Solving Difficult Integrals with the Monte Carlo method.

Çağatay Kavas
Department of Electrical and
Electronics Engineering,
Ankara University
Ankara, TURKEY
cagataykavas2001@gmail.com

Bora KOPARAN
Department of Electrical and
Electronics Engineering,
Ankara University
Ankara, TURKEY
borakoparan829@gmail.com

Abstract*—Some analytical problems may be unsolvable and some may take too much time to find an exact solution that is why we need to apply Monte Carlo method. Applying random variables for solving a problem is known as Monte Carlo method. Monte Carlo Integration method brings a solution to a problem with the relation between area and probability. If we apply less iterations our accuracy of the solution will be more different than the real solution and if we apply many iterations our accuracy of the solution will be closer to the real solution. So, we compared the accuracies of the iterations to determine the number of iterations for similar problems and each iteration 10 times. Some models may require many random variables for example for a moving object in 3 dimensions x, y and z coordinates as well as velocities in x, y and z coordinates so it is better to apply Monte Carlo method in this problem. Accuracy of function is shown by one difference over time graph and two histogram graphs one containing values of function and other one containing difference over iteration.

***Keywords*— Monte Carlo method, random, area, probability, domain, integration, iteration, histogram.**

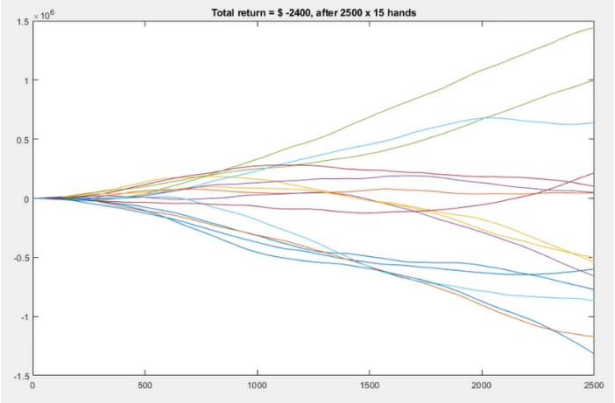
I. INTRODUCTION

In this project we estimated the area of a user entered function within user entered limits according to Monte Carlo integration method on Matlab. Monte Carlo method calculates approximate area by using the relation between area and probability. Code creates a rectangle around the function according to minimum and maximum values of the function between the user entered limits and other two sides are user entered limits. Code visualizes the rectangle around the function.

Our code creates random points in the area of rectangle. After placing that point in the rectangle there are three possible outcomes: point can be in the positive interval of the function, point can be in the negative interval of the function or point is out of the function interval. Code counts the number of points in all these three conditions. For visualization of positive and negative interval of the function y = 0 line is also plotted on the plot.

Area of the function is less than the area of the rectangle around the function and its intervals so some points placed on the rectangle may not be in the interval of the function. And we know that when integrated areas below the y = 0 are considered as negative area. By subtracting the dots in the negative function area from dots in positive function area and dividing the outcome of the subtraction to the number of total points we find a value which is approximately close to area of the function divided by area of the rectangle. Matlab counts the points and calculates the area of rectangle. By multiplying ratio of dots with the area of rectangle we can find an approximate value of the area of function. Area calculated by Monte Carlo method at final iteration is printed on the screen.

Here is the total money at players with different tactics hand made by Monte Carlo simulation of blackjack game.



[5]

This code compares the difference of real area and area we calculated from Monte Carlo method over iteration. Difference at final iteration and accuracy of that calculation at final iteration is also printed on the screen.

Aim of this project is to make a Monte Carlo algorithm to calculate hard integrals. This code is used to demonstrate the operation of Monte Carlo integration. We compared our results of the code for different iterations and for each operation we saved our results ten times to find the optimal iteration number.

There are some integration problems where solution may require complex integrals, may be too complex or even an exact solution is not needed at all but an approximate solution is enough. Trained our code for these conditions.

II. THEORITICAL BACKGROUND

Many times, when we solve a mathematical problem, we want to obtain an analytical solution to that particular problem. Usually, finding the analytical solution to a mathematical problem is the best way to approach the problem. However, sometimes, we can't solve the problem, because one it might take too time to come up with the analytical solution. Maybe sometimes it is just plain impossible to come up with the analytical solution to a particular problem. Many times, in order to solve mathematical problem, we have to go about it in a different way. Rather than finding approximate solutions to these particular problems, it is more convenient to find numerical solutions to these problems. Now these numerical solutions are the estimate of the actual solution to the mathematical problem. Many of the time these solutions themselves are very accurate enough and can be found in a much quicker time then trying to find the analytical solutions to those mathematical problems.

For different mathematical problems there are many different ways that we can approach to obtain a numerical solution. However, these different methods will have different degrees of accuracy and they take different amount of times in order to obtain a solution.

In this project, we will examine the "Monte Carlo Method". Monte Carlo simulations are named after the popular gambling destination in Monaco, since chance and random outcomes are central to the modeling technique, much as they are to games like roulette, dice, and slot machines. The technique was first developed by Stanislaw Ulam, a mathematician who worked on the Manhattan Project. After the war, while recovering from brain surgery, Ulam entertained himself by playing countless games of solitaire. He became interested in plotting the outcome of each of these games in order to observe their distribution and determine the probability of winning. After he shared his idea with John Von Neumann, the two collaborated to develop the Monte Carlo simulation. The "Monte Carlo Method" using in different areas in daily life. For example:

- Stock Exchange Models,
- Distribution Functions,
- Numerical Analysis,
- Simulation of natural events,
- Atomic and Molecular Physics.

The Monte Carlo method is a numerical method of solving mathematical problems by random sampling (or by the simulation of random variables). A Monte Carlo simulation uses repeated sampling to obtain the statistical properties of some phenomenon (or behavior).

The main idea behind this method is that a phenomenon is simulated multiple times on a computer using random-number generation based and the results are aggregated to provide statistical summaries associated to the phenomenon.

Monte Carlo methods vary, but tend to follow a particular pattern:

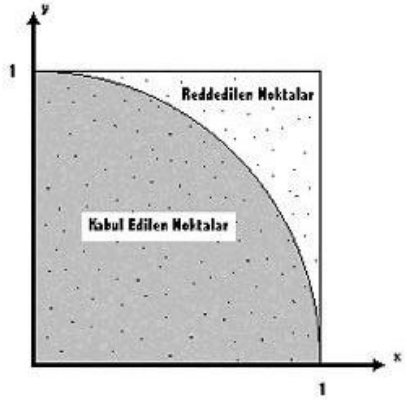
1. Define a domain of possible inputs
2. Generate inputs randomly from a probability distribution over the domain
3. Perform a deterministic computation on the inputs
4. Analyze the results.

If we use the Monte Carlo method to solve a mathematical problem, we can use 4 different techniques. These techniques are:

- **Acceptance-Rejection Method**
- **Averaging Method**
- **Control Variates Method**
- **Importance Sampling Method**

Acceptance-Rejection method

"When this method is desired to be applied, first of all, it is tried to find a solution from the random values obtained by using a selected distribution function. The selected random points are accepted if they are below the curve of the f(x) function, and if they are above it, they are rejected, and it is calculated how many of the total trials yielded successful results. As a result of the operations, the ratio of the total number of attempts to the number of successful attempts gives the ratio of the total area to the area of the f(x) function to be integrated. However, this method is not suitable for functions with one or more sharp vertices. In addition, if the region boundaries on the analytical plane are determined incorrectly in the rejection method, the rejected points cause a waste of time." [4]



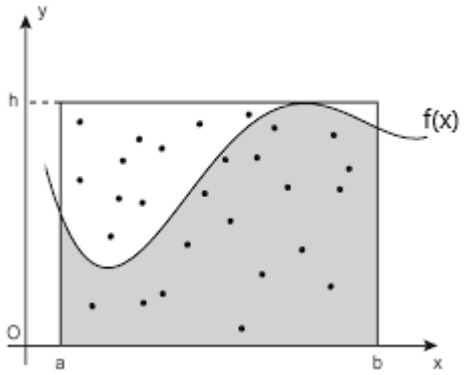
[4]

In this Project, we are going to use the acceptance-rejection method.

When this method is intended to be applied, it is first tried to resolve random values obtained using a selected distribution function.

The selected random points are accepted if they fall below the curve of the function f(x), if they are above this curve, and how many of the total attempts are successful.

As a result of the operations performed, the ratio of the total number of attempts to the number of successful attempts gives the ratio of the total field to the field of the f(x) function to which the integral is taken.



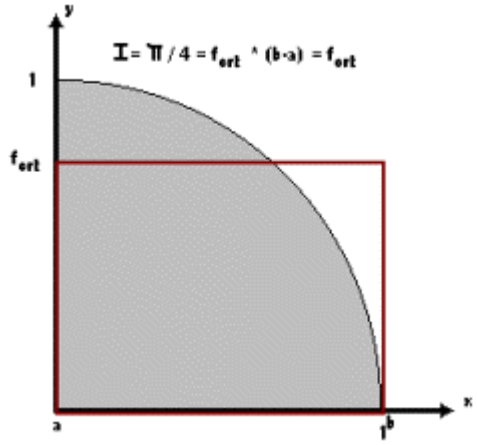
$$\textbf{Probability that point lands under the curve} = \frac{\textbf{Area Under Graph}}{\textbf{Area of Box}}$$

$$\frac{\textbf{Points Under Graph}}{\textbf{Points Generated}} = \frac{\textbf{Area Under Graph}}{\textbf{Area of Box}}$$

$$\int_a^b f(x) * dx = \textbf{Area of Box} * \frac{\textbf{Points Under Graph}}{\textbf{Points Generated}}$$

Averaging Method

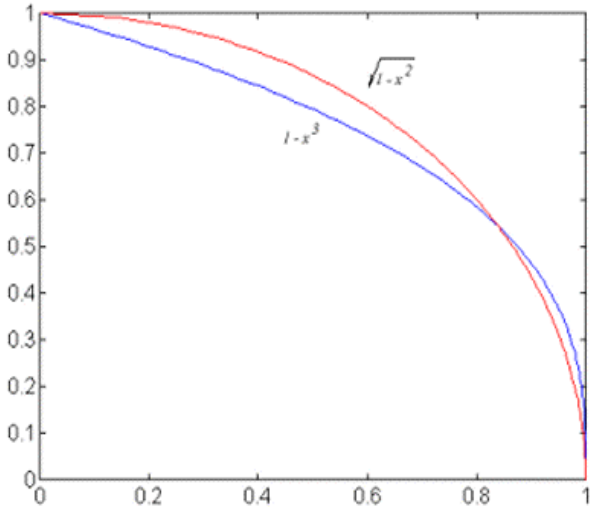
"The averaging method also operates on randomly selected numbers/points. Unlike the reject method, instead of scanning the area, it tries to find the result based on the value of the function at the selected points." [4]



[4

Control Variates Method

"In this method, an auxiliary function h(x) is used, which is very close to the f(x) function to be integrated. The integral value of this auxiliary function can be calculated more easily than our function f(x) or should be a function whose solution is known. Thus, at each random point chosen, the difference between the integral values of the two functions is tried to be found. The obtained difference value is added to the integral value of the h(x) function and the f(x) function is tried to be reached." [4]



Importance Sampling Method

"This method is quite similar to the control variable method. Unlike the control method, the effect of the auxiliary function h(x) is multiplicative, not additive. As in the control variable method, the accuracy of the selected h(x) function reduces the error rate. The obtained difference value is multiplied by the integral value of the h(x) function and the f(x) function is tried to be reached." [4]

Accuracy Parameters

Iteration number, function, different run for same iteration number are the parameters that effect the accuracy of our simulation.

The difference between the real integral value and the Monte Carlo integral is shown as subtraction of two individual vectors one containing real values and other one containing Monte Carlo values.

Histogram is the frequency domain representation of the values. Normally we expect the difference to reduce and Monte Carlo calculation result to get closer to real integral as iteration number increases. In histogram graph of Monte Carlo integration value, we expect more values to be closer to the real value of integration. In histogram of difference over time we expect more values to be closer to zero because difference over time will decrease over iteration.

III. Description of Simulation

Syms function is used to create symbolic scalar variables, functions, and matrix variables. integratedFunction(x) is user entered function and x is its variable. Code asks the user to enter a function then its lower and higher integration limits. maxMinCalcInt is used to load the f(x) values of the entered function into an array in 0.01 increments. According to the function and entered limits of integration minimum and maximum values of the

function is calculated by min and max functions. Entered function and the rectangle around is visualized. User enters the iteration number they wish to examine. We compared outputs of each three different iterations number ten times which is later on described in results.

"area" is calculated with integration (int) function with variable as x and according to user entered limits to compare the difference between calculated with function and measured with Monte Carlo integration method over iteration at the end of the code. Iteration is the total number of points placed on the function.

Rectangle area is calculated because it multiplied with ratio of the dots in integration region over total dots to calculate the measurement from Monte Carlo integration. Also the 0 line is printed to visualize the integration better since the areas under y=0 is considered as negative area in integration.

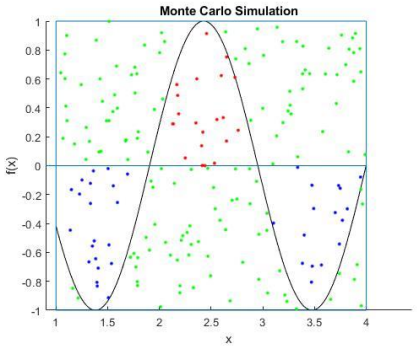
pointsUnder and totalDots is defined. They are points under iteration number and number of total points which is used in measurement of area with Monte Carlo method.

For loop continues until user entered iteration number is reached. With rand function x and y values between x and y limits are created. There are three possible outcomes.

Random point can be in positive interval, negative interval or out of integral. If point is in positive integral number of dots in interval increases.

Code selects two random points according to intervals of x and y values each representing x and y coordinates of the point. If the x coordinate of selected point is smaller than the y coordinate when applied in the function entered by the user and y coordinate of the selected point is greater than 0 the point is in positive interval. If the x coordinate of selected point is greater than the y coordinate when applied in the function entered by the user and y coordinate of the selected point is smaller than 0 the point is in positive interval. Otherwise that point is out of interval

If point is in negative integral number of dots in interval decreases. If the point is out of interval than number of dots do not change. However, in each three of outcomes the point is represented by a different color (red if in positive interval, blue if in negative interval, green if out of interval) and plotted at their location.



Pause function is applied so that user can see points being plotted. Number of total points increase as iteration number increases. differenceOfValues is an array consisting of the difference of calculated area by integration and measured area by Monte Carlo integration.

After for loop ends the measured area value at final iteration is printed on the screen and user can observe it as long as they please until user enters "1" on screen.

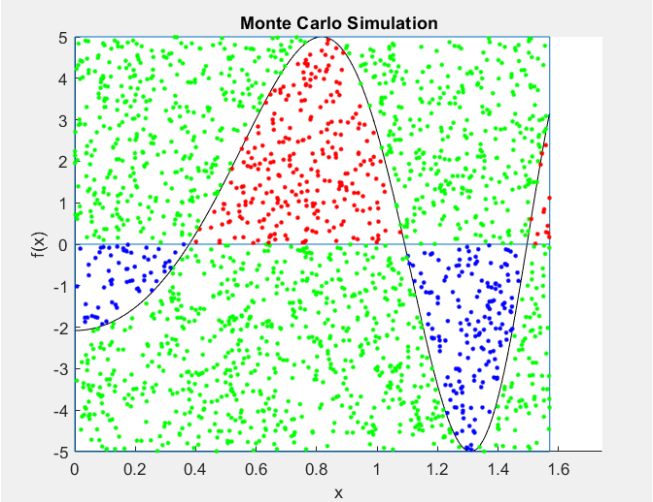
For calculating accuracy, we used 3 parameters. Difference of real area and calculated area over iteration, histogram of calculated area, histogram of difference of real area and calculated area over iteration

Difference over iteration is plotted on the screen. Then the final difference and error percentage of final difference are printed as well.

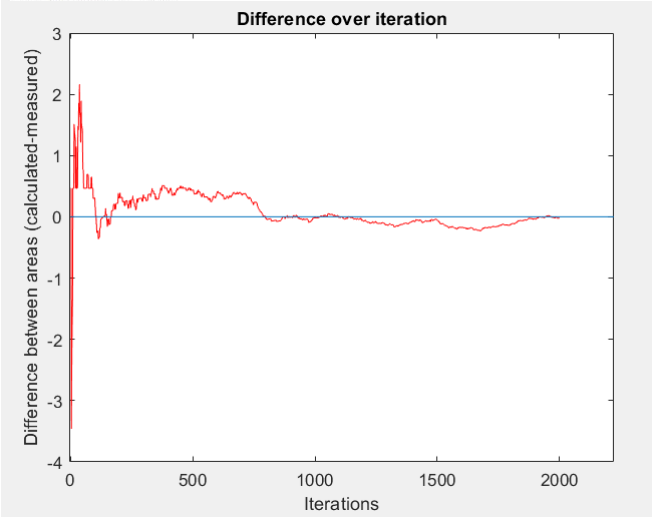
IV. Simulational Results

We compared 3 different function's integration with Monte Carlo method in MATLAB.

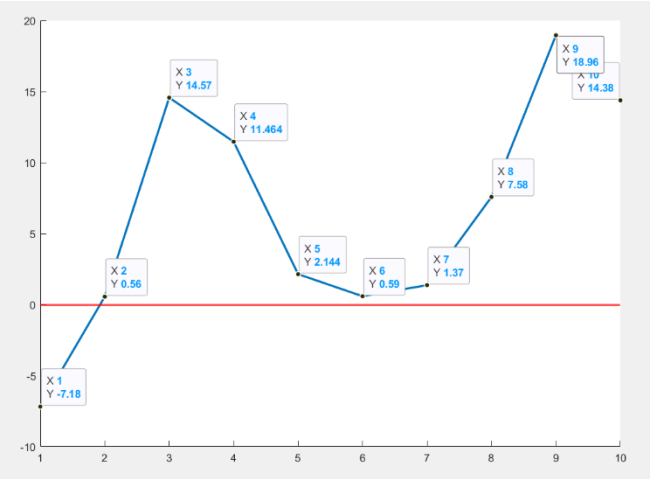
→ For the function : $\int_0^{\pi/2} 5 * \cos(3 * x^2 - 2)$



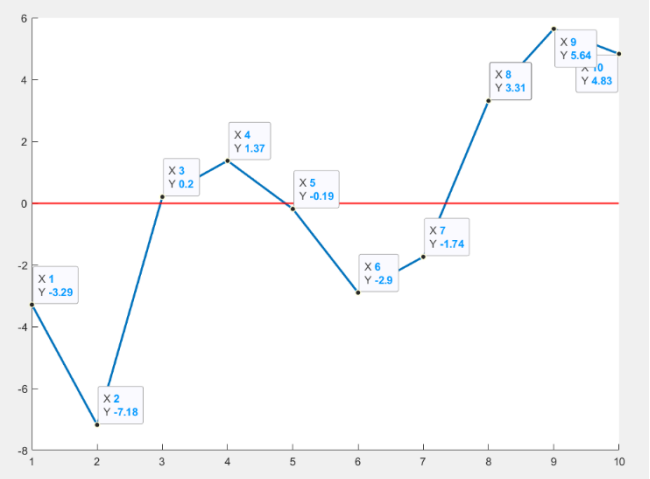

```
Please enter the function you want to integrate. (y=f(x) x as variable) : 5*cos(3*x^2-2)
Please enter a integration lower limit : 0
Please enter a integration upper limit : pi/2
Please enter the iteration number : 2000
Red dots are in positive interval, blue dots are in negative interval and green dots are out of interval.
The answer of integration with using Monte Carlo Method is 0.4869
Press 1 to see difference over time : 1
Answer of the integral is: 0.466
Final difference is: -0.021
Error percentage is: -4.420
```



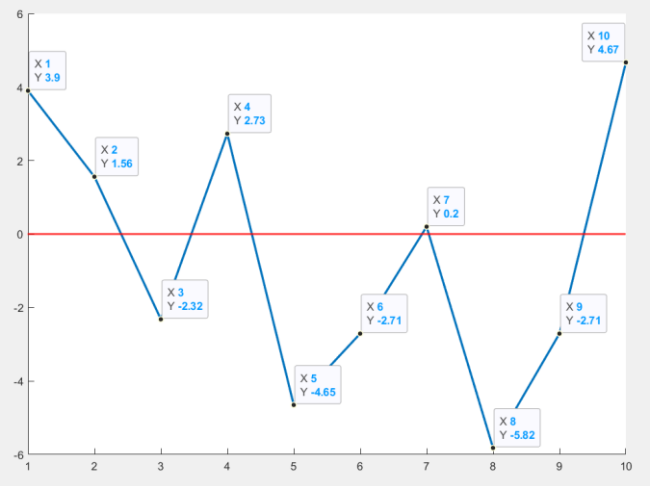
- 500 Iteration



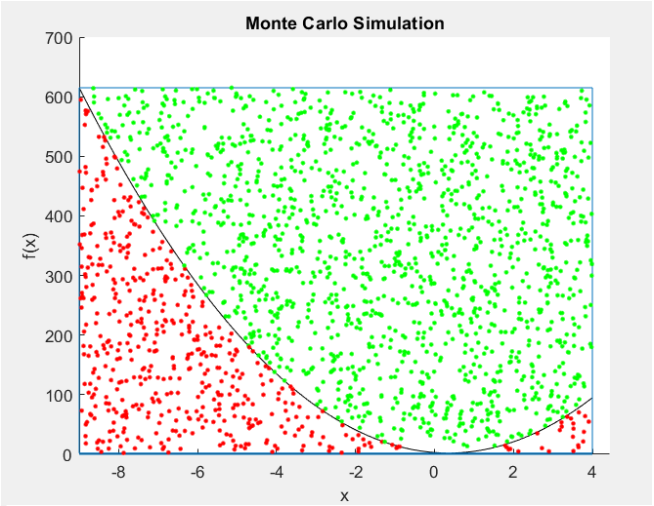
♦ 1000 Iteration



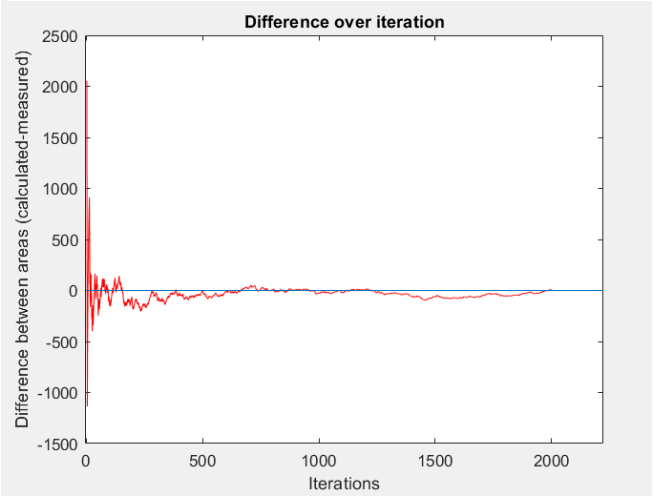
♦ 2000 Iteration



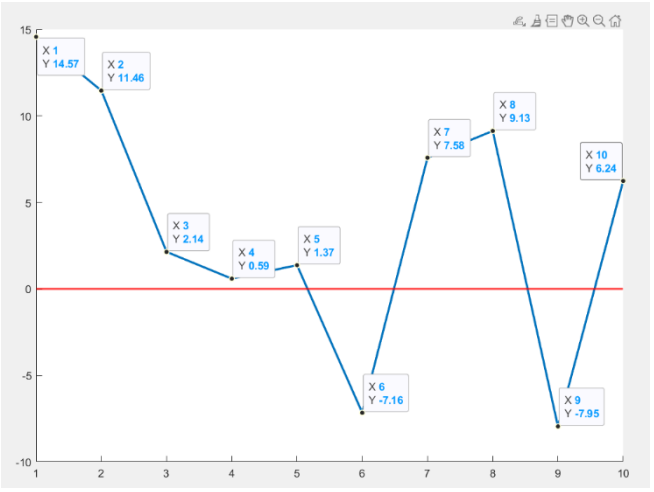
→ For the function : $\int_{-9}^{4} 7 * x^2 - 5 * x + 3$



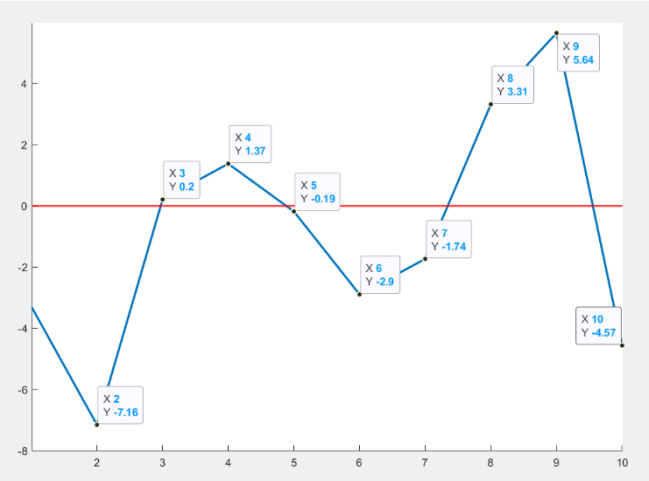

```
Please enter the function you want to integrate. (y=f(x) x as variable) : 7*x^2-5*x+3
Please enter a integration lower limit : -9
Please enter a integration upper limit : 4
Please enter the iteration number : 2000
Red dots are in positive interval, blue dots are in negative interval and green dots are out of interval.
The answer of integration with using Monte Carlo Method is 2047.6748
Press 1 to see difference over time : 1
Answer of the integral is: 2051.833
Final difference is: 4.158
Error percentage is: 0.203
```



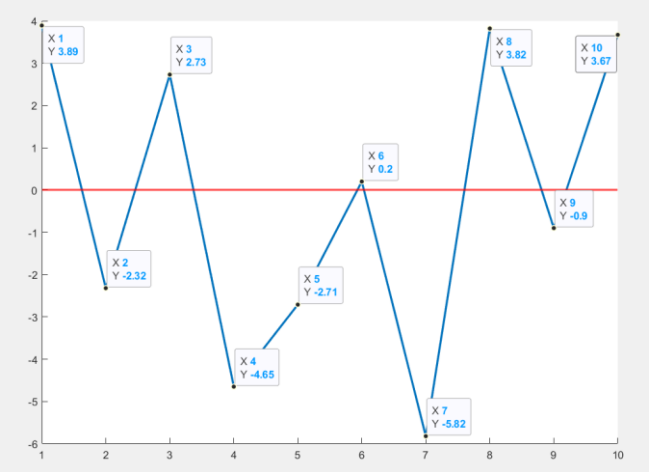
♦ 500 Iteration



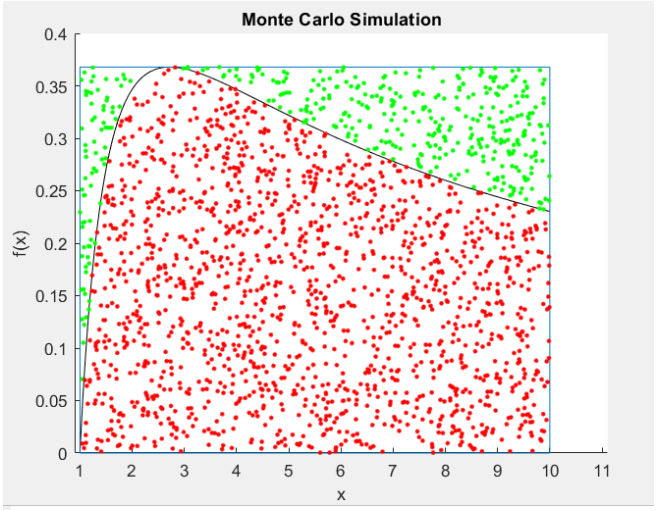
♦ 1000 Iteration



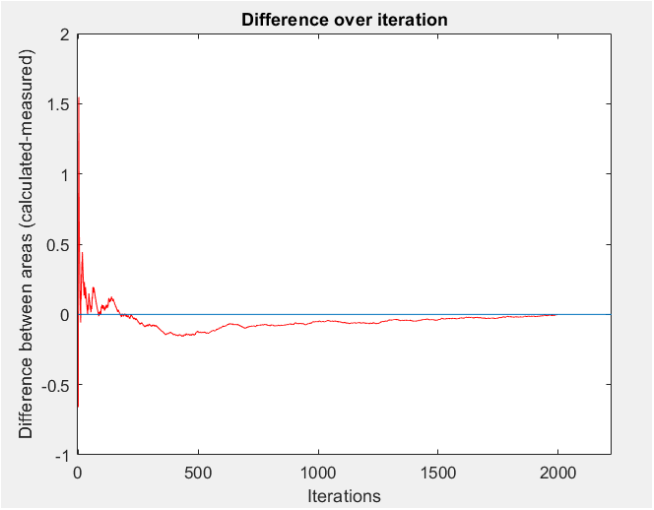
♦ 2000 Iteration



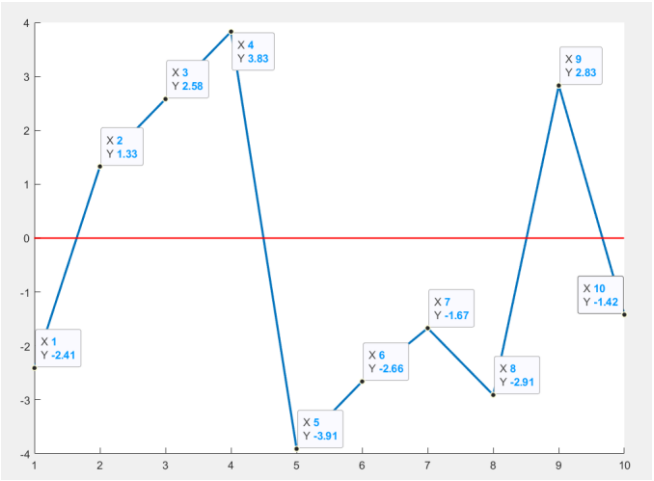
→ For the function : $\int_{1}^{10} \frac{ln(x)}{x}$



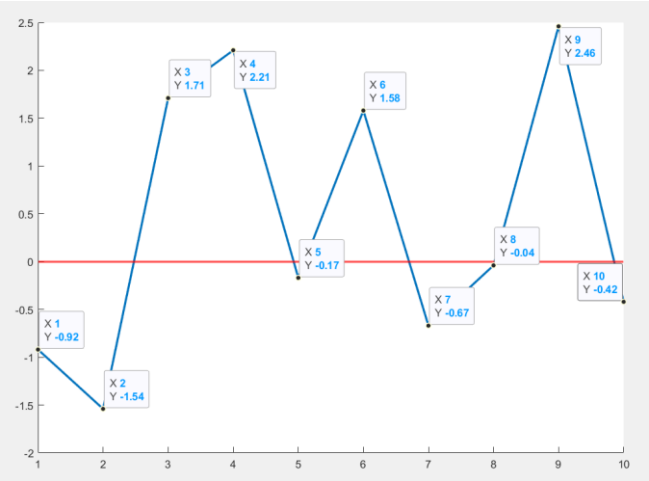

```
Please enter the function you want to integrate. (y=f(x) x as variable) : log(x)/x
Please enter a integration lower limit : 1
Please enter a integration upper limit : 10
Please enter the iteration number : 2000
Red dots are in positive interval, blue dots are in negative interval and green dots are out of interval.
The answer of integration with using Monte Carlo Method is 2.6520
Press 1 to see difference over time : 1
Answer of the integral is: 2.651
Final difference is: -0.001
Error percentage is: -0.041
```

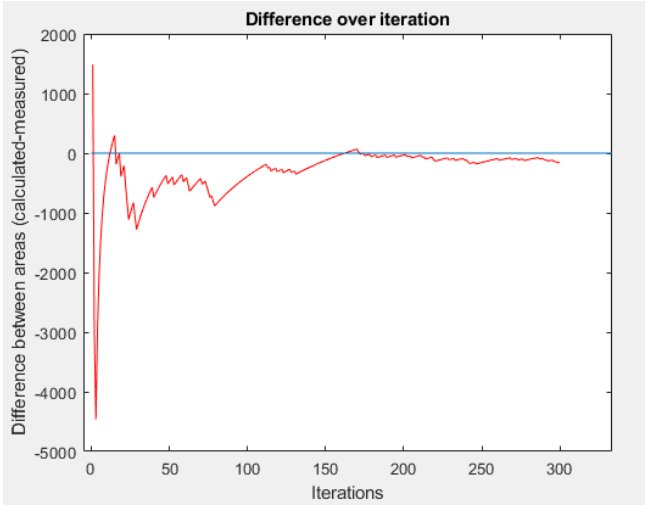


♦ 500 Iteration

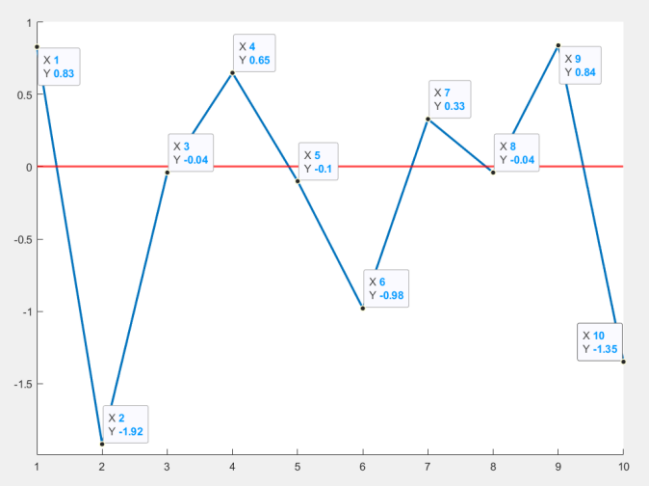


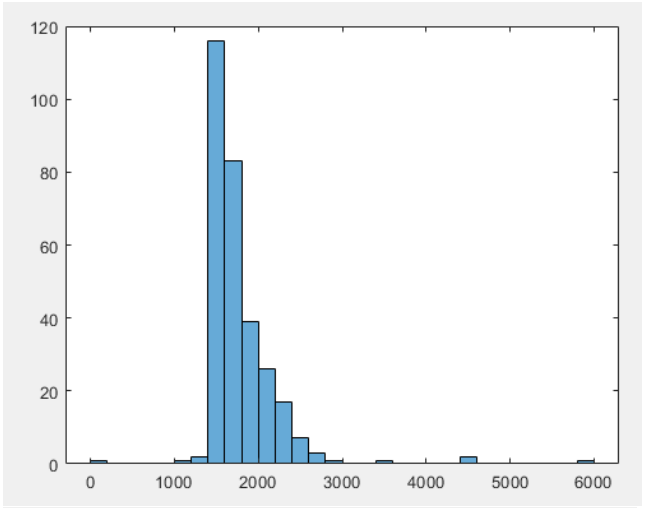
♦ 1000 Iteration





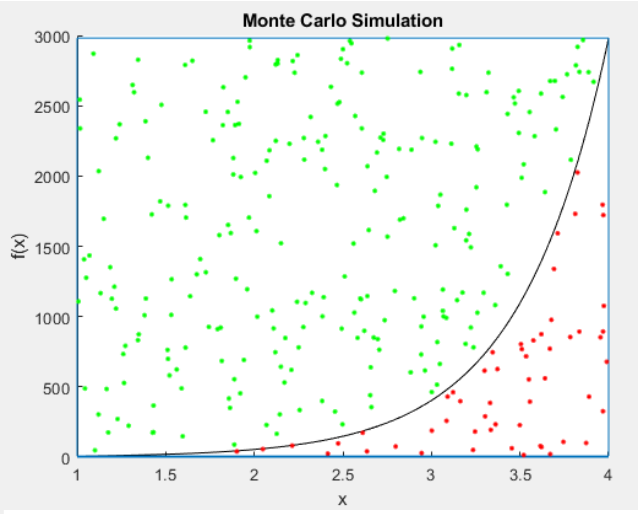
♦ 2000 Iteration

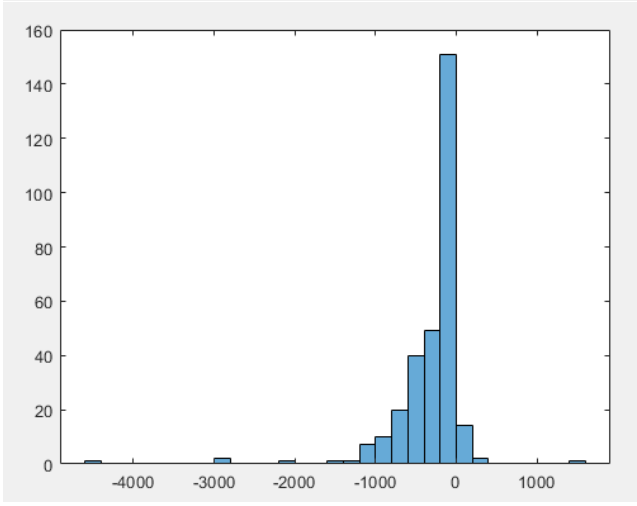




→ For the function : $\int_1^4 e^{2*x}$

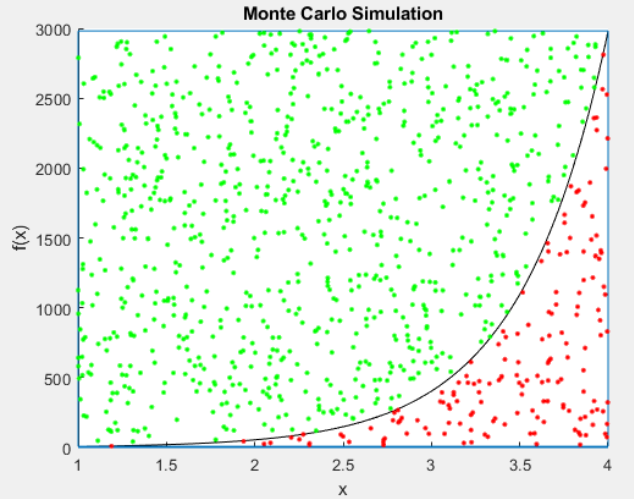
♦ 300 Iteration

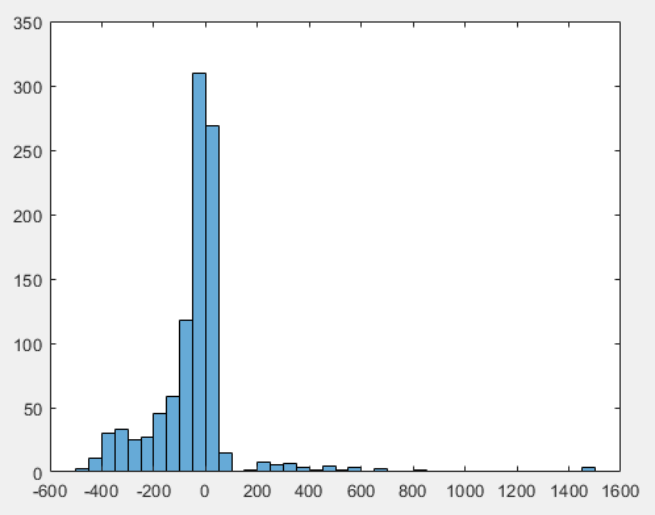




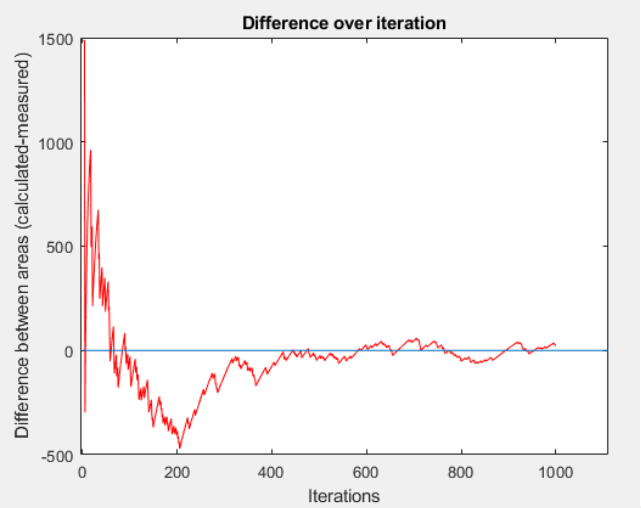
```
Please enter a integration upper limit : 4
Please enter the iteration number : 300
Red dots are in positive interval, blue dots are in negative interval and green dots are out of interval.
The answer of integration with using Monte Carlo Method is 1635.4629
Press 1 to see difference over time : 1
Final difference is: -148.678
Error percentage is: -10.000
```

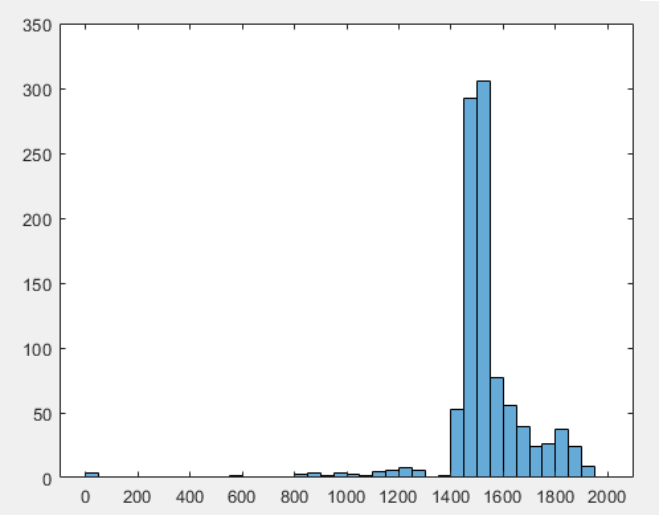
♦ 1000 Iteration



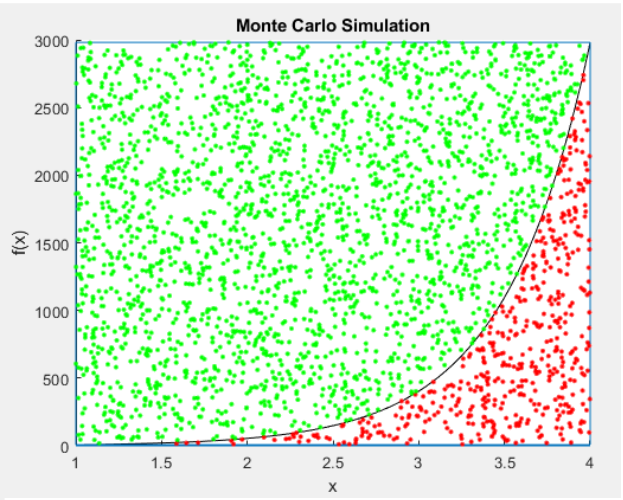


```
Please enter a integration upper limit : 4
Please enter the iteration number : 1000
Red dots are in positive interval, blue dots are in negative interval and green dots are out of interval.
The answer of integration with using Monte Carlo Method is 1462.9959
Press 1 to see difference over time : 1
Final difference is: 23.789
Error percentage is: 1.600
Press 1 to see the histogram of difference over time :
```

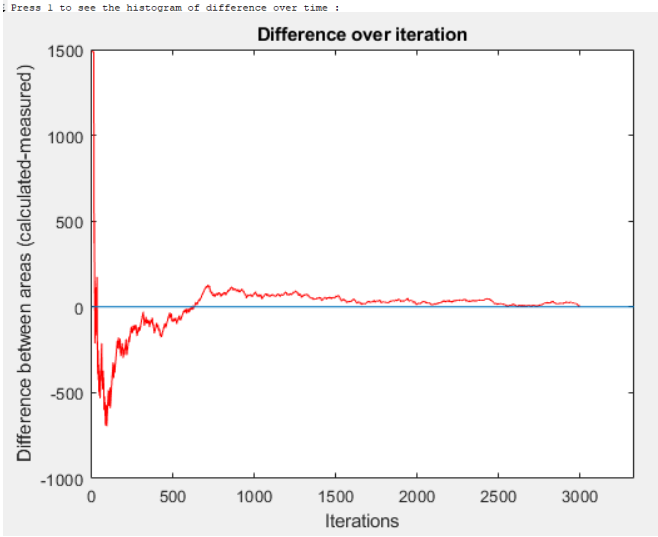


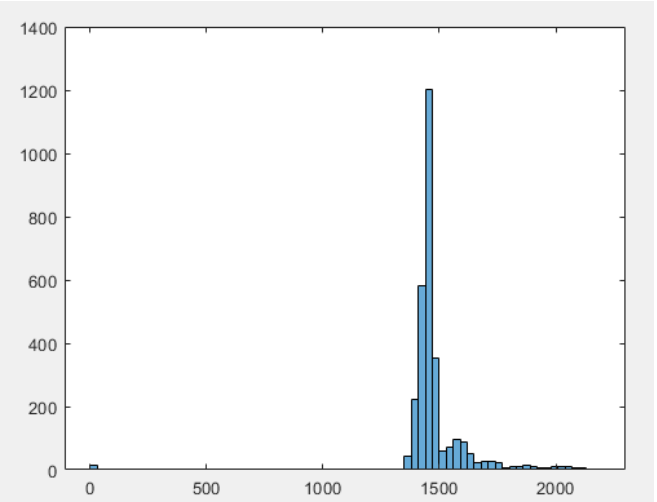


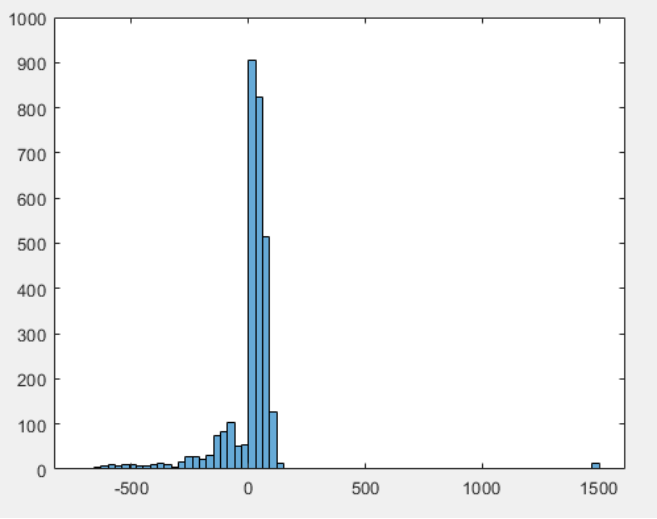
♦ 3000 Iteration




```
Please enter a integration upper limit : 4
Please enter the iteration number : 3000
Red dots are in positive interval, blue dots are in negative interval and green dots are out of interval.
The answer of integration with using Monte Carlo Method is 1477.8638
Press 1 to see difference over time : 1
Final difference is: 8.921
Error percentage is: 0.600
Press 1 to see the histogram of difference over time :
```







V. Conclusion

As iteration number increases we observed that area calculated by Monte Carlo integration method got closer to the actual area value for each function. Difference of accuracies of function are closer to each other and the real area at high iteration numbers. That means as iteration number increases our answer will converge into the real area and also answers won't change much.

Histogram graphs showed us that as iteration number increases difference over iteration reduces and the function values are closer to the real area.

References

[1] RandomMathsInc, The Monte Carlo Method, youtube.com, https://www.youtube.com/watch?v=q6gJ2T0NSwM&t=622s

[2] https://www.investopedia.com/terms/m/montecarlosimulation.asp

[3] https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/monte-carlo-methods-in-practice/monte-carlo-methods

[4] FIRAT ÜNİVERSİTESİ TEKNOLOJİ FAKÜLTESİ, MONTE CARLO SİMÜLASYONU ve Pİ SAYISININ TAHMİNİ, Ahmet Can ÇAKIL, Ali Murat GARİPCAN, Özgür AYDIN, Şahin KARA

[5] https://bookdown.org/manuele_leonelli/SimBook/what-does-monte-carlo-simulation-mean.html

[6] Cleve Moler (2021). Blackjack https://www.mathworks.com/matlabcentral/fileexchange/4404-blackjack, MATLAB Central File Exchange. Retrieved December 23, 2021.