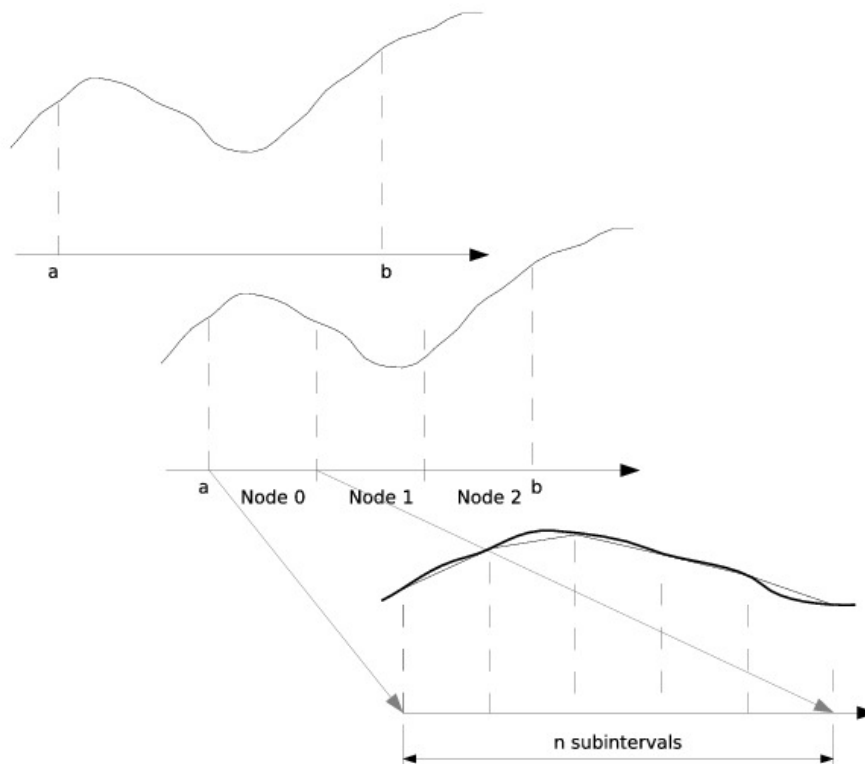


Parallel Numerics

Exercise 2: Numerical Integration & P2P Communication I

1) Numerical Integration



We are to calculate the definite integral of a function $f(x)$ in an interval $[a, b]$ using the trapezoidal rule. Before, we split up the overall interval $[a, b]$ into $\#proc$ equidistant subintervals $[a_k, b_k]$, $k \in \{0, 1, \dots, \#proc - 1\}$. $\#proc$ is the number of nodes available on a parallel computer. Afterwards, we divide the interval $[a_k, b_k]$ into n intervals with length $h = (b_k - a_k)/n$

and approximate the integral by a sum of trapezoids:

$$\begin{aligned} \int_{a_k}^{b_k} f(x) dx &\approx \sum_{i=1}^n \frac{h}{2} \left(f(a_k + (i-1)h) + f(a_k + ih) \right) \\ &= \frac{h}{2} \left(f(a_k) + f(b_k) \right) + h \cdot \sum_{i=1}^{n-1} f(a_k + ih) \end{aligned} \quad (1)$$

- i) Implement the algorithm using the MPI send and receive operations.
- ii) Test the application for $f(x) = -3x^2 + 3$, $[a, b] = [-1, 1]$, $n = 2, 4, 8$, and different numbers of processors.
- iii) Test the application for $f(x) = x \sin(10x)$, $[a, b] = [0, 2\pi]$, $n = 2, 4, 8$ and different numbers of processors.

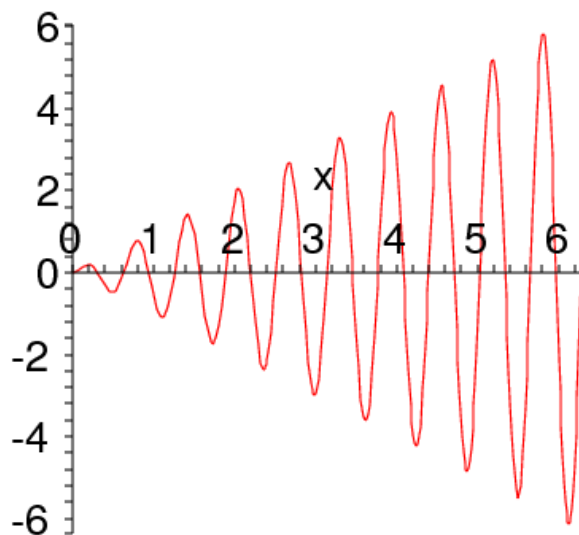


Figure 1: $f(x) = x \sin(10x)$.

Compute the analytical solution and compare it to the numerical one. What do you observe and why?

- iv) Implement the following strategy for determining n on a node:
 - $n = 1$
 - Compute $F_{numerical}^{(1)}$ (see Formula 1)
 - $n = 2$
 - Compute $F_{numerical}^{(2)}$ (see Formula 1)
 - $n = 3$
 - Compute $F_{numerical}^{(3)}$ (see Formula 1)

- ...
- Quit if $F_{numerical}^{(k+1)} - F_{numerical}^{(k)} < \epsilon$

Test the algorithm for

$$f(x) = \begin{cases} 0 & \text{for } x \leq \frac{1}{2} \\ 1 & \text{for } x > \frac{1}{2} \end{cases}, \quad (2)$$

$[a, b] = [0, 1]$, $n = 2, 3, 4$ and different values for ϵ . Try to find out, how many iterations per node are performed. Is there a bottleneck?

- v) Before the application terminates, each node sends its contribution to the numerical result to node 0. You have used MPI send and receive to implement this behaviour. Is there a technical term for such a communication scheme?

2) MPI_Send and MPI_Recv

- i) Write an operation sequence that receives four integers from arbitrary senders (use the constant `MPI_ANY_SOURCE` as source number for the `MPI_Recv` operation) using the message tag 0 for each of them. Store the results in four integers `b0, b1, b2` and `b3`.
- ii) Consider the following code snippet

```
int a=0;
MPI_Send(&a, 1, MPI_INT, 2, 0, MPI_COMM_WORLD);
a=1;
MPI_Send(&a, 1, MPI_INT, 2, 0, MPI_COMM_WORLD);
```

running on node 0, and

```
int a=2;
MPI_Send(&a, 1, MPI_INT, 2, 0, MPI_COMM_WORLD);
a=3;
MPI_Send(&a, 1, MPI_INT, 2, 0, MPI_COMM_WORLD);
```

running on node 1. What are possible values for `b0, b1, b2` and `b3` when executing the four `MPI_Recv` operations from i) on node 2? Is there a unique solution? Are messages allowed to overtake?

- iii) Define the term *blocking* for MPI commands.
- iv) Define the term *not synchronized* for MPI commands.

3) Predefined MPI datatypes

For every MPI send or receive command, one has to specify a datatype. Make yourself familiar with the datatypes offered by the MPI interface. To what C datatypes do they correspond?