**CS 594**
**Understanding Parallel Architectures:**
**From Theory To Practice**
**Spring 2001**
*Assignment 1*

This work must be handed in by 4:30pm on **January 26, 2000**.

---

The sequential program shown in Fig. 1 finds the approximate value of the integral

$$\int_0^{\pi/2} \frac{dx}{\sqrt{1-(\cos^2 x)/2}}$$

The program divides the interval 0 to $\pi/2$ into 128 equally-sized subintervals of width $\pi/256$. The area under the curve for each subinterval is then approximated by that of a trapezoid of width $\pi/256$. The lengths of the parallel sides of the trapezoid are given by the value of the integrand at the start and end of the subinterval.

1. Write an MPI program in which each process is assigned approximately the same number of contiguous subintervals – you may assume that the number of sub-intervals (128) is exactly divisible by the number of processes. For example, for the 4-process case, one processor handles the first 32 subintervals, the second processor the next 32 subintervals, and so on. In your program each process should evaluate its contribution to the integral and output this value in a line that says something like

   ```
   Process number 3: contribution = 0.397765
   ```

   Run the program for 4 and 8 processes.

2. Modify your program to sum the contributions to give the total value of the integral. You should perform the summation using the collective communication routine `MPI_Reduce`, and output the result from process number 0. Run the program for 4 and 8 processes.

3. Assuming that the time to find the rectangular area for a single subinterval is $At_{\text{calc}}$, and the time to perform the sum of the processor contributions is $Bt_{\text{comm}} \log N$, where $A$ and $B$ are constants and $N$ is the number of processes, give an expression for the efficiency of the parallel algorithm for $m$ subintervals. You may assume $m$ is exactly divisible by the number of processors.

4. Comment on the scalability of the algorithm.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main(){
  double deltax;
  double pi = 3.141592654;
  double integral;
  double fleftx, frightx, rightx;
  double f();
  int    nintervals = 128;
  int    i;

  deltax = 0.5*pi/(float)nintervals;
  integral = 0.0;
  fleftx = f(0.0);
  for(i=1;i<=nintervals;i++){
     rightx = deltax*(float)i;
     frightx = f(rightx);
     integral += deltax*0.5*(fleftx+frightx);
     fleftx = frightx;
  }

  printf ("\nIntegral = %f\n",integral);
  exit(0);
}

double f(x)
double x;
{
   double y;

   y = cos(x);
   return (1.0/sqrt(1.0-0.5*y*y));
}
```

Figure 1: Sequential code for integration.