Московский Государственный Университет имени
М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по теоретическому заданию в рамках курса «Суперкомпьютерное моделирование и технологии» Двумерная задача Дирихле для уравнения Пуассона в криволинейной области

Муравицкая Екатерина Ярославовна
616 группа
Вариант 6

Москва 2024

Содержание

1 Введение

Задача представляет вычисление приближенного решения двумерной задачи Дирихле для уравнения Пуассона для области, которая является квадратом с отсеченной вершиной, соответствующий системе неравенств: (x, y) : $|x| + |y| < 2, y < 1$.

Разработан параллельный код программы с использованием библиотек OpenMP и MPI. Задание выполнено на языке C на ПВС Московского университета IBM Polus.

2 Математическая постановка задачи

В области $D \subset R^2$ ограниченной контуром γ рассматривается дифференциальное уравнение Пуассона:

$$-\Delta u = f(x, y)$$

в котором оператор Лапласа

$$\Delta = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

Функция $f(x, y)$ известна и равна 1 в области D. Для выделения единственного решения дополняется граничным условием Дирихле:

$$u(x, y) = 0, (x, y) \in \gamma$$

Область D является квадратом с отсеченной вершиной, соответствующий системе неравенств: $(x, y) : |x| + |y| < 2, y < 1$.

3 Метод фиктивных областей

Для приближенного решения задачи предлагается воспользоваться методом фиктивных областей. Пусть область D принадлежит прямоугольнику $\Pi = \{(x, y) : A_1 < x < B_1, A_2 < y < B_2\}$.

Для данной области D возьмем прямоугольник, такой что: $\Pi = \{(x, y) : -2 < x < 2, -2 < y < 1\}$.

Обозначим через $\overline{D}, \overline{\Pi}$ замкание области D и прямоугольника Π соответственно, через Γ - границу прямоугольника. Разность множеств

$$\hat{D} = \Pi \backslash D$$

называется фиктивной областью. Выберем и зафиксируем малое $\epsilon > 0$.

В прямоугольнике Π рассматривается задача Дирихле:

$$-\frac{\partial}{\partial x}(k(x, y)\frac{\partial v}{\partial x}) - \frac{\partial}{\partial y}(k(x, y)\frac{\partial v}{\partial y}) = F(x, y)$$

$$v(x,y) = 0, (x,y) \in \Gamma$$

с кусочно-постоянным коэффициентом

$$k(x,y) = \begin{bmatrix} 1, (x,y) \in D, \\ 1/\epsilon, (x,y) \in \hat{D} \end{bmatrix}$$

и правой частью

$$F(x,y) = \begin{bmatrix} f(x,y), (x,y) \in D \\ 0, (x,y) \in \hat{D} \end{bmatrix}$$

Требуется найти непрерывную в $\overline{\Pi}$ функцию $v(x,y)$, удовлетворяющую дифференциальному уравнению задачи всюду в $\Pi \backslash \gamma$, равную нулю на границе Γ прямоугольника, и такую, чтобы вектор потока

$$W(x,y) = -k(x,y)(\frac{\partial x}{\partial y}, \frac{\partial v}{\partial y})$$

имел непрерывную нормальную компоненту на общей части криволинейной границы области D и прямоугольника Π.

4 Разностная схема решения задачи

Краевую задачу предлагается решать численно методо конечных разностей. В замыкании прямоугольника $\overline{\Pi}$ определяется равномерная прямоугольная сетка $\overline{\omega}_h = \overline{\omega}_1 * \overline{\omega}_2$, где

$$\overline{\omega}_1 = \{x_i = A_1 + ih_1, i = \overline{0, M}\}, \overline{\omega}_2 = \{y_j = A_2 + jh_2, j = \overline{0, N}\}$$

, где

$$h_1 = (B_1 - A_1)/M, h_2 = (B_2 - A_2)/N.$$

Через ω_h обозначим множество внутренних узлов сетки $\overline{\omega}_h$, т.е. множество узлов сетки прямоугольника, не лежащих на границе $Gamma$.

Дифференциальное уравнение задачи во всех внутренних узлах точках сетки аппроксимируется разностным уравнением

$$-\frac{1}{h_1}(a_{i+1j}\frac{w_{i+1j} - w_{ij}}{h_1} - a_{ij}\frac{w_{ij} - w_{i-1j}}{h_1}) - \frac{1}{h_2}(b_{ij+1}\frac{w_{ij+1} - w_{ij}}{h_2} - b_{ij}\frac{w_{ij} - w_{ij-1}}{h_2}) = F_{ij}, (*)$$

$i = \overline{1, M-1}, j = \overline{1, N-1}$ в котором коэффициенты

$$a_{ij} = \frac{1}{h_2}\int_{y_j - 1/2}^{y_j + 1/2} k(x_{i-1/2}, t)dt, b_{ij} = \frac{1}{h_1}\int_{x_i - 1/2}^{x_i + 1/2} k(t, y_{j-1/2})dt$$

при всех $i = \overline{1, M}, j = \overline{1, N}$. Здесь полуцелые узлы

$$x_{i \pm 1/2} = x_i \pm 0.5h_1, y_{j \pm 1/2} = y_j \pm 0.5h_2$$

Правая часть разностного уравнения

$$F_{ij} = \frac{1}{h_1 h_2} \iint_{\Pi_{ij}} F(x,y)dxdy,$$

$$\Pi_{ij} = \{(x,y) : x_{i-1/2} <= x <= x_{i+1/2}, y_{j-1/2} <= y <= y_{j+1/2}$$

при всех $i = \overline{1,M}, j = \overline{1,N}$.

5 Вычисление коэффициентов

Из замечания в описании задания следует аналитический алгорит вычисления коэффициентов a_{ij}, b_{ij}, F_{ij}.

При вычислении a_{ij} необходимо посчитать интеграл $\int_{y_j-1/2}^{y_j+1/2} k(x_{i-1/2}, t)dt$. По построению прямоугольника П, легко видеть, что этот интеграл равен длине отрезка $[y_j - 1/2, y_j + 1/2]$ внутри области D (квадрат с отсеченной вершиной) + $1/\epsilon$ * длину вне области D.

Для этого необходимо определить точку пересечения (если она есть) данного отрезка с одной из боковых граней фигуры, затем вычислить длину вне и внутри фигуры. Аналогично для b_{ij}.

Для вычисления F_{ij} надо посчитать интеграл $\iint_{\Pi_{ij}} F(x,y)dxdy$. Также из замечания следует, что этот интеграл равен площади прямоугольника $\Pi_{ij} = \{(x,y) : x_{i-1/2} \le x \le x_{i+1/2}, y_{j-1/2} \le y \le y_{j+1/2}$ внутри квадрата с отсеченной вершиной. Для этого необходимо найти пересечение (если оно есть) прямоугольника с боковой стороной фигуры и посчитать площадь образованной трапеции, треугольника или пятигранника/шестигранника внутри исходной фигуры.

Для данных алгоритмов были найдены уравнения боковых сторон, левая-верхняя, правая-верхняя, левая-нижняя и правая-нижняя соответсвенно:

$$y = 2 + x,$$

$$y = 2 - x,$$

$$y = -2 - x,$$

$$y = -2 + x,$$

с помощью которых находятся пересечения боковых сторон с вертикальными или горизонтальными линиями.

6 Метод решения системы линейных алгебраических уравнений

Приближенное решение разностной схемы может быть получено итерационным методом скорейшего спуска. Этот метод позволяет получить последовательность сеточных функций $w^{(k)} \in H, k = 1, 2, \ldots,$ сходящуюся по норме пространства H к решения разностной схемы.

Метод является одношаговым. Итерация $w^{(k+1)}$ вычисляется по итерации $w^{(k)}$ согласно равенствам:

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} - \tau_{k+1} r_{ij}^{(k)},$$

где невязка $r^{(k)} = Aw^k - B$, итерационный параметр

$$\tau_{k+1} = \frac{\left(r^{(k)}, r^{(k)}\right)}{\left(Ar^{(k)}, r^{(k)}\right)}.$$

В качестве условия остановки алгоритма было использовано неравенство

$$||w^{(k+1)} - w^{(k)}|| < \delta,$$

где δ - положительное число, определяющее точность итерационного метода.

7 Использование OpenMP

Решение задачи было написано на языке программирования C, используя распараллеливание с помощью библиотеки OpenMP.

Количество нитей задается через параметр OMP_NUM_THREADS при запуске программы. Например, команда gcc main.c -lm -fopenmp -o a.out && bsub -q normal -W 0:30 -o result -e resultErr OMP_NUM_THREADS=2 ./a.out - скомпилирует и запустит испольняемый файл a.out, в котором будет проведен эксперимент с 2 нитями. Размер сетки и точность алгоритма задается в коде программы (переменные M, N, delta соответственно).

Общий алгоритм работы программы следующий:

1. инициализация нужных матриц и переменных;
2. подсчет матриц коэффициентов a_{ij}, b_{ij}, F_{ij};
3. вычисление приближенного решения w методом скорейшего спуска;
4. подсчет итерационного параметра τ и нормы $||w^{(k+1)} - w^{(k)}||$;
5. если норма больше заданной точности, возвращаемся на пункт 3.

При завершении очередного эксперимента программа выводит число итераций и затраченное время. После завершения последнего эксперимента программа завершается.

В основном цикле метода скорейшего спуска написаны три двойных цикла for, итерирующиеся по i и j. Первый блок циклов вычисляет невязку r, второй блок циклов вычисляет произведение матрицы Ar (с помощью подставления r вместо w в уравнение), третий блок считает норму $||w^{(k+1)} - w^{(k)}||$ для проверки условия остановки. Все три блока циклов были распаралеллены с помощью директивы #pragma omp parallel for collapse(2) для развертывания вложенного цикла с указанием приватных и разделяемых переменных, а так же reduction для правильного расчета суммы внутри.

8 Использование MPI

Решение задачи было написано на языке программирования C, используя распараллеливание с помощью библиотеки MPI.

Для запуска программы необходимо в параметрах командной строки указать требуемое число процессов.

Компиляция программы происходит с помощью команды: mpixlc -lm main.c.

Командная строка mpisubmit.pl -n 2 ./a.out 2 запустит программу с 2 процессами. Для последовательного исполнения: mpisubmit.pl ./a.out 1.

Либо bsub -o 40x40.out -e 40x40.err -R "affinity[core(K)]" -n K -m "polus-c4-ib"mpiexec -n K ./a.out, где 'K' - количество процессов.

Размер сетки и точность алгоритма задается в коде программы (переменные M, N, delta соответственно).

Общий алгоритм работы программы следующий:

1. инициализация нужных матриц и переменных;
2. подсчет матриц коэффициентов a_{ij}, b_{ij}, F_{ij};
3. вычисление приближенного решения w методом скорейшего спуска;
   (a) подсчет r
   (b) отправка граничных значений r
   (c) получение граничных значений r
   (d) подсчет итерационного параметра τ
   (e) подсчет w
   (f) отправка граничных значений w
   (g) получение граничных значений w
4. подсчет нормы как максимум из всех $|r_{ij}|$;
5. если норма больше заданной точности, возвращаемся на пункт 3.

При завершении программа выводит затраченное время и число итераций.

Суть распараллеливания вычислений заключается в разбиении области на n число подобластей, где n соответствует числу процессов. В результате чего n областей считаются одновременно.

Мы выбираем области так, чтобы они делили сетку пополам, но при этом захватывали доп. строку/столбец из смежной области. Таким образом, для сетки 40x40 и для 2 процессов мы разделим область горизонтально с 0 по 20 индекс и с 19 по 39 индекс матрицы.

Данный массив, состоящий из строки/столбца из соседней области, будет передаваться между областями с помощью средств MPI.

9 Использование OpenMP + MPI

Решение задачи было написано на языке программирования C, используя распараллеливание с помощью библиотек OpenMP и MPI.

Для запуска программы необходимо в параметрах командной строки указать требуемое число процессов и нитей, либо запустить через makefile с вызовом соответсвующей команды.

Компиляция программы происходит через команду module avail && module load SpectrumMPI/10.1.0 && mpicc main.c -lm -fopenmp.

Запуск программы происходит через команду bsub -n M -o a.out -eo a.err -m "polus-c2-ib polus-c3-ibR "span[ptile=M/2] affinity[thread(K,same=core)*M]"OMP_-NUM_THREADS=K mpiexec -n M ./a.out, где 'M' - число процессов, 'K' - число нитей. Для одного процесса span[ptile] не указывается.

Размер сетки и точность алгоритма задается в коде программы (переменные M, N, delta соответственно).

Общий алгоритм программы повторяет алгоритмы для OpenMP и MPI программ. В строках pragma указаны внешними только те переменные, которые не зависят от очередности подсчета на другом процессе.

10 Результаты работы

10.1 Последовательная реализация

Разработан код программы, вычисляющий приближенное решение разностной схемы скорейшего спуска. Выполнены расчеты на сгущающихся сетках (M,N) = (10,10), (20,20), (40,40) при точности равной $1e-7$. Результаты приведены в таблице:

Число точек сетки (M × N)	Число итераций	Время решения (с)
10 × 10	655	0.006394
20 × 20	9353	0.525364
40 × 40	112151	22.543114

Таблица 1. Таблица с результатами расчётов последовательного кода

10.2 OpenMP-программа

Были проведены расчеты с сеткой размера (40x40) для 1, 2, 4 и 16 нитях. При использовании OpenMP во всех экспериментах точность взята равной $1e-7$. Результаты приведены в таблице:

Количество OpenMP-нитей	Число точек сетки (M × N)	Число итераций	Время решения (с)	Ускорение
1	40 × 40	112151	22.543114	1
2	40 × 40	112151	12.35378	1.82479
4	40 × 40	112151	9.078966	2.483
16	40 × 40	112151	7.123714	3.16452

Таблица 2. Таблица с результатами расчётов OpenMP на сетке (40x40)

Были проведены расчеты с разными размерами сетки. При использовании OpenMP во всех экспериментах точность взята равной $1e-7$. Результаты приведены в таблице:

Количество OpenMP-нитей	Число точек сетки (M × N)	Число итераций	Время решения (с)	Ускорение
2	80 × 90	1169687	785.909590	1.458028
4	80 × 90	1169687	625.576968	1.831715
8	80 × 90	1169687	423.302507	2.706997
16	80 × 90	1169687	274.971049	4.167271
4	160 × 180	3336901	5850.279298	1.846065
8	160 × 180	3336901	4355.450176	2.479651
16	160 × 180	3336901	1679.503207	6.430472
32	160 × 180	3336901	1054.274152	10.244014

Таблица 3. Таблица с результатами расчётов на ПВС IBM Polus (OpenMP код)

Так же были проведены расчеты ошибок по итерациям и по области. Графики ошибок по итерациям приведены ниже:

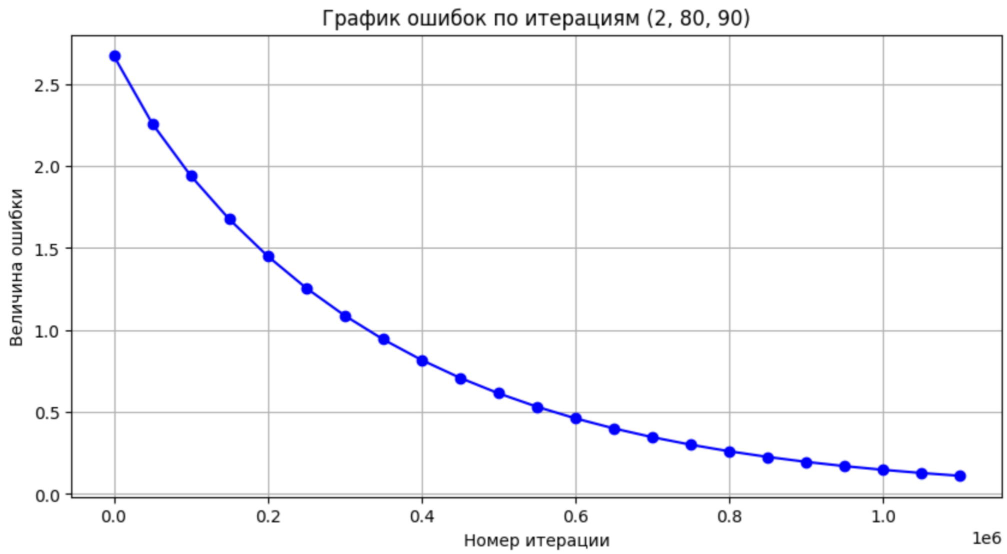


График ошибок по итерациям (2, 80, 90)
Величина ошибки
Номер итерации
1e6
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.0
1.5
2.0
2.5

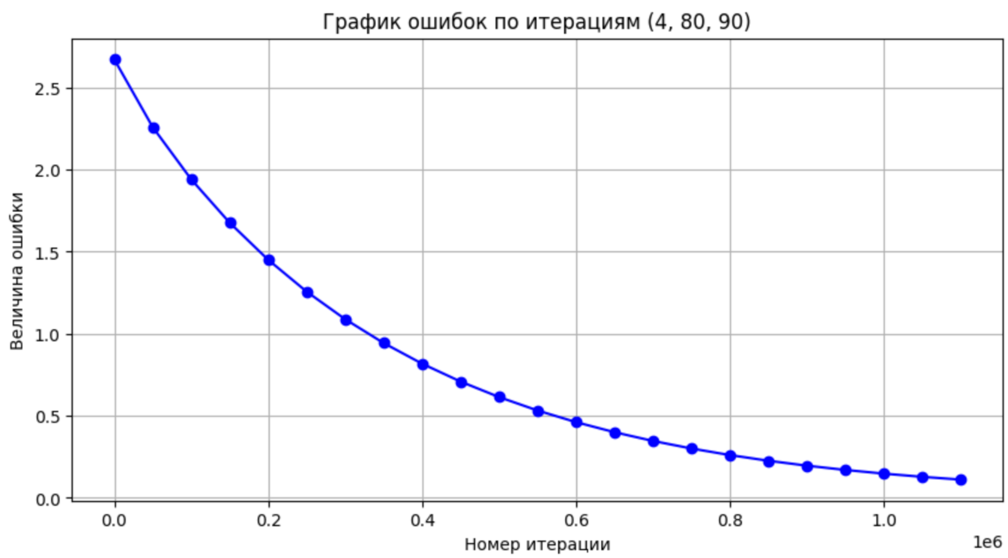
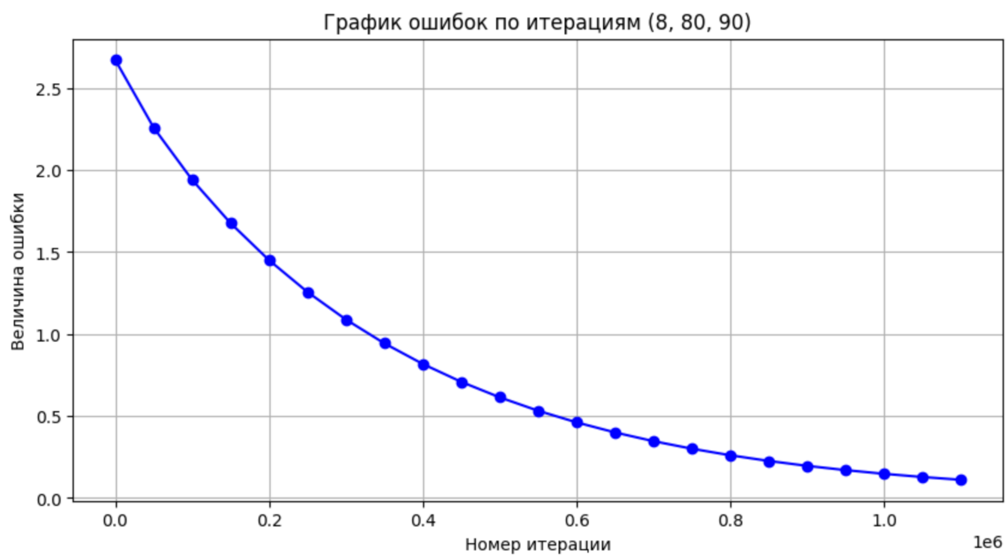
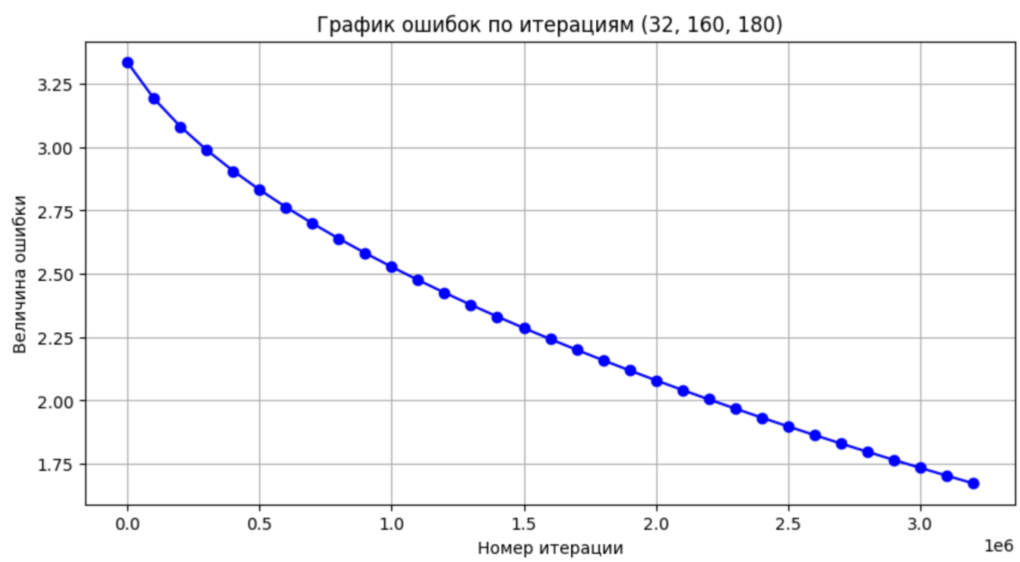


График ошибок по итерациям (4, 80, 90)
Величина ошибки
Номер итерации
1e6
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.0
1.5
2.0
2.5



Таким образом, график ошибки для разного количества нитей одинаковый. Построим график для сетки (160x180):



10.3 MPI-программа

Были проведены расчеты для алгоритма MPI с сеткой размера (40x40) для 1, 2 и 4 процессов. Во всех экспериментах точность взята равной $1e-6$. Результаты приведены в таблице:

Количество процессов	Число точек сетки (M × N)	Число итераций	Время решения (с)	Ускорение
1	40 × 40	267272	34.196799	1
2	40 × 40	267272	18.806876	1.818313
4	40 × 40	267272	11.166045	3.062569

Таблица 4. Таблица с результатами расчётов MPI на сетке (40x40)

10.4 Гибридная реализация MPI/OpenMP

Были проведены расчеты для гибридного алгоритма MPI/OpenMP с сеткой размера (40x40) для 1 и 2 процессов с 4 нитями, а также для последовательного выполнения. Во всех экспериментах точность взята равной $1e-6$. Результаты приведены в таблице:

Количество процессов	Количество OpenMP-нитей	Число точек сетки (M × N)	Число итераций	Время решения (с)	Ускорение
1	1	40 × 40	268000	21.625425	1
1	4	40 × 40	268000	20.573502	1.05113
2	4	40 × 40	268000	13.903866	1.55535

Таблица 5. Таблица с результатами расчётов гибридной реализации на сетке (40x40)

Кроме того, были проведены расчеты для гибридного алгоритма MPI/OpenMP с сеткой размера (80x90) и (160x180). В экспериментах с сеткой (80x90) точность взята равной $1e-5$. Результаты приведены в таблице:

Количество процессов	Количество OpenMP-нитей	Число точек сетки (M × N)	Число итераций	Время решения (с)	Ускорение
2	1	80 × 90	4420000	1547.908524	1.08156
2	2	80 × 90	4420000	1542.942087	1.08504
2	4	80 × 90	4420000	1008.669692	1.65977
2	8	80 × 90	4420000	554.788852	3.01764
4	1	160 × 180	32773000	10672.136474	1.34931
4	2	160 × 180	32773000	10652.510862	1.35179
4	4	160 × 180	32773000	7521.163959	1.9146
4	8	160 × 180	32773000	5612.745985	2.56559

Таблица 6. Таблица с результатами расчётов на ПВС IBM Polus (MPI+OpenMP код)

10.5 Графики реализаций

Рисунок приближенного решения для сетки 40 на 40:

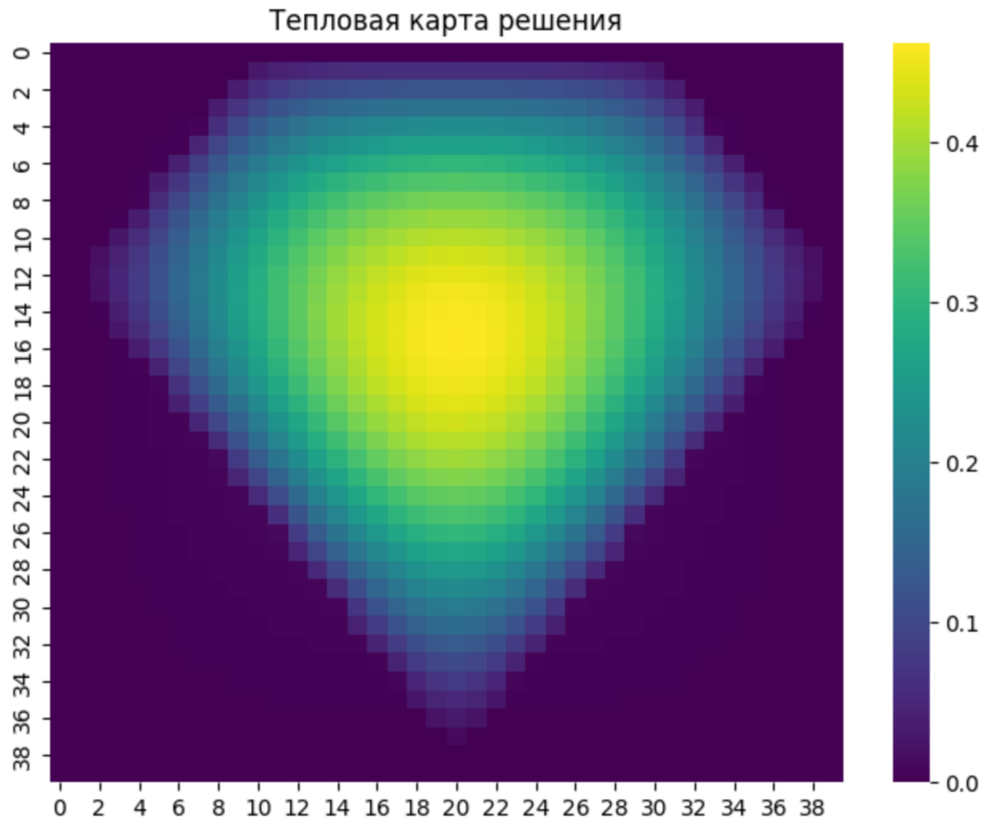
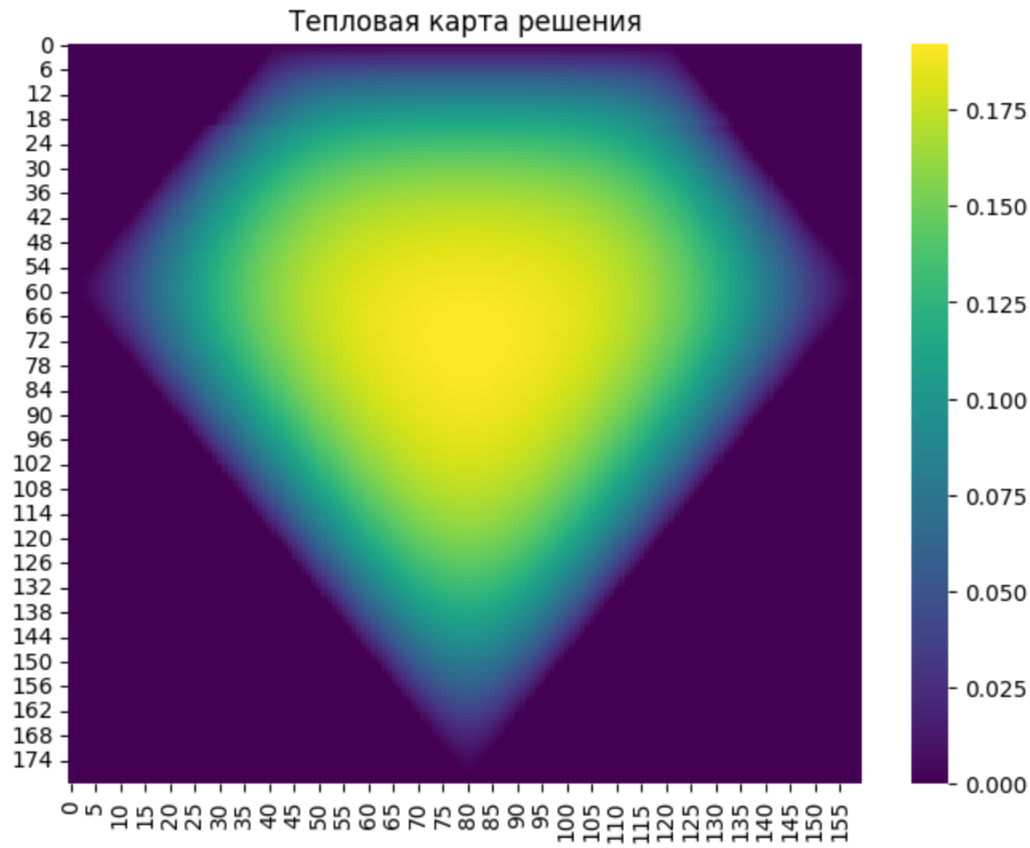
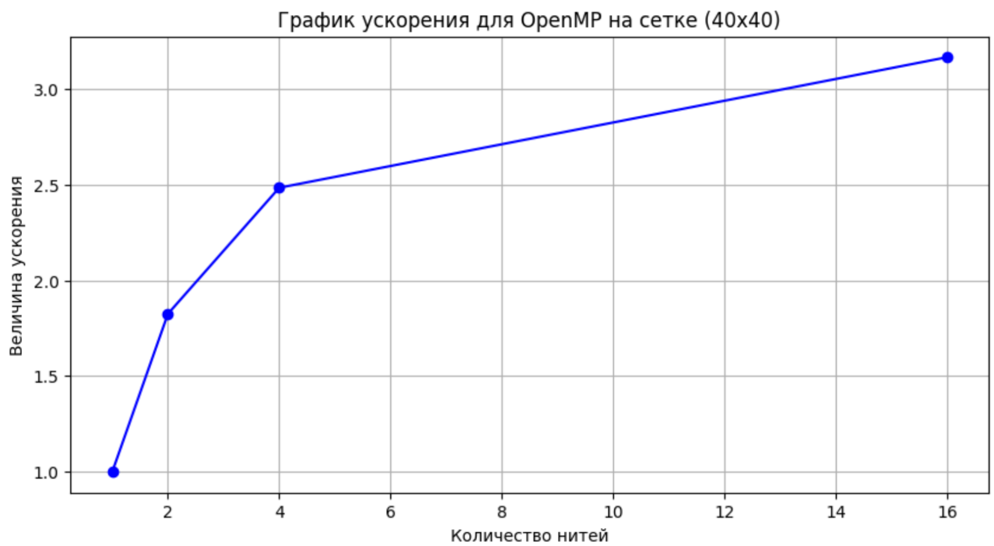
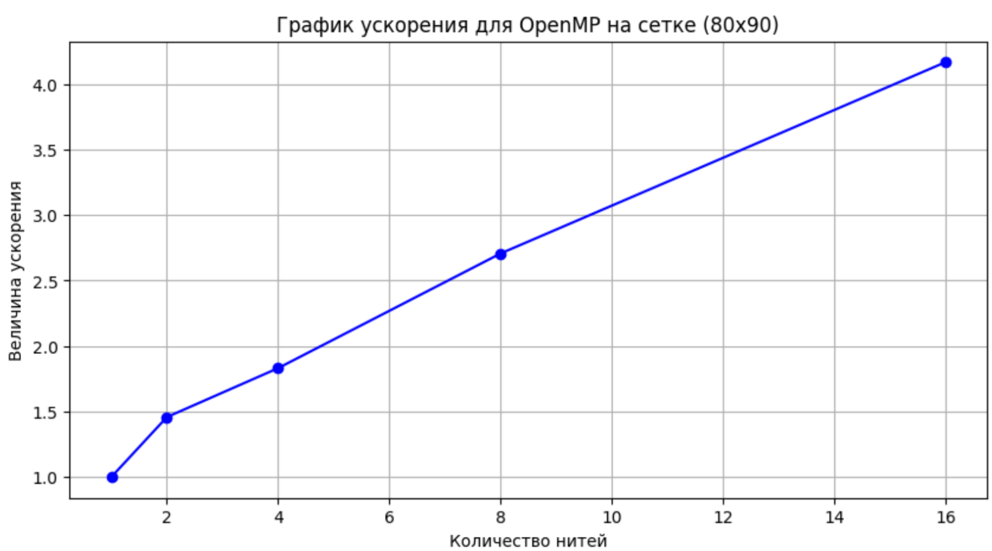


Рисунок приближенного решения для сетки 160 на 180:



Графики ускорений:





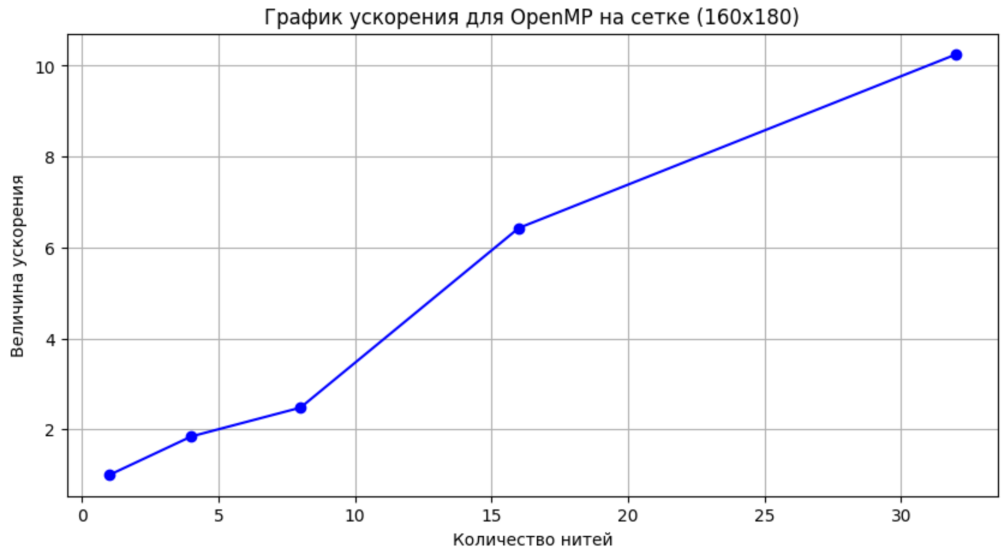


График ускорения для OpenMP на сетке (160x180)
Величина ускорения
Количество нитей
0
5
10
15
20
25
30
2
4
6
8
10

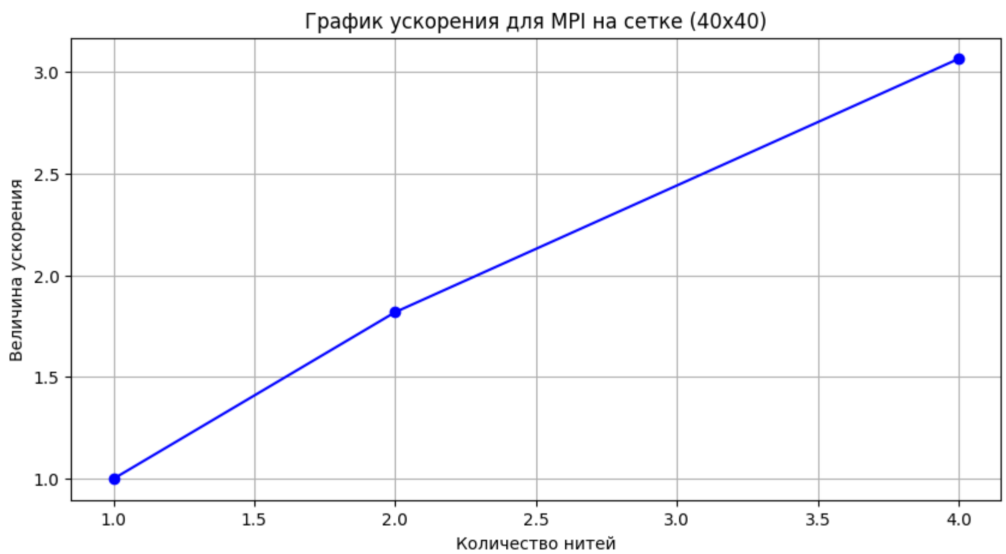


График ускорения для MPI на сетке (40x40)
Величина ускорения
Количество нитей
1.0
1.5
2.0
2.5
3.0
3.5
4.0

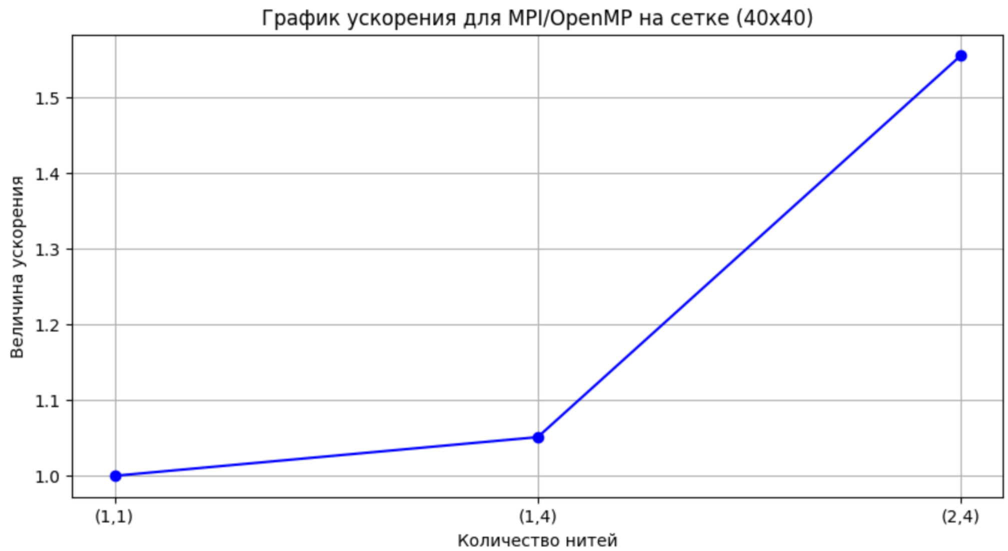


График ускорения для MPI/OpenMP на сетке (40x40)
Величина ускорения
1.0
1.1
1.2
1.3
1.4
1.5
(1,1)
(1,4)
(2,4)
Количество нитей

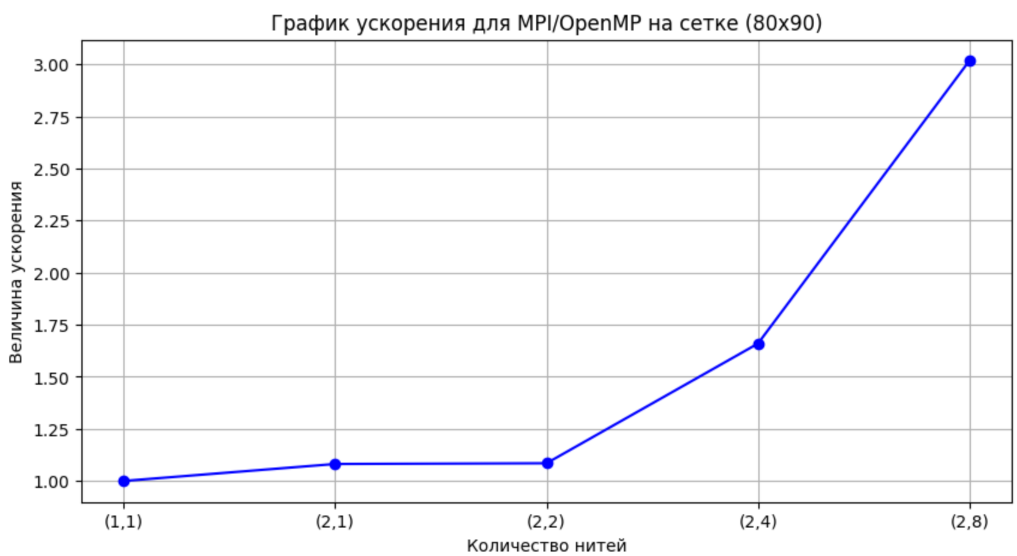


График ускорения для MPI/OpenMP на сетке (80x90)
Величина ускорения
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
(1,1)
(2,1)
(2,2)
(2,4)
(2,8)
Количество нитей

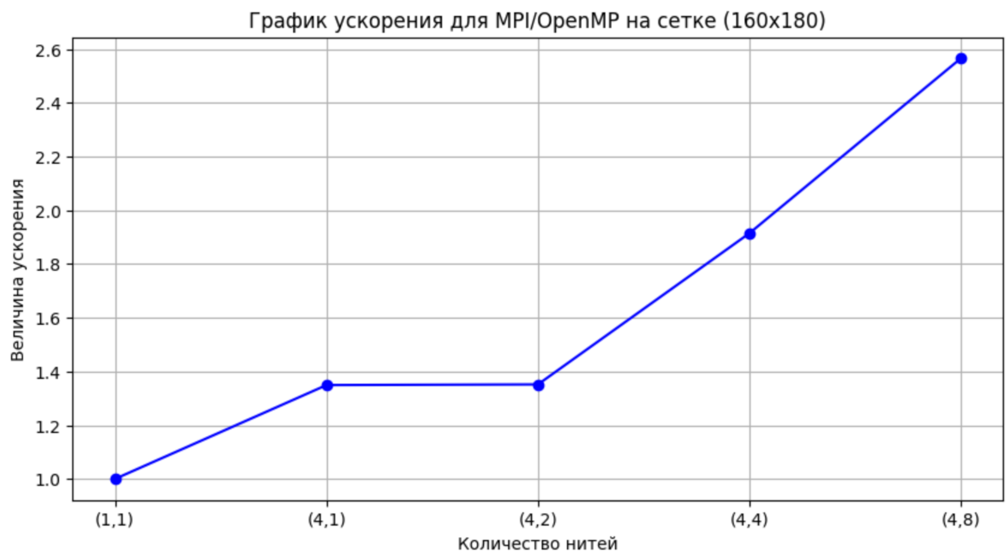


График ускорения для MPI/OpenMP на сетке (160x180)
Величина ускорения
Количество нитей
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6
(1,1)
(4,1)
(4,2)
(4,4)
(4,8)