



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

Орлова Анна Олеговна, 614 группа

Вариант 1

Отчёт по заданию в рамках курса

«Суперкомпьютерное моделирование и технологии»

Численное интегрирование многомерных функций методом Монте-Карло

Москва
2022

1 Математическая постановка задачи и численный метод решения

Функция $f(x, y, z)$ — непрерывна в ограниченной замкнутой области $G \subset \mathbb{R}^3$.

Требуется вычислить определённый интеграл:

$$I = \iiint_G xy^2z^3 dx dy dz$$

,

где область $G = \{(x, y, z) : z = xy, y = x, x = 1, z = 0\}$

Пусть область G ограничена параллелепипедом V :

$$\begin{cases} a_1 = 0 \leq x \leq 1 = b_1, \\ a_2 = 0 \leq y \leq 1 = b_2, \\ a_3 = 0 \leq z \leq 1 = b_3. \end{cases}$$

Объем параллелепипеда V равен 1.

2 Нахождение точного значения интеграла аналитически

$$\begin{aligned} I &= \int_0^1 dx \int_0^x dy \int_0^{xy} xy^2z^3 dz = \int_0^1 x dx \int_0^x y^2 dy \int_0^{xy} z^3 dz = \frac{1}{4} \int_0^1 x dx \int_0^x y^2 x^4 y^4 dy = \\ &= \frac{1}{4} \int_0^1 x dx \int_0^x y^6 x^4 dy = \frac{1}{4} \int_0^1 x^5 dx \int_0^x y^6 dy = \frac{1}{4} \frac{1}{7} \int_0^1 x^5 x^7 dx = \frac{1}{28} \int_0^1 x^{12} dx = \\ &= \frac{1}{28} \frac{1}{13} = \frac{1}{364} \approx 0.0027472527472527475 \end{aligned}$$

3 Описание программной реализации

```
#include <stdio.h>
#include <mpi.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

double generate(x, y)
{
    return ((double)rand()/((double)(RAND_MAX)) * (y - x) + x;
}

int main(int argc, char *argv[])
{
    int size, rank;
    long N = 0, i = 0;
    double integral = 1.0/364.0, epsilon = strtod(argv[1], NULL);
    double a1 = 0.0, a2 = 1.0, b1 = 0.0, b2 = 1.0, c1 = 0.0, c2 = 1.0;
    int par = 1, root = 0;
    double x, y, z;
    double sum = 0.0, f_sum, func, int_appr = 0.0, error = 0.0;
    double time_start, time_end, ttime, timee;
    int seed; // seeds = {1, 13, 20, 614, 1300, 2022}

    sscanf(argv[1], "%lf", &epsilon);
    sscanf(argv[2], "%d", &seed);

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    srand(seed + rank); //+ time(0)

    time_start = MPI_Wtime();

    while (1)
    {
        N++;
        x = generate(a1, a2);
        y = generate(b1, b2);
        z = generate(c1, c2);

        //G: z = xy, y = x, x = 1, z = 0
        //V: a1 <= x <= a2 && b1 <= y <= b2 && c1 <= z <= c2
        if (a1 <= x <= a2 && b1 <= y <= b2 && c1 <= z <= c2 && x * y >= z && y <= x)
        {
            func = x*pow(y, 2)*pow(z, 3);
            //func = x*y*y*z*z*z;
            i++;
        }
    }
}
```

```

    }
    else
    { func = 0.0; }

    MPI_Reduce(&func, &f_sum, 1, MPI_DOUBLE, MPI_SUM, root, MPI_COMM_WORLD);

    if (rank == root)
    {
        sum = sum + f_sum/((double)size);
        int_appr = par * sum/ (double)N;
        error = fabs(integral - int_appr);
    }

    MPI_Barrier(MPI_COMM_WORLD);
    MPI_Bcast(&error, 1, MPI_DOUBLE, root, MPI_COMM_WORLD);

    if (error < epsilon)
        { break;}
}

time_end = MPI_Wtime();
timee = time_end - time_start;
MPI_Reduce(&timee, &ttime, 1, MPI_DOUBLE, MPI_MAX, root, MPI_COMM_WORLD);

MPI_Finalize();

if (rank == root)
{
    printf("Integral approximate = %f\n", int_appr);
    printf("Integral numerical = %f\n", integral);
    printf("Error: %e %.10f\n", error, error);
    printf("Epsilon: %.10f\n", epsilon);
    printf("Seed: %d\n", seed);
    printf("Process number (all points) N*size: %ld\n", N*size);
    printf("Process number (points for sum) i*size: %ld\n", i*size);
    printf("Max time: %.8f\n", ttime);
    printf("Rank: %d\n", rank);
    printf("Size: %d\n", size);
}

return 0;
}

```

4 Исследование масштабируемости программы

Точность ϵ	Число MPI-процессов	Время работы программы (с)	Ускорение	Ошибка
$3.0 \cdot 10^{-5}$	1	0.03529720	1	0.0000285293
	4	0.00055616	63.47	0.0000204714
	16	0.00205942	17.14	0.0000220534
	32	0.00303121	11.644	0.0000194596
$5.0 \cdot 10^{-6}$	1	0.03717403	1	0.0000046822
	4	0.02993409	1.242	0.0000031477
	16	0.00388553	9.567	0.0000047985
	32	0.03834892	0.969	0.0000031924
$1.5 \cdot 10^{-6}$	1	0.03712880	1	0.0000003629
	4	0.01547978	2.399	0.0000013696
	16	0.00384339	9.660	0.0000010394
	32	0.03731755	0.999	0.0000010004

Рис. 1: Значение seed = 13

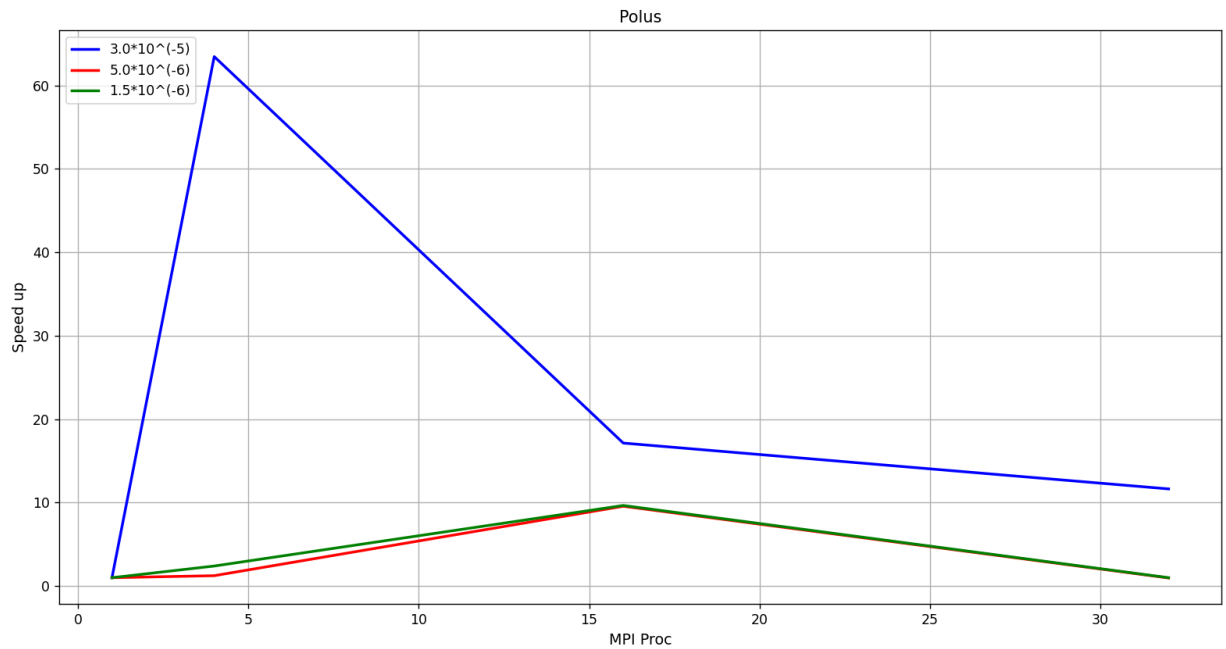


Рис. 2: Зависимость ускорения от числа MPI - процессов (значение seed = 13)