Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчёт СКМИТ

Ковч Н. С.

609 гр.

Москва, 2025

Содержание

Математическая постановка задачи

В области D, ограниченной треугольником γ с вершинами в точках $(-3,0),(3,0),(0,2)$, рассматривается дифференциальное уравнение Пуассона:

$$-\Delta u = 1, \quad (x,y) \in \operatorname{int} D. \tag{1}$$

Решение уравнения дополняется граничным условием Дирихле:

$$u(x,y) = 0, \quad (x,y) \in \gamma. \tag{2}$$

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

Метод фиктивных областей

Для приближенного решения задачи (1), (2) был использован метод фиктивных областей. Область D принадлежит прямоугольнику

$$\Pi = \{(x,y) : -3 < x < 3, 0 < y < 2\}$$

Область $\hat{D} = \Pi \setminus \overline{D}$ называется фиктивной областью. Выберем и зафиксируем малое $\epsilon > 0$. В прямоугольнике Π рассматривается задача Дирихле:

$$-\frac{\partial}{\partial x}(k(x,y)\frac{\partial v}{\partial x}) - \frac{\partial}{\partial y}(k(x,y)\frac{\partial v}{\partial y}) = F(x,y), \quad (x,y) \in \Pi \setminus \gamma \tag{3}$$

$$v(x,y) = 0, \quad (x,y) \in \Gamma. \tag{4}$$

с кусочно-постоянным коэффициентом

$$k(x,y) = \begin{cases} 1, & (x,y) \in D, \\ \dfrac{1}{\epsilon}, & (x,y) \in \hat{D} \end{cases} \tag{5}$$

и правой частью

$$F(x,y) = \begin{cases} 1, & (x,y) \in D, \\ 0, & (x,y) \in \hat{D} \end{cases} \tag{6}$$

Здесь Γ – граница прямоугольника Π. Требуется найти непрерывную в $\overline{\Pi}$ функцию $v(x,y)$, удовлетворяющую дифференциальному уравнению задачи (3) всюду в $\Pi \setminus \gamma$, равную нулю на границе Γ прямоугольника, и такую, чтобы вектор потока

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

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

Разностная схема решения задачи

Краевая задача решается численно методом конечных разностей. В $\overline{\Pi}$ определяется равномерная прямоугольная сетка $\overline{\omega_h} = \overline{\omega_1} \times \overline{\omega_2}$:

$$\overline{\omega_1} = \{x_i = -3 + ih_x, i = \overline{0, M}\}, \quad \overline{\omega_2} = \{y_j = jh_y, j = \overline{0, N}\}, \tag{7}$$

где $h_x = \frac{6}{M}, h_y = \frac{2}{N}$. Через ω_h обозначим множество внутренних узлов сетки $\overline{\omega_h}$, т.е. множество узлов сетки прямоугольника, не лежащих на границе Г. Рассматривается линейное пространство H функций, заданных на сетке ω_h. Обозначим через w_{ij} значение сеточной функции $w \in H$ в узле сетки $(x_i, y_j) \in \omega_h$. В пространстве H задано скалярное произведение и евклидова норма

$$(u, v) = \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} h_x h_y u_{ij} v_{ij}, \quad ||u||_E = \sqrt{(u, u)}. \tag{8}$$

Дифференциальное уравнение задачи (3) во всех внутренних точках сетки $\overline{\omega_h}$ аппроксимируется разностным уравнением

$$-\frac{1}{h_x}(a_{i+1j}\frac{w_{i+1j} - w_{ij}}{h_x} - a_{ij}\frac{w_{ij} - w_{i-1j}}{h_x}) - \frac{1}{h_y}(b_{ij+1}\frac{w_{ij+1} - w_{ij}}{h_y} - b_{ij}\frac{w_{ij} - w_{ij-1}}{h_y}) = F_{ij},$$
$$i = \overline{1, M-1}, j = \overline{1, N-1}, \tag{9}$$

в котором коэффициенты

$$a_{ij} = \frac{1}{h_y}\int_{y_{j-1/2}}^{y_{j+1/2}} k(x_{i-1/2}, t)dt, \quad b_{ij} = \frac{1}{h_x}\int_{x_{i-1/2}}^{x_{i+1/2}} k(t, y_{j-1/2})dt \tag{10}$$

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

$$x_{i\pm1/2} = x_i \pm h_x/2, \quad y_{j\pm1/2} = y_j \pm h_y/2. \tag{11}$$

Правая часть разностного уравнения

$$F_{ij} = \frac{1}{h_x h_y}\int_{\Pi_{ij}} F(x, y)dxdy, \quad \Pi_{ij} = \{(x, y) : x_{i-1/2} \le x \le x_{i+1/2}, y_{j-1/2} \le y \le y_{j+1/2}\} \tag{12}$$

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

Краевые условия Дирихле аппроксимируются точно равенством

$$w_{ij} = w(x_i, y_j) = 0, \quad (x_i, y_j) \in \Gamma. \tag{13}$$

Введя стандартные обозначения для разностной производной вперед и назад, разностная схема (9) записывается в следующем виде

$$-(aw_{\overline{x}})_{x,ij} - (bw_{\overline{y}})_{y,ij} = F_{ij}, \quad i = \overline{1, M-1}, j = \overline{1, N-1}. \tag{14}$$

Получаем систему линейный уравнений

$$Aw = -(aw_{\overline{x}})_x - (bw_{\overline{y}})_y = F. \tag{15}$$

Здесь A – самосопряженный и положительно определенный оператор.

Метод решения СЛАУ

Приближенное решение разностной схемы (9) получено итерационным методом сопряженных градиентов. Для ускорения сходимости метода применяется диагональное предобуславливание. Оператор $D : H \to H$ действует на сеточные функции $w \in H$ по правилу

$$(Dw)_{ij} = \left[(a_{i+1j} + a_{ij})/h_x^2 + (b_{ij+1} + b_{ij})/h_y^2\right] w_{ij}, \quad i = \overline{1, M-1}, j = \overline{1, N-1}. \tag{16}$$

Начальное приближение $w^{(0)}$ выбирается нулевым. Первая итерация совершается по формулам скорейшего спуска. Пусть $r^{(0)} = B - Aw^{(0)} = B$ – невязка начального приближения, а функция $z^{(0)} \in H$ удовлетворяет уравнению $Dz^{(0)} = r^{(0)}$.
Направление спуска $p^{(1)} = z^{(0)}$, а шаг вдоль направления спуска определяется параметром

$$\alpha_1 = \frac{(z^{(0)}, r^{(0)})}{(Ap^{(1)}, p^{(1)})}. \tag{17}$$

Следующее приближение $w^{(1)}$ вычисляется согласно равенству

$$w^{(1)} = w^{(0)} + \alpha_1 p^{(1)}. \tag{18}$$

Дальнейшие вычисления производятся по следующим формулам. Пусть выполнено k итераций метода и функции $r^{(k-1)}, z^{(k-1)}, p^{(k)}, w^{(k)} \in H$, а также коэффициент α_k являются известными.

$$r^{(k)} = r^{(k-1)} - \alpha_k Ap^{(k)}, \quad Dz^{(k)} = r^{(k)}. \tag{19}$$

$$p^{(k+1)} = z^{(k)} + \beta_{k+1} p^{(k)}, \quad \beta_{k+1} = \frac{(z^{(k)}, r^{(k)})}{(z^{(k-1)}, r^{(k-1)})}. \tag{20}$$

$$w^{(k+1)} = w^{(k)} + \alpha_{k+1} p^{(k+1)}, \quad \alpha_{k+1} = \frac{(z^{(k)}, r^{(k)})}{(Ap^{(k+1)}, p^{(k+1)})}. \tag{21}$$

В качестве условия остановки итерационного процесса использовано неравенство

$$||w^{(k+1)} - w^{(k)}||_E < \delta \tag{22}$$

Метод сопряженных градиентов является методом вариационного типа, в основе которого находится задача минимизации квадратичного функционала

$$J(w) = 0.5(Aw, w) - (B, w), \tag{23}$$

который должен монотонно убывать на итерационной последовательности $w^{(k)}$, $k = 0, 1, \ldots$

Последовательный код и результаты

Для получения коэффициентов F_{ij} было использовано упрощенное правило: если клетка Π_{ij} целиком содержится в области $\overline{D}$, то $F_{ij} = 1$, иначе $F_{ij} = 0$. До этого использовалась формула шнурования и Sutherland-Hodgman algorithm для расчета площади пересечения. В качестве констант δ и ϵ взяты $1e-10$ и $1/\max(h_x^2, h_y^2)$ соответственно.

Ниже представленны графики приближенных решений, а также соответствующие им графики энергетических функционалов $J(w^{(k)})$ и норм $||w^{(k+1)} - w^{(k)}||_E$ на сгущающихся сетках. Для построения графиков были использованы библиотеки matplotlib и plotly языка Python.

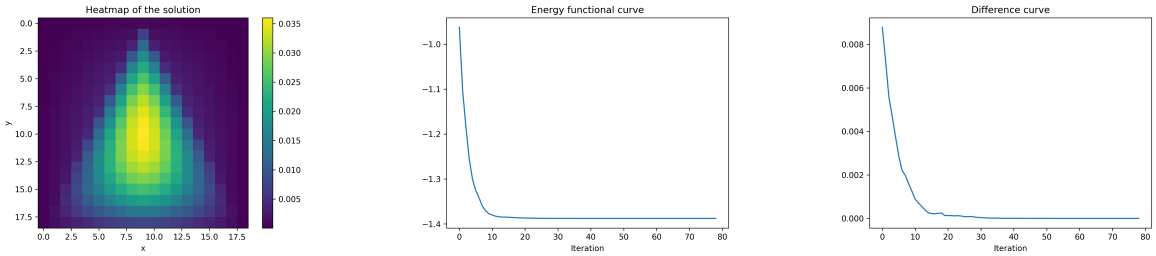


Рис. 1: $M = 20, N = 20$

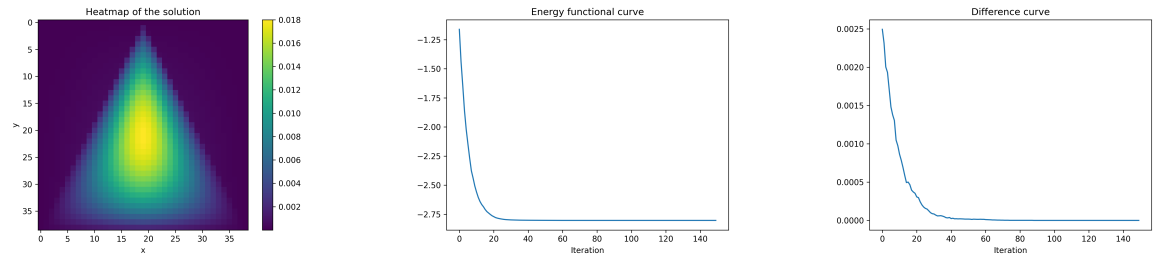


Рис. 2: $M = 40, N = 40$

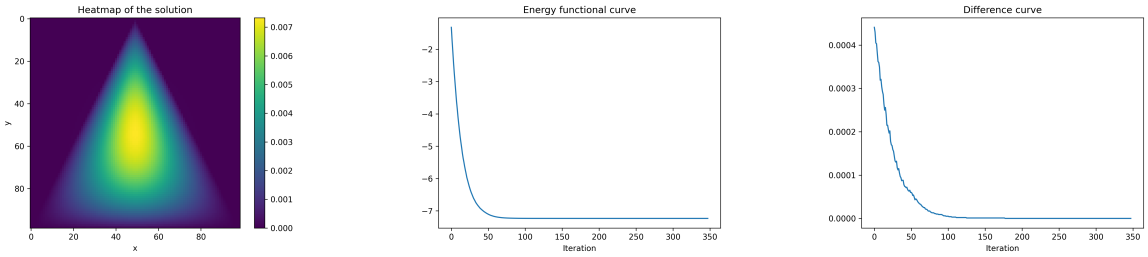


Рис. 3: $M = 100, N = 100$

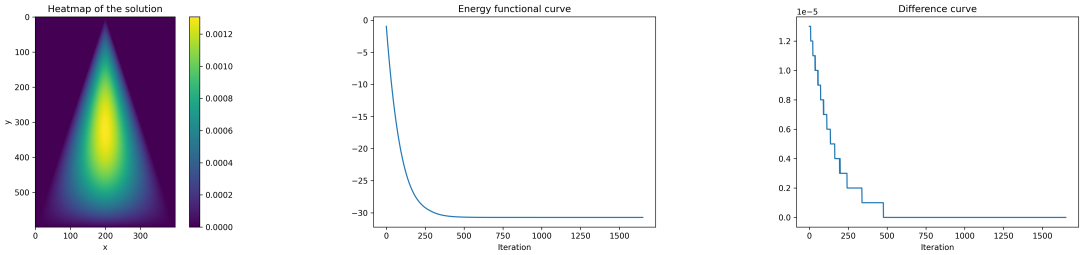


Рис. 4: $M = 400, N = 600$

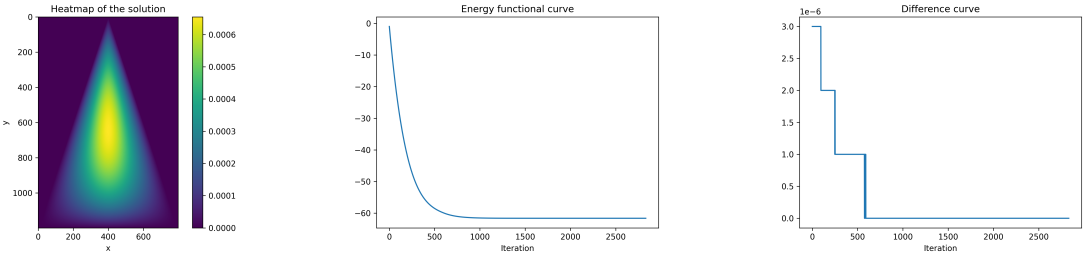


Рис. 5: $M = 800, N = 1200$

Для вычисления времени работы последовательной программы использовался тип `clock_t` и функция `clock()`. Ниже приведены времена вычисления программы на сетках $M = 400, N = 600$ и $M = 800, N = 1200$:

M × N	**Число итераций**	**Время решения**
400×600	1651	5.9
800×1200	2833	43.27

OpenMP версия

Для распараллеливания кода использовались OpenMP директивы

1. `omp parallel for simd` для простых циклов: вычисление сеточной функции z, вычисление суммы двух векторов.

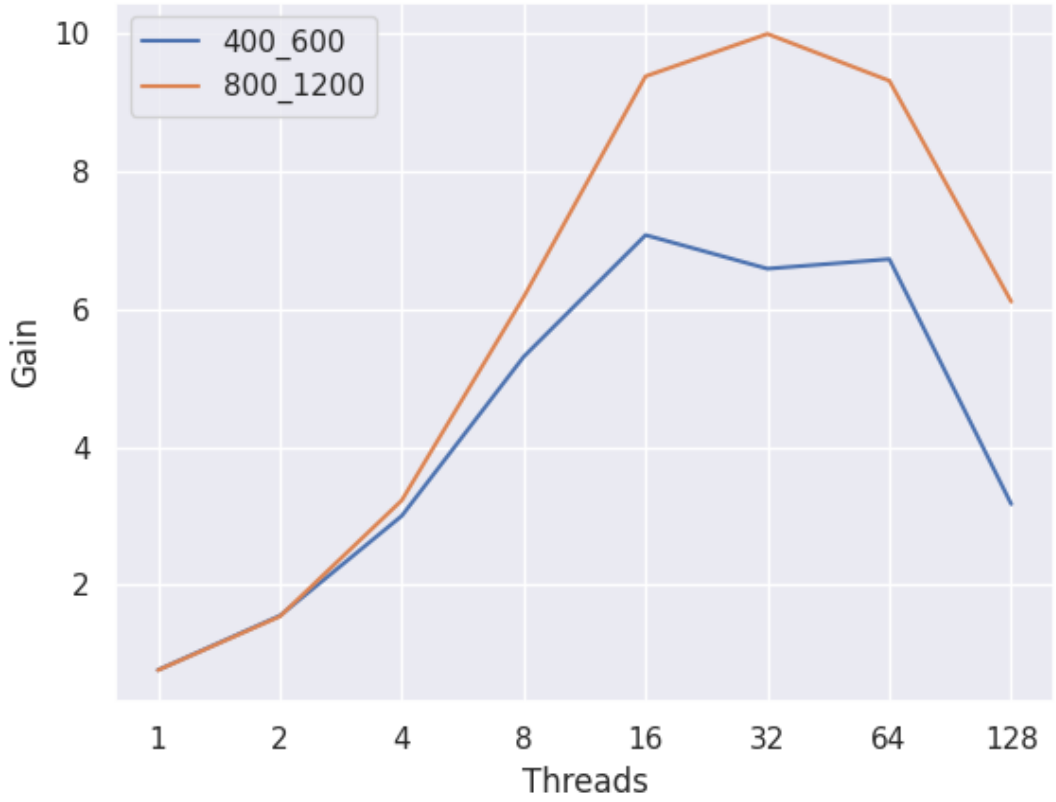
2. `omp parallel for simd reduction(+:x)` для вычисления скалярного произведения (`x = dot_product`) и для вычисления функционала энергии (`x = energy`).

3. `omp parallel for collapse(2)` для вложенных циклов: вычисление параметров a_{ij}, b_{ij}, F_{ij}, для вычисления элементов матрицы D, а также для вычисления величины $Ap^{(k)}$.

Для вычисления времени работы последовательной программы использовался тип `clock_t` и функция `clock()`, а для параллельной – функция `omp_get_wtime()`. Ниже приведена таблица, содержащая информацию о времени вычисления решения на сетках $M = 400, N = 600$ и $M = 800, N = 1200$.

OpenMP-нити	**M × N**	**Число итераций**	**Время решения**	**Ускорение**
1	400 × 600	1651	7.70137	0.766
2	400 × 600	1651	3.79998	1.552
4	400 × 600	1651	1.96632	3.005
8	400 × 600	1651	1.11156	5.307
16	400 × 600	1651	0.83429	7.072
32	400 × 600	1651	0.89607	6.584
64	400 × 600	1651	0.87758	6.723
128	400 × 600	1651	1.86056	3.171
1	800 × 1200	2833	56.73835	0.762
2	800 × 1200	2833	28.05178	1.543
4	800 × 1200	2833	13.41475	3.225
8	800 × 1200	2833	7.00864	6.174
16	800 × 1200	2833	4.61571	9.374
32	800 × 1200	2833	4.33175	9.989
64	800 × 1200	2833	4.64785	9.309
128	800 × 1200	2833	7.08512	6.107

Графики ускорения:



MPI версия

Необходимо разработать двумерное разбиение прямоугольника Π на подобласти так, чтобы

1. отношение количества узлов по переменным x и y в каждой подобласти принадлежало диапазону $[0.5, 2]$

2. количество узлов по переменным x и y любых двух подобластей отличалось не более, чем на единицу

В работе были использованы одинаковые прямоугольные подобласти:

1	2	4	8	16	32
400×600	400×300	200×300	200×150	100×150	100×75
800×1200	800×600	400×600	400×300	200×300	200×150

Каждый MPI процесс вычисляет коэффициенты $a_{ij}, b_{ij}, F_{ij}, D_{ij}$ в своей подобласти и производит расчёт необходимых величин для итерации метода сопряженных градиентов. При вычислении произведения вектора p на матрицу A процессы производят обмен граничными узлами (так называемые ghost cells) при помощи неблокирующих MPI функций `MPI_Isend()` и `MPI_Irecv()`. Для корректного сохранения отправленных границ были использованы тэги `TAG_LEFT, TAG_RIGHT, TAG_TOP, TAG_BOTTOM`. Для редукции глобального направления спуска, погрешности решения, а также величины шага использовалась функция `MPI_Allreduce()`.

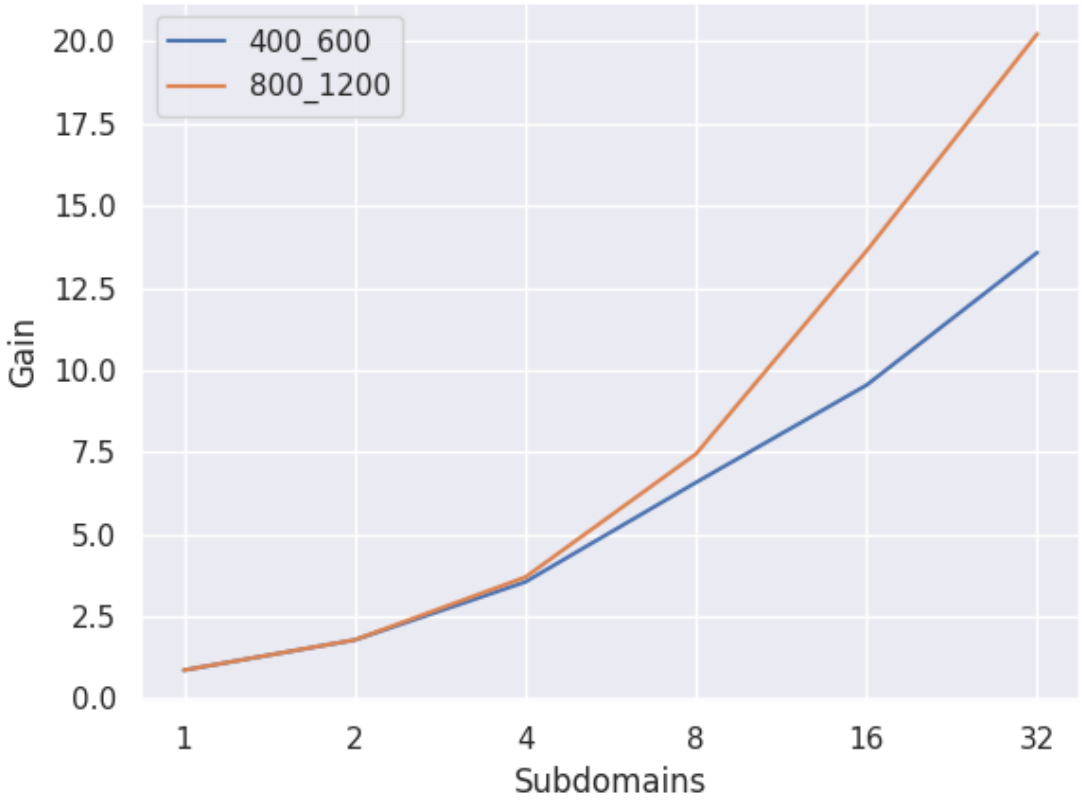
Сохранение энергетического функционала, глобального решения, а также погрешности решения поручено корневому процессу (процессу с рангом 0). Стоит отметить, что глобальное решение собирается функцией `MPI_Gather()` и сохраняется в разобранном виде (следствие выбранного мной способа нумерации подобластей, а также хранения локальных решений в формате `column-major`). Было решено осуществлять корректную расстановку решений в сопровождающем скрипте на языке Python, при помощи которого строятся графики.

Время выполнения программы производилось при помощи функций `MPI_Barrier()` и `MPI_Wtime()`. Из всех времен выбрано максимальное: `MPI_Reduce` с параметром `MPI_MAX`.

Ниже приведена таблица, содержащая информацию о времени решения:

MPI-процессы	**M × N**	**Число итераций**	**Время решения**	**Ускорение**
1	400 × 600	1651	6.798383	0.868
2	400 × 600	1651	3.301295	1.787
4	400 × 600	1651	1.660651	3.553
8	400 × 600	1651	0.896605	6.580
16	400 × 600	1651	0.618166	9.544
32	400 × 600	1651	0.434533	13.577
1	800 × 1200	2833	49.731149	0.870
2	800 × 1200	2833	24.177078	1.789
4	800 × 1200	2833	11.699822	3.698
8	800 × 1200	2833	5.810977	7.446
16	800 × 1200	2833	3.174480	13.630
32	800 × 1200	2833	2.138661	20.232

Графики ускорения:



Гибридная версия: OpenMP + MPI

В MPI версию были добавлены OpenMP директивы, перечисленные в соответствующем разделе. Ниже приведена таблица с временем выполнения:

OpenMP-нити	M × N	Число итераций	Время решения	Ускорение
1	400 × 600	1651	2.522533	2.339
2	400 × 600	1651	2.081140	2.835
4	400 × 600	1651	1.327137	4.445
8	400 × 600	1651	0.937091	6.296
1	800 × 1200	2833	9.286683	4.659
2	800 × 1200	2833	7.398702	5.849
4	800 × 1200	2833	4.040481	10.709
8	800 × 1200	2833	2.095163	20.652

Графики ускорений: