Verena Horak
2nd Homework
HPSC 2004

Exercise: MPI-example: Approximation of π

Advance Note: The assignment says that π can be determined by

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{4}{1+x^2} dx$$

Proofing this analytically leads to

$$\begin{aligned}
\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{4}{1+x^2} dx &= 4 \cdot \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{1}{1+x^2} dx \\
&= 4 \cdot \arctan x \Big|_{-\frac{1}{2}}^{\frac{1}{2}} \\
&= 4 \left(\arctan \frac{1}{2} - \arctan \frac{-1}{2} \right) \\
&= 4 \left(\arctan \frac{1}{2} + \arctan \frac{1}{2} \right) \\
&= 8 \arctan \frac{1}{2} \\
&= 8 \cdot 0.463648 = 3.70918
\end{aligned}$$

Correcting the boundaries of the integral so that it really equals π looks like this:

$$\begin{aligned}
\int_{0}^{1} \frac{4}{1+x^2} dx &= 4 \cdot \int_{0}^{1} \frac{1}{1+x^2} dx \\
&= 4 \cdot \arctan x |_{0}^{1} \\
&= 4 \left(\arctan 1 - \arctan 0 \right) \\
&= 4 \left(\frac{\pi}{4} + 0 \right) \\
&= \pi
\end{aligned}$$

Problem Description: We will approximate the value of π by numerically evaluating the integral we've just evaluated analytically.
We will subdivide the interval $[0, 1]$ into n subintervals. Thus, we have $n + 1$ points $x_0 = 0, x_1, x_2, \ldots, x_n$ in $[0, 1]$. A numerical approximation of the integral

is then given by

$$\int_0^1 \frac{4}{1+x^2}dx \approx \sum_{i=1}^{n} h \cdot \frac{4}{1+\left(\frac{x_1-x_{i-1}}{2}\right)^2} = h \cdot \sum_{i=1}^{n} \frac{4}{1+\left(\frac{x_i-x_{i-1}}{2}\right)^2}$$

The wok to be done can be subdivided on K processors, for example if processor j only calculates the value for every j^{th} interval. The partial result calculated by processor j is thus

$$h \cdot \sum_{k=1}^{\left[\frac{n}{K}\right]} \frac{4}{1+\left(\frac{\lfloor}{x}_{jk} - x_{jk-1}2\right]^2}$$

The final result can be obtained by adding up all partial results from the different processors.
An implementation of this algorithm may look like this:

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

int main(argc,argv) int argc; char *argv[]; {
    int done = 0, n, myid, numprocs, i;
    double PI25DT = 3.14159265358979323846264 3;
    double mypi, pi, h, sum, x;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
```

This part only includes the necessary mpi-files, declares and initializes some variables like π written with 25 digits. After that the MPI-size and -rank are determined, size represents the number of processors and rank the number of one special processor.

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

This if-statement asks if the current processor is root that means its number is set 0. The root-processor asks the user for a number greater or equal 0. 0 will quit the process, any other positiv number sets the number of subintervals.

```
    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    if (n == 0) break;
```

Root broadcasts the number of intervals n.

```
h   = 1.0 / (double) n;
sum = 0.0;
```

h represents the step size according to n

```
for (i = myid + 1; i <= n; i += numprocs) {
    x = h * ((double)i - 0.5);
    sum += 4.0 / (1.0 + x*x);
}
mypi = h * sum;
```

In this part the value of π on the given subinterval should be calculated. At first sight it seems to be something belonging to the above given incorrect integral but it isn't. At least the integral

$$\int_{-\frac{h}{2}}^{1-\frac{h}{2}} \frac{4}{1+x^2} dx$$

is split up and summed up. For $n \to \infty$ - meaning using a pretty high number for n - $h \to 0$. So the integral will give a better approximation for π when increasing the number of n as the boundaries are getting closer to 0 and 1 - the exact boundaries for calculating π analytically.

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

Now all partial solutions for π are reduced to a single value. In our case they are summed up as MPI_SUM is used. The result is evaluated by the root processor (which has $id = 0$).

```
if (myid == 0)
    printf("pi is approximately %.16f, Error is %.16f\n",
       pi, fabs(pi - PI25DT));
}
MPI_Finalize();
return 0;
}
```

Root prints the calculated result and the error, the MPI-environment is closed and 0 is returned signing the programm ended correctly.
Of course, it is quite unusual to determine the quality of an approximate value by comparing it to the exact result. If this is available, there is no need for approximation, actually. Anyway, this example is supposed to illustrate the usage and syntax of MPI rather then assessing the practical use of numerical algorithms.