

Parallel & Distributed Computing

CSE525

Assignment **#4** - to be submitted to **Dr. Masroor Hussain**

**Numerical Integration of vectors with MPI using Trapezoidal Rule**

Submitted by,

**Quswar Mahmood Abid, CS2003**

Report: MPI based Trapezoidal Rule

# As discussed in the class please solve the trapezoidal rule using MPI.

In this assignment, we are required to implement a numerical integral method of trapezoidal rule. Trapezoidal rule uses following formulation to calculate numerical integration for two subsequent inputs.

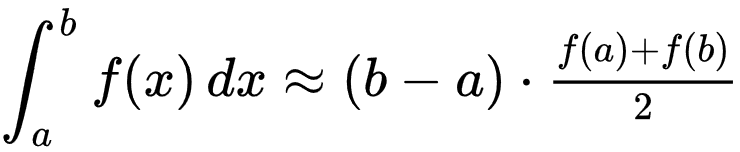


Figure 1. Source: Wikipedia

For an input vector, following calculation must be made.

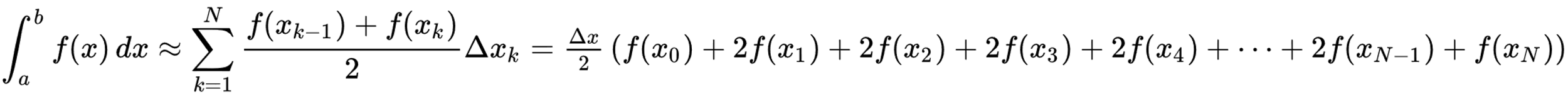


Figure 2. Source: Wikipedia

This makes our implementation very easy because easily availability of its individual components. We can make some editions in how we calculate this by changing how we implement it in code. For example, instead of multiplying each sample by 2, we will 2 out of SOP brackets and subtracting start and end elements. Look at following code to see how it will work.

#include <stdio.h>

#include "mpi.h"

#define SIZE 16

#define P 4

int i;

int main(int argc, char\* argv[]) {

int numtasks, rank, sendcount, recvcount, source;

float sendbuf\_1[SIZE] = {

1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0 };

float recvbuf\_1[SIZE/P];

float result, result\_, dZ;

result = 0;

result\_ = 0;

dZ = 1;

MPI\_Init(&argc,&argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numtasks);

if (numtasks == P) {

source = 0;

sendcount = SIZE/P;

recvcount = SIZE/P;

MPI\_Scatter(sendbuf\_1, sendcount, MPI\_FLOAT, recvbuf\_1, recvcount,

MPI\_FLOAT, source, MPI\_COMM\_WORLD);

/\*Trapezoidal Part\*/

**for (i=0; i<SIZE/P; i++){**

**printf("%f from rank:%d\n", recvbuf\_1[i], rank);**

**//result = result + ( recvbuf\_1[i] \* 2);**

**result = result + recvbuf\_1[i];**

**}**

**result = result \* 2;**

**if (rank == 0){**

**result = result - recvbuf\_1[0];**

**}**

**if (rank == P-1){**

**result = result - recvbuf\_1[(SIZE/P)-1];**

**}**

//printf("Result = %f, from process# %d\n ", result, rank);

MPI\_Reduce(&result, &result\_ , 1, MPI\_FLOAT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

**result\_ = dZ \* result\_ / 2;**

//printf("Result = %f, from process# %d\n ", result, rank);

if (rank == 0)

printf("\nResult = %f \n\n", result\_);

} else {

printf("Must specify %d processors. Terminating.\n",SIZE);

}

MPI\_Finalize();

return 0;

}

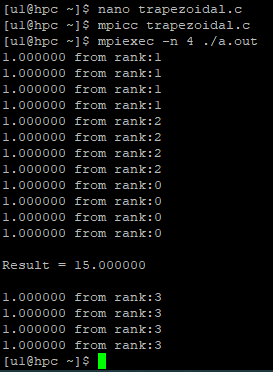


Figure 3. input vector containing 1 at 16 points, giving 100% correct output

Well, the above code is my initial code, but during final exam, I implemented it with a better way.

Following snapshot show the time taken by serial version of trapezoidal rule to integrate to run:

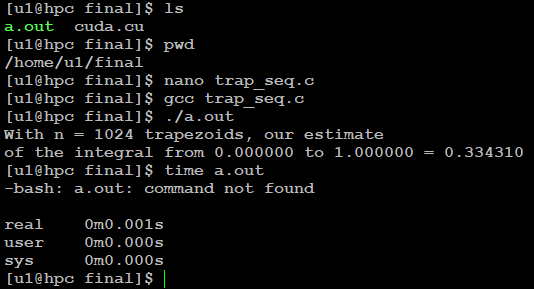


Figure 4. run time of sequential code, also notice the result of integration

Following snapshot shot shows that same time is spent in parallel implementation:

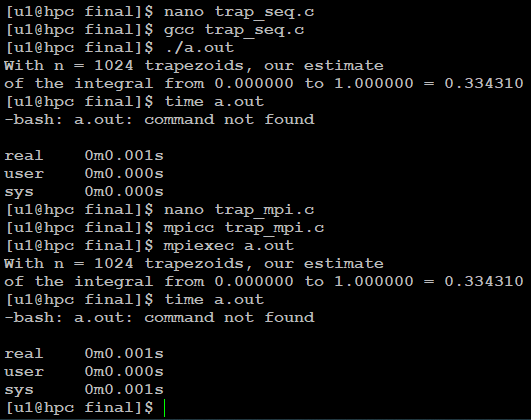


Figure 5. some parallel implementations do not show good speedups at small programs

Note: I took help from online code posted [at this link](http://homepages.math.uic.edu/~hanson/MPI_Reduce_PPMPI.c).

Following is a relevant code snippet for parallelization:

MPI\_Init(&argc,&argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numtasks);

{

source = 0;

MPI\_Bcast(&a, 1, MPI\_DOUBLE, source, MPI\_COMM\_WORLD);

MPI\_Bcast(&b, 1, MPI\_DOUBLE, source, MPI\_COMM\_WORLD);

MPI\_Bcast(&n, 1, MPI\_INT, source, MPI\_COMM\_WORLD);

double h\_local = (b-a)/n; /\* h is the same for all processes \*/

int n\_local = n/numtasks; /\* So is the number of trapezoids \*/

double a\_local = a + rank\*n\_local\*h\_local;

double b\_local = a\_local + n\_local\*h\_local;

double integral = Trap(a\_local, b\_local, n\_local, h\_local);

/\* Add up the integrals calculated by each process \*/

MPI\_Reduce(&integral, &total, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

# FULL CODE:

//Help taken from this code: http://homepages.math.uic.edu/~hanson/MPI\_Reduce\_PPMPI.c, and my assignment code of MPI

//Completed at 07:36PM

#include <stdio.h>

#include "mpi.h"

int i;

int main(int argc, char \*\* argv) {

int numtasks, rank, sendcount, recvcount, source;

double a = 0.0; /\* Left endpoint \*/

double b = 1.0; /\* Right endpoint \*/

int n = 1024; /\* Number of trapezoids \*/

double h; /\* Trapezoid base length \*/

double total; /\* Total integral \*/

double Trap(double local\_a, double local\_b, int local\_n, double h);

//h = (b-a)/n; /\* base of trapezoids \*/

//Initialize MPI

MPI\_Init(&argc,&argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numtasks);

{

source = 0;

MPI\_Bcast(&a, 1, MPI\_DOUBLE, source, MPI\_COMM\_WORLD);

MPI\_Bcast(&b, 1, MPI\_DOUBLE, source, MPI\_COMM\_WORLD);

MPI\_Bcast(&n, 1, MPI\_INT, source, MPI\_COMM\_WORLD);

double h\_local = (b-a)/n; /\* h is the same for all processes \*/

int n\_local = n/numtasks; /\* So is the number of trapezoids \*/

double a\_local = a + rank\*n\_local\*h\_local;

double b\_local = a\_local + n\_local\*h\_local;

double integral = Trap(a\_local, b\_local, n\_local, h\_local);

/\* Add up the integrals calculated by each process \*/

MPI\_Reduce(&integral, &total, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

/\* Print the result \*/

if (rank == 0) {

printf("With n = %d trapezoids, our estimate\n", n);

printf("of the integral from %f to %f = %f\n", a, b, total);

}

}/\*

else{

printf("Must specify %d processors. Terminating.\n",P);

}\*/

MPI\_Finalize();

//printf("With n = %d trapezoids, our estimate\n",n);

//printf("of the integral from %f to %f = %f\n",a,b,total);

return 0;

}

double Trap(double local\_a, double local\_b, int local\_n, double h) {

double integral;

double x;

int i;

double f(double x);

integral = (f(local\_a) + f(local\_b))/2.0;

x = local\_a;

for (i=1; i<=local\_n;i++) {

x = x+h;

integral = integral + f(x);

}

integral = integral\*h;

return integral;

}

double f(double x) {

return x\*x;

}