



Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

**Отчет задания №1 по курсу
«Суперкомпьютерное моделирование и
технологии»**

Вариант 8

Студент: Чжай Хао
Группа: 619/1

Москва, 2024

Оглавление

0.1	Математическая постановка задачи	3
0.2	Метод фиктивных областей	3
0.3	Разностная схема решения задачи	5
0.4	Метод решения системы линейных алгебраических уравнений	7
0.4.1	метод скорейшего спуска	7
0.4.2	Метод сопряженных градиентов	8
0.5	Краткое описание проделанной работы по созданию OpenMP-программы	10
0.6	Краткое описание проделанной работы по созданию MPI-программы	13
0.6.1	Декомпозиция расчетной области	13
0.6.2	Структуры данных	13
0.6.3	Организация межпроцессного взаимодействия	13
0.6.4	Особенности реализации	14
0.7	Краткое описание проделанной работы по созданию MPI+OpenMP-программы	15
0.8	Экспериментальная среда и выполнение тестов	16
0.8.1	Ручная компиляция и запуск	18
0.8.2	Результаты расчетов для OpenMP программы	20
0.8.3	Результаты расчетов для MPI программы	22
0.8.4	Результаты расчетов для MPI+OpenMP программы	23
0.8.5	Графики приближенного решения	24
0.9	Описание файлов и их назначения в проекте	27

0.1 Математическая постановка задачи

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

$$-\Delta u = f(x, y) \quad (1)$$

в котором оператор Лапласа:

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

Для выделения единственного решения уравнение дополняется граничным условием Дирихле:

$$u(x, y) = 0, \quad (x, y) \in \gamma \quad (2)$$

Требуется найти функцию $u(x, y)$, удовлетворяющую уравнению (1) в области D и краевому условию (2) на ее границе.

Рассмотрим задачу в случае, когда правая часть уравнения $f(x, y) = 1$, а область D представляет собой область, ограниченную дугой параболы и отрезком прямой:

$$D = \{(x, y) \mid y^2 < x < 1\}$$

0.2 Метод фиктивных областей

Для приближенного решения задачи (1),(2) предлагается воспользоваться методом фиктивных областей.

Пусть область D принадлежит прямоугольнику $\Pi = \{(x, y) : A_1 < x < B_1, A_2 < y < B_2\}$. Обозначим через \overline{D} , $\overline{\Pi}$ замыкание области D и прямоугольника Π соответственно, через Γ границу прямоугольника. Разность множеств:

$$\hat{D} = \Pi \setminus \overline{D}$$

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

В прямоугольнике Π рассматривается задача Дирихле:

$$\begin{cases} -\frac{\partial}{\partial x} \left(k(x, y) \frac{\partial v}{\partial x} \right) - \frac{\partial}{\partial y} \left(k(x, y) \frac{\partial v}{\partial y} \right) = F(x, y) \\ v(x, y) = 0, \quad (x, y) \in \Gamma \end{cases} \quad (3)$$

с кусочно-постоянным коэффициентом:

$$k(x, y) = \begin{cases} 1, & (x, y) \in D \\ 1/\varepsilon, & (x, y) \in \hat{D} \end{cases}$$

и правой частью:

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

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

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

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

Переход к новой задаче позволяет получить решение исходной задачи с любой наперед заданной точностью $\varepsilon > 0$, решая при этом задачу Дирихле в прямоугольнике Π , содержащем исходную область:

$$\max_{P \in \bar{D}} \|v(x, y) - u(x, y)\| \leq C\varepsilon, C > 0$$

Для случая, когда область D представляет собой область, ограниченную дугой параболы и отрезком прямой: $D = \{(x, y) \mid y^2 < x < 1\}$, выберем прямоугольник

$$\Pi = \{(x, y) \mid 0 < x < 1.0, -1.0 < y < 1.0\}$$

0.3 Разностная схема решения задачи

В замыкании прямогольника $\bar{\Pi}$ определим равномерную прямоугольную сетку $\bar{\omega}_h = \bar{\omega}_1 \times \bar{\omega}_2$, где

$$\begin{aligned}\bar{\omega}_1 &= \{x_i = A_1 + ih_1, i = 0, \dots, M\}, \quad h_1 = (B_1 - A_1)/M \\ \bar{\omega}_2 &= \{y_j = A_2 + jh_2, j = 0, \dots, N\}, \quad h_2 = (B_2 - A_2)/N\end{aligned}$$

Множество внутренних узлов сетки $\bar{\omega}_h$ обозначим ω_h .

Рассмотрим линейное пространство H функций, заданных на сетке ω_h .

Обозначим через w_{ij} значение сеточной функции H в узле сетки $(x_i, y_j) \in \omega_h$.

Определим скалярное произведение и норму в пространстве сеточных функций H :

$$\begin{aligned}(u, v) &= \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} h_1 h_2 u_{ij} v_{ij} \\ \|u\| &= \sqrt{(u, u)}\end{aligned}$$

Будем использовать метод конечных разностей, который заключается в замене дифференциальной задачи математической физики на конечно-разностную операторную задачу вида:

$$A\omega = B$$

$$A : H \rightarrow H$$

Дифференциальное уравнение задачи (3) во всех внутренних точках сетки аппроксимируется разностным уравнением:

$$\begin{aligned}-\frac{1}{h_1}(a_{i+1j}\frac{\omega_{i+1j} - \omega_{ij}}{h_1} - a_{ij}\frac{\omega_{ij} - \omega_{i-1j}}{h_1}) - \frac{1}{h_2}(b_{ij+1}\frac{\omega_{ij+1} - \omega_{ij}}{h_2} - b_{ij}\frac{\omega_{ij} - \omega_{ij-1}}{h_2}) &= F_{ij} \\ i = 1, \dots, M-1, j = 1, \dots, N-1\end{aligned}$$

в котором коэффициенты при $i = 1, \dots, M, j = 1, \dots, N$

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

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

и правая часть при $i = 1, \dots, M - 1, j = 1, \dots, N - 1$

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

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

Краевые условия Дирихле в задаче (3) аппроксимируются точно равенством

$$w_{ij} = w(x_i, y_j) = 0, (x_i, y_j) \in \Gamma$$

Полуцелые узлы означают:

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

Полученная система является линейной относительно неизвестных величин и может быть представлена в виде $A\omega = B$ с самосопряженным и положительно определенным оператором A . Построенная разностная схема линейна и имеет единственное решение при любой правой части.

Интегралы a_{ij}, b_{ij} будем вычислять аналитически: $a_{ij} = h_2^{-1}l_{ij} + (1 - h_2^{-1}l_{ij})/\varepsilon$, где l_{ij} длина части отрезка $[y_{j-1/2}, y_{j+1/2}]$, которая принадлежит области D . Для вычисления l_{ij} для заданного $\hat{x} = x_{i-1/2}$ вычислим точки пересечения прямой $x = \hat{x}$ с границей эллипса γ . Тогда $l_{ij} = \min(y_1, y_{j+1/2}) - \max(y_2, y_{j-1/2})$, где

$$y_{1,2} = \pm \frac{1}{4} \sqrt{1 - \hat{x}^2}$$

Правую часть разностной схемы приближенно заменим на значение в центре квадрата Π_{ij} :

$$F_{ij} = F(x_i, y_j) = \begin{cases} 1, & (x_i, y_j) \in D, \\ 0, & (x_i, y_j) \in \hat{D} \end{cases}$$

0.4 Метод решения системы линейных алгебраических уравнений

0.4.1 метод скорейшего спуска

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

$$\|w - w^{(k)}\|_E \rightarrow 0, \quad k \rightarrow +\infty$$

Начальное приближение $w^{(0)}$ можно выбрать равным нулю во всех точках расчетной сетки.

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:

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

В качестве критерия останова можно использовать условие:

$$\|r^{(k)}\|_E < \delta$$

с некоторой положительной константой $\delta > 0$, задающей точность приближенного решения.

0.4.2 Метод сопряженных градиентов

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

$$\|w - w^{(k)}\|_E \rightarrow 0, \quad k \rightarrow +\infty$$

Начальное приближение $w^{(0)}$ можно выбрать равным нулю во всех точках расчетной сетки.

Метод является итерационным. На каждой итерации вычисляются следующие величины:

1. Начальная невязка:

$$r^{(0)} = B - Aw^{(0)}$$

2. Начальное направление:

$$p^{(0)} = r^{(0)}$$

3. Для $k = 0, 1, 2, \dots$ выполняются шаги:

- (a) Вычисление шагового множителя:

$$\alpha_k = \frac{(r^{(k)}, r^{(k)})}{(Ap^{(k)}, p^{(k)})}$$

- (b) Обновление приближения решения:

$$w^{(k+1)} = w^{(k)} + \alpha_k p^{(k)}$$

- (c) Обновление невязки:

$$r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)}$$

(d) Вычисление параметра сопряженности:

$$\beta_k = \frac{(r^{(k+1)}, r^{(k+1)})}{(r^{(k)}, r^{(k)})}$$

(e) Обновление направления поиска:

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$$

В качестве критерия останова можно использовать условие:

$$\|r^{(k)}\|_E < \delta$$

где $\delta > 0$ - заданная точность приближенного решения.

Важные свойства метода:

1) Для симметричной положительно определенной матрицы A метод сходится не более чем за n итераций, где n - размерность задачи.

2) Последовательные направления поиска $p^{(k)}$ являются A -сопряженными:

$$(Ap^{(i)}, p^{(j)}) = 0, \quad i \neq j$$

3) Последовательные невязки $r^{(k)}$ ортогональны:

$$(r^{(i)}, r^{(j)}) = 0, \quad i \neq j$$

4) Скорость сходимости метода определяется оценкой:

$$\|w - w^{(k)}\|_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|w - w^{(0)}\|_A$$

где $\kappa = \lambda_{\max}/\lambda_{\min}$ - число обусловленности матрицы A .

0.5 Краткое описание проделанной работы по созданию OpenMP-программы

Для выполнения задания был разработаны два варианта параллельной реализации для численного решения уравнения Пуассона - с использованием метода наискорейшего спуска (`openmp.cpp`) и метода сопряженных градиентов (`openmp_cg.cpp`). Программы используют метод конечных разностей для дискретизации двумерного уравнения Пуассона и метод фиктивных областей для работы с криволинейной областью.

Основные функции сетки

- `SolverStats`: структура для хранения статистики решателя (количество итераций, время решения, достигнутая точность)
- `create_grid`: создание и инициализация структуры сетки заданного размера
- `get_x`, `get_y`: вычисление координат точек на основе индексов сетки

Функции разностных вычислений

- `der_x_right`, `der_x_left`: вычисление правой и левой разности по x
- `der_y_right`, `der_y_left`: вычисление правой и левой разности по y

Расчет области и границ

- `get_parabolic_y`: вычисление y-координаты параболы
- `get_y_length`: вычисление длины пересечения по y для заданного x
- `get_x_length`: вычисление длины пересечения по x для заданного y
- `calculate_area`: Функция для вычисления площади пересечения сетки с областью, ограниченной параболой $y = x$ и прямой $x = 1$, используя ме-

тод разбиения на многоугольники и вычисления площади через векторное произведение.

Вычисление коэффициентов

- `calculate_aij`: вычисление матрицы коэффициентов по x с использованием OpenMP
- `calculate_bij`: вычисление матрицы коэффициентов по y с использованием OpenMP

Основные функции численных вычислений

- `apply_diff_operator`: применение разностного оператора к сеточной функции
- `scalar_product`: вычисление скалярного произведения двух сеточных функций
- `init_grid`: инициализация значений сетки с учетом геометрии области

Основные функции решения

- `solve_poisson`: реализация метода наискорейшего спуска (`openmp.cpp`)
- `solve_poisson_cg`: реализация метода сопряженных градиентов (`openmp_cg.cpp`)

Параллельная реализация

Основные функции, распараллеленные с помощью OpenMP:

- `calculate_aij` и `calculate_bij`: параллельное вычисление коэффициентов
- `apply_diff_operator`: параллельное применение разностного оператора

- `scalar_product`: параллельное вычисление скалярного произведения с редукцией
- `init_grid`: параллельная инициализация значений сетки

В параллельных функциях использована директива `#pragma omp parallel for collapse(2)` со стратегией планирования `guided` для оптимальной балансировки нагрузки. Для `scalar_product` дополнительно применена редукция для корректного суммирования результатов от разных потоков.

Программа позволяет задавать количество потоков через аргументы командной строки и сохраняет результаты расчетов и статистику производительности в файлы для последующего анализа.

0.6 Краткое описание проделанной работы по созданию MPI-программы

Для решения двумерного уравнения Пуассона разработана параллельная реализация с использованием MPI. Программа основана на методе конечных разностей и методе фиктивных областей для работы с криволинейной областью, ограниченной параболой.

0.6.1 Декомпозиция расчетной области

Реализовано двумерное разбиение прямоугольной области на домены с использованием виртуальной топологии типа решетка. Для создания оптимальной декомпозиции используется функция MPI_Dims_create, которая автоматически определяет размерности процессорной решетки. Каждый процесс обрабатывает свой домен с учетом граничных (ghost) ячеек для обеспечения корректных вычислений на границах доменов.

0.6.2 Структуры данных

Основной структурой данных является Grid, которая содержит:

- Локальные данные процесса (values) и их размеры (local_m, local_n)
- Информацию о глобальной сетке (size_m, size_n, step_x, step_y)
- Параметры декомпозиции (coords[2], dims[2])
- Коммуникатор с виртуальной топологией (cart_comm)
- Буферы для обмена данными между процессами (send_buf, recv_buf)

0.6.3 Организация межпроцессного взаимодействия

Реализованы следующие схемы взаимодействия процессов:

1. Обмен граничными значениями через функцию `exchange_boundaries`, использующую неблокирующие операции `MPI_Isend`/`MPI_Irecv` для одновременной передачи данных по всем направлениям.
2. Вычисление глобальных редукций через `MPI_Allreduce` для:
 - Скалярных произведений векторов в методе наискорейшего спуска
 - Определения глобальной невязки для критерия остановки
3. Сбор результатов на главном процессе для сохранения в файл с использованием индивидуальных передач данных от каждого процесса.

0.6.4 Особенности реализации

В программе реализованы следующие ключевые алгоритмические компоненты:

- Метод фиктивных областей с вычислением коэффициентов на основе пересечения расчетных ячеек с криволинейной границей
- Метод наискорейшего спуска для решения системы разностных уравнений
- Эффективная схема обмена граничными значениями с использованием неблокирующих коммуникаций

0.7 Краткое описание проделанной работы по созданию MPI+OpenMP-программы

На основе существующих программ MPI и OpenMP была разработана гибридная версия. Основные модификации включают:

- Добавление OpenMP-параллелизма в MPI-процессы:
 - Использование MPI_Init_thread для инициализации среды MPI с поддержкой гибридного параллелизма
 - Управление количеством OpenMP-нитей для каждого процесса через аргументы командной строки
 - Добавление директив OpenMP parallel for в вычислительно-интенсивные циклы
- Гибридная реализация ключевых функций:
 - scalar_product: комбинация OpenMP-редукции и глобальной редукции MPI
 - apply_diff_operator_local: параллельные вычисления OpenMP с обменом границ через MPI
 - calculate_ajj/bij_local: параллельные циклы OpenMP с доменной декомпозицией MPI

0.8 Экспериментальная среда и выполнение тестов

Конфигурация вычислительной среды

Тестирование проводилось на суперкомпьютере IBM Polus МГУ:

- Узлы: polus-c3-ib и polus-c4-ib
- 10 ядер на задачу

Автоматизация тестирования

Разработан скрипт `run_tests.sh` для автоматизации процесса тестирования всех версий программы:

1. Этап компиляции

- Последовательная версия (`sequential.cpp`)
- OpenMP версия (`openmp.cpp`)
- OpenMP с методом сопряженных градиентов (`openmp_cg.cpp`)
- MPI версия (`mpi.cpp`)
- MPI+OpenMP версия (`mpi_omp.cpp`)

2. Конфигурации тестов

Выходные файлы

- Логи выполнения: `seq.log`, `omp_.log`, `mpi_.log`, `mpi_omp_.log`
- Логи ошибок: `error_.log`
- Времена выполнения: `serial_times.tsv`, `omp_times.tsv`, `mpi_times.tsv`, `mpi_omp_times.tsv`
- Численные решения: `serial_.tsv`, `omp_.tsv`, `mpi_.tsv`, `mpi_omp_.tsv`

Версия	Размер сетки	Конфигурация
Последовательная	$10 \times 10, 20 \times 20, 40 \times 40$	-
OpenMP	40×40	1,2,4,8,16 потоков
OpenMP	80×90	1,2,4,8,16,32 потоков
OpenMP	160×180	1,2,4,8,16,32 потоков
MPI	40×40	1,2,4 процесс
MPI	80×90	1,2,4 процесс
MPI	160×180	1,2,4 процесс
MPI+OpenMP	80×90	2 процесс 1,2,4,8 потоков
MPI+OpenMP	160×180	4 процесс 1,2,4,8 потоков

Таблица 1: Конфигурации тестов

Инструкции по запуску

```
$ module load SpectrumMPI/10.1.0
$ module load OpenMPI/4.0.2
$ chmod +x run_tests.sh
$ ./run_tests.sh
```

Скрипт предлагает запустить все тесты или выбрать конкретные конфигурации.

0.8.1 Ручная компиляция и запуск

Варианты компиляции

```
# Версия последовательная
g++ -fopenmp {sequential_program_name}.cpp -O3 -std=c++11
                                         -o {sequential_program_name}

# Версия OpenMP
g++ -fopenmp {openmp_program_name}.cpp -O3 -std=c++11
                                         -o {openmp_program_name}

# Версия MPI
mpicxx {mpi_program_name}.cpp -O3 -std=c++11
                                         -o {mpi_program_name}
[MPI На Polus:]
# module load SpectrumMPI/10.1.0
# mpicc -std=c++11 -O3 -qstrict mpi.cpp -o mpi

# Версия MPI+OpenMP
mpicxx -fopenmp {mpi_program_name}.cpp -O3 -std=c++11
                                         -o {mpi_program_name}
[MPI На Polus:]
# module load SpectrumMPI/10.1.0
# module load OpenMPI/4.0.2
# mpicxx -fopenmp {mpi_program_name}.cpp -O3 -std=c++11
                                         -o {mpi_program_name}
```

Команды запуска

```
# Версия последовательная
.{sequential_program_name} {M} {N} {точность решения}

# Версия OpenMP
.{openmp_program_name} {M} {N} {точность решения}
{количество потоков OpenMP}

# Версия MPI
mpirun -np {Число процессов MPI} ./{openmp_program_name} {M} {N}
{точность решения}

# Версия MPI+OpenMP
mpirun -np {Число процессов MPI} ./{openmp_program_name} {M} {N}
{точность решения} {количество потоков OpenMP}
```

0.8.2 Результаты расчетов для OpenMP программы

Использовал метод наискорейшего спуска для тестирования сетки размером 40x40.

Нить	Число точек сетки ($M \times N$)	итерации	точности	Время	Ускорение
1	40×40	188005	1.00e-07	16.368	0.883
2	40×40	188005	1.00e-07	11.356	1.272
4	40×40	188005	1.00e-07	10.950	1.320
8	40×40	188005	1.00e-07	18.026	0.801
16	40×40	188005	1.00e-07	37.712	0.383

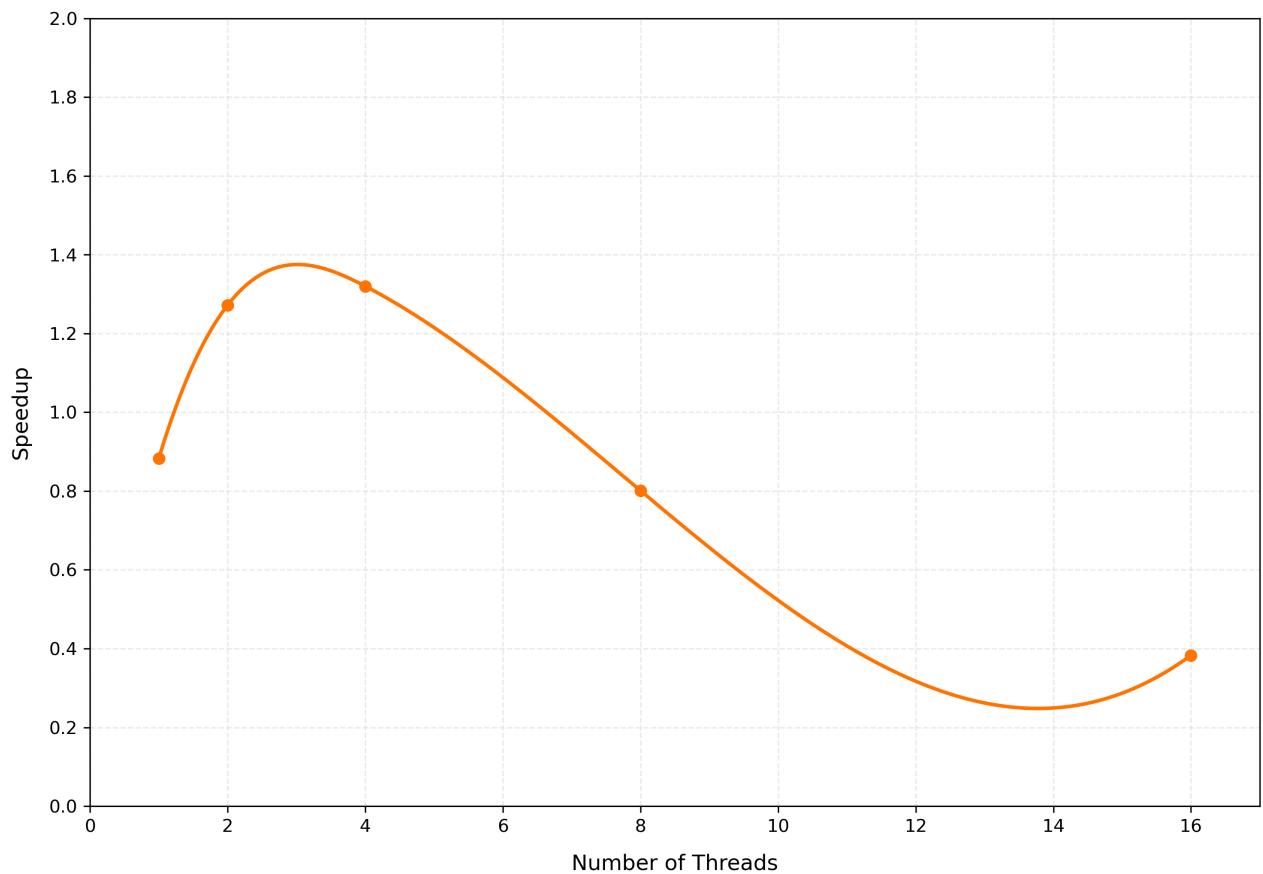


Рис. 1: График ускорения OpenMP при размере сетки (40, 40)

Потом также использовал метод наискорейшего спуска протестировал сетки размером 80×90 и 160×180 , и получили следующие данные:

Число нити	Сетки ($M \times N$)	итерации	точности	Время	Ускорение
1	80×90	2620783	1.00e-08	1134.308	1.000
2	80×90	2620783	1.00e-08	678.345	1.673
4	80×90	2620783	1.00e-08	407.717	2.783
8	80×90	2620783	1.00e-08	425.574	2.665
16	80×90	2620783	1.00e-08	765.398	1.482
32	80×90	2620783	1.00e-08	1665.492	0.681
1	160×180	1330973	2.50e-09	2285.413	1.000
2	160×180	1330973	2.50e-09	1167.513	1.957
4	160×180	1330973	2.50e-09	638.965	3.577
8	160×180	1330973	2.50e-09	428.395	5.334
16	160×180	1330973	2.50e-09	544.465	4.197
32	160×180	1330973	2.50e-09	1059.309	2.157

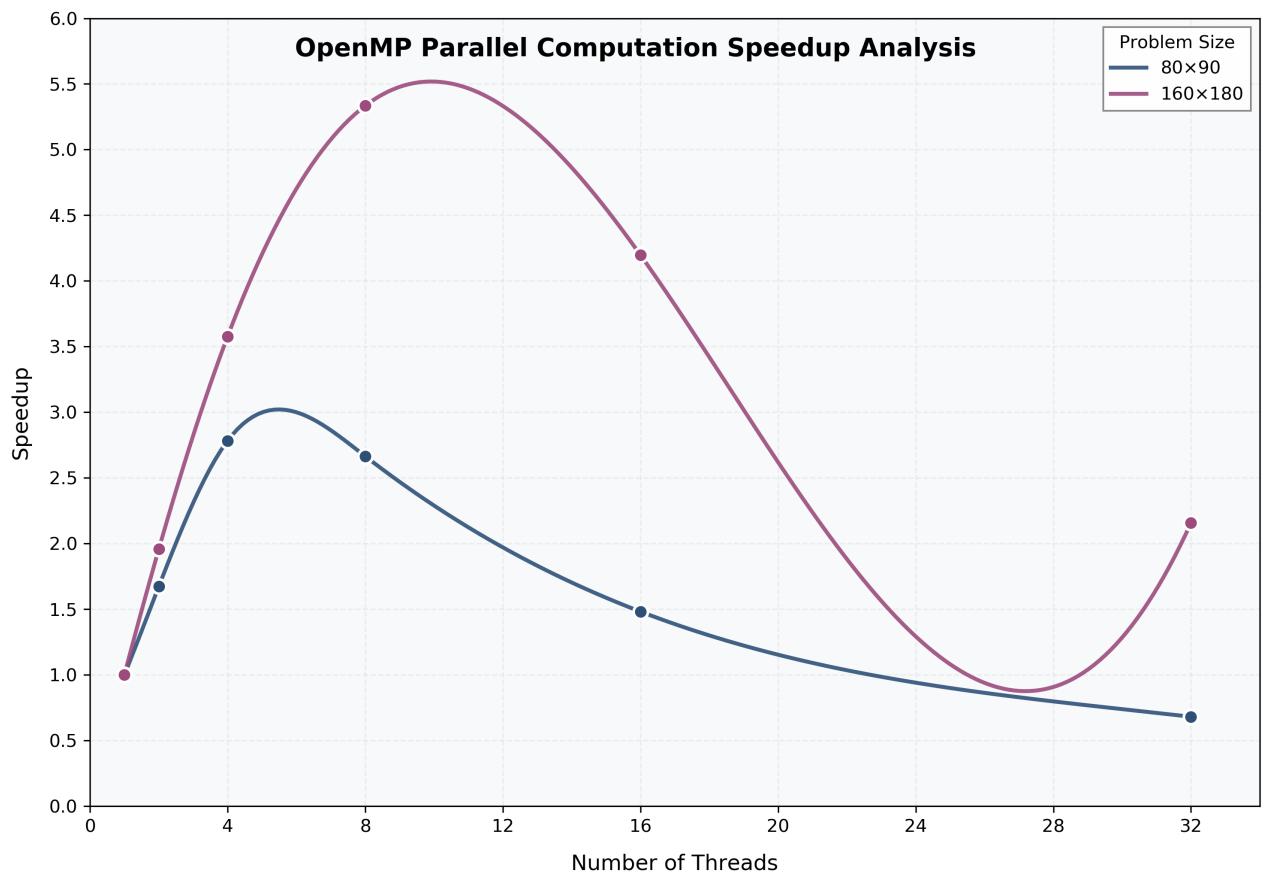


Рис. 2: График ускорения OpenMP при размере сетки $(80, 90)$ и $(160, 180)$

0.8.3 Результаты расчетов для MPI программы

Использовал метод наискорейшего спуска для тестирования MPI.

Число процессов	Сетки ($M \times N$)	итерации	точности	Время	Ускорение
1	40×40	188005	1.00e-07	16.854	1.000
2	40×40	188005	1.00e-07	9.732	1.732
4	40×40	188005	1.00e-07	5.828	2.892
1	80×90	2620783	1.00e-08	1032.559	1.000
2	80×90	2620783	1.00e-08	541.186	1.908
4	80×90	2620783	1.00e-08	286.711	3.601
1	160×180	1330973	2.50e-09	2085.781	1.000
2	160×180	1330973	2.50e-09	1057.868	1.971
4	160×180	1330973	2.50e-09	538.497	3.873

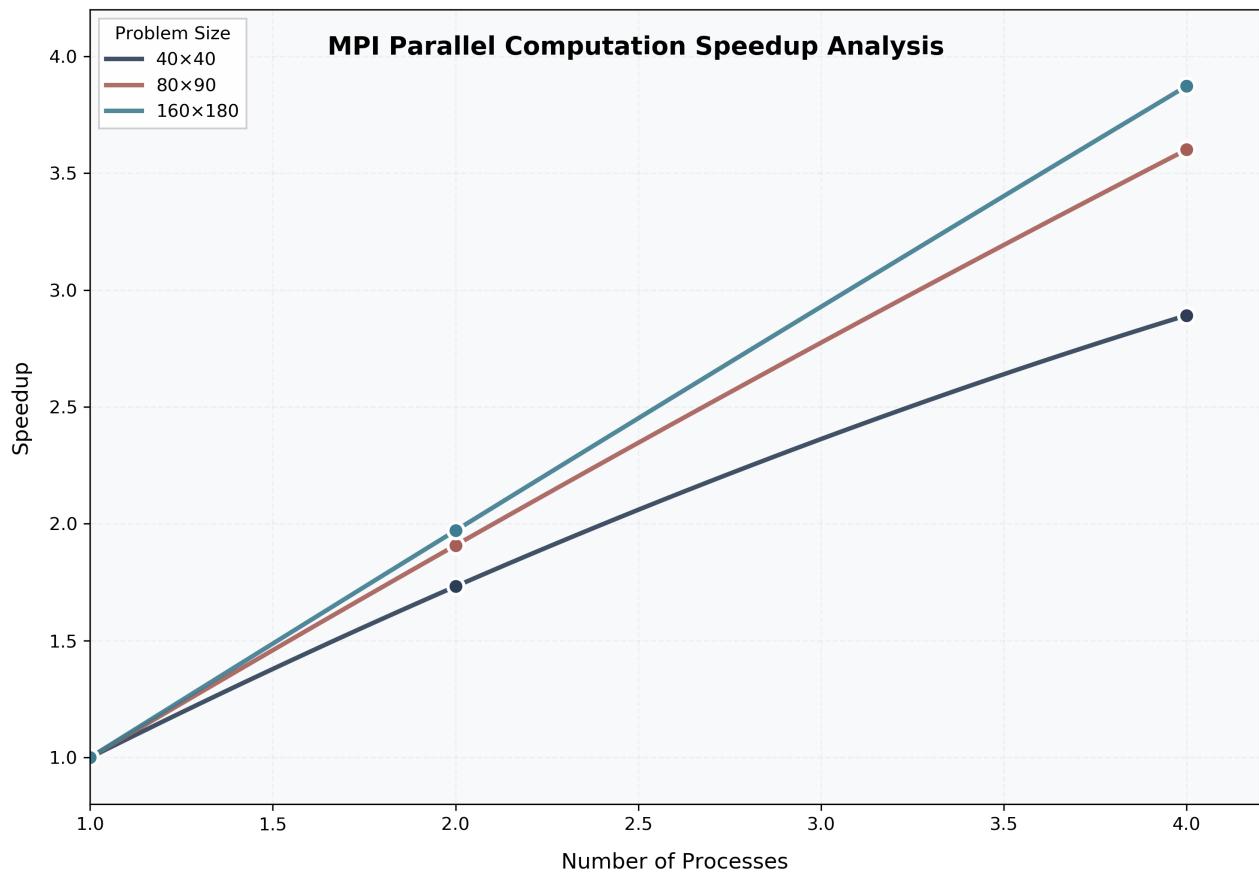


Рис. 3: График ускорения MPI при размере сетки (80, 90) и (160, 180)

0.8.4 Результаты расчетов для MPI+OpenMP программы

Использовал метод наискорейшего спуска для тестирования MPI+OpenMP. Здесь рассчитано ускорение в сравнении с результатами выполнения при 1 процессе и 1 потоке.

Процессы	Нити	Сетки ($M \times N$)	итерации	точности	Время	Ускорение
2	1	80×90	2620783	1.00e-08	1846.795	1.30
2	2	80×90	2620783	1.00e-08	1369.179	1.75
2	4	80×90	2620783	1.00e-08	983.165	2.44
2	8	80×90	2620783	1.00e-08	1049.561	2.29
4	1	160×180	1330973	2.50e-09	938.597	4.69
4	2	160×180	1330973	2.50e-09	435.000	10.12
4	4	160×180	1330973	2.50e-09	334.510	13.16
4	8	160×180	1330973	2.50e-09	439.640	10.01

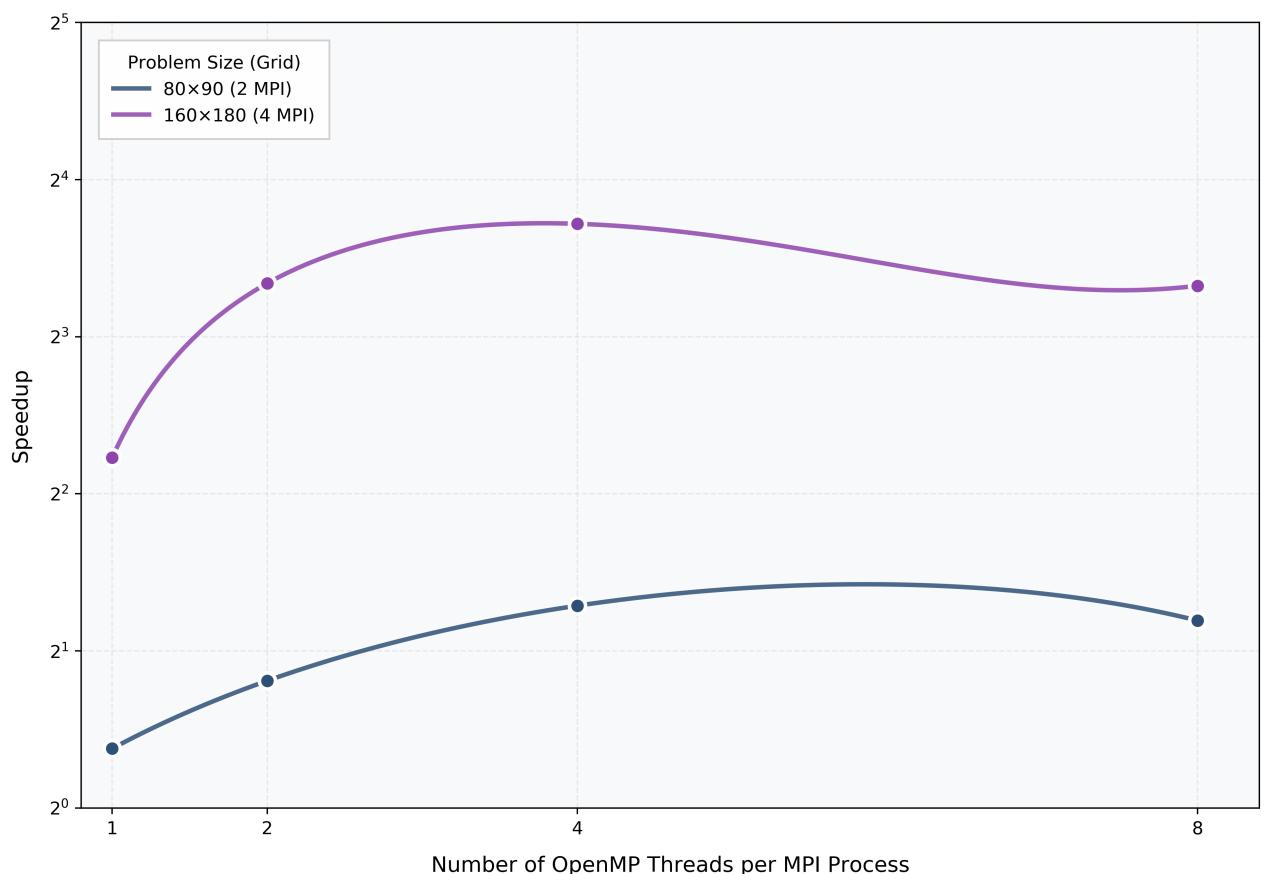


Рис. 4: График ускорения MPI+OpenMP при разном размере сетки ($80, 90$) и ($160, 180$)

0.8.5 Графики приближенного решения

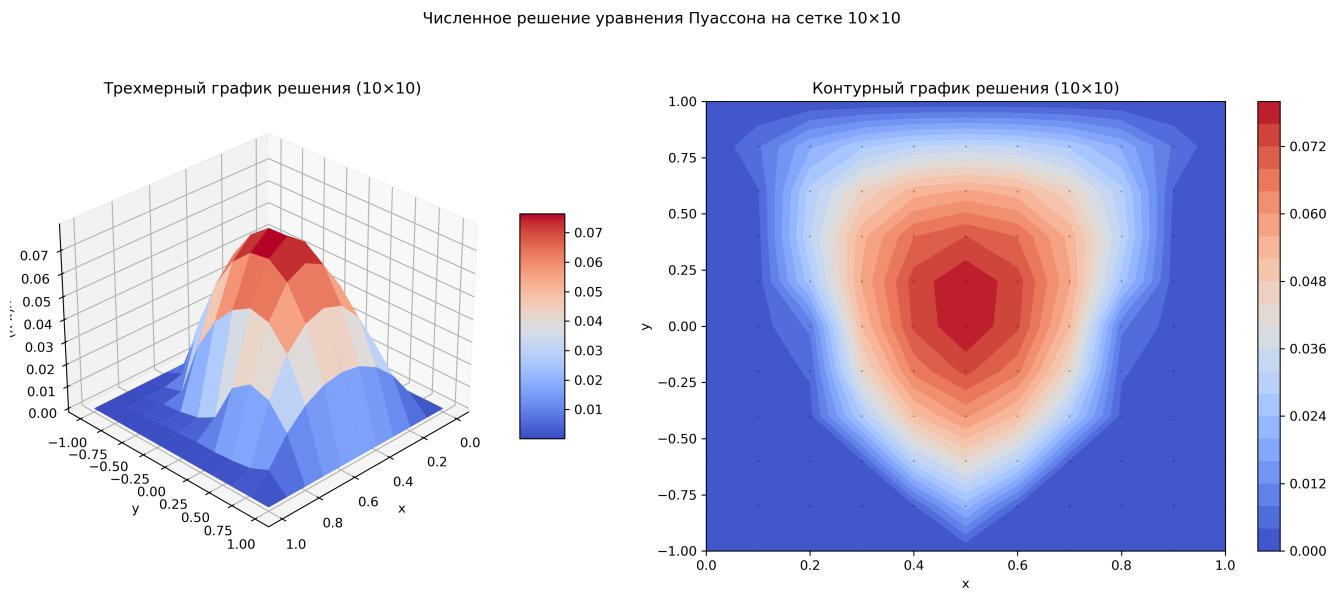


Рис. 5: График решения с сеткой 10×10 точек

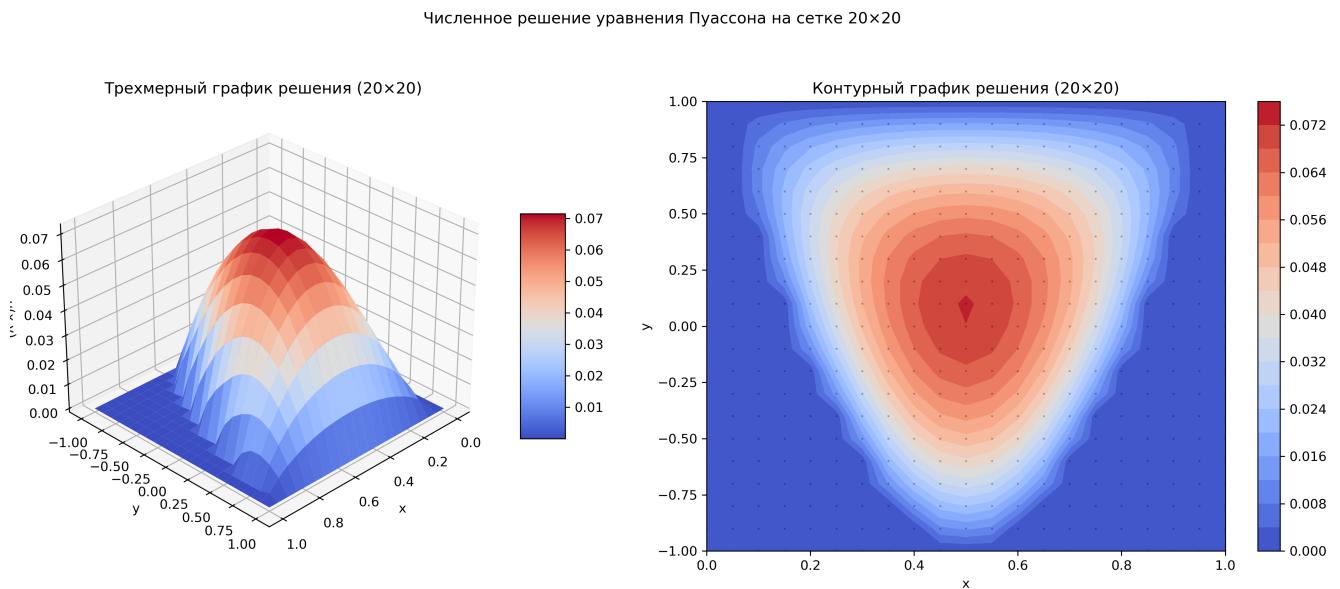


Рис. 6: График решения с сеткой 20×20 точек

Численное решение уравнения Пуассона на сетке 40×40

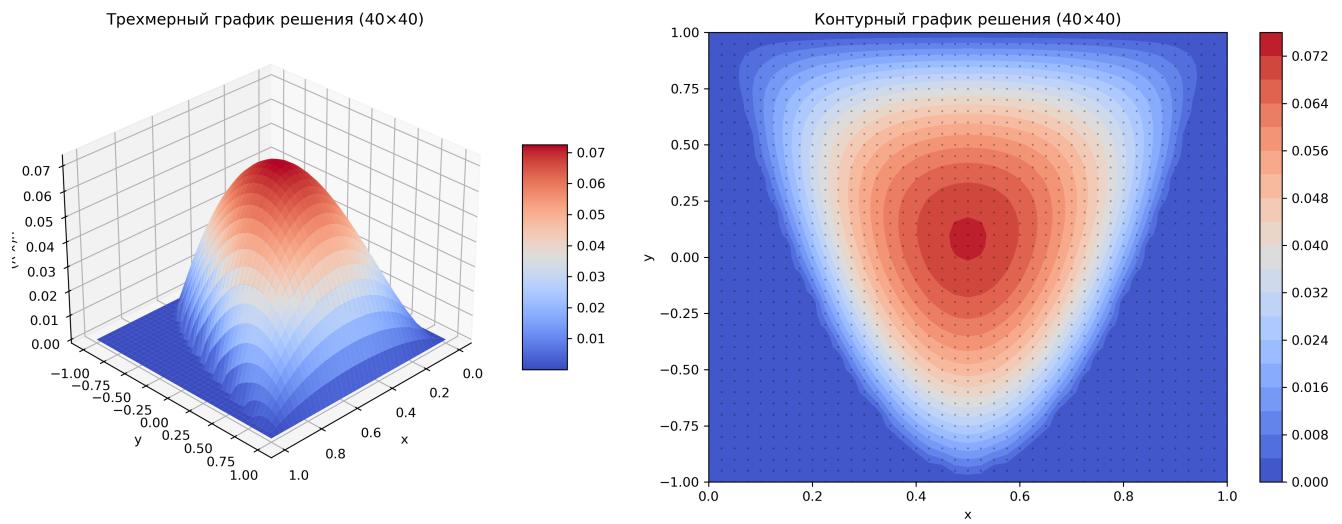


Рис. 7: График решения с сеткой 40×40 точек

Численное решение уравнения Пуассона на сетке 80×80

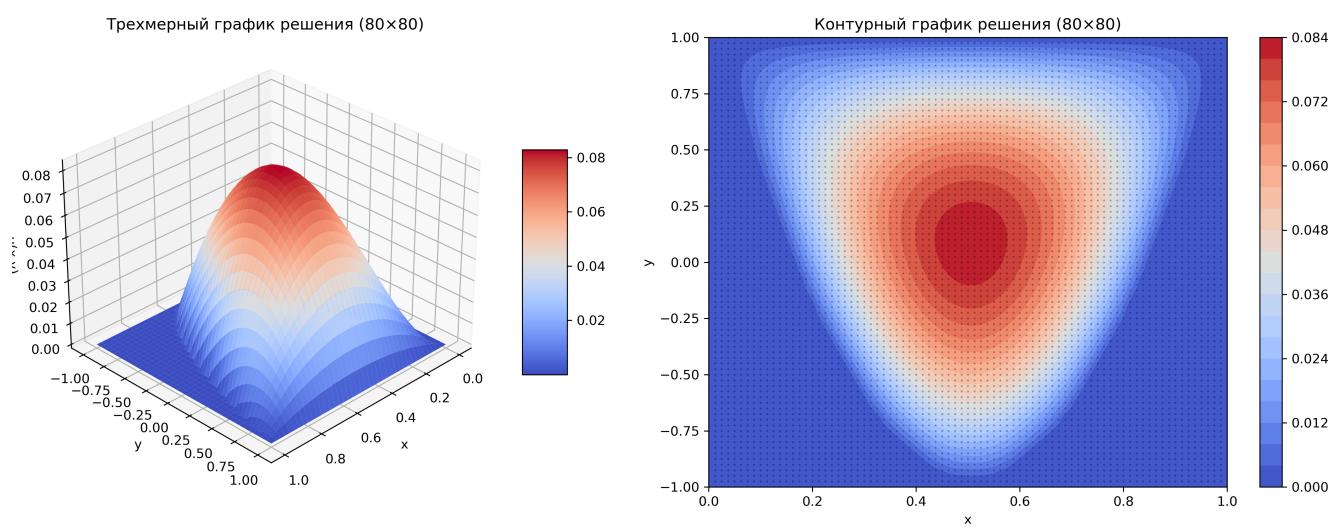


Рис. 8: График решения с сеткой 80×80 точек

Численное решение уравнения Пуассона на сетке 160×160

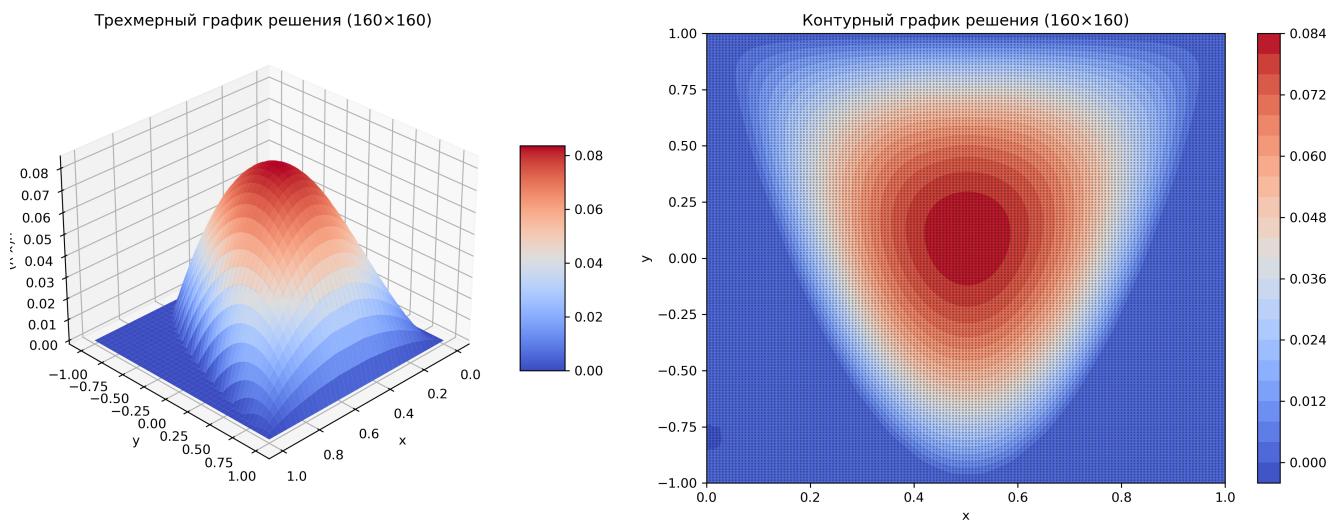


Рис. 9: График решения с сеткой 160×160 точек

0.9 Описание файлов и их назначения в проекте

Перечислю и объясню файлы, содержащиеся в представленном архиве: В представленном архиве содержатся следующие файлы:

- "**sequential.cpp**": Базовая версия программы без параллелизации, реализующая метод сопряженных градиентов для решения задачи Пуассона методом фиктивных областей.
- "**openmp.cpp**": OpenMP-версия программы, использующая метод наискорейшего спуска.
- "**openmp_cg.cpp**": OpenMP-версия программы, использующая метод сопряженных градиентов.
- "**mpi.cpp**": MPI-версия программы.
- "**mpi_omp.cpp**": MPI+OpenMP-версия программы.
- "**run_tests.sh**": Bash-скрипт для автоматизации тестирования программы на различных размерах сетки и количестве потоков. Собирает статистику производительности и создает лог-файлы с результатами.
- "**log/**" Папка содержит лог-файлы, сгенерированные в процессе проведения экспериментов. Все данные, представленные в отчёте, были получены из этих логов.
- "**Отчет_.pdf**": Отчет о проделанной работе, содержит описание реализации и анализ результатов.