

Arpit Kumar

1RV17CS024

7TH CSE A1, PADP

PROGRAM 8

CODE:

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

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

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

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

```
# include <time.h>
```

```
int main ( int argc, char *argv[] );
```

```
double f ( double x );
```

```
void timestamp ( );
```

```
int main ( int argc, char *argv[] )
```

```
{
```

```
    double a;
```

```
    double b;
```

```
    double error;
```

```
    double exact;
```

```
    int i;
```

```
    int master = 0;
```

```
    double my_a;
```

```
    double my_b;
```

```
    int my_id;
```

```
    int my_n;
```

```
    double my_total;
```

```
    int n;
```

```
    int p;
```

```

int p_num;

int source;

MPI_Status status;

int tag;

int target;

double total;

double wtime;

double x;


a = 0.0;
b = 10.0;
n = 10000000;
exact = 0.49936338107645674464;

/* Initialize MPI.*/

MPI_Init ( &argc, &argv );

/* Get this processor's ID.*/

MPI_Comm_rank ( MPI_COMM_WORLD, &my_id );

/* Get the number of processes.*/

MPI_Comm_size ( MPI_COMM_WORLD, &p_num );


if ( my_id == master )
{
/* We want N to be the total number of evaluations.

If necessary, we adjust N to be divisible by the number of processes.*/

my_n = n / ( p_num - 1 );
n = ( p_num - 1 ) * my_n;


wtime = MPI_Wtime ( );


//timestamp ( );//Prints the current time

//printf ( "\n" );

printf ( "QUAD_MPI - C/MPI version\n" );

```

```

printf ( " Estimate an integral of f(x) from A to B.\n" );
printf ( " f(x) = 50 / (pi * ( 2500 * x * x + 1 ) )\n" );
printf ( "\n" );
printf ( " A = %f\n", a );
printf ( " B = %f\n", b );
printf ( " N = %d\n", n );
printf ( " EXACT = %24.16f\n", exact );
//printf ( "\n" );
printf ( " Use MPI to divide the computation among\n" );
printf ( " multiple processes.\n" );
}
source = master;
MPI_Bcast ( &my_n, 1, MPI_INT, source, MPI_COMM_WORLD );
/* Process 0 assigns each process a subinterval of [A,B].*/
if ( my_id == master )
{
for ( p = 1; p <= p_num - 1; p++ )
{
my_a = ( ( double ) ( p_num - p ) * a
+ ( double ) ( p - 1 ) * b )
/ ( double ) ( p_num - 1 );

target = p;
tag = 1;
MPI_Send ( &my_a, 1, MPI_DOUBLE, target, tag, MPI_COMM_WORLD );

my_b = ( ( double ) ( p_num - p - 1 ) * a
+ ( double ) ( p ) * b )
/ ( double ) ( p_num - 1 );

target = p;
tag = 2;
MPI_Send ( &my_b, 1, MPI_DOUBLE, target, tag, MPI_COMM_WORLD );

```

```

    }
total = 0.0;
    my_total = 0.0;
}

/*Processes receive MY_A, MY_B, and compute their part of the integral.*/
else
{
    source = master;

    tag = 1;
    MPI_Recv ( &my_a, 1, MPI_DOUBLE, source, tag, MPI_COMM_WORLD, &status );

    source = master;

    tag = 2;
    MPI_Recv ( &my_b, 1, MPI_DOUBLE, source, tag, MPI_COMM_WORLD, &status );

    my_total = 0.0;
    for ( i = 1; i <= my_n; i++ )
    {
        x = ( ( double ) ( my_n - i ) * my_a
            + ( double ) ( i - 1 ) * my_b )
            / ( double ) ( my_n - 1 );
        my_total = my_total + f ( x );
    }

    my_total = ( my_b - my_a ) * my_total / ( double ) ( my_n );

    printf ( " Process %d contributed MY_TOTAL = %f\n", my_id, my_total );
}

/*Each process sends its value to the master process.*/
MPI_Reduce ( &my_total, &total, 1, MPI_DOUBLE, MPI_SUM, master, MPI_COMM_WORLD );

/* Compute the weighted estimate.*/
if ( my_id == master )

```

```

{
    error = fabs ( total - exact );
    wtime = MPI_Wtime ( ) - wtime;

    printf ( "\n" );
    printf ( " Estimate = %24.16f\n", total );
    printf ( " Error = %e\n", error );
    printf ( " Time = %f\n\n", wtime );
}
/*
    Terminate MPI.
*/
MPI_Finalize ( );
/*
    Terminate.
*/
if ( my_id == master )
{
    printf ( "\n" );
    printf ( "QUAD_MPI:" );
    printf ( " Normal end of execution.\n" );
    //printf ( "\n" );
    //timestamp ( );//Prints the current time
}

return 0;
}

double f ( double x )
{
    double pi;
    double value;

```

```
pi = 3.141592653589793;
value = 50.0 / ( pi * ( 2500.0 * x * x + 1.0 ) );
```

```
return value;
}
```

```
void timestamp ( void )
```

```
{
# define TIME_SIZE 40

static char time_buffer[TIME_SIZE];
const struct tm *tm;
time_t now;

now = time ( NULL );
tm = localtime ( &now );

strftime ( time_buffer, TIME_SIZE, "%d %B %Y %I:%M:%S %p", tm );

printf ( "%s\n", time_buffer );

return;
# undef TIME_SIZE
}
```

OUTPUT:

```
QUAD_MPI - C/MPI version
  Estimate an integral of f(x) from A to B.
  f(x) = 50 / (pi * ( 2500 * x * x + 1 ) )

A = 0.000000
B = 10.000000
N = 10000000
EXACT =      0.4993633810764567
Use MPI to divide the computation among
multiple processes.
Process 1 contributed MY_TOTAL = 0.498735
Process 2 contributed MY_TOTAL = 0.000637

Estimate =      0.4993712392373716
Error = 7.858161e-06
Time = 0.078624
```

