MATHEMATICS AND COMPUTATIONAL SCIENCES

Vol 4(2), 2023, PP:46-54
DOI:10.30511/MCS.2023.1990951.1115
ISSN:2717-2708
http://mcs.qut.ac.ir/

Qom University of Technology

Numerical modelling of double integration with different data spacing: A Python-based approach

Priyanshi Mishra[1], Pramiti Tewari[2], Dhananjay R. Mishra[3], Pankaj Dumka[4]

Abstract: In this article, an attempt has been made to model the process of double integration for different data spacing. The trapezoidal rule has been used to perform integration, whereas Newton's divided difference handles the inconsistent data points. The whole process of numerical integration has been automated in Python programming language. The developed code is tested against two example problems, and the results obtained agree with the one shown in the literature.

Keywords: Numerical double integration; Trapezoidal method of integration; Numerical integration; Python programming; Double integration
2020 Mathematics Subject Classification: 65D30
Receive: 04 March 2023, **Accepted:** 02 April 2023

1 Introduction

Numerical integration involves predicting the value of a certain integral from the collection of numerical values for the integrand [2, 8]. On many occasions, a technological problem generates a set of differential equations that cannot be solved in closed form, i.e. analytically, so numerical methods which approximate the solution are used [7]. These techniques rely on series expansions or may be purely numerical techniques that evaluate the unknown integral at predetermined intervals of definitions, but they all use straightforward arithmetic operations [7]. Solutions from numerical approaches are only approximate representations of true values. For example, determining the area of a region, the volume beneath the surface, and the mean value of a two-variable function over a rectangular region requires double integrals [21]. At the same time, it is not that easy to solve double integrals analytically as has been reported by several researchers [13, 18].
Hence, a numerical approach is needed to find the answer to a double integral. The multiple-segment trapezoidal rule [1, 4, 9] can be used in the first dimension (inner integral) while holding each value of the second dimension

[1] Department of Computer Science Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India. Email: 24priyanshimishra@gmail.com
[2] Department of Computer Science Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India. Email: 5287.pramiti@gmail.com
[3] Department of Mechanical Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India. Email: dm30687@gmail.com
[4] Corresponding author: Department of Mechanical Engineering, Jaypee University of Engineering and Technology, Guna-473226, Madhya Pradesh, India. Email: p.dumka.ipec@gmail.com

constant to evaluate the double integral numerically. The second dimension would then be integrated using a similar procedure. Many solutions to the numerical double integration problem for equal data sets have been discovered. But when the data sets are unequal, researchers have struggled to find solutions. Therefore, the major goal of this study was to develop a mathematical method for computing double numerical integration in the case of uneven data spaces. Even after developing the method, the solution procedure is not easy while solving it on pen and paper. Moreover, the time required will be too high if one is interested in finding out the result, which is accurate up to 5-6 decimal points.

Here comes the importance of mathematical programming. By automating the process of mathematical computations, one can not only save computation time but also produce error-free results (with a certain order of accuracy). Python programming is one such language that is easy to implement and has a very light syntax [11, 20]. This programming language has a vast pool of libraries and very rich community support [3, 5, 16]. Numpy [10] which stands for numerical Python is a Python module dedicated to perform numerical computations with very ease. It can also be thought as the cornerstone Python module for scientific computing [6, 12]. A multidimensional array object, various derived objects (like masked arrays and matrices), and a variety of routines for quick operations on arrays are provided by Numpy. These operations include discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more [14, 15, 17, 19].

In this article double integration has been performed numerically with unequal data distribution. The numerical procedure is implemented in Python, and code has been prepared to solve any type of double integrals with constant limits. 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

Consider a double integration of a two dimensional function $f(x, y)$ with intervals (y_i, y_{i+1}) in y and (x_i, x_{i+1}) in x (which are unequal) as follows:

$$A = \int_{y_j}^{y_{j+1}} \int_{x_i}^{x_{i+1}} f(x, y)\, dx\, dy \tag{2.1}$$

First, the trapezoidal rule is applied to the inner integral considering y as constant and then on to the outer integral for which the x is constant. Consider the following equation first:

$$B = \int_{x_i}^{x_{i+1}} y\, dx$$

Then by applying Newton's divided difference formula (up to two values), the value of the above integral can be written as:

$$B = \int_{x_i}^{x_{i+1}} (y_i + (x - x_i)\, \Delta y)\, dx$$

Where, $\Delta y = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$. On performing the integration and substituting the value of Δy, we will get the following:

$$B = \frac{x_{i+1} - x_i}{2}(y_i + y_{i+1}) \tag{2.2}$$

Now replace the value of B in the double integration formula as follows:

$$A = \int_{y_j}^{y_{j+1}} B\, dy = \int_{y_j}^{y_{j+1}} \frac{x_{i+1} - x_i}{2}(f_{i,j} + f_{i+1,j})\, dy$$

$$A = \left(\frac{x_{i+1} - x_i}{2}\right)\left[\int_{y_j}^{y_{j+1}} f_{i,j}\, dy + \int_{y_j}^{y_{j+1}} f_{i+1,j}\, dy\right]$$

In the above formula $f_{i,j} = f(x_i, y_j)$. The way in which B has been evaluated in a similar way the above integrals becomes:

$$A = \left(\frac{x_{i+1} - x_i}{2}\right)\left[\frac{y_{j+1} - y_j}{2}\left(f_{i,j} + f_{i,j+1}\right) + \frac{y_{j+1} - y_j}{2}\left(f_{i+1,j} + f_{i+1,j+1}\right)\right]$$

Finally, the double difference formula gets reduced to:

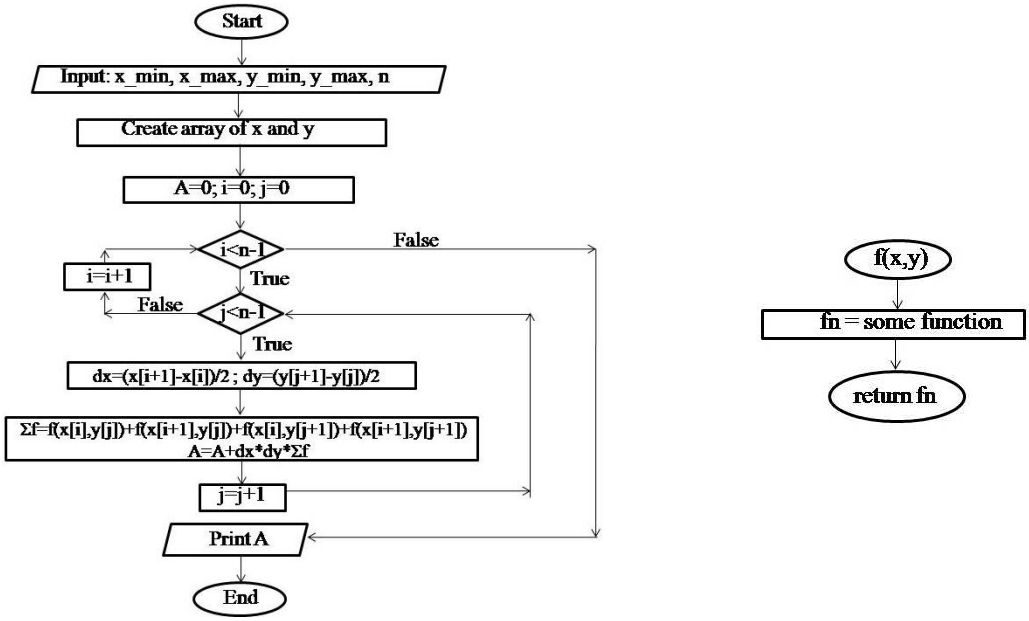
$$A = \left(\frac{x_{i+1} - x_i}{2}\right)\left(\frac{y_{j+1} - y_j}{2}\right)\left[f_{i,j} + f_{i,j+1} + f_{i+1,j} + f_{i+1,j+1}\right] \tag{2.3}$$

3 Python Implementation

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

- Define the function which has to be integrated
- Supply the limits for x and y
- Supply number of data points (n)
- Create n unequal-spaced data points arrays of x and y
- Set output variable (A) to 0, then within two for loops (one for outer integral and the other for inner integral), solve Eq. 3 and increment A by it.

The flow chart based on the above algorithm is shown in Fig. 1.



(a) Main Program **(b) Function**

Fig. 1: Flow chart

The Python program is as follows:

```
from numpy import *

#=========Part 1============#
# Defining function
def f(x,y):
    # Write your function in place of DF
    fn = # DF
    return fn


#======== Part 2============#
# Setting integral limits

# x limits
x_min=float(input("Enter the x_min\n"))
x_max=float(input("Enter the x_max\n"))

# y limits
y_min=float(input("Enter the y_min\n"))
y_max=float(input("Enter the y_max\n"))

#======== Part 3============#
# Set the number of data points
n= int(input("Enter number of data points: "))


#===========================#

# Creating array of x and y data points
# Equal spacing
x=linspace(x_min, x_max,n)
y=linspace(y_min, y_max,n)

# Unequal spacing
x=x**2/x[-1]
y=y**2/y[-1]

#===========================#
A=0
for i in range(n-1):
    for j in range(n-1):
        dx=(x[i+1]-x[i])/2
        dy=(y[j+1]-y[j])/2
        Σf=f(x[i],y[j])+f(x[i+1],y[j])+f(x[i],y[j+1])+f(x[i+1],y[j+1])
        A=A+dx*dy*Σf

print(f"The value of double integration is : {round(A,5)}")
```

4 Example Problems

In this section some example problems are solved to demonstrate the use of the algorithm presented in the article. Same computer program has been used with the change in the target function and the integration limits. In total five numerical problems have been taken to show the use and efficiency of the developed Python code.

Example 1: Evaluate below mentioned integral

$$A = \int_{y=0}^{3} \int_{x=0}^{5} e^{y+x}\, dx\, dy$$

Solution:

Supply the function of the problem in hand by replacing DF in Part 1 of the code with e^{y+x} as follows:

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

Run the program and supply the following x and y limits:

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

Also, supply the number of unequal data points which are to be created between minimum and maximum values of x and y (in this case, $n = 50$). The program output will be as follows:

```
Enter the x_min
0
Enter the x_max
5
Enter the y_min
0
Enter the y_max
3
Enter number of data points: 50
The value of double integration is : 2823.85924
```

Example 2: Evaluate below mentioned integral

$$A = \int_{y=1}^{7} \int_{x=1}^{5} \ln(x+y)\, dx\, dy$$

Solution:

Supply the function of the problem in hand by replacing DF in Part 1 of the code with $\ln(x+y)$ as follows:

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

Run the program and supply x & y limits and the number of data points (n):

$x_{\min}$	1
$x_{\max}$	5
$y_{\min}$	1
$y_{\max}$	7
n	50

The program output will be as follows:

```
Enter the x_min
1
Enter the x_max
5
Enter the y_min
1
Enter the y_max
7
Enter number of data points: 50
The value of double integration is : 56.65021
```

Example 3: Evaluate below mentioned integral

$$A = \int_{y=0}^{2} \int_{x=0}^{1} (2x + y)^8 \, dx \, dy$$

Solution:

Supply the function of the problem in hand by replacing DF in Part 1 of the code with $(2x + y)^8$ as follows:

```
def f(x,y):
    # Replace fn with your function
    fn = (2*x+y)**8
    return fn
```

Run the program and supply x & y limits and the number of data points (n):

$x_{\min}$	0
$x_{\max}$	1
$y_{\min}$	0
$y_{\max}$	2
n	50

The program output will be as follows:

```
Enter the x_min
0
Enter the x_max
1
Enter the y_min
0
Enter the y_max
2
Enter number of data points: 50
The value of double integration is : 5842.23122
```

Example 4: Evaluate below mentioned integral

$$A = \int_{y=\pi}^{3\pi} \int_{x=0}^{\pi} (\sin(x) + \cos(y))\, dx\, dy$$

Solution:

Supply the function of the problem in hand by replacing DF in Part 1 of the code with $(\sin(x) + \cos(y))$ as follows:

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

Run the program and supply x & y limits and the number of data points (n):

$x_{\min}$	0
$x_{\max}$	3.14
$y_{\min}$	3.14
$y_{\max}$	9.42
n	50

The program output will be as follows:

```
Enter the x_min
0
Enter the x_max
3.14
Enter the y_min
3.14
Enter the y_max
9.42
Enter number of data points: 50
The value of double integration is : 14.03601
```

Example 5: Evaluate below mentioned integral

$$A = \int_{y=0}^{2} \int_{x=0}^{1} \frac{xy^3}{x^2 + 1}\, dx\, dy$$

Solution:

Supply the function of the problem in hand by replacing DF in Part 1 of the code with $\left(\frac{xy^3}{x^2+1}\right)$ as follows:

```
def f(x,y):
    # Replace fn with your function
    fn = x*y**3/(x**2+1)
    return fn
```

Run the program and supply x & y limits and the number of data points (n):

$x_{\min}$	0
$x_{\max}$	1
$y_{\min}$	0
$y_{\max}$	2
n	50

The program output will be as follows:

```
Enter the x_min
0
Enter the x_max
1
Enter the y_min
0
Enter the y_max
2
Enter number of data points: 50
The value of double integration is : 1.38756
```

5 Conclusion

Numerical procedures are adopted when an analytical answer cannot be achieved easily. For example, double numerical integration is frequently employed to determine the area of a region and the volume beneath the surface for difficult functions. But, the difficulty is even more when the data points are not evenly distributed. This article discusses the double integration algorithm for uneven data points, and the procedure is automated with the help of a Python program. The numerical integration has been done by trapezoidal rule with a supplement of Newton's divided difference formula (to handle uneven data points). The developed computer program has been tested on two example problems, and the obtained results agree with the results shown in the literature. The mathematical strategy and the developed Python program will help the researchers perform numerical integration with less time and great precision.

References

[1] A.F. Abdulhameed, Q.A. Memon, An improved Trapezoidal rule for numerical integration, Journal of Physics: Conference Series, 2090(1) 2021, 12104.

[2] G. Beer, B. Marussig, C. Duenser, Numerical integration, Lecture Notes in Applied and Computational Mechanics, 2020, 129-149.
[3] P. Dumka, R. Chauhan, A. Singh, G. Singh, D. Mishra, Implementation of Buckingham's Pi theorem using Python, Advances in Engineering Software, 173 2022, 103232.
[4] P. Dumka, R. Dumka, D.R. Mishra, Numerical Methods Using Python, Blue Rose, 2022.
[5] P. Dumka, P.S. Pawar, A. Sauda, G.Shukla, D.R. Mishra, Application of He's homotopy and perturbation method to solve heat transfer equations: A python approach, Advances in Engineering Software, 170 2022, 103160.
[6] P. Dumka, K. Rana, S. Pratap, S. Tomar, P.S. Pawar, D.R. Mishra, Modelling air standard thermodynamic cycles using python, Advances in Engineering Software, 172 2022, 103186.
[7] J.F. Epperson, An introduction to numerical methods and analysis, In An Introduction to Numerical Methods and Analysis, John Wiley & Sons, 2021.
[8] J.D. Faires, R.L. Burden, Numerical methods, Thomson, 2003.
[9] B. Fornberg, Improving the accuracy of the trapezoidal rule, SIAM Review, 63(1) 2021, 167-180.
[10] K. Gajula, V. Sharma, B. Sharma, D.R. Mishra, P. Dumka, Modelling of Energy in Transit Using Python. International Journal of Innovative Science and Research Technology, 7(8) 2022, 1152-1156.
[11] K.R. Srinath, Python-the fastest growing programming language, International Research Journal of Engineering and Technology 4(12) 2017, 354-357.
[12] A. Marowka, On parallel software engineering education using python, Education and Information Technologies, 23(1) 2018, 357-372.
[13] M. Omidpanah, S.A.A. Mirjalily, H. Kargarsharifabad, S. Shoeibi, Numerical simulation of combined transient natural convection and volumetric radiation inside hollow bricks, Journal of Heat and Mass Transfer Research, 8(2) 2021, 151-162.
[14] P.S. Pawar, D. R.Mishra, P.Dumka, Solving First Order Ordinary Differential Equations using Least Square Method: A comparative study. International Journal of Innovative Science and Research Technology, 7(3) 2022, 857-864.
[15] P.S. Pawar, D. R.Mishra, P.Dumka, M. Pradesh, Obtaining exact solutions of visco-incompressible parallel flows using python, International Journal of Engineering Applied Sciences and Technology, 6(11) 2022, 213-217.
[16] J. Ranjani, A. Sheela, K. Pandi Meena, Combination of NumPy, SciPy and Matplotlib/Pylab-A good alternative methodology to MATLAB-A Comparative analysis, Proceedings of 1st International Conference on Innovations in Information and Communication Technology, ICIICT 2019, 1-5.
[17] A. Saha, Doing Math with Python: Use Programming to Explore Algebra, Statistics, Calculus, and More! No Starch Press, 2015.
[18] S. Shoeibi, N. Rahbar, A.A. Esfahlani, H. Kargarsharifabad, Energy matrices, exergoeconomic and enviroeconomic analysis of air-cooled and water-cooled solar still: Experimental investigation and numerical simulation, Renewable Energy, 171 2021, 227-244.
[19] V. Sulzer, S.G. Marquis, R. Timms, M. Robinson, S.J. Chapman, Python Battery Mathematical Modelling (PyBaMM), Journal of Open Research Software, 9(1) 2021, 1-8.
[20] S. Van Der Walt, S.C. Colbert, G. Varoquaux, The NumPy array: A structure for efficient numerical computation. Computing in Science and Engineering, 13(2) 2011, 22-30.
[21] J. Weisman, A.G. Holzman, Engineering Design Optimization under Risk, In Management Science, 19(3) 1972, 235-249.