Obliczenia Naukowe

Laboratorium Lista Nr 5
Piotr Popis
245162

6 grudzień 2019

1 Wstęp

1.1 Streszczenie

Problemem jest rozwiązanie równania liniowego $Ax = b$,gdzie $A \epsilon R^{nxn}$ jest podaną macierzą, a $b \epsilon R^n$ zadanym wektorem prawych stron(przy założeniu, iż $n \geq 4$). Dodatkowo macierz A jest macierzą rzadką- taką, która ma dużo elementów zerowych oraz blokową.

$$A = \begin{bmatrix} A_1 & C_1 & 0 & ... & 0 \\ B_2 & A_2 & C_2 & .. & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & ... & B_{v-1} & A_{v-1} & C_{v-1} \\ 0 & ... & 0 & B_v & A_v \end{bmatrix}$$

, gdzie $v = \frac{n}{l}$ przy założeniu iż l zawsze dzieli n(n jest podzielne przez l) oraz $l \geq 2$. l jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych - bloków: A_k, B_k, C_k. Mianowicie:

$$A_k \epsilon R^{lxl}, k = 1, ..., v,$$

A jest macierzą gęstą,
0 jest kwadratową macierzą zerową stopnia l,
Natomiast macierz

$$B_k \epsilon R^{lxl}, k = 2, ..., v,$$

B_k ma tylko dwie ostatnie kolumny niezerowe i jest postaci:

$$B_k = \begin{bmatrix} 0 & ... & 0 & b^k_{1l-1} & b^k_{1l} \\ 0 & ... & 0 & b^k_{2l-1} & b^k_{2l} \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & ... & 0 & b^k_{ll-1} & b^k_{ll} \end{bmatrix}$$

Ostani z bloków

$$C_k \epsilon R^{lxl}, k = 1, ..., v-1,$$

C_k jest macierzą diagonalną i jest postaci:

$$C_k = \begin{bmatrix} c_1^k & 0 & 0 & ... & 0 \\ 0 & c_2^k & 0 & .. & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & ... & 0 & c_{l-1}^k & 0 \\ 0 & ... & 0 & 0 & c_l^k \end{bmatrix}$$

Z treści n jest ogromne co wiąże się dużym obciążeniem pamięciowym jak i czasowym w przypadku zwykłej tablicy. Należy skorzystać z pakietu SparseArrays, która zawiera specjalną strukturę efektywnie pamiętająca specyficznie macierze, tj rzadkość lub regularność występowania elementów zerowych i niezerowych. Istniejące algorytmy do rozwiązywania takich problemów trzeba po prostu zmodyfikować do użycia tej spejcalnej struktury. Jeśli l jest stałe Algorytmy da się zoptymalizować czasowo z $\mathcal{O}(n^3)$ do $\mathcal{O}(n)$.

1.2 Treść

Zadanie 1 Należy stworzyć funkcję rozwiązującą układ $Ax = b$ metodą eliminacji Gaussa uwzględniającą postać macierzy A zadanej w streszczeniu dla dwóch wariantów
(a)bez wyboru elementu głównego
(b)z częściowym wyborem elementu głównego

Zadanie 2 Należy napisać funkcję wyznaczającą rozkład LU macierzy A metodą eliminacji Gauss'a uwzględniającą specyficzną postać macierzy A dla
(a)bez wyboru elementu głównego
(b)z częściowym wyborem elementu głównego

Zadanie 3 Należy napisać funkcję rozwiązującą układ równań $Ax = b$ (uwzgledniającą specyficzną postać macierzy A).

Wszystkie funkcje powinny byc umieszczone w module o nazwie blocksys. Należy przeczytać Sparse Arrays manual Julia. Założyć, że dostęp do elementu macierzy jest w czasie stałym. Nie można używać $x = \frac{A}{b}$ oraz lu z modułu LinearAlgebra.

2 Zadanie 1

2.1 Opis standardowej procedury wraz z analizą złożoności algorytmu

Na czym polega metoda eliminacji Gauss'a? Metoda ta polega na sprowadzeniu układu równań(macierzy) do równoważnego układu z wykorzystaniem macierzy trójkątnej górnej, następnie rozwiązaniu tego układu przy pomocy algorytmu podstawiania wstecz.

Na czym polega algorytm podstawiania wstecz? Algorytm ten bazuje na zerowaniu kolejnych elementów macierzy poniżej diagonali(czyli tej niezerowej przekątnej).

Przebieg procedury 1. Zerowanie elementów poniżej pierwszego wiersza w pierwszej kolumnie.
2. Ogólnie, aby wyzerować a_{i1} od wiersza i-tego odejmowany jest wyraz pierwszy pomnżony przez liczbę $\frac{a_{i1}}{a_{11}}$
3. Następnie przechodzimy do kolejnej kolumny(tutaj drugiej itd) i powtarzamy powyższe procedury z taką zmianą, że teraz odejmowany wiersz i(tutaj drugi a_{22} itd).
Niestety procedura nie zadziała jeśli któртkolwiek z diagonalnych elementów będzie zerem(jak widać we wzorze). Aby rozwiązać ten problem należy przeprowadzić odpowiednią modyfikację. W i-tym kroku , w i-tej kolumnie należy wyszukać w kolejnych wierszach j-ty element o wartości co do modułu największej i zamienić wtedy a_{ii} z a_{ji} (wzór: $a_{wierszkolumna}$).

Następnie korzystamy z algorytmu wstecz, czyli matematycznie wzoru: $x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}}{a_{ii}}$.

Począwszy od ostatniego indeksu.(n)
Zakładając, że n jest rozmiarem macierzy złożonośc obliczeniowa eliminacji Gaussa wynosi co najwyżej $\mathcal{O}(n^3)$, a algorytm podstawiania wstecz $\mathcal{O}(n^2)$. Łącznie, aby rowiązać układ należy wykonać $\mathcal{O}(n^3)$ operacji.

2.2 Opis implementacji wraz z analizą złożoności algorytmu

2.2.1 SparseMatrix pamięć

Celem zadania jest modyfikacja i optymalizacja algorytmu. Zauważmy, że rozpatrywana macierz ma dość specyficzną, nietypową postać. Jest macierzą rzadką. Ma $(l+3)n - 3l$ elementów, które nie są zerami.

l^2 - w każdym z v bloków A_k,
$2l$ - w każdym z v-1 bloków B_k,
l - w każdym z v-1 bloków C_k

Do przechowywania macierzy wykorzystamy struktuę do przechowywania macierzy rzadkich SparseMatrixCSC. Macierze takie są przechowywane w skompresowanym porządku kolumnowym. Algorytm Gauss'a natomiast ma przebieg wierszowy, zatem w implementacji musimy zamienić miejscami indeksy kolumny i wiersza i pracować na macierzach trasnponowanych. Aby ułatwić proces zrozumienia algorytmu uznaję to za problem implementacyjny i indeksuję w roważaniach w sposób standardowy.
Dzięku użyciu takiej struktury mamy szybszy dostęp do elementów.

2.2.2 Modyfikacja, optymalizacja algorytmu

Zwrócmy uwagę na postać macierzy A. Jest to macierz diagonalna, a nawet trójdiagonalna. W dodatku jest to macierz blokowa(A_k, B_k, C_k). Zauważmy, że nie jest konieczne zerowanie wszystkich elementów poniżej diagonali(przekątnej), bo już są wyzerowane.
Pozwala to zredukować ilość wykonywanych obliczeń.
W pierwszych $l-2$ kolumnach potencjalne niezerowe elementy znajdują sie w l-pierwszych wierszach i są to elementy bloku A_1, dla kolejnych l kolumn elementy niezerowe znajdują się prawdopodobnie w pierwszych $2l$ wierszach są to elementy bloku A_3 oraz dwie ostatnie kolumny bloku B_2. W kolejnych l kolumnach niezerowe elementy znajdują się w peirwszych 3l wierszach są nimi elementy bloku B_3 oraz elementy bloku A_4 rzecz jasna niezerowe elementy.
Zatem ostatni niezerowy element w danej kolumnie można obliczyć korzystając z funkcji min(). Ostatecznie ostatni niezerowy element w kolumnie wyrażamy wzorem: $min\Big(n,\, l + l\Big\lfloor \frac{column+1}{l} \Big\rfloor\Big)$

2.3 Wyniki eksperymentów porównujących zaimplementowane algorytmy dla danych testowych(tabele, wykresy) oraz interpretacja

2.4 Wnioski