

MCS 572: Homework 1
University of Illinois at Chicago (Professor Nicholls)
Fall 2009

Due Friday, September 18 by 2pm.

1. Get the code `hellow.c` to work on the parallel computer of your choice. Print the output for $N = 4, 8$ processes.
2. Modify the code `cpi.c` to compute the number $e \approx 2.71828182845905$. Answer the following questions:
 - (a) What is the smallest value of n required to obtain accuracy of 10^{-2} ? 10^{-4} ? 10^{-10} ?
 - (b) What “speedup” do you get for $N = 4$ processes? $N = 8$ processes?
3. Exercises (Heath): 1.2, 1.6, 1.10, 1.22.
4. Computer Problems (Heath): 1.1, 1.4, 1.7, 1.14.

```
/* -*- Mode: C; c-basic-offset:4 ; -*- */
/*
 * (C) 2001 by Argonne National Laboratory.
 * See COPYRIGHT in top-level directory.
 */

#include <stdio.h>
#include "mpi.h"

int main( int argc, char *argv[] )
{
    int rank;
    int size;

    MPI_Init( 0, 0 );
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf( "Hello world from process %d of %d\n", rank, size );
    MPI_Finalize();
    return 0;
}
```

```

/* -*- Mode: C; c-basic-offset:4 ; -*- */
/*
 * (C) 2001 by Argonne National Laboratory.
 * See COPYRIGHT in top-level directory.
 */

#include "mpi.h"
#include <stdio.h>
#include <math.h>

double f(double);

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

int main(int argc, char *argv[])
{
    int    n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;
    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 is on %s\n",
            myid, numprocs, processor_name);
    fflush(stdout);

    n = 10000; /* default # of rectangles */
    if (myid == 0)
        startwtime = MPI_Wtime();

    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

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

MPI_Finalize();
return 0;
}
```