The estimation of Pi using Monte Carlo technique with interactive animations

3 authors:

Svetlana Strbac-Savic
Visoka škola elektrotehnike i računarstva strukovnih studija u Beogradu

Ana Miletic
Academy of Technical and Art Applied Studies School of Electrical and Computer …

Hana Zoran Stefanović
Visoka škola elektrotehnike i računarstva strukovnih studija u Beogradu

THE ESTIMATION OF PI USING MONTE CARLO TECHNIQUE WITH INTERACTIVE ANIMATIONS

Svetlana Strbac-Savic[1], MSc; Ana Miletic[1], MSc; Hana Stefanovic[1], PhD
[1] School of Electrical and Computer Engineering of Applied Studies, Belgrade, SERBIA,
svetlanas@viser.edu.rs, anam@viser.edu.rs, stefanovic.hana@yahoo.com

Summary: *In this paper some basic concepts of the Monte Carlo technique for estimating the value of a parameter are given, including some important issues of confidence intervals and convergence. A simple Monte Carlo estimator is proposed, in order to approximate the value of Pi. The random experiment is defined as taking random samples over the bounding box, after defining the event of interest, while the estimation is based on different numbers of replications of the underlying experiment. The uniform random number generator included in the MATLAB library is used to generate random points in a square around a circle of radius 1, while the points falling within the inscribed circle are counted. The probability of data point landing within the circle presents a ratio of the circle's area to the area of the square, which is used for estimation the value of Pi. Some interactive animations, realized using Scratch programming language, are also provided.*

Keywords: *approximate value of Pi, event of interest, Monte Carlo simulations, random experiment, random number generator*

1. INTRODUCTION

Monte Carlo simulations are based on games of chance [1], which is the reason for using the name of the Mediterranean city famous for casino gambling. Monte Carlo technique describes a simulation in which a system parameter value is estimated by performing an underlying stochastic or random experiment [2]. Monte Carlo simulations are used to model financial systems, to simulate telecommunication networks, to compute results for high-dimensional integrals or to approximate values such as constants or numeric integrals [3, 4]. Monte Carlo technique is also used in modeling a wide range of physical systems at the forefront of scientific research today, based on a series of random numbers [5].
Strictly speaking, Monte Carlo estimation is based on the relative frequency interpretation of probability [6, 7]. In defining relative frequency, the first step is specifying a random experiment and the event of interest, A. In random experiments the result (outcome) can not be predicted exactly, but it can be defined statistically [8]. The most basic experiment is flipping a coin, with outcomes defined by set {Heads, Tails}, with equal probability of each outcome, if experiments are independent. A coin-tossing experiment modeling the binary transmission in digital communication system is illustrated in [8-10] showing the estimator is unbiased and consistent.
After defining a random experiment and an event of interest, the random experiment is executed a large number of times, N, and the number of occurrences, N_A, is counted. The number of occurrences N_A corresponds to an event A, while the probability of event A is approximated by the relative frequency of the event defined by N_A/N and obtained by replicating the random experiment an infinite number of times. Using a finite number of experiments N, gives an estimator of $Pr(A)$, which is also a random variable. The statistics of this random variable determine the accuracy of the estimator and the quality of estimation [8, 11].
In this paper the value of Pi is estimated using Monte Carlo simulation [8, 12]. Pi is one of the fundamental constants of mathematics [13], presenting the ratio of a circle's circumference to its diameter. Pi is an irrational number, meaning it can never be written as a fraction of two whole numbers, and its decimal expansion therefore does not terminate or repeat. The first 40 places are: 3.14159 26535 89793 23846 26433 83279 50288 41971...
More than 12 trillion digits of Pi are counted using the Chudnovsky formula [14], which was achieved in 94 days, needing a lot of disk space and storage capacity [15].
An approximate value 22/7 is only good for 2 places after the decimal point, since the difference between Pi and 22/7 is 0.00126…≈1/791. An approximate value 333/106 gives exact value of Pi till 4 places after the decimal point, while an approximate 355/113 is good for 5 places after the decimal point.
The decimal expansion of Pi goes on forever, never showing any repeating pattern. Since Pi is irrational, it is sometimes helpful to have good fractional approximations to Pi or simple estimation of Pi, as it is illustrated in this paper.
In order to estimate the value of Pi using Monte Carlo simulation it is necessary to determine the region whose area is to be estimated, and which is completely bounded by a box of known area. The random experiment is defined as taking random sample point over the bounding box, while the event A presents a sample falls within the region whose area is to be determined. For an unbiased estimator of known area it is assumed that the random sample points are uniformly distributed within the bounding region of known area [8].
In this paper, the approximate value of Pi is estimated by generating random points in a square around a circle of radius 1, and then using the relationship between the area of the square and the circle. Random points are generated using uniform random number generator in MATLAB command language [11]. Many other programming languages useful for developing simulation process also contain random number generator as a part of the library of "built-in" functions.

Strictly speaking, these random number generators do not generate random numbers, but they produce sequences that appear random over the observation interval, so they can be used to approximate a sample function of a random process for given simulation. Such sequences are referred as pseudo-random sequences because, even deterministic, they appear random when used for a given application. A number of procedures have been developed for testing the randomness of a given sequence, like Chi-square test, the Kolmogorov-Smirnov test and the spectral test [16-18].

In this paper, the convergence properties of estimator of Pi are analyzed using different numbers of sample points during simulation process, and after that several independent approximations are averaged. Some interactive animations during the simulation process, realized using Scratch programming language, are also provided.

2. DEFINING THE EVENT OF INTEREST

In order to estimate the value of Pi using Monte Carlo simulation the random experiment is defined as taking random sample point over the bounding box, while the event *A* presents a sample falls within the region whose area is to be determined.

For estimating the value of Pi, a pie-shaped area corresponding to the first quadrant of a circle is bounded, as it is illustrated in Fig.1. The first quadrant of a circle with radius R=1 is bounded by a box of unit area, as presented in Fig.1.

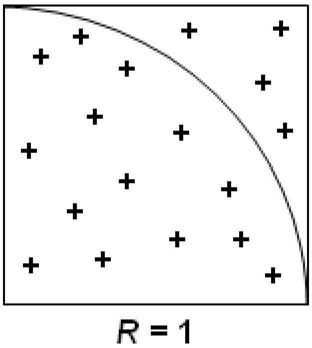

Figure 1: A pie-shaped area corresponding to the first quadrant of a circle with radius R=1 bounded by a box of unit area

The bounding box area is $A_{box}=R^2=1$, with N_{box} total sample points, while the area of interest is a pie-shaped region, presenting a quarter circle, defined with $A_{pie}=1/4*R^2*\text{Pi}=\text{Pi}/4$.

Since the sample points are uniformly distributed within the bounding box, the ratio of the area of interest to the area of the bounding box A_{pie}/A_{box} is approximately equal to the ratio of the number of sample points falling in the area of interest to the number of points falling in the bounding box:

$$\frac{A_{pie}}{A_{box}} \approx \frac{N_{pie}}{N_{box}} \tag{1}$$

Assuming that the samples are uniformly distributed, the ratio of N_{pie} to N_{box} will constitute an unbiased and consistent estimator of A_{pie}/A_{box}. From (1) it is obtained:

$$\frac{N_{pie}}{N_{box}} \approx \frac{A_{pie}}{A_{box}} = \frac{\pi}{4} \tag{2}$$

The estimator of Pi is:

$$\hat{\pi} = \frac{4N_{pie}}{N_{box}} \tag{3}$$

It can be concluded that the value of Pi can be estimated by covering the bounding box with uniformly distributed points, counting the points falling within the inscribed circle, and applying (3).

3. IMPLEMENTATION OF MONTE CARLO TECHNIQUE IN MATLAB

MATLAB command language is used to demonstrate the estimation of Pi based on previously described random experiment. The uniform random number generator rand(.) included in the MATLAB library [8, 11] is used. This number generator, which is targeted at the generation of floating-point numbers rather scaled integers, is briefly described in [19].

The results of MATLAB program implementing the described procedure are shown in Fig.2 for different number of sample points. The Monte Carlo estimator for the value of Pi using 500, 1000, 5000 and 10000 sample points is illustrated in Fig.2 a), b), c) and d) respectively.

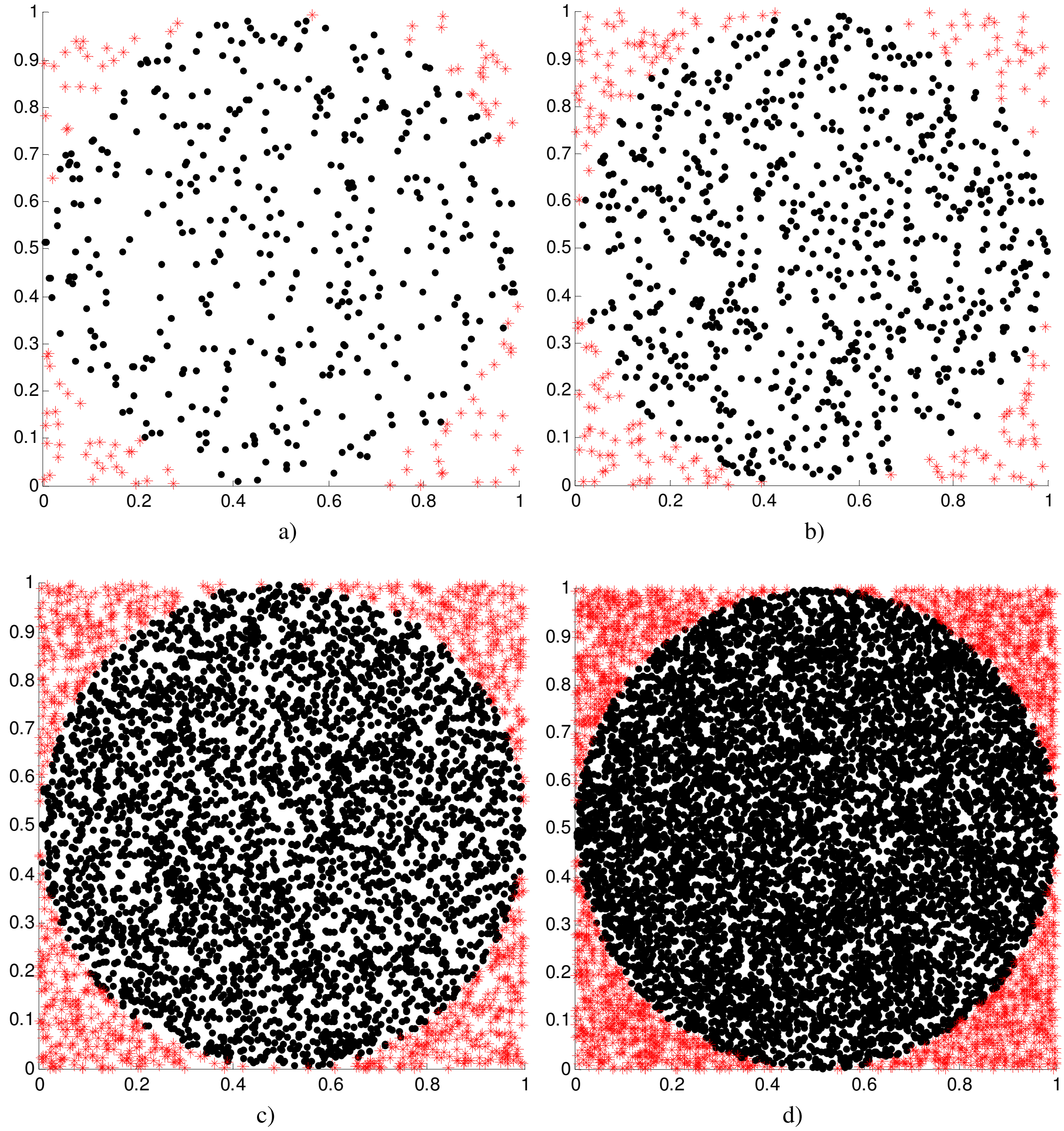

Figure 2: The estimation of Pi using different number of sample points: a) 500, b) 1000, c) 5000 and d) 10000 generated sample points

For 500 sample points the estimation value is 3.1680, for 1000 points it is 3.2240, for 5000 points it is 3.1280, while for 10000 the estimation has the value 3.1496.

It can be concluded that the approximation improves as the number of sample points increase.

The Monte Carlo simulation is then performed a number of times resulting in a collection of estimates of the random variable of interest. An average of 3 trials of the estimation of Pi after 500, 1000, 5000 and 50000 is also calculated in this section. The average versus number of trials is illustrated in Fig.3 a), b), c) and d) for 500, 1000, 5000 and 10000 sample points, respectively.

It can be concluded that using more points or averaging several approximations gives a better results, as it is illustrated in Fig.5.

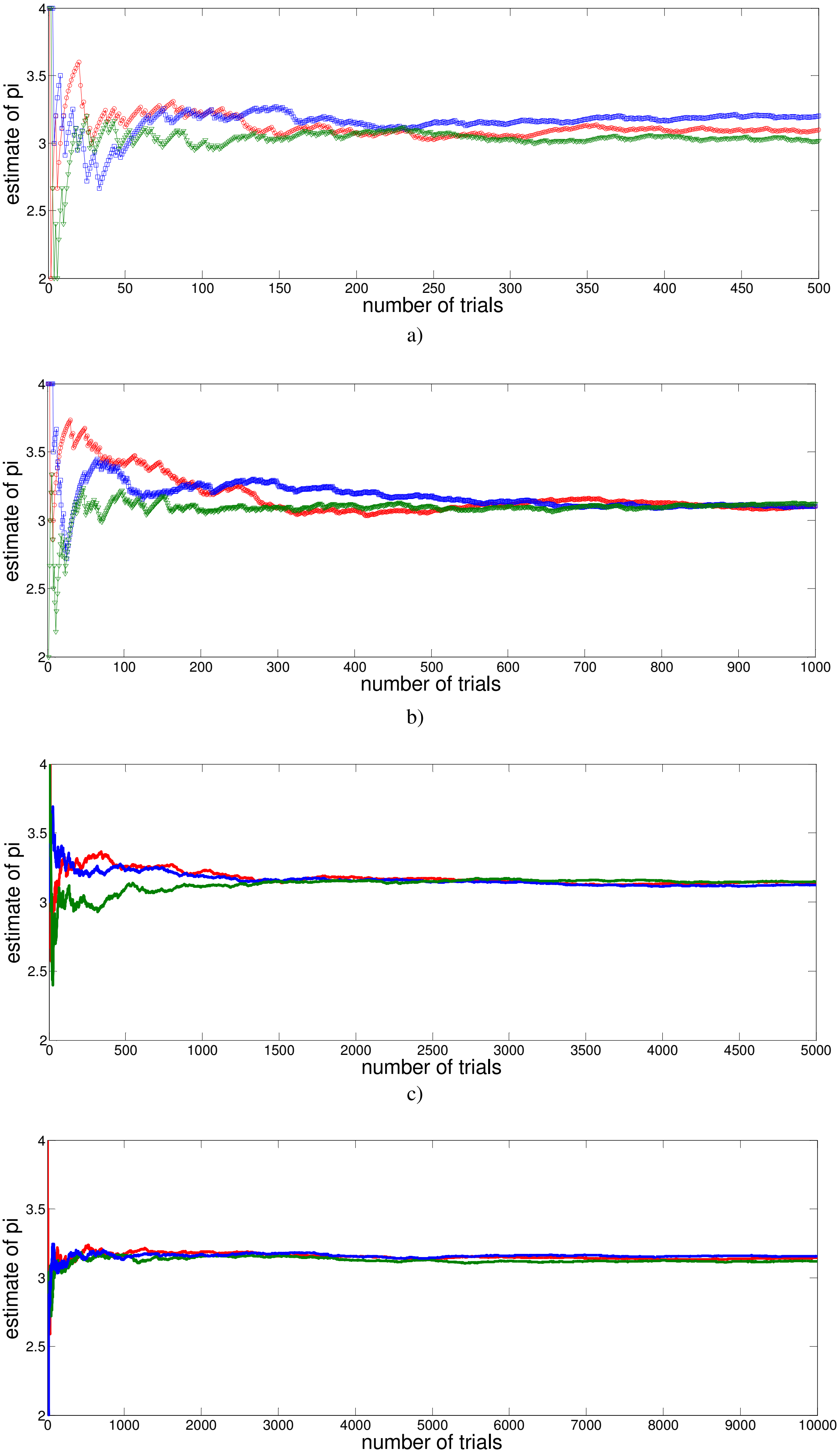

Figure 3: An average of 3 trials for different number of sample points

4. REALIZATION OF INTERACTIVE ANIMATIONS IN SCRATCH

Scratch is a programming language for creating interactive stories, animations and games, developed by the Lifelong Kindergarten Research group at the MIT Media Lab [20].
Simple algorithm for estimating the value of Pi, based on Monte Carlo technique [12, 21] and realized using Scratch blocks is presented in Fig.4.

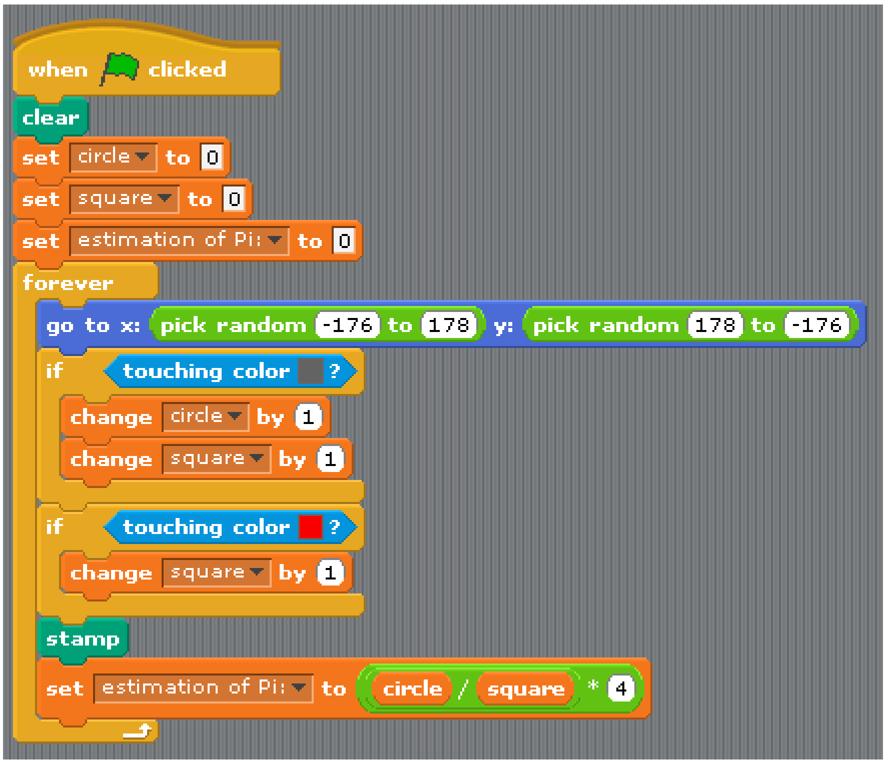

Figure 4: Sample algorithm for estimating the value of Pi realized in Scratch

The Green Flag presented in Fig.4 provides a convenient way to start script, while the Stop All block allows user to make a choice of estimation process duration. During the simulation process, the sample points are plotted using pen tool 'stamp', as it is presented in Fig.5.

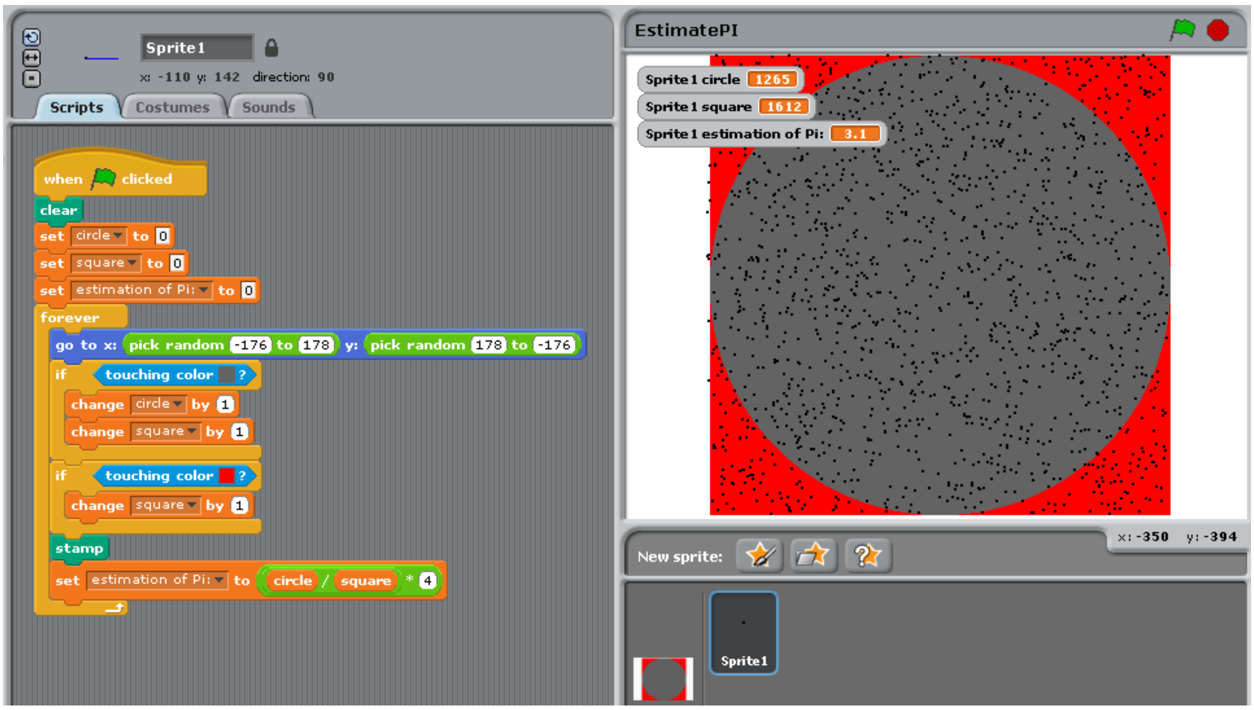

Figure 5: An interactive view of generated points in Scratch

5. CONCLUSION

Monte Carlo technique uses randomly generated numbers or events to simulate random processes and estimate complicated results. Monte Carlo simulations are used to model financial systems, to simulate telecommunication networks, to compute results for high-dimensional integrals or to approximate values such as constants or numeric integrals. In this paper the approximate value of Pi is estimated using Monte Carlo simulations, by generating random points in a square around a circle, using the relationship between the area of the square and the circle. MATLAB's built-in random number generation functions are used, while some interactive animations during the simulation process are also provided.

REFERENCES

[1] FISHMAN, G.S.: *Monte Carlo: Concepts, Algorithms, and Applications*, New York: Springer, 1995.
[2] NIEDERREITER, H.: *Random Number Generation and Quasi-Monte Carlo Methods*, SIAMCBMS-NSF, Philadelphia, 1992.
[3] RUBINSTEIN, R.Y.; KROESE, D.P.: *Simulation and the Monte Carlo Method*, 2nd Ed, New York: John Wiley & Sons, 2007.
[4] PAPOULIS, A.: *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, 1991.
[5] BERG, BERND A.: *Markov Chain Monte Carlo Simulations and Their Statistical Analysis (With Web-Based Fortran Code)*, Hackensack, NJ: World Scientific, 2004.
[6] CAFLISCH, R.E.: "Monte Carlo and quasi-Monte Carlo methods", *Acta Numerica*, No.**7**., Cambridge University Press, 1998, pp. 1–49.
[7] SIMON, M.K.: *Probability Distributions Involving Gaussian Random Variables*, Kluwer, 2002.
[8] TRANTER, W.H.; SHANMUGAN K.S.; RAPPAPORT T.S.; KOSBAR K.L.: *Principles of Communication Systems Simulation with Wireless Applications*, Prentice Hall, Upper Saddle River, NJ, 2004.
[9] STAFANOVIC, Z.H,; STRBAC-SAVIC, S.; MILIC, D.: "Some methods for estimating the performance of a digital communication system using a coin-tossing experiment", *In Proc. Int. Conf. Science and Higher Education in Function of Sustainable Development-SED 2014*, 2014, pp. 2-40 – 2-44.
[10] STARK, H.; WOODS, J.W.: *Probability and Random Processes with Applications to Signal Processing*, Prentice Hall PTR, 2002.
[11] PROAKIS, J.; SALEHI, M; BAUCH, G.: *Contemporary Communication Systems Using MATLAB*, 3rd Ed, Global Engineering: C.M. Shortt, 2013.
[12] http://demonstrations.wolfram.com/MonteCarloEstimateForPi/
[13] GRADSHTEYN, I.S.; RYZHIK, I.M.: *Table of Integrals, Series and Products*. Academic Press Inc, 7th Ed., 2007.
[14] CHUDNOVSKY, D.V.; CHUDNOVSKY, G.V.: "Computation and Arithmetic Nature of Classical Constants", *IBM Research Report, IBM T. J. Watson Research Center*, RC14950 (66818), 1989.
[15] http://www.numberworld.org/misc_runs/pi-12t/
[16] MASSEY, F.J.: "The Kolmogorov-Smirnov Test for Goodness of Fit", *Journal of the American Statistical Association*. Vol. 46, No. 253, 1951, pp. 68–78.
[17] MILLER, L.H.: "Table of Percentage Points of Kolmogorov Statistics", *Journal of the American Statistical Association*. Vol. 51, No. 273, 1956, pp. 111–121.
[18] MARSAGLIA, G.; TSANG, W.; WANG, J.: "Evaluating Kolmogorov's Distribution", *Journal of Statistical Software*. Vol. 8, Issue 18, 2003.
[19] MOLER, C.: "Random Thoughts: 10^{435} Is a Very Long Time", *Matlab News&Notes*, Natick, MA: MathWorks, Fall, 1995.
[20] http://llk.media.mit.edu
[21] http://scratch.mit.edu