Obliczenia naukowe - Lista nr 5

Paweł Narolski

27 grudnia 2018 r.

1 Optymalizowanie obliczeń przeprowadzanych na macierzach rzadkich

Naszym zadaniem w ramach ostatniej listy laboratoryjnej z *Obliczeń naukowych* było rozwiązanie układu równań liniowych

$$Ax = b \tag{1}$$

dla danej macierzy współczynników $\mathbf{A} \in R^{n\times n}$ i wektora prawych stron $b \in R^n$.

Dana macierz $\mathbf{A}$ była *macierzą rzadką*, czyli macierzą, która posiada dużo elementów zerowych. Co więcej, była to także macierz blokowa o następującej strukturze:

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

przy czym zakładając, że $\ell | n$, $v = n/\ell$ (gdzie ℓ jest rozmiarem wszystkich bloków - kwadratowych macierzy wewnętrznych $\mathbf{A}_k$, $\mathbf{B}_k$ oraz $\mathbf{C}_k$.

Macierz $\mathbf{A}_k \in R^{n\times n}$ jest macierzą gęstą, $\mathbf{0}$ jest kwadratową macierzą zerową stopnia ℓ, macierz $\mathbf{B}_k \in R^{n\times n}$ jest macierzą rzadką, w której tylko ostatnia kolumna jest niezerowa (3)

$$\mathbf{B}_k = \begin{bmatrix} 0 & \dots & 0 & b_1^k \\ 0 & \dots & 0 & b_2^k \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & b_\ell^k \end{bmatrix} \tag{3}$$

zaś macierz $\mathbf{C}_k \in R^{n\times n}$ jest macierzą rzadką diagonalną (4).

$$\mathbf{C}_k = \begin{bmatrix} c_1^k & 0 & 0 & \dots & 0 \\ 0 & c_2^k & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \dots \\ 0 & \dots & 0 & c_{\ell-1}^k & 0 \\ 0 & \dots & 0 & 0 & c_\ell^k \end{bmatrix} \tag{4}$$

W celu rozwiązania układu równań liniowych (1) zaimplementowano funkcje w języku *Julia*, które uwzględniając specyficzną postać macierzy $\mathbf{A}$:

1. Dokonywały rozwiązania układu równań $Ax = b$ metodą eliminacji Gaussa
   (a) bez wyboru elementu głównego, oraz

(b) z częściowym wyborem elementu głównego

2. Wyznaczały rozkład LU macierzy $\mathbf{A}$ przy użyciu metody eliminacji Gaussa
   (a) bez wyboru elementu głównego, oraz
   (b) z częściowym wyborem elementu głównego
3. Rozwiązywały układ równań $Ax = b$ na podstawie wcześniej wyznaczonego rozkładu LU.

Powyższe funkcje zostały umieszczone w module o nazwie *blocksys* zgodnie ze specyfikacją zadania.

1.1 Metoda eliminacji Gaussa

Metoda eliminacji Gaussa to efektywny pod względem numerycznym algorytm pozwalający na obliczenie rozwiązań układów równań liniowych, a także rzędu macierzy, wartości wyznacznika macierzy czy rozkładu LU danej macierzy.

W ramach niniejszej listy laboratoryjnej *metodę eliminacji Gaussa* zastosowano w celu rozwiązania układów równań $Ax = b$ oraz wyznaczenia rozkładu $\boldsymbol{LU}$ macierzy $\mathbf{A}$.

1.1.1 Przebieg metody eliminacji Gaussa

Z *metody eliminacji Gaussa* korzystamy w celu redukcji niewiadomych w danym układzie równań liniowych $Ax = b$, doprowadzając do jego przekształcenia do równoważnego układu z *macierzą trójkątną górną*.

Eliminacja niewiadomych w *metodzie eliminacji Gaussa* dokonuje się podczas tzw. *kroku eliminacyjnego*. W k-tym kroku eliminacyjnym obliczane są *mnożniki eliminacyjne*

$$l_{ik} = A_{ik}^{(k)} / A_{kk}^{(k)}, \tag{5}$$

gdzie macierz $A^{(k)}$ jest macierzą A w k-tym kroku eliminacyjnym (dla i = k + 1, . . . , n). Po wyznaczeniu mnożników dokonujemy przekształcenia macierzy, w rezultacie otrzymując macierz $A^{(k+1)}$:

$$A_{ij}^{(k+1)} := \begin{cases} 0 & \text{dla } i = k+1, \ldots, n \text{ oraz } j = k \\ A_{ij}^{(k)} - l_{ik} \cdot A_{kj}^{(k)} & \text{dla } i, j = k+1, \ldots, n \\ A_{ij}^{(k)} & \text{dla pozostałych par wskaźników} \end{cases}. \tag{6}$$

Analogicznie dokonujemy także przekształcenia wektora prawych stron b:

$$b_i^{(k+1)} := \begin{cases} b_i^{(k)} - l_{ik} \cdot b_k^{(k)} & \text{dla } i = k+1, \ldots, n \\ b_i^{(k)} & \text{dla pozostałych wskaźników} \end{cases} \tag{7}$$

W rezultacie otrzymujemy układ równań liniowych $Ax = b$ z macierzą trójkątną górną równoważny pierwotnie danemu układowi.

1.1.2 Częściowy wybór elementu głównego w metodzie eliminacji Gaussa

Aby możliwe było wykonanie powyżej przedstawionych kroków każdy z elementów diagonali macierzy musi być różny od zera.

Dla macierzy, których elementy leżące na diagonali są różne od zera, dokonujemy modyfikacji *metody eliminacji Gaussa*, polegającej na dokonaniu tzw. *częściowego wyboru elementu głównego*.

Intuicyjnie chcielibyśmy rozwiązać powyższy problem poprzez dokonanie zamian wierszy w danej macierzy tak, aby w rezultacie na diagonali nie znajdowały się elementy zerowe. Metoda *częściowego wyboru*

elementu głównego posiłkuje się powyższym rozumowaniem w celu zapewnienia rozwiązania przedstawionego problemu.

W k-tym kroku *metody eliminacji Gaussa z częściowym wyborem elementu głównego* w k-tej kolumnie macierzy wyszukiwany jest wiersz z tzw. *elementem głównym* o największej wartości bezwzględnej, który następnie zamieniamy z k-tym wierszem. Powyższej operacji nie można dokonać jedynie dla macierzy osobliwej, czyli macierzy, której wyznacznik równy jest 0.

1.1.3 Rozwiązywanie układów równań liniowych przy pomocy metody eliminacji Gaussa

Aby rozwiązać układ równań liniowych $Ax = b$ korzystamy z *metody eliminacji Gaussa* w celu redukcji niewiadomych występujących w danym układzie.

Po przeprowadzeniu powyższych operacji pozostaje już tylko rozwiązać otrzymany układ równań liniowych $Ax = b$ z macierzą trójkątną górną.

Poszczególne wartości wektora $x = [x_1, x_2, ..., x_n]^T$ obliczamy za pomocą *algorytmu podstawienia wstecz*:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}}{a_{ii}} \tag{8}$$

dla wierszy o indeksach $i = n, n-1, \ldots, 1$.

Złożoność obliczeniowa metody eliminacji Gaussa wynosi co najwyżej $O(n^3)$ operacji, zaś algorytmu podstawienia wstecz - $O(n^2)$. Zatem w celu rozwiązania układu równań konieczne jest wykonanie łącznie $O(n^3)$ operacji[1].

1.1.4 Optymalizacja liczby wykonywanych operacji w ramach rozwiązywania układu równań liniowych z macierzą rzadką A przy użyciu metody eliminacji Gaussa

W implementacji *metody eliminacji Gaussa z częściowym wyborem elementu głównego* z uwagi na możliwość wystąpienia dużej liczby potencjalnych przestawień kolumn i wierszy w obrębie danej macierzy definiujemy wektor permutacji *perm*, który posłuży nam do przechowywania dokonywanych na poszczególnych etapach obliczeń przestawień.

Co więcej, poczynione przez nas obserwacje odnośnie specyficznej postaci macierzy **A** pozwoliły na wprowadzenie pewnych modyfikacji w wyżej opisanych algorytmach, które w efekcie przyczyniły się do znacznej redukcji liczby operacji koniecznych do wykonania w celu rozwiązania zadanego układu równań liniowych $Ax = b$ i uzyskania złożoności obliczeniowej rzędu $O(n)$.

Dokonując zarówno równoważnych przekształceń układów równań liniowych w ramach metody Gaussa, jak i później rozwiązując otrzymane układy przy pomocy algorytmu podstawienia wstecz będziemy korzystali ze zdefiniowanej przez nas zależności, która pozwoli wyznaczyć indeks ostatniej kolumny w macierzy **A**, w której danym rzędzie r znajduje się element różny od zera:

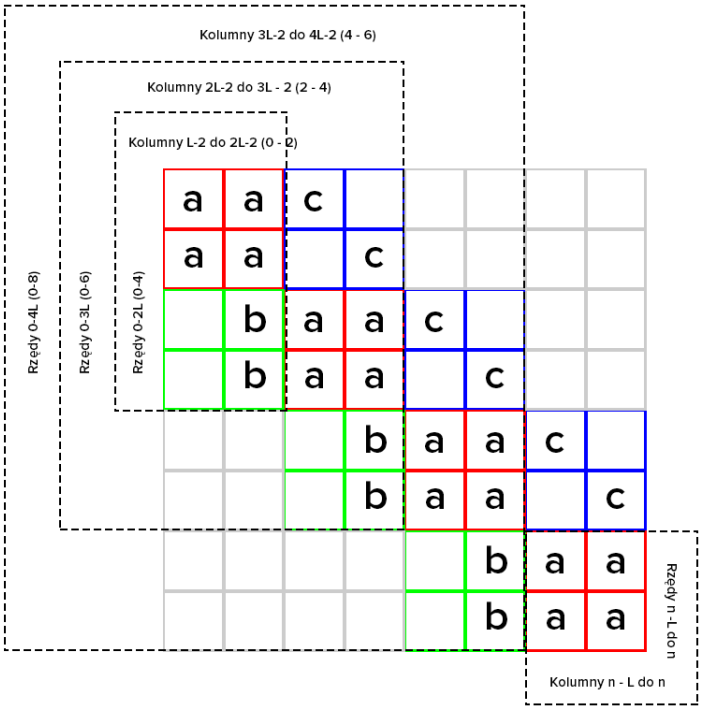
$$column_{max}(r) = min\{r + \ell, n\} \tag{9}$$

Zdefiniowanie powyższej zależności było możliwe dzięki dostrzeżeniu poniższych prawidłowości spełnionych dla macierzy **A**:

1. W kolumnach o indeksach od 0 do $\ell - 2$ elementy bloków macierzy różne od zera znajdują się w rzędach macierzy **A** o indeksie od 0 do ℓ .
2. W kolumnach o indeksach od $\ell - 2$ do $2 \cdot \ell - 2$ elementy bloków macierzy różne od zera znajdują się w rzędach macierzy **A** o indeksie od 0 do $2 \cdot \ell$.

[1] Na podstawie "Numeryczna algebra liniowa: wprowadzenie do obliczeń zautomatyzowanych", Andrzej Kiełbasiński, Hubert Schwetlick, WNT, 1994

3. W kolumnach o indeksach od $2 \cdot \ell - 2$ do $3 \cdot \ell - 2$ elementy bloków macierzy różne od zera znajdują się w rzędach macierzy $\mathbf{A}$ o indeksie od 0 do $3 \cdot \ell$ (itd...).
4. Ostatnie niezerowe elementy macierzy $\mathbf{A}$ znajdują się w jej n-tej kolumnie, ponieważ w kolumnach od $n - \ell$ do n i w rzędach od $n - \ell$ do n leży blok $\mathbf{A}_v$.
5. Podczas każdego z etapów metody eliminacji Gaussa nie dochodzi do zmiany wartości elementów leżących w kolumnach o indeksach większych, niż $column_{max}(r)$.
6. W każdym, poza l ostatnimi wierszami ostatni element różny od zera należy do bloku $\mathbf{C}_i$



Graficzna wizualizacja zauważonych prawidłowości dla $\mathbf{A} \in R^{8\times 8}$, $v = 4$, $\ell = 2$.

Na podstawie powyższych obserwacji definiujemy także zależność pozwalającą wyznaczyć indeks rzędu, w którym znajduje się ostatni w kolumnie c macierzy $\mathbf{A}$ element różny od zera:

$$row_{max}(c) = min\{n, \ell + \ell \cdot \lfloor \frac{c}{\ell} \rfloor\}. \tag{10}$$

W wyniku zastosowania metody eliminacji Gaussa dla macierzy $\mathbf{A}$ przy ustalonych granicach wykonywanych obliczeń definiowanych przez powyższe wzory otrzymujemy układ równań liniowych z macierzą trójkątną górną, który rozwiązujemy przy pomocy *algorytmu podstawiania wstecz.*

W rezultacie powyższych modyfikacji dokonujemy w sumie $(n-1)\cdot\ell^2+n*\ell$ iteracji. Złożoność obliczeniowa obu algorytmów jest rzędu $O(n)$.

1.2 Rozkładu LU

Rozkład LU danej macierzy A polega na rozłożeniu macierzy A na czynniki L i U, gdzie L jest dolną macierzą trójkątną, a U - górną macierzą trójkątną, których iloczyn

$$L \cdot U = A. \tag{11}$$

Rozkładu LU macierzy $A \in R^{n \times n}$ można dokonać tylko wtedy, gdy spełniony jest warunek:

$$Rank(A_{11}) + k \geqslant Rank(\begin{bmatrix} A_{11} \\ A_{12} \end{bmatrix}) + Rank(\begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}) \tag{12}$$

dla $k \in 1, 2, \ldots, n-1$[2].

1.2.1 Wyznaczanie rozkładu LU

Rozkład LU wyznaczamy, korzystając z wcześniej opisanej *metody elimiacji Gaussa*. Za jej pomocą dokonujemy wcześniej opisanego rozkładu macierzy A na czynniki L i U.

Analogicznie do *metody eliminacji Gaussa* w przypadku, gdy mamy do czynienia z macierzą posiadającą zerowe elementy na diagonali, konieczne będzie dokonanie rozkładu LU z tzw. *częściowym wyborem elementu głównego* - wówczas rozkładu LU dokonujemy przy pomocy *metody elimiacji Gaussa z częściowym wyborem elementu głównego*.

Złożoność obliczeniowa rozkładu LU macierzy A jest rzędu $O(n^3)$.

1.2.2 Rozwiązywanie układów równań liniowych przy pomocy rozkładu LU

Dzięki dokonaniu rozkładu LU macierzy A możliwe jest rozwiązanie układu równań liniowych $Ax = b$, który przyjmuje postać $LUx = b$.

Zadanie to sprowadza się wówczas do rozwiązania układu dwóch równań liniowych z macierzami trójkątnymi:

$$\begin{cases} L \cdot z = y \\ U \cdot x = z \end{cases} \tag{13}$$

W celu rozwiązania powyższych układów równań wykorzystujemy *algorytm podstawienia wstecz* (dla otrzymanej macierzy trójkątnej górnej U) oraz *algorytm podstawienia w przód* (dla macierzy trójkątnej dolnej L), który od poprzedniego różni się jedynie kolejnością wykonywanych operacji:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}}{a_{ii}} \tag{14}$$

dla wierszy o indeksach $i = 1, 2, \ldots, n$.

Złożoność obliczeniowa rozkładu LU jest rzędu $O(n^3)$, zaś złożoność obliczeniowa rozwiązywania układu równań liniowych przy pomocy wyznaczonego rozkładu LU wynosi $O(n^2)$.

1.2.3 Optymalizacja wykonywanych operacji w ramach rozwiązywania układu równań liniowych z macierzą rzadką A przy użyciu rozkładu LU

Celem dokonania rozkładu LU macierzy **A** do przygotowanych implementacji *metody eliminacji Gaussa* z i bez *częściowego wyboru elementu głównego* wprowadzone zostały następujące zmiany:

1. Obliczanie w poszczególnych krokach *metody eliminacji Gaussa* mnożniki eliminacyjne l_{ik} zostają zapisane, tworząc elementy macierzy trójkątnej dolnej L.
2. W wypadku *metody eliminacji Gaussa z częściowym wyborem elementu głównego* wektor permutacji zostaje zapisany i zwrócony w celu umożliwienia późniejszego rozwiązania liniowego układu równań $Ax = b$.

[2]Na podstawie "Necessary And Sufficient Conditions For Existence of the LU Factorization of an Arbitrary Matrix", Charles R. Johnson, Pavel Okunev

W celu rozwiązania układu równań liniowych:

$$\begin{cases} L \cdot z = y \\ U \cdot x = z \end{cases} \tag{15}$$

korzystamy z wcześniej zaimplementowanej optymalizacji *algorytmu podstawienia wstecz*, który umożliwia rozwiązanie układu równań $Ux = z$ z macierzą trójkątną górną, oraz z *algorytmu podstawienia w przód*, dla którego dokonano analogicznych optymalizacji w celu efektywnego rozwiązania układu równań $Lz = y$ z macierzą trójkątną górną.

Podjęte działania miały na celu uzyskanie lepszej złożoności obliczeniowej wykonywanych operacji dla danej macierzy rzadkiej $\mathbf{A}$ i taki efekt w istocie udało się osiągnąć. Podobnie jak dla wcześniejszych modyfikacji rozwiązywania układu równań liniowych przy użyciu *metody eliminacji Gaussa* wynikowa złożoność obliczeniowa jest rzędu $O(n)$.

1.3 Problem efektywnej pamięciowo reprezentacji macierzy rzadkich

Naszym zadaniem było także zaproponowanie efektywnego pamięciowo sposobu reprezentowania macierzy rzadkiej postaci $\mathbf{A}$.

Rozwiązaniem tego problemu jest przechowywanie macierzy $\mathbf{A}$ w obrębie struktury *SparseMatrixCSC* dostępnej dla programistów języka Julia, która reprezentuje daną macierz w formacie *Compressed Sparse Column (CSC)*.

Struktura *SparseMatrixCSC* jest zoptymalizowna pod kątem uzyskania możliwie jak najlepszej złożoności obliczeniowej wykonywanej na nich operacji. Umożliwia ona także szybki dostęp do elementów w kolumnach przechowywanej macierzy.

Przypomnijmy, że macierz rzadka jest macierzą posiadającą dużą liczbę elementów zerowych. W wypadku naszego zadania dla danej macierzy $\mathbf{A} \in R^{n \times n}$ o liczbie bloków równej ℓ możemy policzyć, ile jej elementów nie będzie zerami:

1. W blokach $\mathbf{A}_k$ występuje $v \cdot \ell^2$ elementów niezerowych
2. W blokach $\mathbf{B}_k$ występuje $(v-1) \cdot \ell$ elementów niezerowych
3. W blokach $\mathbf{C}_k$ występuje $(v-1) \cdot \ell$ elementów niezerowych

Sumaryczna liczba elementów niezerowych w macierzy $\mathbf{A}$ wynosi zatem:

$$n = v^2 \cdot \ell + (v-1) * \ell + (v-1) * \ell = v\ell^2 + 2\ell(v-1) = n\ell + 2(n-\ell), \tag{16}$$

co potwierdza, że dzięki zastosowaniu reprezentacji *SparseMatrixCSC* macierzy $\mathbf{A}$ możemy znacznie ograniczyć złożoność pamięciową naszych implementacji.

W implementacjach powyższych metod w języku *Julia* przeprowadzamy operacje na macierzach transponowanych, tak aby uzyskać możliwie jak najlepszą wydajność przeprowadzanych obliczeń dzięki wykorzystaniu cech struktury *SparseMatrixCSC*.

1.4 Eksperymentalne porównanie zaimplementowanych algorytmów

Dokonamy teraz porównania zaimplementowanych algorytmów dla danych testowych dołączonych do niniejszej listy laboratoryjnej z *Obliczeń naukowych*.

Dane testowe zawierały reprezentacje dobrze uwarunkowanych macierzy $\mathbf{A}$ oraz wektorów prawych stron b w formacie dołączonym do specyfikacji zadania. Ich odczyt odbywał się za pośrednictwem opracowanej na podstawie specyfikacji funkcji w języku *Julia*.

Za pomocą dołączonego do listy laboratoryjnej modułu *matrixgen* wygenerowano dodatkowe dane testowe w celu przeprowadzenia dokładniejszych porównań poszczególnych algorytmów.

1.4.1 Porównanie czasu działania i wykorzystania zasobów pamięciowych przez zaimplementowane algorytmy rozwiązywania równań liniowych metodą eliminacji Gaussa i domyślny solver języka Julia

Nasze zestawienie czasów działania i wykorzystania zasobów pamięciowych przez zaproponowane przez nas algorytmy zoptymalizowane pod kątem złożoności obliczeniowej dla rzadkiej postaci macierzy **A** rozpoczniemy od porównania działania implementacji *metod eliminacji Gaussa* z oraz bez *częściowego wyboru elementu głównego* oraz *algorytmu podstawienia wstecz* podczas rozwiązywania zadanego układu równań liniowych $Ax = b$.

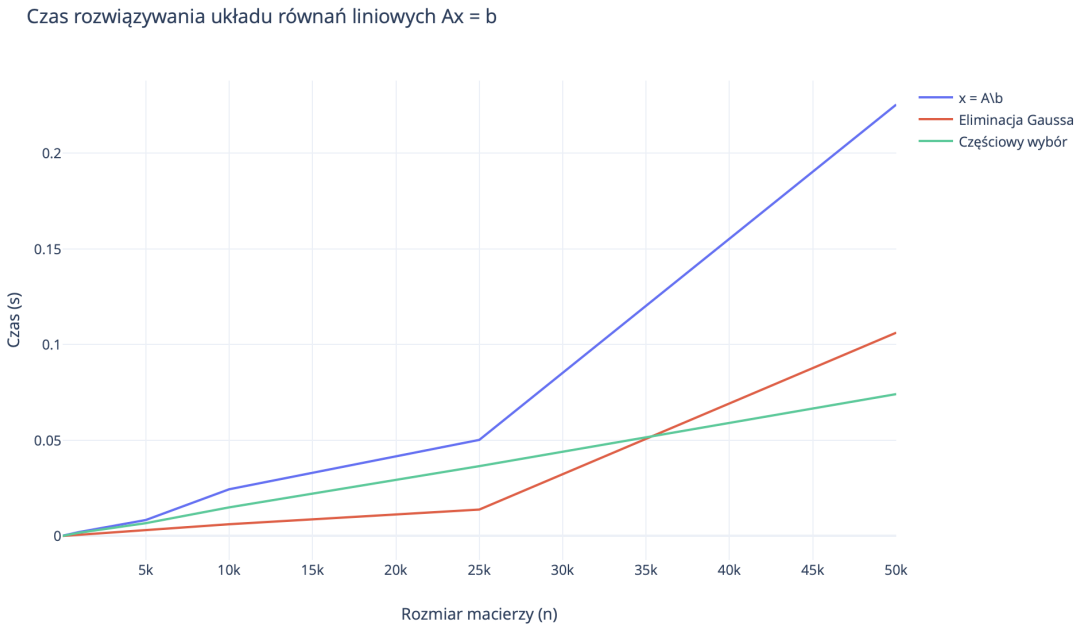
Działanie obu funkcji zostało porównane z dostępnym dla programistów języka *Julia* domyślnym *solverem* układów równań liniowych, wywoływanego za pomocą polecenia $x = Anb$, który nie uwzględnia w swoim działaniu specyficznej postaci macierzy **A**.

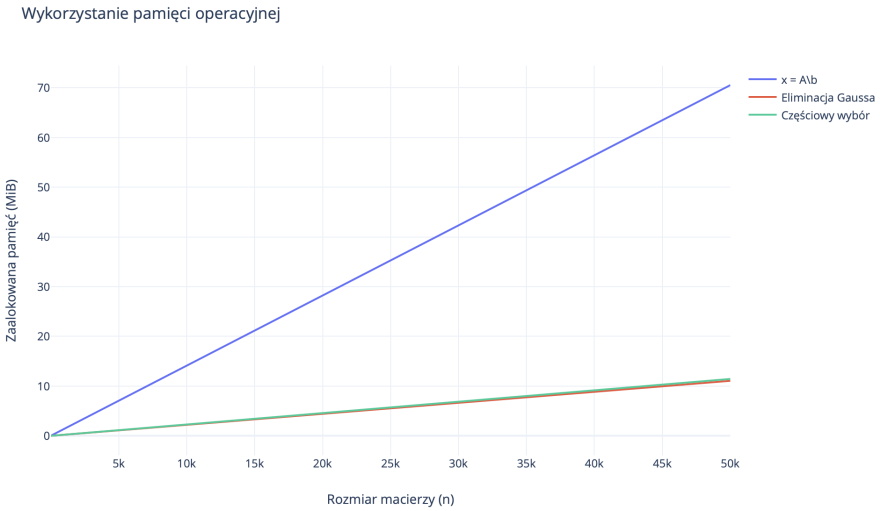
W poniższej tabeli zamieszczono porównanie czasów działania poszczególnych implementacji dla udostępnionych danych testowych (macierze o rozmiarze 16, 10000 i 50000) oraz dla wygenerowanych za pomocą funkcji *blockmat()* z modułu *matrixgen* macierzy (rozmiaru 300, 1000, 5000 oraz 25000):

	(1) x = A \b		(2) Metoda Gaussa		(2) z wyborem	
n	Czas (s)	(MiB)	Czas (s)	(MiB)	Czas (s)	(MiB)
16	0.000153	0.0396	0.000017	0.0032	0,000031	0.0034
300	0.000577	0.4418	0.000187	0.0661	0.000475	0.0685
1000	0.002049	1.4287	0.000684	0.2209	0.001689	0.2287
5000	0.008333	7.0681	0.002759	1.1059	0.006689	1.1441
10000	0.024407	14.1177	0.006138	2.2121	0.014949	2.2885
25000	0.050216	35.2664	0.013801	5.5309	0.036499	5.722
50000	0.22543	70.5142	0.106238	11.0625	0.074141	11.4438

Zauważamy znaczące różnice zarówno pomiędzy czasem działania, jak i zużyciem pamięci domyślnego *solvera* a zaimplementowanymi funkcjami. Zaimplementowane *metody eliminacji Gaussa* z i bez *częściowego wyboru elementu głównego* potrzebowały zająć ponad 6-krotnie mniej miejsca w pamięci operacyjnej komputera, a obliczenia zostały wykonane w średnio 3-krotnie krótszym czasie.

Powyższe zależności możemy zaobserwować na wygenerowanych wykresach zależności czasu wykonywania obliczeń rozwiązań układu równań liniowych $Ax = b$ od rozmiaru macierzy A i b.





Dokonano także porównania błędów względnych występujących podczas obliczeń poprzez wygenerowanie wektora prawych stron b dla $x = [1, 1, ..., 1]^T$. Wyniki zaprezentowano w tabeli poniżej:

n	Błąd dla A \b	Błąd dla eliminacji Gaussa	Błąd dla częściowego wyboru
16	5.782226546457898e-16	1.4210854715201953e-14	1.8240485749378531e-16
10000	2.8255993909143035e-16	3.330669073875471e-15	3.4834756484939387e-16
50000	3.1316044147344984e-16	3.9650683064945447e-15	3.0850578339487432e-16

Otrzymane wyniki podczas rozwiązywania układu $Ax = b$

Porównując wyniki otrzymane dla *metody eliminacji Gaussa* z i bez *częściowego wyboru elementu głównego* zauważamy, że dla drugiej metody wartości błędów względnych są o co najmniej rząd wielkości mniejsze.

1.4.2 Porównanie czasu działania i wykorzystania zasobów pamięciowych przez zaimplementowane algorytmy rozwiązywania równań liniowych przy pomocy rozkładu LU i domyślny solver języka Julia

Dokonamy teraz porównania zaimplementowanych metod rozwiązywania układów równań liniowych $Ax = b$ przy pomocy *rozkładu LU* z oraz bez *częściowego wyboru elementu głównego* uwzględniających rzadką postać macierzy **A** z domyślnym *solverem* języka *Julia*.

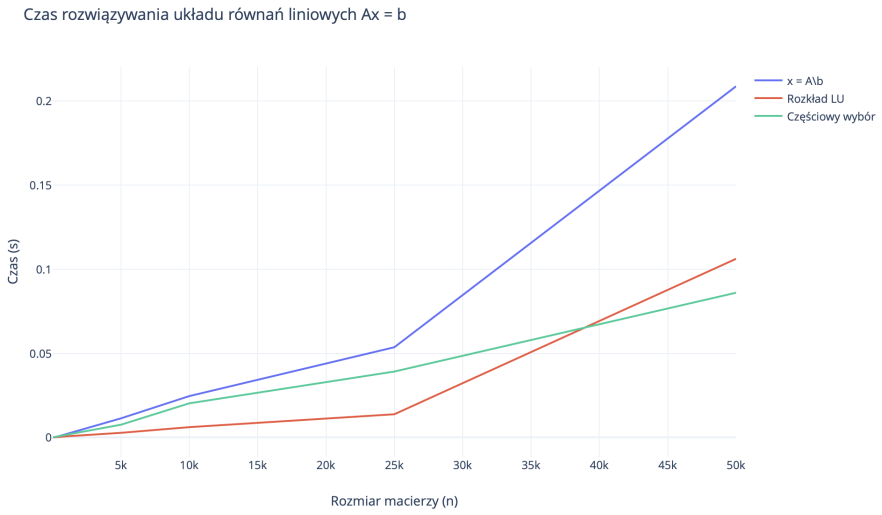
W poniższej tabeli zamieszczono porównanie czasów działania poszczególnych implementacji dla udostępnionych danych testowych (macierze o rozmiarze $16, 10000$ i 50000) oraz dla wygenerowanych za pomocą funkcji *blockmat()* z modułu *matrixgen* macierzy (rozmiaru 300, 1000, 5000 oraz 25000) dla wskaźnika uwarunkowania $cond = 25$:

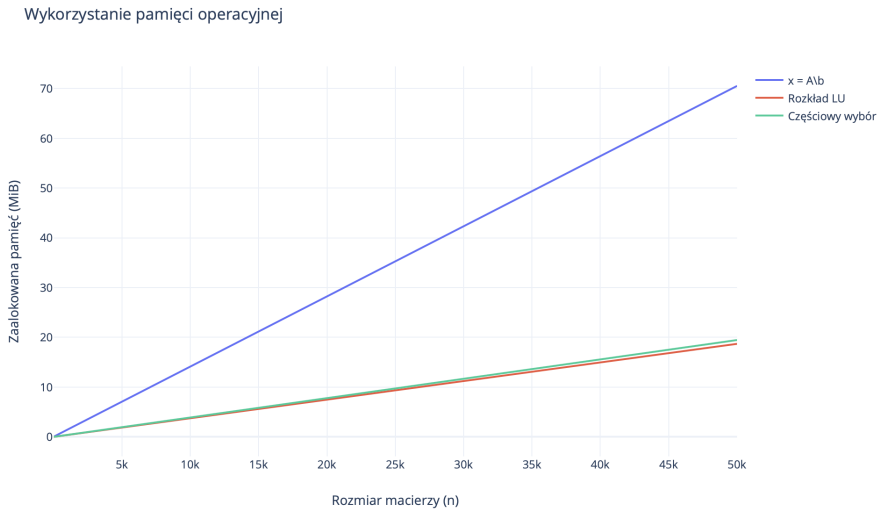
	(1) x = A \b		(2) Rozkład LU		(2) z wyborem	
n	Czas (s)	(MiB)	Czas (s)	(MiB)	Czas (s)	(MiB)
16	0.000183	0.0396	0.000015	0.0055	0.000027	0.0059
300	0.000402	0.4418	0.000188	0.1118	0.000498	0.1167
1000	0.002168	1.4287	0.000669	0.3734	0.001501	0.389
5000	0.006515	7.0681	0.003211	1.8687	0.007288	1.9452
10000	0.025295	14.1177	0.009886	3.7379	0.014735	3.8907
25000	0.05324	35.2664	0.016928	9.3455	0.037179	9.7272
50000	0.201139	70.5142	0.113246	18.6917	0.08109	19.4547

Otrzymane wyniki podczas rozwiązywania układu $Ax = b$

Ponownie zauważamy znaczne różnice pomiędzy wynikami otrzymanymi po zastosowaniu domyślnego *solvera* a zaimplementowanymi przez nas metodami, które charakteryzują się blisko 4-krotnie mniejszym zużyciem pamięci operacyjnej oraz 2-3 krotnie krótszym czasem wykonywania obliczeń.

Powyższe zależności możemy zauważyć także na przygotowanych wykresach.





Wartości błędów względnych są analogiczne co do wartości uzyskanych dla *metod eliminacji Gaussa* i zostały one pominięte.

1.4.3 Porównanie czasu działania i wykorzystania zasobów pamięciowych przez zaimplementowane algorytmy rozwiązywania układów równań liniowych

Porównamy teraz ze sobą wszystkie zaimplementowane przez nas na potrzeby niniejszej listy laboratoryjnej z *Obliczeń naukowych* metody rozwiązywania układów równań liniowych $Ax = b$ uwzględniające postać macierzy $\mathbf{A}$.

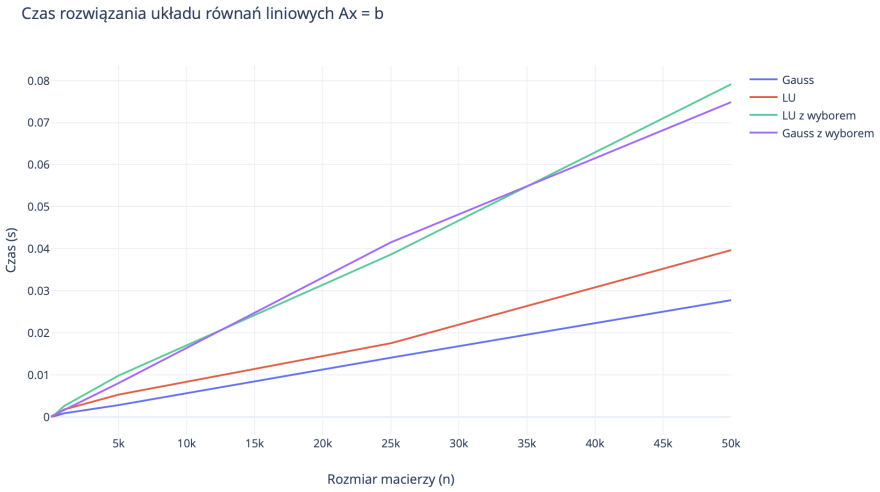
W poniższej tabeli zamieszczono porównanie czasów działania poszczególnych implementacji dla udostępnionych danych testowych (macierze o rozmiarze $16, 10000$ i 50000) oraz dla wygenerowanych za pomocą funkcji *blockmat()* z modułu *matrixgen* macierzy (rozmiaru 60, 300, 1000, 5000 oraz 25000) dla wskaźnika uwarunkowania $cond = 25$:

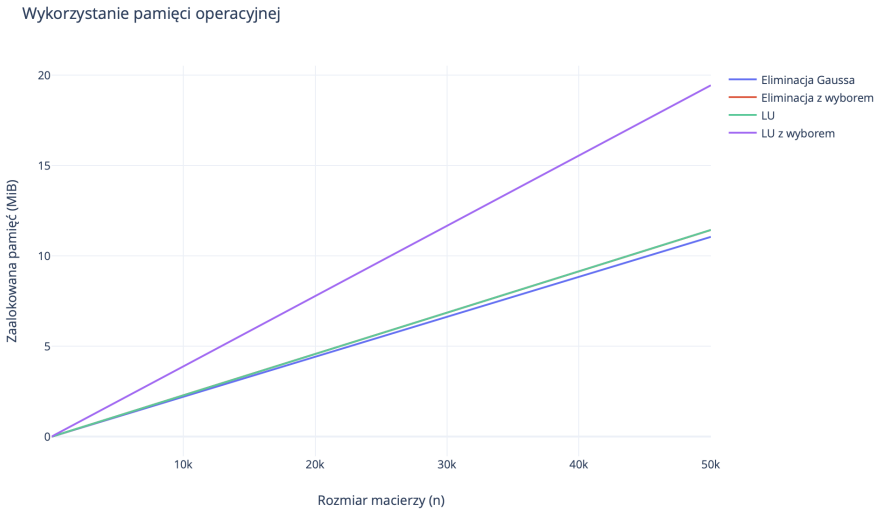
	(1) Metoda Gaussa		(1) z wyborem		(2) Rozkład LU		(2) z wyborem	
n	Czas (s)	(MiB)	Czas (s)	(MiB)	Czas (s)	(MiB)	Czas (s)	(MiB)
16	0.000011	0.0	0.000023	0.0	0.000012	0.01	0.000023	0.01
60	0.000034	0.01	0.000085	0.01	0,000040	0.02	0.000199	0.02
300	0.000186	0.07	0.00047	0.07	0.000221	0.11	0.000545	0.12
1000	0.000856	0.22	0.00172	0.23	0.000927	0.37	0.002085	0.39
5000	0.007186	1.11	0.006618	1.14	0.003927	1.87	0.008462	1.95
10000	0.005468	2.21	0.013964	2.29	0.007033	3.74	0.060184	3.89
25000	0.015135	5.53	0.037886	5.72	0.019679	9.35	0.039823	9.73
50000	0.030238	11.06	0.071582	11.44	0.035302	18.69	0.079134	19.45

Otrzymane wyniki podczas rozwiązywania układu $Ax = b$

Zauważamy stosunkowo niewielką różnicę pomiędzy ilością pamięci operacyjnej wykorzystywaną przez implementację *metody eliminacji Gaussa* z i be *wyboru elementu głównego*. Zauważamy także, że metoda rozwiązywania układów równań liniowych $Ax = b$ przy pomocy rozkładu LU wykorzystuje nieco większą ilość pamięci operacyjnej komputera z uwagi na konieczność zapamiętania elementów macierzy L i U oraz wektora permutacji w przypadku rozkładu LU *z wyborem elementu głównego*.

Porównanie zaimplementowanych metod możemy także zaobserwować na przygotowanych wykresach.





1.4.4 Podsumowanie

Zauważamy, że przygotowane optymalizacje mają diametralny wpływ zarówno na złożoność obliczeniową, jak i złożoność pamięciową implementacji *metod eliminacji Gaussa* z i bez *częściowego wyboru elementu głównego*, dokonywania *rozkładu* LU z i bez *częściowego wyboru elementu głównego* oraz rozwiązywania układów równań liniowych $Ax = b$ za pośrednictwem algorytmów *podstawiania wstecz* i *podstawiania w przód*.

Po raz kolejny przekonywujemy się, że w celu rozwiązania różnego rodzaju problemów natury obliczeniowej kluczowe jest rozpoczęcie prac od dogłębnej analizy posiadanych przez nas informacji na temat zadania. Warto odnieść przy tym do dostępnych źródeł wiedzy teoretycznej, jak i tych zawierających praktyczne rozwiązania i implementacje interesujących nas problemów z zakresu analizy numerycznej.