Багатопотоковий алгоритм розв’язання системи лінійних рівнянь методом Гауса

Одним із найпоширеніших методів розв’язання систем лінійних рівнянь є метод Гауса.

У цьому методі для спрощення матриці, використовують послідовні елементарні операції перетворення матриці для модифікації матриці доки нижній лівий кут матриці не буде заповнено нулями, настільки наскільки це можливо.

Існує три типи елементарних перетворень матриці:

1) Заміна двох рядків,

2) Множення рядка на ненульове число,

3) Додавання одного рядка до іншого.

Використовуючи ці операції, матрицю завжди можна перетворити на верхню трикутну матрицю, а фактично і у скорочену рядкову ступінчасту форму.

Багатопотоковий алгоритм розв’язання системи лінійних рівнянь методом Гауса

Як тільки всі перші коефіцієнти (ті що знаходяться ліворуч і є ненульовими входженням в кожному рядку) стають рівними 1, і кожен стовпець, який містить перший ненульовий коефіцієнт має в усіх інших місцях нулі, тоді говорять, що матриця знаходиться у скороченій рядковій ступінчастій формі. Ця фінальна форма є унікальною; іншими словами, вона не залежить від того, яку послідовність перетворень буде здійснено. Цю послідовність перетворення матриці іноді називають спрощенням або скороченням матриці.
Якщо система лінійних рівнянь є невиродженою, то метод Гауса гарантує знаходження рішення з похибкою, яка визначається точністю машинних обчислень.

Багатопотоковий алгоритм розв’язання системи лінійних рівнянь методом Гауса

Для побудови паралельного алгоритму розв’язання системи лінійних рівнянь методом Гауса застосуємо стандарт OpenMP. Обчислювальна програма буде мати наступний вигляд:

```
#include <math.h> // підключаємо необхідні заголовні файли
#include <stdlib.h>
#include <stdio.h>
#ifdef _OPENMP
        #include <omp.h>
#endif
int N; // оголошуємо змінні
double *A;
#define A(i,j) A[(i)*(N+1)+(j)] // корисно в коді провести таку заміну на етапі прекомпіляції
double *X;
```

Багатопотоковий алгоритм розв’язання системи лінійних рівнянь методом Гауса

```
int main (){
        FILE *in;
        int i, j, k;
        in = fopen ("data.dat", "r"); // щоб пригадати, як провести в програмі читання з файлу, вимагаємо, щоб кількість
                                      рівнянь у системі вказувалося у файлі data.dat
        if (in == NULL) { printf ("Can not open 'data.dat' "); exit (1);} // перевіримо, чи існує такий файл
        i = fscanf (in, "%d", &N);
        if (i < 1) {printf("Wrong 'data.dat' (N ...)"); exit (2);} // перевіримо, чи вірно у файлі вказана кількість рівнянь
        A = (double *) malloc (N * (N + 1) * sizeof (double)); // створюємо масив коефіцієнтів
        X = (double *) malloc (N * sizeof (double)); // створюємо масив змінних
        printf ("GAUSS %dx%d\n----------------------------------\n", N, N);
```

Багатопотоковий алгоритм розв’язання системи лінійних рівнянь методом Гауса

```
#pragma omp parallel private (i, j, k) shared (A, X)  // відкриваємо паралельний блок з локальними змінними i, j, k та
                                                      спільними змінними A та X
{
        #pragma omp for
        for (i = 0; i <= N - 1; i++){ // цей цикл буде виконуватися паралельно
                for (j = 0; j <= N; j++){
                        if (i == j || j == N) A(i,j) = 1.; // ініціалізація масиву A
                        else A(i,j) = 1.1;
                }
        }
```

Багатопотоковий алгоритм розв’язання системи лінійних рівнянь методом Гауса

```
for (i = 0; i < N - 1; i++){
        #pragma omp for
        for (k = i + 1; k <= N - 1; k++){ // цей цикл буде виконуватися паралельно
                for (j = i + 1; j <= N; j++){
                        A(k,j) = A(k,j) - A(k,i) * A(i,j) / A(i,i); // приведення матриці до верхнього трикутного
                                                                    вигляду
                }
        }
        #pragma omp single
                X[N-1] = A (N - 1, N) / A(N - 1, N - 1); // визначення однієї з невідомих
        for (j=N-2; j>=0; j--){
```

Багатопотоковий алгоритм розв’язання системи лінійних рівнянь методом Гауса

```
                #pragma omp for
                for (k=0; k <= j; k++){ // цей цикл буде виконуватися паралельно
                        A(k,N) = A(k,N) - A(k, j + 1) * X[j + 1]; // переносимо доданки, значення яких вже
                                                                  визначили, до правої частини кожного з рівнянь
                }
                X[j] = A(j,N) / A(j,j); // визначення значень інших змінних
        }
    }
} // кінець паралельного блоку коду
```

Багатопотоковий алгоритм розв’язання системи лінійних рівнянь методом Гауса

```
#ifdef _OPENMP
        printf("Number of threads=%d\n",omp_get_max_threads());
#endif
printf ("\n X = (  ");
for (i = 0; i < N; i++){
        printf ("%.4g  ", X[i]); // виводимо на екран результат
}
printf (").\n");
return 0;
}
```