Obliczenia Naukowe

Sprawozdanie z laboratorium nr 5

Piotr Kawa

Wrocław 03.01.2018

0.Wprowadzenie

Celem listy 5 było stworzenie algorytmów rozwiązujących problemy obliczeniowe pewnej firmy. Zagadnienia dotyczyły operacji na macierzy specyficznej postaci. Miała ona następującą strukturę:

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

Macierz opisywały następujące zmienne:

n – wymiar macierzy A (rzadka, blokowa macierz) oraz wektora prawych stron b ($n \geq 4$)

l – wymiar macierzy wewnętrznych ($l \geq 2$)

$v - \frac{n}{l}$ zakładając, że n jest podzielne przez l

Struktura składała się z 4 rodzajów macierzy wewnętrznych rozmiaru $l\ x\ l$:

0 – macierz zerowa

A_k – macierz gęsta

B_k – macierz w poniższej postaci, gdzie tylko ostatnie dwie kolumny były niezerowe

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

C_k – macierz diagonalna

$$C_k = \begin{bmatrix} c^k_1 & 0 & 0 & \dots & 0 \\ 0 & c^k_2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & c^k_{l-1} & 0 \\ 0 & \dots & 0 & 0 & c^k_l \end{bmatrix}$$

Złożoność obliczeniowa zaimplementowanego rozwiązania, w odróżnieniu od standardowej eliminacji Gaussa wynoszącej $O(n^3)$, miała wynosić $O(n)$ (przy założeniu, że l jest stałą), co wymagało pewnych modyfikacji. Funkcje zostały napisane w języku Julia oraz umieszczone w module $blocksys$. Dane wejściowe przekazywane były poprzez odpowiednio sformatowane pliki – jeden zawierający dane dotyczące macierzy, drugi natomiast wektora. Rezultat końcowy zapisywany był do pliku o nazwie podanej przez użytkownika.

Stworzone zostały również funkcje, które umożliwiały testowanie zaimplementowanego algorytmu oraz wyliczenie błędu względnego. W przypadku, gdy wektor prawych stron b był wyliczany, plik wynikowy zawierał oprócz wektora wynikowego również informację o błędzie względnym.

1. Zadanie pierwsze

1.1. Cel zadania

Celem zadania pierwszego było napisanie funkcji, która rozwiązywała układ $Ax = b$ metodą eliminacji Gaussa. Musiała ona uwzględniać specyficzną postać macierzy oraz umożliwiać dwa tryby wykonywania:

- bez wyboru elementu głównego
- z częściowym wyborem elementu głównego.

1.2. Przechowywanie danych

Wiele z elementów zadanej macierzy było elementami zerowymi – przechowywanie wszystkich danych (czyli n^2) było zatem wielce nieoptymalnym sposobem. Rozwiązaniem była struktura należąca do języka Julia o nazwie $SparseMatrixCSC$. Zawierała ona tylko elementy niezerowe, co znacznie zmniejszyło złożoność pamięciową.

1.3. Eliminacja Gaussa

Eliminacja Gaussa jest algorytmem służącym do rozwiązywania układów równań liniowych. Składa się ona z dwóch etapów – pierwszy z nich polega na doprowadzeniu podanej macierzy do postaci macierzy trójkątnej górnej poprzez redukcję do zera elementów znajdujących się poniżej przekątnej. Następuje po nim rozwiązanie powstałego układu – w przedstawianej implementacji był to algorytm podstawiania wstecz. Całość wykorzystuje elementarne operacje. Należą do nich między innymi dodanie do siebie równań (wierszy) pomnożonych przez liczbę czy też zamiana równań miejscami.

Ze względu na specyfikę podanej macierzy oraz założenie, że dane, które miała ona przechowywać, były duże, potrzebne było wprowadzenie pewnych modyfikacji do algorytmów. Główną ich myślą było wykonywanie jak najmniejszej ilości obliczeń na macierzach 0 poprzez wykonywanie obliczeń możliwie tylko w zakresie bloków A_k, B_k, C_k. Skróciło to znacząco czas wykonywania programu – widoczny był kilku, kilkunastokrotny skok wydajności. Zmniejszenie ilości operacji polegało na odpowiednim modyfikowaniu zakresów pętli, tak by możliwie cały czas operować tylko na wartościach niezerowych – możliwe to było dzięki znajomości struktury macierzy A.

Podczas sprowadzania macierzy do postaci schodkowej (jest to poprawne określenie z racji, iż macierz A jest kwadratowa) wyliczane były współczynniki $coeff = \frac{mat_{m,k}}{mat_{k,k}}$. Określały one to ile razy pomnożony musi być jeden wiersz, by odjęty od drugiego dał zero.

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix}$$

Przykład macierzy trójkątnej górnej – wyniku pierwszego etapu eliminacji Gaussa

Podstawowa wersja eliminacji Gaussa nie sprawdzała się w przypadku, gdy elementem głównym było 0 – miało to między innymi miejsce w jednym z zadanych testów – mianowicie $Dane8$. Zostało stworzone zabezpieczenie ($matrix[k,k] < eps(Float64)$), które kończyło program z komunikatem opisującym problem, jednakże nie pozwalało to nadal na obliczenie pewnych macierzy.

Był to powód dla którego właśnie należało zaimplementować możliwość tzw. częściowego wyboru elementu głównego. Algorytm ten wykonywany był za każdą iteracją głównej pętli programu – tj. podczas zerowania każdej kolumny. Polegał on na znalezieniu największego elementu ($abs(element)$) w danej kolumnie zaczynając od aktualnego wiersza. Specyfikacja macierzy powodowała naturalnie znowu to, że pojawiały się zbędne kroki – od pewnego momentu w danej kolumnie znajdowały się tylko wartości zerowe, dlatego sprawdzane były jedynie wartości znaczące – maksymalnie oddalone o $2 * l$.

Kolejnym etapem była zamiana wiersza zawierającego największy element z wierszem aktualnym. W danym wierszu mogło być maksymalnie $3 * l$ wartości niezerowych – przy większych macierzach generowało to bardzo dużo zamian pomiędzy zerami, czyli operacji nic nie wnoszących do wyniku końcowego. Remedium ponownie było ograniczenie zamian tylko do elementów niezerowych.

Algorytm częściowego wyboru elementu głównego kończył się zamianą dwóch elementów wektora b – elementu p-tego z k-tym (gdzie p i k to aktualny indeks oraz indeks wiersza zawierającego największy element).

Wynikiem zerowania kolumn była macierz trójkątna górna współczynników. W celu rozwiązania tego układu zastosowany został algorytm podstawiania wstecz wyliczający kolejne wartości wektora wynikowego x w następujący sposób:

$$x_n = \frac{b_n}{a_{nn}}, x_i = \left(b_i - \sum_{j=i+1}^{n} (a_{ij} * x_j) \right) / a_{ii}$$

Wyliczony wektor był następnie, zgodnie ze specyfikacją zadania, zapisywany do pliku. Jego nazwa była wybierana przez użytkownika.

$$\begin{bmatrix} 1 \\ -3 \\ -2 \\ 1 \end{bmatrix}$$

Przykładowy wektor wynikowy będący rezultatem eliminacji Gaussa

1.5 Analiza złożoności obliczeniowej

Standardowy algorytm eliminacji Gaussa ma złożoność obliczeniową wynoszącą $O(n^3)$. Złożoność zaimplementowanego algorytmu miała wynosić $O(n)$.

Główna pętla programu wykonuje się $n - 1$ razy. Do algorytmu należy również wybór elementu głównego oraz stworzenie macierzy trójkątnej (zerowanie elementów poniżej przekątnej) – odpowiednio o złożoności $O(3 * l)$ oraz $O(2 * l)$, co daje ogólną złożoność na poziomie $O(2l * 3l * n)$, co przy założeniu, że l jest stałą skutkuje złożonością $O(n)$ – zgodnie z założeniami polecenia.

1.6 Dokładność rozwiązania

W celu weryfikacji różnicy między wynikami otrzymanymi dzięki wersji z lub bez wyboru elementu głównego wykorzystany został następujący wzór $\frac{norm(result-ones(n,1))}{norm(ones(n,1))}$ obliczający błąd względny wyników otrzymanych dla tych samych zestawów danych.

n	$wybór$	$brak\ wyboru$
16	$2.528653e-16$	$8.825225e-16$
10000	$5.351056e-16$	$1.101415e-14$
50000	$5.215335e-16$	$1.932599e-13$

1.7 Wnioski

Powyższe wyniki jasno pokazują większą dokładność metody z wyborem elementu głównego. Wyniki nie tylko odznaczały się większą precyzją, ale również dawały większe możliwości – pracę na macierzach, gdzie elementem głównym były 0.

Problem przedstawiony na powyższej liście pozwala ponadto wyciągnąć następujące wnioski - nie każde zadanie da się wykonać korzystając ze standardowych rozwiązań czy też algorytmów. Nawet jeśli zwracają one poprawny wynik to być może nie w takim czasie lub używając takich zasobów jakie chcielibyśmy na nie przeznaczyć. Jednakże drobne modyfikacje – takie właśnie jak zmiana zakresów kilku pętli, które wymagały jedynie dokładnego prześledzenia struktury – pozwoliły na bardzo duże skoki wydajności.