Obliczenia naukowe
Lista 5

Aleksandra Wójcik

Styczeń 2023

1 Opis zadania

1.1 Ogólne przedstawienie problemu

Zadanie na liście piatej nawiazuje do rozwiazywania układów równań typu **Ax=b**, dla danej macierzy $\mathbf{A} \in \mathrm{R}^{nxn}$ i wektora lewych stron $\mathbf{b} \in \mathrm{R}^{n}$. Macierz **A** jest macierza rzadka i blokowa o nastepujacej strukturze:

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

Blok A_k jest macierza gesta, blok B_k posiaga wartości niezerowe jedynie w ostatniej kolumnie, a C_k jest macierza diagonalna. Celem zadania jest efektywnie rozwiazanie układu równań **Ax=b**, pod wzgledem czasowym, ale także pamieciowym.

2 Sposób pamietania macierzy A

W celu przechowywania w pamiecie dużych macierzy rzadkich wykorzystuje uzywam typu *SparseMatrixCSC*, który jest wbudowany w standartowa biblioteke Julii *SparseArray*. SparseMatrixCSC jest efektywne pod wzgledem pamieci, ponieważ przechowuje tylko niezerowe elementy oraz informacje o ich pozycjach. W przypadku dużych macierzy rzadkich, takie podejście znacznie redukuje zużycie pamieci w porównaniu do tradycyjnych, pełnych macierzy.

3 Dostosowywanie algorytmów do macierzy blokowych, rzadkich

W celu dostaosowania algoytmów do tak specyficznej macierzy, należy zastosowac szereg jej własności. Przede wszytkim bardzo pomocnya obserwacja okazuje sie zauważenie na jakich miejsacch macierzy znajduja sie niezerowe elmenty. Wiemy, że A$\in$ R^{nxn}, natomiast bloki $A_k, B_k, C_k \in R^{lxl}$. Z tymi wiadomościami jestesmy w stanie łatwo obliczyć indeksy elementów z niezerowymi wartościiami. Weźmy przykładowa macierz:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

1. *Wiersze*

   (a) *Obliczanie pierwszego niezerowego indeksu*

   Zerkajac na powyzej przedstawiona przykładowa macierz spełniajaca wymogi zadania, możemy dostrzec powtarzajacy sie wzorzec.Jak mozemy dostzrec w wierszach macierzy za przekatna pojawiajaa sie na zmiane 1 lub 2 niezerowe wartości w sposób cykliczny. W obrebie wysokości jednego bloku, zawsze wiersz o nizszym indeksie zawiera mniejsza liczbe wartości niezerowych. Toteż można dojść do konkluzji, że skoro macierz B_k posiaza niezerowe wartości jedynie w swojej ostatniej kolumnie, to dla wierszy w obrebie wysokości tego bloku, każdy kolejny wiersz bedzie miał wiecej wartości niezerowych przed przekatna niż poprzedni, a liczba tych elementów bedzie nalezała do zbioru 1,..,k. A wiec łatwo można obliczyć, że

   $$\textit{first row index} = k - (l - (k \mod l))$$

   Oczywiście należy jeszcez pamietać, że dla pierwszych wierszy nie ma bloku B, a wiec należy obliczyć

   $$max(1, \textit{first row index})$$

   (b) *Obliczanie ostatniego niezerowego indeksu*

   Obliczenie ostatniego ostatniego indeksu jest zupełnie trywialne, ponieważ łatwo zauważyć, że dla każdego weirsza, liczba elementów wystepujacych za przekatna zawsze wynosi l. Wiec

   $$\textit{las row index} = k + l$$

Oczywiście należy uwzglednić, ze l ostatnich wierszy nie posiada bloku C, wiec należy obliczyć $max(n, k + l)$.

2. *Kolumny*

   (a) *Obliczanie pierwszego niezerowego indeksu*

   Obliczenie pierwszego indeksu zawierajacego niezerowa wartość ogranicza sie do zauważenia, że nad przekatna zawsze znajduje sie l niezerwych elementów. Z tego wynika, że

   $$\textit{first column index} = k - l$$

   Natomiast trzeba wziać pod uwage fakt, że pierwsze l kolumn nie zawiera bloku C, wiec tak naprawde trezba obliczyć $max(1, \textit{first col index})$.

   (b) *Obliczanie ostatniego niezerowego indeksu*

   Łatwo można zauważyć , że liczba niezerowych elementów pojawiajacych sie pod przekatna zmienia sie cyklicznie. To zachowanie jest bardzo podobne do zachowania opisanego w podpunkcie opisujacym wyznaczanie pierwszego indeksu zawierajacy element niezerowyy w wierszu. A wiec łatwo można zauważyć, że

   $$\textit{last column index} = k + l - (k \mod l)$$

   Natomiast trzeba wziac pod uwage fakt, że w l ostatnich kolumnach nie wystepuje dolny blok B, a wiec należy obliczyć $min(n, k + l - (k \mod l))$

Używajac powyżej wyznaczonych indeksow, jestem w stanie operować jedynie na elementach niezerowych bez konieczności przegladania macierzy. Zastosowanie tej wiedzy znaczaco przyspiesza działanie algorytmów.

4 Zadanie 1

4.1 Opis zadania

Należy napisać funkcja rozwiazujaca układ **Ax=b** metoda eliminacji Gaussa dla dwóch wariantów:

- bez wyboru elementu głównego
- z cześciowym wyborem elementu głównego

4.2 Opis metody Gaussa bez wyboru elementu głównego

Eliminacja Gausa to popularna metoda rozwiazywania układów równań. Rozwiazujac układ m równań liniowych z n niewiadomymi, należy za pomoca operacji elementarnych wyłacznie na wierszach sprowadzić macierz układu równań liniowych do postaci schodkowej, by nastepnie wyznaczyć finalne rozwiazanie.

Algorithm 1: Metoda Eliminacji Gaussa

Input: Macierz współczynników A, Wektor prawych stron b
Output: Rozwiazanie układu równań $Ax = b$

```
for k ← 1 to n − 1 do
    for i ← k + 1 to min(n, k + l − k%l) do
        m ← A[i,k] / A[k,k];
        A[i, k] ← 0;
        for j ← k to n do
            A[i, j] ← A[i, j] − m · A[k, j];
        b[i] ← b[i] − m · b[k];
for i ← n 1 do
    x[i] ← b[i] / A[i,i];
    for j ← i + 1 to n do
        x[i] ← x[i] − A[i,j] / A[i,i] · x[j];
return x;
```

4.3 Opis metodu Gaussa z cześciowym wyborem elementu głównego

W powyższej metodze nie wybierano elementu głównego, natomiast teraz zastosuje pewnego rodzaju modyfikacje. Metoda eliminacji Gaussa z cześciowym wyborem ma na celu zminimalizowanie błedów numerycznych zwiazanych z dzieleniem przez małe liczby. W tej metodzie, przed dokonaniem eliminacji w danym kroku, wybierany jest element główny jako najwiekszy (w wartości bezwzglednej) element w danej kolumnie, a nastepnie zamieniane sa ze soba odpowiednie wiersze. Reszta algorytmu wykonywana jest bez zmian. Do łatwiejszego zobrazowania problemu w pseudokodzie używam spaw, natomiast a algorytmie bede pracować na permutacjach na wektorze b, co ułatwi mi późniejsze odczytanie wyników.

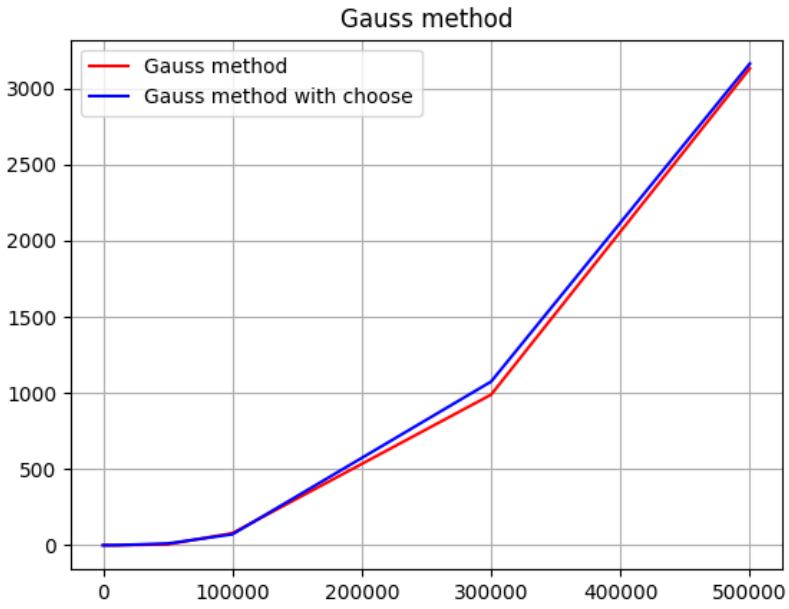
4.4 Wyniki

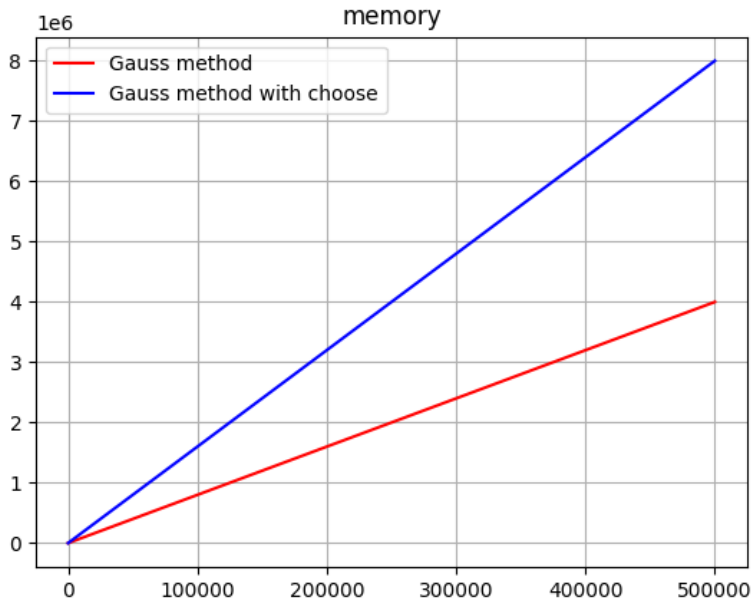
Wyniki dla zwyczajnej metody eliminacji gaussa:

Algorithm 2: Metoda Eliminacji Gaussa

Input: Macierz współczynników A, Wektor prawych stron b
Output: Rozwiazanie układu równań $Ax = b$

for $k \leftarrow 1$ **to** $n - 1$ **do**
  swap(max, k);
  for $i \leftarrow k + 1$ **to** $\min(n, k + l - k\%l)$ **do**
    $m \leftarrow \frac{A[i,k]}{A[k,k]}$;
    $A[i, k] \leftarrow 0$;
    for $j \leftarrow k$ **to** n **do**
      $A[i, j] \leftarrow A[i, j] - m \cdot A[k, j]$;
    $b[i] \leftarrow b[i] - m \cdot b[k]$;
for $i \leftarrow n$ 1 **do**
  $x[i] \leftarrow \frac{b[i]}{A[i,i]}$;
  for $j \leftarrow i + 1$ **to** n **do**
    $x[i] \leftarrow x[i] - \frac{A[i,j]}{A[i,i]} \cdot x[j]$;
return x;





4.5 Obserwacje

Z powyżej przedstawionych wykresów można zaobserwować, że zurzycie pamieci jet bardzo podobne. Natomiast złożoność czasowa dla algorytmu eliminacji Gaussa z cześciowym wyborem jest zauważalnie wyższa.

4.6 Wnioski

Powodem takiego stanu rzeczy jest wykonywanie dodatkowych operacji, majacych na celu wstawienie w dane miejsce macierzy weirsza z najwieksza wartościa elementu w danej kolumnie.

5 Zadanie 2

Zadanie 2 polega na implementacji rozkładu LU otrzymanej macierzy rzdakiej **A**, dla dwóch wariantów:

- bez wyboru elementu głównego
- z czesciowym wyborem elementu głównego

5.1 Opis rozkładu macierzy LU

Rozkład LU to dekompozycja macierzy A na iloczyn dwóch macierzy L i U, gdzie L to macierz trójkatna dolna o jedynkach na przekatnej. U to macierz trójkatna górna. Dzieki zaimplementowaniu metody eliminacji Gaussa mam już podstawe do wyznaczenia rozkładu LU, ponieważ powsatła w ten sposób macierz odpowieda macierzy U. Zawiera ona niezerowe elementy jedynie na przekatnej i powyżej jej. Z tego wynika, pozostaje wyznaczenie macierzy L.

Macierz dolno trójkatna posiada 1 na przekatnej, z tego powodu konstrukcje L rozpoczniemy od stworzenia macierzy jednostkowej o zadanych wymiarach. Nastepnie L również bedzie konstruowana podczas przebiegu algorytmu eliminacji gaussa, zapisujac współczynniki użyte do eliminacji w odpowiednich miejscach macierzy.

Jednakże, w celu efektywnego pamietania macierzy L i U, wystarczy jedna macierz, badżac połaczeniem tych dwóch.

Pseudokod:

Algorithm 3: Rozkład LU

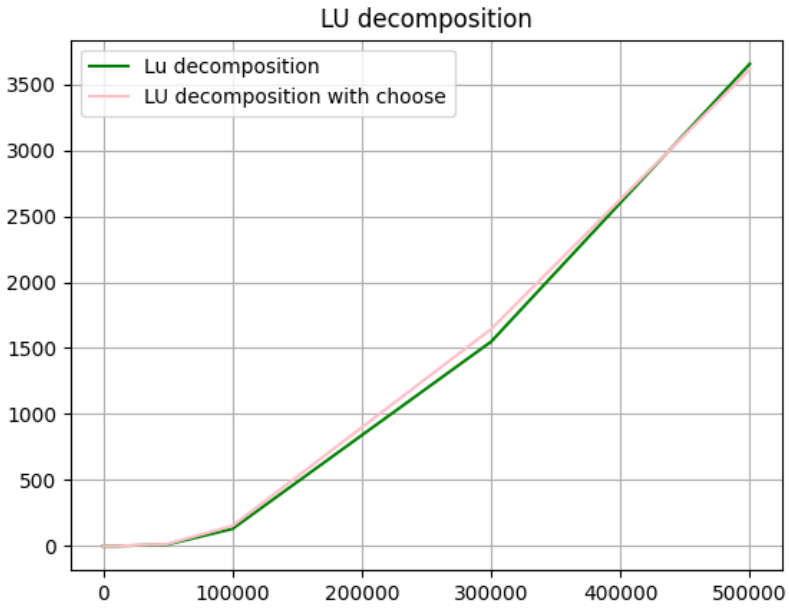
Input: Macierz współczynników A, macierz jednostkowa L
Output: Dekompozycja LU

```
1 for k ← 1 to n − 1 do
2     for i ← k + 1 to min(n, k + l − k%l) do
3         m ← A[i,k] / A[k,k];
4         A[i, k] = 0;
5         L[i, k] = m;
6         for j ← k to n do
7             A[i, j] ← A[i, j] − m · A[k, j];
8 return x;
```

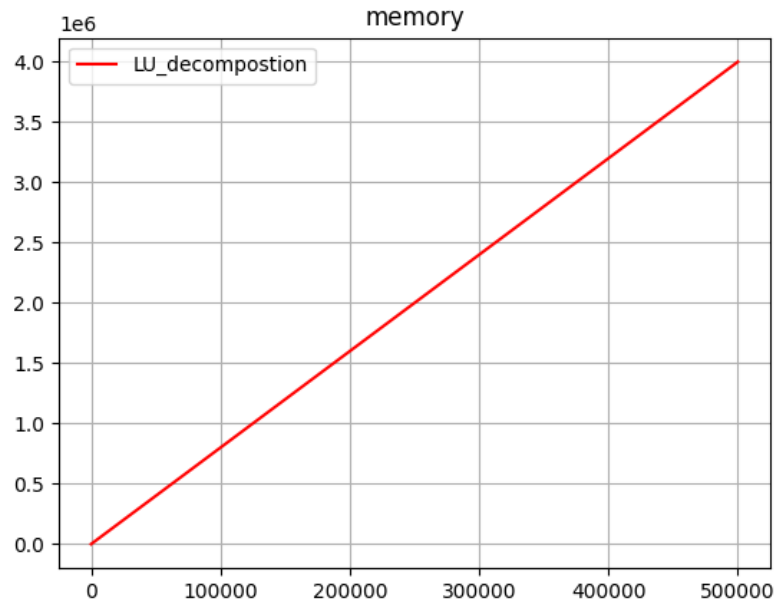
Pseudokod dla rozkładu LU z czesciowym wyborem zawiera dodawtowo wybór i zamiane wiersza, który ma aktualnie najwyzsza wartość w kolumnie.

5.2 Wyniki

Wyniki dla dekompozycji LU bez wyboru elementu głównego:



Wyniki dla dekompozycji LU z cześciowym wybore elementu głównego:



5.3 Obserwacje

Z powyżej przedstawionych wykresów można zaobserwować, że zurzycie pamieci jet bardzo podobne.

5.4 Wnioski

Wyższa złożoność czasowa dla przypadku z cześciowym wyborem wynika z wykonywania dodatkowych operacji polegajacych na wybraniu najwiekszego elenemu z danej kolumny, bedazego pod rozważanym wierszem, a nastepnie namianie wierszy, w przypadku znalezienia wiekszego elementu.

6 Wnioski ogólne

Rozwiazanie dla owawianych metodm zawsze maja lepsza złożoność czasowa dla przypadków bez wyboru elementu głównego. Mim togo, że ustalanie elementu głównego powoduje zwiekszenie sie czasu oczekiwania na wynik, to trzeba zauważyć, że wybór elementu głównego pozytywnie wpływa na jakość otzrymywanych wyników, ponieważ eliminuje błedy numeryczne, wynikajace z dzielenia liczb przez niewielkie wartości.