

Integral Numerica - Metodo do Trapezio Simples - Implementacao Paralela com MPI

June 10, 2019

```
In [ ]: """
Nome: Vinícius Barreto de Sousa Neto

Entrada: N.A.
Saída: Aproximação da integral no intervalo  $[a, b]$  da função  $f(x)$ 
       utilizando o método do trapézio simples com  $n$  subdivisões.

Algoritmo:
1. Cada processo calcula sua partição local do intervalo  $[a, b]$ 

2. Cada processo calcula a aproximação da integral conforme sua partição
   local do intervalo  $[a, b]$ .

3a. Cada processo de rank  $\neq 0$  envia sua aproximação da integral para o
    processo de rank 0.

3b. O processo de rank 0 soma as integrais "parciais" enviadas a ele por
    cada outro processo e exibe o resultado.

Obs.:  $n/p$  deve ser exato. O número de subdivisões deve ser igual para cada processo.

int    my_rank    Rank do processo
int    p          Número de processos
float  a          Limite inferior da integral
float  b          Limite superior da integral
int    n          Número de subdivisões (trapézios)
float  h          Tamanho da base do trapézio (tamanho da subdivisão)
float  local_a    Limite inferior da partição local do intervalo em um processo
float  local_b    Limite superior da partição local do intervalo em um processo
int    local_n    Número de subdivisões (trapézios) para cálculo local
float  integral   Integral no intervalo
float  total      Integral total
int    source     Processo de origem
int    dest       Processo de destino
"""
```

```

In [ ]: def f(x):
        return 4/(1+x*x)

In [ ]: def trap(a, b, n, h):

        integral = (f(a) + f(b))/2.0

        x = a

        for i in range(1, int(n)):
            x = x + h
            integral = integral + f(x)

        return integral * h

In [ ]: from mpi4py import MPI

        comm = MPI.COMM_WORLD
        my_rank = comm.Get_rank()
        p = comm.Get_size()

        a = 0.0
        b = 1.0
        n = 1024
        dest = 0
        total = -1.0

In [ ]: h = (b-a)/n
        local_n = n/p

In [ ]: local_a = a + my_rank * local_n * h
        local_b = local_a + local_n * h
        integral = trap(local_a, local_b, local_n, h)

In [ ]: if my_rank == 0:

        total = integral
        for source in range(1, p):
            integral = comm.recv(source=source)
            print("PE ", my_rank, "<-", source, ",", integral, "\n")
            total = total + integral

        else:

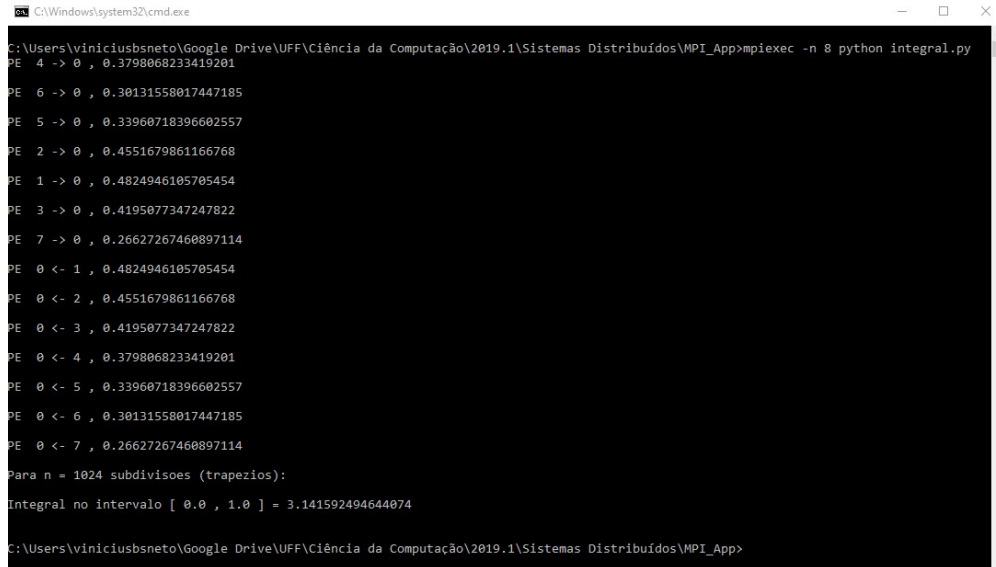
            print("PE ", my_rank, "->", dest, ",", integral, "\n")
            comm.send(integral, dest=0)

        if my_rank == 0:

```

```
print("Para n =", n, "subdivisoas (trapezios): \n")
print("Integral no intervalo [",a,",", b, "] =", total, "\n")
```

MPI.Finalize



```
C:\Windows\system32\cmd.exe
C:\Users\viniciusbsneto\Google Drive\UFF\Ciência da Computação\2019.1\Sistemas Distribuídos\MPI_App>mpiexec -n 8 python integral.py
PE 4 -> 0 , 0.3798068233419201
PE 6 -> 0 , 0.30131558017447185
PE 5 -> 0 , 0.33960718396602557
PE 2 -> 0 , 0.4551679861166768
PE 1 -> 0 , 0.4824946105705454
PE 3 -> 0 , 0.4195077347247822
PE 7 -> 0 , 0.26627267460897114
PE 0 <- 1 , 0.4824946105705454
PE 0 <- 2 , 0.4551679861166768
PE 0 <- 3 , 0.4195077347247822
PE 0 <- 4 , 0.3798068233419201
PE 0 <- 5 , 0.33960718396602557
PE 0 <- 6 , 0.30131558017447185
PE 0 <- 7 , 0.26627267460897114
Para n = 1024 subdivisoas (trapezios):
Integral no intervalo [ 0.0 , 1.0 ] = 3.141592494644074
C:\Users\viniciusbsneto\Google Drive\UFF\Ciência da Computação\2019.1\Sistemas Distribuídos\MPI_App>
```