Rozwiązywanie układów równań metodą Seidla

Mateusz Chłopek

Metoda Seidla

- 1. Otrzymanie danych wejściowych:

Macierz A – współczynniki

Wektor B – wyrazy wolne

Metoda Seidla

- 2. Przygotowanie danych pośrednich:

Macierz alfa:

$$alfa_{ij} = -A_{ij} / A_{ii}$$

$$alfa_{ii} = 0$$

Wektor Beta:

$$Beta_{ij} = B_{ij} / A_{ii}$$

Wektor X_{akt}:

$$X_{akt} = Beta$$

Metoda Seidla

- 3. Wyliczanie niewiadomych w kolejnych iteracjach pętli do-while

przypisanie $X_{pop} = X_{akt}$

przypisanie $X_{akt} = Beta$

obliczenie nowej wartości niewiadomych:

$$X_{akt_i} += alfa_{ij} \cdot X_{pop_j}$$

$$X_{akt_i} += alfa_{ij} \cdot X_{akt_j}$$

Metoda Seidla

Warunki pętli do-while:

liczba_iteracji < max_liczba_iteracji

precyzja > ε

W pętli do-while

inkrementacja licznika iteracji

obliczanie aktualnej precyzji:

precyzja += abs($X_{akt_i} - X_{pop_i}$)

precyzja /= liczba_niewiadomych

Problem metody Seidla

Metoda Seidla nie jest w stanie rozwiązać każdego układu równań liniowych.

Wymagane jest, aby elementy na głównej przekątnej [macierzy A] były silnie dominujące.

Pliki zawierające dane testowe zostały utworzone przy pomocy generatora, który tworzy układ równań rozwiązywalny metodą Seidla.

Wielowątkowość

Metoda Seidla nie jest najlepszym algorytmem dla wielowątkowości.

Problem występuje m. in. podczas wyliczania i-tej niewiadomej.

Rozwiązaniem problemu (poza zrezygnowaniem z wielowątkowości) jest wprowadzenie synchronizacji pracy wątków.

Zrealizowane jest to z wykorzystaniem tablicy booli, która informuje czy obliczenia dla danej niewiadomej zostały zakończone.

Przykład kodu w C++

```
12      for (int i = lowerBound; i < upperBound; i++) {
13          divisor = A[i][i];
14          for (int j = 0; j < variablesNumber; j++)
15              alfa[i][j] = -A[i][j] / divisor;
16
17          beta[i] = B[i] / divisor;
18          xNew[i] = beta[i];
19          alfa[i][i] = 0;
20
21          isReady[0][i] = true;
22      }
23
24      while (!boolCondition) {
25          boolCondition = true;
26          for (int i = 0; i < variablesNumber; i++)
27              if (!isReady[0][i])
28                  boolCondition = false;
29      }
30      boolCondition = false;
```

Przykład kodu w asm

```
    ;PĘTLA FOR
    ;PRZYPISANIE i = lowerBound
    MOV RAX, lowerBound
    MOV R8, RAX
LOOP1:
    ;PORÓWNANIE i < upperBound
    MOV RAX, R8
    CMP RAX, upperBound
    JE END1
    ;PRZYPISANIE divisor = A[i][i]
    MOV RCX, R8
    MOV RDX, QWORD PTR [Aaddr]
    MOV RAX, QWORD PTR [RDX + RCX * 8]
    MOVSS XMM0, DWORD PTR [RAX + RCX * 4]
    MOVSS XMM1, minus
    XORPS XMM0, XMM1
    MOVSS XMM2, XMM0
    ;ZAGNIEŻDŻONA PĘTLA FOR
    ;PRZYPISANIE j = 0
    MOV R9, 0
LOOP2:
    ;PORÓWNANIE j < variablesNumber
    MOV RAX, R9
    CMP RAX, variablesNumber
    JE END2
    ;A[i][j] -> XMM0
    MOV RAX, R8
    MOV RCX, R9
    MOV RDX, QWORD PTR [Aaddr]
    MOV RAX, QWORD PTR [RDX + RAX * 8]
    MOVSS XMM0, DWORD PTR [RAX + RCX * 4]
    ; XMM0 / XMM2
    DIVSS XMM0, XMM2
    ; alfa[i][j] = XMM0 (= -A[i][j] / XMM2)
    MOV RAX, R8
    MOV RCX, R9
    MOV RDX, QWORD PTR [alfa]
    MOV RAX, QWORD PTR [RDX + RAX * 8]
    MOVSS DWORD PTR [RAX + RCX * 4], XMM0
    INC R9
    JMP LOOP2
END2:
    ;beta[i] = B[i] / XMM2
    ;xNew[i] = beta[i]
    MOVSS XMM1, minus
    XORPS XMM2, XMM1
    MOV RAX, R8
    MOV RCX, QWORD PTR [beta]
    MOV RDX, QWORD PTR [Baddr]
    MOVSS XMM0, DWORD PTR [RDX + RAX * 4]
    DIVSS XMM0, XMM2
    MOVSS DWORD PTR [RCX + RAX * 4], XMM0
    MOV RCX, QWORD PTR [xNew]
    MOVSS DWORD PTR [RCX + RAX * 4], XMM0
    ;alfa[i][i] = 0
    MOV RAX, R8
    MOV RCX, R8
    MOV RDX, QWORD PTR [alfa]
    MOV RAX, QWORD PTR [RDX + RAX * 8]
    PXOR XMM0, XMM0
    MOVSS DWORD PTR [RAX + RCX * 4], XMM0

    ;isReady[0][i] = true
    MOV RAX, R8
    MOV RDX, QWORD PTR [isReady]
    MOV RDX, QWORD PTR [RDX + 0]
    MOV BYTE PTR [RDX + RAX], 1
    ;LOCK DEC boolCondition0
    INC R8
    JMP LOOP1
END1:
    ;synchronizacja wątków

SynchLoop0:
    MOV RAX, boolCondition
    CMP RAX, 1
    JE SynchLoop0End
    MOV boolCondition, 1
    MOV R8, 0
ForLoop0:
    MOV RAX, r8
    CMP RAX, variablesNumber
    JE ForLoop0End
    INC R8
    JMP ForLoop0
ForLoop0End:
    JMP SynchLoop0
SynchLoop0End:
    MOV boolCondition, 0
    MOV BYTE PTR [RDX + RAX], 1
    ;LOCK DEC boolCondition0
    INC R8
    JMP LOOP1
END1:
    ;synchronizacja wątków

SynchLoop0:
    MOV RAX, boolCondition
    CMP RAX, 1
    JE SynchLoop0End
    MOV boolCondition, 1
    MOV R8, 0
ForLoop0:
    MOV RAX, r8
    CMP RAX, variablesNumber
    JE ForLoop0End
    INC R8
    MOV RDX, QWORD PTR [isReady]
    MOV RDX, QWORD PTR [RDX + 0]
    MOV AL, BYTE PTR [RDX + RAX]
    CMP AL, 1
    JE ForLoop0
    MOV boolCondition, 0
ForLoop0End:
    JMP SynchLoop0
SynchLoop0End:
    MOV boolCondition, 0
```

Porównanie czasów dla 1 wątku

	C++	ASM
SMALL – 200 równań	2ms	2ms
MEDIUM – 1000 równań	16ms	13ms
LARGE – 7000 równań	850ms	550ms

Porównanie czasów dla 8 wątków

	C++	ASM
SMALL – 200 równań	27ms	26ms
MEDIUM – 1000 równań	72ms	50ms
LARGE – 7000 równań	1277ms	815ms

Porównanie czasów dla > 8 wątków

	C++	ASM
16 wątków	1.43s	1.21s
32 wątki	4,78s	3.77s
64 wątki	21.50s	17.03s

Podsumowanie

- Metoda Seidla nie nadaje się do obliczeń równoległych.
- Synchronizacja pracy wątków spowalnia je.
- Instrukcje wektorowe sprawiają, że funkcja asemblerowa działa szybciej niż ta napisana w C++ (przynajmniej w trybie debug).