Tuesday, 04/17/12:

Ch. 3 and 4 from Pacheco:

MPI = Message Passing Interface

(=) Idea of MPI is to make all communications between parallel processes explicit, meaning the programmer has to explicitly request each communication.

Programming model of SPMD = single program multiple data

really: single program, but code can take a different path on each process. =>

You can find out your own process rank id and make decisions based on it or do algebra depending on it.

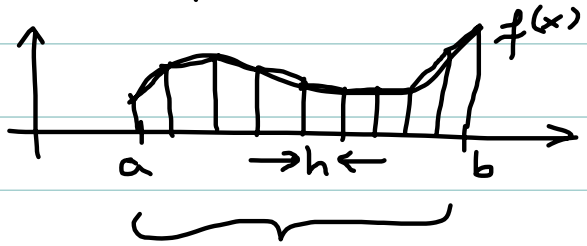
Design most basic MPI program:

```
#include <mpi.h>
int id, np;

MPI_Init(&argc, &argv)        // argc, argv: command-line arguments
 MPI_Comm_rank (MPI_COMM_WORLD, &id)
 MPI_Comm_size (MPI_COMM_WORLD, &np)
        :

MPI_Finalize()
```

=> We downloaded hello_parallel.c and developed greetings.c from it by putting MPI_Send and MPI_Recv in it.

This code is equivalent to Ch. 3 Pacheco, but with some aspects more conventional.

Ch. 4 Trapezoidal Rule (really: composite trapezoidal rule)



mesh points

$x_i = a + ih$,

$i = 0, \ldots, n$

$h = (b-a)/n$

n subintervals, area under $f(x)$ approximated by one trapezoid each.

$$\underbrace{\int_a^b f(x)\,dx}_{=I} \approx \underbrace{\frac{h}{2}\left[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)\right]}_{=I_n}$$

with error of order h^2

Idea of parallel code: Compute trap. rule for $\frac{n}{np}$ sub-intervals on each process, then send each local result to Process 0, which adds them up for global result.

Assume have serial function trap.

```
int id, np, j, local_n
double a, b, h, local_a, local_b, In, ... ...

h = (b-a)/n
local_n = n/np
local_a = a + id * (h*local_n)
local_b = local_a + h* local_n
local_In = trap(local_a, local_b, local_n

if (id == 0) {
    In = local_In
    for (j = 1; j < np; j++) {
        MPI_Recv(&local_In, 1, MPI_DOUBLE,
                 MPI_ANY_SOURCE, .....)
        In += local_In
    }
} else {
  MPI_Send(&local_In, 1, MPIDOUBLE, 0, ...)
}
if (id == 0) {
    printf("In = %24.16e \n", In)
```