Uma Abordagem Paralela para o Cálculo da Integral Utilizando as Regras de Newton-Cotes

Murillo Freitas Bouzon

***Resumo*—Uma alternativa para resolver integrais em problemas computacionais é o uso de integrais numéricas, também chamada de quadratura, onde é feita a aproximação da área sobre a curva de uma função. As regras de Newton-Cotes são um grupo fórmulas para integração numérica, aplicado em pontos igualmente espaçados, porém se a quantidade de pontos for muito alta, o resultado pode acabar demorando muito para retornar a resposta. Para melhorar o tempo, pode-se calcular a área entre os pontos de forma paralela, distribuindo os pontos entre *threads*. Para realizar este tipo de programa, existe o padrão openMP criado para facilitar o desenvolvimento de programas que são executados de forma paralela. Sendo assim, neste trabalho é implementado as regras de Newton-Cotes para integração numérica de forma paralela, utilizando o openMP para melhorar o tempo de processamento.**

***Index Terms*—Integração numérica, Regra de Simpson, Regra do ponto médio, Regra do trapézio, Programação Paralela, OpenMP**

I. INTRODUÇÃO

Computadores que possuem mais de um núcleo em seu processador é algo muito comum nos dias de hoje. Isso ocorre devido à necessidade cada vez maior de computação paralela para as diversas aplicações que são utilizadas, onde os processos são divididos entre vários núcleos para obter um ganho de performance.

Com isso, programas que utilizam paralelismo começaram a ser escritos de outra forma em relação aos programas sequenciais, tendo que especificar as partes do código que serão executados em paralelo. Desta forma, é necessário que o programador evite que tarefas dependentes sejam executadas de forma paralela.

Para facilitar o desenvolvimento de programas que utilizam paralelismo, foram criados padrões e interfaces de programação paralela. Uma dessas interfaces é o *openMP (Open Multi-Processing)*, uma biblioteca para programação paralela a partir de diretivas de compilação e variáveis de ambiente.

Um método que tem como natureza ser facilmente paralelizável é o método integração numérica utilizando as regras de Newton-Cotes, onde divide o intervalo desejado em N sub-intervalos para serem calculados e então serem juntados para formar a solução final.

Portanto, este trabalho tem como objetivo a implementação das regras de Newton-Cotes para realizar a integração numérica de uma função de forma paralela com o intuito de melhorar a performance do algoritmo original.

II. FUNDAMENTAÇÃO TEÓRICA

Neste seção serão apresentados os conceitos a respeito das técnicas utilizadas na realização deste trabalho.

A. Regra do ponto médio

A regra do ponto médio ou dos retângulos é uma das fórmulas de Newton-Cotes aberta para integração numérica. Pode ser descrita pela Equação (1). A Figura 1 mostra um exemplo da aplicação da regra do ponto médio.

$$\int_a^b f(x)dx \simeq (b-a)f(\frac{a+b}{2}) \tag{1}$$

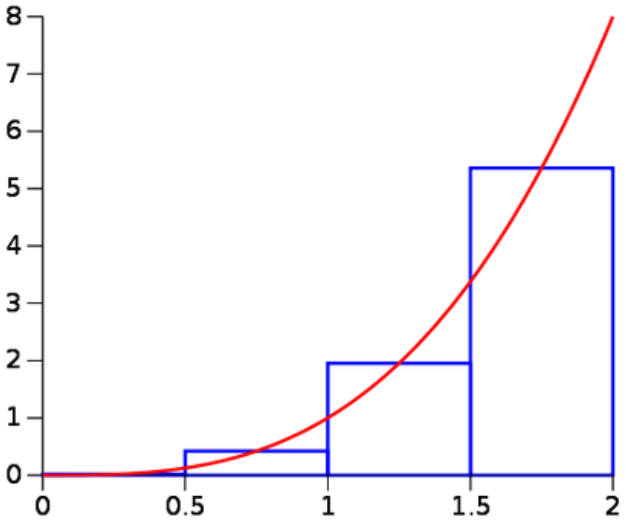

Figura 1: Ilustração da integração numérica pela regra do ponto médio, utilizando $n = 3$.

B. Regra dos Trapézios

A regra dos trapézios é uma das fórmulas de Newton-Cotes fechada para integração numérica, onde a integral é dada pela soma da área de n trapézios. Pode ser descrita pela Equação (2) [1]. A Figura 2 mostra um exemplo da aplicação da regra dos trapézios.

$$\int_a^b f(x)dx \simeq (b-a)\frac{f(a)+f(b)}{2} \tag{2}$$

C. Regra de Simpson

A regra de Simpson é outra formula de Newton-Cotes fechada para integração numérica, onde a ideia é dividir f(x) em dois intervalos para ser aproximada por uma parábola neste intervalo. Pode ser descrita pela Equação (3) [1]. A Figura 3 mostra um exemplo da aplicação da regra de Simpson.

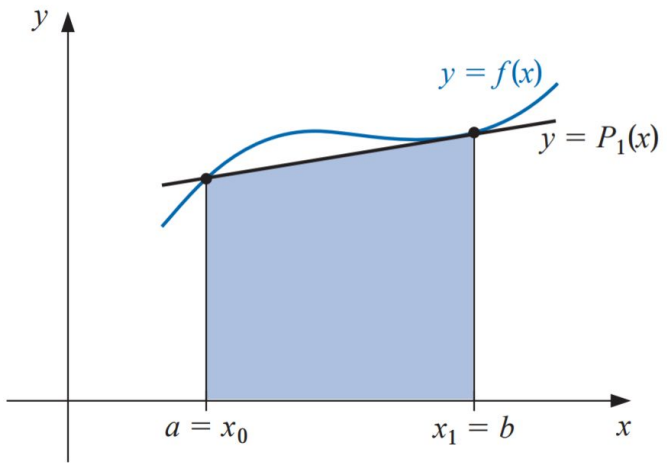

Figura 2: Ilustração da integração numérica pela regra dos trapézios, utilizando $n = 1$. Fonte: [1].

$$\int_a^b f(x)dx \simeq (b-a)\frac{f(a) + 4 * f(\frac{a+b}{2}) + f(b)}{6} \quad (3)$$

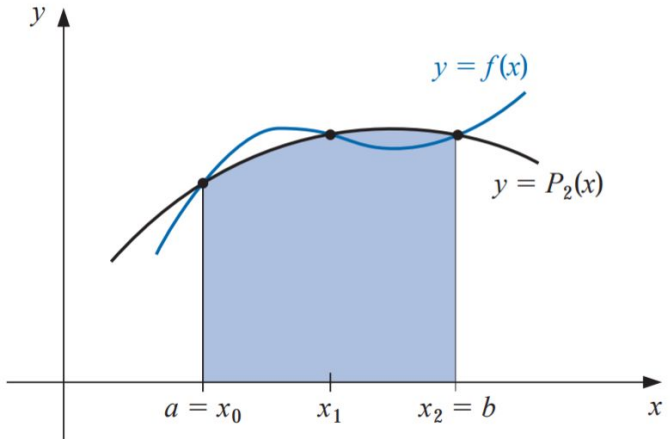

Figura 3: Ilustração da integração numérica pela regra de Simpson, utilizando $n = 1$. Fonte: [1]

D. OpenMP

O OpenMP é uma API (*Application Programming Interface)* proposta em [2], sendo um conjunto de diretivas de compilação e uma biblioteca de rotinas chamáveis em tempo de execução do programa, sendo criada originalmente para a linguagem Fortran e depois estendida para as linguagens C/C++.

Ela funciona de acordo com conceito de *multithreading*, onde existe uma *thread* mestre, marcada pelo "id"0, que divide uma tarefa para N *threads* "escravas"que também são marcadas por um "id". Cada *thread* é então executada simultaneamente.

Para executar uma parte do código em paralelo, é preciso marcá-la com uma diretriz de compilação que cria as *threads* que receberão as partes da tarefa a ser dividida, executando por padrão cada parte de forma independente.

A seguir é mostrado um exemplo de diretiva de compilação para executar um trecho do código em paralelo:

- #pragma omp parallel

III. METODOLOGIA

Para a metodologia deste trabalho foi feita a implementação de uma classe Function representando a função dada pela Equação (4).

$$f(x) = e^{-x^2} \quad (4)$$

Esta classe possui como atributos:

- a: limite inferior da integral
- b: limite superior da integral

Esta classe possui os métodos Midpoint(), Trapezoidal() e Simpson() que utilizam as suas respectivas fórmulas apresentadas na Seção II, recebendo como parâmetro um inteiro N e um inteiro T, sendo o número de sub-intervalos que a função será dividida e o número de *threads* que serão utilizadas, retornando como resposta o valor da integral.

IV. EXPERIMENTOS E RESULTADOS

Para validar a metodologia deste trabalho, foi realizado um experimento para avaliar o tempo necessário para realizar o cálculo da integral para cada um dos métodos de Newton-Cotes. O valor sub-intervalos N utilizado foi variado de 10 à 10^7 e o número de núcleos T utilizados foi variado entre 2 e 32.

A Figura 4 apresenta um gráfico com os resultados obtidos pela regra do Ponto Médio, onde o eixo x representa o número N de divisões feitas, o eixo y o tempo e mili segundos e cada cor representa a quantidade T de *threads* utilizadas.

Figura 4: Resultado do método do Ponto Médio.

O gráfico mostra que para $N \geq 10^6$ o tempo começa a crescer de forma exponencial para $T = 1$. Para $T = 32$ o tempo fica bem próximo com os valores de $T = 4$ e $T = 8$, sendo que o menor tempo obtido para o maior valor de N foi com $T = 16$.

Figura 5: Resultado do método dos Trapézios.

A Figura 5 apresenta um gráfico com os resultados obtidos pela regra dos Trapézios, onde os eixos representam os mesmos atributos da Figura 4.

Observando o gráfico, pode-se afirmar que o tempo de processamento da regra dos trapézios também cresceu de forma exponencial para $N \geq 10^6$, sendo que com o valor de $T = 32$ e $T = 16$ foi obtido um tempo muito próximo do resultado de $T = 8$ devido à quantidade de núcleos que o computador utilizado para teste ser igual a 8.

Por fim, a Figura 6 apresenta um gráfico com os resultados obtidos pelo método de Simpson, que possui os mesmos eixos dos gráficos anteriores. O gráfico mostra que o método de Simpson foi o método que levou mais tempo para $T = 1$ e $N = 10^7$, obtendo um valor de ≈ 1.4 segundos. Assim como no método dos trapézios, o tempo obtido com $T = 32$ e $T = 16$ se manteve bem próximo de $T = 8$ e também de $T = 4$ devido a quantidade máxima de núcleos do computador utilizado.

Figura 6: Resultado do método de Simpson.

V. Trabalhos Relacionados

A literatura aponta que a programação paralela têm sido cada vez mais frequente nos diversos estudos que são feitos atualmente, devido à necessidade de paralelizar o processamento em cima de dados gigantescos, sendo aplicado nos mais diversos problemas computacionais.

Um exemplo é o trabalho de [3] que propôs a paralelização do algoritmo *QuickSort* utilizando o OpenMP. A ideia é ler uma palavra como entrada e dividi-la em diversos *sub-arrays* de acordo com o número de caracteres que ela possui. Foram utilizados duas bases de dados para avaliar o método e variado o número de *threads* entre $1, 2$ e 4. Os resultados mostraram que em média o tempo foi reduzido quase pela metade, provando a eficiência do método.

Outro trabalho foi o de [4] onde foi proposto uma metodologia baseada no algoritmo SIFT e em programação paralela para construção automática de mosaicos, utilizando imagens aéreas de alta resolução de regiões agrícolas. Foi utilizado o OpenMP para paralelizar as etapas de cálculo do descritor do SIFT, paralelizando cerca de 75% do código sequencial e reduzindo o tempo de construção de mosaicos em 60% utilizando 8 *threads*.

No trabalho de [5] foi proposto uma abordagem paralela da implementação do algoritmo SMO (*Sequential Minimal Optimization* utilizando OpenMP. Foi utilizado uma base de dados com 3175 amostras e uma dimensionalidade de tamanho 60. Os resultados mostraram que foi possível obter um aumento de velocidade maior que 1.5, aprimorando o método SMO quando utilizado em grandes bases de dados.

VI. Conclusão

Neste trabalho foi feita a implementação da regra do ponto médio, dos trapézios e de Simpson para integração numérica de forma paralela, utilizando o padrão OpenMP para paralelizar os métodos.

Os métodos foram avaliados variando o número de núcleos utilizados e o número de divisões feitas para realizar a quadratura. Os resultados mostraram que foi possível melhorar o tempo de resposta para valores de $N \geq 10^6$, porém não houve muita variação nos resultados de $T = 8$, $T = 16$ e $T = 32$ devido ao número de núcleos que a máquina utilizada possui. Em trabalhos futuros pode ser feita a comparação com outros método numéricos e utilizar outros padrões de programação paralela.

Referências

[1] Richard L. Burden and John Douglas Faires. Numerical analysis, 9th edition. Brooks/Cole, Cengage Learning.

[2] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. 1998.

[3] Sinan Sameer Mahmood Al-Dabbagh and Nawaf Hazim Barnouti. Parallel quicksort algorithm using openmp. 2016.

[4] André de Souza Tarallo, Alan Kazuo Hiraga, Germán Andrés Gaviria Martínez, Maria Stela Veludo de Paiva, Lúcio André de Castro Jorge, and Hermes Senger. Parallel processing applied to image mosaic generation. 2013.

[5] Pengfei Chang, Zhuo Bi, and Yiyong Feng. Parallel smo algorithm implementation based on openmp. *2014 IEEE International Conference on System Science and Engineering (ICSSE)*, pages 236–240, 2014.