ProyectoFinal Ingles

August 11, 2019

Final Simulation Project

0.0.1 Systems Engineer

0.0.2 Simulation

Members

- ***Maria Jose Pelaez***
- ***Pedro Bermeo***

- ***1. Objective***

The present work contains the development of the final project of the Simulation subject, which consists in solving a double integral by means of the Monte Carlo Integral, and estimating the error that occurs with a value of N for the pseudorandom numbers.

- ***2. Theoretical Mark***

- ***History***

The Monte Carlo methods originated at Los Alamos National Laboratory shortly after World War II. Although there were some antecedents like those of Comte de Buffon that threw a needle on a board with a grid or those of Kelvin who studied the kinetic of gases, these random methods were developed and imposed with the appearance of the computer. Once the first United States electronic computer (ENIAC) was completed, Los Alamos scientists considered its use for the design of thermonuclear weapons. In 1946 Stanislaw Ulam suggested the use of random sampling to simulate the neutron path and John von Neumann developed a detailed proposal in early 1947. These small-scale simulations were decisive to complete the project. Metropolis and Ulam (1949) published a paper in which they described their ideas, which triggered a lot of research in the 50s. The name of Monte Carlo comes from the homonymous city in Monaco, famous for its casinos. An important application of this method is the calculation of integrals, especially of high size, either because it is not possible to calculate them exactly or their computation is very difficult with the desired degree of precision. The idea of integration by Monte Carlo is to evaluate the integral.

- ***Application***

An important application of this method is the calculation of integrals, especially of high dimension, either because it is not possible to calculate them exactly or their computation is very difficult with the desired degree of precision. The idea of integration by Monte Carlo is to evaluate the integral using random sampling, use the following equation.

- ***Double Integral***

For the resolution of a double integral by the Monte Carlo method, it is based on the same process as for solving the simple integral, the only thing that varies is that for the solution of double integrals, the application of the following formula is necessary.

- ***Error Estimation***

In order to estimate the error of N pseudorandom numbers, it is necessary to use the following formulas or equations, the same ones that consist of the application of the Variance, and for this it is necessary to calculate $(^2)$ and also $()^2$, for this the equations that are specified below are used.

- ***a. First Equation***

This equation consists in the calculation of the Variance that exists between $ (^2) $ and $ ()^2 $, for this you must first calculate the Second and Third Equation.

- ***b. Second Equation***

This equation consists of replacing the pseudo-random values generated for both $ x $ and $ y $ in the function $ f (x, y) $ for that what is done is the pseudo-random values generated to pass them to the FXY method, this will return us a value corresponding to the calculation and replacement of the values in the function, to all these values a sum must be made and then divided for the amount of numbers generated.

- ***c. Third Equation***

This equation consists in doing the same for the calculation of the second equation, that is, the values of $ x, and $ in the function must be replaced, and that value is squared, and if there is a sum of all the values, and in the end it is divided for the number of numbers generated.

Finally, after the first equation must be applied which consists in subtracting the value of the second equation (f^2) less the value of the first equation (f) squared, and to this the root is applied square and thus the value of the Error will be obtained with the N generated, as exemplified in the source code.

- ***3. Integral to Solve***

For the development of this project, we start from the following equation to be solved through the Monte Carlo Integral:

- ***4. Resolution in WolframAlpha***

In order to be able to solve in this software the integral was the income of the following Sentence:

- ***(integral_0$^{3(integral_0}$3 $(log(4x)+9x^{2+13x}3y^{2+(sin((3x+2y)/(6+y)+cos(12x}$2))/(log(12-$x^{2)+(cos(log(7x+21y)))))-sin((2x)}$1/3 -y^1/2)) dx) dy)***

And as a result of the Wolfram execution, the following was obtained:
As you can see, the result that the Wolfram gave us is:

Software	Result
WolframAlpha	2629

Next, the process of the resolution of the final project is detailed.

- *5. Project development*

As a first step, we proceed to declare the Authorship of this project

```
In [1]: __author__ = "Pedro Bermeo, Maria Jose Pelaez"
        __copyright__ = "Simulacion - Universidad Politecnica Salesiana"
        __credits__ = ["Pedro Bermeo, Maria Jose Pelaez"]
        __license__ = "GPL"
        __version__ = "1.0"
        __maintainer__ = "Pedro Bermeo, Maria Jose Pelaez"
        __email__ = "pbermeoa@est.ups.edu.ec, mpelaezc@est.ups.edu.ec"
        __status__ = "Production"
```

0.0.3 Import of Libraries

Now we import the libraries needed for the simulation.

```
In [2]: import random
        import numpy as np
        from sympy import Symbol, Function, diff, solve,sqrt,cos,ln
        import scipy.integrate
        import math
        import matplotlib.pyplot as plt
        from mpl_toolkits import mplot3d
        %matplotlib inline
```

0.0.4 Declaration of a Method FXY()

We proceed to declare the method fxy, which will return the value of $f(x, y)$, of the function by replacing the values of X, Y. This method will help us to make the final graph of the function.

```
In [3]: def FXY(x,y):
            return ((9*np.power(x,2.0))+(13*(np.multiply(np.power(x,3.0),np.power(y,2.0))))+(np.
```

0.0.5 Declaration of a Method FXY2()

This method receives as parameters the values generated for x,and as well as it receives the limits for which we are going to perform the integral, this because we need to apply the montecarlo to obtain the values of x, and and replace them in the equation.

```
In [4]: def FXY2(xa,ya,a,b,c,d):
            x=(b-a)*xa+a
            y=(d-c)*ya+c
            fxy = ((9*np.power(x,2.0))+(13*(np.multiply(np.power(x,3.0),np.power(y,2.0))))+(np.l
            return fxy
```

0.0.6 Declaration of a Method estimarError()

This method will be responsible for estimating the error for N pseudorandom values, applying the formulas previously detailed.

```
In [5]:  def estimarError(a,b,c,d,nAleatorios):
                nSubs = 0
                f = 0
                f2 = 0
                fxy = 0
                for i in range(nAleatorios):
                    x = random.random()
                    y = random.random()
                    fxy += FXY2(x,y,a,b,c,d)
                    nSubs = FXY(x,y)
                    f += (nSubs)
                    f2 += np.power(nSubs, 2)
                f = f / nAleatorios
                f2 = f2 / nAleatorios
                errorCalculado = sqrt(f2 - np.power(f,2))
                print("Error con: ", nAleatorios, " #Aleatorios es de: ", errorCalculado)
                return fxy
```

0.0.7 Declaration of a Method calcularIntegralDoble()

This method will calculate the value of the double integral.

```
In [6]: def calcularIntegralDoble(a,b,c,d,n):

            fxy = (estimarError(a,b,c,d,n))

            res = ((b-a)*(d-c)/n)*fxy
            return res
```

We can see that the result obtained is 2627.92 and is very similar to the 2629 that the WolframAlfa gave us.

```
In [14]: res = calcularIntegralDoble(0.0, 3.0, 0.0, 3.0,(10000))

         print("Respuesta: ",res)
```

```
Error con:  10000  #Aleatorios es de:  4.69764577770725
Respuesta:  2627.9273786242024
```

- *5. Another form of Resolution through SCIPY*

In order to verify that the results of the resolution of the integral in wolframalpha and through the Monte Carlo method gave us correct results, then we resolved the same integral through the scripy library, since thanks to integrate.dblquad we can solve double integrals, passing it as Parameters the function and the limits with which we want to perform the two integrals, the code is as follows:

```
In [10]: import scipy.integrate as integrate
         I  = integrate.dblquad(FXY,0,3,0,3)

         print("La Repuesta de la integral es: \n", str(I[0]), "\nCon un error de: \n",str(I[1])
```

```
La Repuesta de la integral es:
 2629.003475648896
Con un error de:
 2.2479299360969132e-05
```

As you can see the result is 2629.003, and is very similar to the other two results we obtained, it should be noted that this result may vary, depending on the amount of pseudorandom numbers generated. Once the three results have been obtained, it can be concluded that the equation is well resolved.

Next, a comparative table is presented in which the three results obtained, of the initial function resolution, are detailed.

Software	Result
WolframAlpha	2629
Monte Carlo	2627.927
SCIPY	2629.003

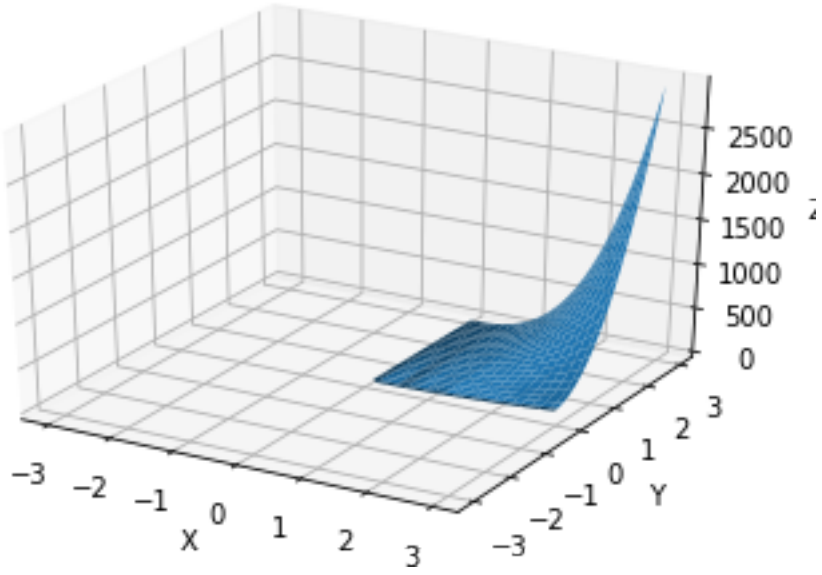
- ***6. Graph of the Function Entered***

We proceed to make the graph of the function that was entered, for it was done with the following code.

```
In [9]: fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        x = y = np.arange(-3.0, 3.0, 0.05)
        X, Y = np.meshgrid(x, y)
        zs = np.array([FXY(x,y) for x,y in zip(np.ravel(X), np.ravel(Y))])
        Z = zs.reshape(X.shape)
        ax.plot_surface(X, Y, Z)
        ax.set_xlabel('X')
        ax.set_ylabel('Y')
        ax.set_zlabel('Z')
        plt.show()
```

```
/home/mary/anaconda3/envs/simulacion/lib/python3.7/site-packages/ipykernel_launcher.py:2: Runtim
```

```
/home/mary/anaconda3/envs/simulacion/lib/python3.7/site-packages/ipykernel_launcher.py:2: Runtim
```



1 References

[1] Northeastern University. (2018). Monte Carlo error analysis. 06/08/2019, de Northeastern University Sitio web: https://web.northeastern.edu/afeiguin/phys5870/phys5870/node71.html

[2] College of Natural Sciences and Mathematics Department of Mathematics. (2018). Module for Monte Carlo Integration. 06/08/2019, de College of Natural Sciences and Mathematics Department of Mathematics Sitio web: http://mathfaculty.fullerton.edu/mathews/n2003/MonteCarloMod.html

[3] José Ignacio Illana. (2017). Métodos Monte Carlo. 06/08/2019, de Universidad de Granada Sitio web: https://www.ugr.es/~jillana/Docencia/FM/mc.pdf

[4] Universidad de Buenos Aires. (2016). Integracion por el metodo de Monte Carlo. 05/08/2019, de Universidad de Buenos Aires Sitio web: http://www.dm.uba.ar/materias/probabilidades_estadistica_C/2006/2/TpR20062.pdf

[5] Patricia Kisbye. (2016). Integración por el método de Monte Carlo. 06/08/2019, de Universidad Nacional del Nordeste Sitio web: http://www.ing.unne.edu.ar/assets/pdf/academica/departamentos/computacion/montecarlo.pdf

[6] Juan C. Cortés. (2015). Método de Montecarlo para calcular integrales. 05/08/2019, de Universitat Politècnica de València Sitio web: https://www.researchgate.net/publication/50839519_Metodo_de_Montecarlo_para_calcular_integrales

Video

Link of Video