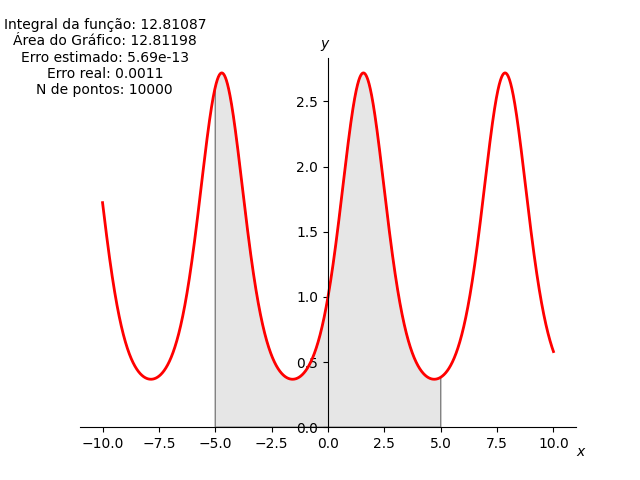
**Relatório trabalho A1 Programação Paralela**

Aluno: João Pedro Silva de Lima

Este relatório tem como objetivo detalhar o passo-a-passo do programa criado em python com o propósito de calcular a área da uma determinada função, em um limite definido. Para encontrar essa área, nós utilizamos a Integral da função, que, dados os limites de integração, nos retorna a área naquela seção.



No gráfico acima podemos destacar duas informações: A linha vermelha, que é a visualização da função exp(sen(x)) ao longo do eixo das abscissas. A área acinzentada representa a área que vamos analisar, situada entre -5 e 5 do eito x, sendo a área que queremos calcular.

A visualização, assim como o cáculo da Integral da função, foram realizadas utilizando uma biblioteca científica, que retorna, a partir de uma função e um intervalo, o valor da integral (área da função) e o seu erro (Erro estimado). Mas, para podermos controlar a precisão do gráfico e então cotrolar seu erro, precisaremos de mais liberdade para escolhermos a forma de calcular a área da função. Então, utilizando de vários retângulos com tamanhos pré-determinados (distância entre os pontos definida de acordo com a quantiade de pontos pedida pelo usuário), são somados todos os retângulos para conseguir a área do gráfico entre os limites. Este modelo é mais simples, mas perde precisão por conta do alto erro. Outra possível solução seria utilizar trapezoides, sendo a precisão mais entre dois pontos de um gráfico maior ao se utilizar um trapézio.

Para o código, utilizamos o matplotlib para desenhar os gráficos, o numpy para realizar operações com vetores/listas e o scypy para realizar o cáculo da integral de uma função de forma programática, sendo utilizado para analisar a precisão e resultado da minha própria função exponencial.

Cálculo de speedup esperado utilizando multithreading/multiprocessamento

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Qtd. pontos** | **Tempo real (ms)** | **Tempo para calcular um ponto (ms)** | **Speedup esperado (CPU infinita)** | **Speedup esperado (4 processos)** |
| 5.000,00 | 2,75 | 2,59E-06 | 106.382.887% | 5419,144366% |
| 50.000,00 | 42,70 | 2,59E-06 | 1.651.836.105% | 8359,180525% |
| 500.000,00 | 322,69 | 2,59E-06 | 12.483.161.422% | 6341,580711% |

Considerei primeiro uma máquina com threads infinitas, podendo calcular todos os nós simultaneamente. Na sequência, uma máquina utlizando 4 processos para calcular 4 nós por vez. Nos dois exemplos, estamos reduzindo, na teoria, a quantidade de pontos a serem calculados (podemos ver pela perspectiva do tempo, para fazer mais sentido), assim sendo, quanto mais processadores, menor será o tempo de espera para a conclusão dos cálculos.