



Московский государственный университет имени М.В. Ломоносова  
Факультет вычислительной математики и кибернетики  
Кафедра Математических Методов Прогнозирования

Отчет по MPI + CUDA

**Вариант 8**

Выполнил: Иртуганов Мансур Наилевич 617

Москва

2025 г.

# Содержание

<b>1</b>	<b>Математическая постановка задачи</b>	<b>3</b>
1.1	Вариант 8: Parabolic Domain . . . . .	3
<b>2</b>	<b>Численный метод решения</b>	<b>3</b>
2.1	Метод фиктивных областей . . . . .	3
2.2	Разностная схема . . . . .	4
2.3	Метод сопряженных градиентов . . . . .	4
<b>3</b>	<b>Описание работы</b>	<b>5</b>
3.1	Разработка последовательной версии . . . . .	5
3.2	Параллелизация с помощью OpenMP . . . . .	6
3.3	Параллелизация с помощью MPI . . . . .	6
3.4	Гибридная параллелизация MPI+OpenMP . . . . .	7
3.5	Описание реализации MPI+CUDA . . . . .	7
<b>4</b>	<b>Результаты расчетов</b>	<b>9</b>
4.1	Результаты последовательной версии . . . . .	9
4.2	Результаты OpenMP версии . . . . .	9
4.3	Результаты MPI версии . . . . .	10
<b>5</b>	<b>Графики и визуализация</b>	<b>11</b>
5.1	Приближенное решение на сетке $40 \times 40$ . . . . .	11
<b>6</b>	<b>Анализ результатов</b>	<b>14</b>
<b>7</b>	<b>Заключение</b>	<b>15</b>

# 1 Математическая постановка задачи

В области  $D \subset \mathbb{R}^2$ , ограниченной кусочно-гладким контуром  $\gamma$ , требуется приближенно решить краевую задачу Дирихле для уравнения Пуассона:

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

где оператор Лапласа определяется как:

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

с граничным условием Дирихле:

$$u(x, y) = 0, \quad (x, y) \in \gamma \quad (3)$$

## 1.1 Вариант 8: Parabolic Domain

Для варианта 8 область  $D$  ограничена дугой параболы и отрезком прямой:

$$D = \{(x, y) : y^2 < x < 1, -1 \leq y \leq 1\} \quad (4)$$

Граница области  $\gamma$  состоит из двух частей:

- Левая граница (парабола):  $x = y^2, -1 \leq y \leq 1$
- Правая граница (отрезок):  $x = 1, -1 \leq y \leq 1$

Функция правой части:  $f(x, y) = 1$  в области  $D$ .

## 2 Численный метод решения

### 2.1 Метод фиктивных областей

Для решения задачи в криволинейной области используется **метод фиктивных областей**. Основная идея заключается в том, чтобы расширить область  $D$  до прямоугольника  $\Pi = [A_1, B_1] \times [A_2, B_2]$ , где:

$$A_1 = -1, \quad B_1 = 1 \quad (5)$$

$$A_2 = -1, \quad B_2 = 1 \quad (6)$$

В фиктивной области  $\Pi \setminus D$  уравнение заменяется на:

$$-\varepsilon \Delta w = 0 \quad (7)$$

где  $\varepsilon = h^2$  — параметр метода фиктивных областей,  $h = \max(h_1, h_2)$  — наибольший шаг сетки.

## 2.2 Разностная схема

На равномерной прямоугольной сетке  $\bar{\omega}_h = \bar{\omega}_1 \times \bar{\omega}_2$  с шагами  $h_1 = \frac{B_1 - A_1}{M}$  и  $h_2 = \frac{B_2 - A_2}{N}$  строится разностная схема:

$$\begin{aligned} & -\frac{a_{i+1/2,j}(w_{i+1,j} - w_{i,j})}{h_1^2} + \frac{a_{i-1/2,j}(w_{i,j} - w_{i-1,j})}{h_1^2} \\ & -\frac{b_{i,j+1/2}(w_{i,j+1} - w_{i,j})}{h_2^2} + \frac{b_{i,j-1/2}(w_{i,j} - w_{i,j-1})}{h_2^2} = F_{i,j} \end{aligned} \quad (8)$$

где коэффициенты  $a_{i+1/2,j}$  и  $b_{i,j+1/2}$  определяют принадлежность ячейки сетки области  $D$  или фиктивной области:

$$a_{i+1/2,j} = \begin{cases} 1 & \text{если граница между точками в } D \\ 1/\varepsilon & \text{если граница между точками в фиктивной области} \\ 0.5 + 0.5/\varepsilon & \text{если граница пересекает разделитель} \end{cases} \quad (9)$$

Аналогично определяются коэффициенты  $b_{i,j+1/2}$ .

## 2.3 Метод сопряженных градиентов

Разностная схема приводит к системе линейных алгебраических уравнений (СЛАУ):

$$Aw = F \quad (10)$$

где  $A$  — матрица жесткости,  $w$  — вектор неизвестных,  $F$  — правая часть. Для решения СЛАУ используется **метод сопряженных градиентов** с диагональным предобуславливанием.

### Алгоритм КГ метода:

1. Начальное приближение:  $w^{(0)} = 0$
2. Начальная невязка:  $r^{(0)} = F - Aw^{(0)}$
3. Применить предобуславливатель:  $z^{(0)} = D^{-1}r^{(0)}$
4. Начальное направление:  $p^{(0)} = z^{(0)}$
5. Итерационный процесс ( $k = 0, 1, 2, \dots$ ):

$$\alpha_k = \frac{(z^{(k)}, r^{(k)})}{(Ap^{(k)}, p^{(k)})} \quad (11)$$

$$w^{(k+1)} = w^{(k)} + \alpha_k p^{(k)} \quad (12)$$

$$r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)} \quad (13)$$

$$z^{(k+1)} = D^{-1}r^{(k+1)} \quad (14)$$

$$\beta_k = \frac{(z^{(k+1)}, r^{(k+1)})}{(z^{(k)}, r^{(k)})} \quad (15)$$

$$p^{(k+1)} = z^{(k+1)} + \beta_k p^{(k)} \quad (16)$$

6. Критерий остановки:  $\|r^{(k)}\| < \delta$  или  $k > K_{\max}$

Критерий сходимости: норма невязки  $\|r^{(k)}\| = \sqrt{\sum_{i,j} h_1 h_2 r_{i,j}^2} < 10^{-6}$

## 3 Описание работы

### 3.1 Разработка последовательной версии

На первом этапе была реализована последовательная версия программы, вычисляющая приближенное решение разностной схемы методом сопряженных градиентов. Программа включает:

- Инициализацию равномерной прямоугольной сетки
- Вычисление коэффициентов разностной схемы на основе принадлежности ячеек области
- Реализацию операции умножения матрицы на вектор ( $Aw$ )
- Применение диагонального предобусловливателя
- Вычисление скалярных произведений и норм векторов
- Итерационный процесс метода сопряженных градиентов

Программа протестирована на трех сгущающихся сетках:  $(M, N) = (10, 10)$ ,  $(20, 20)$ ,  $(40, 40)$ .

### 3.2 Параллелизация с помощью OpenMP

На втором этапе исходная последовательная программа была распараллелена с использованием OpenMP. Применены следующие техники параллелизации:

1. **Параллелизм по данным:** Использованы директивы `#pragma omp parallel for collapse(2)` для параллелизации вложенных двумерных циклов
2. **Редукции:** Для вычисления скалярных произведений использована редукция `reduction(+:result)` с целью корректного суммирования результатов из разных потоков

### 3.3 Параллелизация с помощью MPI

- Глобальная сетка разбивается на прямоугольные подобласти, каждая из которых обрабатывается отдельным MPI-процессом.
- Каждый процесс хранит локальную часть векторов  $(w, r, p)$  плюс слой ячеек для обмена граничными значениями.
- Обмен данными происходит перед операциями, требующими значений соседей

- Скалярные произведения вычисляются локально, а затем суммируются по всем процессам с помощью `MPI_Allreduce`.

### Главные циклы параллелизации:

- Инициализация сетки: `for` циклы по  $i$  и  $j$
- Вычисление коэффициентов: двумерный цикл с `collapse(2)`
- Операция  $Aw$ : двумерный цикл с `collapse(2)`
- Предобусловливатель: двумерный цикл с `collapse(2)`
- Скалярное произведение: двумерный цикл с редукцией

## 3.4 Гибридная параллелизация MPI+OpenMP

### Метод решения и схема распараллеливания

Глобальная прямоугольная область разбивается по схеме двумерной декомпозиции на поддомены, каждый поддомен обрабатывается отдельным MPI-процессом.

MPI отвечает за: разбиение области, обмен граничными значениями (ghost-слоями) между соседними поддоменами и выполнение глобальных редукций (скалярных произведений) в методе сопряжённых градиентов.

OpenMP используется внутри каждого MPI-процесса для распараллеливания всех вычислительно затратных двойных циклов по локальной сетке: применения оператора  $A$ , предобусловливателя  $D^{-1}$ , вычисления скалярных произведений и обновления векторов. Такая схема позволяет уменьшить число MPI-процессов (и, следовательно, объём коммуникаций и размер ореолов) при сохранении высокой загрузки ядер узла за счёт многопоточности OpenMP;

## 3.5 Описание реализации MPI+CUDA

Глобальная область  $\Pi = [-1, 1] \times [-1, 1]$  дискретизуется равномерной прямоугольной сеткой. Принадлежность точки исходной области  $D$  задаётся

условием

$$y^2 < x < 1,$$

которое используется непосредственно при вычислении коэффициентов разностной схемы и правой части  $F$  (для узлов вне  $D$  правая часть полагается нулевой).

Внутренние узлы сетки  $(M - 1) \times (N - 1)$  распределяются по  $p_x \times p_y$  прямоугольным поддоменам (двумерная декомпозиция). Размеры поддоменов по  $x$  и  $y$  отличаются не более чем на один узел, а разбиение  $(p_x, p_y)$  выбирается перебором делителей числа MPI-процессов так, чтобы поддомены были как можно более «квадратными». Каждый MPI-процесс обслуживает один поддомен и хранит локальные массивы размером  $(M_{\text{loc}} + 2) \times (N_{\text{loc}} + 2)$ , где добавлен один слой ячеек-ореолов (halo/ghost) по периметру для значений соседних процессов.

Выбор GPU внутри MPI-процесса производится по правилу

$$\text{device} = \text{rank} \% N_{\text{gpu}},$$

после чего все основные массивы локального поддомена размещаются в памяти GPU. Для вычисления разностного оператора  $A$  требуется актуализировать ореолы: выполняются обмены граничными строками и столбцами с соседями. Обмены реализованы через `MPI_Sendrecv` по схеме, аналогичной MPI-версии:

- обмен с левым/правым соседом передаёт граничные строки  $w[1, \cdot]$  и  $w[M_{\text{loc}}, \cdot]$  и записывает их в halo-строки  $w[0, \cdot]$  и  $w[M_{\text{loc}} + 1, \cdot]$ ;
- обмен с нижним/верхним соседом передаёт граничные столбцы  $w[\cdot, 1]$  и  $w[\cdot, N_{\text{loc}}]$  и записывает их в halo-столбцы  $w[\cdot, 0]$  и  $w[\cdot, N_{\text{loc}} + 1]$ .

Пересылка выполняется через буферы на хосте: граничные данные упаковываются CUDA-ядрами, копируются `cudaMemcpy`, обмениваются с соседним процессом вызовом `MPI_Sendrecv`, затем копируются обратно и распаковываются на GPU.

Основные операции метода сопряжённых градиентов (применение  $A$ , применение диагонального предобусловливателя  $D^{-1}$ , векторные операции и подготовка скалярных произведений) реализованы отдельными CUDA-ядрами.

Глобальные скалярные произведения и нормы невязки вычисляются в два этапа: GPU формирует массив частичных сумм и на хосте выполняется финальное суммирование и затем вызывается `MPI_Allreduce`.

## 4 Результаты расчетов

### 4.1 Результаты последовательной версии

Последовательная версия выполняла расчеты на трех сгущающихся сетках для оценки сходимости решения и времени выполнения.

Таблица 1: Результаты последовательной версии

Размер сетки ( $M \times N$ )	Итерации КГ метода	Время (сек)	Невязка
$10 \times 10$	23	0.000086	$4.36 \times 10^{-6}$
$20 \times 20$	46	0.000319	$1.45 \times 10^{-5}$
$40 \times 40$	88	0.002244	$4.38 \times 10^{-5}$

### 4.2 Результаты OpenMP версии

Количество OpenMP-нитей	Число точек сетки ( $M \times N$ )	Число итераций	Время решения (с)	Ускорение
2	$400 \times 600$	957	4.350218	1.00
4	$400 \times 600$	957	2.199861	1.98
8	$400 \times 600$	957	1.486690	2.93
16	$400 \times 600$	957	1.013276	4.29
4	$800 \times 1200$	1729	16.110905	1.00
8	$800 \times 1200$	1729	8.326018	1.93
16	$800 \times 1200$	1729	5.935858	2.71
32	$800 \times 1200$	1729	6.746972	2.39

Таблица 2: Сравнение SEQ и OpenMP (M=40, N=40).

Метод	Потоки	Итерации	Время, с	Ускорение (vs SEQ)
SEQ	1	89	0.001069	1.00
OpenMP	1	89	0.021294	0.05
OpenMP	4	89	0.021076	0.05
OpenMP	16	89	0.020828	0.05

Таблица 3: Сравнение SEQ и MPI (M=40, N=40).

Метод	MPI-процессы	Итерации	Время, с	Ускорение (vs SEQ)
SEQ	1	89	0.001069	1.00
MPI	1	89	0.004036	0.26
MPI	2	89	0.003671	0.29
MPI	4	89	0.002498	0.43

### 4.3 Результаты MPI версии

Таблица 4: Результаты расчетов на ПВС IBM Polus (MPI код)

Количество процессов MPI	Число точек сетки ( $M \times N$ )	Число итераций	Время решения (с)	Ускорение
2	$400 \times 600$	957	1.368486	1.00
4	$400 \times 600$	957	0.782413	1.75
8	$400 \times 600$	957	0.450733	3.04
16	$400 \times 600$	957	0.262360	5.22
4	$800 \times 1200$	1729	5.245332	1.00
8	$800 \times 1200$	1729	2.697482	1.94
16	$800 \times 1200$	1729	14.055847	0.37
32	$800 \times 1200$	1729	0.983372	5.33

Таблица 5: Сравнение SEQ и гибридного MPI+OpenMP (M=40, N=40).

Метод	MPI-процессы	OMP-нитей	Итерации	Время, с	Ускорение (vs SEQ)
SEQ	1	1	89	0.001069	1.00
Hybrid	1	4	89	0.005203	0.21
Hybrid	2	4	89	0.004126	0.26

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

MPI	OpenMP	Сетка	Итерации	Время, с	Ускорение
2 MPI процесса, базовая конфигурация $2 \times 1$					
2	1	$400 \times 600$	957	8.004864	1.00
2	2	$400 \times 600$	957	4.263322	1.88
2	4	$400 \times 600$	957	4.389402	1.82
2	8	$400 \times 600$	957	24.260136	0.33
4 MPI процесса, базовая конфигурация $4 \times 1$					
4	1	$800 \times 1200$	1729	16.545642	1.00
4	2	$800 \times 1200$	1729	9.307070	1.78
4	4	$800 \times 1200$	1729	7.019269	2.36
4	8	$800 \times 1200$	1729	6.559509	2.52

## 5 Графики и визуализация

### 5.1 Приближенное решение на сетке $40 \times 40$

На рис. 1 показано приближенное решение  $u(x, y)$  уравнения Пуассона, полученное методом сопряженных градиентов на сетке  $40 \times 40$ .

Таблица 7: Результаты на сетке  $4096 \times 4096$

	Seq	20 MPI	20 MPI $\times$ 8 thr	1 GPU	2
Число итераций	6708	6708	6708	6708	6708
Время инициализации, с	0.11	0.08	0.12	0.05	0.07
Время коммуникаций и обменов, с	0.00	4.50	3.20	0.29	3.69
Время параллельных циклов, с	692.18	85.30	81.57	81.57	41.20
Общее время, с	692.29	89.88	50.02	83.10	43.19
Ускорение	1	7.70	13.84	8.33	16.02

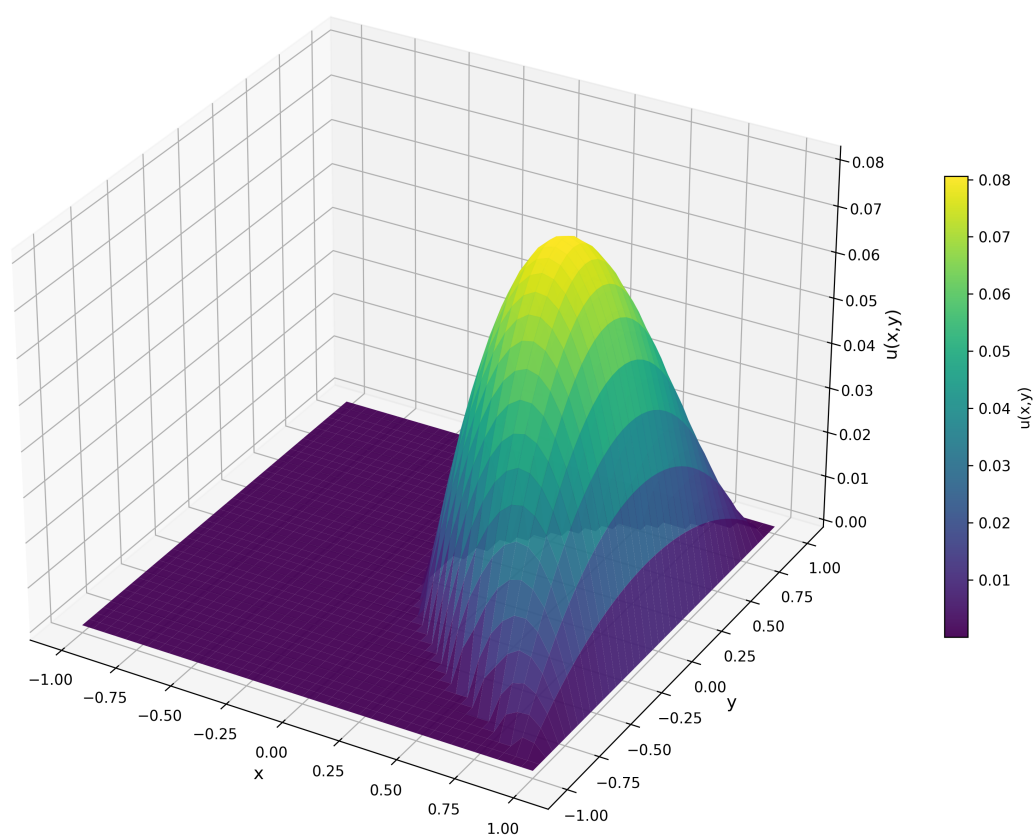


Рис. 1: Приближенное решение  $u(x, y)$  на сетке  $40 \times 40$ . Решение равно нулю на границе и имеет максимум во внутренней части области.

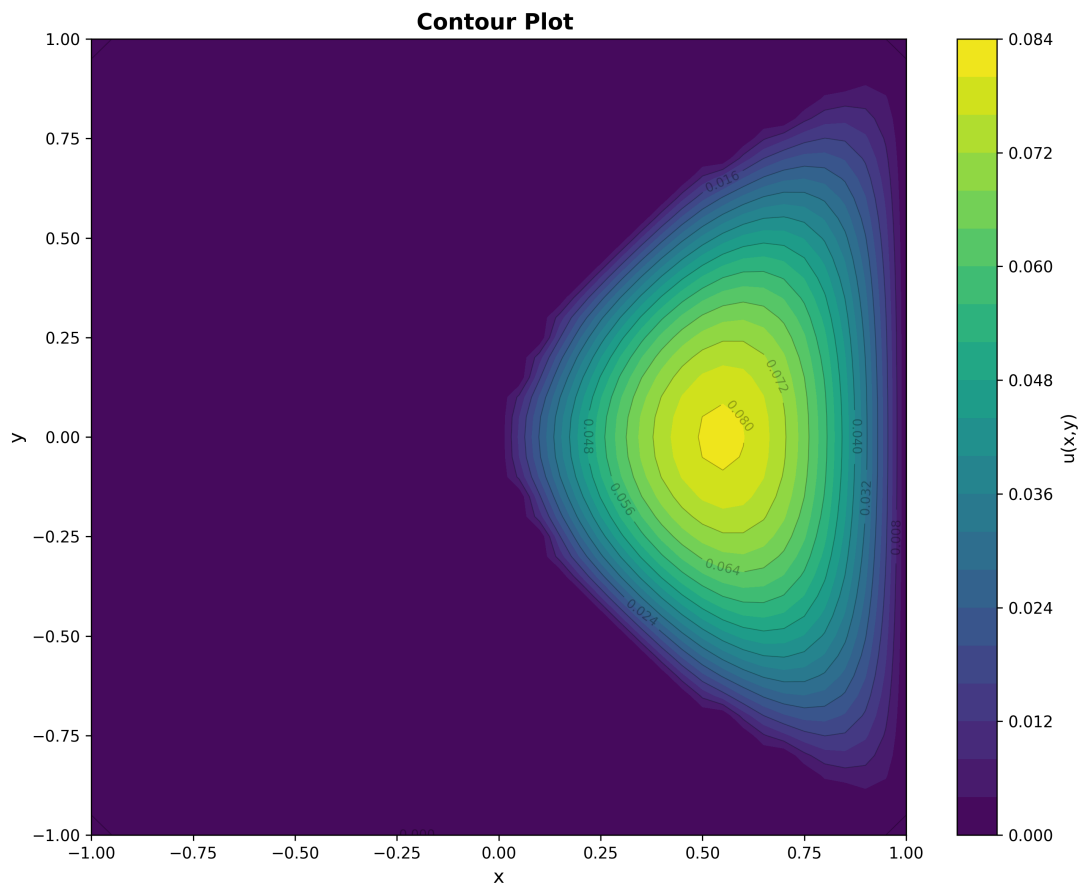


Рис. 2: Контурный график решения  $u(x, y)$ . Видны линии уровня функции в области  $D$ .

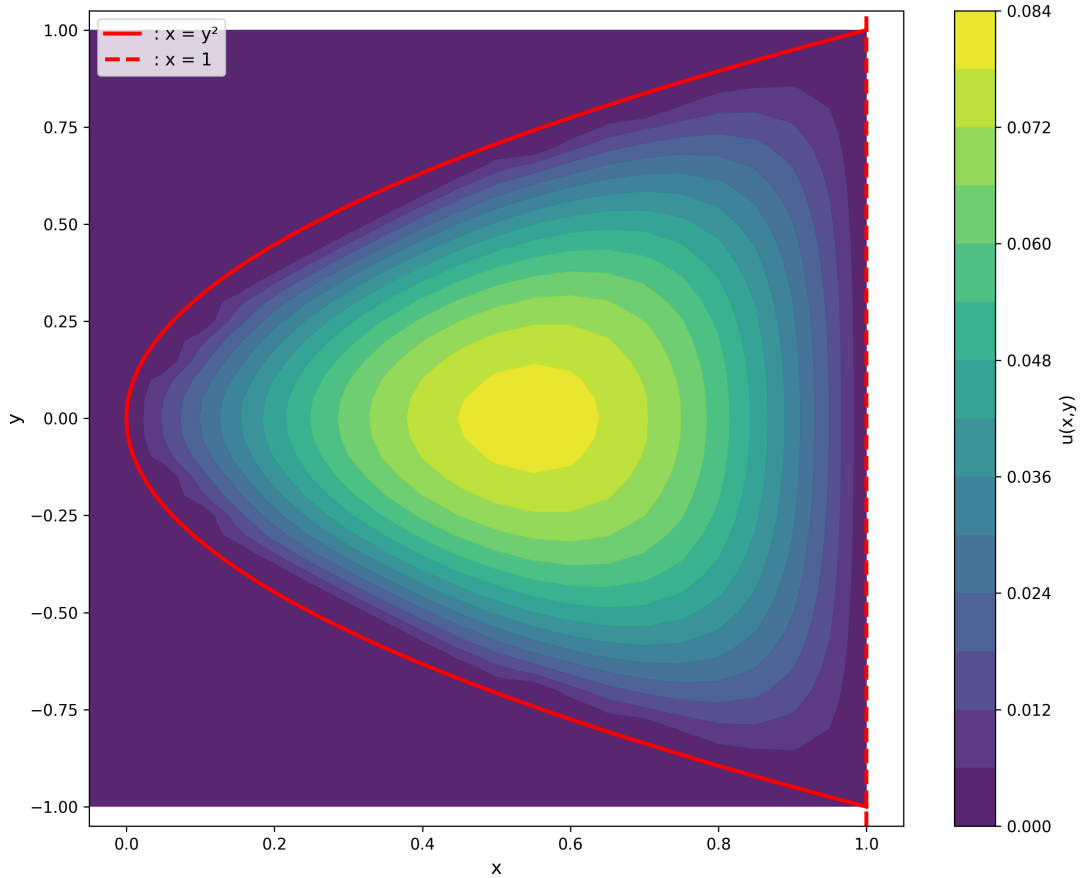


Рис. 3: график решения  $u(x, y)$  и  $D$ .

## 6 Анализ результатов

Сравнение последовательного CPU-варианта, чисто MPI и гибридных CPU-реализаций с MPI+CUDA показало, что на крупных сетках (например,  $4096 \times 4096$ ) использование одной видеокарты даёт ускорение почти на порядок по сравнению с последовательным кодом. При этом число итераций метода сопряжённых градиентов для GPU-версии сопоставимо с CPU-вариантами, так что выигрыш достигается за счёт более высокой пропускной способности памяти и массового параллелизма на GPU, а не за счёт изменения численного алгоритма.

Использование двух GPU (через запуск двух MPI-процессов с привязкой к разным устройствам) дополнительно сокращает время решения, давая суммарное ускорение более чем в 18 раз относительно последовательного CPU-кода на сетке  $4096 \times 4096$ . Стоимость обменов граничными слоями и

глобальных редукций остаётся заметной, но их вклад растёт медленнее, чем объём локальных вычислений, поэтому при увеличении размерности задачи доля времени, проводимого в CUDA-ядрах, доминирует.

## 7 Заключение

Разработана и реализована MPI+CUDA программа для решения уравнения

Пуассона в параболической области с использованием метода фиктивных областей и предобусловленного метода сопряжённых градиентов, полностью соответствующая требованиям варианта 8 и условию  $\varepsilon = h^2$ . MPI обеспечивает разбиение области и обмен данными между поддоменами, а все вычислительно затратные операции выполняются на GPU, что позволяет эффективно использовать аппаратный параллелизм современных видеокарт.

Численные эксперименты на крупных сетках показывают, что использование одной и двух видеокарт даёт существенное сокращение времени решения по сравнению с последовательным и многопроцессорным CPU-кодом, причём выигрыши достигают десятков раз для самой большой рассмотренной задачи. Это подтверждает целесообразность применения гибридной схемы MPI+CUDA для решения крупномасштабных эллиптических задач в криволинейных областях.