

Pi MPI Project Report

For the Pi MPI project I used the send and receive method of communication between processors. To solve the integral for Pi using rectangles and the mpi library I began by initializing the MPI environment and set the amount of the processors and the process id's. Then stored the start time to be able to compare the output of the program to that of the Pi serial program.

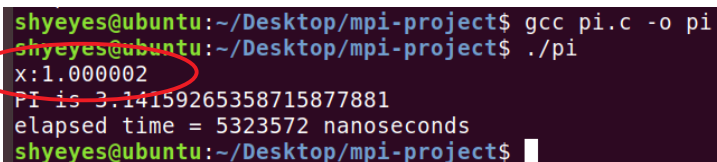
To solve for Pi I had to calculate a few components. The first one was the base length for the rectangles, range of integral divided by the NUMSTEPS. Then get the amount of NUMSTEPS assigned to every processor, NUMSTEPS divided by the amount of the processors, stored in variable epg. Next, every processor will return the area calculated, stored in local_sum, within its range using the rect() function.

Most of the work is done within the function rect(). This function takes three parameters: the amount of steps per processor (epg), the base length, process id. To calculate the area of a rectangle we need to multiply the length x width. Here the base length for the rectangle was already calculated, this value will be the same for all the steps taken. The missing value to find the area is the x and y value. The y value is found by using the function $4/(1+x^2)$ and the x value is the left x value of the rectangle in question. To find the left x value, I used an if/else statement to set the starting range of the root process and the rest of the processes. The starting value of the range for the root process is always 0 and the starting range for the other processes is found by multiplying the steps per process by the base length and the process id. This is done so that every process can work within its assigned area. Then, a for loop would iterate through the steps assigned to each process. For every iteration, the x value would be incremented by the base length this x value would be plugged into the function $4/(1+x^2)$ to calculate the y-value. After the for loop, the base and the y value were multiplied to find area. The base area is used once because it is the common number in every area calculation. The function rect() returns the area.

The next step is to have every process - except the root process - send its area calculated to the root process using the function MPI_Send. The root process adds its area calculated to the pi variable as well as receive all the areas sent by the other processors and adds them to pi.

Finally, the root process will get the end time, calculate the elapsed time, print the result of pi and elapsed time. Once all computation is completed the MPI environment will be finalized.

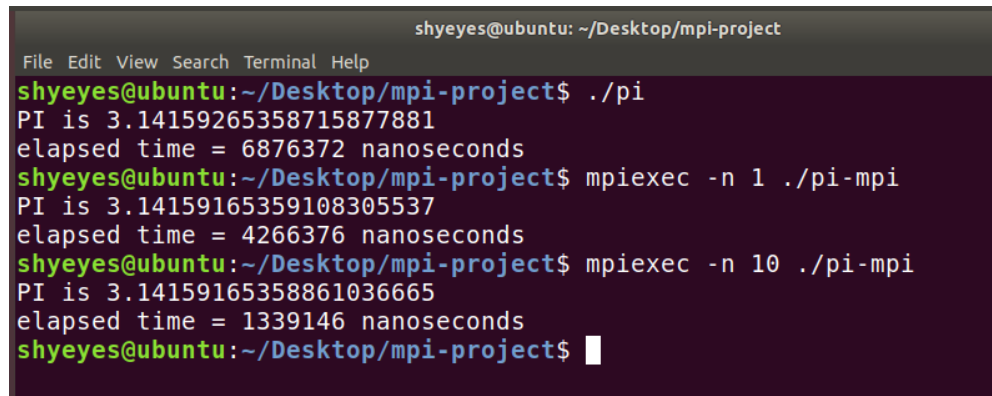
The results of the pi-mpi program were similar to the serial program but not the same. I believe the result values are different because while running the serial program, I printed the final x value and noticed the x value to be 1.000002 instead of 1, similar to that of the range in the original integral, as the screenshot below shows.



```
shyeyes@ubuntu:~/Desktop/mpi-project$ gcc pi.c -o pi
shyeyes@ubuntu:~/Desktop/mpi-project$ ./pi
x:1.000002
Pi is 3.14159265358715877881
elapsed time = 5323572 nanoseconds
shyeyes@ubuntu:~/Desktop/mpi-project$
```

The comparison in time is also different. The serial program takes a lot more time to compute than the mpi program running with 1 processor. I also ran the program and with 10 processors and the time comparison was very significant, with 10 processors completing the problem faster.

Output Screenshot:

A terminal window titled 'shyeyes@ubuntu: ~/Desktop/mpi-project' showing the execution of a program. The user runs './pi' and receives the output 'PI is 3.14159265358715877881' and 'elapsed time = 6876372 nanoseconds'. Then, the user runs 'mpiexec -n 1 ./pi-mpi' and receives 'PI is 3.14159165359108305537' and 'elapsed time = 4266376 nanoseconds'. Finally, the user runs 'mpiexec -n 10 ./pi-mpi' and receives 'PI is 3.14159165358861036665' and 'elapsed time = 1339146 nanoseconds'. The terminal has a dark background with green and blue text for prompts and standard white text for output.

```
shyeyes@ubuntu: ~/Desktop/mpi-project
File Edit View Search Terminal Help
shyeyes@ubuntu:~/Desktop/mpi-project$ ./pi
PI is 3.14159265358715877881
elapsed time = 6876372 nanoseconds
shyeyes@ubuntu:~/Desktop/mpi-project$ mpiexec -n 1 ./pi-mpi
PI is 3.14159165359108305537
elapsed time = 4266376 nanoseconds
shyeyes@ubuntu:~/Desktop/mpi-project$ mpiexec -n 10 ./pi-mpi
PI is 3.14159165358861036665
elapsed time = 1339146 nanoseconds
shyeyes@ubuntu:~/Desktop/mpi-project$
```