I. J. Mathematical Sciences and Computing, **2023, 3, 58-65**
Published Online on August 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijmsc.2023.03.05

Integration based on Monte Carlo Simulation

Priyanshi Mishra
Department of Computer Science Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India
ORCID iD: https://orcid.org/0009-0006-5844-7501

Pramiti Tewari
Department of Computer Science Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India
ORCID iD: https://orcid.org/0009-0006-8045-6957

Dhananjay R. Mishra
Department of Mechanical Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India
ORCID iD: https://orcid.org/0000-0002-5107-0012

Pankaj Dumka*
Department of Mechanical Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India
Email: p.dumka.ipec@gmail.com
ORCID iD: https://orcid.org/0000-0001-5799-6468
*Corresponding Author

Received: 04 March, 2023; Revised: 18 April, 2023; Accepted: 23 May, 2023; Published: 08 August, 2023

Abstract: In this short article an attempt has been made to model Monte Carlo simulation to solve integration problems. The Monte Carlo method employs random sampling and the theory of big numbers to generate values that are very close to the integral's true solution. Python programming has been used to implement the developed algorithm for integration. The developed Python functions are tested with the help of six different integration examples which are difficult to solve analytically. It has been observed that that the Monte Carlo simulation has given results which are in good agreement with the exact analytical results.

Index Terms: Integration, Definite Integral, Monte Carlo Simulation, Python Programming

1. Introduction

Integration which is also called as the area finding method is used in all the branches of engineering and science [1]. In integration the area under the function is evaluated within prescribed limits. For some problems the integration is very easy to implement but there are problems whose integration cannot be performed by simple procedures [2]. For these types of problems often numerical integration (trapezoidal or Simpson's method of integration) are used [3–5]. Apart from numerical methods Monte Carlo [6,7] is also a method which can be used to model the integration. The Monte Carlo process uses the theory of large numbers [8] and random sampling to approximate values [9] that are very close to the actual solution of the integral [10,11].

The probability of various outcomes in a process that cannot be easily anticipated due to the interference of random factors is modelled using a Monte Carlo simulation [12]. It is a method for comprehending the effects of risk and uncertainty. In various disciplines, including finance, business, physics, and engineering, a Monte Carlo simulation is used to solve a variety of problems [13–16]. A simulation with multiple probabilities is another name for it. The Monte Carlo method acknowledges a problem with all simulation techniques: random variable interference makes it difficult to pinpoint the likelihood of different outcomes. An emphasis in a Monte Carlo simulation is therefore placed on repeatedly repeating random samples [17,18].

A random value is given to the unknown variable in a Monte Carlo simulation. After running the model, a conclusion is given. The variable in question is given several distinct values while this process is repeated repeatedly. The results are averaged after the simulation is finished to get an estimate [19,20].

In this article, first a detailed explanation is given on how to implement the Monte Carlo simulation to obtain the value of definite integral for the problems which are difficult to solve analytically. Thereafter, the algorithm is developed which is modelled as a function in Python programming language. The purpose of using Python is its simple syntax and good number of libraries [21]. NumPy and Matplotlib are used for random variables, arrays, and data plotting [22–26]. At last the developed functions are tested against some numerical problems and the simulation results are compared with the exact analytical results.

2. Monte Carlo simulation and its implementation

To implement the Monte Carlo simulation considers the Fig. 1.

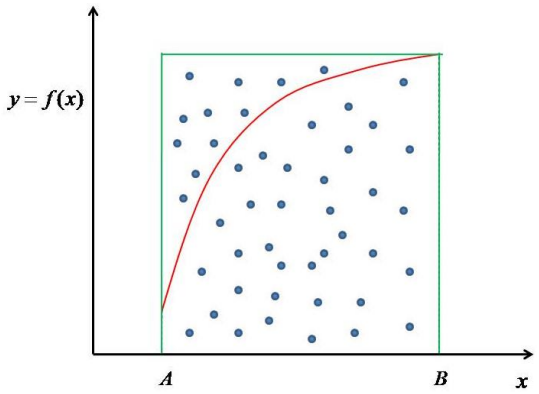

Fig. 1. Functions with random data points

In this the red colour arc is the function which has to be integrated between the limits A and B. The green colour box is the one which encloses the figure. Now according to the Monte Carlo method, first some random x is chosen between AB. Then some random y value is chosen such that its value does not exceed the maximum height of the box. It can be seen in the Fig. 1 that many random (x, y) dots are there. Next all the dots which are below the function curve are of interest as they will form the area below the cure. If N random locations (dots) are there in the rectangular box and n is the number of locations which are below the curve then the probability of dots below the curve will be n/N. Therefore the area below the curve to that of the area of the rectangle can be roughly written as:

$$\frac{A_{below}}{A_{rectangle}} = \frac{n}{N} \tag{1}$$

And as the area under any curve is the integration, therefore the value of integration will be simply given by:

$$I = A_{below} = \frac{n}{N} A_{rectangle} \tag{2}$$

Now the conditions which has to be checked for each random location is that it should lie between the x-axis and the function. Now four conditional checks can be there as shown in the Fig. 2.

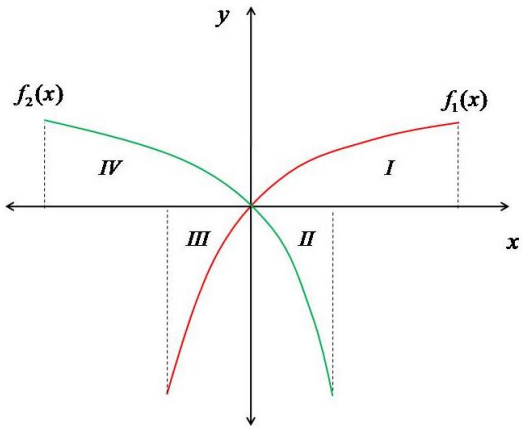

Fig. 2. Different integration regions

As can be seen there are regions I, II, III, and IV. For points to lie in these regions the conditions shown in Table 1. should be satisfied and then only the count is added or subtracted.

Table 1. Conditions for different parts of quadrants

Region	$f(x)$	x	Condition on y	Counter (increase/decrease)
I	$f_1(x) > 0$	$x > 0$	$y < f_1(x)$	$+1$
II	$f_2(x) < 0$	$x > 0$	$y > f_2(x)$	-1
III	$f_1(x) < 0$	$x < 0$	$y > f_1(x)$	-1
IV	$f_2(x) > 0$	$x < 0$	$y < f_2(x)$	$+1$

3. Python Programming

Following function is developed to perform the Monte Carlo simulation:

```
def Monte_carlo(f,dom,N):

    # Evaluating area of rectangular box
    height_rec=max(f(i) for i in linspace(dom[0],dom[1]))
    area_rec=abs((dom[1]-dom[0])*height_rec)


    #Number of data points lying in the expected region
    count_hit=0

    for i in range(N):

        # x random location in the domain
        x=random.uniform(dom[0],dom[1])

        # y random location within the box
        if f(x)<0:
            y=random.uniform(-1,0)*height_rec
        else:
            y=random.uniform(0,1)*height_rec

        # Conditions for the point to be in expected region
        if f(x)>0 and x>0 and y<f(x):
            count_hit=count_hit+1
        elif f(x)<0 and x>0 and y>f(x):
            count_hit=count_hit-1
        elif f(x)>0 and x<0 and y<f(x):
            count_hit =count_hit +1
        elif f(x)<0 and x<0 and y>f(x):
            count_hit=count_hit-1
        else:
            pass

    # Fraction of points in the expected region
    fraction_hit=count_hit/N

    # Area under the function
    area=fraction_hit*area_rec

    return area
```

This function requires the mathematical function to be integrated, its domain, and the number of random data points. Apart from this one function is also written to plot the mathematical function which is to be integrated, the function is as follows:

```
def plt(f,dom,name='plot'):
    figure(1,dpi=300)
    x=linspace(dom[0],dom[1],50)
    plot(x,f(x),'r-')
    plot(dom,[0,0],'b-')
    xlabel('x')
    ylabel('f(x)')
    xlim(dom)
    savefig(f'{name}.jpg')
    show()
```

Let us take some examples to demonstrate the use of the above functions.

4. Example problems

Following are some numerical problems which are solved based on the functions developed in the previous section. The point to be noted here is that the simulation result will not be the same every time it is being run as the method is stochastic in nature. But the result will be very close to the exact result i.e. differing only in decimals. To get more accuracy in results one can go for more number of data points.

Example 1: Solve integral equation shown in Eq. (3).

$$I = \int_0^{\pi} \sin(x)dx \tag{3}$$

Solution: The code is shown in the left side box whereas, the right box shows the function plot and the result of integration. This approach will be followed in all the subsequent examples as well. In this problem $\sin(x)$ is integrated from 0 to π. Total 50,000 random points were taken.

```
# Function to be integrated
def f(x):
    return sin(x)

# Domain of integration
dom=(0,pi)

# Number of random data points
N=50000

# Calling Monte Carlo fn
I=Monte_carlo(f,dom,N)
print(f'I = {round(I,4)}')

# Calling plotting fn
plt(f,dom,'sin(x)')
```

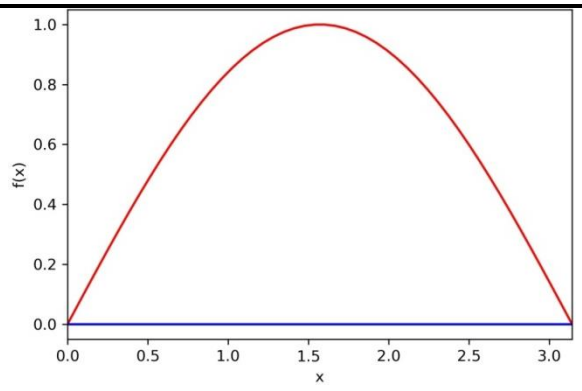

Fig. 3. Function plot

```
Program output: I = 2.01
Exact answer: I = 2.00
```

Example 2: Solve integral equation shown in Eq. (4).

$$I = \int_{-\pi}^{\pi} \cos(x)dx \tag{4}$$

Solution: In this problem $\cos(x)$ is integrated from - π to π. Total 10^5 random points were taken.

```
# Function to be integrated
def f(x):
    return cos(x)

# Domain of integration
dom=(-pi,pi)

# Number of random data points
N=10**5

# Calling Monte Carlo fn
I=Monte_carlo(f,dom,N)
print(f'I = {round(I,4)}')

# Calling plotting fn
plt(f,dom,'cos(x)')
```

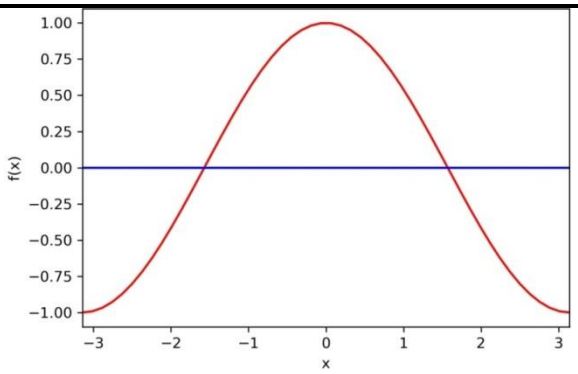

Fig. 4. Function plot

```
Program output: I = 0.0031
  Exact answer: I = 0.00
```

Example 3: Solve integral equation shown in Eq. (5).

$$I = \int_{-5}^{7} |x-1| dx \tag{5}$$

Solution: In this problem $|x-1|$ is integrated from - 5 to 7. Total 10^5 random points were taken.

```
# Function to be integrated
def f(x):
    return abs(x-1)

# Domain of integration
dom=(-5,7)

# Number of random data points
N=10**5

# Calling Monte Carlo fn
I=Monte_carlo(f,dom,N)
print(f'I = {round(I,4)}')

# Calling plotting fn
plt(f,dom,'cos(x)')
```

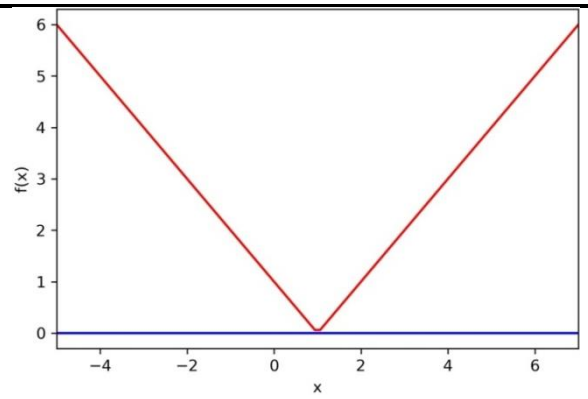

Fig. 5. Function plot

```
Program output: I = 36.1519
  Exact answer: I = 36.00
```

Example 4: Solve integral equation shown in Eq. (6).

$$I = \int_{0}^{1} \frac{x^4(1-x)^4}{1+x^2} dx \tag{6}$$

Solution: In this example $\frac{x^4(1-x)^4}{1+x^2}$ has been integrated from 0 to 1 with 10^5 random data points. The answer is accurate up to three decimal points. But accuracy can further be improved by taking more number of data points.

```
# Function to be integrated
def f(x):
    return x**4*(1-x)**4/(1+x**2)

# Domain of integration
dom=(0,1)

# Number of random data points
N=10**5

# Calling Monte Carlo fn
I=Monte_carlo(f,dom,N)
print(f'I = {round(I,4)}')

# Calling plotting fn
plt(f,dom,'new_fn')
```

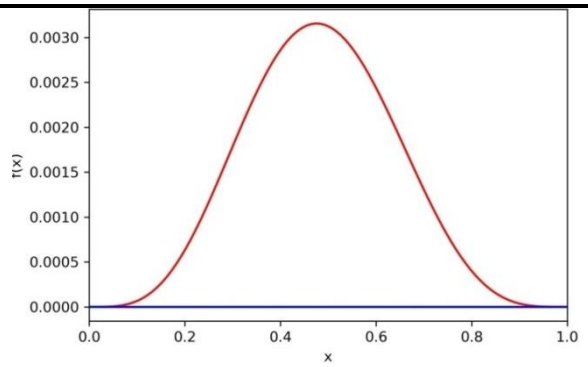

Fig. 6. Function plot

```
Program Output: I = 0.00130
Exact answer: I = 0.00126
```

Example 5: Solve integral equation shown in Eq. (7).

$$I = \int_{0}^{\frac{1}{3}} \frac{dx}{1+e^{x}} \tag{7}$$

Solution: In this problem $\frac{1}{1+e^{x}}$ has been integrated in the limit (0,1/3) with 10^5 random points.

```
# Function to be integrated
def f(x):
    return 1/(exp(x)+1)

# Domain of integration
dom=(0,1/3)

# Number of random data points
N=10**5

# Calling Monte Carlo fn
I=Monte_carlo(f,dom,N)
print(f'I = {round(I,4)}')

# Calling plotting fn
plt(f,dom,'new_fn2')
```

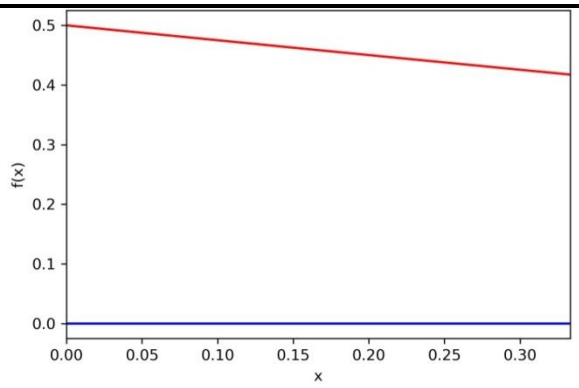

Fig. 7. Function plot

```
Program output: I = 0.1528
Exact answer: I = 0.1528
```

Example 6: Solve integral equation shown in Eq. (8).

$$I = \int_{8.5}^{41} \frac{dx}{\sqrt{2x-1}+\sqrt[4]{2x-1}} \tag{8}$$

Solution: In this problem $\frac{dx}{\sqrt{2x-1}+\sqrt[4]{2x-1}}$ has been integrated between 8.5 to 41 with 10^5 random data points.

```
# Function to be integrated
def f(x):
    return 1/(sqrt(2*x-1)-
(2*x-1)**(1/4))

# Domain of integration
dom=(08.5,41)

# Number of random data points
N=10**5

# Calling Monte Carlo fn
I=Monte_carlo(f,dom,N)
print(f'I = {round(I,4)}')

# Calling plotting fn
plt(f,dom,'new_fn3')
```

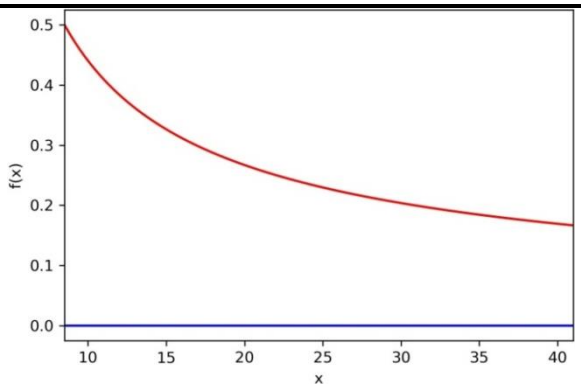

Fig. 8. Function plot

```
Program output: I = 8.3764
Exact answer: I = 8.3862
```

From the above examples it is clear that Monte Carlo function developed to solve the integration problems can handle any level of complexity and hence, is capable to produce results within acceptable limits.

5. Conclusion

In this article the integration has been performed with the help of Monte Carlo simulation which is a stochastic method based on random walks. The simulation methodology is explained and the algorithm thus developed is programmed using Python language. The function Monte_carlo( ) has been tested on six different problems of increasing level of difficulty and the program outputs are in good agreement with the exact analytical results. The function is very robust and can handle engineering problems with single variable without any difficulty. The article will help the practicing engineers to learn the Monte Carlo simulation. Moreover, they will also understand that how to model a stochastic approach in Python programming.

References

[1] M.N. Hounkonnou, J.D.B. Kyemba, R (p, q)-calculus: differentiation and integration, SUT J. Math. 49 (2013) 145–167.
[2] P.W. Thompson, G. Harel, Ideas foundational to calculus learning and their links to students' difficulties, ZDM - Math. Educ. 53 (2021) 507–519. https://doi.org/10.1007/s11858-021-01270-1.
[3] W. Miranker, E. Isaacson, H.B. Keller, Analysis of Numerical Methods, Courier Corporation, 1967. https://doi.org/10.2307/2003280.
[4] J.D. Faires, R.L. Burden, Numerical methods., Thomson, 2003.
[5] P. Dumka, R. Dumka, D.R. Mishra, Numerical Methods Using Python, BlueRose, 2022.
[6] M. Fippel, Basics of monte carlo simulations, in: Monte Carlo Tech. Radiat. Ther., CRC Press, 2016: pp. 17–28. https://doi.org/10.1587/bplus.2008.6_11.
[7] R.M. Feldman, C. Valdez-Flores, R.M. Feldman, C. Valdez-Flores, Basics of Monte Carlo Simulation, Appl. Probab. Stoch. Process. (2010) 45–72.
[8] B. V Gnedenko, Theory of probability, Routledge, 2006. https://doi.org/10.2307/3028701.
[9] P. Dattalo, Strategies to Approximate Random Sampling and Assignment, Oxford University Press, 2010. https://doi.org/10.1093/acprof:oso/9780195378351.001.0001.
[10] R.L. Harrison, Introduction to Monte Carlo simulation, in: AIP Conf. Proc., 2009: pp. 17–21. https://doi.org/10.1063/1.3295638.
[11] P.L. Bonate, A brief introduction to Monte Carlo simulation, Clin. Pharmacokinet. 40 (2001) 15–22. https://doi.org/10.2165/00003088-200140010-00002.
[12] R.L. Harrison, Introduction to Monte Carlo simulation, in: AIP Conf. Proc., 2009: pp. 17–21. https://doi.org/10.1063/1.3295638.
[13] K. Binder, D. Heermann, L. Roelofs, A.J. Mallinckrodt, S. McKay, Monte Carlo Simulation in Statistical Physics, Comput. Phys. 7 (1993) 156. https://doi.org/10.1063/1.4823159.
[14] Y.H. Kwak, L. Ingall, Exploring monte carlo simulation applications for project management, IEEE Eng. Manag. Rev. 37 (2009) 83–83. https://doi.org/10.1109/emr.2009.5235458.
[15] G.A. Bird, Monte-Carlo Simulation in an Engineering Context, Rarefied Gas Dyn. Parts I II. 74 (1981) 239–255. https://doi.org/10.2514/5.9781600865480.0239.0255.
[16] S. Mordechai, Applications of Monte Carlo Method in Science and Engineering, InTech, 2012. https://doi.org/10.5772/1954.
[17] E. Zio, E. Zio, Monte Carlo simulation: The method, Springer, 2013. https://doi.org/10.1007/978-1-4471-4588-2_3.
[18] J.C. Walter, G.T. Barkema, An introduction to Monte Carlo methods, Phys. A Stat. Mech. Its Appl. 418 (2015) 78–87. https://doi.org/10.1016/j.physa.2014.06.014.

[19] P. Brandimarte, Handbook in Monte Carlo Simulation: Applications in Financial Engineering, Risk Management, and Economics, John Wiley \& Sons, 2014. https://doi.org/10.1002/9781118593264.
[20] P. Mishra, A. Sharma, D.R. Mishra, P. Dumka, Solving Double Integration With The Help of Monte Carlo Simulation : A Python Approach, Int. J. Sci. Res. Multidiscip. Stud. 9 (2023) 6–10.
[21] Y.C. Huei, Benefits and introduction to python programming for freshmore students using inexpensive robots, in: Proc. IEEE Int. Conf. Teaching, Assess. Learn. Eng. Learn. Futur. Now, TALE 2014, 2015: pp. 12–17. https://doi.org/10.1109/TALE.2014.7062611.
[22] T.A. Review, C.E. Issn, Comparative Study of Wind Pressure Variations on Rectangular Buildings Using Python Programming, 11 (2022) 15–24.
[23] P. Dumka, N. Samaiya, S. Gandhi, D.R. Mishra, Modelling of Hardy Cross Method for Pipe Networks, 10 (2023) 1–8.
[24] P. Dumka, D.R. Mishra, Understanding the TDMA / Thomas algorithm and its Implementation in Python, Int. J. All Res. Educ. Sci. Methods. 10 (2022) 998–1002.
[25] P. Dumka, K. Rana, S. Pratap, S. Tomar, P.S. Pawar, D.R. Mishra, Modelling air standard thermodynamic cycles using python, Adv. Eng. Softw. 172 (2022) 103186. https://doi.org/10.1016/j.advengsoft.2022.103186.
[26] P. Dumka, R. Chauhan, A. Singh, G. Singh, D. Mishra, Implementation of Buckingham ' s Pi theorem using Python, Adv. Eng. Softw. 173 (2022) 103232. https://doi.org/10.1016/j.advengsoft.2022.103232.

Authors' Profiles

Priyanshi Mishra has done her 10th and 12th from Little Angels High School and Sanskar Public School, Gwalior, respectively. Currently she is pursuing B.Tech. in Computer Science and Engineering from Jaypee University of Engineering and Technology, Guna, Madhya Pradesh (India). Her keen interest is in Scientific and Mathematical computations.

Pramiti Tewari has done her schooling from Puranchandra Vidyaniketan, Kanpur. Currently, she is pursuing B.Tech. in Computer Science and Engineering from Jaypee University of Engineering and Technology, Guna, Madhya Pradesh (India). Her interests include Design Technology and Modelling along with a proclivity towards Applied Mathematics and Combinatorics.

Dhananjay R. Mishra has obtained his bachelor's degree (Mechanical Engineering) from Amravati University (M.S.), Master's degree (Production Engineering) in 2007 from BIT, Durg of Pt. Ravi Shankar Shukla University, Raipur (C.G.), and Doctor of Philosophy in Mechanical Engineering from National Institute of Technology Raipur, Raipur, on August 2016. He is working as an Assistant Professor (SG) in the Mechanical Engineering Department of Jaypee University of Engineering and Technology, Guna (M.P.). He has published more than 125 research articles in peer-reviewed international, national journals, and conferences. His area of interest includes Solar thermal Applications, Renewable Energy, Python Programming, Numerical Computations, Internal combustion engines, and Solar Water Desalination.

Pankaj Dumka has completed his B. Tech. (Mechanical Engineering) from Inderprastha Engineering College, Ghaziabad in 2007, M. Tech. from Indian Institute of Technology, Kanpur in "Fluids and Thermal Sciences" in 2010, and PhD from Jaypee University of Engineering and Technology, Guna in 2021. He has been working with the Jaypee University of Engineering and Technology as an assistant professor in the Department of Mechanical Engineering since 2011. He has published more than 40 research articles in reputed SCI and SCOPUS indexed Journals. His area of interest includes Thermodynamics, Fluid Dynamics, Computational Fluid Dynamics, Numerical Computations, Python programming, and Solar water desalination.

How to cite this paper: Priyanshi Mishra, Pramiti Tewari, Dhananjay R. Mishra, Pankaj Dumka, "Integration based on Monte Carlo Simulation", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.9, No.3, pp. 58-65, 2023. DOI: 10.5815/ijmsc.2023.03.05