Assignment #5 - Due Nov 17

Start Assignment

Due Wednesday by 11:59pm **Points** 100
Submitting a text entry box, a website url, or a file upload

Exercise #1

Run the MPI program below, which calculates π-number by integrating f(x) = 4 /(1+x^2). To do this calculation the Area under the curve of a functions is divided into rectangles, and the rectangles are distributed to the processors.

Each process node: - Receives the number of rectangles used in the approximation. - Calculates the areas of it's rectangles. - Synchronizes for a global summation. - if you are Node 0 then prints the result.

Your tasks are to run the program below, an example as follows:

-bash-4.2$ mpicc picalc.c

-bash-4.2$ mpirun -np 140 ./a.out

Process 0 of 140 on o0752.ten.osc.edu

pi is approximately 3.1415926544231270, Error is 0.0000000008333338
wall clock time = 0.013971

Process 16 of 140 on o0752.ten.osc.edu
Process 135 of 140 on o0756.ten.osc.edu .....

Exercise #2: Modify the program to compute pi with a number of intervals ranging over powers of 10, say from 10^1 to 10^10 intervals. From the output produce a pair of tables or plots with the data indicating the changing accuracy of the approximations, and the changing wall clock time. You are free to experiment with any potential improvements to the code. For example, one is suggested in the program comments.

```
/*  example from MPICH  */
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
 double f(double);

double f(double a)
{
    return (4.0 / (1.0 + a*a));
```

```
}

int main(int argc,char *argv[])
{
    int done = 0, n, myid, numprocs, i;
    double PI25DT = 3.14159265358979323846264643;
    double mypi, pi, h, sum, x;
    double startwtime = 0.0, endwtime;
    int  namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Get_processor_name(processor_name,&namelen);

    fprintf(stdout,"Process %d of %d on %s\n",
            myid, numprocs, processor_name);

    n = 0;
    while (!done)
    {
        if (myid == 0)
        {
/*
            printf("Enter the number of intervals: (0 quits) ");
            scanf("%d",&n);
*/
            if (n==0) n=10000; else n=0;

            startwtime = MPI_Wtime();
        }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
        if (n == 0)
            done = 1;
        else
        {
            h   = 1.0 / (double) n;
            sum = 0.0;
            /* A slightly better approach starts from large i and works back */
            for (i = myid + 1; i <= n; i += numprocs)
            {
                x = h * ((double)i - 0.5);
                sum += f(x);
            }
            mypi = h * sum;

            MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

            if (myid == 0)
            {
                printf("pi is approximately %.16f, Error is %.16f\n",
                       pi, fabs(pi - PI25DT));
                endwtime = MPI_Wtime();
                printf("wall clock time = %f\n", endwtime-startwtime);
                fflush( stdout );
            }
        }
    }
    MPI_Finalize();
    return 0;
}
```