Métodos Numéricos - Monte Carlo utilizando MPI

Gustavo Aparecido de Souza Viana

***Abstract*— Métodos númericos são utilizados a fim de aproximar a solução de um determinado problema, como por exemplo resolução de integral, derivada e entre outros. De contra partida, essas técnicas de resolução são mais custosos do que soluções fechadas. Neste trabalho será abordado a implementação da resolução de Integral utilizando Monte Carlo, uma das técnicas de Integração Numérica. Os resultados mostraram que é possível estimar o valor da integral, mas dependendo da quantidade de tentativas aplicadas a região de integração pode ser que a resposta não apresente a acurácia desejada.**

I. INTRODUÇÃO

Métodos númericos são utilizados em diversas áreas como por exemplo em redes neurais para saber se a rede converge ou não, e para resolver problemas computacionais mas não de forma exata, ou seja, métodos númericos chegam a solução desejada com uma taxa de erro, assim o usuário minimizará o erro para que seja irrelevante.

Métodos númericos pelo fato de serem técnicas de otimização, ou seja, necessitam de diversas iterações para se aproximarem da solução desejada. Ou seja, a cada iteração irá se aproximando da solução final, consequentemente são métodos que possuem um alto custo computacional.

Neste trabalho foi abordado a implementação de métodos de Integração Numérica com objetivo de resolver integral, a técnica abordada foi a de Monte Carlo.

II. CONCEITOS FUNDAMENTAIS

Nesta seção, será apresentado o conceito básico de Monte Carlo e MPI utilizado na integração numérica.

A. Monte Carlo

No processo de integração usando a técnica de Monte Carlo valor da integral é aproximada por meio de valores aleatórios gerados para cada váriavel da função, validando a região de integração, a partir disso é tirado a média dos resultados calculados. A Equação 1 representa a fórmula para obter o valor da Integral e a Equação 2 a fórmula para aproximar o erro.

$$\frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{1}$$

$$V\sqrt{\frac{< f^2 > - < f >^2}{N}} \tag{2}$$

$$< f^2 >= \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)^2 \tag{3}$$

$$< f >= \frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \tag{4}$$

A Equação 5 representa o cáculo de volume, que no caso do trabalho proposto, foi feito em um toróide. Sendo que o VolumeTotal é calculado pela região de integração toda, cáculo simples de volume, já o Volume é a Equação 1 e por fim N_{total} é a quantidade de tentativas feitas na região de integração.

$$VolumeTotal * \frac{Volume}{N_{total}} \tag{5}$$

B. Message Passing Interface - MPI

Como o próprio nome diz é um passador de mensagem, criado por um grupo de pesquisadores com objetivo de disponibilizar uma forma de paralelizar a execução de um programa, sendo C, C++ ou Fortran. A ideia é que cada instância do programa rode independentemente, e assim no final de todas as execuções o resultado é disponibilizado.

Existem diversas versões do MPI, mas a príncipio ele foi criado para versões UNIX, mas com o ganho de importância no mundo de Sistema Distribuídos, uma versão Windows foi criada e é a mesma que foi utilizada no trabalho proposto.

III. METODOLOGIA

Nesta seção será apresentado a metodologia utilizada para implementar a ténica de Monte Carlo, implementados em C++. O código fonte pode ser encontrado em https://github.com/apparecidoo/ProgramacaoCientifica.

Grande maioria dos métodos utilizaram função como parâmetros, para isso foi utilizado o método *bind* para passar a função como parâmetro e para receber a classe *function* da biblioteca do C++ chamada *functional.h*.

Para a técnica de Monte Carlo foi implementado dois métodos com a responsabilidade de calcular a integral e o outro calcular a taxa de erro. Ambos métodos seguem as fórmulas citadas na sessão de Conceitos Fundamentais.

O método para calcular o erro, chamado de "monte_carlo_error", recebe três parâmetros, sendo que o primeiro consiste em ser a função que deverá ser integrada e o segundo o range de integração e o terceiro a quantidade de tentativas feitas na região de integração.

O método que implementa a Monte Carlo chamado de "monte_carlo_by_attempts_distributed" que recebe três parâmetros, o primeiro sendo a função de integração e o segundo o range de integração e o terceiro a quantidade de tentativas de valores que será feita na região de integração. Como podemos ver no pseudocódigo 1.

Algorithm 1: monte_carlo_by_attempts_distributed(Func f, IntegrationRange range, int attempts)

```
number_process = get_number_process();
// para cada processo terá um range de iteração
i = (attempts / number_process * id_process)
while i < attempts / number_process * (id_process + 1) do
    x = rand()
    result_part += f(x)
    i++
end
result = SUM(result_part)
return result / attempts
```

Outro método que implementa a técnica de Monte Carlo chamado de "monte_carlo_by_error_rate_distributed" usa como referência erro máximo que a técnica utilizada deverá apresentar para achar a solução final. O método recebe dois parâmetros, o primeiro sendo a função de integração, o segundo o erro máximo. Como podemos ver no pseudocódigo 2.

Algorithm 2: monte_carlo_by_error_rate_distributed(Func f, IntegrationRange range, double error_rate)

```
while error_numeric_method > error do
    result_previous = result;
    result = monte_carlo_by_attempts(f, range, attempts);
    if (result == result_previous)
    break;
    error_numeric_method = monte_carlo_error(f, range,
      attempts);
    attempts++;
end
return result;
```

IV. EXPERIMENTOS E RESULTADOS

Nesta seção será apresentado os experimentos e seus respectivos resultados.

A Figura 3 representa os resultados contendo o valor de integração e quantidade de tentativas na região de integração, representando a Equação 6, Equação 7 e Equação 8 de um toróide respectivamente. O Experimento executado utilizou quatro testes variando a quantidade de tentativas ou gerações de números aleatórios para região de integração, variando em 10, 100, 1000 e 10000. Com a utilização do MPI, com a utilização de 4 processos, cada processo executou a função *Attempts / 4* vezes com valores de variáveis geradas aleatoriamente e no por fim esses valores foram consignados e gerado o resultado final.

$$\int_0^1 \frac{4}{(1+x^2)} dx \tag{6}$$

```
MONTE CARLO - Master process: The number of processes available is 4

MONTE CARLO - Attempts: 10
MONTE CARLO - Master process: Integral value is 2.65671
MONTE CARLO - Wall clock elapsed seconds: 0.0053334

MONTE CARLO - Attempts: 100
MONTE CARLO - Master process: Integral value is 3.16743
MONTE CARLO - Wall clock elapsed seconds: 0.0069935

MONTE CARLO - Attempts: 1000
MONTE CARLO - Master process: Integral value is 3.14405
MONTE CARLO - Wall clock elapsed seconds: 0.0261163

MONTE CARLO - Attempts: 10000
MONTE CARLO - Master process: Integral value is 3.15786
MONTE CARLO - Wall clock elapsed seconds: 0.0365733
```

Fig. 1: Representação dos resultados para cada Equação 6

$$\int_0^1 \sqrt{x+\sqrt{x}} dx \tag{7}$$

```
MONTE CARLO - Master process: The number of processes available is 4

MONTE CARLO - Attempts: 10
MONTE CARLO - Master process: Integral value is 0.800263
MONTE CARLO - Wall clock elapsed seconds: 0.0292938

MONTE CARLO - Attempts: 100
MONTE CARLO - Master process: Integral value is 1.04563
MONTE CARLO - Wall clock elapsed seconds: 0.0284921

MONTE CARLO - Attempts: 1000
MONTE CARLO - Master process: Integral value is 1.04917
MONTE CARLO - Wall clock elapsed seconds: 0.0173203

MONTE CARLO - Attempts: 10000
MONTE CARLO - Master process: Integral value is 1.04461
MONTE CARLO - Wall clock elapsed seconds: 0.0225868
```

Fig. 2: Representação dos resultados para cada Equação 7

$$\int_1^4 \int_{-3}^4 \int_{-1}^1 z^2 + (\sqrt{x^2+y^2}-3)^2 dxdydz \tag{8}$$

```
MONTE CARLO - Master process: The number of processes available is 4

MONTE CARLO - Attempts: 10
MONTE CARLO - Master process: Volume of Integral value is 13.9924
MONTE CARLO - Wall clock elapsed seconds: 0.0086267

MONTE CARLO - Attempts: 100
MONTE CARLO - Master process: Volume of Integral value is 21.158
MONTE CARLO - Wall clock elapsed seconds: 0.0737741

MONTE CARLO - Attempts: 1000
MONTE CARLO - Master process: Volume of Integral value is 21.0774
MONTE CARLO - Wall clock elapsed seconds: 0.222029

MONTE CARLO - Attempts: 10000
MONTE CARLO - Master process: Volume of Integral value is 20.8186
MONTE CARLO - Wall clock elapsed seconds: 0.0867223
```

Fig. 3: Representação dos resultados para cada Equação 8

V. CONCLUSÃO

Após a análise do conceito e resultados obtidos após a aplicação do métodos comparando os mesmos, podemos concluir que todas as técnicas utilizadas conseguiram atingir o objeto de integração e que dependendo do número de tentativas que é feita na região de integração temos uma precisão com uma maior acurácia. Além disso, outro fator muito importante que impacta nos resultados é a geração

dos números aleatórios como valores de entrada para as váriaveis da função, caso esses números não forem dispersos, ou seja, não atender grande parte da região de integração os resultados poderão ser valores totalmente incorretos. E com a utilização do paralelismo (MPI) o tempo de execução do método diminuiu.

REFERENCES