Obliczenia Naukowe - Lista 5

Szymon Brzeziński

9 stycznia 2022

1 Opis problemu

Opracowanie sposobów rozwiązywania układów liniowych $\mathbf{Ax}=\mathbf{b}$.
Dla danej macierzy współczyników $\mathbf{A} \in \mathbb{R}^{n\times n}$,
oraz wektora prawych stron $\mathbf{b} \in \mathbb{R}^{n\times n},$ dla $n \geqslant 4$
Macierz $\mathbf{A}$ jest macierzą rzadką w następującej postaci:

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

gdzie $v = n/l, l$ rozmiar kwadratowych macierzy wewnetrzynych: $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k$:

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

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

Opracowanie sposobu rozwiązązywnia ów układów polega na optymalizacji wymagań czasowych i pamięciowych poprzez lepszy sposób zapamiętywania macierzy gestych, czyli takich które posiadają dużą ilość elementów zerowych. Mając na uwadze specyficzną postać macierzy **A**, napisać funkcję rozwiązującą układ metodą elimacji Gaussa z brakiem wyboru elementu głównego oraz z jego częściowym wyborem.

2 Rozwiązanie

2.1 Sposób przechowywania

Pamiętanie macierzy **A** w sposób standardowy,
jako tablicy $n \times n$ ($O(n^2)$ miejsca) jest niefektywne z powodu rozmiaru n,
Ponieważ mamy do czynienia z macierzą gęstą, lepszym podejściem jest pamiętanie tylko elementów niezerowych.
W języku Julia istnieje struktura Sparse Arrays, która to poprzez sparsowaną macierz(bez elementów zerowych) przechowyję w formacie Compressed Sparse Column (CSC) o następującej reprezentacji:

```
struct SparseMatrixCSC{Tv,Ti<:Integer} <:
    AbstractSparseMatrixCSC{Tv,Ti}
   m::Int                # Number of rows
   n::Int                # Number of columns
   colptr::Vector{Ti}    # Column j is in colptr[j]:(colptr[j+1]-1)
   rowval::Vector{Ti}    # Row indices of stored values
   nzval::Vector{Tv}     # Stored values, typically nonzeros
end
```

gdzie Tv jest typem przechowywanych wartościm Ti natomiast typem całkowym do przechowywania wskaźników kolumn i indeksów wierszy. Dzięki pamiętaniu w trzech tablicach kolejno numeru kolumny, wiersza i wartości, jesteśmy w stanie iterować się po wszystkich wartościach czytając te same indeksy dla każdech z ów tablic.

3 Opis algorytmów

3.1 Eliminacja Gaussa

Algorytm eliminacja Gaussa, polega na przekszłaceniu macierzy przy użyciu elementarnych operacjach na macierzy, w macierz trójkątną, to znaczy taką której wszystkie wartości pod przekątną są wyzezerowane.

Eliminacja przebiega poprzez iterację po wierszach macierzy i na wykonywaniu w każdym kroku odejmowania kolejnych wierszy przemnożonych przez czynnik znajdujący się pod nimi.
Wykonanie k-tego kroku polega na następującym podstawieniu,
dla każdego $i > k$:
$[a_{i,1}\ a_{j,2}\ ...\ a_{i,j}\ b_i] = [a_{i,1}\ a_{i,2}\ ...\ a_{i,j}\ b_i]$ - $\frac{a_{i,k}}{a_{k,k}}$ $\times$[a$_{k,1}a_{k,2}...a_{k,l}b_k$]
Po otrzymaniu macierzy trójkątnej, wystarczy iteracja od n do 1 i wykorzystanie wzoru:

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

3.2 Eliminacja Gaussa z częściowym wyborem elementu głównego

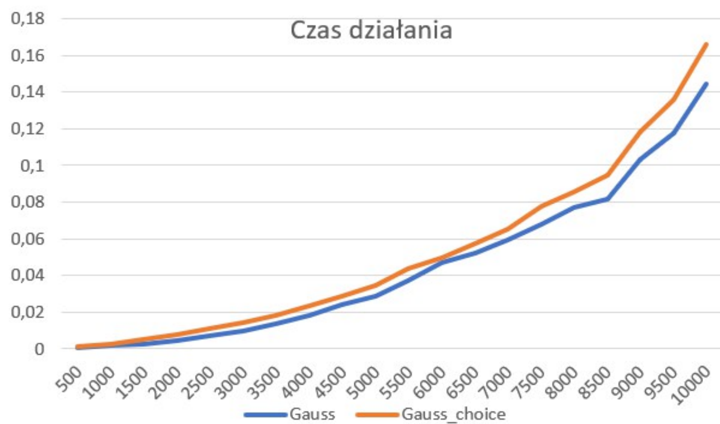
Algorytm eliminacji Gaussa w momencie w którym na wartości na przekątnej są zerowe lub bliskie zeru, przestaje działać. By się przed tym uchronić wubiera się w każdym kroku działania algorytmu rzędu z elementem o największej wartości(bezwględnej), tak zwany element główny. Wybrany rząd zostaje zamieniony z aktualnie iterowanym wierszem.

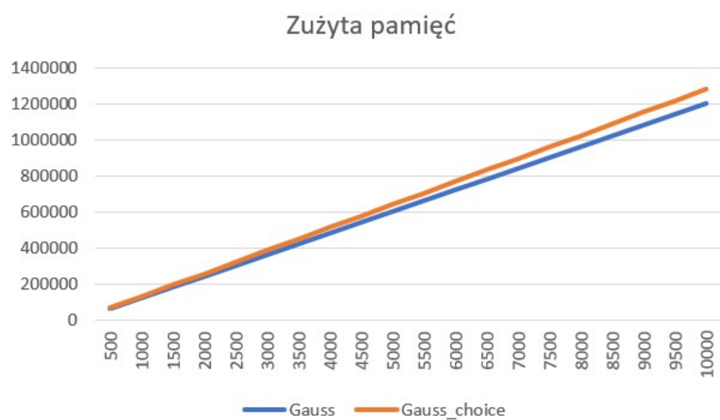
4 Optymalizacja algorytmów

Złożoność elimacji Gaussa wynosi O(n^3), wynika ona z odjęcie n-k wierszy od k-tego wiersza a także aktualizacji n+1 wartości we wierszu. Algorytm z wyborem częściowego elementu głównego ma taką samą złożoność obliczeniową, lecz samo znalezienie i podmiana elementu wydłuża czas działania.
Wieksza część wartości pod przekątną macierzy posiada wartości zerowe, co oznacza, że nei musimy ich już zerować co z kolei oznacza, że mozna zoptymalizować algorytm.
Maksymalnym elementem niezerowym w danej kolumnie będzie l(rozmiar bloku czyli rozmiar macierzy blokowych)
Dodatkowo ilość odjęć od następnych wierszy wynosi także l, ponieważ następne wiersze są już wierzami zerowymi.
Powyższe ograniczenia iteracji zmieniają złożoność obliczeniową do O(l^2n).

5 Wyniki eksperymentów

Poniżej znajduja się wykresy czasu trwania, oraz ilości użycia pamięci dla obydwu algortytmów: Eliminacji Gaussa oraz Eliminacji Gaussa z częściowym wyborem głównym.





Wariant z częściowym wyborem elementu głównego, ma gorsze wyniki, co wynika z faktu szukania i podstawiania elementu głównego, lecz zgodne z tezą wyżej złożoność obliczeniowa jest taka sama jak w wariancje bez wyboru. Widać jednak rozbieżność wykresów względem oczekiwanych złożonośći $\mathrm{O}(n)$, (nie $\mathrm{O}(l^2 n)$ponieważ w testowanych przypadkach l jest stałe), wynikają one z braku założenoego w obliczeniach dostępu do rzedów w SCS w czasie stałym.

6 Wnioski

Dostowanie rozwiązań względem specyfikacji badanych obiektów potrafi w znaczący sposób zmiejszyć złożoność obliczeniową oraz pamięciową. Można nawet sądzić, że w niektórych przypadkach jest to konieczne podejście do problemu.