

Отчет по заданию №1 Вариант 4

Ван Син, студент 614 группы, второй поток
Факультет вычислительной математики и кибернетики
МГУ им. М. В. Ломоносова
Москва, 2024 год

1 Введение

Требуется приближенно решить задачу Дирихле для уравнения Пуассона в криволинейной области. Задание необходимо выполнить на ПВС Московского университета IBM Polus.

2 Математическая постановка задачи

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

$$-\Delta u = f(x, y), \quad (2.1)$$

в котором оператор Лапласа

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

функция $f(x, y)$ считается известной. Для выделения единственного решения уравнение дополняется граничными условием Дирихле:

$$u(x, y) = 0, \quad (x, y) \in \gamma \quad (2.2)$$

Требуется найти функцию $u(x, y)$, удовлетворяющую уравнению (2.1) в области D и краевому условию (2.2) на ее границе.

3 Численные методы решения

Метод фиктивных областей и т.д. Как указано в документе "Задание по курсу «Суперкомпьютерное моделирование и технологии».

4 Программная реализация

Программа вводит M, N , точность и количество потоков (для OpenMP) в главную функцию `main` и постоянно вызывает функцию `kernel` в `main` для завершения решения задачи.

Функция `kernel` инициализирует координаты, соответствующие каждой точке (i, j) , и предполагает, что исходное решение w , которое не подвергалось итерации, является нулевой матрицей. Функция вычисляет соответствующую систему линейных уравнений $Aw = F$ постоянный член F_{ij} и затем выполняет итерации в соответствии с соответствующим численным методом. Для реализации вычитания между матрицами, а также умножения констант на матрицы и матриц на матрицы в процессе итерации используются функция `MM_minus`, функция `NM_pro` и функция `ScalarPro`. Условием остановки итерации является то, что число разности между предыдущим вычислением и текущим результатом меньше или равно некоему значению ϵ (все принимается равным $1e - 6$).

При вычислении F_{ij} вызывается функция `F_ij`, которая учитывает позиционную связь соответствующего узла с областью D (трапеция) в данной задаче и вычисляет интеграл по схеме площади поперечника, где для суждения о позиционной связи используется функция `PointClassific`.

В процессе вычисления Aw вызываются функции a_{ij} и b_{ij} для расчета необходимых коэффициентов. Для точек, расположенных на краях виртуальной области (не имеющих левых, правых, верхних или нижних соседей), принимается единое значение 0.

Для технологии OpenMP распараллеливание осуществляется с использованием соответствующих операторов там, где это необходимо, что приводит к значительному повышению эффективности программы.

Используя `write2DVectorToFile`, можно вывести w в виде матрицы в некоторый txt-файл, а затем с помощью программы на языке Python нарисовать соответствующее изображение.

5 Численный эксперимент

Для последующих экспериментов равномерно возьмем $\epsilon = \max(h_1, h_2)^2$ (параметр, используемый для вычисления коэффициентов a_{ij} и b_{ij}) с $\delta = 1e - 6$.

Путем экспериментов было рассчитано численное решение w для размеров 10, 20, 40, 80 и 160, решение было записано в txt-файл, а результаты отображены на графике с помощью программы на языке Python следующим образом:

Число точек сетки $M \times N$	Время решения (s)
10×10	0.077018
20×20	2.9020
40×40	21.7525

Таблица 1: Таблица с результатами расчетов

В рамках задания были выполнены следующие этапы:

1. реализована параллельная программа на языке C++14, приближенно решающая двумерную задачу Дирихле для уравнения Пуассона
2. синхронный код был ускорен средствами OpenMP
3. проверено качество работы алгоритма для сеток разных размеров с разным количеством потоков на ПВС IBM Polus
4. визуализированы результаты.

Кол-во OpenMP-нитей	Число точек сетки (М x N)	Число итераций	Время решения	Ускорение
1	80×90	67481	203.80	1
2	80×90	48381	67.18	3.02985
4	80×90	59261	47.90	4.25470
8	80×90	54267	35.65	5.71669
16	80×90	47582	31.21	6.52996
1	160×180	123276	1788.15	1
4	160×180	142375	1012.16	1.76667
8	160×180	123468	770.35	2.32121
16	160×180	129788	376.49	4.74952
32	160×180	130309	238.87	7.48587

Таблица 2: Таблица с результатами расчетов на ПВС IBM Polus (OpenMP код).

Количество процессов MPI	Кол-во OpenMP-нитей	Число точек сетки (М x N)	Число итераций	Время решения	Ускорение
2	1	80×90	50087	187.15	1.08897
2	2	80×90	43478	145.78	1.39800
2	4	80×90	39978	101.16	2.01463
2	8	80×90	45505	77.35	2.63477
4	1	160×180	117259	508.73	3.51492
4	2	160×180	124352	329.42	5.42817
4	4	160×180	133042	288.65	6.19487
4	8	160×180	107983	143.46	12.46445

Таблица 3: Таблица с результатами расчетов на ПВС IBM Polus (MPI код).

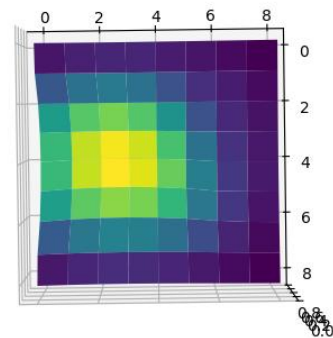
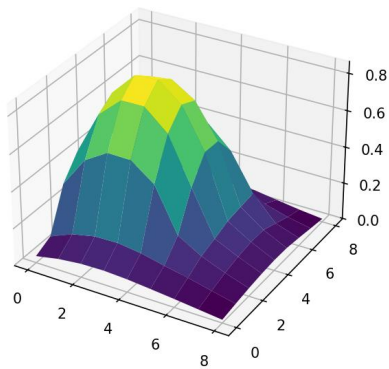


Рис. 1: Solution 10x10

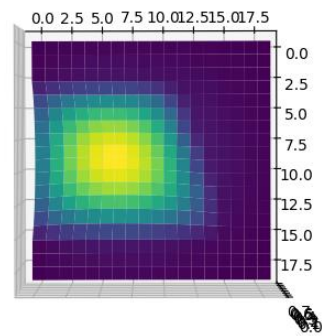
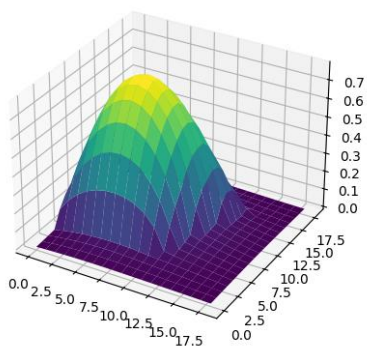


Рис. 2: Solution 20x20

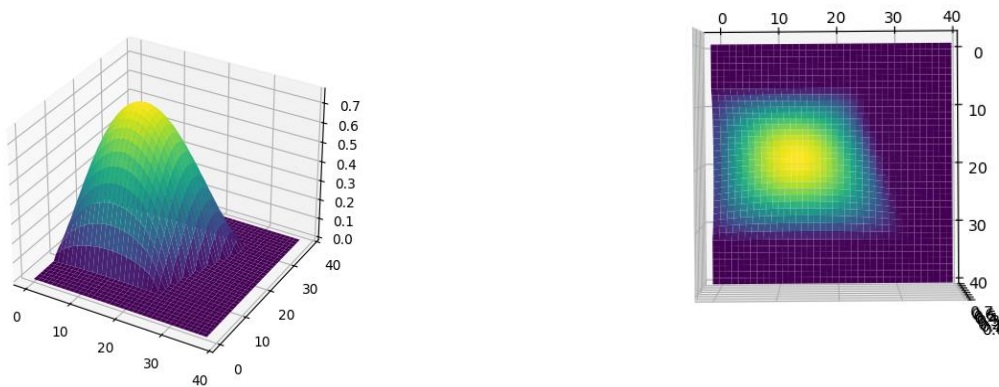


Рис. 3: Solution 40x40

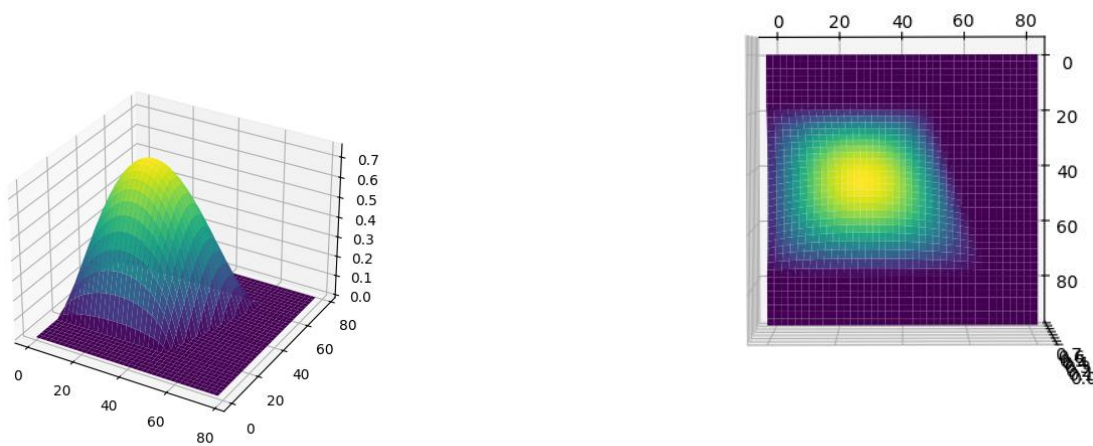


Рис. 4: Solution 80x90

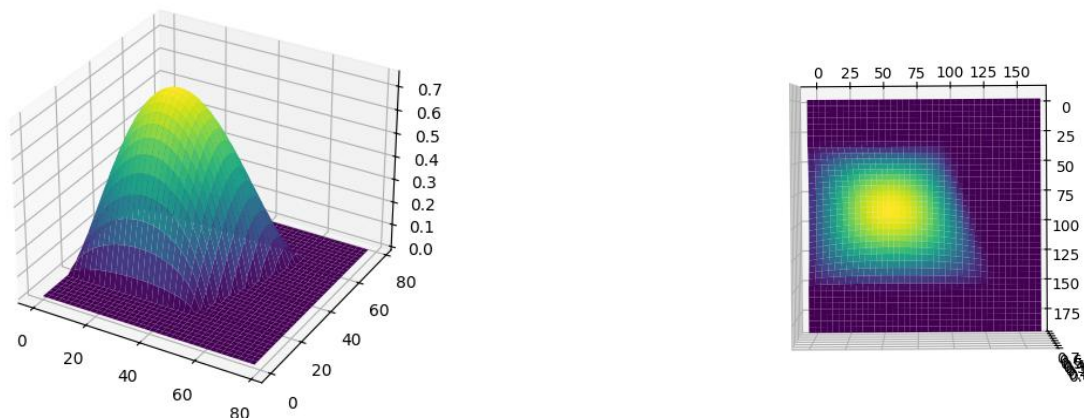


Рис. 5: Solution 160x180

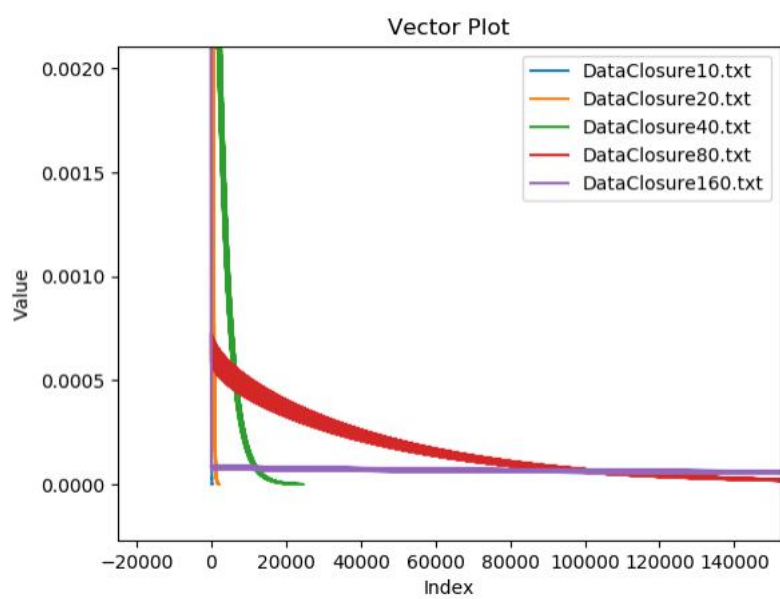


Рис. 6: График ошибок для 10×10 , 20×20 , 40×40 , 80×90 и 160×180