HIGH PERFORMANCE COMPUTING CENTER

HPCC

TEXAS TECH UNIVERSITY

Quadrature

- Quadrature is a numerical technique to produce approximations to definite integrals
- The mid-point rule is one of the simplest quadrature techniques

$$\int_a^b f(x)dx \approx (b-a)f([b+a]/2)$$

Composite midpoint rule

$$\int_a^b f(x)dx \approx h\sum_{i=1}^{n} f(a+[i-.5]h)$$

where $h = (b - a)/n$

Computing p

$$\int_{0}^{1} 4/(1+x\text{^}2)\,dx = \pi$$

Code

```
#include <stdio.h>
#include <time.h>
#include <mpi.h>
#define FALSE 0
#define TRUE  1
#define MASTER_RANK 0
double f(a)
double a;
{
   return (4.0 / (1.0 + a*a));
}
```

```
int main ( int argc, char **argv )
{
  int n, i, pool_size, my_rank, i_am_the_master = FALSE;
  time_t  t0, t1, t2;
  clock_t c0, c1, c2;
  long count;
  double  b,c;
  double mypi, pi, h, sum, x, a;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &pool_size);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
  n=100000;
 t0 = time(NULL);
 c0 = clock();
  if (my_rank == MASTER_RANK) i_am_the_master = TRUE;
  if (i_am_the_master) {
    if (n==0) n=100;
  }

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

  h   = 1.0 / (double) n;
  sum = 0.0;
  for (i = my_rank + 1; i <= n; i += pool_size) {
    x = h * ((double)i - 0.5);
    sum += f(x);
  }
  mypi = h * sum;
  MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, MASTER_RANK,
          MPI_COMM_WORLD);

  if (i_am_the_master) {
                  printf("\npi is approximately %.16f\n", pi);
            t1 = time(NULL);
            c1 = clock();
           printf  ("\t Current Value of n is :%d\n", n);
            printf ("\tTotal elapsed wall clock time: %ld\n", (long) (t1 - t0));
            printf ("\tTotal elapsed CPU time:       %f\n", (float) (c1 -       c0)/CLOCKS_PER_SEC);
            printf ("--------------------------------------------------------------------------------\n");
                 }

  MPI_Finalize ();
 t2 = time(NULL);
 c2 = clock();
 printf ("\telapsed CPU time:      %f on processor:%d\n ", (float) (c2 - c0)/CLOCKS_PER_SEC , my_rank);
}
```

Spend more time in in f

```
double f(a)
double a;
{
   int count=10000;
   double aa,sh,ni;
   while (count > 0)
   {
     ni = sqrt(count);
     sh = 1.0/ni;
     aa = sh - ni;
     count=count -1;
   }
   return (4.0 / (1.0 + a*a));
}
```

Speed up

Procs	Time	Speedup	Efficiency
1	117	1	1
2	58	2	1
4	31	3.7	.94
8	15	7.8	.97
16	10	11.7	.73
32	5	23.4	.73

HIGH PERFORMANCE COMPUTING CENTER

HPCC

TEXAS TECH UNIVERSITY