Отчет по выполнению практического задания «Методы Монте-Карло» по курсу Естественные модели параллельных вычислений студента 523 группы Ухина Сергея Алексеевича.

ser191097@gmail.com

- Графики зависимостей T(N) - времени, S(N) - ускорения, E(N) - эффективности от количества частиц N при фиксированном значении количества процессов P.

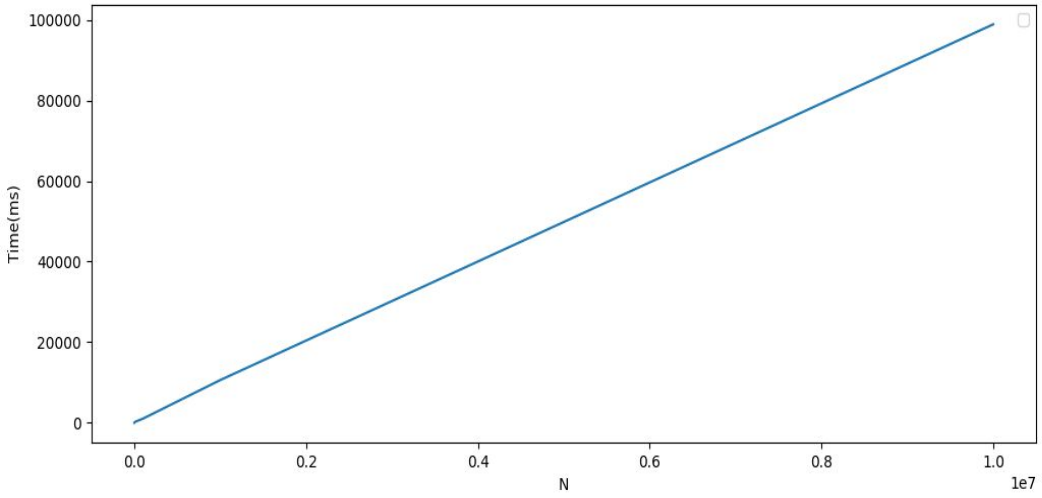

Рис. 1: Зависимость времени работы программы от количества частиц. Используется 10 процессов.

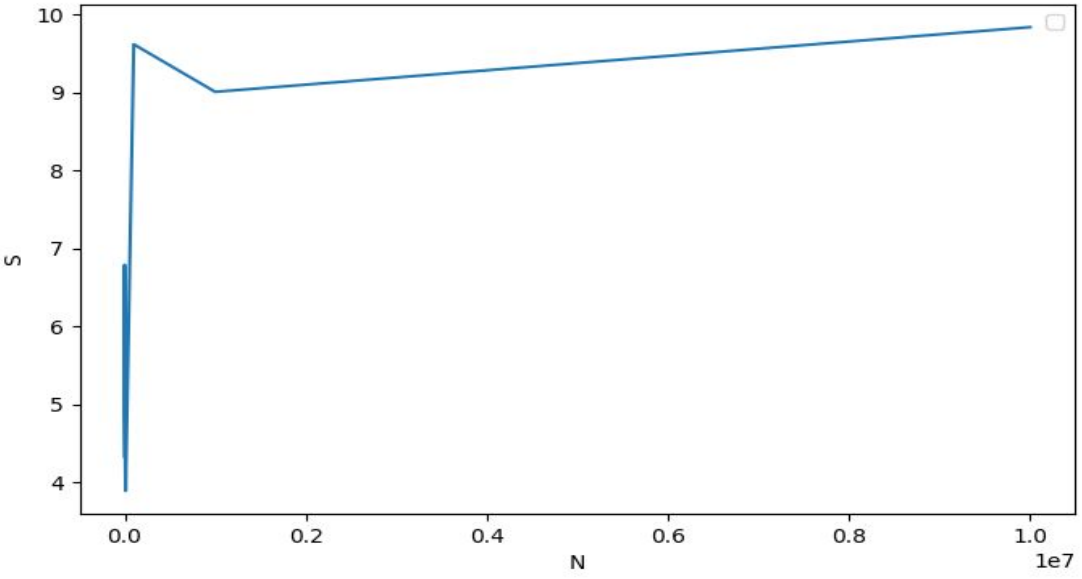

Рис. 2: Зависимость ускорения программы от количества частиц относительно не параллельной версии. Используется 10 процессов.

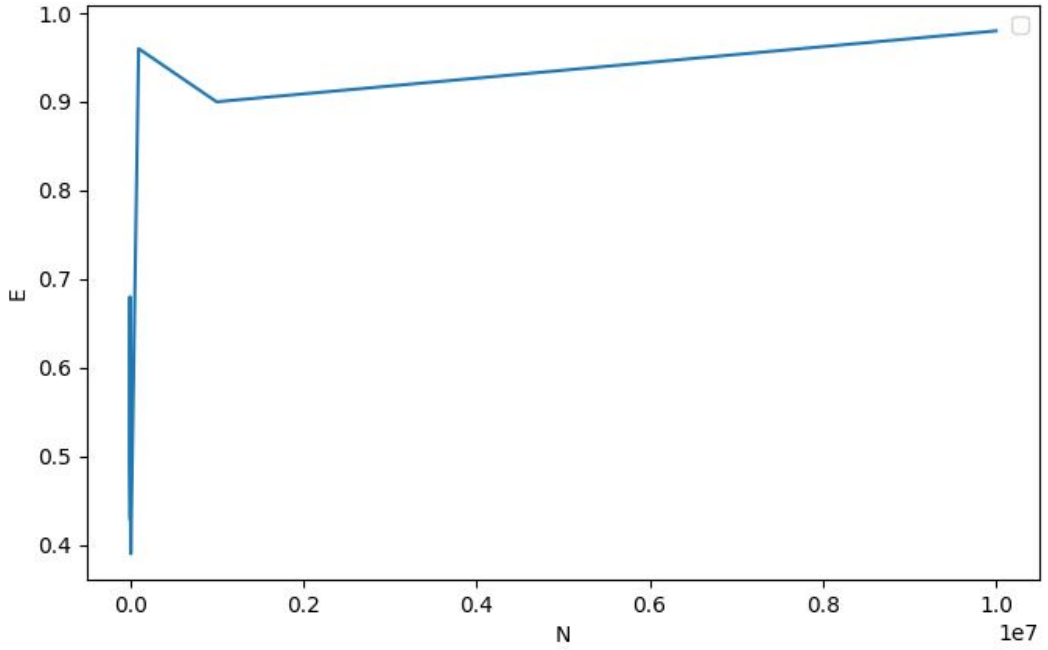

Рис. 3: Зависимость эффективности программы от количества частиц относительно не параллельной версии. Используется 10 процессов.

Нестабильность 2 последних графиков в начале экспериментов связана с тем, что при малом количестве N (10, 100, 1000) время, которое уходит на время основного цикла программы сравнимо с накладными расходами на инициализацию библиотеки MPI.

- Графики зависимостей T(P), S(P), E(P) от количества используемых процессов P при фиксированном количестве частиц N.

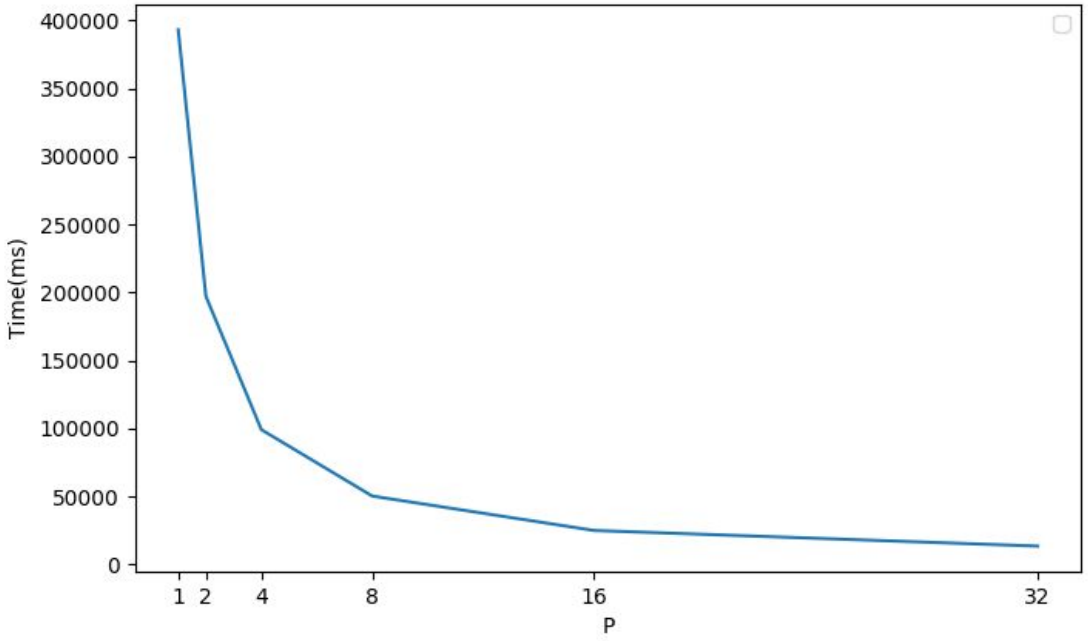

Рис. 4: Зависимость времени работы программы от количества процессов. N = 10^6.

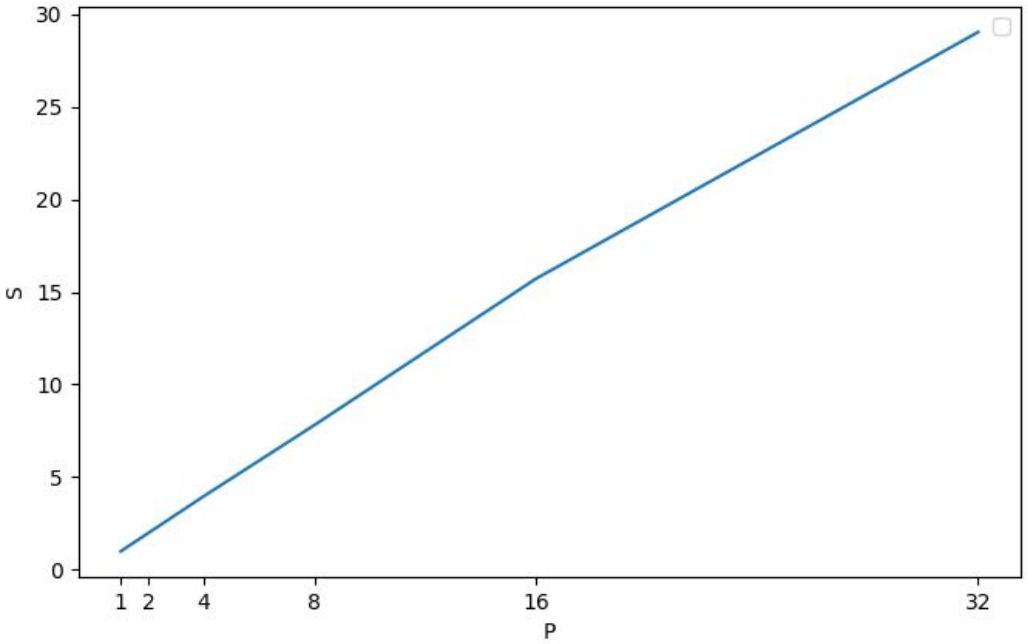

Рис. 5: Зависимость ускорения программы от количества процессов. N = 10^6.

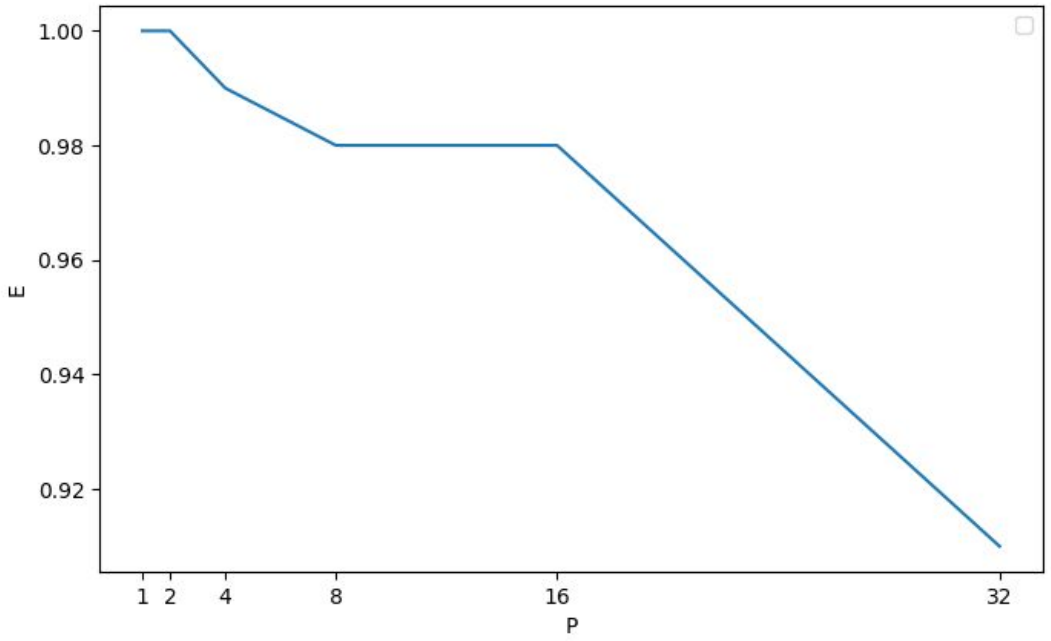

Рис. 6: Зависимость эффективности программы от количества процессов. N = 10^6.

- Графики зависимостей T(P), S(P), E(P) от количества используемых процессов P при количестве частиц N = 10^3 · P

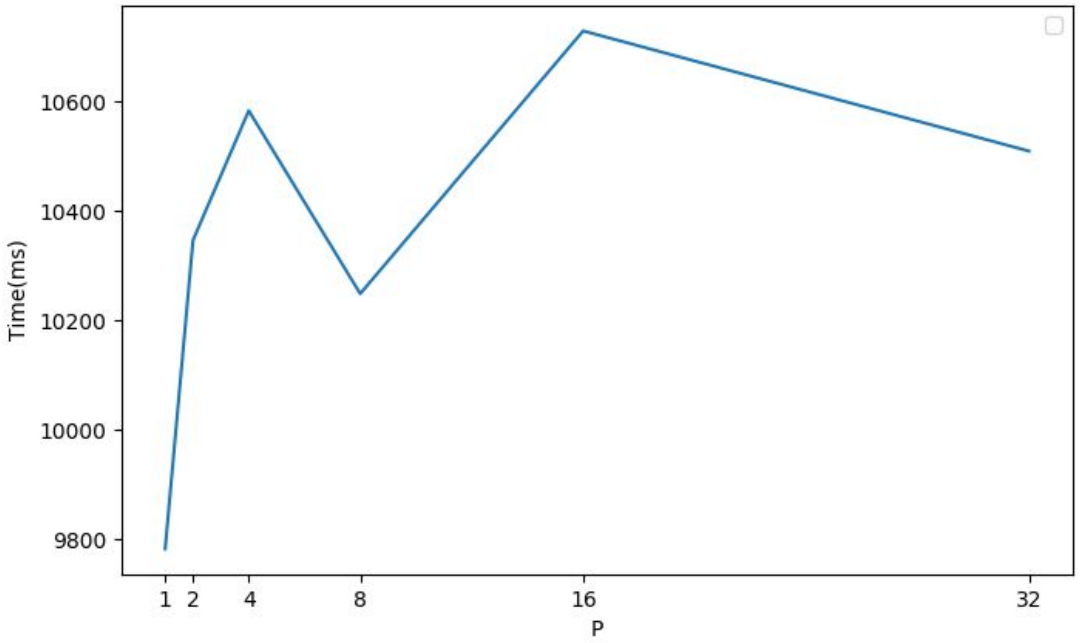

Рис. 7: Зависимость времени работы программы от количества используемых процессов. Количество частиц равно 10^3 · *P*.

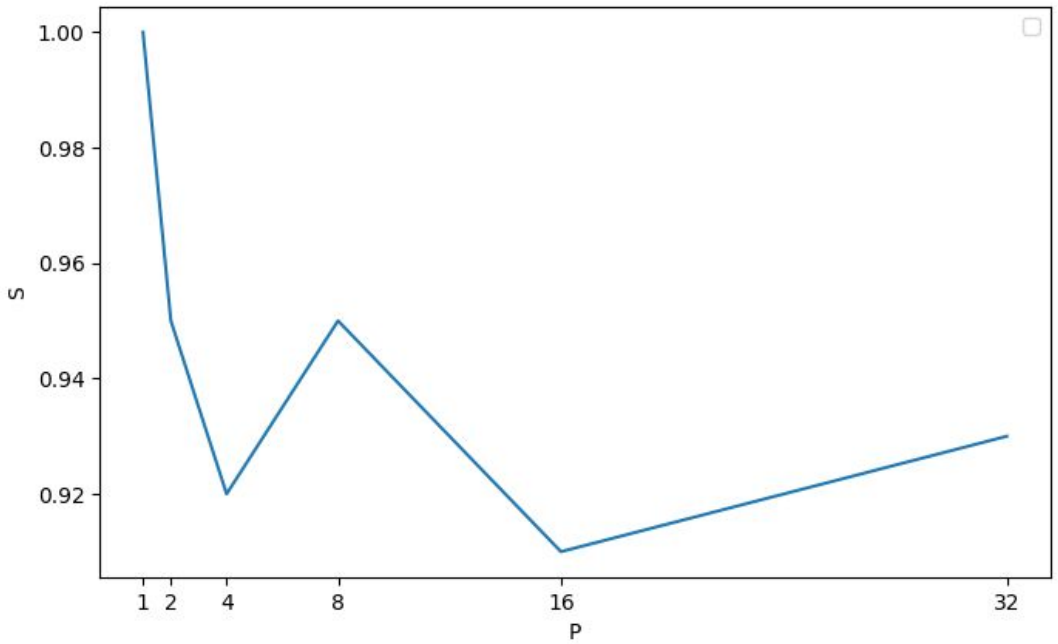

Рис. 8: Зависимость ускорения программы от количества используемых процессов. Количество частиц равно 10^3 · *P*.

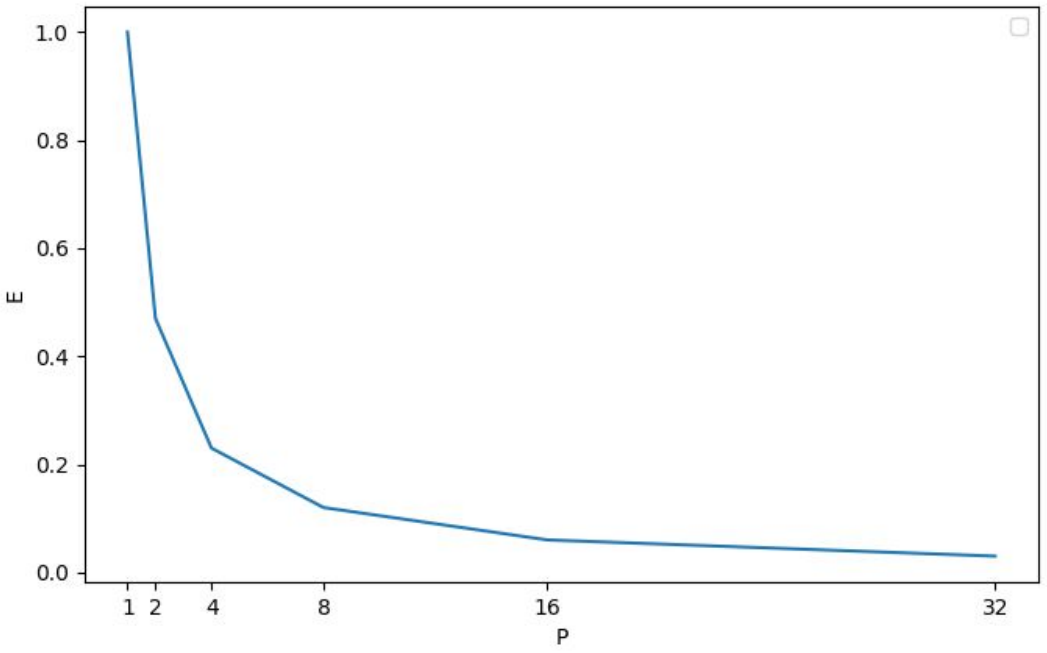

Рис. 9: Зависимость эффективности программы от количества используемых процессов. Количество частиц равно 10^3 · P.

Полный код параллельной программы.

```
#include <iostream>
#include <mpi.h>
#include <utility>
#include <cstdlib>
#include <fstream>
#include <string>
#include <time.h>

using namespace std;



pair<int, long long int> dowalk(int a, int b, int x, double p) {
   long long dt = 0;
   while (a < x && x < b) {
      double r = ((double) rand()) / RAND_MAX;
      if (r < p) {
         ++x;
      } else {
         --x;
      }
      ++dt;
   }
   return pair<int , long long int>(x, dt);
}


int main(int argc, char **argv) {
   if(argc != 6) {
      cout << "input: a b x p n" << endl;
      return 1;
   }

   int a = atoi(argv[1]);
   int b = atoi(argv[2]);
   int x = atoi(argv[3]);
   double p = atof(argv[4]);
   int n = atoi(argv[5]);
   MPI_Init(&argc, &argv);
   int rank, size;
   MPI_Comm_size(MPI_COMM_WORLD, &size);
```

```
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int myn = n / size;
    srand(time(NULL) + rank);
    int w = 0;
    long long t = 0;
    double start_time, end_time;
    MPI_Barrier(MPI_COMM_WORLD);
    start_time = MPI_Wtime();
    for(int i = 0; i < myn; ++i) {
        pair<int, long long int> temp = dowalk(a, b, x, p);
        if (temp.first == b) {
            ++w;
        }
        t += temp.second;
    }
    MPI_Barrier(MPI_COMM_WORLD);
    end_time = MPI_Wtime();
    double time = end_time - start_time;
    double buf[2];
    double total[2];
    total[0] = 0;
    total[1] = 0;
    buf[0] = ((double) w) / myn;
    buf[1] = ((double) t) / myn;
    MPI_Reduce(buf, total, 2, MPI_DOUBLE, MPI_SUM, 0,
MPI_COMM_WORLD);
    if (rank == 0) {
        ofstream fout("output.txt", std::ofstream::out | std::ofstream::app);
        fout << total[0]/size << ' ' << total[1]/size << endl;
        fout <<
"--------------------------------------------------------------------------------" <<
endl;
        fout.close();
        fout.open("stat.txt", std::ofstream::out | std::ofstream::app);
        fout << "time = " << time << " P = " << size << endl;
        fout << a << ' ' << b << ' ' << x << ' ' << p << ' ' << n <<endl;
        fout <<
"--------------------------------------------------------------------------------" <<
endl;
        fout.close();
    }
    return 0;
```

}