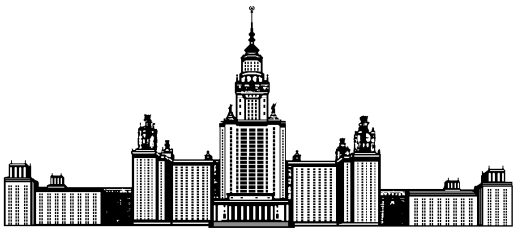

Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Вершков Станислав Александрович

Задание по курсу «Суперкомпьютерное моделирование и технологии»

Численное решение краевой задачи для уравнения Пуассона

ОТЧЕТ

Москва, 2022

Оглавление

1 Математическая постановка задачи

Вариант 2.

Требуется методом конечных разностей приближенно решить краевую задачу для уравнения Пуассона с потенциалом в прямоугольной области.

В прямоугольнике $\Pi = [0, 4] \times [0, 3]$ рассматривается дифференциальное уравнение Пуассона с потенциалом

$$-\Delta u + q(x, y)u = F(x, y), \tag{1}$$

в котором оператор Лапласа

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

и функции заданы:

$$k(x, y) = 4 + x + y,\ q(x, y) = x + y.$$

Для выделения единственного решения уравнение дополняется граничными условиями. На отрезке нижней границы прямоугольника Π задается условие третьего типа:

$$(k\frac{\partial u}{\partial n})(x, y) + \alpha u(x, y) = \psi(x, y),$$

где n – единичная внешнаяя нормаль к границе прямоугольника. Заметим, что нормаль n не определена в угловых точках прямоугольника. Коэффициент α предлагается взять равным 1. На остальных границах прямоугольника используется граничное условие первого типа:

$$u(x, y) = \phi(x, y).$$

Функции $F(x,y), \phi(x,y), \psi(x,y)$ получены аналитически.

$$F(x,y) = \frac{(4+x+y)(x^2+y^2)-2(x+y)(4+xy)}{4(4+xy)^{\frac{3}{2}}} + (x+y)\sqrt{4+xy},$$

$$\phi(x,y) = u(x,y),$$

$$\psi(x,y) = \frac{x(4+x+y)}{2\sqrt{4+xy}} + \sqrt{4+xy}.$$

2 Численный метод решения задачи

Краевую задачу для уравнения Пуассона с потенциалом 1 предлагается численно решить методом конечных разностей. В расчетной области П определяется равномерная прямоугольная сетка $\tilde{\omega_h} = \tilde{\omega_1} \times \tilde{\omega_2}$, где

$$\tilde{\omega_1} = \{x_i = ih_1, i = 0 \dots M\},\ \tilde{\omega_2} = \{y_j = jh_2, j = 0 \dots N\}.$$

Здесь $h_1 = \frac{4}{M}$, $h_2 = \frac{3}{N}$. Через ω_h обозначим множество внутренних узлов сетки $\tilde{\omega_h}$, т.е. множество узлов сетки прямоугольника, не лежащих на границе Г.

Рассмотрим линейное пространство H функций, заданных на сетке $\tilde{\omega_h}$. Обозначим через ω_{ij} значение сеточной функции $\omega \in H$ в узле сетки. Будем считать, что в пространстве H задано скалярное произведение и евклидова норма

$$??[u,v] = \sum_{i=0}^{M} h_1 \sum_{j=0}^{N} h_2 \rho_{ij} u_{ij} v_{ij}, |u| = \sqrt{[u,u]}. \quad (2)$$

Весовая функция $\rho_{ij} = \rho^{(1)}(x_i)\rho^{(2)}(y_j)$, где

$$\rho^{(1)}(x_i) = \begin{cases} 1, 1 \le i \le M-1 \\ \frac{1}{2}, i = 0, i = M \end{cases} \quad \rho^{(2)}(y_j) = \begin{cases} 1, 1 \le j \le N-1 \\ \frac{1}{2}, j = 0, j = N \end{cases}$$

В методе конечных разностей дифференциальная задача математической физики заменяется конечно-разностной операторной задачей вида

$$A\omega = B, \tag{3}$$

где $A : H \to H$ – оператор, действующий в пространстве сеточных функций, $B \in H$ – известная правая часть. Задача 3 называется разностной схемой. Решение этой задачи считается численным решением исходной задачи.

Уравнение 1 во всех внутренних точках сетки аппроксимируется разностным уравнением

$$-\Delta_h \omega_{ij} + q_{ij}\omega_{ij} = F_{ij}, i = 1 \dots M-1, j = 1 \dots N-1, \tag{4}$$

в котором $F_{ij} = F(x_i, y_j)$, $q_{ij} = q(x_i, y_j)$.

Введем обозначения правой и левой разностных производных по переменным x, y соответственно:

$$\omega_{x,ij} = \frac{\omega_{i+1j} - \omega_{ij}}{h1}, \ \omega_{\tilde{x},ij} = \frac{\omega_{ij} - \omega_{i-1j}}{h1},$$

$$\omega_{y,ij} = \frac{\omega_{ij+1} - \omega_{ij}}{h2}, \ \omega_{\tilde{y},ij} = \frac{\omega_{ij} - \omega_{ij-1}}{h2},$$

а также определим сеточные коэффициенты

$$a_{ij} = k(x_i - 0.5h_1, y_j), \ b_{ij} = k(x_i, y_j - 0.5h_2).$$

С учетом принятых обозначений разностый оператор Лапласа представляется как

$$\Delta_h \omega_{ij} = (a\omega_{\tilde{x}})_{x,ij} + (b\omega_{\tilde{y}})_{y,ij}.$$

Краевые условия первого типа аппроксимируются равенством

$$\omega_{ij} = \phi(x_i, y_j).$$

Краевое условие третьего типа для нижней стороны аппроксимируется следующими равенствами

$$-(2/h2)(b\omega_{\tilde{y}})_{i1} + (q_{i0} + 2/h2)\omega_{i0} - (a\omega_{\tilde{x}})_{x,i0} = F_{i0} + (2/h_2)\psi_{i0},\ i = 1 \ldots M-1.$$

Краевые условия для угловых точек выбраны следующим образом

$$\omega_{00} = \phi(0,0),\ \omega_{M0} = \phi(M,0),\ \omega_{0N} = \phi(0,N),\ \omega_{MN} = \phi(M,N).$$

Приближенное решение системы уравнений 3 для сформулированных выше краевых задач получается с помощью итерационного метода наименьших невязок. Этот метод позволяет получить последовательность сеточных функций $\omega^{(k)} \in H, k = 1, 2, \ldots$, сходящуюся по норме пространства H к решению разностной схемы, то есть

$$|\omega - \omega^{(k)}| \to 0,\ k \to +\infty.$$

Начальное приближение ω^0 выбирается равным нулю во всех точках.

Итерация $\omega^{(k+1)}$ вычисляется по итерации $\omega^{(k)}$ согласно равенствам:

$$\omega_{ij}^{(k+1)} = \omega_{ij}^{(k)} - \tau_{k+1} r_{ij}^{(k)},$$

где невязка $r^{(k)} = A\omega^{(k)} - B$, итерационный параметр

$$\tau_{k+1} = \frac{[Ar^{(k)}, r^{(k)}]}{|Ar^{(k)}|^2}.$$

В качестве условия остановки итерационного процесса выбрано неравенство

$$|w^{(k+1)} - w^{(k)}| < \epsilon,$$

где ϵ – положительное число, определяющее точность итерационного метода.

3 Описание MPI программы и гибридной реализации MPI/OpenMP

Для решения задачи с использованием технологии MPI рассматриваемая область разбивается на подобласти прямоугольной формы, причем число подобластей равно числу процессов. В каждой из них отношение θ количества узлов по ширине и длине удовлетворяет неравенстам $0.5 \leq \theta \leq 2$. Используется вызов функции *MPI_ Cart_ create*, которая возвращает новый коммуникатор. Далее, каждый процесс, используя функции *MPI_ Cart_ coords* и *MPI_ Cart_ shift* получает свое местоположение в сетке процессов, ранги соседей и расчитывает диапазон точек матрицы, которые ему необходимо расчитать.

В процессе работы алгоритма процессам необходимо знать значения граничных областей, рассчитываемых другими процессами. Для этого процесса обмениваются граничными областями с использованием функции *MPI_ Sendrecv*.

Для расчета скалярного произведения необходимо вычислять сумму по всей области. Каждый процесс рассчитывает локальную сумму в своей области, а затем используется опервция *MPI_ Allreduce* с функцией аггрегации *MPI_ SUM* для того, чтобы все процессы получили общую сумму. Аналогичная операция используется и при расчете условия остановки итерациного процесса, однако там используется аггрегации *MPI_ MAX*. В программах в качестве точности выбрано $\epsilon = 10^{-6}$.

В гибридной MPI/OpenMP программе дополнительно с помощью технологии OpenMP производилось распараллеливание циклов. Для этого использовалась следующие директива: *#pragma omp parallel for default(shared) private(li, gi, lj, gj) schedule(dynamic)*, которая изменялась в зависимости от существования приватных переменных в цикле. В случае, когда вычисляется сумма в скалярном произведении используется директива: *#pragma omp parallel for default(shared) private(li, gi, lj, gj) schedule(dynamic) reduction(+:local_sum)*.

4 Описание DVMH программы

Для реализации DVMH версии последовательная программа была подготовлена специальным образом: все массивы объявлены глобальными и статичными, также некоторые переменные тоже стали глобальными. Многие функции были устранены путем подставления их тела в место вызова, в остальных функциях количество передаваемых параметров уменьшено.

Для распределения массивов использовались директивы *#pragma dvm array distribute [block][block]*, *#pragma dvm array align([i][j] with B[i][j])* и *#pragma dvm array align([i][j] with B[i][j]), shadow[1:1][1:1]* в случае, когда массиву требуется обновлять границу от соседей.

Для распределения вычислений использовались директивы вида *#pragma dvm parallel([i][j] on r[i][j])*, *#pragma dvm parallel([i][j] on r[i][j]) shadow_renew(w)* в случае, когда массиву требуется обновлять границу от соседей; и другие различные вариации этой директивы.

Для вычисления скалярного произведения использована директива *#pragma dvm parallel([i][j] on Ar[i][j]) reduction(sum(s))* с редукцией суммы.

Все вычисления обернуты внутри директивы *#pragma dvm region* для выполнения на ускорителе.

5 Результаты расчетов

Ускорение расчитывается по формуле $S = \frac{Timeonseq}{Timeonparallel}$.

Были выбраны размеры сетки $(M, N) = (160, 160)$ и $(M, N) = (160, 320)$ в силу того, что мои программы получают time-out на ПВС IBM Polus при увеличении размера сетки.

Последовательная программа на ПВС IBM Polus отработала на сетках 160×160 и 160×320 соответственно 204.586 (с) и 852.287 (с).

Число точек сетки	Число MPI-процессов	Время (с)	Ускорение S
160×160	4	51.160	3.916
	8	29.768	6.872
	16	18.575	11.014
	32	14.919	13.713
160×320	4	229.417	3.715
	8	124.321	6.855
	16	82.307	10.354
	32	52.391	16.267

Таблица 1. Таблица с результатами расчетов на ПВС IBM Polus (MPI код).

Всюду в таблице 2 количество OMP-нитей в процессе равно четырем.

Число точек сетки	Число MPI-процессов	Время (с)	Ускорение S
160×160	1	54.497	3.754
	2	34.535	5.924
	4	21.126	9.684
	8	17.216	11.878
160×320	1	220.571	3.864
	2	149.629	5.696
	4	65.849	12.943
	8	49.425	17.244

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

В таблице 3 выбран размеры сетки $(M, N) = (500, 500)$. Как видно из таблицы 3, удалось добиться значительного ускорения на GPU с использованием технологии DVMH, в то время, как на CPU программы даже не смогли завершиться из-за ограничения времени.

Число вычислителей	Время (с)
20 MPI	> 3600
40 MPI	> 3600
20 MPI + 4 OMP-threads	> 3600
40 MPI + 4 OMP-threads	> 3600
1 GPU	312.940

Таблица 3. Таблица с результатами расчетов на ПВС IBM Polus (MPI, MPI+OpenMP, DVMH код).

Статистика для 1 GPU. Копирование с CPU на GPU заняло около 5 секунд, полезное исполнение на GPU заняло около 102 секунд, потерянное время на редукцию 33 секунды, потерянное время на операции с памятью составили около 40 секунд. Такая большая потеря времени по сравнению с полезным исполнением связанна с малым размером сетки: GPU слишком быстро обсчитывает и вынужден простаивать из-за памяти.

В таблице 4 выбран размеры сетки $(M, N) = (17500, 17500)$ для получения «хороших» результатов. Так же для такого размера поставлено ограничение на количество итераций в 500.

Число вычислителей	Время (с)
1 GPU	65.012
2 GPU	34.859
4 GPU	121.930

Таблица 4. Таблица с результатами расчетов на ПВС IBM Polus (DVMH код).

Замедление для 4 GPU связано с тем, что не удается запустить задачу эксклюзивно на 2-х хостах, также слишком много издержек по пересылкам памяти, из-за чего доля полезного времени составляет всего $1/3$ от общего числа. Отсюда можно сделать вывод, что данную задачу лучше всего запускать на 1 хосте с 2 GPU для минимизаии издержек.

Статистика для 1 GPU. Копирование с CPU на GPU заняло около 0.7 секунд, полезное исполнение на GPU заняло около 63 секунд, потерянное время на редукцию 6 секунд, потерянное время на операции с памятью составили

около 0.2 секунд. В данном случае видно, что доля полезного времени значительна, так как GPU заполнен по памяти на 3/4.

Выбраны графики решений на сетке 160×160, так как они уже визуально достаточно точны и неотличимы.

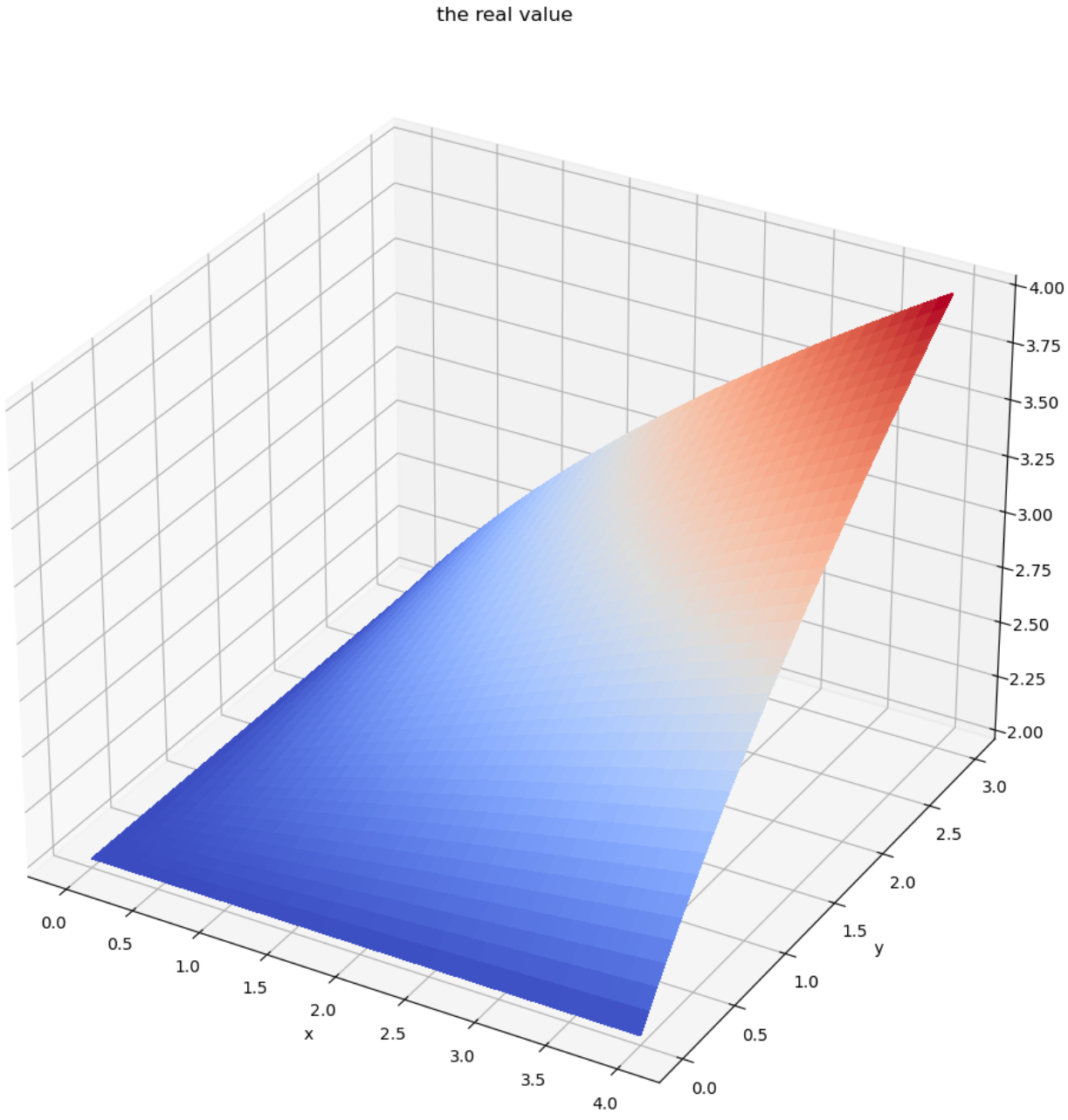

Рисунок 1. График точного решения на сетке 160×160.

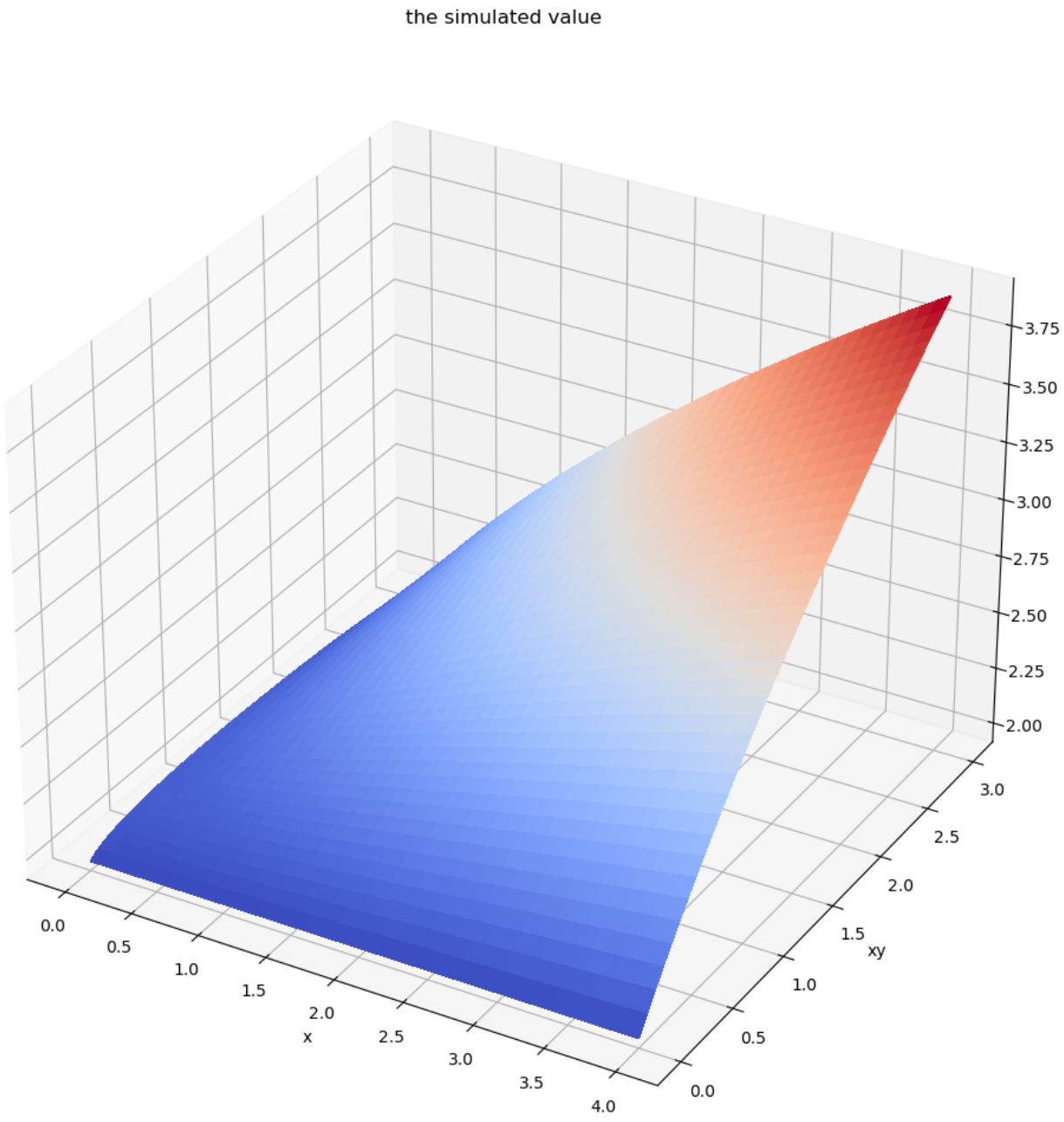

Рисунок 2. График приближенного решения на сетке 160×160.