Teoria Współbieżności

Ćwiczenie 7

Wiktor Satora 411502

11.01.2024

Zwielowątkowienie eliminacji Gaussa

1. Niepodzielne czynności wykonywane przez algorytm

W algorytmie rozróżnione są 3 rodzaje niepodzielnych operacji arytmetycznych:

A. Dzielenie i-tego elementu k-tego wiersza przez i-ty element i-tego wiersza

$A_{k,i}$ - wykonanie dzielenia $M_{k,i}/M_{i,i} \rightarrow m_{k,i}$

B. Mnożenie j-tego elementu i-tego wiersza przez i-ty element k-tego wiersza

$B_{k,j,i}$ - pomnożenie elementu $M_{i,j} * m_{k,i} \rightarrow n_{k,j,i}$

C. Odejmowanie j-tego elementu k-tego wiersza od j-tego element i-tego wiersza

$C_{k,j,i}$ – odjęcie $M_{k,j} - n_{k,j,i} \rightarrow M_{k,j}$

Gdzie: $M_{k,j}$ to pole macierzy o wierszu k i indeksie j,
m,n to zmienne pomocnicze do obliczeń z odpowiadającymi indeksami

2. Alfabet teorii śladów

$\Sigma = \{A_{k,i} \mid 1 \leq i < r,\ i < k \leq r\} \cup \{B_{k,j,i} \mid 1 \leq i < r,\ i < k \leq r,\ i \leq j \leq r+1\} \cup \{C_{k,j,i} \mid 1 \leq i < r,\ i < k \leq r,\ i \leq j \leq r+1\}$

Gdzie: r - rozmiar macierzy

3. Relacja zależności

- dzielenie i mnożenie, jeżeli dzielony i mnożony wiersz są takie same, lub obie operacje są wykonywane w celu odjęcia tego samego wiersza

  $\{(A_{k,i}, B_{k,j,i}) \mid B_{k,j,i}, A_{k,i} \in \Sigma\})$

- mnożenie i odejmowanie, jeżeli dotyczą tych samych wierszy, kolumn i wiersza, który ma być odjęty

  $\{(B_{k,j,i}, C_{k,j,i}) \mid B_{k,j,i}, C_{k,j,i} \in \Sigma\}$

- odejmowanie i dzielenie, jeżeli którykolwiek z wierszy w dzieleniu jest taki sam jak wiersz od którego odejmujemy oraz indeks kolumny odejmowanej jest taki sam jak indeks kolumny w dzieleniu

  $\{(C_{kc,jc,ic}, A_{ka,ia}) \mid C_{kc,jc,ic}, A_{ka,ia}\} \in \Sigma \wedge j_c = i_a \wedge (k_c = i_a \vee k_c = k_a)$

- odejmowanie i mnożenie, jeżeli indeksy kolumn są takie same i wiersz od którego odejmujemy jest taki sam jak wiersz przez który mnożymy

  $\{(C_{kc,j,ic}, B_{kb,j,ib}) \mid B_{kb,j,ib}, C_{kc,j,ic}\} \in \Sigma \wedge i_b = k_c$

- odejmowanie i odejmowanie, jeśli kolumny są takie same, wiersze od których odejmujemy są takie same i odejmowane wiersze różnią się o 1.

  $\{(C_{k,j,i1}, C_{k,j,i2}) \mid C_{k,j,i1}, C_{k,j,i2} \in \Sigma\}$

Relację niezależności wyznaczamy w oparciu o relację zależności:

$I = \Sigma^2 - D$

4. Opis algorytmu eliminacji Gaussa

Konstruujemy ciąg operacji odpowiadający algorytmowi eliminacji. Dla uproszczenia zdefiniujmy podciągi:

$p_{k,i}$ - odjęcie i-tego wiersza od k-tego skutkujące wyzerowaniem elementu $M_{k,i}$

$p_{k,i} = (A_{k,i}, B_{k,i,i}, C_{k,i,i}, B_{k,i+1,i}, C_{k,i+1,i}, \ldots, B_{k,r,i}, C_{k,r,i})$

Cały algorytm eliminacji (bez podstawiania wstecz) wyraża się więc poprzez:

$(p_{2,1}, p_{3,1}, \ldots, p_{r,1}, p_{3,2}, p_{4,2}, \ldots, p_{r,2}, \ldots, p_{r-1,r-2}, p_{r,r-2}, p_{r,r-1})$

5. Graf Diekerta

Zbiór wszystkich bezpośrednich zależności zależności E:
(krawędzie w grafie)

$E_1 = \{(A_{k,i}, B_{k,j,i}) \mid A_{k,i}, B_{k,j,i} \in \Sigma\}$

$E_2 = \{(B_{k,j,i}, C_{k,j,i}) \mid B_{k,j,i}, C_{k,j,i} \in \Sigma\}$

$E_3 = \{(C_{kc,j,ic}, B_{kb,j,ib}) \mid C_{kc,j,ic}, B_{kb,j,ib}\} \in \Sigma \wedge k_c = i_b \wedge i_c = i_{b-1} \wedge j \neq i_b\}$

$E_4 = \{(C_{k,j,i1}, C_{k,j,i2}) \mid C_{k,j,i1}, C_{k,j,i2} \in \Sigma \wedge i_1 = i_2 - 1 \wedge i_2 \neq j\}$

$E_5 = \{(C_{kc,j,ic}, A_{ka,ia}) \mid C_{kc,j,ic}, A_{ka,ia}\} \in \Sigma \wedge j = i_a \wedge i_c = i_{a-1} \wedge (k_c = k_a \vee k_c = i_a)\}$

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

6. Klasy Foaty

Przyjmując n=1,2,3,…,r-1 zdefiniujmy:

$F_{An} = \{A_{k,n} \mid n<k\leq r\}$

$F_{Bn} = \{B_{k,j,n} \mid n<k\leq r \land n<j\leq r+1\}$

$F_{Cn} = \{C_{k,j,n} \mid n<k\leq r \land n<j\leq r+1\}$

Wtedy wszystkie klasy Foaty we właściwym porządku to:

$[F_{A1}][F_{B1}][F_{C1}]\ [F_{A2}][F_{B2}][F_{C2}]\ \dots\ [F_{Ar-1}][F_{Br-1}][F_{Cr-1}]$

7. Zawartość projektu

- Makefile

Makefile służący do budowania napisanych w C programów

- in.txt, out.txt

Przykładowe wejście dla programu wykonującego eliminację Gaussa i oczekiwane wyjście

- gauss.c

Program implementujący wielowątkową eliminację Gaussa w sposób odzwierciedlający podział na klasy Foaty opisany wyżej, następnie realizuje jednowątkowy *backward substitution* i zapisuje obliczoną macierz do pliku matrix_out.txt

- diekert.c

Program generujący plik .dot dla graphviza opisujący graf Diekerta

8. Kompilacja i uruchomienie

- Program *gauss.c* możemy zbudować komendą: `make gauss`
- Program *diekert.c* możemy zbudować komendą: `make diekert`
- Uruchomienie obliczania macierzy odbywa się poprzez wpisanie `make generate` Program przeczyta dane z pliku in.txt, wykona obliczenia i wydrukuje je do matrix_out.txt

- Uruchomienie programu do generowania pliku .dot dla grafu diekerta:
  `make draw size=(rozmiar),` w miejsce ‘rozmiar’ podać rozmiar macierzy

  Plik .dot można wkleić do dowolnego wizualizatora online np.:
  https://dreampuf.github.io/GraphvizOnline/