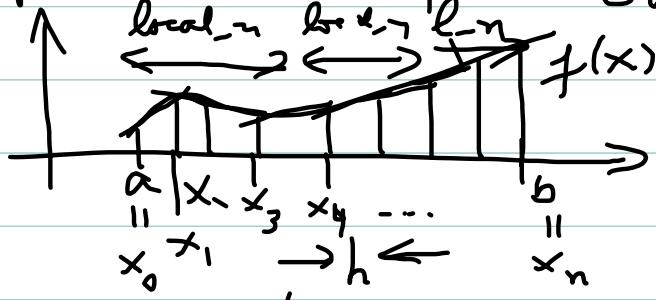
Monday, 12/12/11

Last time: Greetings program from Ch. 3
⇔ first 'real' MPI program
⇔ first program with MPI communications
MPI_Send / MPI_Recv

Ch. 4 Trapezoidal Rule

Example of calculating something useful (potentially) faster than in serial

Trapezoidal rule for $I = \int_a^b f(x)\,dx$



$$I_n = \int_{x_0}^{x_1} f + \int_{x_1}^{x_2} f + \int_{x_2}^{x_3} f + \cdots + \int_{x_{n-1}}^{x_n} f$$

$$\approx \frac{f(x_0)+f(x_1)}{2}h + \frac{f(x_1)+f(x_2)}{2}h + \cdots + \frac{f(x_{n-1})+f(x_n)}{2}h$$

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

Error $|I - I_n| = O(h^2)$ second-order

Task: Parallelize this method

Idea: Divide up the work of evaluating the function values $f(x_i)$ and also accumulate local sums of terms in the summation

Here, in context of quadrature think of each local sum as a trapezoidal rule for a subinterval of $[a, b]$.

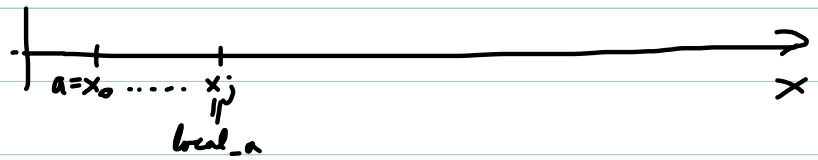
Input: a, b, $f(x)$ for $I = \int_a^b f\,dx$
and n = numerical parameter

Note: making $f(x)$ an actual input in C is too hard $\Rightarrow$ hardwire $f(x) = x^2$

Define variables: float local_In, In
$h = (b-a)/n$
MPI_Comm_size $\Rightarrow p = np$
MPI_Comm_rank $\Rightarrow$ id

Do in parallel on all processes:
Compute local trap_rule from local_a to local_b with local_n subintervals $= n/p$ for p parallel processes.

Collect results from all process onto Process 0 and sum up to find I_n.



What is j here?

On Process 0, $x_0, x_1, \ldots, x_{local\_n}$

1, $x_{local\_n}, x_{local\_n+1}, \ldots, x_{2\,local\_n}$

$\Rightarrow j = local\_n * id$

$\Rightarrow local\_a = a + h * local\_n * id$

$local\_b = a + h * local\_n * (id+1)$

$local\_In = trap(local\_a, local\_b, local\_n, h)$

Idea: Send local_In to Process 0

```
if (id == 0) {
    In = local_In
    for (i = 1; i < np; i++) {
    MPI_Recv(&local_In, 1, MPI_FLOAT, i, 0, ....)
    In += local_In
    }
} else {
  MPI_Send(&local_In, 1, MPIFLOAT, 0, 0, MPI_COMM_WORLD)
}
```

pass-by-reference

dest