Programação Paralela com MPI

ELC139 - Programação Paralela

João Vicente Ferreira Lima (UFSM)

Universidade Federal de Santa Maria
jvlima@inf.ufsm.br
http://www.inf.ufsm.br/~jvlima

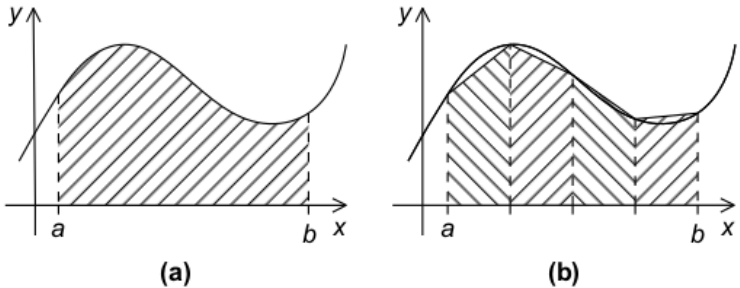
2023/1

Outline

Outline

Aproximação trapezoidal

- Cálculo para aproximar o valor da integral de uma função $f(x)$ através da soma das áreas dos sub-intervalos
- Quanto maior o número de trapézios
  - Melhor aproximação
  - Maior custo computacional

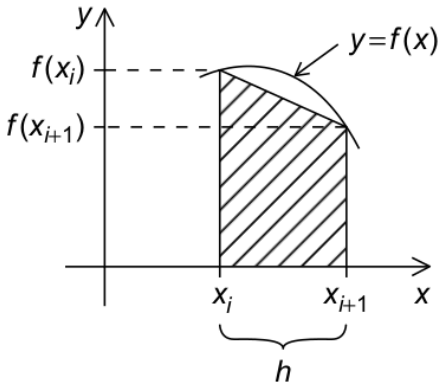


Aproximação trapezoidal

- Área de um trapezoide $= \frac{h}{2}[f(x_i) + f(x_{i+1})]$
- $h = \frac{b-a}{n}$
- $x_0 = a$, $x_1 = a+h$, $x_2 = a+2h$, ..., $x_{n-1} = a+(n-1)h$, $x_n = b$
- Soma das áreas dos trapezoides:

$$h[\frac{f(x_0)}{2} + f(x_1) + f(x_2) + ... + f(x_{n-1}) + \frac{f(x_n)}{2}]$$

Aproximação trapezoidal



An Introduction to Parallel Programming, Peter Pacheco, 2011.

Aproximação trapezoidal

Pseudo-código sequencial da aproximação trapezoidal.

```
1  /* Input: a, b, n */
2  h = (b - a)/n;
3  approx = (f(a) + f(b))/2.0;
4  for (i = 1; i <= n - 1; i++) {
5    x i = a + i*h;
6    approx += f(x i);
7  }
8  approx = h*approx;
```

Aproximação trapezoidal paralela

- Particionar o problema em tarefas.
- Identificar dependências entre tarefas.
- Agregar tarefas em tarefas compostas.
- Mapear tarefas compostas para processadores.

Aproximação trapezoidal paralela

Pseudo-código paralelo.

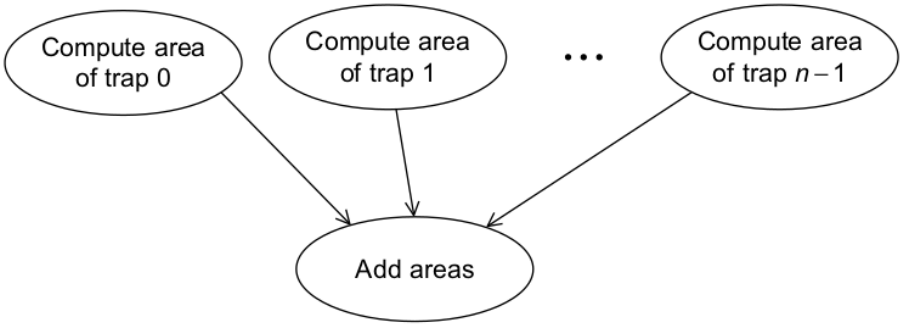
```
1 Get a, b, n;
2 h = (b - a)/n;
3 local_n = n/comm_sz;
4 local_a = a + my_rank*local_n*h;
5 local_b = local_a + local_n*h;
6 local_integral = Trap(local_a, local_b, local_n, h);
```

Aproximação trapezoidal paralela

Pseudo-código paralelo.

```
1  if (my_rank != 0)
2    Send local_integral to process 0;
3  else /* my_rank == 0 */
4    total_integral = local_integral;
5    for (proc = 1; proc < comm_sz; proc++) {
6      Receive local_integral from proc;
7      total_integral += local_integral;
8    }
9  }
10 if (my_rank == 0)
11   print result;
```

Aproximação trapezoidal



An Introduction to Parallel Programming, Peter Pacheco, 2011.

Aproximação trapezoidal sequencial

```
1  double Trap(double a, double b, int n, double h) {
2    double integral;
3    int k;
4
5    integral = (f(a) + f(b))/2.0;
6    for (k = 1; k <= n-1; k++) {
7      integral += f(a+k*h);
8    }
9    integral = integral*h;
10
11   return integral;
12 }  /* Trap */
```

Aproximação trapezoidal com MPI

```
1  int main(void) {
2    int my_rank, comm_sz, n = 1024, local_n;
3    double a = 0.0, b = 3.0, h, local_a, local_b;
4    double local_int, total_int;
5    int source;
6
7    MPI_Init(NULL, NULL);
8    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
9    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
10
11   h = (b-a)/n;
12   local_n = n/comm_sz;
13   local_a = a + my_rank*local_n*h;
14   local_b = local_a + local_n*h;
15   local_int = Trap(local_a, local_b, local_n, h);
16
```

Aproximação trapezoidal com MPI

```
1  int main(void) {
2    int my_rank, comm_sz, n = 1024, local_n;
3    double a = 0.0, b = 3.0, h, local_a, local_b;
4    double local_int, total_int;
5    int source;
6
7    MPI_Init(NULL, NULL);
8    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
9    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
```

Aproximação trapezoidal com MPI

```
1   h = (b-a)/n;
2   local_n = n/comm_sz;
3   local_a = a + my_rank*local_n*h;
4   local_b = local_a + local_n*h;
5   local_int = Trap(local_a, local_b, local_n, h);
```

Aproximação trapezoidal com MPI

```
1   /* Add up the integrals calculated by each process */
2   if (my_rank != 0) {
3       MPI_Send(&local_int, 1, MPI_DOUBLE, 0, 0,
4             MPI_COMM_WORLD);
5   } else {
6       total_int = local_int;
7       for (source = 1; source < comm_sz; source++) {
8         MPI_Recv(&local_int, 1, MPI_DOUBLE, source, 0,
9             MPI_COMM_WORLD, MPI_STATUS_IGNORE);
10        total_int += local_int;
11      }
12  }
```

Aproximação trapezoidal com MPI

```
1     /* Print the result */
2     if (my_rank == 0) {
3         printf("With n = %d trapezoids, our estimate\n", n);
4         printf("of the integral from %f to %f = %.15e\n",
5             a, b, total_int);
6     }
7
8     /* Shut down MPI */
9     MPI_Finalize();
10
11    return 0;
12  } /*  main  */
```

informática
Universidade Federal de Santa Maria

Universidade Federal de Santa Maria
1960

https://joao-ufsm.github.io/par2023a/