International Journal of Scientific Research in Multidisciplinary Studies
Vol.9, Issue.3, pp.06-10, March 2023
E-ISSN: 2454-9312 P-ISSN: 2454-6143
Available online at: www.isroset.org

Research Paper

Solving Double Integration With The Help of Monte Carlo Simulation: A Python Approach

Priyanshi Mishra[1], Ayush Sharma[2], Dhananjay R. Mishra[3], Pankaj Dumka[4*]

[1,2]Dept. of Computer Science Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India
[3,4]Dept. of Mechanical Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India

**Corresponding Author: p.dumka.ipec@gmail.com*

Received: 08/Feb/2023; **Accepted**: 11/Mar/2023; **Published**: 31/Mar/2023. | **DOI**: https://doi.org/10.26438/ijsrms/ v9i3.610

Abstract— In this article, a model of the double integration process has been attempted using Monte Carlo simulation. Both non-rectangular and rectangular domains are dealt separately. Python programming has been used to automate the entire random number-based integration process. Three example problems are used to test the newly created cods, and the results achieved are consistent with those reported in the literature.

Keywords— Double integration; Monte Carlo simulation; Monte Carlo integration; Python programming; Integration; Stochastic Integration

1. Introduction

Many times, a technological issue produces a system of differential equations that cannot be resolved in closed form, or analytically, hence numerical/probabilistic techniques that approximate the solution are applied [1]–[3]. Numerical methods involve simple arithmetic operations, whether they are based on series expansions or are purely numerical methods that assess the unknown integral at predefined intervals of time. Whereas, in probabilistic approach random variables are used to evaluate the area under the curve or volume of the region [4]–[6]. The results of numerical/probabilistic methods are simply rough approximations of the genuine values.

Double integrals are necessary, for instance, to calculate a region's area, the volume below the surface, and the mean value of a two-variable function over a rectangular region [7]. At the same time, it is not that easy to solve double integrals analytically. Hence, a numerical/probabilistic approach is needed to find the answer to a double integral. The multiple-segment trapezoidal rule [8]–[10]can be used to handle the numerical integration. Whereas, Monte Carlo method is a probabilistic approach where random variables are generated in the domain and the probability of occurrence of the points in the domain will return the result [11], [12]. The double integration based on the Monte Carlo approach can be understood by staring with an integration problem as shown in Eq. 1.

$$I = \int_{x_1}^{x_2} \int_{y_1}^{y_2} f(x,y)dydx \tag{1}$$

The above integration can also be written as the product of area of rectangle formed by rectangle (x_1, x_2) & (y_1, y_2) and the average value of function $f(x, y)$ as follows:

$$I = \int_{x_1}^{x_2} \int_{y_1}^{y_2} f(x,y)dydx = A \times f_{average} \tag{2}$$

where, $A = (x_2 - x_1) \times (y_2 - y_1)$

So, the integral might be assessed if there had been a method for determining the average value of the integrand that was independent. The random numbers can be applied in this situation. Consider a list of random values, *x* and *y*, that are uniformly spaced apart between (x_1, x_2) & (y_1, y_2). Simply evaluate $f(x, y)$ at each of the randomly chosen points and divide the result by the total number of points to determine the function average, as shown in Eq. 3 [13], [14].

$$f_{average} = \frac{1}{N} \sum_{i=1}^{N} f(x,y) \tag{3}$$

Now Eq. 3 can be solved to obtain average value of the function and finally from Eq. 2 the value of integration can be obtained.

The development of a mathematical technique for computing double numerical integration based on the general Monte Carlo approach mentioned above is the main objective of this study. Even after creating the approach, the process for solving it with pen and paper with high degree of accuracy is not possible. The significance of mathematical programming is now apparent. Mathematical calculations can be automated,

which not only speeds up processing but also yields results that are error-free (with a certain order of accuracy). Programming languages with simple syntax and straightforward implementation include Python [15]–[18]. This programming language has a large number of modules and very rich in the community support [19], [20]. NumPy [16], [21]–[24] which stand for 'Numerical Python' is a Python module dedicated to perform numerical computations. In this article double integration has been performed based on Monte Carlo simulation. The numerical procedure is developed for rectangular and non-rectangular domain and the method is implemented in Python programming. Two sets of code have been prepared to solve any type of double integrals with constant limits for rectangular and non-rectangular cases. The authors are optimistic that the scientific community will find our work useful in solving numerical double integration problems involving uneven data values.

2. Mathematical Formulation

In case of double integration two basic problems are encountered:

a) Double integrals on fixed rectangular regions

$$\int_{x_1}^{x_2}\int_{y_1}^{y_2} f(x,y)dydx$$

x_1, x_2, y_1, y_2 are all constant values (numbers).

b) Double integration on non-rectangular regions

$$\iint_R f(x,y)dxdy$$

where, *R* is a region in *x*-*y* plane over which the function has to be integrated.

In case (a), the Monte Carlo simulation is applied as follows:

- First *x* and *y* domains are set. In this domain generate *N* number of uniform random numbers *x* and *y*. (*N* is the large number of points)
- Evaluate $f(x,y)$ at all the *N* random data points and then add them up.
- Take the average and multiply by the area of the rectangle formed by (x_1, x_2) and (y_1, y_2). This will return the integration value.

In case (b), the modified Monte Carlo methodology will be as follows:

- First *x* and *y* domains are set. In this domain generate *N* number of uniform random numbers *x* and *y*. (*N* is the large number of points)
- Evaluate $f(x,y)$ at all the random data points that lies in the region *R* and then add them up.
- Take the average and multiply by the area of the rectangle formed by (x_1, x_2) and (y_1, y_2). This will return the integration value.

3. Implementation in Python

The following steps were used to model the above mathematical procedure to obtain a double integral using Monte Carlo simulation:

(a) Fixed rectangular regions

- Define the function which has to be integrated
- Supply the limits for *x* and *y* domain
- Supply number of sample points (*N*)
- Within a loop that runs *N* times perform following:
- Generate random points *x* and *y* within the domain.
- Evaluate $f(x,y)$ at each random point and add them all ($\sum f(x,y)$).
- Evaluate the area *A* of the rectangular domain
- Multiply A and $\sum f(x,y)$ and assign this product to *I*.
- Print *I* as final answer.

The Python program is as follows:

```
from numpy import*

#=========DEFINE FUNCTION=========#
# Defining Function f(x,y)
# Write your function in place of DF
def f(x,y):
    fn =# DF
return fn

#=========DEFINE DOMAIN============#
#replace x_min,x_max by the num. val.
x_domain=#(x_min,y_min)
#replace y_min,y_max by the num. val.
y_domain=#(y_min,y_max)

#==SUPPLY NUMBER OF RANDOM POINTS==#
# Number of sample points in domain
N=100000


#==================================#
#               MAIN PROGRAM
#==================================#
# variable for Σf
Σf=0

for i inrange(N):

#Generating rand. x and y coordinates
x=random.uniform(x_domain[0],x_domain[
1])
y=random.uniform(y_domain[0],y_domain[
1])

#Eval. f(x,y) based on rand. pts.
    Σf=Σf+f(x,y)

# Area of rectangle obtained by x,y
A=(x_domain[1]-
```



```
    x_domain[0])*(y_domain[1]-y_domain[0])

38  # Finding average of <f>
39  f_average=Σf/N
40
41  # Final integration
42  I=A*f_average
43
44  print(I)

45
```

(b) Non-rectangular regions

- Define the function which has to be integrated
- Define the function for region
- Supply the limits for *x* and *y* domain
- Supply number of sample points (*N*)
- Within a loop that runs *N* times perform following:
  - Generate random points *x* and *y* within the domain.
  - Evaluate $f(x,y)$ at each random point which lie in the region R and then add them all ($\sum f(x,y)$).
- Evaluate the area *A* of the rectangular domain
- Multiply *A* and $\sum f(x,y)$ and assign this product to *I*.
- Print *I* as final answer.

The Python program is as follows:

```
 1  from numpy import*
 2
 3  #====DEFINE FUNCTION=======#
 4  # Defining Function f(x,y)
 5  # Write your fun. in place of DF
 6
 7  def f(x,y):
 8      fn=#DF
 9  return fn
10
11  #=====DEFINE THE REGION (R)======#
12  # Defining Function R(x,y)
13  #        R(x,y)<=rb
14
15  # Write your region in place of RF
16  def R(x,y):
17      r=#RF
18  return r
19
20  # Supply the region bound
21  # Replace RB by the value of bound
22  rb=# RB
23
24  #=======DEFINE DOMAIN============#
25  #replace x_min,x_max by the num.
26  val.
27  x_domain=#(x_min,y_min)
28  #replace y_min,y_max by the num.
29  val.
30  y_domain=#(y_min,y_max)

31  #=SUPPLY NUMBER OF RANDOM POINTS=#
    # No. of sample points in domain
32  N=100000
33
34
35  #================================#
36  #              MAIN PROGRAM
37  #================================#
38  # variable for Σf
39  Σf=0
40
41  for i inrange(N):
42
43  #Generating rand. x and y coord.
44      x=random.uniform(x_domain[0],
    x_domain[1])
45      y=random.uniform(y_domain[0],
    y_domain[1])
46
47  # Eval. f(x,y) based on rand. pts.
48  # Condition check:
49  if R(x,y)<=rb:
50          Σf=Σf+f(x,y)
51
52  # Area of rectangle obtained by x,y
53  A=(x_domain[1]-x_domain[0])*
    (y_domain[1]-y_domain[0])
54
55  # Finding average of <f>
56  f_average=Σf/N
57
58  # Final integration
59  I=A*f_average
60
61  print(I)
```

(Note: The above codes are written in Jypyter notebook due to which we were able to use the symbols as the variables.)

4. Example Problems

First the case (a) i.e., fixed rectangular region is taken then the case (b) i.e., non-rectangular region will be taken.

Case (a)

Example 1: Evaluate

$$I=\int_{y=0}^{3}\int_{x=0}^{5} e^{y+x}dxdy$$

Solution:

As this is case (a) so the first code in section 3 will be taken. Replacing DF in line 7 of the code by e^{y+x} as follows:

```
def f(x,y):
# Replace fn with your function
    fn =exp(y+x)
   return fn
```


Replace the x_{min} & x_{max} and y_{min} & y_{max} in line 12 and 14 of the code with the following data points:

x_{min}	0	y_{min}	0
x_{max}	5	y_{max}	3

Also, supply the number of random data points (*N*), in the code we have used 10^5. The program output will be as follows:

I= 2823.360716173284

Example 2: Evaluate

$$I = \int_{y=1}^{3}\int_{x=1}^{4} \ln(x+2y)dxdy$$

Solution:

As this is case (a) so the first code in section 3 be taken. Replacing DF in line 7 of the code by $\ln(x+2y)$ as follows:

```
def f(x,y):
# Replace fn with your function
    fn =log(x+2*y)
 return fn
```

Replace the x_{min} & x_{max} and y_{min} & y_{max} in line 12 and 14 of the code with the following data points:

x_{min}	1	y_{min}	1
x_{max}	4	y_{max}	3

Also, supply the number of random data points (N). The program output will be as follows:

I= 11.067136407685592

Case (b)

Example 3: Evaluate

$$I = \iint_R e^{-(x^2+y^2)}dxdy$$

where, $R = \{(x,y) \mid x^2 + y^2 \le 1\}$

Solution:

As this is case (b) so the second code in section 3 will be taken. Replacing DF in line 8 of the code by $e^{-(x^2+y^2)}$ as follows:

```
def f(x,y):
# Replace fn with your function
    fn =exp(-(x**2+y**2))
   return fn
```

Replace the RF by the function $x^2 + y^2$ in line 17 and RB in line 22 by 1 as follows:

```
def R(x,y):
    r= x**2+y**2
   return r

# Supply the region bound
# Replace RB by the value of bound
 rb=1
```

As the circle (with centre (0,0) and radius 1 unit) will be bounded by the square of side length 2 units so, replace the x_{min} & x_{max} and y_{min} & y_{max} in line 26 and 28 of the code with the following data points:

x_{min}	-1	y_{min}	-1
x_{max}	1	y_{max}	1

Also, supply the number of random data points (N). The program output will be as follows:

I= 1.986231067553296

5. Conclusion and Future Scope

When it is difficult to arrive at an analytical solution, numerical or probabilistic approaches are used. For complex functions, double integration is widely used to calculate the area of a region and the volume below the surface. A double integration approach, which is based on the Monte Carlo simulation, is discussed in this article, and the process is modelled using Python programming. For both rectangular and non-rectangular domains, the technique is created. Three example problems were examined using the developed computer program (two for rectangular domain and one for non-rectangular domain), and the findings produced are consistent with those reported in the literature. The mathematical strategy and the developed Python program will help the researchers perform integration based on non-conventional probabilistic approach.

Conflict of Interest
Authors declare that they do not have any conflict of interest.

Funding Source
None

Authors' Contributions
Author-1: Methodology and Programming. Author-2: Data analysis and testing and validation of the research. Author-3: Revised that final draft and checked the mathematics. Author-4: Methodology, Programming, and wrote the first draft of the manuscript. (All authors reviewed and edited the manuscript and approved the final version of the manuscript.)

References

[1] J. D. Faires and R. L. Burden, *Numerical methods*. Thomson, 2003.

[2] G. Beer, B. Marussig, and C. Duenser, "Numerical integration," *Lect. Notes Appl. Comput. Mech.*, vol. **90**, pp. **129–149, 2020**, doi: 10.1007/978-3-030-23339-6_7.

[3] S. Mordechai, *Applications of Monte Carlo Method in Science and Engineering*. InTech, **2012**. doi: 10.5772/1954.

[4] R. L. Harrison, "Introduction to Monte Carlo simulation," in *AIP Conference Proceedings*, 2009, vol. **1204, no. 1, pp. 17–21**. doi: 10.1063/1.3295638.

[5] C. Doloi, J. Gogoi, and G. Pradhani, "Research Paper A Cascade System Reliability Model for Rama Distributed Random Variables," *Int. J. Sci. Res. Math. Stat. Sci.*, vol. **10, no. 1, pp. 1–9, 2023.**

[6] Nikita and S. Malik, "Generalized Logarithmic Ratio and Product-Type Estimators in Simple Random Sampling (SRS)," *Int. J. Sci. Res. Math. Stat. Sci.*, vol. **9**, no. **6**, pp. **56–60, 2022.**

[7] J. F. Epperson, *An introduction to numerical methods and analysis*. John Wiley \& Sons, **2021**. doi: 10.1002/9781119604754.

[8] P. Dumka, R. Dumka, and D. R. Mishra, *Numerical Methods Using Python*. BlueRose, **2022**.

[9] B. Fornberg, "Improving the accuracy of the trapezoidal rule," *SIAM Rev.*, vol. **63**, no. 1, pp. **167–180, 2021,** doi: 10.1137/18M1229353.

[10] A. F. Abdulhameed and Q. A. Memon, "An improved Trapezoidal rule for numerical integration," in *Journal of Physics: Conference Series*, 2021, vol. **2090, no. 1, p. 12104**. doi: 10.1088/1742-6596/2090/1/012104.

[11] K. Binder, D. Heermann, L. Roelofs, A. J. Mallinckrodt, and S. McKay, "Monte Carlo Simulation in Statistical Physics," *Comput. Phys.*, vol. 7, no. 2, p. **156, 1993,** doi: 10.1063/1.4823159.

[12] R. L. Harrison, "Introduction to Monte Carlo simulation," in *AIP Conference Proceedings*, 2009, vol. 1204, pp. 17–21. doi: 10.1063/1.3295638.

[13] M. N. Akhtar, M. H. Durad, and A. Ahmed, "Solving Initial Value Ordinary Differential Equations By Monte Carlo Method," *Proceeding IAM*, pp. **149–174, 2015.**

[14] Wei Zhong and Zhou Tian, "Solving initial value problem of ordinary differential equations by Monte Carlo method," in *Multimedia Technology (ICMT), IEEE*, **2011**, pp. **2577–2579.** doi: 10.1109/icmt.2011.6002604.

[15] S. K. R., "Python -The Fastest Growing Programming Language," *Int. Res. J. Eng. Technol.*, vol. 4, no. **12**, pp. **354–357, 2017.**

[16] K. Gajula, V. Sharma, B. Sharma, D. R. Mishra, and P. Dumka, "Modelling of Energy in Transit Using Python," *Int. J. Innov. Sci. Res. Technol.*, vol. 7, no. **8**, pp. **1152–1156, 2022.**

[17] P. Dumka, A. Deo, K. Gajula, V. Sharma, R. Chauhan, and D. R. Mishra, "Load and Load Duration Curves Using Python," *Int. J. All Res. Educ. Sci. Methods*, vol. **10**, no. 8, pp. **2127–2134, 2022.**

[18] P. Dumka and D. R. Mishra, "Understanding the TDMA / Thomas algorithm and its Implementation in Python," *Int. J. All Res. Educ. Sci. Methods*, vol. **10**, no. **10**, pp. **998–1002, 2022.**

[19] P. Dumka, P. S. Pawar, A. Sauda, G. Shukla, and D. R. Mishra, "Application of He's homotopy and perturbation method to solve heat transfer equations: A python approach," *Adv. Eng. Softw.*, vol. 170, no. May, p. 103160, 2022, doi: 10.1016/j.advengsoft.2022.103160.

[20] J. Ranjani, A. Sheela, and K. Pandi Meena, "Combination of NumPy, SciPy and Matplotlib/Pylab-A good alternative methodology to MATLAB-A Comparative analysis," in *Proceedings of 1st International Conference on Innovations in Information and Communication Technology, ICIICT 2019*, 2019, pp. 1–5. doi: 10.1109/ICIICT1.2019.8741475.

[21] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011, doi: 10.1109/MCSE.2011.37.

[22] P. Dumka, R. Chauhan, A. Singh, G. Singh, and D. Mishra, "Implementation of Buckingham ' s Pi theorem using Python," *Adv. Eng. Softw.*, vol. **173**, no. July, p. **103232, 2022,** doi: 10.1016/j.advengsoft.2022.103232.

[23] P. Dumka, K. Rana, S. Pratap, S. Tomar, P. S. Pawar, and D. R. Mishra, "Modelling air standard thermodynamic cycles using python," *Adv. Eng. Softw.*, vol. 172, no. July, p. **103186, 2022,** doi: 10.1016/j.advengsoft.2022.103186.

[24] P. S. Pawar, D. R. Mishra, and P. Dumka, "Solving First Order Ordinary Differential Equations using Least Square Method : A comparative study," *Int. J. Innov. Sci. Res. Technol.*, vol. 7, no. 3, pp. **857–864, 2022.**

AUTHORS PROFILE

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.

Ayush Sharma has done his schooling from Sir Padampat Singhania Education Centre, Kanpur. Currently he is pursuing B. Tech. in Computer Science and Engineering from Jaypee University of Engineering and Technology, Guna, Madhya Pradesh (India). His interest is in Mathematical Programming and Numerical methods.

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, 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.

Call for Papers:
Authors are cordially invited to submit their original research papers, based on theoretical or experimental works for publication in the journal.

Make a Submission

All submissions:
- must be **original**
- must be **previously unpublished research results**
- must be **experimental or theoretical**
- must be in **the journal's prescribed Word template**
- and will be **peer-reviewed**
- may not be **considered for publication elsewhere at any time during the review period**