



**Нижегородский государственный университет  
им. Н.И. Лобачевского**

***Факультет Вычислительной математики и кибернетики***

# **Реализация явной разностной схемы решения уравнения теплопроводности с использованием CUDA**

**Горшков А.В., Бастратов С.И.**

**ВМК ННГУ**

**[anton.v.gorshkov@gmail.com](mailto:anton.v.gorshkov@gmail.com)**

# Постановка задачи

- Рассматривается задача Дирихле для двумерного уравнения теплопроводности:

$$\frac{\partial u(x, y, t)}{\partial t} = \Delta u(x, y, t) + f(x, y, t)$$

$$a \leq x \leq b, \quad c \leq y \leq d, \quad 0 \leq t \leq T$$

$$u(x, y, 0) = u_0(x, y)$$

$$\Gamma_y \left\{ \begin{array}{l} u(a, y, t) = \varphi_1(y, t), \quad u(b, y, t) = \varphi_2(y, t) \\ u(x, c, t) = \varphi_3(x, t), \quad u(x, d, t) = \varphi_4(x, t) \end{array} \right.$$



# Постановка задачи

- Введем равномерную сетку с шагами по пространству и времени  $\Delta x$ ,  $\Delta y$  и  $\Delta t$  соответственно.
- Обозначим  $v^{(k)}_{ij}$  – сеточное значение в точке  $(x_i, y_j)$  в  $k$ -й момент времени.
- Будем использовать следующую явную разностную схему для внутренних узлов:

$$\begin{aligned} v^{(k+1)}_{ij} &= v^{(k)}_{ij} + \Delta t \\ &\quad * \left( f(x_i, y_j) + \frac{v^{(k)}_{i+1j} - 2v^{(k)}_{ij} + v^{(k)}_{i-1j}}{\Delta x^2} \right. \\ &\quad \left. + \frac{v^{(k)}_{ij+1} - 2v^{(k)}_{ij} + v^{(k)}_{ij-1}}{\Delta y^2} \right) \end{aligned}$$



# Постановка задачи

---

- ❑ Будем производить вычисления слой за слоем по времени. Вычисления всех элементов одного слоя по времени независимы.
- ❑ Для простоты будем считать, что граничные условия стационарны (не зависят от времени). Это означает, что при программной реализации функции вычисления значений в очередном слое будем вычислять лишь значения во внутренних узлах.



# Задание

- ❑ Ознакомьтесь с предоставленным проектом `CUDA_HeatEquation`, убедитесь в том, что он компилируется, запускается и выдает “Test PASSED.”
- ❑ Реализуйте GPU версию функции `heat_equation`, не меняя ее интерфейс:
  - имеет смысл добавить в проект файлы `heat_equation_gpu.h` и `heat_equation_gpu.cu`
  - реализуйте ядро, которое вычисляет новые значения в одном слое по времени (выполняет работу аналогично функции `step` в `heat_equation_cpu.cpp`)
  - не переделывайте функцию `step` в ядро – она нужна для других целей
  - пусть каждый поток вычисляет один элемент, используя двумерные индексы



# Задание

- для вызова функции правой части из ядра создайте аналог функции `f` из `heat_equation_cpu.cpp` со спецификатором `__device__`
- реализуйте функцию `step_gpu` в файле `heat_equation_gpu.cu`, которая вычисляет количество блоков и потоков и запускает ядро. Не меняйте интерфейс функции
- рекомендуется использовать блоки фиксированного размера, например, 16 на 16.
- убедитесь, что полученный результат совпадает с результатом CPU-версии, в этом случае будет выведено сообщение “Test PASSED.”

