Sprawozdanie

Współbieżna eliminacja Gaussa

Mateusz Nowak

1 Zapis notacji proplemu

Na wejściu dostajemy kwadratową macierz o rozmiarze N oraz wektor wyrazów wolnych o długości N. Dla uproszczenia macierz wraz z wektorem zapisujemy do jednej macierzy o rozmiarze Nx(N+1). Element leżący w i-tym wierszu oraz j-tej kolumnie oznaczamy jako $M_{i,j}$. Połączona macierz:

$$\begin{pmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,N} & M_{1,N+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ M_{N,1} & M_{N,2} & \cdots & M_{N,N} & M_{N,N+1} \end{pmatrix}$$

Wektor wyrazów wolnych to $\left\langle M_{1,N+1}, M_{2,N+1}, ..., M_{N,N+1} \right\rangle$

2 Algorytm

Przechodzimy po kolejnych kolumnach macierzy, aby wyzerować elementy znajdujące się pod przekątną.
Dla i-tej kolumny przechodzimy po kolejnych k-tych wierszach będących pod i-tym wierszem, aby wyzerować element $M_{k,i}$.
W tym celu wyznaczamy mnożnik.
Przemnażamy i-ty wiersz, a następnie odejmujemy go od k-tego wiersza.

3 Pseudo kod

for i in 1 to N-1:
    for k in i+1 to N:
        $m_{i,k} = M_{k,i}/M_{i,i}$
        for j in i to N+1:
            $p_{i,j,k} = m_{i,k} \cdot M_{i,j}$
            $M_{k,j} -= p_{i,j,k}$

4 Niepodzielne czynności wykonywane przez algorytm

Na podstawie powyższego pseudo kodu można wyznaczyć trzy klasy niepodzielnych zadań
$A_{i,k}$ - wyznaczenie $m_{i,k}$, czyli mnożnika dla i-tego wiersza, aby go odjąć od k-tego wiersza
$B_{i,j,k}$ - wymnożenie j-tego elementu i-tego wiersza przez mnożnik $m_{i,k}$
$C_{i,j,k}$ - odjęcie wartości obliczonej przez $B_{i,j,k}$ od elementu $M_{k,j}$
Na podstawie tych definicji od razu można zauważyć, że klasa B korzysta z wyniku A, a klasa C z wyniku B.

5 Alfabet w sensie teorii śladów

Alfabetem będzie zbiór wszystkich niepodzielnych czynności wykonywanych przez algorytm

$$\Sigma = \left\{ A_{i,k}, B_{i,j,k}, C_{i,j,k} \;:\; 1 \le i < N, i \le j \le N+1, i < k \le N \right\}$$

Ograniczenia:

1. i - numer aktualnie zerowanej kolumny, N-ta kolumna ma 0 elementów pod przekątną, więc już nic nie wyzerujemy
2. j - numer kolumny elementu, który jest mnożony lub odejmowany, na lewo od i-tej kolumny są już same 0, więc przyjmuje on pozostałe wartości
3. k - numer wiersza od którego odejmujemy i-ty wiersz, znajduje się poniżej i, czyli zaczyna się od i + 1 oraz idzie aż do N

6 Słowo w sensie teorii śladów

Dla problemu eliminacji Gaussa słowo będzie zawierało wszystkie elementy alfabetu dokładnie raz, a alfabet to zbiór wszystkich niepodzielnych czynności wykonywanych przez algorytm. Oznacza to, że aby otrzymać słowo wystarczy, zacząć z pustym słowem i wykonując kolejne kroki algorytmu dodawać na koniec słowa, aktualnie wykonywaną niepodzielną czynność. Modyfikując pseudo kod z punktu 3, otrzymujemy następujący generator:

```
ω = []
for i in 1 to N-1:
    for k in i+1 to N:
        ω.append (A_{i,k})
        for j in i to N+1:
            ω.append (B_{i,j,k})
            ω.append (C_{i,j,k})
return ω
```

7 Relacja zależności i niezależności

Na relację zależności składa się kilka zbiorów. Najprostrze wynikają z definicji niepodzielnych zadań algorytmu.
Klasa B korzysta z wyniku A:

$$D_1 = \left\{ \left(A_{i,k}, B_{i,j,k}\right) \;:\; A_{i,k}, B_{i,j,k} \in \Sigma \right\}$$

Klasa C korzysta z wyniku B:

$$D_2 = \left\{ \left(B_{i,j,k}, C_{i,j,k}\right) \;:\; B_{i,j,k}, C_{i,j,k} \in \Sigma \right\}$$

Zależności wynikające z dostępu do macierzy

Zadanie	Odczytuje	Modyfikuje
$A_{i,k}$	$M_{k,i}, M_{i,i}$	-
$B_{i,j,k}$	$M_{i,j}$	-
$C_{i,j,k}$	$M_{k,j}$	$M_{k,j}$

Tylko klasa C modyfikuje macierz, mamy więc następujące zależności

$$D_3 = \left\{ \left(C_{i,j,k}, A_{l,m} \right) \ : \ C_{i,j,k}, A_{l,m} \in \Sigma \land l = j \land (k = l \lor k = m) \land i < l \right\}$$

$$D_4 = \left\{ \left(C_{i,j,k}, B_{l,m,n} \right) \ : \ C_{i,j,k}, B_{l,m,n} \in \Sigma \land l = k \land m = j \land i < l \right\}$$

$$D_5 = \left\{ \left(C_{i,j,k}, C_{l,m,n} \right) \ : \ C_{i,j,k}, C_{l,m,n} \in \Sigma \land n = k \land m = j \land i < l \right\}$$

Na tak zdefiniowanych zbiorach można już zapisać relację zależności

$$D = sym\left((D_1 \cup D_2 \cup D_3 \cup D_4 \cup D_5)^+ \right) \cup I_\Sigma$$

Relacja niezależności to po prostu:

$$I = \Sigma^2 - D$$

8 Graf zależności Diekerta

Graf to zbiór wierzchołków oraz krawędzi $G = (V, E)$. Zbiór wierzchołków jest równy elementom słowa wejściowego, a w naszym przypadku te są równe alfabetowi, więc V = Σ. Natomiast zbiór krawędzi to zbiór bezpośrednich zależności. Należy więc ze zbiorów wyprowadzonych w poprzednim punkcie wyeliminować zależności pochodne.
W zbiorach D_1 i D_2 nie ma zależności pochodnych, składają się one z operacji wykonywanych podczas zerowania elementów pod przekątną i-tej kolumny. Można je potraktować jako podgraf i. Wiemy, że tylko klasa C modyfikuje elementy macierzy, a dokładniej element $M_{i+1,i}$ wyznacza lewy górny róg podmacierzy, której elementy będą modyfikowane w i-tej iteracji. W następnej iteracji elementy podmacierzy w kolumnie i, są już wyzerowane, więc nie mają kolejnych zależności, kolumna i+1 jest odczytywana przez klasę A, a wiersz i+1 przez klasę B, podmacierz zaczynająca się od $M_{i+2,i+1}$ jest modyfikowana przez klasę C. Jak widać każdy element podmacierzy i kończy zależność, albo jest bezpośrednio zależny w następnej iteracji, co oznacza, że wszyskie zależności z i do i+k, gdzie k > 1 to zależności pochodne.

D_1 i D_2 nie ma zależności pochodnych

$$E_1 = D_1$$

$$E_2 = D_2$$

D_3 tylko zależności z i do i+1

$$E_3 = \left\{ \left(C_{i,j,k}, A_{l,m}\right) \; : \; C_{i,j,k}, A_{l,m} \in \Sigma \land l = i + 1 \land l = j \land (k = l \lor k = m) \right\}$$

co można uprościć do

$$E_3 = \left\{ (C_{i,i+1,k}, A_{i+1,m}) \; : \; C_{i,j,k}, A_{i+1,m} \in \Sigma \land (k = i + 1 \lor k = m) \right\}$$

D_4 i D_5 tylko zależności z i do i+1 oraz bez zależności w i+1 kolumnie, bo te są zależna od A, więc była by to zależność przechodnia

$$E_4 = \left\{ \left(C_{i,j,k}, B_{l,m,n}\right) \; : \; C_{i,j,k}, B_{l,m,n} \in \Sigma \land l = i + 1 \land m \neq i + 1 l = k \land m = j \right\}$$

$$E_5 = \left\{ \left(C_{i,j,k}, C_{l,m,n}\right) \; : \; C_{i,j,k}, C_{l,m,n} \in \Sigma \land l = i + 1 \land m \neq i + 1 \land n = k \land m = j \right\}$$

co można uprościć do

$$E_4 = \left\{ \left(C_{i,j,i+1}, B_{i+1,j,n}\right) \; : \; C_{i,j,i+1}, B_{i+1,j,n} \in \Sigma \land j \neq i + 1 \right\}$$

$$E_5 = \left\{ \left(C_{i,j,k}, C_{i+1,j,k}\right) \; : \; C_{i,j,k}, C_{i+1,j,k} \in \Sigma \land j \neq i + 1 \right\}$$

Wobec tego zbiór krawędzi to

$$E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$$

9 Postać normalna Foaty

Jak wcześniej zauważyliśmy zadania są wykonywane iterując po kolejnych podmacierzach, gdzie zadania z podmiacierzy i+1 nie można wykonać wcześniej niż po i-tej iteracji. Każdą taką iteracja można podzielić na trzy klasy Foaty:

$$F_i = [A_{i,k}] \left[B_{i,j,k}\right] \left[C_{i,j,k}\right] \; : \; i \leq j \leq N + 1 \land i < k \leq N$$

$$FNF = F_1, F_2, ..., F_N$$

10 Implementacja w Javie

Cała logika rozwiązania problemu znajduje się w klasie o nazwie GaussElimination.
Klasa MatrixFileReader odczytuje i zapisuje dane do pliku.
Klasa Main to klasa uruchomieniowa, można podać argumenty jako [input file] [output file], bez podania drugiego argumentu domyśly plik wyjściowy to "output.txt", a wywołanie bez żadnych argumentów przyjmuje "input.txt" jako domyślny plik wejściowy.