реалізація методу Гауса рішення лінійної системи рівнянь

Мета заняття

Проаналізувати можливість використання засобів побудови паралельних програм для збільшення продуктивності функцій, а саме:

- використання SIMD операцій;
- використання можливостей компілятора;
- використання Open MP;
- використання PPL

на прикладі рішення системи лінійних рівнянь.

Методичні вказівки до організації самостійної роботи

Вивчить алгоритм Гауса для рішення системи лінійних рівнянь [http://www.cleverstudents.ru/system_of_equations/solving_systems_Gauss_method.html].
Вивчить автоматичне будування засобів паралельної обробки, а саме SIMD операцій та потоків.
Вивчить можливості Вашого компілятора по використанню SIMD операцій при побудові коду. Для цього перевірте встановлене значення для properties→C++→Code generation→Enable Enhanced instruction Set. Задайте максимальне можливе значення, яке підтримує Ваш процесор.
Для визначення, чи вбудовані векторні операції, в properties→C++→CommandLine додайте додатковий параметр /Qvec-report:2 (В разі завдання / Qvec-report:1 видається інформація тільки про ті цикли, які виконуються паралельно, а в разі / Qvec-report:2 – про усі цикли. Якщо цикл не виконується паралельно, то вказуються причина, чому).
Для додавання потоків перевірте можливість встановлення в properties→C++→Code generation→Enable parallel Code Generation параметру /Qpar. Для визначення, чи вбудоване паралельне виконання циклу, в properties→C++→CommandLine додайте додатковий параметр /Qpar-report:2 (В разі завдання /Qpar-report:1 видається інформація тільки про ті цикли, які виконуються паралельно, а в разі /Qpar-report:2 – про усі цикли. Якщо цикл не виконується паралельно, то вказуються причина, чому).

Контрольні запитання і завдання

1 Чому алгоритм Гауса передбачає обрання головного елементу?
2 З чим пов’язані похибки при обчисленнях на комп’ютері, які типи похибок Ви знаєте і як оцінити значення похибок?
3 Які гілки першого проходу можна реалізувати паралельно?
4 Які гілки другого проходу можна реалізувати паралельно?
5 Які переваги та недоліки Open MP?

6 Які переваги та недоліки PPL?

Приклади аудиторних задач

Приклад 1.
Визначити основні функції для реалізації методу Гаусса
1 Перетворення лінійної системи рівнянь загального виду в лінійну систему рівнянь з трикутною матрицею лівих частин:
Для кожної колонки 2 та решти рівнянь
Визначення головного елементу;
Заміна порядку рівнянь;
Формування нульової колонки
2 Рішення системи рівнянь
Приклад 2
Реалізувати функцію визначення головного елементу. Визначити обчислювальну складність функції в залежності від n та i.

```
int GetMaxLine (double **a, int n, int i){
        int j;
        double max = fabs(a [i][i]);
        int nmax = i;
        for (j = i + 1; j < n; j++){
                double r = fabs (a [j][i]);
                if (r > max){
                        max = r;        nmax = j;
                }
        }
        return nmax;
    }
```

Приклад 3. Реалізувати функцію для заміни порядку рівнянь. Визначити обчислювальну складність функції в залежності від n та i.

```
// Послідовне виконання
void swap (double **a, double *b, int i, int j, int n){
        double temp;
        for (int k = i; k < n; k++){
                temp = a [i][k]; a [i][k] = a [j][k]; a [j][k] = temp;
        }
        temp = b [i]; b [i] = b [j]; b [j] = temp;
}
// Паралельне виконання. SIMD
void SSEswap (double **a, double *b, int i, int j, int n){
        double temp;
        __m128d  temp128;
        __m128d * pai = (__m128d *) a [i];
        __m128d * paj = (__m128d *) a [j];
        for (int k = i/2; k < n/2; k++) {
                temp128 = pai [k]; pai [k] = paj [k];  paj [k] = temp128;
        }
        temp = b [i]; b [i] = b [j]; b [j] = temp;
}
```

```
// Паралельне виконання. OpenMP
void ParSSEswap (double **a, double *b, int i, int j, int n){
		double temp;
		__m128d  temp128;
		__m128d * pai = (__m128d *) a [i];
		__m128d * paj = (__m128d *) a [j];
		#pragma omp parallel for if(i >= 512)
		//	#pragma omp parallel for
		for (int k = i/2; k < n/2; k++) {
			temp128 = pai [k]; pai [k] = paj [k];  paj [k] = temp128;
		}
		temp = b [i]; b [i] = b [j]; b [j] = temp;
}
```

Приклад 4. Реалізувати функцію для обнуління заданої колонки. Визначити обчислювальну складність функції в залежності від n та i.

```
void SetColZero (double **a, double *b, int i, int n){
	int j;
	for (j = i + 1; j < n; j++) {
		double temp = a [j][i] / a [i][i];
		for (int k = i; k < n; k++) a [j][k]-= a [i][k] * temp;
		b [j] -= b [i] * temp;
	}
	}
// Паралельне виконання. SIMD
// Функція SetColZero
void SSESetColZero (double **a, double *b, int i, int n){
	double temp;
	__m128d *pai = (__m128d *)a [i], * paj, temp128;
	for (int j = i + 1; j < n; j++){
		paj = (__m128d *)a [j];
		temp = a [j][i] / a [i][i]; temp128 = _mm_set1_pd (temp);
			for (int k = i/2; k < n/2; k++)
				paj[k] = _mm_sub_pd (paj[k], _mm_mul_pd (pai[k], temp128));
			b [j] -= b [i] * temp;
		}
}
// Паралельне виконання. Open MP
void ParSSESetColZero (double **a, double *b, int i, int n){
		__m128d *pai = (__m128d *)a [i], * paj;
		#pragma omp parallel for
		for (int j = i + 1; j < n; j++){
			double temp;   __m128d temp128;
			paj = (__m128d *)a [j];
			temp = a [j][i] / a [i][i];
			temp128 = _mm_set1_pd (temp);
			for (int k = i/2; k < n/2; k++)
				paj[k] = _mm_sub_pd (paj[k], _mm_mul_pd (pai[k], temp128));
			b [j] -= b [i] * temp;
		}
}
```

Приклад 5. Реалізувати функцію для перетворення системи рівнянь загального вигляду в трикутну систему. Визначити обчислювальну складність функції в залежності від n з урахуванням обчислювальної складності внутрішніх функцій.
Увага! Чи потрібно перевіряти необхідність виконання операції swap при виконанні функції Step1.

```
void Step1(double **a, double *b, double *x, int n){
        double temp;
        for (int i = 0; i < n - 1; i++)    {
                int j = GetMaxLine (a, n, i);
                swap (a, b, i, j, n);
                SetColZero (a, b, i, n);
        }
}
```

Паралельне виконання. SIMD

```
void SSEStep1 (double **a, double *b, double *x, int n){
                double temp;
                for (int i = 0; i < n - 1; i++)             {
                        int j = GetMaxLine (a, n, i);
                        SSEswap (a, b, i, j, n);
                        SSESetColZero (a, b, i, n);
                }
}
```

Паралельне виконання. Open MP

```
void ParSSEStep1 (double **a, double *b, double *x, int n){
                double temp;
                for (int i = 0; i < n - 1; i++)    {
                        int j = GetMaxLine (a, n, i);
                        ParSSEswap (a, b, i, j, n);
                        ParSSESetColZero (a, b, i, n);
                }
}
```

Приклад 6. Реалізувати функцію для обчислення коренів системи для трикутної системи. Визначити обчислювальну складність функції

```
void Step2 (double **a, double *b, double *x, int n){
        for (int i = n - 1; i >=0; i--){
                x [i]= b [i]/ a [i][i];
                for (int j = i - 1; j >= 0; j--){
                        b [j]-= a [j][i] * x [i];
                }
        }
}
```

Паралельне виконання. SIMD

```
void SSEStep2 (double **a, double *b, double *x, int n){
                double *ad = a [0];
                int i;
                for (i = 0; i < n-1; i++){
                        ad[i] = a [i][i];
                        for (int j = i + 1; j < n; j++)
                                a [j][i] = a [i][j];
                }
```

```
                ad [n-1] = a [i][i];
                __m128d* pb = (__m128d*)b, *pai;
                for ( i = n - 1; i >=0; i--){
                        pai = (__m128d*)a [i];  x [i]= b [i]/ ad[i];
                        __m128d x128 = _mm_set1_pd (x [i]);
                        for (int j = 0; j < (i + 1)/2; j++){
                                pb [j] = _mm_sub_pd (pb [j], _mm_mul_pd ( pai [j], x128));
                        }
                }
}
```

Паралельне виконання. Open MP

```
void ParSSEStep2 (double **a, double *b, double *x, int n){
        double *ad = a [0];
        for (int i = 0; i < n-1; i++){
                ad[i] = a [i][i];
                for (int j = i + 1; j < n; j++) a [j][i] = a [i][j];
        }
        ad [n-1] = a [n-1][n-1];
        __m128d* pb = (__m128d*)b, *pai;
for ( int i = n - 1; i >=0; i--){
                pai = (__m128d*)a [i]; x [i]= b [i]/ ad[i];
                __m128d x128 = _mm_set1_pd (x [i]);
                #pragma omp parallel for
                for (int j = 0; j < (i + 1)/2; j++){
                        pb [j] = _mm_sub_pd (pb [j], _mm_mul_pd ( pai [j], x128));
                }
        }
}
```

Приклад 7. Реалізувати функцію для обчислення коренів системи для лінійної системи загального вигляду. Визначити обчислювальну складність функції

```
void Eq (double **a, double *b, double *x, int n){
                Step1 (a, b, x, n);
                Step2 (a, b, x, n);
}
```

Паралельне виконання. SIMD

```
void SSEEq (double **a, double *b, double *x, int n){
                SSEStep1 (a, b, x, n);
                SSEStep2 (a, b, x, n);
}
```

Паралельне виконання. Open MP

```
void ParSSEEq (double **a, double *b, double *x, int n){
                ParSSEStep1 (a, b, x, n);
                ParSSEStep2 (a, b, x, n);
}
```

Приклад 8. Перетворити функцію Step2 з урахуванням властивостей кешу

```
void Step2 (double **a, double *b, double *x, int n){
                // Trans
                double *ad = a [0];
                for (i = 0; i < n-1; i++){
                        ad[i] = a [i][i];
```

```
                for (j = i + 1; j < n; j++)
                        a [j][i] = a [i][j];
        }
        ad [n-1] = a [i][i];
        // Step 2
        for ( i = n - 1; i >=0; i--){
                x [i]= b [i]/ ad[i];
                for (j = 0; j < i; j++){
                        b [j]-= a [i][j] * x [i];
                }
        }
}
```

Приклад 9

Реалізувати функцію SWAP за допомогою PPL

```
void PPLSSEswap (double **a, double *b, int i, int j, int n){
        double temp;
        __m128d * pai = (__m128d *) a [i];
        __m128d * paj = (__m128d *) a [j];
        parallel_for (i/2, n/2, [=](int k){
                __m128d  temp128;
                temp128 = pai [k]; pai [k] = paj [k]; paj [k] = temp128;
        });
        temp = b [i]; b [i] = b [j]; b [j] = temp;
}
```

Приклади домашніх задач

1 Реалізуйте усі розглянуті функції.

2 Для кожного варіанта визначте їх часові характеристики

3 Оберіть найбільш ефективний варіант для кожної функції

4 Визначте часові характеристики функції для рішення системи лінійних рівнянь

ПЕРЕЛІК ПОСИЛАНЬ

1 Качко О.Г. Паралельне програмування. – Х.:ХНУРЕ, 2016. – 400 с.

2 Качко Е.Г. Параллельное программирование. – Х.: Изд-во «Форт», 2011. – 528 с.

3 https://msdn.microsoft.com/en-us/library/dd492418.aspx

4 Бондаренко М.Ф., Качко О.Г. Операційні системи.– Х.: Компанія Сміт, 2008. – 432 с.

5 Качко Е.Г. Программирование на ассемблере. – Х.: ХНУРЭ, 2002. – 172 с.

6 Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.

7 Антонов А.С. Параллельное программирование с использованием технологи Open MP. – М.: Изд-во Московского университета, 2009-09-03.

8 Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows. – СПб.: Питер; М.: Издательско-торговый дом «Русская редакция», 2001. – 752 с.

9 Рихтер Дж., Кларк Дж. Программирование серверних приложений Microsoft для Windows 2000. Мастер класс. – СПб.: Питер; М.: Издательско-торговый дом «Русская редакция», 2001. – 592 с.