

pi_accuracy_graficos.c

```
1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <time.h>
5 #include <math.h> // Necessário para M_PI
6
7 double calculate_pi_sequential(long long num_iterations) {
8     double sum = 0.0;
9     int sign = 1;
10
11     for (long long i = 0; i < num_iterations; i++) {
12         double term = (double)sign / (2.0 * i + 1.0);
13         sum += term;
14         sign *= -1;
15     }
16
17     return 4.0 * sum;
18 }
19
20
21 double calculate_pi_ilp(long long num_iterations) {
22     double sum1 = 0.0, sum2 = 0.0, sum3 = 0.0, sum4 = 0.0;
23
24     for (long long i = 0; i < num_iterations; i += 4) {
25         sum1 += 1.0 / (2.0 * i + 1.0);
26         sum2 -= 1.0 / (2.0 * i + 3.0);
27         sum3 += 1.0 / (2.0 * i + 5.0);
28         sum4 -= 1.0 / (2.0 * i + 7.0);
29     }
30
31     double total_sum = sum1 + sum2 + sum3 + sum4;
32
33     return 4.0 * total_sum;
34 }
35
36
37 int main() {
38     long long iterations[] = {
39         10, 100, 1000, 10000, 100000, 1000000,
40         10000000, 100000000, 1000000000
41     };
42     int num_tests = sizeof(iterations) / sizeof(iterations[0]);
43
44     // Abre o ficheiro CSV para escrita
45     FILE *csv_file = fopen("pi_results.csv", "w");
46     if (csv_file == NULL) {
47         perror("Erro ao abrir o ficheiro CSV");
48         return 1;
49     }
50
51     // Escreve o cabeçalho do CSV
```

```
52     fprintf(csv_file, "Metodo,Iteracoes,PiCalculado,ErroAbsoluto,Tempo_s\n");
53     printf("A gerar resultados em pi_results.csv...\n");
54
55     // --- Testes para a Versão Sequencial ---
56     for (int i = 0; i < num_tests; i++) {
57         long long current_iterations = iterations[i];
58         struct timespec start_time, end_time;
59
60         clock_gettime(CLOCK_MONOTONIC, &start_time);
61         double pi_approximation = calculate_pi_sequential(current_iterations);
62         clock_gettime(CLOCK_MONOTONIC, &end_time);
63
64         double time_spent = (end_time.tv_sec - start_time.tv_sec) +
65                             (end_time.tv_nsec - start_time.tv_nsec) / 1e9;
66         double error = fabs(M_PI - pi_approximation);
67
68         // Escreve os dados no ficheiro CSV
69         fprintf(csv_file, "Sequencial,%lld,%.18f,%.18f,%.9f\n",
70                 current_iterations, pi_approximation, error, time_spent);
71     }
72
73     // --- Testes para a Versão Otimizada com ILP ---
74     for (int i = 0; i < num_tests; i++) {
75         long long current_iterations = iterations[i];
76         struct timespec start_time, end_time;
77
78         clock_gettime(CLOCK_MONOTONIC, &start_time);
79         double pi_approximation = calculate_pi_ilp(current_iterations);
80         clock_gettime(CLOCK_MONOTONIC, &end_time);
81
82         double time_spent = (end_time.tv_sec - start_time.tv_sec) +
83                             (end_time.tv_nsec - start_time.tv_nsec) / 1e9;
84         double error = fabs(M_PI - pi_approximation);
85
86         // Escreve os dados no ficheiro CSV
87         fprintf(csv_file, "ILP_Otimizado,%lld,%.18f,%.18f,%.9f\n",
88                 current_iterations, pi_approximation, error, time_spent);
89     }
90
91     // Fecha o ficheiro
92     fclose(csv_file);
93
94     printf("Resultados guardados com sucesso em pi_results.csv\n");
95     printf("Valor de referência de M_PI: %.18f\n", M_PI);
96
97     return 0;
98 }
99
```