



## CARLO SIMULATIONS

**Dr. Mohammad Rafiq Muqri, DeVry University, Pomona**

Dr. Mohammad R. Muqri is a Professor in College of Engineering and Information Sciences at DeVry University. He received his M.S.E.E. degree from University of Tennessee, Knoxville. His research interests include modeling and simulations, algorithmic computing, analog and digital signal processing.

## Objective

This teaching module was designed to enhance the knowledge and expertise of our students which enabled them to successfully apply Monte Carlo Methods to solve differential and integral equations, for finding eigen values, for inverting matrices, for evaluating multiple integrals different configurations, simulating random collisions of neutrons and subatomic particles, in statistics, in queueing models, in games of strategy, in EEG diagnostics and interpretation, thereby enhancing computational literacy and critical thinking.

A Monte Carlo Simulation is a way of approximating the value of a function where calculating the actual value is difficult or impossible. It uses random sampling to define constraints on the value and then makes a sort of "best guess."

The advances in computing have given a new meaning to the term stochastic or Monte Carlo simulations. The name "Monte Carlo" comes from the city in Monaco, famous for its gambling casinos. The Monte Carlo method (MCM), also known as the method of statistical trials is a traditional marriage of two major branches of theoretical physics the probabilistic theory of random process dealing with Brownian motion or random walk experiments and potential theory, which studies the equilibrium states of a homogenous medium. It is a method of approximately solving problems using sequences of random numbers. It is a means of treating mathematical problems by finding a probabilistic analog and then obtaining approximate answers to this analog by some experimental sampling procedure.

Monte Carlo methods have three characteristics:

1. Random sample generation
2. Known input distribution
3. Numerical experiments

The direct output of the Monte Carlo simulation method is the generation of random sampling. Other performance or statistical outputs are indirect methods which depend on the applications. There are many different numerical experiments that can be done, probability distribution is one of them. Probability distribution identifies either the probability of each value of an unidentified random variable (when the variable is discrete), or the probability of the value falling within a particular interval (when the variable is continuous). That is equivalent to saying that for random variables  $X$  with the distribution in question,  $\Pr[X = a] = 0$  for all real numbers  $a$ . That is, the probability that  $X$  attains the value  $a$  is zero, for any number  $a$ . If the distribution of  $X$  is continuous, then  $X$  is called a continuous random variable. Normal distribution, continuous uniform distribution, beta distribution, and Gamma distribution are well known absolutely continuous distributions.

## Simple Monte Carlo Estimation Examples

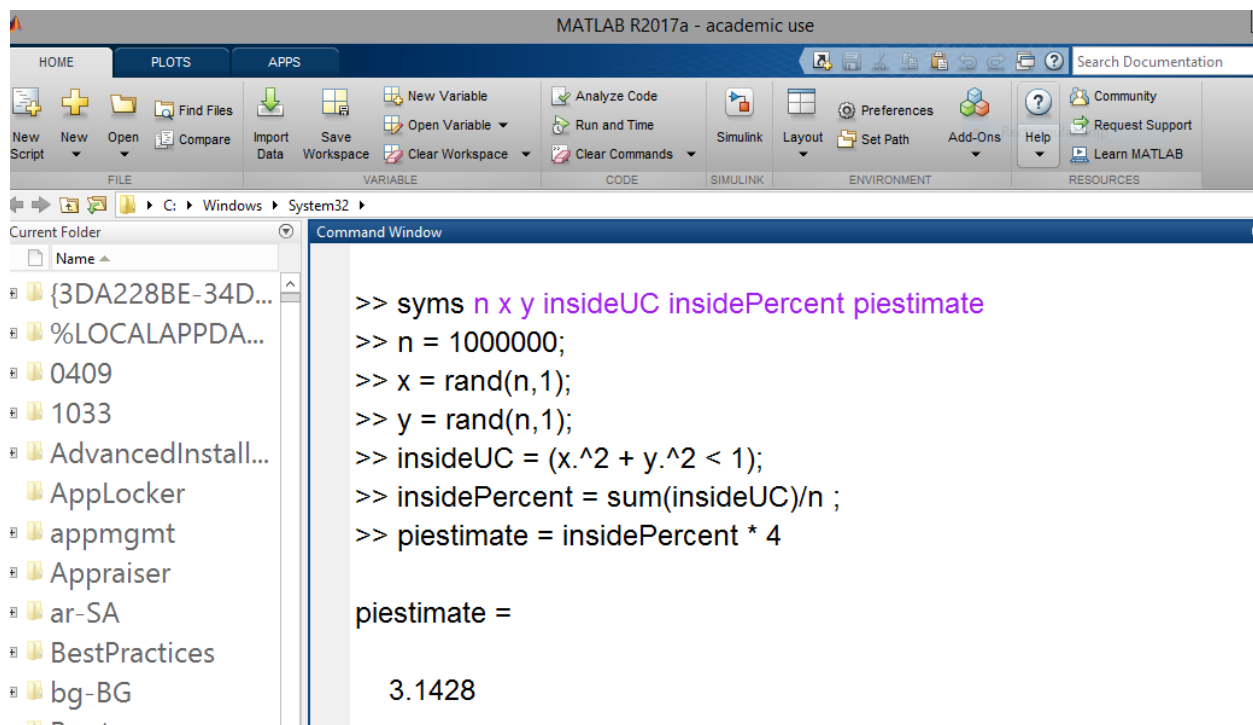
Monte Carlo methods is a class of numerical methods that relies on random sampling. If you had a circle and a square where the length of a side of the square was the same as the diameter of the circle, the ratio of the area of the circle to the area of the square would be  $\pi/4$ . So, if you put this circle inside the square and select many random points inside the square, the number of points

inside the circle divided by the number of points inside the square and the circle would be approximately  $\pi/4$

Consider the following Monte Carlo method which computes the value of  $\pi$ :

1. Uniformly scatter some points over a unit square  $[0, 1] \times [0, 1]$ .
2. For each point, determine whether it lies inside the unit circle.
3. The percentage of points inside the unit circle is an estimate of the ratio of the area inside the circle and the area of the square, which is  $\pi/4$ . Multiply the percentage by 4 to estimate  $\pi$ .

The Matlab script depicted below in Figure 1 performs the Monte Carlo computation:



```
>> syms n x y insideUC insidePercent piestimate
>> n = 1000000;
>> x = rand(n,1);
>> y = rand(n,1);
>> insideUC = (x.^2 + y.^2 < 1);
>> insidePercent = sum(insideUC)/n ;
>> piestimate = insidePercent * 4

piestimate =

    3.1428
```

Figure 1. Monte Carlo Matlab computation to estimate pi

This example represent a general procedure of Monte Carlo methods: First, the input random variables ( $x$  and  $y$ ) are sampled. Second, for each sample, a calculation is performed to obtain the outputs (whether the point is inside or not). Due to the randomness in the inputs, the outputs are also random variables. Finally, the statistics of the output random variables (the percentage of points inside the circle) are computed, which estimates the output.

Students were then given the handout and advised to compute value of pi for each value of  $n = 100$ ,  $n = 10000$  and  $n = 1000000$ , run the script 3 times.

Ex.1: Compute and interpret as to how accurate is the estimated  $\pi$  using:

1. MatLab
2. Microsoft Visual Studio and C#
3. Python

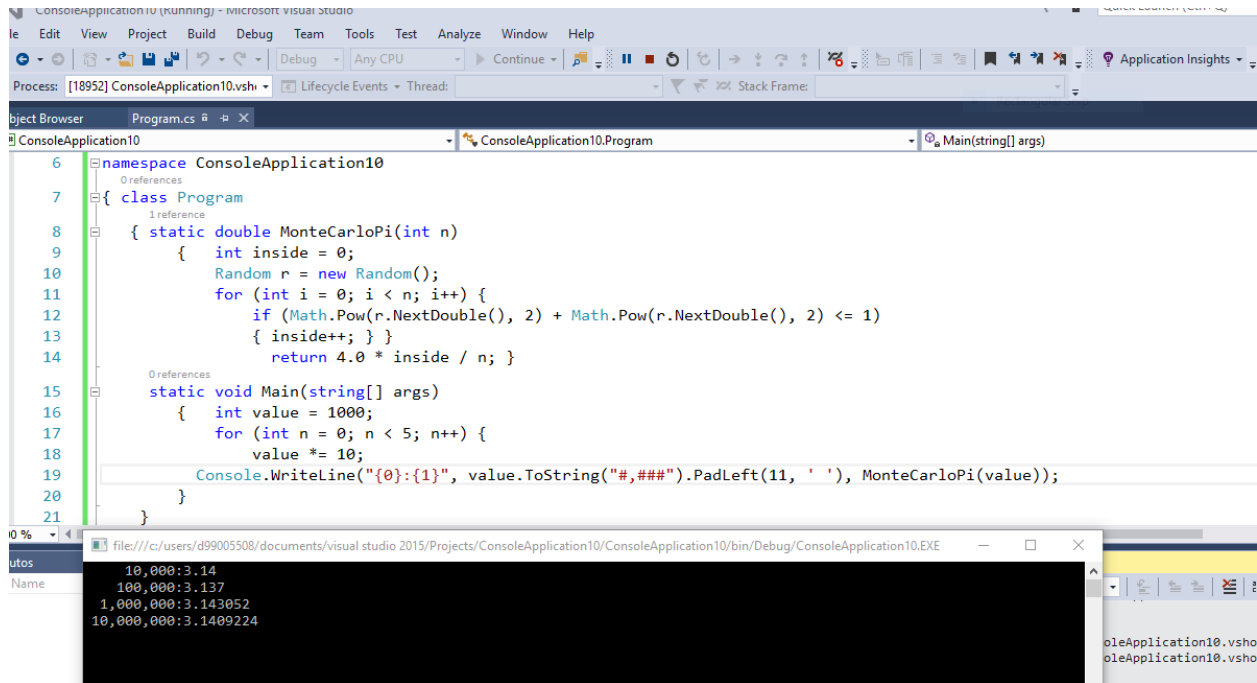


Figure 2. C# Program to compute the value of PI

```

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\d99005508
IPython console
Console 1/A
115Week5Lab.py C:\Users\d99005508\OneDrive - DeVry\Desktop\CIS115\New folder\untitled0.py
115Week5Lab.py CIS115wk4Ex1.py MC Simulation crude File 1A.py MC Sim

In [5]: from random import random
...: from math import hypot
...: try:
...:     import psyco
...:     psyco.full()
...: except:
...:     pass
...:
...: def pi(nthrows):
...:     inside = 0
...:     for i in range(nthrows):
...:         if hypot(random(), random()) < 1:
...:             inside += 1
...:     return 4.0 * inside / nthrows
...:
...: for n in [10**4, 10**6, 10**7, 10**8]:
...:     print ("%9d: %07f" % (n, pi(n)))
10000: 3.139200
1000000: 3.143648
10000000: 3.141912
100000000: 3.141712

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Feb 3 15:39:45 2020
4
5 @author: D99005508
6 """
7 from random import random
8 from math import hypot
9 try:
10     import psyco
11     psyco.full()
12 except:
13     pass
14
15 def pi(nthrows):
16     inside = 0
17     for i in range(nthrows):
18         if hypot(random(), random()) < 1:
19             inside += 1
20     return 4.0 * inside / nthrows
21
22 for n in [10**4, 10**6, 10**7, 10**8]:
23     print ("%9d: %07f" % (n, pi(n)))
24
25

```

Figure 3. Python program to compute the value of PI using a function

Ex. 2 Write the Matlab scripts and C# programs to demonstrate Monte Carlo Method to estimate the volume of a 3-dimensional ball and a ten dimensional hyperball.

Uniform random variable is special in Monte Carlo methods and in computation – most psuedo random number generators are designed to generate uniform random numbers. In MATLAB, for example, the following command generates an  $m$  by  $m$  array of  $U(0,1)$  uniform random numbers.

$x = \text{rand}(m,n)$ ; To generate an  $U(a,b)$  uniform random numbers, one can simply scale the  $U(0,1)$  random numbers by

$x = \text{rand}(m,n) * (b-a) + a$ ;

Almost all other languages used for scientific computation have similar random number generators.

Ex. 3. Determine the mean, variance and standard deviation of a  $U(a,b)$  random variable.

Non-uniform distributions are those whose probability density functions are not constant. Several simple but important non-uniform distributions are:

Triangular distribution. It is characterized by three parameters  $a, b, c$ . The probability density function is as follows:

$$f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)}, & a \leq x \leq c \\ \frac{2(b-x)}{(b-a)(c-a)}, & c \leq x \leq b \\ 0, & x < a \text{ or } x > b \end{cases}$$

Exponential distribution. It is characterized by a single parameter  $\lambda$ . The probability density function is

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Normal distribution, also known as Gaussian distribution.

Applications to EEG:

Advances in technology and the widespread availability of powerful computing devices have given a new meaning to the term applied signal processing for real world EEG analysis and applications with introduction of Monte Carlo simulations in recording and analysis of electrical activity of the human brain.

Mastery and expertise in clinical EEG interpretation is one of the most desirable diagnostic clinical skills in interpreting seizures, epilepsy, sleep disorders, and other neurocognitive studies. In most cases EEG activity is described in terms of frequency, amplitude, distribution or location, symmetry, synchrony, reactivity, morphology, rhythmicity and regulation.

The dynamic nature of epileptic phenomena causes EEG signals to exhibit **stochastic** and non-stationary behavior. The time frequency distributions are potentially very useful for detecting and analyzing non-stationary epileptic EEGs. Although visual analysis of raw EEG traces is still the major clinical tool and the point of reference for other methods, we can relate visual analysis to mathematics with a time-frequency description. The EEG signal analysis is often complemented with MEG and functional magnetic resonance imaging (fMRI) to correlate specific EEG findings with pathology of the brain and selectively demonstrate the diagnosis of certain neuronal disease processes, and assessment parameters.

One of the first applications of matching pursuit (MP) in EEG analysis was detection and parameterization of sleep spindles--structures present in sleep EEG recordings. The matching pursuit (MP) algorithm<sup>3</sup> is often used instead of popular time-frequency (TF) domain approaches such as short-time Fourier transform and the wavelet transform because of its higher temporal spatial resolution in the TF space. Some preliminary applications of the matching pursuit method have appeared, analyzing routine EEG, ictal scalp EEG, sleep EEG, and evoked potentials. Other method is the Gabor representation<sup>5</sup> that does not assume that the signal is known at arbitrary time and frequency points, but at a lattice points:  $t = nT$ ,  $w = kF$ ; where  $n, k \in \mathbb{Z}$ ;  $T$  is the sampling interval in the time domain; and  $F$  is the sampling interval in the

frequency domain. Figure 4 demonstrates examples of different shapes of Gabor functions, which can be included in the dictionary used for MP decomposition.

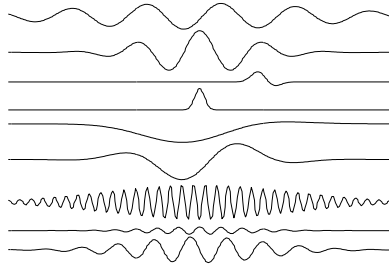


Figure 4. Shapes of different Gabor Functions

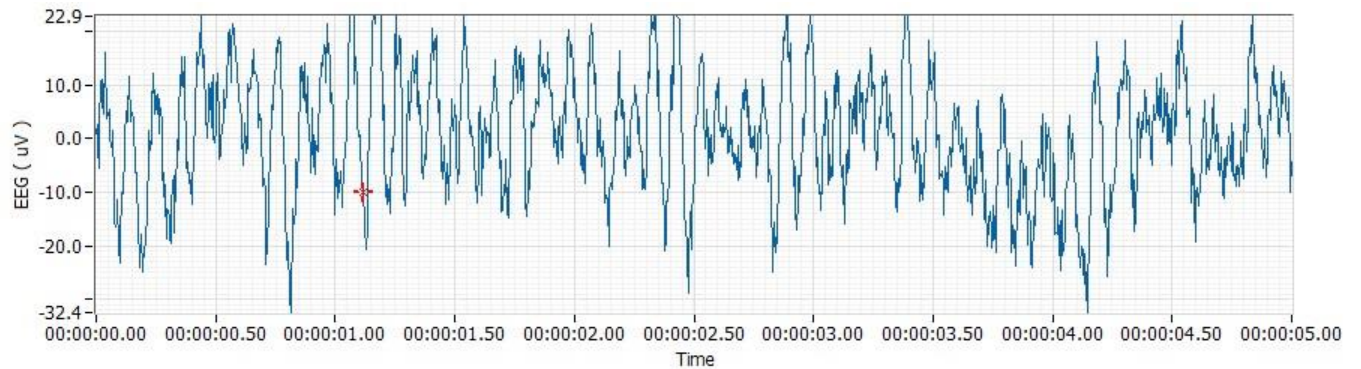
This interactive teaching module proved to be highly beneficial since it facilitated progressive learning of students by enhancing their understanding of clinical EEG parameters and their relationship with differential diagnosis of the patients. Previously, abnormal spike activity associated with epilepsy was mainly accomplished by converting into frequency domain using FFT algorithm and using the cursor position as shown in Figure 5.

---

#### Measurements Result

Channel Name	Cursor Position
EEG	( 00:00:01.11 , -10.11uV )

#### Plots



Cursor position: ( 00:00:01.11 , -10.11uV )

Figure 5. EEG Measurement Results and Plot

Students were introduced early only to object based programming with Python and C#, but also to basic concepts of discrete Fourier, fast Fourier Transforms, continuous and discrete time Wavelet Transforms, and MP algorithm. Having being exposed to this C#, python programming, MCMs, Wavelet transform, learning module has far reaching implications.

### Selective Application Highlights

Monte Carlo methods have also been used in Quantum Physics to determine neutron trajectories, simulate atomic clusters and nuclear cascades studies. When a particle collides with a nucleus, a nuclear cascade is produced. The particle's path after the collision determines whether or not a new subatomic particles, pions are created. Monte Carlo methods keep track of the colliding particle's path after the collision and determine the probability of the production of pions. Monte Carlo algorithms are useful because it is difficult and time-consuming to simulate the collisions. By running the algorithm enough times, the results are usually statistically significant, and help successfully calculate the probability of the production of nuclear cascades.

Monte Carlo methods are often used to evaluate integrals for computational electromagnetics. Monte Carlo methods overcome multidimensional complexity issues, lack of lack of molecular defined boundaries, and electron density ranges, Monte Carlo methods use random number generation and a probability distribution function to evaluate the volumes of molecules.

A major limitation inherent with the standard MCMs for computational Electromagnetics which the students came across in the beginning was that they permit single point calculations. Research of literature proposed using shrinking boundary and inscribed figure methods for whole field Monte Carlo computations. Later they found out that using Markov Chains for whole field computations is more efficient than the shrinking boundary method and the inscribed figure method.

### Monte Carlo Simulation and Risk Analysis

A standard Monte Carlo simulation, a software program samples a random value from each input distribution and runs the model using those values. After repeating the process a number of times (typically 100 to 10,000), it estimates probability distributions for the uncertain outputs of the model from the random sample of output values. The larger the sample size, the more accurate the estimation of the output distributions. Microsoft Excel and other spreadsheets do not support Monte Carlo simulation directly. But, there are a number of software products that are add-ins to Excel that let you perform Monte Carlo simulation.

The best known are Oracle Crystal Ball and Palisade Software's @Risk, which are neat products. What is different about Analytica is that Lumina designed Analytica from its inception to perform Monte Carlo simulation (and LHS methods), so, probabilistic analysis is kind of fully integrated into the product from the start. This gives Analytica's Monte Carlo features certain



advantages over spreadsheet add-ins, in terms of ease of use and speed of computation. You can define any variable, or any cell in an array, as a discrete or continuous distribution. You can view the probability distribution for any resulting variable as a set of probability bands (selected percentiles), as a probability density function, cumulative distribution function, or even view the underlying random sample.

## Language Comparison for Monte Carlo

When it comes to programming languages, a general notion shared by people is that Python, Ruby and JavaScript are interpreted languages. They are easy to learn and to use, but not fast enough for intensive calculations. Let us analyze which languages can compete with Python and MatLab <sup>11</sup> and may be well suited on a simple Monte-Carlo simulation of a forward start option under the Black model. To keep it simple we will only compare these other languages and compare the reported runtime for executing 16K simulations times a multiplier. A word of caution, this reported data was run on an old 2015 desktop and may only serve the purpose of relative comparison.

Multiplier	Scala	Julia	JuliaA	Dart	Rust
1	0.03	0.08	0.09	0.03	0.004
10	0.07	0.02	0.06	0.11	0.04
100	0.51	0.21	0.40	0.88	0.23
1000	4.11	2.07	4.17	8.04	2.01

Table 1. Reported Language Runtime for executing 16K simulations times a multiplier

With newer laptops using enhanced GPU design, these results are ten times faster than what is depicted above. It really depends on the field of study and varies from application to application.

Time did not permit us to explore applications of the Monte Carlo algorithms in the field of finance. Monte Carlo methods aid the analysis of financial instruments, portfolios, assets, various price paths and final option value computations. Since Monte Carlo methods work well with highly complex equations, their use becomes vital in the calculation of uncertain values, which then in turn help analyze the final value of the instrument or asset in question. A specific ‘Monte Carlo Option Model’ is used to evaluate future prices of options.

As stated earlier, Monte Carlo simulation is a useful tool for predicting future results by calculating a formula multiple times with different random inputs and not just limited to EEG analysis. A simple example is for example predicting future value by calculating a formula multiple times with different random inputs. This is a process which one can execute in Excel but it is not simple to do without some VBA or potentially expensive third party plugins. Using numpy and pandas to build a model and generate multiple potential results and analyze them is relatively straightforward. The other added benefit is that analysts can run many scenarios by changing the inputs and can move on to much more sophisticated models in the future if the needs arise.

## Feedback and Assessment

In general, parametric tests <sup>8</sup> have requirements about the nature or shape of the populations involved: nonparametric tests do not require that samples come from populations with normal distribution or any other particular distribution. Due to the nature of our diverse student population, a rank correlation method will be described. The rank correlation test, or Spearman's rank correlation, is a non parametric test that uses ranks of sample data consisting of matched pairs. It is used to test for an association between two variables, so the null and alternate hypotheses are as follows (where  $p_s$  denotes the rank correlation coefficient for the entire population):

$H_0$ : = 0 (There is correlation between the two variables.)

$H_a$ :  $\neq 0$  (There is no correlation between the two variables.)

The notation  $r_s$  will be used for the Spearman rank relation coefficient so as not to confuse it with the linear correlation coefficient  $r$ .

### Rank Correlation Procedure and Notation

$n$  = number of pairs of sampled data

$d$  = difference between the ranks for the two values within a pair

$r_s$  = rank correlation coefficient for sample paired data ( $r_s$  is a sample statistic)

$p_s$  = rank correlation coefficient for all the population data ( $p_s$  is a population parameter)

### Test Statistic:

No ties: After converting the data in each sample to ranks, if there are no ties among the ranks for the first variable and there are no ties among the ranks for the second variable, the exact value of the test statistic can be calculated using this formula <sup>10</sup>:

$$r_s = 1 - \frac{6 \sum d^2}{n(n^2 - 1)}$$

Ties: After converting the data in each sample to ranks, if either variable has ties among its ranks, the exact value of the test statistic can be found by using this formula:

$$r_s = \frac{n \sum xy - \sum x \sum y}{\sqrt{(n \sum x^2 - (\sum x)^2) * (n \sum y^2 - (\sum y)^2)}}$$

Table 2 depicted below includes Students Subject Areas Grade Results Ranked by Pre-Monte Carlo Test and Post-Monte Carlo Test scores. It includes the difference  $d$  and the squares of the differences  $d^2$ .

The value of the rank correlation coefficient is computed in order to determine whether there is a correlation between the rankings of the Pre-Monte Carlo Test scores and the rankings of the Post-Monte Carlo Test scores using a 0.05 significance level.

Subject Area	Pre-Monte Carlo	Post-Monte Carlo	Difference	
	Test Grade	Test Grade	d	d <sup>2</sup>
	Ranks	Ranks		
Logic and Design	1	1	0	0
Anatomy& Physiology	3	2	1	1
Bioinstrumentation I	2	3	1	1
Bioinstrumentation II	5	7	2	4
Medical Imaging	4	5	1	1
Telemedicine	7	6	1	1
Bioinformatics	6	4	2	4
Ethics	8	6	2	4
Total				16

Table 2: BMET Subject Areas Grade Results Ranked by Pre-Monte Carlo Test and Post-Monte Carlo Test

Following the procedure, the data in the Table 3, are in the form of ranks and the neither of the two variables has ties among ranks, so the exact value of the test statistic can be calculated as shown below using the equations above.

We use  $n = 8$  (for 8 pairs of data) and  $\sum d^2 = 16$  (as shown in Table 2) to get

$$\begin{aligned}
 r_s &= 1 - (6 \sum d^2) / n(n^2 - 1) = 1 - [ 6(16) / 8( 8^2 - 1 ) ] \\
 &= 1 - [96/504] \\
 &= 0.8095
 \end{aligned}$$

If  $n > 30$ , critical values are found by the following formula, where the value of  $z$  corresponds to the significance level (for example if  $\alpha = 0.05$ ,  $z = 1.96$ ).

$$r_s = \pm z / \sqrt{n - 1}$$

If  $n \leq 30$ , critical values are found by using <sup>9</sup> the Table for Critical values of Spearman's Rank Correlation.

For our case  $n = 8$ , so we determine that the critical values are  $\pm 0.738$  (based on  $\alpha = 0.05$  and  $n = 8$ ).

Because the absolute value of the test statistic  $r_s = 0.8095$  does exceed the positive critical value of 0.738, we reject the alternative hypothesis and conclude that there is a correlation

It has thus been demonstrated that there is significant correlation between the Pre-Monte Carlo Test Results and the rankings of the Post-Monte Carlo Test results. Subjects appear to better

learn the BMET course subjects and prep material by studying more and going through rigorous practice and testing.

We have offered the mini Monte Carlo Methods with Statistics crash course as a special BMET topic course that is one way new courses are piloted locally at our university campus. This hands-on course is instrumental in the progressive learning of the students by reviewing, relating and applying fundamentals of Matlab, Minitab, EXCEL and SPSS for experimental design, using the estimated regression equation for estimation and prediction, trend projection and time series analysis, forecasting of variance, and statistical control.

Although special topics are not evaluated in the same manner as standard session-long courses, feedback directly from students indicates that initial offering was well received.

A compilation of feedback was from eight junior students enrolled in the course for credit, and four sophomore students attended the lectures and discussions but did not receive the credit. The feedback from the student indicates that:

1. Students gained an appreciation of the Stochastic processes, Monte Carlo Methods statistical tools and their capabilities. Using Matlab, C#, Python, EXCEL, SPSS and R puts many concepts in numerical methods, statistics analysis and inference to practice.
2. Students gained an appreciation for the difficulties involved in developing nonlinear models, detecting outliers and residual analysis.
4. Students spent a significant amount of time on solving differential and integral equations, for finding eigen values, for inverting matrices, and particularly for evaluating multiple integrals.
5. Estimation and data analysis using python, scripting with R and Matlab programming assignments (presumably relative to their other course work) but the results were satisfying.
6. We did not receive any complaints about the level of effort required by, nor the time spent on the programming assignments, although there were issues early on.
7. Student's performance in the initial course offering and in the course of capstone projects was exceptionally high. This result was due to a biased sampling; the four juniors taking the special topic course initiated the effort, and the sophomores that attended regularly were invited by the instructor. We hope to see better understanding of basic principles and excellent performance in the future versions of the course.

## Conclusions

Statistics Literacy and critical thinking is necessary in today's world that is fascinated with numbers and data. Even if one is not responsible for conducting Monte Carlo simulations, one needs the basic understanding to properly use the information for making decisions. With proper guidance, monitoring, and diligent care, students were exposed early on scripting, discrete probability distributions, sampling distributions, statistical inference, design of experiments, and analysis of variance.

End of the course survey and diagnostic quizzes demonstrated the enhanced student understanding of application of Stochastic processes and MonteCarlo simulations which is again attributed to early exposure of Statistics, Matlab, Python scripting, C#, Java

Programming and the reinforcement of EKG, EMG and EEG component analysis as part of BMET (biomedical engineering technology) to which they had been exposed later on during their junior and senior years.

The authors wish to stress that this paper is no attempt to challenge previous clinical or diagnostic knowledge. It is hoped that the concepts covered in this paper will instigate future research and development in the application of stochastic processes and Monte Carlo Methods not just for EEG analysis but aid aspiring student researchers and physicians to optimize and ultimately provide more cost effective solutions in quantum physics, computational Electromagnetics, field of Finance and other potential medical diagnostics applications. Monte Carlo methods (MCMs) uses random sampling to define constraints on the value and then makes a sort of "best guess." The solution of a problem by this method is closer in spirit to physical experiments than to classical numerical techniques

## Bibliography

1. Muqri, M., Shakib, J., *A Taste of Java-Discrete and Fast Fourier Transforms*, American Society for Engineering Education, AC 2011-451.
2. Shakib, J., Muqri, M., *Leveraging the Power of Java in the Enterprise*, American Society for Engineering Education, AC 2010-1701.
3. Mallat, S., Zhang, Z., *Matching pursuit with time-frequency dictionaries*, IEEE Trans. Signal Process., 41, 1993, 3397-3415.
4. Rangayyan, R., *Biomedical Signals analysis. A case-study approach*, IEEE Press on Biomedical Signals. Calgary, Alberta, Canada, 2002
5. Feichtinger, H., Strohmer, T., *Gabor Analysis and algorithms: Theory and applications*, Editors. Birkhauser, Boston, 1998.
6. Blinowska, K., Durka, P., *The application of wavelet transform and matching pursuit to the time-varying EEG signals*, in Intelligent Engineering Systems Through Artificial Neural Networks, Editors, Dagli & Fernandez, volume 4, pp. 535-540, ASME Press, New York, 1994.
7. Deitel, H.M., Deitel, P.J., *Java How to program*, Prentice Hall, 2003.
8. Triola, T., F., *Essentials of Statistics*, Addison Wesley, 2007.
9. Monte Carlo Simulation and Python. Ref: <https://pythonprogramming.net/monte-carlo-simulator-python/>
10. Greenfield, L., Geyer, J., Carney, P., *Reading EEGs*, Lippincott, 2010.
11. Modern Programming Language for Monte-Carlo, April 2015. <https://chasethedevil.github.io/post/modern-programming-language-for-monte-carlo>
12. ZhijunGu, NI Biomedical Start up Kit 3.0, July 2010, <https://decibel.ni.com/content/docs/DOC-12646> .
13. Skill Modules, *Advanced Physical Diagnosis Learning and Teaching at the Bedside*, Department of Medicine, University of Washington, <http://depts.washington.edu/physdx.html>
14. Help Center Documentation. *Integrate MATLAB with External Programming Languages and Systems*. [https://www.mathworks.com/help/matlab/matlab\\_external/integrate-matlab-with-external-programming-languages-and-systems.html](https://www.mathworks.com/help/matlab/matlab_external/integrate-matlab-with-external-programming-languages-and-systems.html).