Obliczenia naukowe lista 5

Stanisław Tomkowiak

4 stycznia 2024

Opis zadania

Problem przedstawiony na liście 5 sprowadza się do rozwiązania układu równań liniowych:

$$\texttt{Ax=b.}$$

dla danej macierzy współczynników $\texttt{A} \in R^{n \times n}$ i wektora prawych stron $\texttt{b} \in R^n$, $n \geqslant 4$. Macierz $\texttt{A}$ jest rzadką, tj. mającą dużą elementów zerowych, i blokową o następującej strukturze:

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

$v = n/l$ zakładając, że n jest podzielne przez l, gdzie $l \geqslant 2$ jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych (bloków): $\texttt{A}_i \in R^{l \times l}$, $\texttt{B}_i \in R^{l \times l}$, $\texttt{C}_i \in R^{l \times l}$, $i = 1, ..., v$. $\texttt{A}_i$ jest macierzą gęstą, $\texttt{B}_i, i \neq 1$ jest macierzą mającą elementy niezerowe tylko w dwóch ostatnich kolumnach, $\texttt{C}_i, i \neq v$ jest macierzą mającą elementy niezerowe tylko na przekątnej. Zakładamy, że $l = O(1)$. Dla bardzo dużych n napotykamy problem dotyczący wymagań pamięciowych i czasowych. Zadanie polega na dostosowaniu użytych struktur oraz algorytmów do szczególnej postaci macierzy, aby z standardowego czasu $O(n^3)$ zredukować go do $O(n)$. Rozwiązanie układu równań liniowych:

$$\texttt{Ax=b.}$$

mamy otrzymać i zbadać za pomocą czterech algorytmów:

- Standardowego rozwiązania bez wyboru elementu głównego układu równań za pomocą eliminacji Gaussa.
- Standardowego rozwiązania z częściowym wyborem elementu głównego układu równań za pomocą eliminacji Gaussa.
- Rozwiązania bez wyboru elementu głównego układu równań z wyznaczonym rozkładem `LU`.
- Rozwiązania z częściowym wyborem elementu głównego układu równań z wyznaczonym rozkładem `LU`.

Rozwiązanie

Struktura danych

Klasyczny sposób przechowywania macierzy w postaci tablicy dwuwymiarowej jest tutaj nieskuteczny. Wynika to z tego, że w założeniach mamy, że n jest duże. W takim przypadku samo zapisanie

tablicy zajmowałoby $O(n^2)$ miejsca. Z treści wiemy, że macierz jest rzadka co sprawia, że zapamiętalibyśmy bardzo dużo wartości równych 0 co nie jest nam potrzebne. Musimy zatem znaleźć inną strukturę. Rozwiązaniem tego jest zapamiętywanie jedynie komórek niezerowych.

Język Julia udostępnia strukturę idealną do tego zadania. Znajduje się ona w pakiecie `SparseArrays`. Struktura nosi nazwę `SparseMatrixCSC`. Przechowuje ona trzy tablice. Pierwsza z numerami kolumn, druga z numerami wierszy i trzecia z wartościami. Czytając każdą z nich na tym samym indeksie dostaniemy numer wiersza, kolumny i wartość. Dzięki temu iterowanie po wierszach i kolumnach w żadnym stopniu nie uwzględnia iterowania po wartościach zerowych.

Eliminacja Gaussa

Klasyczny algorytm i podstawy teoretyczne

W sytuacji ogólnej dla gęstych macierzy układ równań ma postać:

$$a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + ... + a_{1n}^{(1)}x_n = b_1^{(1)}$$

$$a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + ... + a_{2n}^{(1)}x_n = b_2^{(1)}$$

$$...$$

$$a_{n1}^{(1)}x_1 + a_{n2}^{(1)}x_2 + ... + a_{nn}^{(1)}x_n = b_n^{(1)}$$

W pierwszym kroku algorytmu eliminujemy zmienną x_1 z równań od 2-go do n-tego oraz mnożymy 1-sze równanie przez

$$I_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, i = 2, ..., n$$

i odejmujemy od pozostałych.

Postępując dalej w ten sposób po 1-szym kroku układ wygląda w następujący sposób:

$$a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + ... + a_{1n}^{(1)}x_n = b_1^{(1)}$$

$$a_{22}^{(2)}x_2 + ... + a_{2n}^{(2)}x_n = b_2^{(2)}$$

$$...$$

$$a_{n2}^{(2)}x_2 + ... + a_{nn}^{(2)}x_n = b_n^{(2)}$$

Po $k-1$ krokach otrzymujemy:

$$a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + ... + a_{1n}^{(1)}x_n = b_1^{(1)}$$

$$a_{22}^{(2)}x_2 + ... + a_{2n}^{(2)}x_n = b_2^{(2)}$$

$$...$$

$$a_{kk}^{(k)}x_k + ... + a_{kn}^{(k)}x_n = b_k^{(k)}$$

$$...$$

$$a_{nk}^{(k)}x_k + ... + a_{nn}^{(k)}x_n = b_n^{(k)}$$

Eliminujemy zmienną x_k z równań od $k+1$-szego do n-tego oraz mnożymy k-te równanie przez

$$I_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, i = k+1, ..., n$$

i odejmujemy od pozostałych.

Po $n-1$ krokach otrzymujemy układ z macierzą trójkątną górną:

$$a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + ... + a_{1n}^{(1)}x_n = b_1^{(1)}$$

$$a_{22}^{(2)}x_2 + ... + a_{2n}^{(2)}x_n = b_2^{(2)}$$

$$...$$

$$a_{nn}^{(n)}x_n = b_n^{(n)}$$

Na tej podstawie zakładając, że elementy na przekątnej są niezerowe, możemy wyznaczyć rozwiązanie układu równań w następujący sposób:

$$x_n = \frac{b_n}{a_{nn}^{(n)}}$$

Dalej wyznaczamy x_k dla $k = n-1, ..., 1$

$$x_k = \frac{b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j}{a_{kk}^{(k)}}$$

Taki algorytm ma złożoność obliczeniową $O(n^3)$ oraz złożoność pamieciową $O(n^2)$.

Wybór częściowy elementu głównego

Algorytm w tej zależności wygląda bardzo podobnie. Jedyną zmiana polega na znalezieniu elementu takiego, że

$$|a_{pk}^{(k)}| = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}|$$

i przestawieniu w macierzy $A^{(k)}$ wiersza p-tego z k-tym oraz elementu p-tego z k-tym w wektorze prawych stron $b^{(k)}$. Dzięki temu zabiegowi zmniejszamy błąd względny.

Optymalizacja algorytmu

Konkretna struktura macierzy `A` pozwoli nam na znaczną optymalizację algorytmów. Mianowicie założeniem eliminacji Gaussa jest zerowanie elementów pod diagonalą. Większość tych elementów jeszcze przed użyciem algorytmu jest równa 0. Należy zatem zrewidować zakresy na których powinniśmy działać.

Zaczynając od kolumn maksymalnym wychyleniem pod diagonalę będzie rozmiar bloku l. Najbardziej wewnętrzna pętla powinna natomiast wykonywać się do czasu ostatniej nie zerowej wartości w wierszu. Sprawia to, że złożoność zmienia się z $O(n^3)$ na $O(n * l * l)$ a przy założeniu $O(l) = 1$ daje nam złożoność na poziomie $O(n)$.

Rozkład `LU`

Klasyczny algorytm i podstawy teoretyczne

Każdy krok pierwszego etapu eliminacji Gaussa można przestawić poprzez przemnożenie macierzy

$$A^{(k+1)} = L^{(k)}A^{(k)}, b^{(k+1)} = L^{(k)}b^{(k)},$$

gdzie

$$L^{(k)} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -I_{k+1,k} & 1 & & \\ & & -I_{k+2,k} & & & \\ & & \vdots & & \ddots & \\ & & -I_{k+n,k} & & & 1 \end{pmatrix}$$

Zatem proces sprowadzenia macierzy $A^{(1)} = A$ do macierzy trójkątnej górnej $A^{(n)} = U$ można zapisać jako

$$U = L^{(n-1)}...L^{(2)}L^{(1)}A$$

$$A = (L^{(n-1)}...L^{(2)}L^{(1)})^{-1}U$$
$$A = L^{(1)-1}L^{(2)-1}...L^{(n-1)-1}U$$

Oznaczamy $L = L^{(1)-1}L^{(2)-1}...L^{(n-1)-1}$.

$$L = \begin{pmatrix} 1 & & & & \\ I_{21} & 1 & & & \\ I_{31} & I_{32} & 1 & & \\ \vdots & \vdots & & \ddots & \\ I_{n1} & I_{n2} & ... & I_{n,n-1} & 1 \end{pmatrix}$$

W ten sposób pierwszy etap eliminacji Gaussa możemy równoważnie zapisać jako $A = LU$.

W realizacji algorytmu pamiętamy wszystko w jednej macierzy, gdzie elementy macieży U są pamiętane jak wcześniej w górnym trójkącie i na przekątnej, a elementy macierzy L w dolnym trójkącie, pamiętamy dodatkowo o tym, że ma ona jedynki na przekątnej, których nie trzeba zapisywać w pamięci.

Rozkład LU wygląda zatem tak samo jak pierwszy etap eliminacji Gaussa, tylko zamiast zmieniać odpowiednie elementy w macierzy b, zapisujemy w dolnym trójkącie elementy macierzy L. Indeksy, po których się poruszamy pozostają takie same, czyli złożoność obliczeniowa i pamięciowa pozostaje $O(n)$.

Znając rozkład `A` = `LU` zadanie `A`x = `b` sprowadzamy do rozwiązania dwóch układów trójkątnych:

$$\texttt{L}y = \texttt{b}$$
$$\texttt{U}x = y$$

Wybór częściowy elementu głównego

Algorytm w tej zależności wygląda bardzo podobnie. Jedyną zmiana polega na znalezieniu elementu takiego, że

$$|a_{pk}^{(k)}| = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}|$$

i przestawieniu w macierzy $A^{(k)}$ wiersza p-tego z k-tym oraz zapisaniu w dodatkowej tablicy jakie wiersze zostały zamienione aby w kolejnym kroku odpowiednio zmodyfikować wektor `b`. Dzięki temu zabiegowi zmniejszamy błąd względny.

Optymalizacja algorytmu

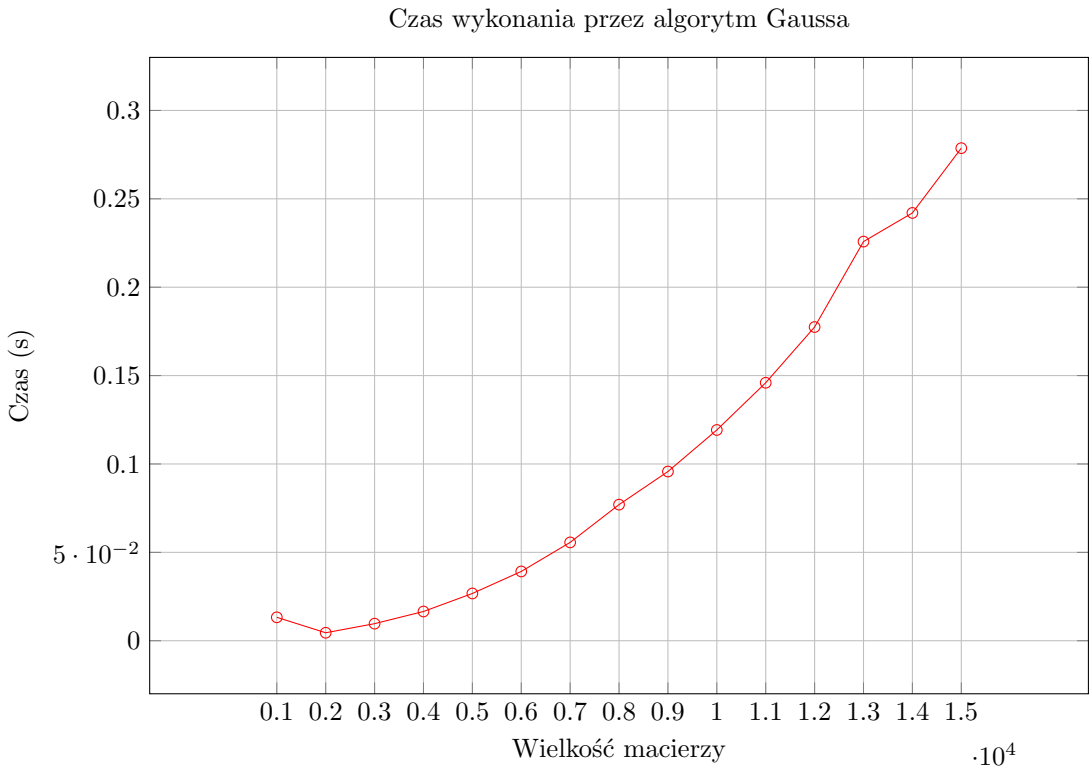
Podobnie jak w prypadku eliminacji Gaussa należy zrewidować zakresy na których powinniśmy działać. Zakresy te są takie same. Sprawia to, że złożoność zmienia się z $O(n^3)$ na $O(n * l * l)$ a przy założeniu $O(l) = 1$ daje nam złożoność na poziomie $O(n)$.

Metodologia badania

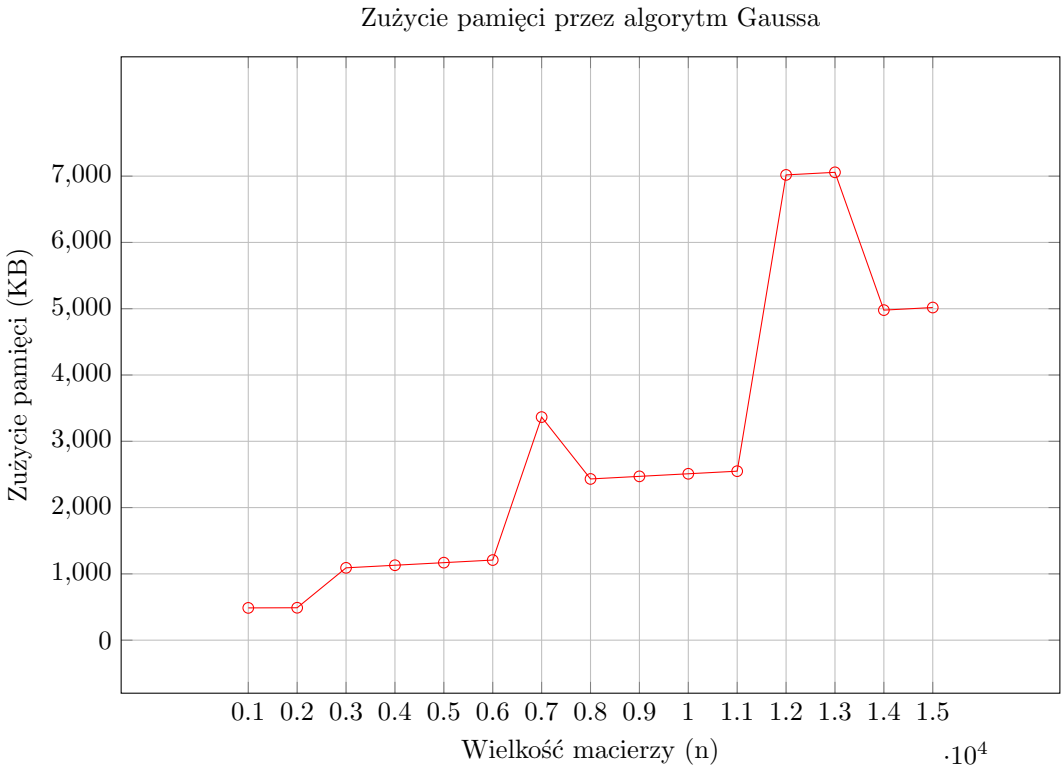
Aby łatwo porównać wyniki użyta została funkcja `blockmat` z modułu `matrixgen`. Za jej pomocą wygenerowane zostały macierze zgodne ze specyfikacją zadania o wielkości od $n = 1000$ do $n = 15000$ z krokiem równym 1000. Pozostałe parametry to $l = 4$ i wskaźnikiem uwarunkowania równym $ck = 10$. Następnie dla wygenerowanych macierzy uruchomiono testy, które liczyły czas wykonania każdego algorytmu, zaalokowaną pamięć oraz błąd względny. Każdy test został wykonany 3 razy dla każdej funkcji aby wyeliminować błędy. Wektor `b` był wyliczany za pomocą funkcji `getB`.

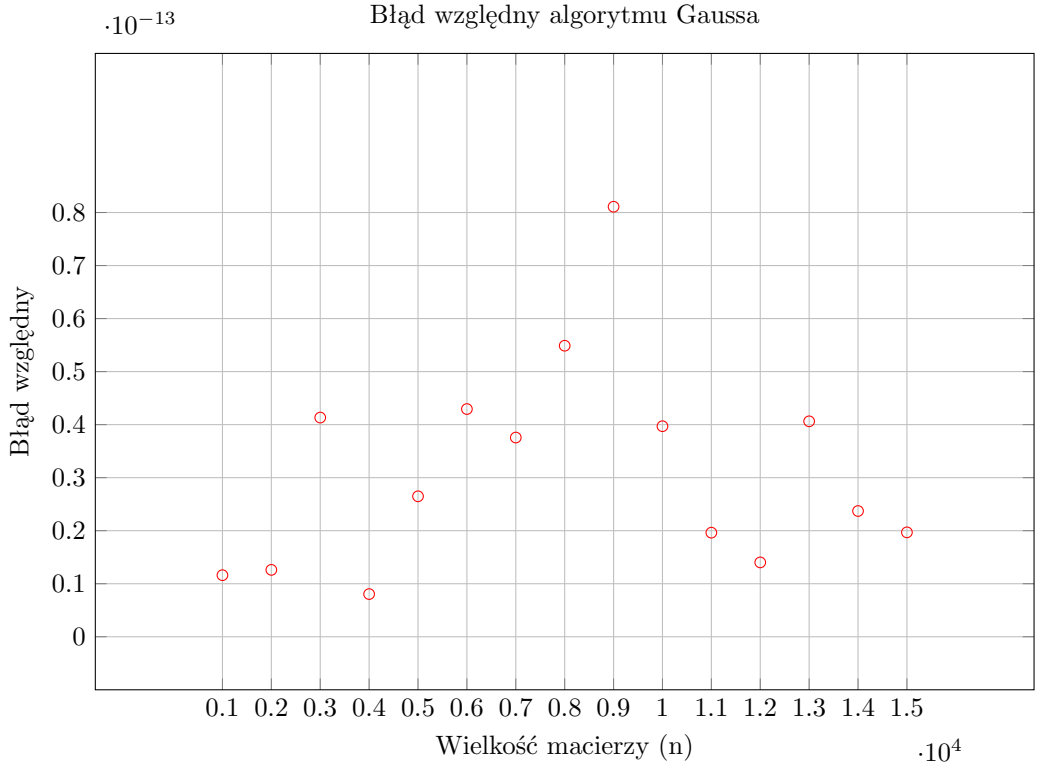
Wyniki oraz interpretacja

Eliminacja Gaussa bez częściowego wyboru elementu głównego



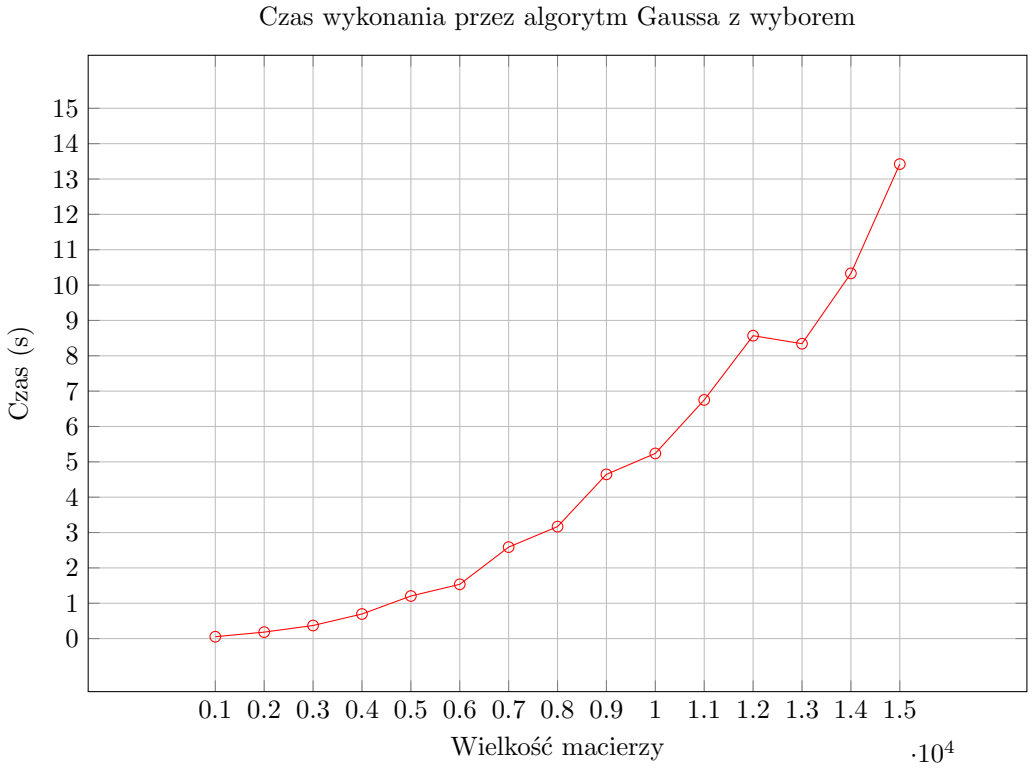
Czas wykonania odbiega od założonego O(n) dzieje się tak ponieważ w rzeczywistości $O(l) \neq 1$. Czasy natomiast w porównaniu do niezoptymalizowanego algorytmu w dalszej mierze są bardzo dobre.



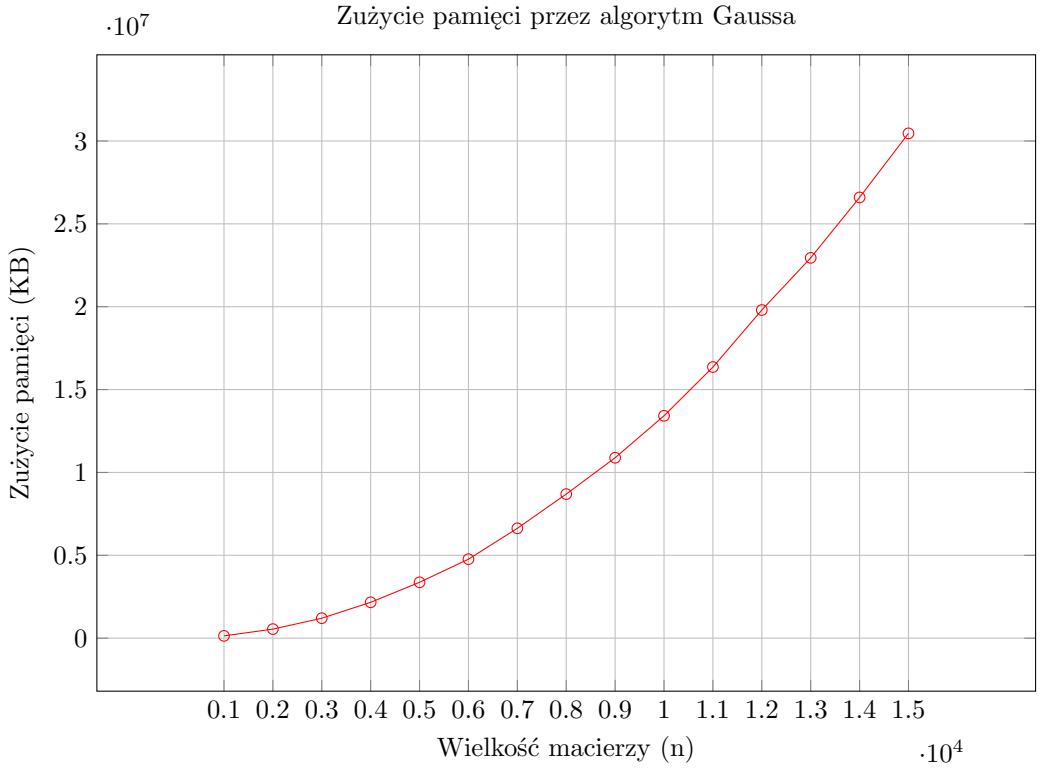


Dla większości macierzy błąd względny był rzędu 10^{-14}. Jest to akceptowalny wynik błędu. Można go jednak poprawić częściowym wyborem.

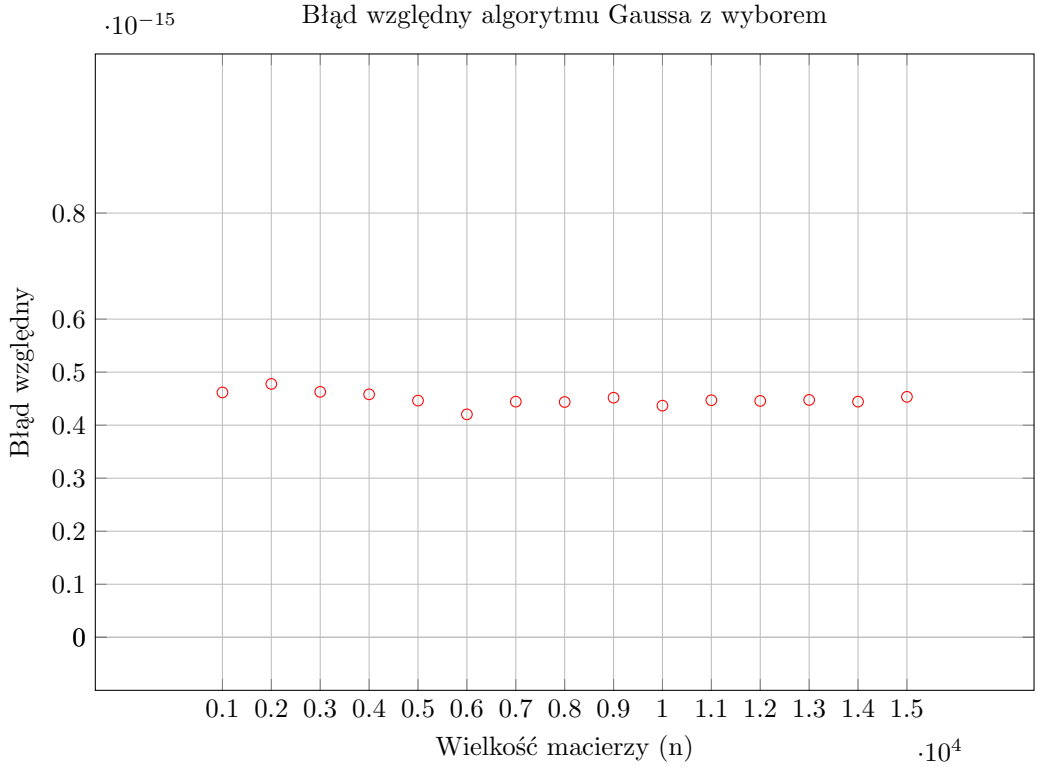
Eliminacja Gaussa z częściowym wyborem elementu głównego



Czas wykonania odbiega od założonego O(n) dzieje się tak, ponieważ w rzeczywistości $O(l) \neq 1$. Czasy natomiast w porównaniu do niezoptymalizowanego algorytmu w dalszej mierze są bardzo dobre. Dodatkowo możemy zauważyć, że czasy są większe niż czasy eliminacji Gaussa bez wyboru. Zjawisko to zachodzi, ponieważ musimy dodatkowo odpowiednio permutować macierz oraz wektor `b`.



Jest alokowane znacznie więcej pamięci niż w eliminacji bez wyboru. Znowu jest to kwestia permutowania macierzy.

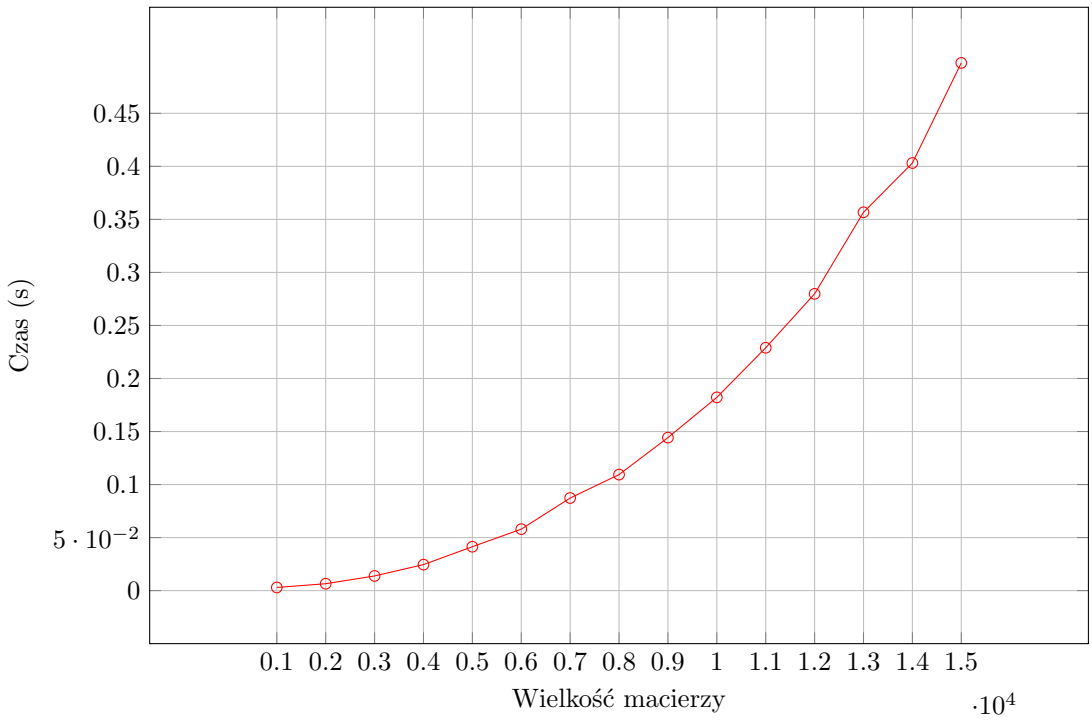


Błąd względny jednak jest dużo mniejszy niż w eliminacji bez wyboru. Wpływ na to ma wybieranie elementu głównego. Wyeliminowanie elementów bliskich zeru z przekątnej macierzy znacznie

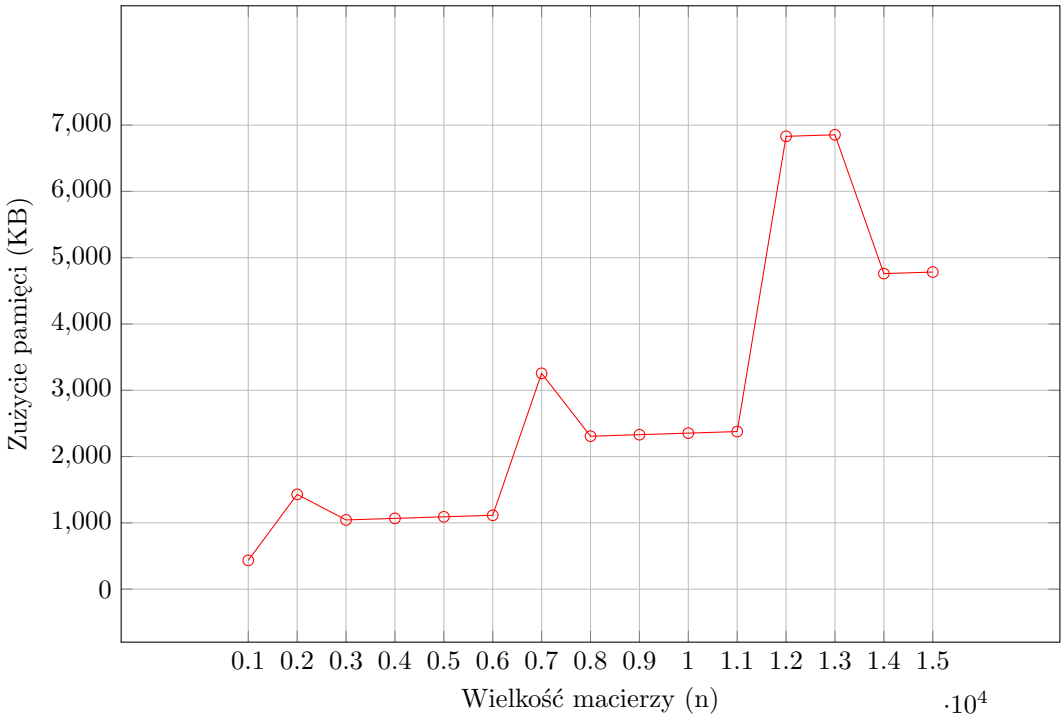
poprawiło błąd względny, który w każdym przypadku jest teraz rzędu epsilona maszynowego.

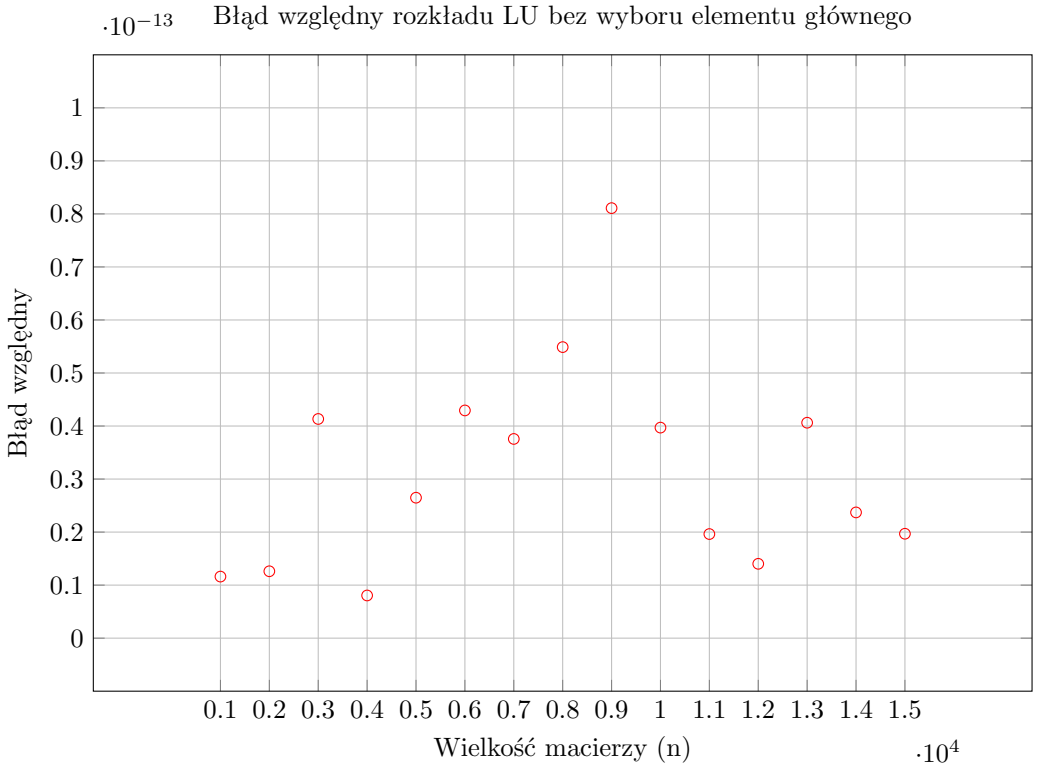
Rozkład LU bez wyboru elementu głównego

Czas wykonania z rozkładem LU bez wyboru elementu głównego



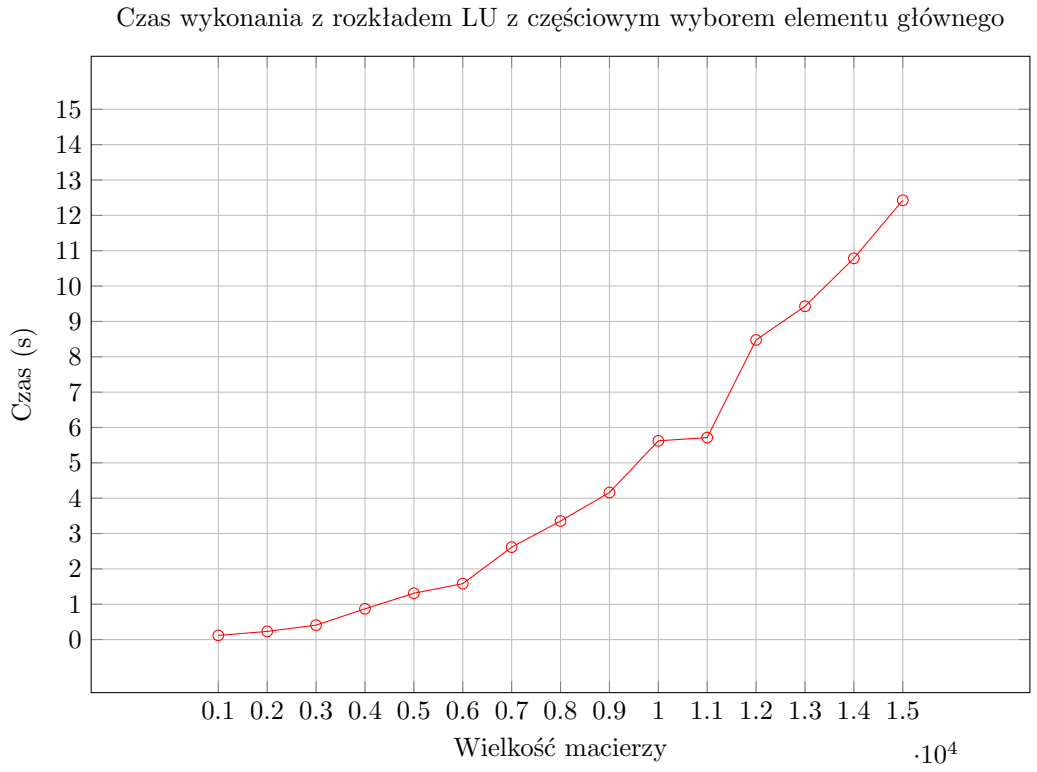
Zużycie pamięci z rozkładem LU bez wyboru elementu głównego

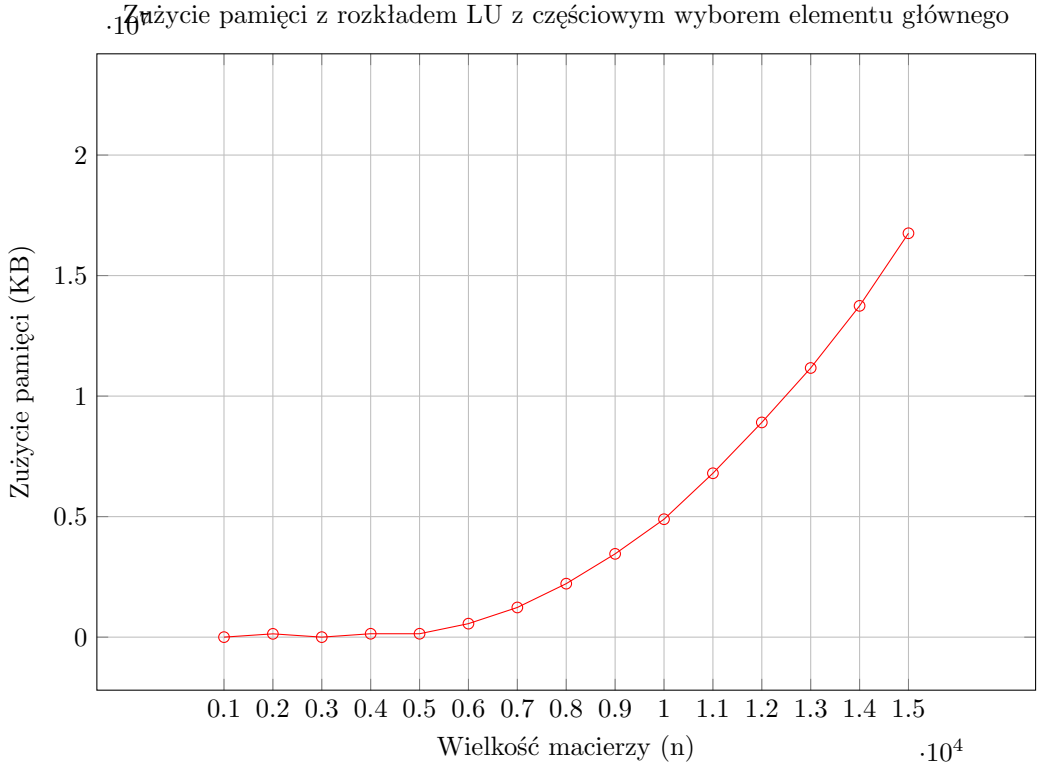


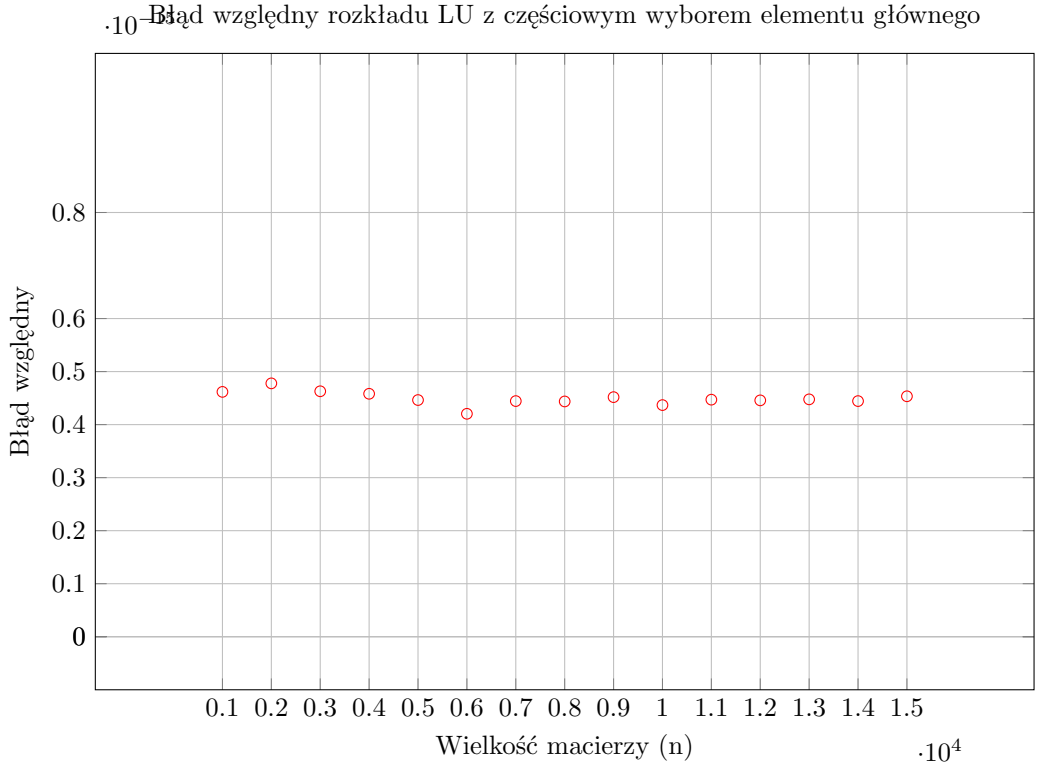


Wyniki zbliżone do otrzymanych w wyniku eliminacji Gaussa.

Rozkład LU z częściowym wyborem elementu głównego







Wyniki zbliżone do otrzymanych w wyniku eliminacji Gaussa.

Wnioski

Metoda rozwiązywania układu przy pomocy rozkładu LU cierpi na te same bolączki co eliminacja Gaussa. Wyniki są bardzo podobne. Zyskałaby ona jednak bardzo w momencie potrzeby wyliczenia rozwiązania dla różnych b. W takiej sytuacji liczylibyśmy tylko raz rozkład LU, co sprawiłoby, że wyniki byłyby miażdżące eliminację Gaussa. Obie metody: z częściowym wyborem oraz bez takiego wyboru są skuteczne. Wybór odpowiedniej powinien zależeć od tego, czy bardziej zależy nam na czasie, czy na jak najbardziej dokładnych wyliczeniach.