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

Чижов Пётр Сергеевич
608 группа
10 вариант

Численное интегрирование многомерных функций методом Монте-Карло

Москва
2022

1 Введение

В качестве модельной задачи предлагается задача вычисления многомерного интеграла методом Монте-Карло.

Программная реализация должна быть выполнена на языке C/C++ с использованием библиотеки параллельного программирования Message Processing Interface(MPI).

Требуется исследовать масштабируемость параллельной MPI-программы на параллельной вычислительной системе ВМК МГУ: IBM Polus.

2 Математическая постановка задачи

Функция $f(x,\, y,\, z)$ — непрерывна в ограниченой замкнутой области $G \subset \mathbb{R}^3$. Требуется вычислить определённый интеграл:

$$I = \iiint_G f(x,\, y,\, z)\, dx\, dy\, dz \ ,$$

$$\text{где функция } f(x,\, y,\, z) = \tfrac{1}{(1+x+y+z)^3} \ ,$$

$$\text{область G ограничена поверхностями } x + y + z = 1,\ x = 0,\ y = 0,\ z = 0 \ .$$

3 Численный метод решения задачи

Пусть область G ограниченна параллелепипедом

$$\Pi : \begin{cases} a_1 \leqslant x \leqslant b_1, \\ a_2 \leqslant y \leqslant b_2, \\ a_3 \leqslant z \leqslant b_3 \end{cases}$$

Рассмотрим функцию:

$$F(x,y,z) = \begin{cases} f(x,y,z)\,, & (x,y,z) \in G \\ 0\,, & (x,y,z) \notin G \end{cases}$$

Преобразуем искомый интеграл:

$$I = \iiint_G f(x,\, y,\, z)\, dx\, dy\, dz \ = \ \iiint_\Pi F(x,\, y,\, z)\, dx\, dy\, dz$$

Пусть $p_1(x_1,\, y_1,\, z_1), p_2(x_2,\, y_2,\, z_2), \ldots$ — случайные точки, равномерно распределённые в П. Возьмём n таких случайных точек. В качестве приближённого значения интеграла предлагается использовать выражение:

$$I \;\approx\; |\Pi| \cdot \tfrac{1}{n} \sum_{i=1}^{n} F(p_i),$$

где $|\Pi|$ - объём параллелепипеда П. $|\Pi| = (b_1 - a_1)(b_2 - a_2)(b_3 - a_3)$

4 Нахождение точного значения интеграла аналитически

$I = \iiint_G \frac{1}{(1+x+y+z)^3}\, dx\, dy\, dz \;=\; \int_0^1 dx \int_0^{1-x} dy \int_0^{1-x-y} (\frac{dz}{(1+x+y+z)^3}) \;= \int_0^1 dx \int_0^{1-x} (-\frac{1}{8} + \frac{1}{2(1+x+y)^2}) dy \;=$

$\int_0^1 (-\frac{3}{8} + \frac{x}{8} + \frac{1}{2(1+x)}) dx \;= [-\frac{3}{8}x + \frac{1}{16}x^2 + \frac{1}{2} ln(1+x)]_0^1 \;=\; \frac{1}{2} ln(2) - \frac{5}{16}$

5 Описание программной реализации

Параллельная MPI-программа принимает на вход требуемую точность и генерирует случайные точки до тех пор, пока требуемая точность не будет достигнута. Программа вычисляет точность как модуль разности между приближённым значением, полученным методом Монте-Карло, и точным значением, вычислинным аналитически.

Программа считывает в качестве аргумента командной строки требуемую точность ϵ и выводит четыре числа:

- Посчитанное приближённое значение интеграла
- Ошибка посчитанного значения: модуль разности между приближённым и точным значениями интеграла
- Количество сгенерированных случайных точек
- Время работы программы в секундах

Время работы программы измеряется следующим образом. Каждый MPI-процесс измеряет своё время выполнения, затем среди полученных значений берётся максимум.

В моём варианте реализована парадигма «мастер–рабочие»: один из процессов («мастер») генерирует по очереди для каждого из остальных процессов («рабочих») n / (количество процессов$-$ 1) случайных точек и передаёт их. Все процессы–рабочие получают свой набор точек и вычисляют свою часть суммы. Затем «рабочие» отправляют полученные части суммы «мастеру», который, в свою очередь, вычисляет общую сумму. После чего вычисляется ошибка (разность между посчитанным значением и точным значением, вычисленным аналитически). В случае если ошибка выше требуемой точности, которую подали на вход программе, то генерируются дополнительные n / (количество процессов $-$ 1) точек для каждого процесса-рабочего и расчёт продолжается.

6 Исследование мастшабируемости программы на системе Polus

Таблица 1: Таблица с результатами расчётов для системы Polus (seed = 10)

Точность ϵ	**Число MPI-процессов**	**Время работы программы (с)**	**Ускорение**	**Ошибка**
$3.0 \cdot 10^{-5}$	2	0.003448	1.0	1.70e-6
	4	0.001516	2.274	1.09e-5
	16	0.000260	13.262	2.088e-6
$5.0 \cdot 10^{-6}$	2	0.003449	1.0	1.7e-06
	4	0.001726	1.99	2.09e-06
	16	0.000209	16.49	2.09e-06
$1.5 \cdot 10^{-6}$	2	0.084498	1.0	1.36e-06
	4	0.029472	2.87	4.53e-07
	16	0.005593	15.11	1.42e-06

Таблица 2: Таблица с результатами расчётов для разных seed на точности $1.5 \cdot 10^{-6}$

Seed	**Число MPI-процессов**	**Время работы программы (с)**	**Ускорение**	**Ошибка**
10	2	0.084498	1.0	1.36e-06
	4	0.029472	2.87	4.53e-07
	16	0.005593	15.11	1.42e-06
12	2	0.028620	1.	1.44e-06
	4	0.015141	1.89	1.27e-06
	16	0.003955	7.24	1.24e-06
256	2	0.079841	1.0	1.00e-06
	4	0.033828	2.36	8.54e-07
	16	0.015158	5.267	6.98e-07
2048	2	0.047868	1.0	1.20e-06
	4	0.024944	1.92	1.48e-06
	16	0.002333	20.52	1.10e-06
10000	2	0.012868	1.0	7.24e-07
	4	0.006628	1.94	1.73e-07
	16	0.002169	5.93	1.28e-06

График ускорения для различных точностей

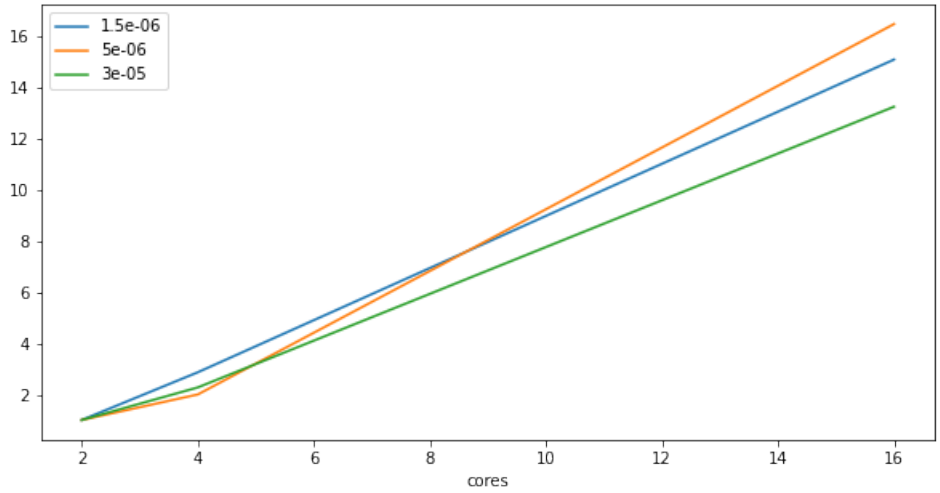

График ускорения для различных seed

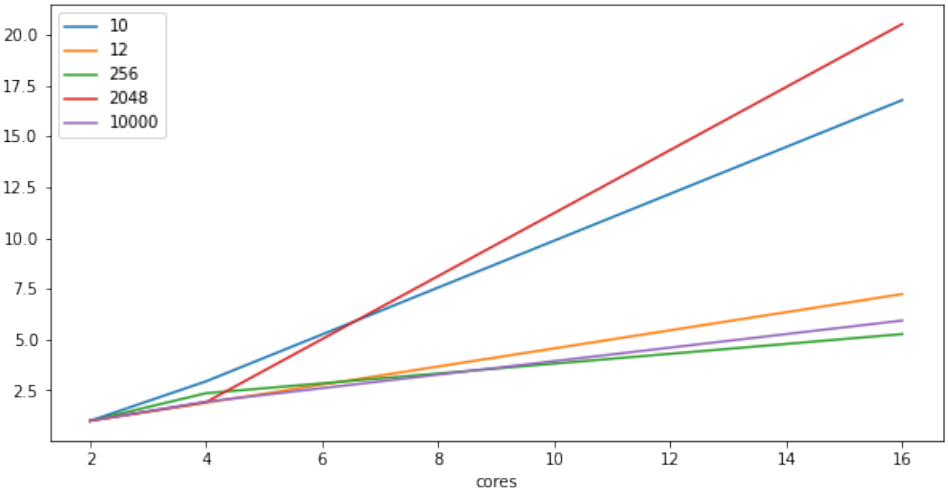

График ускорения усредненный по seed