac{n}{\ell}$ zakładając że $\ell | n$, gdzie $\ell \geq 2$ jest rozmiarem każdej z kwadratowych macierzy wewnętrznych (bloków):

- $\boldsymbol{A}_k \in \mathbb{R}^{\ell \times \ell}$, $k = 1, ..., v$ są macierzami gęstymi
- $\boldsymbol{B}_k \in \mathbb{R}^{\ell \times \ell}$, $k = 2, ..., v$ są postaci:

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

- $\boldsymbol{C}_k \in \mathbb{R}^{\ell \times \ell}$, $k = 1, ..., v-1$ są diagonalne:

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

- $\mathbf{0} \in \mathbb{R}^{\ell \times \ell}$ to macierz zerowa

W celu rowiązania układów równań liniowych $\boldsymbol{Ax} = \boldsymbol{b}$ użyte zostaną cztery metody:

- Eliminacji Gaussa bez wyboru elementu głównego
- Eliminacji Gaussa z wyborem elementu głównego
- Obliczyć $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ bez wyboru elementu głównego, a następnie obliczyć $\boldsymbol{LUx} = \boldsymbol{b}$
- Obliczyć $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ z wyborem elementu głównego, a następnie obliczyć $\boldsymbol{LUx} = \boldsymbol{b}$

Efektywne przechowywanie macierzy rzadkiej

W konstruowaniu rozwiązania niezwykle istotny był odpowiedni sposób przechowywania macierzy w pamięci, ponieważ naiwne przechowywanie przykładowej macierzy o krawędzi $n = 50000$ będzie niemożliwe ze względu na ograniczoną pamięć.

Zważywszy na to że $\boldsymbol{A}$ jest macierzą rzadką - posiada tylko $n\ell + 3(n - \ell)$ elementów niezerowych:

- ℓ^2 w każdym z v bloków $\boldsymbol{A}_k$
- 2ℓ w każdym z $v - 1$ bloków $\boldsymbol{B}_k$
- ℓ w każdym z $v - 1$ bloków $\boldsymbol{C}_k$

Istnieje sposób na efektywne przechowywanie tylko niezerowych elementów macierzy. Jest to możliwe przy pomocy dostępnej w języku *julia* struktury ***SparseMatrixCSC***. Przechowuje ona poszczególne wartości macierzy skompresowane (tylko niezerowe wartości są przechowywane) w porządku kolumnowym. W związku z tym, że algorytm eliminacji Gaussa ma przebieg wierszowy, zdecydowałem się także na transpozycję macierzy (dzięki transpozycji czas dostępu do elementów jest krótszy).

Metoda eliminacji Gaussa - Teoria

W sposobie rozwiązywania równań liniowych tą metodą, wyróżnić można dwa etapy:

Pierwszy etap polega na doprowadzeniu układu do układu równoważnego z macierzą trójkątną górną. Zasada działania tej części algorytmu sprowadza się do zerowania kolejnych elementów znajdujących się pod diagonalą. Przykładowo w celu wyzerowania elementu a_{i1} od i-tego wiersza zostanie odjęty pierwszy wiersz pomnożony przez $\frac{a_{i1}}{a_{11}}$ (*mnożnik*). W analogiczny sposób wyzerowane zostają wszystkie elementy poniżej pierwszego wiersza w pierwszej kolumnie, następnie poniżej drugiego wiersza w drugiej kolumnie itd. Warty zauważenia jest fakt, że do poprawnego działania, algorytm ten wymaga aby żaden z elementów na diagonali nie był zerem. Aby algorytm mógł działać poprawnie również w takiej sytuacji, konieczna jest możliwość zamiany miejscami wierszy.

Drugi etap to zastosowanie algorytmu *podstawiania wstecz*, ktory opiera się na obliczeniu

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

dla kolejnych i-tych wierszy, rozpocznając od ostatniego z nich.

Algorytm eliminacji gaussa ma złożoność $O(n^3)$, a algorytm postawiania wstecz - $O(n^2)$. Złożoność całego rozwiązania wynosi więc $O(n^3)$.

Metoda eliminacji Gaussa - Implementacja

Opis

W sposobie rozwiązania uwzględniona została specyficzna, trójdiagonalno-blokowa postać macierzy $\boldsymbol{A}$. Oto graficzne przedstawienie przykładowej macierzy $\boldsymbol{A}$ o dowolnym n podzielnym przez 4, oraz $\ell = 4$:

$$\begin{pmatrix}
a^1_{1\,1} & a^1_{1\,2} & a^1_{1\,3} & a^1_{1\,4} & c^1_{1\,1} & 0 & 0 & 0 & \cdots \\
a^1_{2\,1} & a^1_{2\,2} & a^1_{2\,3} & a^1_{2\,4} & 0 & c^1_{2\,2} & 0 & 0 & \cdots \\
a^1_{3\,1} & a^1_{3\,2} & a^1_{3\,3} & a^1_{3\,4} & 0 & 0 & c^1_{3\,3} & 0 & \cdots \\
a^1_{4\,1} & a^1_{4\,2} & a^1_{4\,3} & a^1_{4\,4} & 0 & 0 & 0 & c^1_{3\,3} & \cdots \\
0 & 0 & b^2_{1\,1} & b^2_{1\,2} & a^2_{1\,1} & a^2_{1\,2} & a^2_{1\,3} & a^2_{1\,4} & \cdots \\
0 & 0 & b^2_{2\,1} & b^2_{2\,2} & a^2_{2\,1} & a^2_{2\,2} & a^2_{2\,3} & a^2_{2\,4} & \cdots \\
0 & 0 & b^2_{3\,1} & b^2_{3\,2} & a^2_{3\,1} & a^2_{3\,2} & a^2_{3\,3} & a^2_{3\,4} & \cdots \\
0 & 0 & b^2_{4\,1} & b^2_{4\,2} & a^2_{4\,1} & a^2_{4\,2} & a^2_{4\,3} & a^2_{4\,4} & \cdots \\
0 & 0 & 0 & 0 & 0 & 0 & b^3_{1\,1} & b^3_{1\,1} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}$$

Jak można zauważyć, nie jest konieczne zerowanie wszystkich elementów znajdujących się pod diagonalą. Dla pierwszych $\ell - 2$ kolumn niezerowe elementy mogą znajdować się jedynie w pierwszych ℓ rzędach ($\boldsymbol{A}_1$), dla $2\ell - 2$ kolumn w pierwszych 2ℓ rzędach ($\boldsymbol{B}_2$ oraz $\boldsymbol{A}_2$) itd. Schemat jest powtarzalny, daje więc możliwość na wyprowadzenie ogólnego wzoru na indeks ostatniego niezerowego elementu w danej kolumnie *column*:

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

Poza ostatnimi ℓ wierszami, w każdym wierszu ostatni element należy zawsze do diagonali macierzy blokowej $\boldsymbol{C}_i$. Te elementy znajdują się zawsze w odległości ℓ od właściwej diagonali $\boldsymbol{A}$. W ostatnich ℓ rzędach to $\boldsymbol{A}_v$ jest leżącym najbardziej z prawej strony blokiem, a więc ostatnie niezerowe elementy leżą na kolumnie n-tej. Stąd wzór na ostatni niezerowy element w rzędzie *row*:

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

Dodatkowym faktem wartym zauważenia jest brak jakichkolwiek nowych elementów niezerowych nad diagonalą bloków $\boldsymbol{C}_i$, co pozwala nam zoptymalizować algorytm *podstawiania wstecz* tak by sumować tylko elementy do kolumny określonej powyższym wzorem na ostatnią kolumnę ($last(row)$).

Analiza złożoności

Przy założeniu że ℓ jest stałą, złożoność obliczeniowa zmodyfikowanej metody wynosi $O(n)$ - liczba przebiegów wszystkich pętli to $(n-1) * 2\ell * \ell + n * \ell$.

Algorytm

Data:
A - macierz
b - wektor prawych stron macierzy A
n - rozmiar macierzy A
ℓ - rozmiar bloku macierzy wewnetrznych w A
Result:
x - wektor długości n zawierający rozwiązanie $Ax = b$
it - licznik iteracji

$it \leftarrow 0$
for $k \leftarrow 1$ ***to*** $n - 1$ **do**
  for $i \leftarrow k + 1$ ***to*** $\min\left\{n,\ \ell + \ell \cdot \left\lfloor \frac{k+1}{\ell} \right\rfloor\right\}$ **do**
    if $|A[k,k] = 0$ **then**
      return Error - Na diagonali znajduje się zero.
    end
    $mult \leftarrow \frac{A[k,i]}{A[k,k]}$
    $A[k,i] \leftarrow 0$
    for $j \leftarrow k + 1$ ***to*** $\min\{n,\ k + \ell\}$ **do**
      $A[j,i] \leftarrow A[j,i] - mult * A[j,k]$
      $it \leftarrow it + 1$
    end
    $b[i] \leftarrow b[i] - mult * b[k]$
  end
end
for $i \leftarrow n$ ***downto*** 1 **do**
  for $j \leftarrow k + 1$ ***to*** $\min\{n,\ i + \ell\}$ **do**
    $sum \leftarrow sum + x[i] * A[j,i]$
  end
  $x[i] \leftarrow \frac{b[i] - sum}{A[i,i]}$
end
return x, it

Wariant eliminacji Gaussa z częściowym wyborem elementu głównego

Opis

W sytuacji gdy nie możemy zastosować poprzedniego algorytmu (zero na diagonali $\boldsymbol{A}$), musimy zastosować jego inny wariant, który pozwala na rozwiązywanie takich układów.

Zasada działania jest prosta - wybrany zostaje wiersz, dla którego element w eliminowanej kolumnie k ma największą bezwzględną wartość i zostaje on zamieniony z k-tym wierszem. Dalsza eliminacja przebiega bez zmian.

W związku z dużym kosztem takiego rozwiązania, stosuję dodatkowy wektor p służący do przechowywania informacji o pozycji danego wiersza w macierzy.

Algorytm

Data:
A - macierz
b - wektor prawych stron macierzy A
n - rozmiar macierzy A
ℓ - rozmiar bloku macierzy wewnetrznych w A
Result:
x - wektor długości n zawierający rozwiązanie $Ax = b$
it - licznik iteracji

$it \leftarrow 0$
$p \leftarrow \{i : i \in \{1, \ldots, n\}\}$
for $k \leftarrow 1$ ***to*** $n - 1$ **do**
  for $i \leftarrow k + 1$ ***to*** $\min\left\{n,\ \ell + \ell \cdot \left\lfloor \frac{k+1}{\ell} \right\rfloor\right\}$ **do**
    $max\_row \leftarrow k$
    $max = |A[k, p[k]]|$
    for $j \leftarrow i$ ***to*** $\min\left\{n,\ \ell + \ell \cdot \left\lfloor \frac{k+1}{\ell} \right\rfloor\right\}$ **do**
      if $|A[k, p[j]]| > max$ **then**
        $max\_row \leftarrow j$
        $max \leftarrow |A[k, p[j]]|$
        $it \leftarrow it + 1$
      end
    end
    if $|max| = 0$ **then**
      return Error - macierz osobliwa.
    end
    $p[k], p[max\_row] \leftarrow p[max\_row], p[k]$
    $mult \leftarrow \frac{A[k,p[i]]}{A[k,p[k]]}$
    $A[k, p[i]] \leftarrow 0$
    for $j \leftarrow k + 1$ ***to*** $\min\left\{n,\ 2\ell + \ell \cdot \left\lfloor \frac{k+1}{\ell} \right\rfloor\right\}$ **do**
      $A[j, p[i]] \leftarrow A[j, p[i]] - mult * A[j, p[k]]$
      $it \leftarrow it + 1$
    end
    $b[p[i]] \leftarrow b[p[i]] - mult * b[p[k]]$
  end
end
for $i \leftarrow n$ ***downto*** 1 **do**
  for $j \leftarrow k + 1$ ***to*** $\min\left\{n,\ 2\ell + \ell \cdot \left\lfloor \frac{p[i]+1}{\ell} \right\rfloor\right\}$ **do**
    $sum \leftarrow sum + x[j] * A[j, p[i]]$
  end
  $x[i] \leftarrow \frac{b[p[i]]-sum}{A[i,p[i]]}$
end
return x, it

Analiza złożoności

Podstawową różnicą między tą a poprzednią wersją algorytmu, jest szersze oszacowanie możliwego położenia ostatniego niezerowego elementu w rzędzie:

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

Analogicznie do poprzedniej metody, także tutaj możemy zauważyć że w trakcie eliminacji $\ell - 2$ pierwszych kolumn najdalszy niezerowy element można stworzyć w kolumnie 2ℓ itd.

Zmiany te zmieniają jednak tylko stałe, więc ogólna złożoność pozostaje $O(n)$

Rozkład LU

Teoria

Rozkład $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ to przedstawienie zadanej macierzy w postaci iloczynu $\boldsymbol{A} = \boldsymbol{LU}$, gdzie $\boldsymbol{L}$ jest macierzą trójkątną dolną, a $\boldsymbol{U}$ jest macierzą trójkątną górną. Dodatkowo, elementy na diagonali $\boldsymbol{L}$ mają wartość 1.

Do uzyskania rozkładu $\boldsymbol{LU}$ ponownie użyłem metody eliminacji Gaussa - macierz $\boldsymbol{A}$ zostaje przekształcona do macierzy górnotrójkątnej ($\boldsymbol{U}$), a macierz $\boldsymbol{L}$ jest uzyskana poprzed zapamiętywanie mnożników użytych do przekształceń (mnożnik $mult_{j,i}$ służący do wyzerowania elementu $a_{j,i}$ bedzie zapisany w $L[j,i]$). Zważywszy na duży rozmiar macierzy i cechy rozkładu $\boldsymbol{LU}$, zapisywany on będzie na jednej macierzy o rozmiarze oryginalnej macierzy $\boldsymbol{A}$.

Przeprowadzenie rozkładu $\boldsymbol{LU}$ ma złożoność $O(n^3)$, jednakże sposób ten umozliwia znacząco szybsze rozwiązywanie układów w których macierz pozostaje taka sama, a zmienia się jedynie wektor prawych stron. W takim przypadku rozwiązanie sprowadza się jedynie do rozwiązania układu:

$$\begin{cases} \boldsymbol{Lz} = \boldsymbol{b} \\ \boldsymbol{Ux} = \boldsymbol{z} \end{cases}$$

Co dzięki trójkątnej postaci macierzy $\boldsymbol{L}$ i $\boldsymbol{U}$ możemy wykonać w $O(n^2)$ (algorytm postawiania wstecz)

Opis algorytmów

Algorytm przeprowadzający rozkład $\boldsymbol{LU}$ jest analogiczny do już przeanalizowanych algorytmów - eliminacji gaussa oraz eliminacji gaussa z częściowym wyborem elementu głównego. Jedyna różnica to:

- W algorytmie bez wyboru elementu głównego - zamiast zerowania elementów a_{ij} przypisywana jest im wartość mnożnika ($mult = \frac{A[j,i]}{A[j,k]}$)
- W algorytmie z wyborem elementu głównego - zamiast zerowania elementów a_{ij} przypisywana jest im wartość mnożnika ($mult = \frac{A[j,p[i]]}{A[j,p[j]]}$)

Analiza złożoności

Złożoność jest taka sama jak w przypadku zmodyfikowanej eliminacji gaussa $O(n)$ (przy założeniu że ℓ jest stałą), gdzie wersja z częściowym wyborem elementu głównego różni się o stałą.

Rozwiązanie układu równań z LU

Rozwiązanie układu:

$$\begin{cases} \boldsymbol{Lz} = \boldsymbol{b} \\ \boldsymbol{Ux} = \boldsymbol{z} \end{cases}$$

składa się z dwóch etapów:

1. rozwiązanie $\boldsymbol{Ux} = \boldsymbol{z}$ to omawiany już wcześniej algorytm podstawiania wstecz, indeks ostatniej niezerowej kolumny w wierszu przedstawia się wzorem:

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

2. rozwiązanie $\boldsymbol{Lz} = \boldsymbol{b}$ to algorytm podstawiania w przód, w przeciwieństwie do algorytmu podstawiania wstecz zaczyna on iterację od pierwszego elementu i sumuje coraz dalsze kolumny a nie coraz wcześniejsze. Indeks pierwszej kolumny można wyznaczyć wzorem:

$$firstColumn(row) = \max\left\{1,\ \ell \cdot \left\lfloor \frac{row-1}{\ell} \right\rfloor - 1\right\}$$

Analiza złożoności

W związku z tym że rozwiązanie to ma tyle samo przebiegów pętli jak część podstawiania wstecz w algorytmach eliminacji gaussa, także tu złożoność wynosi $O(n)$

Algorytm

Przestawiony algorytm dotyczy metody bez wyboru elementu głównego, w algorytmie z wyborem jedyną różnicą jest odwoływanie się do wektora permutacji p zamiast do konkretnego wiersza.

Data:
A - macierz
b - wektor prawych stron macierzy A
n - rozmiar macierzy A
ℓ - rozmiar bloku macierzy wewnetrznych w A
Result:
x - wektor długości n zawierający rozwiązanie $Ax = b$
it - licznik iteracji

$it \leftarrow 0$
for $i \leftarrow 1$ ***to*** n **do**
  $sum \leftarrow 0$
  for $j \leftarrow \max\left\{1,\ \ell \cdot \left\lfloor \frac{i-1}{\ell} \right\rfloor - 1\right\}$ ***to*** $i-1$ **do**
    $sum \leftarrow sum + z[j] * A[j,i]$
  end
  $z[i] \leftarrow b[i] - sum$
end
for $i \leftarrow n$ ***downto*** 1 **do**
  $sum \leftarrow 0$
  for $j \leftarrow k+1$ ***to*** $\min\{n,\ i+\ell\}$ **do**
    $sum \leftarrow sum + x[i] * A[j,i]$
  end
  $x[i] \leftarrow \frac{z[i]-sum}{A[i,i]}$
end
return x, it

Wyniki

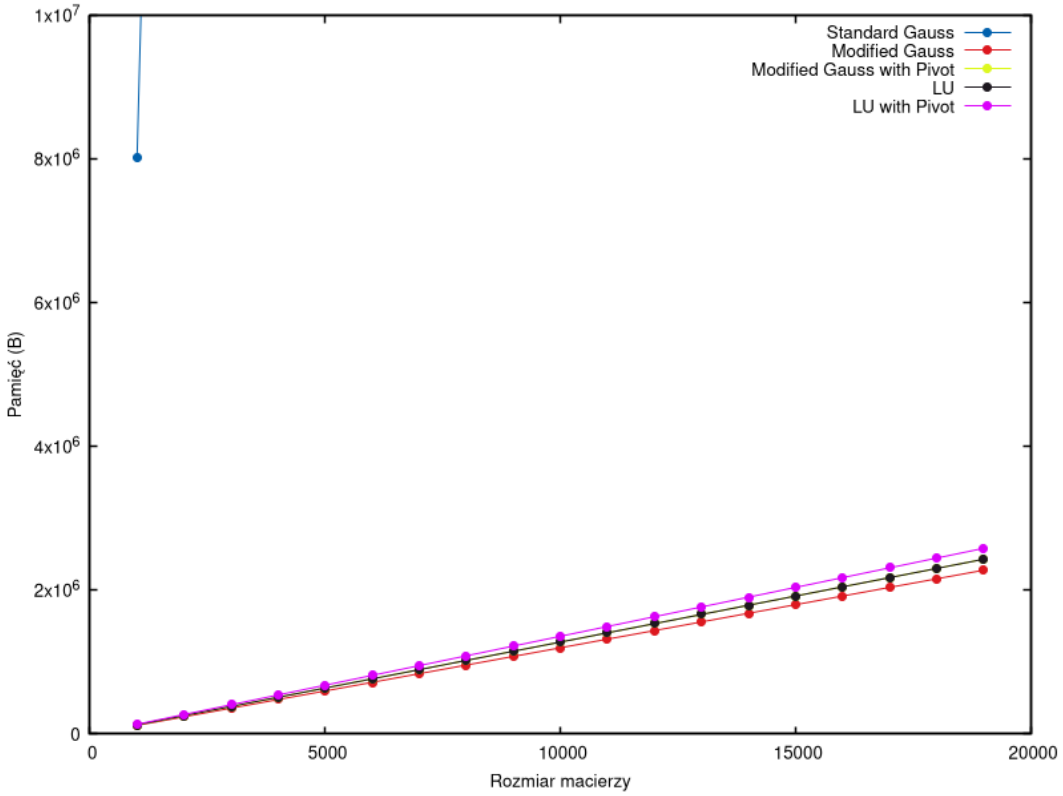
W celu potwierdzenia poprawności algorytmów wygenerowałem takie wektory prawych stron aby rozwiązaniem układu był zawsze $(1, ..., 1)^T$. Następnie testowałem algorytmy, mierząc ich średni błąd względny, czas obliczeń, wielkość alokowanej pamięci oraz ilość operacji (średnia z 10 prób). Próby przeprowadzone byłe dla macierzy o $n \in \{16, 1000, 5000, 10000, 15000, 20000, 50000\}$, $\ell = 4$ oraz $c_k = 2.0$

błędy względne:

n	gauss (julia)	gauss	gauss z wyborem	LU bez wyboru	LU z wyborem
16	$3.754704674e-16$	$5.453105385e-15$	$5.003707553e-16$	$4.574031282e-15$	$5.064917260e-16$
1000	$2.234004883e-16$	$7.552064752e-15$	$2.313956221e-16$	$1.182375532e-14$	$2.123111046e-16$
5000	$2.281285449e-16$	$1.296241376e-14$	$2.319702017e-16$	$1.475353849e-13$	$2.184968556e-16$
10000	$4.076968120e-16$	$3.673820219e-14$	$3.773321219e-16$	$3.630437761e-14$	$4.469595776e-16$
50000	$---$	$9.447293155e-14$	$5.427796231e-16$	$9.127843964e-14$	$5.322902898e-16$

pobór pamięci :

n	gauss (julia)	gauss	gauss z wyborem	LU bez wyboru	LU z wyborem
1000	7.641 MiB	117.156 KiB	125.094 KiB	125.125 KiB	133.062 KiB
5000	190.792 MiB	585.859 KiB	625.000 KiB	625.032 KiB	664.172 KiB
10000	763.054 MiB	1.144 MiB	1.221 MiB	1.210 MiB	1.301 MiB
15000	1.677 GiB	1.717 MiB	1.831 MiB	1.836 MiB	1.951 MiB
20000	2.690 GiB	2.174 MiB	2.319 MiB	2.321 MiB	2.482 MiB



Różnica dla rozkładów i rozwiązań LU :

Pamięć:

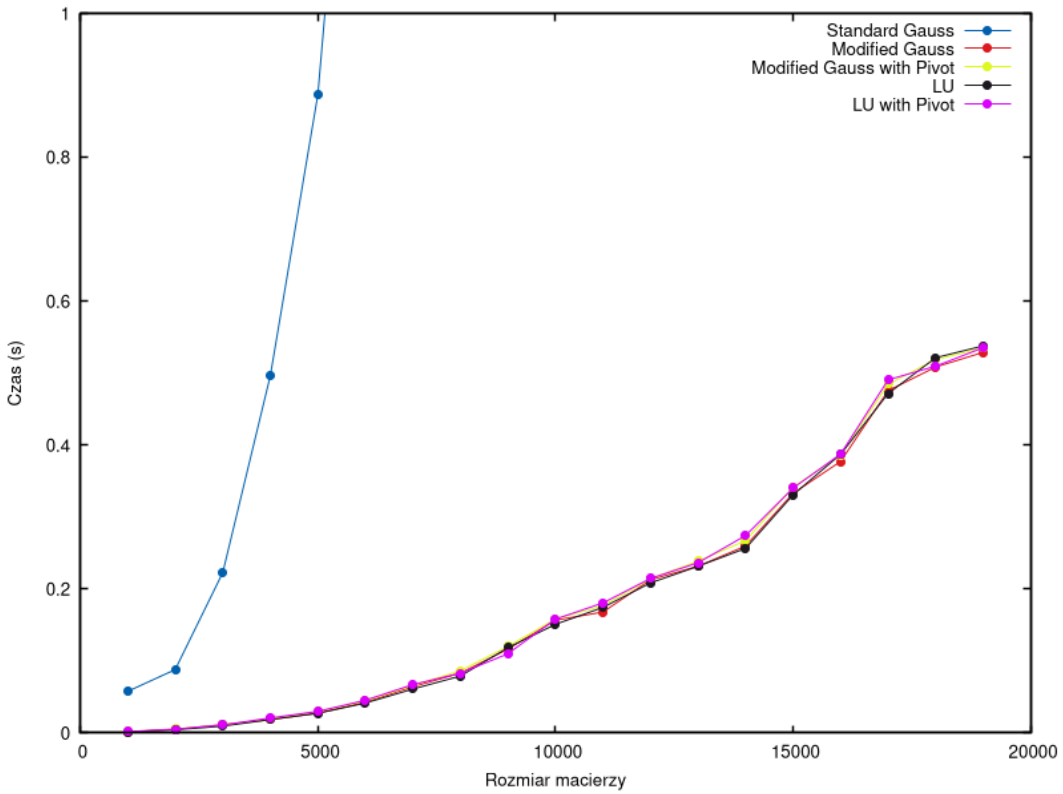
n	rozkład LU bez wyboru	rozwiązanie LU bez wyboru	rozkład LU z wyborem	rozwiązanie LU z wyborem
1000	109.219 KiB	15.906 KiB	117.156 KiB	15.906 KiB
5000	546.719 KiB	78.313 KiB	585.859 KiB	78.313 KiB
10000	1.068 MiB	156.438 KiB	1.144 MiB	156.438 KiB
15000	1.602 MiB	234.563 KiB	1.717 MiB	234.563 KiB
20000	2.029 MiB	297.063 KiB	2.174 MiB	297.063 KiB

Czas:

n	rozkład LU bez wyboru	rozwiązanie LU bez wyboru	rozkład LU z wyborem	rozwiązanie LU z wyborem
1000	0.001404	0.000059	0.001620	0.000092
5000	0.026826	0.000475	0.029455	0.000552
10000	0.150215	0.000794	0.157064	0.001153
15000	0.330964	0.000948	0.340329	0.001394
20000	0.527977	0.001210	0.535301	0.001869

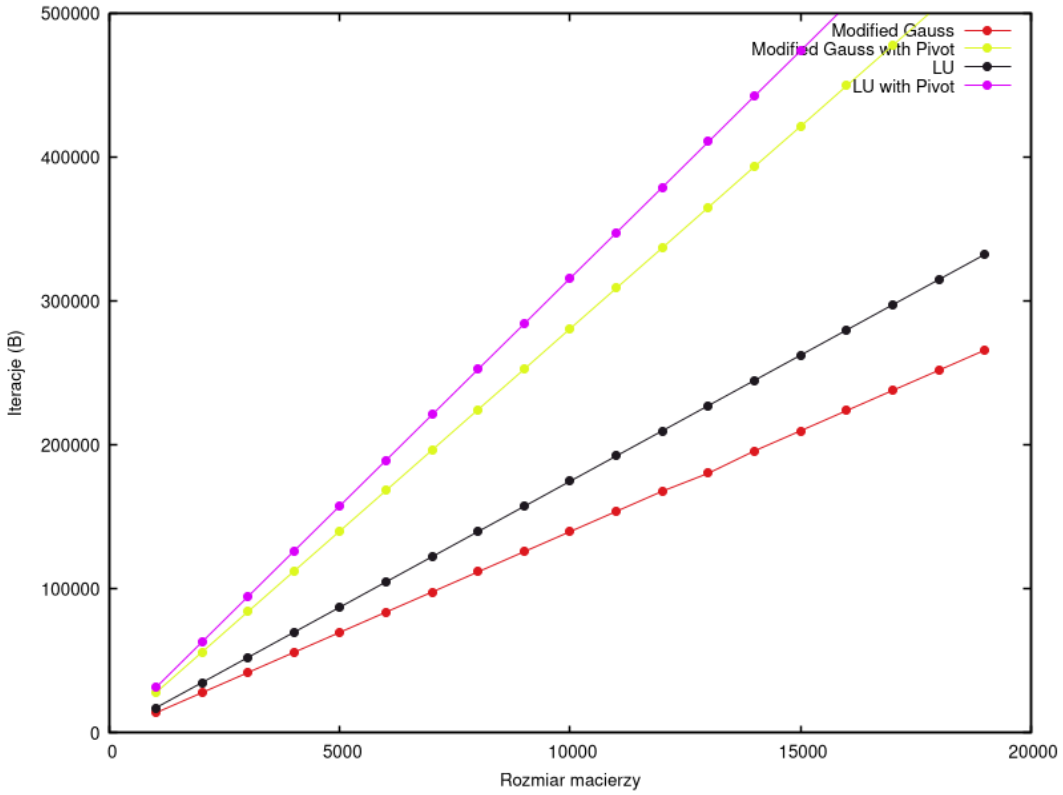
czas obliczeń :

n	gauss (julia)	gauss	gauss z wyborem	LU bez wyboru	LU z wyborem
1000	0.057868	0.001203	0.001788	0.001463	0.001712
5000	0.886813	0.027649	0.029836	0.027301	0.030007
10000	6.801013	0.156763	0.158280	0.151009	0.158218
15000	24.141143	0.333071	0.341681	0.330964	0.387559
20000	49.588023	0.528979	0.535050	0.537977	0.535301



ilość operacji :

n	gauss	gauss z wyborem	LU bez wyboru	LU z wyborem
1000	13953	27955	17450	31447
5000	69958	140388	87450	157880
10000	139941	280957	174950	315949
15000	209958	421406	262450	473898
20000	265963	533996	332450	600488



Obserwacje

- Dla algorytmów z częściowym wyborem elementu głównego błąd względny jest zawsze o przynajmniej rząd wielkości mniejszy niż w ich odpowiednikach bez wyboru.
- Złożoność pamięciowa zaimplementowanych metod jest liniowa.
- Algorytmy nie używające rozkładu ***LU*** wykorzystują mniej zasobów.
- Rozwiązanie układu przy już obliczonym rozkładzie ***LU*** jest wielokrotnie tańsze, zarówno czasowo jak i pamięciowo, niż sam rozkład.
- Algorytmy z częściowym wyborem elementu głównego wykonują więcej operacji od ich odpowiedników bez wyboru (mimo to wziąć mają złożoność $O(n)$).
- Złożoność czasowa wszystkich zaimplementowanych algorytmów (wszystkie poza *standard gauss*) jest kwadratowa (jest to skutkiem wykorzystania struktury *SparseArrayCSC*, w której dostęp do elementu nie jest tak naprawdę stały).

Wnioski

- Przedstawione wyniki zgadzają się z wstępną analizą złożoności algorytmów, według której jest ona liniowa.
- Dla osiągnięcia dokładniejszych wyników warto używać metod z częściowym wyborem elementu głównego.
- Mimo że dla pojedynczej macierzy algorytm z rozkładem $\boldsymbol{LU}$ osiąga gorsze wyniki, to jest on opłacalny gdy danej macierzy będziemy używać więcej niż raz (dla różnych wektorów prawych stron).
- Dzięki optymalizacji ogólnej postaci algorytmu do konkrentego zastosowania można osiągnąć dużo lepsze wyniki (w tym przypadku zredukować złożoność z $O(n^3)$ do $O(n)$).