Задание 5
OpenMP, метод Monte-Carlo
Отчёт

Фролова О.В

1 Постановка задачи

Требуется реализовать параллельный алгоритм для модели случайных блужданий с использованием OpenMP
Программа должна работать при любых значениях a, b, p, x, N, P (число потоков) и выдавать в качестве результата:

- вероятность достижения b
- среднее время жизни одной частицы
- время работы основного цикла

Составить график зависимости T(N), S(N), E(N) при фиксированном значении P != 1
Составить график зависимости T(P), S(P), E(P) при фиксированном большом значении N. Для значений P достаточно брать 1, 2, 4, 8, 16.

T - время работы программы (основного цикла), S - ускорение, E - эффективность распараллеливания

2 Результаты выполнения

Был выбран отрезок [0,100], точка x = 50, вероятность перехода вправо(p) - 0.5 и количество частиц (N) - 1000000

Число нитей	Вероятность достижения b	Среднее время жизни одной частицы	Время работы основного цикла
1	0.499965	0.000050 sec	50.117150 sec
2	0.500042	0.000050 sec	25.211611 sec
3	0.500048	0.000050 sec	16.863608 sec
4	0.499799	0.000051 sec	12.733048 sec
5	0.499825	0.000051 sec	10.260505 sec
6	0.500196	0.000051 sec	8.626088 sec
7	0.499762	0.000053 sec	7.639234 sec
8	0.500114	0.000051 sec	6.439253 sec
9	0.500107	0.000055 sec	6.250096 sec
10	0.499889	0.000056 sec	5.719520 sec
11	0.500120	0.000058 sec	5.323886 sec
12	0.500112	0.000059 sec	5.006684 sec
13	0.500191	0.000060 sec	4.722091 sec
14	0.500392	0.000061 sec	4.448141 sec
15	0.500035	0.000062 sec	4.237502 sec
16	0.500262	0.000063 sec	4.024994 sec

3 Подсчет ускорения и эффективности распараллеливания программы

Для каждого увеличения числа нитей считалось ускорение по формуле

$$S_p = \frac{T_1}{T_p},$$

где T_1 - время работы программы на одной нити, а T_p - время работы программы на p нитях.

А эффективность распараллеливания $E = S_p/P$

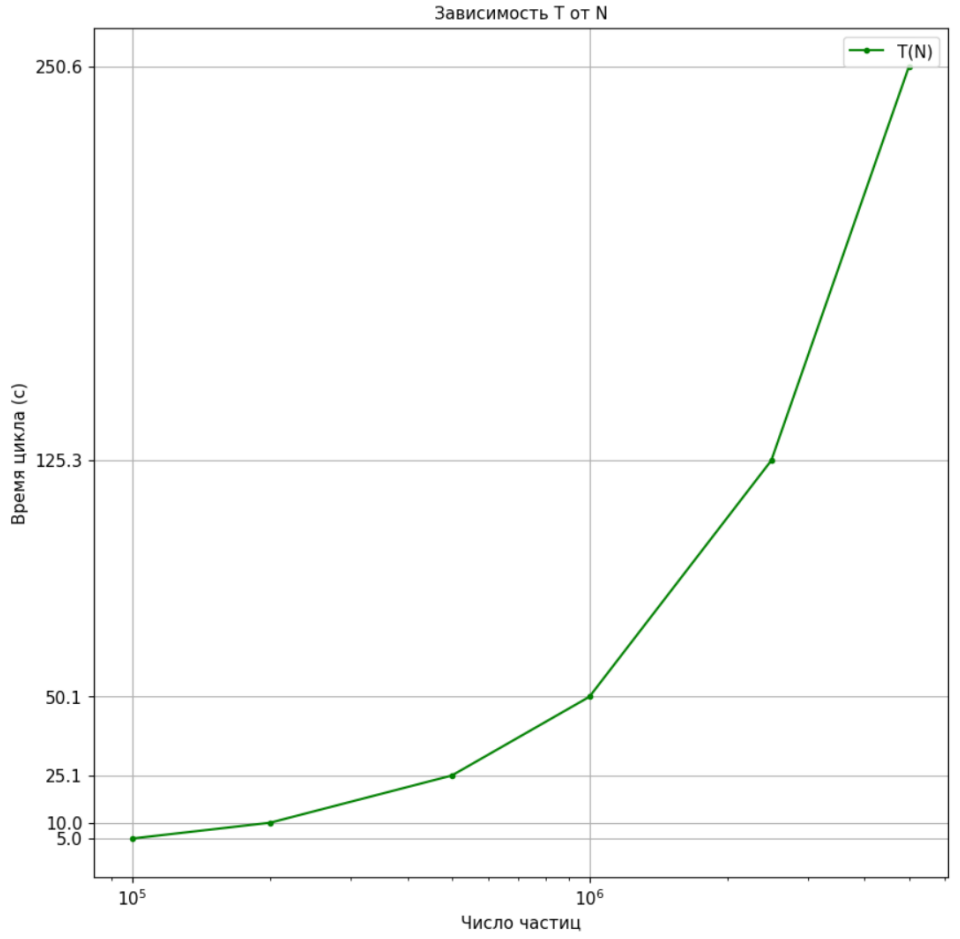
Число нитей	Ускорение программы	Эффективность распараллеливания программы
1	1.0	1.0
2	1.98786	0.99393
3	2.97191	0.99064
4	3.93599	0.98399
5	4.88447	0.97689
6	5.80995	0.96833
7	6.56049	0.93721
8	7.78307	0.97288
9	8.01862	0.89096
10	8.76247	0.87625
11	9.41364	0.85578
12	10.01005	0.83417
13	10.61334	0.81641
14	11.26698	0.80478
15	11.82705	0.78847
16	12.45148	0.77821

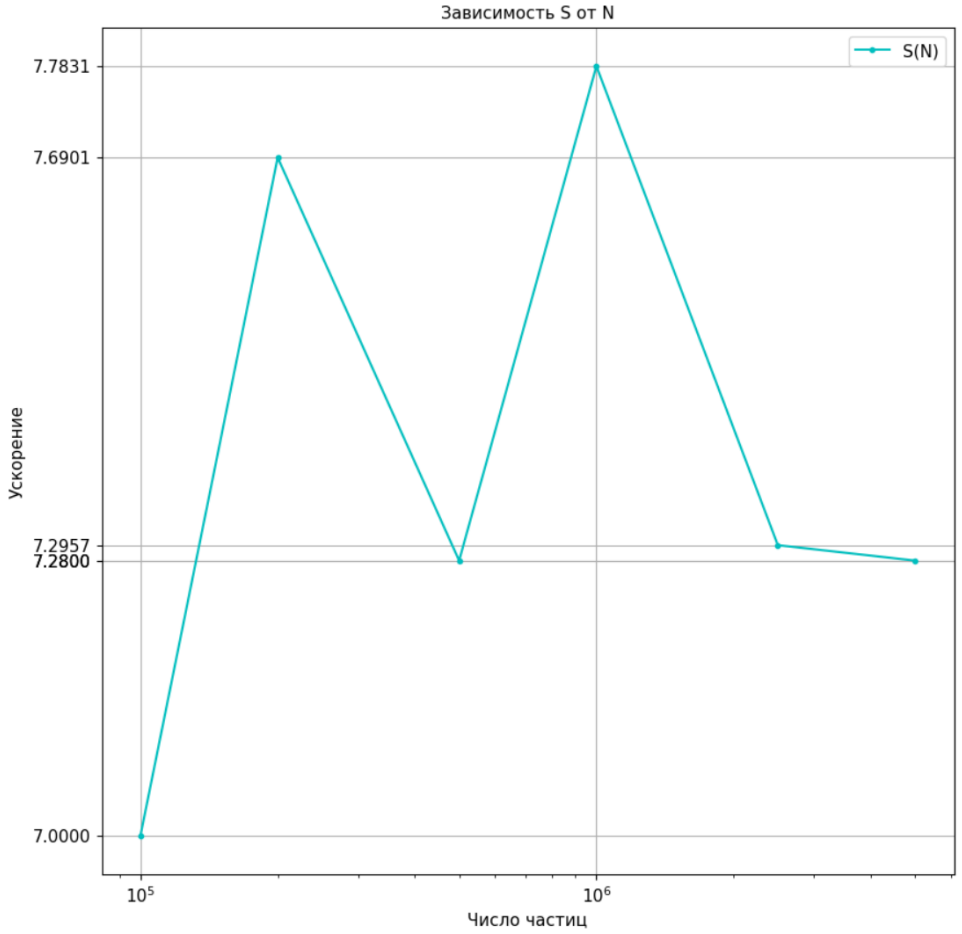
Вывод: Из результатов видно, что после 8 нитей, эффективность распараллеливания значительно снижается

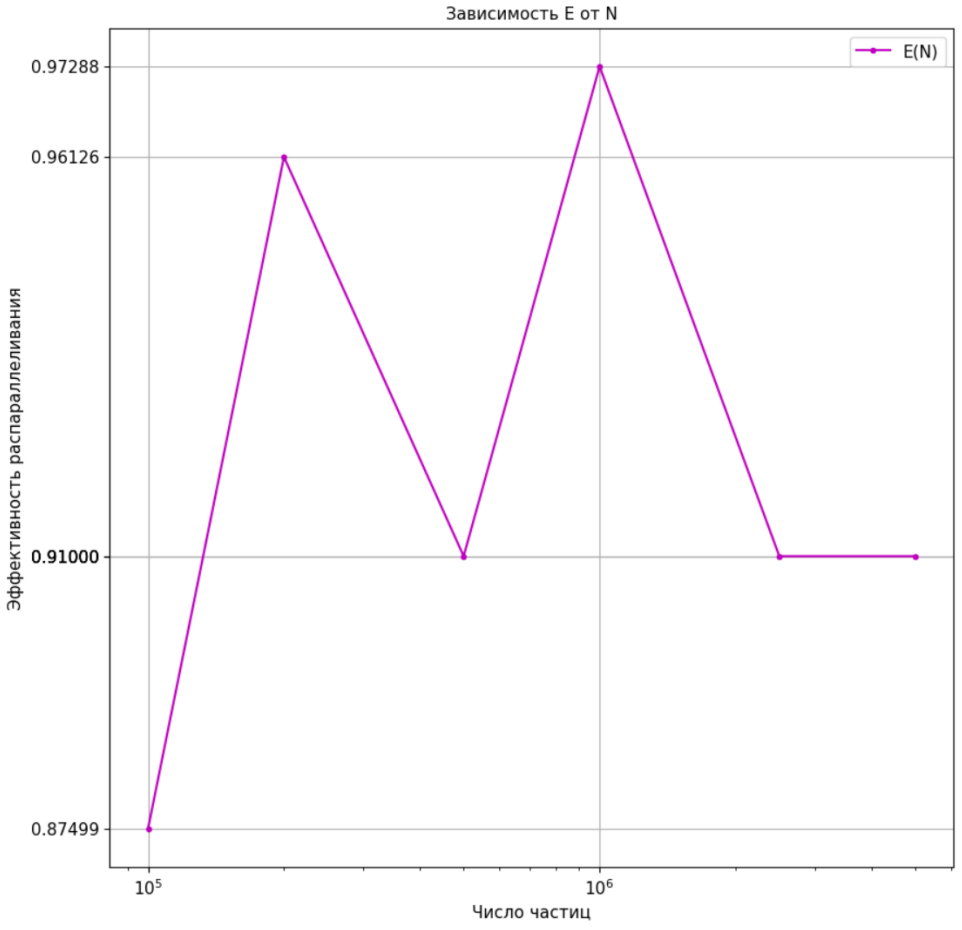
4 Графики зависимости

Возьмем $P = 8$. Исследуем изменение времени программы, ускорения и эффективности распараллеливания

N (10^3)	Время выполнения(P=1)	Время выполнения(P=8)	Ускорение	Эффективность распараллеливания
100	5.013116 sec	0.716161 sec	6.99998	0.87499
200	10.027537 sec	1.303960 sec	7.69006	0.96126
500	25.061647 sec	3.441847 sec	7.28145	0.91018
1000	50.117150 sec	6.439253 sec	7.78307	0.97288
2500	125.310372 sec	17.175974 sec	7.29567	0.91196
5000	250.624391 sec	34.429452 sec	7.27936	0.90992

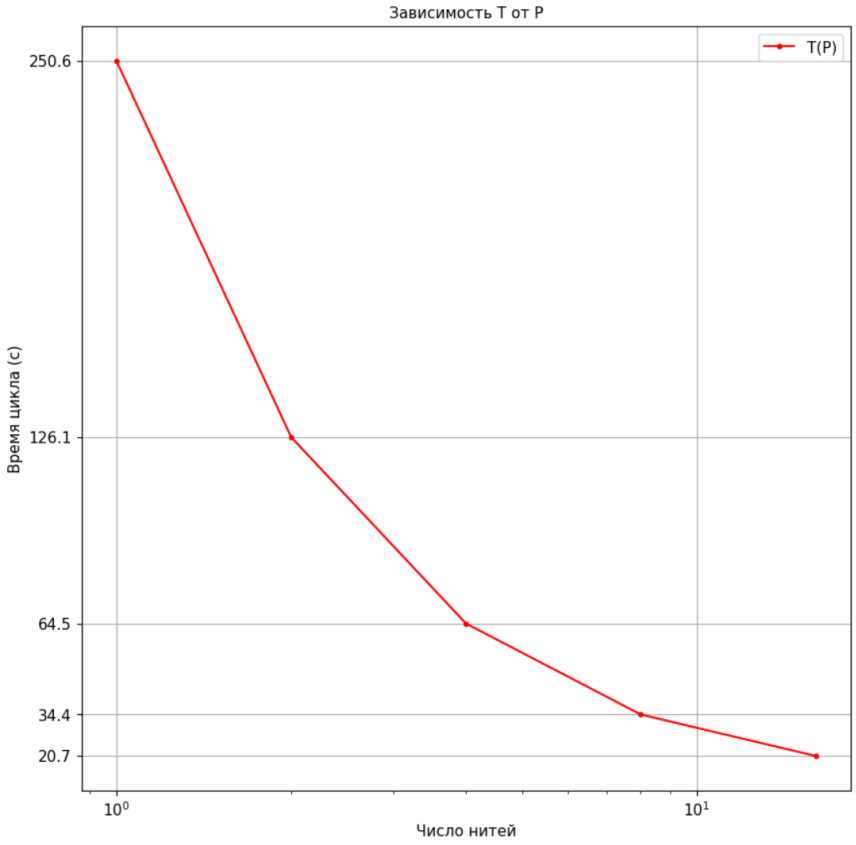


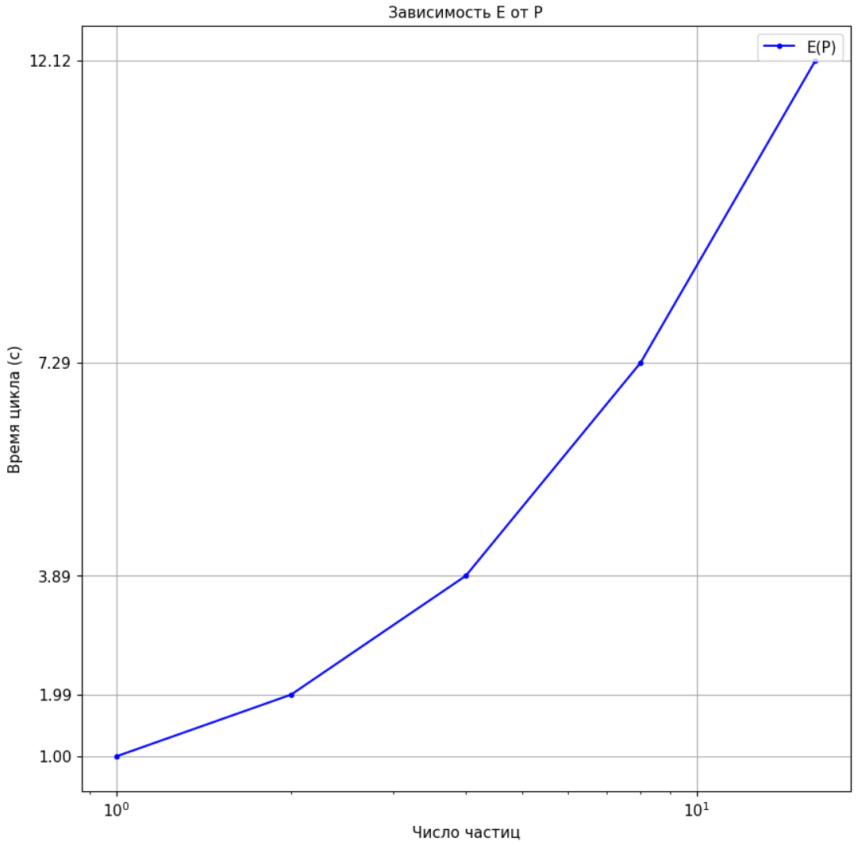

Зависимость S от N
S(N)
7.7831
7.6901
7.2957
7.2800
7.0000
Ускорение
10^5
10^6
Число частиц

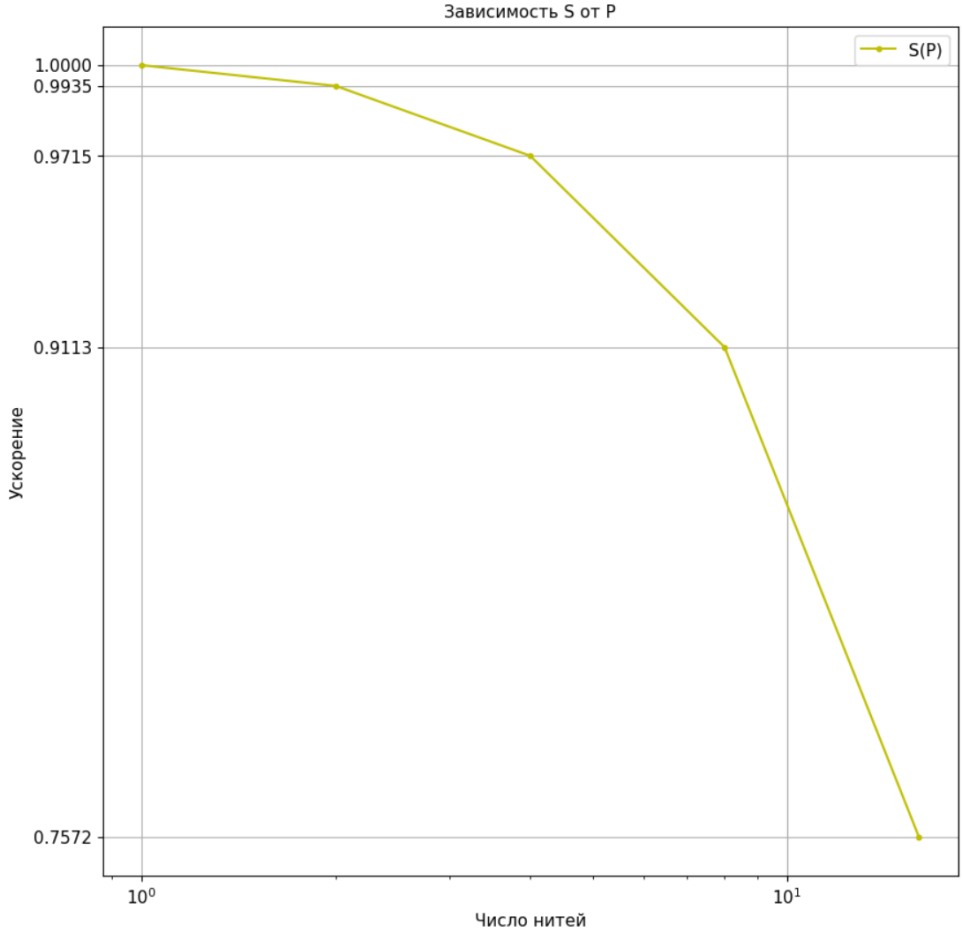

Зависимость E от N
E(N)
0.97288
0.96126
0.91000
0.87499
Эффективность распараллеливания
10^5
10^6
Число частиц

Теперь зафиксируем количество частиц: $N = 5 * 10^6$. И протестируем программу на $P = 1, 2, 4, 8, 16$ нитях

Число нитей	Время выполнения	Ускорение	Эффективность распараллеливания
1	250.624391 sec	1.0	1.0
2	126.135671 sec	1.98694	0.99347
4	64.493835 sec	3.88602	0.97150
8	34.377288 sec	7.29040	0.91130
16	20.686054 sec	12.115621	0.75722






Зависимость S от P
S(P)
1.0000
0.9935
0.9715
0.9113
0.7572
Ускорение
10^0
10^1
Число нитей