International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 11 | Nov 2019 www.irjet.net p-ISSN: 2395-0072

Parallelization of Definite Integration

Yash Bhojwani[1], **Rishab Singh**[2]

[1,2]*Student, SCOPE, Vellore Institute of Technology, Vellore, Tamil Nadu, India*

Abstract - *Many areas of science and technology has shown a fast increase in many areas with the help of computational science. Efficient collaboration in interdisciplinary groups performed by computational scientists and engineers is in a growing demand. Courses on numerical analysis and parallel computing requires training such specialists includes. Definite integration is a method to find the area under the curve and to find definite integrals is very important in many fields. Sometimes finding definite integral can take a lot of time as the task is compute intensive and finding accurate and fast numeric result is very important. Hence, the role of parallelization comes in. Using PDC will divide the problem into sub-problems such that every sub-problem is independent of each other, hence increasing the computational speed till a great extent. In this research paper we present Parallel Numerical Methods which bridges the gap between theoretical aspects of numerical methods and issues of implementation for modern multicore and manycore systems. This research paper helps in the process of calculating the time difference between parallel and serial computing.*

***Keyword* — Parallelization; Definite Integrals; Distributed computing; Numerical Analysis; Parallel algorithms; Definite Calculus; Integrals; Integration**

1. INTRODUCTION

Basic difficulties in the field of computer science and engineering majorly includes numerical computation of a definite integral of a function. Therefore, to increase the efficiency of algorithm, substantial efforts have been made to find different methods to exploit the power of advanced computer architectures like multi-core computer. The motivation for parallelizing the integration formula originates from the fact that the evaluation of an integral over an interval can be obtained by dividing the interval into smaller subintervals, and then summing up the individual evaluations of the integral over the subintervals. This is equivalent to the evaluation of the integral over different subintervals on different processors, and later summing the results obtained from all of processors. Evaluation of numerical integrals in engineering problems can take a considerable amount of time.

Theoretically, we found that parallel algorithm can provides solution of numerical integration better and efficiently when the evaluation of function is time consuming. With the availability of parallel technologies and resources, it becomes essential to develop a parallel algorithm for numerical integration and to test the results and performance of algorithms.

In this paper following techniques will be used to calculate numerical integration:

- Gaussian Quadrature
- Simpsons 1/3 rule

2. ALGORITHM

Simpsons Rule:

Using the midpoint rule, we approximated regions of area under curves by taking rectangles. In simple words, we estimated the curve with piecewise constant functions. Using the trapezoidal rule, we estimate the area under the curve using piecewise linear functions. Using Simpson's rule, we estimate = $\int(x)dx$a curve using piecewise linear functions. We divided the interval into an even number of sub intervals, each of them having equal width.

Over the first pair of subintervals we approximate ∫ x2x0f(x)dx x0x2f(x)dx with ∫ x2x0p(x)dx x0x2p(x)dx, where p(x)=Ax2+Bx+Cp(x)=Ax2+Bx+C is the quadratic function passing through (x0,f(x0)),(x1,f(x1)),(x0,f(x0)),(x1,f(x1)) and (x2,f(x2))(x2,f(x2)) (Figure 1).

Over the next pair of subintervals, we approximate x4x2f(x)dx x2x4f(x)dx with the integral of another quadratic function passing through (x2,f(x2)),(x3,f(x3)),(x2,f(x2)),(x3,f(x3)), and (x4,f(x4)).(x4,f(x4)). This process is continued with each upcoming pair of subintervals

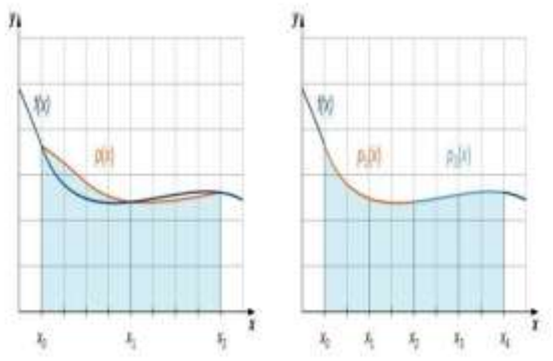


Figure 1 Defining area under curve

To understand the formula that we obtain for Simpson's rule, we begin by deriving a formula for this approximation over the first two subintervals. As we go through the derivation, we need to keep in mind the following relationships:

$f(x0) = p(x0) = Ax20 + Bx0 + C$

$f(x1) = p(x1) = Ax21 + Bx1 + C$

$f(x2) = p(x2) = Ax22 + Bx2 + C$

x2−x0=2Δx, where Δx is the length of a subinterval.

x2+x0=2x1 since x1=(x2+x0)/2

Thus, $x2x0f(x)dx$

$\approx \int x2x0p(x)dx$

$= \int x2x0(Ax2 + Bx + C)\,dx$

$=(A3x3 + B2x2 + Cx) ||| x2x0$

$= A3(x32 - x30) + B2(x22 - x20) + C(x2 - x0)$

$= A3(x2 - x0)(x22 + x2x0 + x20) + B2(x2 - x0)(x2 + x0) + C(x2 - x0)$

$= x2 - x06(2A(x22 + x2x0 + x20) + 3B(x2 + x0) + 6C)$

$=\Delta x3((Ax22 + Bx2 + C) + (Ax20 + Bx0 + C) + A(x22 + 2x2x0 + x20)$

$=\Delta x3(f(x2) + f(x0) + A(x2 + x0)2 + 2B(x2 + x0) + 4C)$

$=\Delta x3(f(x2) + f(x0) + A(2x1)2 + 2B(2x1) + 4C)$

$=\Delta x3(f(x2) + 4f(x1) + f(x0))\ x2x0f(x)dx$

$=\int x2x0p(x)dx$

$=\int x2x0(Ax2 + Bx + C)dx$

$= (A3x3 + B2x2 + Cx) ||| x2x0 = A3(x32 - x30) + B2(x22 - x20) + C$

$= A3(x2 - x0)\ (x22 + x2x0 + x20) + \quad B2(x2 - x0)\ (x2 + x0) + C\ (x2 - x0$

$= x2 - x06(2A\ (x22 + x2x0 + x20) + 3B\ (x2 + x0) + 6C)$

$=\Delta x3((Ax22 + Bx2 + C) + (Ax20 + Bx0 + C) + A\ (x22 + 2x2x0 + x20)$

$=\Delta x3(\mathrm{f}\ (x2) + f\ (x0) + A\ (x2 + x0)2 + 2B\ (x2 + x0) + 4C) = \Delta x3(\mathrm{f}\ (x2)$

$=\Delta x3(\mathrm{f}\ (x2) + 4f\ (x1) + f\ (x0))$

Assume that f(x)f(x) is continuous over

[a,b][a,b]. Let nn be a positive even integer and Δx=b−anΔx=b−an. Let [a,b][a,b] be divided into nn subintervals, each of length ΔxΔx, with endpoints at-

P={x0,x1,x2,...,xn}.P={x0,x1,x2,...,xn}. Set then we will get the following algorithm:

```
1. Given a function f(x):
2.(Get user inputs)
Input
a,b=endpoints of interval n=number of intervals
(Do the integration)
3.Set h= (b-a)/n.
4.Set sum=0.
  5.Start For i= 1 to n -1
  Apply x =a + h*i.
  If i%2=0
  Next Change sum=sum+2*f(x)
  Else
  Set sum=sum+4*f(x) End For
  6. Set sum = sum + f(a)+f(b)
  7. Set ans = sum*(h/3).
  8. End
```

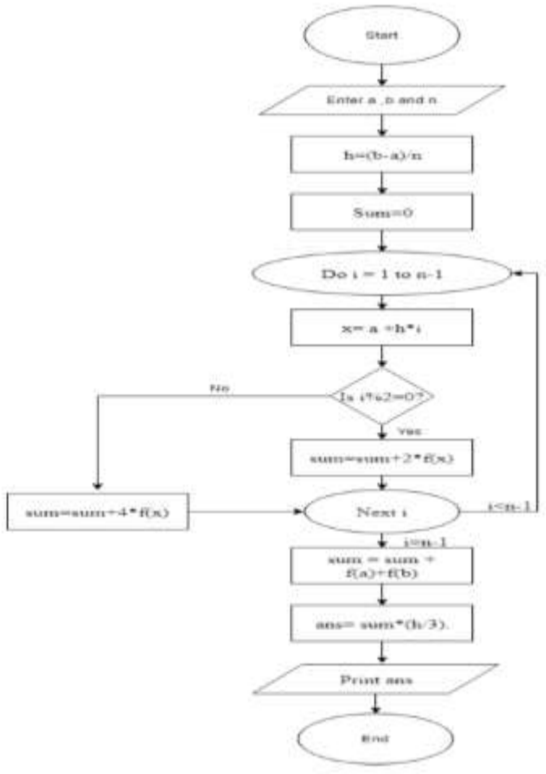


Figure 1 Flow chart for Simpson's rule algorithm

Gaussian Quadrature

Quadrature rule stated as a weighted sum of function values at specified points in a particular domain of integration, is basically an approximation based on the definite integral of a function in this rule a definite integral is first divided into n number of parts and then area is calculated for each part. The sum of area of each part results in approximate value of the definite integral. Greater the number of parts the graph is divided into means more accurate value and vice versa. The approx. value of a definite integral can be calculated using the following formula.

1. Given a function f(x): Formula of quadrature is given by

$$\int_a^b f(x)dx = h/3(f0 + fn + 4 * \sum_{i=1,3,5}^{n-1} fi + 2 * \sum_{i=2,4,6}^{n-2} fi)$$

$$\int_b^a f(x) = \sum_{i=0}^{i=n} (f((n-i-1)*a + i*b)/(n-1))$$

Algorithm:

1. Given a function f(x):

2. (Get user inputs)

Input

a,b=endpoints of interval

n=number of intervals

(Do the integration)

```
3.  Set sum=0.
4. Begin For i= 1 to n -1
Set x = ((n-i-1)*a+(i)*b)/(n-1)
Set sum=sum+f(x)
End For
4. Set sum = (b-a)*sum/n
5. Set ans = sum
6. End
```

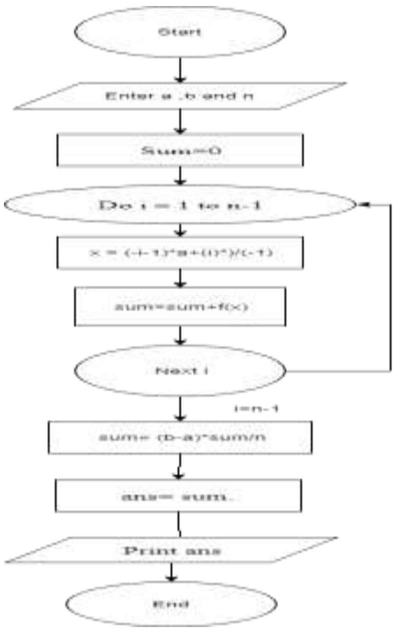


Figure 2. Flowchart for Gaussian Quadrature algorithm

3. APPLICATION OF NUMERICAL INTEGRATION

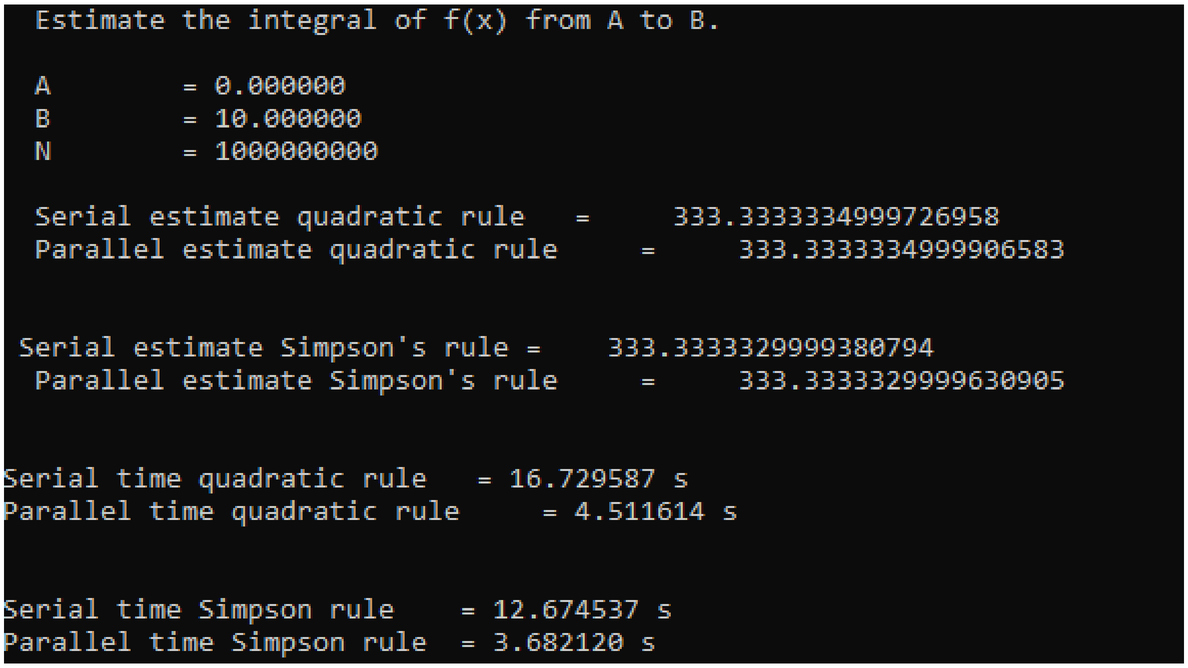
It helps to:

- Find the area.
- Locate the centroid.
- Find the arc length of a graph.
- Calculate the surface area of a solid.
- Calculate the volume of a figure.
- Solve for the work done.
- Solve the moment of inertia.

- It is also used to find Sectional area.
- Waterplane area.
- Submerged volume.
- Longitudinal center of floatation (LCF).
- Vertical center of buoyancy (VCB).

4. RESULT

Given two rules are beneficial to find the value of those definite integrals whose integration is either can't be found or is complicated. Parallel computation is faster when the calculation is big else for small calculations the parallel computation results in more time consuming when compared with serial computing. In our task to find integral we need to divide the graph in many parts to get more accurate value, so parallel computation is beneficial to us. Here for f(x)= x2 a considerable change is seen in both the algorithms. Also, it can be seen that Simpson's rule is faster as compared to Gaussian quadratic rule.



```
  Estimate the integral of f(x) from A to B.

  A        = 0.000000
  B        = 10.000000
  N        = 1000000000

  Serial estimate quadratic rule   =     333.3333334999726958
  Parallel estimate quadratic rule     =     333.3333334999906583


 Serial estimate Simpson's rule =    333.3333329999380794
  Parallel estimate Simpson's rule     =     333.3333329999630905


Serial time quadratic rule   = 16.729587 s
Parallel time quadratic rule     = 4.511614 s


Serial time Simpson rule    = 12.674537 s
Parallel time Simpson rule  = 3.682120 s
```

5. CONCLUSION

The Simpson method with smaller segmentation give better estimates with smaller errors than the Quadrature method. Simpson's rule takes less computation time than quadrature rule. Parallel computing with more iterations (smaller segments) takes less time than serial computation.

6. REFERENCES

[1] M. Concepcion Ausin, 2007, an introduction to quadrature and other numerical integration techniques, Encyclopedia of Statistics in Quality and reliability. Chichester, England.

[2] Gordon K. Smith, 2004, Numerical Integration, Encyclopedia of Biostatistics.2nd edition, Vol-6

[3] Rajesh Kumar Sinha, Rakesh Kumar ,2010, Numerical method for evaluating the integrable function on a finite interval, International Journal of Engineering Science and Technology.Vol-2(6)

[4] Gerry Sozio, 2009, Numerical Integration, Australian Senior Mathematics Journal, Vol-23(1)

[5] J. Oliver, 1971, The evaluation of definite integrals using high-order formulae, The Computer Journal, Vol-14(3)

[6] S.S Sastry, 2007, Introductory Method of Numerical Analysis, Fourth Edition, Prentice hall of India Private Limited.

[7] Richard L. Burden, 2007, Numerical Analysis, Seven Edition, International Thomson Publishing Company.

[8] Jonh H. Mathew, 2000, Numerical Method for Mathematics, Science and Engineering, Second Edition, Prentice Hall of India Private Limited.

[9] Andrews, G. R. (1999). Foundations of Parallel and Distributed Programming. 1st.

[10] Mathews, J. H., & Fink, K. D. (2004). Numerical methods using MATLAB (Vol. 4). Upper Saddle RiveNJ: Pearson Prentice Hall.

[11] Stoer, J., & Bulirsch, R. (2013). Introduction to numerical analysis (Vol. 12). Springer Science & Business Media.

[12] Bastrakov, S., Meyerov, I., Gergel, V., Gonoskov, A., Gorshkov, A., Efimenko, E& Petrov, V.(2013). High performance computing in biomedical applications. Procedia Computer Science, 18, 10-19.

[13] Jeffers, J., & Reinders, J. (2015). High performance parallelism pearls volume two: multicore and many-core programming approaches. Morgan Kaufmann.