1 Введение

В этом эксперименте будет проведено сравнение двух алгоритмов, численно решающих задачу Дирихле для уравнения Пуассона, методом конечных разностей. Первый алгоритм работает в однопоточном режиме. Второй, это блочно-волновой алгоритм,параллельность которого реализована с помощью OpenMP. Оба алгоритма реализованы на языке Fortran.

1.1 Постановка задачи

В нашем примере рассматривается проблема численного решения задачи Дирихле для уравнения Пуассона, оределяемая как задача нахождения функции $u = u(x, y)$, удовлетворяющей в области определения D уравнению

$$\begin{cases} \frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} = f(x, y), & (x, y) \in D, \\ u(x, y) = g(x, y) & (x, y) \in D^0 \end{cases}$$

и принимающей значения $g(x, y)$ на границе D^0 области D (f и g являются функциями, задаваемыми при постановке задачи). Подобная модель может быть использована для описания установившегося течения жидкости, стационарных тепловых полей, процессов теплопередачи с внутренними источниками тепла и деформации упругих пластин и др. В качестве области задания D функции $u(x, y)$, будет использоваться еденичный квадрат

$$D = \left\{(x, y) \in D : 0 \leq x, y \leq 1\right\}.$$

Для решения задачи я использовал метод конечных разностей(метод сеток). Прямоугольная сетка в области D задана в виде

$$\begin{cases} D_h = \{(x_i, y_j) : x_i = ih, y_j = jh, 0 \leq i, j \leq N + 1, \} \\ h = 1/(N + 1) \end{cases}$$

где величина N задает количество узлов по каждой из координат области D .

Обозначим оцениваемую при подобном дискретном представлении аппроксимацию функции u(x, y) в точках (x_i, y_j) через u_{ij} . Тогда, используя пятиточечный шаблон для вычисления значений производных, мы можем представить уравнение Пуассона в конечно-разностной форме

$$\frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}}{h^2} = f_{i,j}.$$

Данное уравнение может быть разрешено относительно u_{ij}

$$u_{ij} = 0.25(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 f_{ij}).$$

Разностное уравнение, записанное в подобной форме, позволяет определять значение u_{ij} по известным значениям функции u(x, y) в соседних узлах используемого шаблона. Данный результат служит основой для построения различных итерационных схем решения задачи Дирихле, в которых в начале вычислений формируется некоторое приближение для значений , а затем эти значения последовательно уточняются в соответствии с приведенным соотношением.

Выполнение итераций обычно продолжается до тех пор, пока получаемые в результате итераций изменения значений u_{ij} не станут меньше некоторой заданной величины (требуемой точности вычислений. Обознычена как eps и в нашем эксперименте равна 0.1).

2 Об алгоритмах

2.1 Источник

Реализации алгоритмов выполнены с помощью книги В.П. Гергеля "Высокопроизводительные вычисления для многоядерных многопроцессорных систем". Последовательный алгоритм Гаусса-Зейделя расссмотрен в пункте 11.1, а алгоритм, который использует блочный подход к методу волновой обработки данных рассмотрен в пункте 11.2.6.

2.2 Корректность алгоритмов

Параллельный блочно-волновой алгоритм под номером 11.6 основан на основан на алгоритме 11.1. Алгоритм 11.6 выполняет те же самые вычисления, что и обычный алгоритм, основанный на методе конечных разностей. Отличие заключается в том, что он делает это в ином порядке. Я запустил эксперименти увидел, что выводы этих алгоритмов полностью совпадают до 10 знака после запятой.

Более подробно содержание этих алгоритмов описано в главе 11.

3 Эксперимент

3.1 Железо

Процессор: AMD Ryzen 5 3500U 4 ядра, 8 логических процессов Кэш первого уровня : 384КБ Кэш второго уровня : 2МБ Кэш третьего уровня : 4МБ

Версия ОС: Ubuntu 20.04.1 LTS

Версия компилятора: GNU Fortran 9.3.0

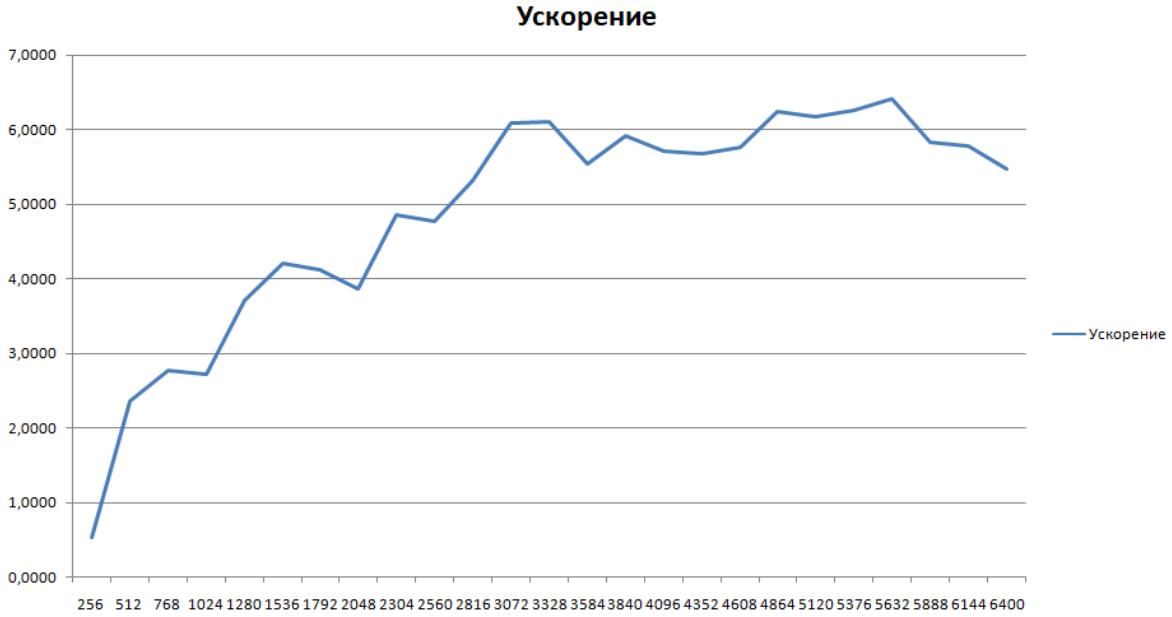
3.2 Задача

Сравним скорость алгоритмов на задаче дирихле для уравнения Лапласа (частный Случай уравнения Пуассона). Зададим граничные условия:

$$\begin{cases} f(x,y) = 0, & (x,y) \in D \\ 100 - 200x, & y = 0, \\ 100 - 200y, & x = 0, \\ -100 + 200x, & y = 1, \\ -100 + 200y, & x = 1 \end{cases}$$

Возьмем eps = 0,1 и инициализируем матрицу нулями.

Номер	N	Время параллельного	Время последовательного	Ускорение
1	256	0,1921	0,1027	0,5343
2	512	0,2108	0,4964	2,3550
3	768	0,4271	1,1806	2,7644
4	1024	1,0369	2,8239	2,7235
5	1280	1,3419	4,9871	3,7163
6	1536	1,7579	7,3949	4,2066
7	1792	2,2935	9,4501	4,1204
8	2048	3,3335	12,8817	3,8644
9	2304	3,6854	17,8799	4,8515
10	2560	5,0342	23,9959	4,7666
11	2816	5,4207	28,8036	5,3136
12	3072	5,6342	34,2647	6,0816
13	3328	6,6967	40,8499	6,1000
14	3584	8,5773	47,4241	5,5290
15	3840	9,1622	54,1053	5,9053
16	4096	10,9975	62,8002	5,7104
17	4352	12,3561	70,1429	5,6768
18	4608	13,5661	78,1949	5,7640
19	4864	14,4099	89,7725	6,2299
20	5120	16,6045	102,3978	6,1669
21	5376	18,0225	112,8388	6,2610
22	5632	19,6375	125,8087	6,4066
23	5888	23,8371	138,9360	5,8286
24	6144	26,2946	151,7478	5,7711
25	6400	28,9765	158,3428	5,4645



Удалось достигнуть ускорения в 6.4 раза. Заметим, что при малом количестве узлов сети последовательный алгоритм обгоняет многопоточный. Это происходит из-за того, что организация параллелизма требует много времени, относительно

времени вычислений. Параллельные алгоритмы становятся эффективны при N >=512.

Также проверим какой размер блока наиболее подходит для параллельного алгоритма.Для этой проверки N = 4096

3.3 Другая задачи

Изменим гранмчные условия.

$$\begin{cases} f(x,y) = 10(x+y), & (x,y) \in D \\ 100 - 200x, & y = 0, \\ 100 - 200y, & x = 0, \\ -100 + 200x, & y = 1, \\ -100 + 200y, & x = 1 \end{cases}$$

Видно, что при разных граничных условиях графики временни параллельного алгоритма, относительно последовательного, практически не отличаются.

3.4 Подбор размера блока

Размер блока	Время работы
2	22,13913
4	16,42399
8	12,29849
16	10,53916
32	10,92360
64	10,63420
128	12,13523
256	13,11786
512	18,89232

Для работы алгоритма 11.6 на моем устройстве лучше всего подкодят блоки размером 16, 32 и 64.