Universidade Federal do Rio Grande do Norte - UFRN
Departamento de Engenharia da Computação e Automação - DCA
DCA3703 - Programação Paralela

Tarefa 03: Aproximação Matemática de Pi

Discente: Quelita Míriam
Docente: Samuel Xavier de Souza

I. INTRODUÇÃO

O número π é uma constante matemática fundamental em inúmeras áreas da ciência e engenharia, especialmente em problemas envolvendo geometria, trigonometria, física e computação científica. Devido à sua natureza irracional, π não pode ser representado exatamente por números decimais ou fracionários, o que motiva o desenvolvimento de algoritmos que permitam sua aproximação com diferentes níveis de precisão.

Neste trabalho, implementou-se um programa em linguagem C que calcula uma aproximação de π utilizando a série de Leibniz, variando o número de iterações e medindo o tempo de execução. O objetivo é observar a evolução da precisão da aproximação em função do poder de processamento empregado, além de refletir sobre como esse comportamento se manifesta em aplicações reais, como simulações físicas e inteligência artificial.

II. FUNDAMENTAÇÃO TEÓRICA

A série de Leibniz é uma das formas mais clássicas de aproximação de π e está representada pela seguinte fórmula infinita:

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

Imagem 1: Série de Leibniz

Portanto:

$$\pi \approx 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right)$$

Imagem 2: Simplificação da série de Leibniz

Embora seja uma forma conceitualmente simples de se calcular π, ela converge de forma lenta. Isso significa que um número elevado de iterações é necessário para se obter uma aproximação com boa precisão.

A precisão de uma aproximação numérica é comumente medida pelo erro absoluto, dado por:

$$Erro\ absoluto\ = |\pi_{real} - \pi_{aproximado}|$$

Além disso, um indicador útil para avaliar o quão próxima está a aproximação do valor real é o número de dígitos corretos consecutivos. Esse conceito é especialmente importante em computação científica e inteligência artificial, onde pequenas imprecisões podem escalar em grandes simulações ou modelos. Embora seja uma forma simples e intuitiva de aproximar π, ela converge lentamente. Por isso, é uma boa candidata para estudar como o número de iterações afeta a precisão do cálculo.

III. METODOLOGIA

O código[1] foi realizado na linguagem C. A função "*calculate_pi()*" utiliza a série de Leibniz para calcular uma aproximação de π, recebendo como parâmetro o número de iterações. Para avaliar o desempenho, foi utilizada a função "*gettimeofday()*" para medir o tempo de execução de cada cálculo. Além disso, a função "*count_correct_digits()*" foi responsável por comparar a string do valor real de π com a string da aproximação, contabilizando os dígitos corretos consecutivos.

Foram testados os seguintes números de iteração: 100, 1.000, 10.000, 100.000, 1.000.000 e 10.000.000. Para cada caso de iteração testada, foram registrados a quantidade de dígitos corretos, o valor aproximado de π, o erro absoluto e o tempo de execução em milissegundos. Esses dados permitiram avaliar a relação entre o número de iterações, a acurácia e o custo computacional.

IV. RESULTADOS E DISCUSSÕES

Os resultados obtidos a partir da implementação da série de Leibniz mostram claramente a relação entre o número de iterações e a precisão da aproximação de π, como mostrado na imagem 3. Com um número pequeno de iterações, o erro absoluto é significativo, resultando em poucos dígitos corretos na aproximação. No entanto, à medida que aumenta-se as iterações, a precisão melhora gradualmente. No início, o ganho de precisão é expressivo, como observado ao passar de 100 para 1.000 iterações, onde os dígitos corretos dobram de 1 para 2. Contudo, conforme a aproximação se torna mais precisa, a taxa de melhoria desacelera. Esse comportamento é um indicativo do fenômeno de diminuição dos retornos, onde cada novo dígito correto exige um aumento exponencial no número de iterações.

Iterações	Dígitos Corretos	Aproximação de Pi	Erro Absoluto	Tempo (ms)
100	1	3.131592903558554	0.0099997500	0.001 ms
1000	2	3.140592653839794	0.0009999997	0.004 ms
10000	3	3.141492653590034	0.0001000000	0.044 ms
100000	4	3.141582653589720	0.0000100000	0.460 ms
1000000	5	3.141591653589774	0.0000010000	3.769 ms
10000000	6	3.141592553589792	0.0000001000	40.491 ms

Imagem 3: Resultado da compilação do código fonte utilizado para a tarefa proposta

O gráfico abaixo ilustra bem essa tendência. No eixo horizontal, tem-se o número de iterações, enquanto o eixo vertical representa a quantidade de dígitos corretos na aproximação de π. Observa-se

[1] Código fonte para a tarefa em: Github: Quelita2 – <https://github.com/quelita2/programacao-paralela/blob/main/topico01/tarefa03/src/main.c>

um crescimento inicial rápido da precisão, mas, conforme avança para ordens de grandeza maiores de iterações, o ganho de novos dígitos corretos se torna mais lento. Esse padrão demonstra que, apesar de ser possível obter aproximações cada vez mais precisas, há um custo computacional crescente associado a essa melhoria. Para alcançar seis dígitos corretos, por exemplo, foram necessárias 10 milhões de iterações, um salto significativo comparado às 100.000 iterações necessárias para atingir quatro dígitos corretos.

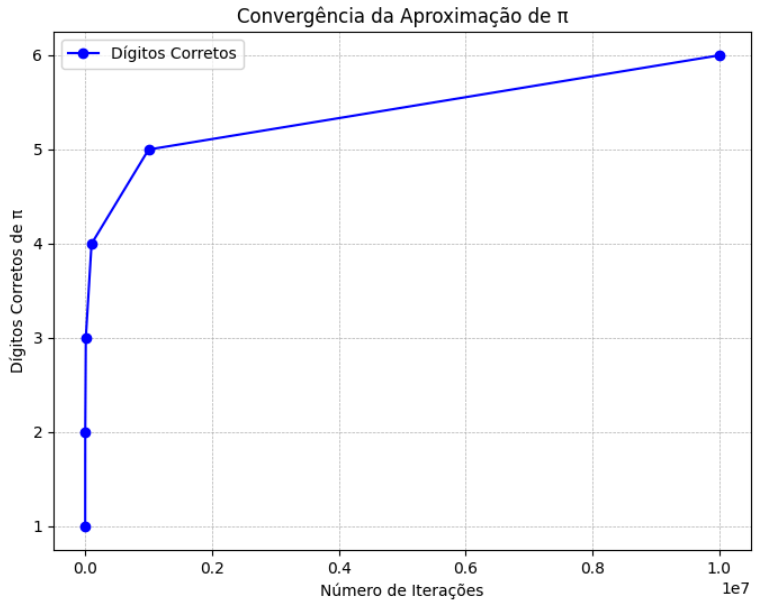

Gráfico 1: Dígitos corretos em decorrência da iteração

Esse comportamento tem implicações diretas em diversas aplicações computacionais que exigem alta precisão. Em simulações físicas, como modelos climáticos ou simulações de dinâmica de fluidos, aumentar a precisão dos cálculos pode reduzir erros acumulados e melhorar previsões, mas a um custo computacional elevado. Da mesma forma, em inteligência artificial, especialmente no treinamento de redes neurais profundas, melhorias marginais na acurácia geralmente demandam um crescimento exponencial dos recursos computacionais, evidenciando a necessidade de técnicas de otimização. Algumas aplicações requerem maior poder computacional continuamente porque lidam com grandes volumes de dados, processamento intenso e execução em tempo real. Esse desafio ressalta a importância de encontrar um equilíbrio entre precisão e eficiência, garantindo que os recursos computacionais sejam utilizados de forma inteligente e econômica.

V. CONCLUSÃO

A implementação da série de Leibniz demonstrou que a precisão na aproximação de π melhora com o aumento do número de iterações, porém de forma logarítmica. Inicialmente, pequenos aumentos nas iterações resultam em ganhos expressivos na acurácia, mas conforme a aproximação se torna mais precisa, a necessidade de poder computacional cresce exponencialmente. O gráfico evidenciou essa tendência, mostrando que a cada novo dígito correto, o custo computacional aumenta significativamente. Esse fenômeno reflete desafios presentes em diversas áreas da computação, como

simulações físicas e inteligência artificial, onde a busca por maior precisão deve ser balanceada com a viabilidade computacional. Assim, estratégias como algoritmos otimizados e computação paralela tornam-se essenciais para alcançar resultados precisos sem comprometer o desempenho.

VI. REFERÊNCIAS BIBLIOGRÁFICA

STEWART, J. *Cálculo*. 8. ed. São Paulo: Cengage Learning, 2016.