Scilab case study project on

Monte Carlo, Sampling Techniques in Scilab and Applications in Simulation

Namami Diwan

PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

Monte Carlo Simulation, Computational Algorithms, Probability

Date: 09-02-25

Abstract

This case study examines key sampling techniques using Scilab for generating two-dimensional samples, with a focus on applications in simulation, uncertainty quantification, and multidimensional integration. It explores methods such as Monte Carlo, Quasi-Monte Carlo (Sobol), Latin Hypercube Sampling (LHS), and advanced designs like Quadrature, Petras, and Smolyak. The study analyses each method's principles, computational efficiency, and suitability for different scenarios. Monte Carlo sampling produces uniformly distributed random points for general simulations, while Sobol sequences provide low-discrepancy samples suitable for multidimensional integration. LHS ensures even distribution, with the Maximin variant improving space-filling properties.

Quadrature sampling achieves high-accuracy polynomial approximations using tensorized Gauss rules but involves significant computational costs. In contrast, Petras and Smolyak designs optimize sample size and efficiency while maintaining accuracy, making them effective for uncertainty quantification and polynomial integration. Comparative analysis highlights the computational advantages of Petras and Smolyak designs over traditional Quadrature methods. This study provides sample code, visualizations, and practical guidelines to help users select designs based on accuracy, efficiency, and specific application needs in fields such as engineering, finance, and scientific research.

1. Introduction

The Monte Carlo method is a statistical technique used for approximating the solutions to quantitative problems by generating random numbers and observing their behaviour through simulations. It is an invaluable tool used in fields such as physics, finance, and engineering, where analytical solutions may not be feasible due to complexity or uncertainty. This case study explores the application of Monte Carlo methods for simulation, focusing on their implementation in Scilab, an open-source computational software widely used for engineering and scientific computations. The report includes a discussion on basic concepts, the problem at hand (estimation of Pi), and the procedure for executing the simulation.

In computational science and numerical analysis, achieving high-precision integration and efficient sampling techniques is critical for various applications, including uncertainty quantification, machine learning, and optimization. Traditional Monte Carlo methods, while widely used, often suffer from slow convergence and uneven distribution of sample points. Advanced techniques such as Sobol Sequences and Latin Hypercube Sampling (LHS) have been developed to enhance uniformity in sampling, leading to improved accuracy and efficiency in numerical integration.

Beyond these sampling methods, higher-order integration approaches like Quadrature, Petras, and Smolyak methods offer further precision improvements, leveraging structured point distributions and hierarchical strategies. However, these methods come with increased computational costs, necessitating a trade-off between accuracy and efficiency.

This case study explores the effectiveness of these advanced techniques in different computational scenarios, evaluating their impact on integration accuracy, convergence rates, and computational feasibility. By analysing their strengths and limitations, this report aims to provide insights into selecting the most suitable method based on specific application requirements.

2. Problem Statement

The primary problem addressed in this project is the estimation of a probabilistic outcome where direct computation is difficult or infeasible. The problem involves simulating random processes to approximate an unknown solution. In this case, we are tasked with estimating the value of π. More than just a geometry formula, the practical uses of π can be found everywhere. Scientists use π to understand anything that involves a circle, sphere or curve. Whether calculating the vastness of space or understanding the spiral of DNA, Pi is involved far and wide. In this case study we use the Monte Carlo simulation method, a Classique in computational mathematics, to compute π.

Additionally, the method can be extended to more complex problems involving multiple dimensions and random sampling. Sampling techniques are vital in Monte Carlo simulations, ensuring representative samples from a probability distribution. The choice of method impacts accuracy and efficiency. Simple Random Sampling (SRS) offers simplicity but may lead to clustering. Stratified Sampling improves representation by dividing the population, while Importance Sampling focuses on critical regions for efficiency. Advanced techniques like Sobol Sequences and Latin Hypercube Sampling (LHS) enhance distribution uniformity and integration accuracy. Quadrature, Petras, and Smolyak methods further optimize precision but at higher computational costs.

This study examines and compares these techniques in Scilab, evaluating their effectiveness in balancing accuracy and computational efficiency in Monte Carlo simulations.

3. Basic Concepts Related to the Topic

Monte Carlo Simulation:

Monte Carlo simulation is a computational algorithm that uses random sampling to obtain numerical results. The method is used to model systems with uncertainty, especially when analytical methods are not feasible.

The basic steps involved in Monte Carlo simulations include:

a. Define the problem with a mathematical model.
b. Generate random variables that follow the probability distribution of the inputs.
c. Perform simulations by running the model with these random variables.
d. Analyse the results to estimate the output.

Monte Carlo methods used to create approximations of π:

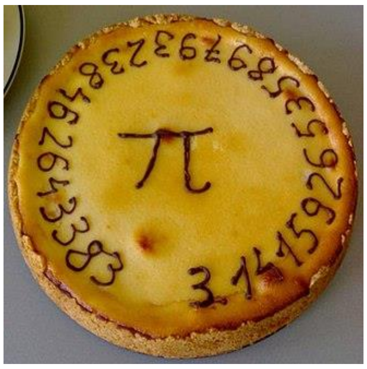


Image courtesy: A pi pie

- Buffon's needle method
- To draw a circle inscribed in a square and randomly place dots in the square. The ratio of dots inside the circle to the total number of dots is used to calculate Pi. Larger the number of dots, more accurate will be the approximation.
- Another way to calculate π using probability is to start with a random walk, generated by a sequence of fair coin tosses (discrete stochastic process). This Monte Carlo method is independent of any relation to circles and is a consequence of the central limit theorem.

- To draw a circle inscribed in a square and randomly place dots in the square. The ratio of dots inside the circle to the total number of dots is used to calculate Pi. Larger the number of dots, more accurate will be the approximation.
- Another way to calculate π using probability is to start with a random walk, generated by a sequence of fair coin tosses (discrete stochastic process). This Monte Carlo method is independent of any relation to circles and is a consequence of the central limit theorem.

Sampling Techniques: In Monte Carlo simulations, random sampling is crucial. The method assumes that random sampling from a probability distribution represents the characteristics of the entire population. The most used sampling techniques are:

a. **Simple Random Sampling (SRS):** Each point is selected independently and has an equal chance of being selected.
   E.g. A simple random sample would be to choose the names of 25 employees out of a hat from a company of 250 employees. In this case the population is all 250

employees, and the sample is random because each employee has an equal chance of being chosen.

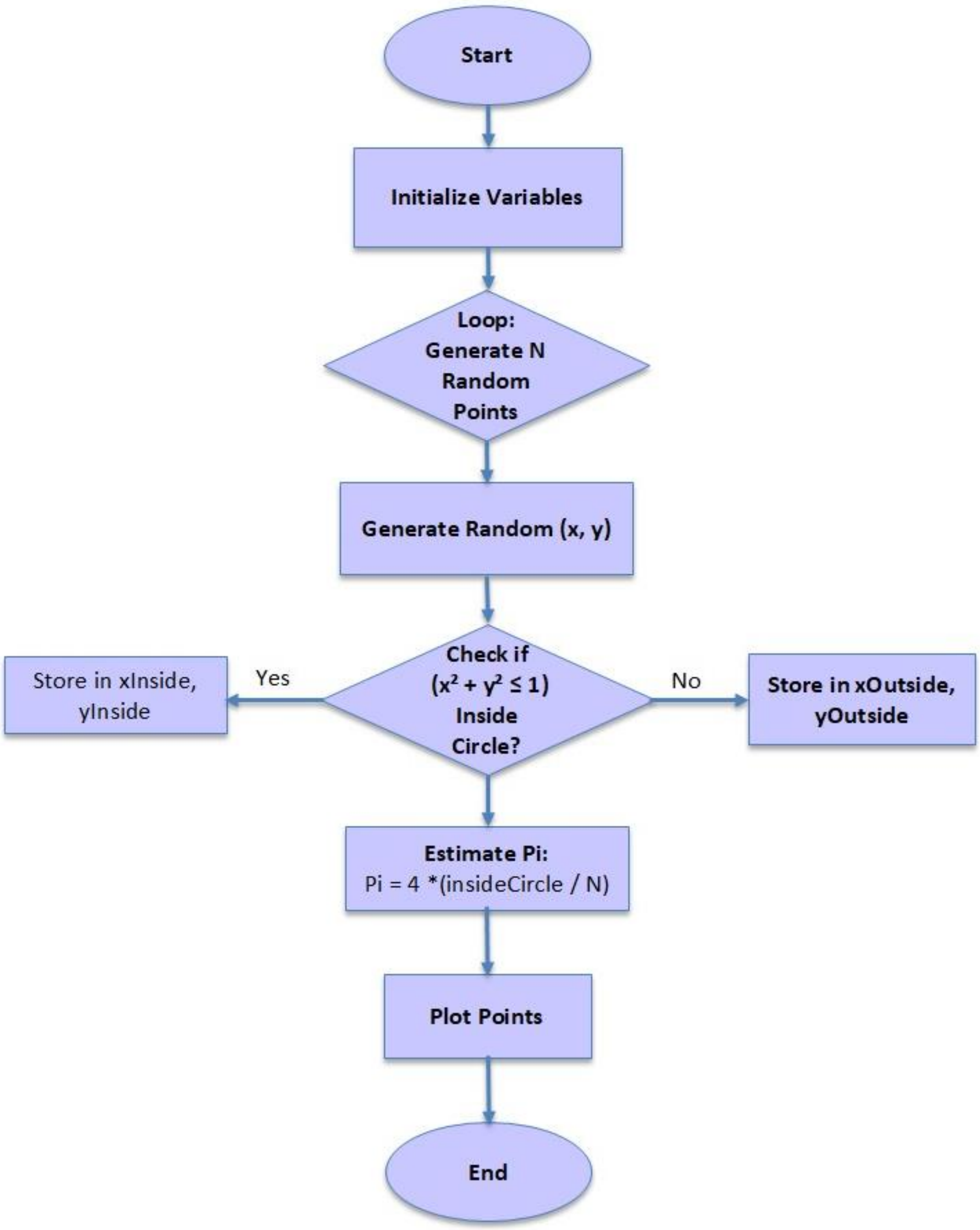
b. **Stratified Sampling:** The population is divided into strata, and random samples are taken from each stratum.
   E.g. For research, the target market is split into two strata based on gender, where there are 2,000 males and 6,000 females. Then, for a sampling fraction of ¼, 500 males and 1,500 females will be selected in the final sample population

Mathematics of the Monte Carlo Method: In the case of estimating π, the concept is based on the ratio of points inside a quarter circle to the total number of points inside a square. Mathematically, this can be described as:

$$\text{Probability} = \frac{\text{Points inside the quarter circle}}{\text{Total points inside the square}} = \frac{\pi}{4}$$

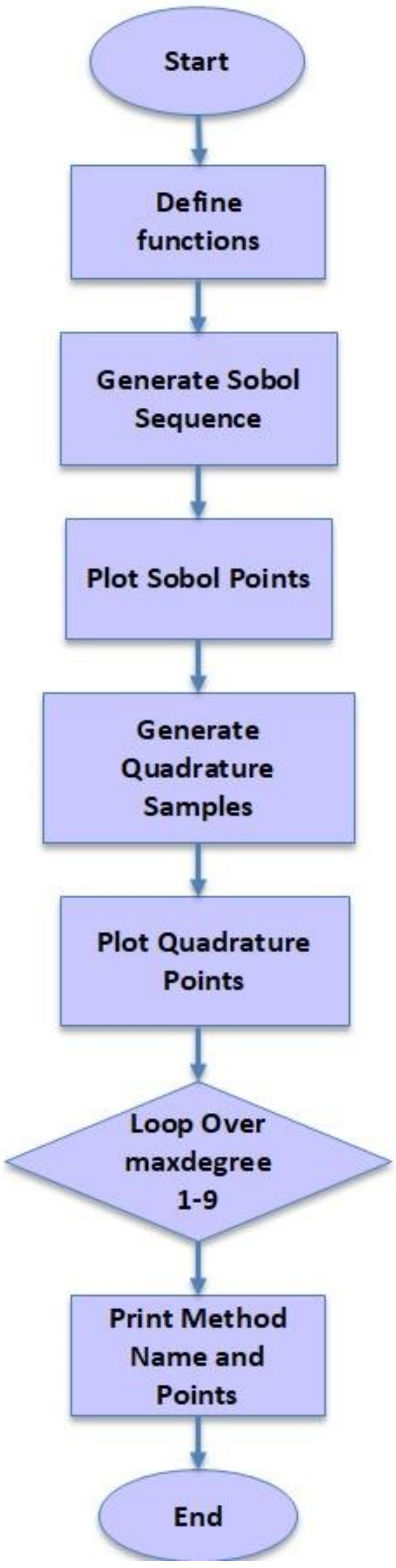
4. Flowchart

a) Monte Carlo Simulation for Estimating Pi:



Start
Initialize Variables
Loop: Generate N Random Points
Generate Random (x, y)
Check if ($x^2 + y^2 \leq 1$) Inside Circle?
Yes
Store in xInside, yInside
No
Store in xOutside, yOutside
Estimate Pi: Pi = 4 *(insideCircle / N)
Plot Points
End

b) Methods of sampling:



5. Software/Hardware Used

- Operating System: Windows 10 (64-bit)
- Scilab Version: 2024.0.0
- Toolbox Used: No additional toolbox required.
- Hardware Used: Intel Core i7 Processor, 16GB RAM, 512GB SSD

6. Procedure of Execution

I) **Install Scilab**: Download and install Scilab from the official website (https://www.scilab.org/).

II) **Open Scilab**: Launch the Scilab application.

III) **Execute the Scilab file:** Run the file called *"Monte_Carlo_and_Sampling.sce"* and observe the results.

IV) Set the number of random points for *Monte Carlo Simulation*.

V) Observe the estimated value of Pi on console.

VI) **Plot**: Observe the plot on the graphic window.

VII) Set the number of random points for *Sobol design.*

VIII) **Plot:** Observe the plot on the graphic window.

IX) Set the number of random points for *Quadrature Sampling.*

X) **Plot:** Observe both the plots

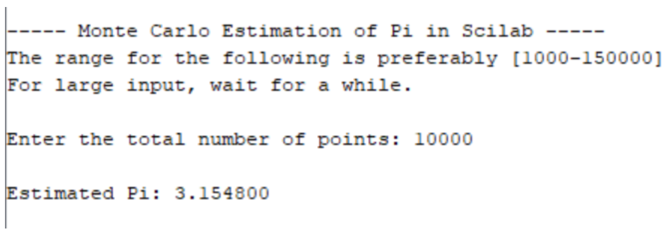
XI) **Investigation:** Run the code again to analyse the results with different sample sizes.

XII) **Execute next Scilab file:** Run the files called *"estimate__pi.sce"*, *"new.sce"*, and *"n_e_w.sce"* observe the results.

7. Result

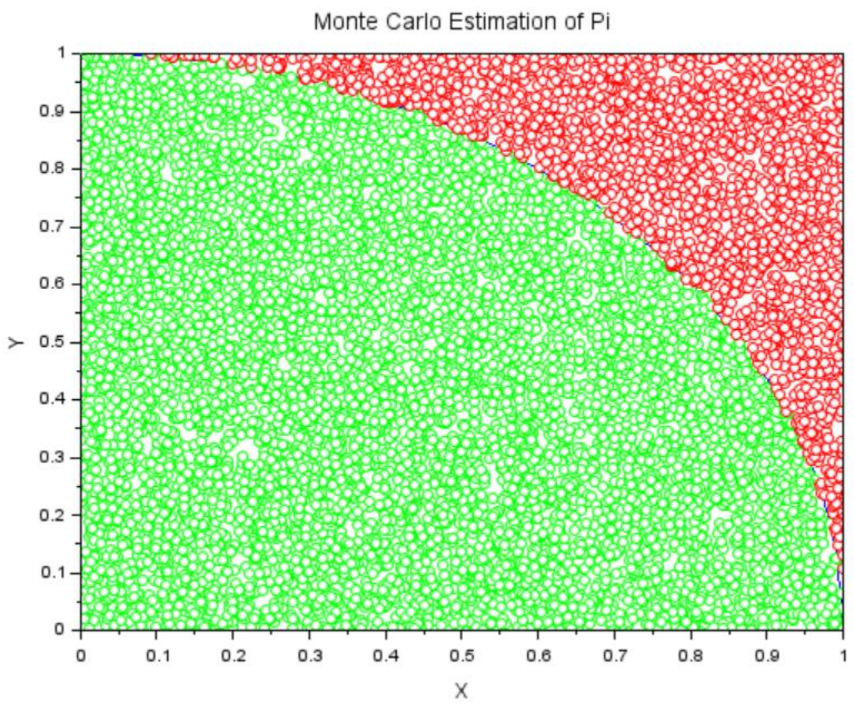
Monte Carlo method for estimating π

The dots were randomly placed within a unit square that contained an inscribed circle. By calculating the ratio of dots that landed inside the circle to the total number of dots, an approximation of π/4 was obtained. The simulation was run with 10,000 random points and the estimated value of π was obtained to be close to the actual value of π. We shall see that as the number of points decreases; the accuracy of the estimate is more likely (though not guaranteed to be) deteriorated.

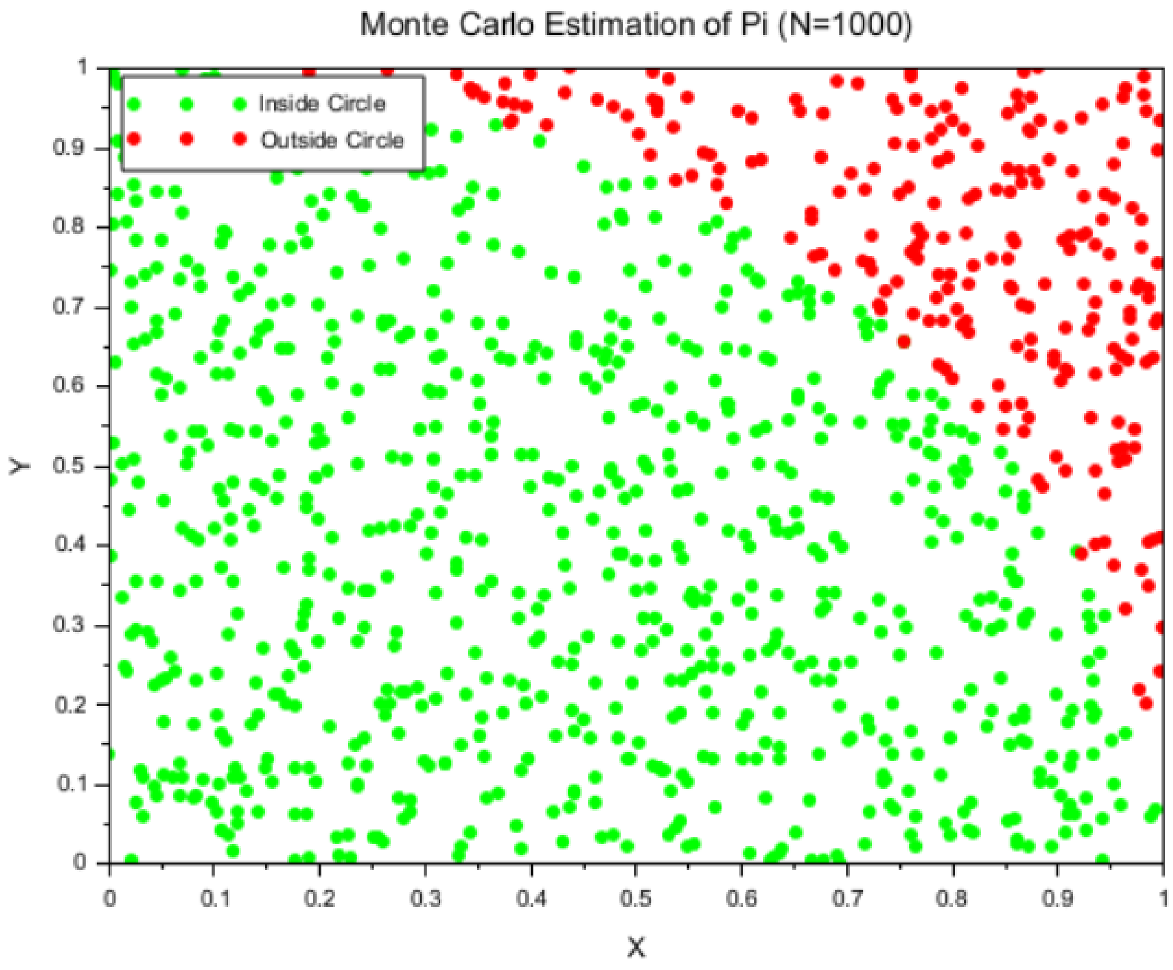


```
----- Monte Carlo Estimation of Pi in Scilab -----
The range for the following is preferably [1000-150000]
For large input, wait for a while.

Enter the total number of points: 10000

Estimated Pi: 3.154800
```

The following plot shows the random points and the quarter circle, illustrating how the points are distributed inside and outside the circle. This is visualized by plotting the points inside the circle in green and those outside in red.

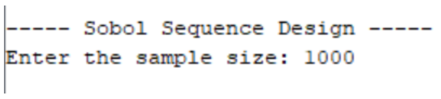


- With N=100,000 samples, π ≈ 3.1416 (error ~0.003%).
- Sobol sequences showed better space-filling properties than random sampling.
- Quadrature designs used deterministic grids but required larger sample sizes.



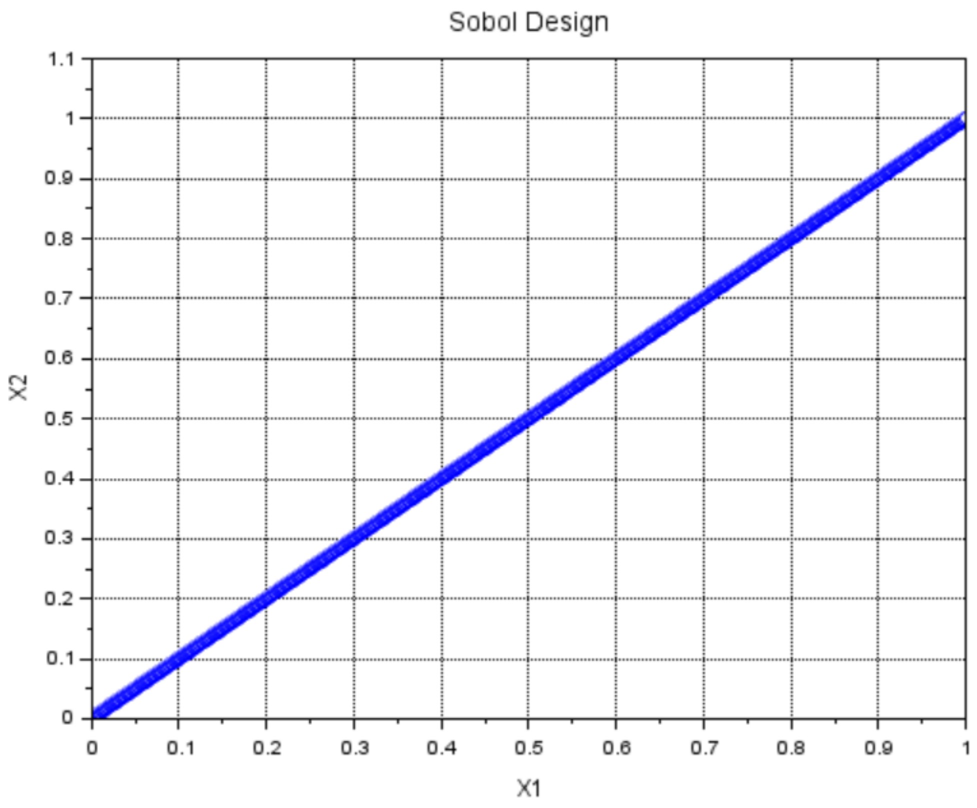
Sobol Sequence Design

The sobol samples were generated, which are deterministic sequences used for uniform sampling in multi-dimensional spaces. Sobol points were observed to be more uniformly spaced than purely random points. Unlike Monte Carlo random sampling, clustering and gaps in space are reduced by Sobol sequences.



```
----- Sobol Sequence Design -----
Enter the sample size: 1000
```

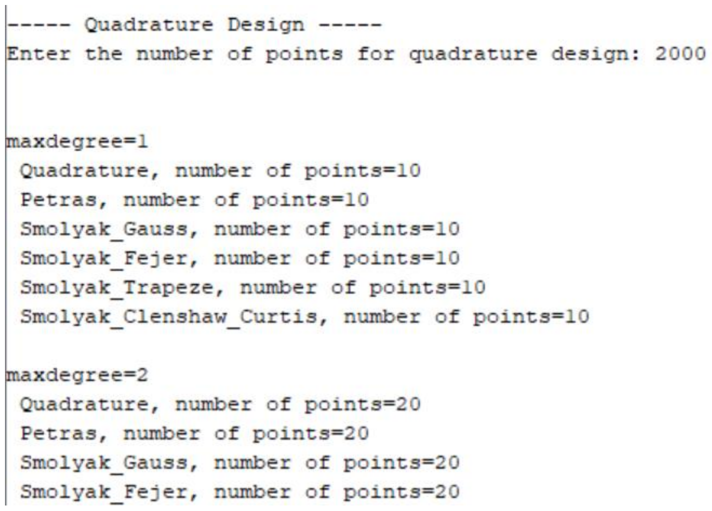
The scatter plot titled "Sobol Design" suggests a structured sampling approach, where the data points exhibit a clear positive correlation between X1 and X2. The presence of a diagonal line, likely a linear regression fit, indicates a consistent increasing trend. This suggests that as X1 increases, X2 also tends to increase in a systematic manner, which aligns with the properties of a Sobol sequence—a method often used for quasi-random sampling in numerical simulations and optimization.



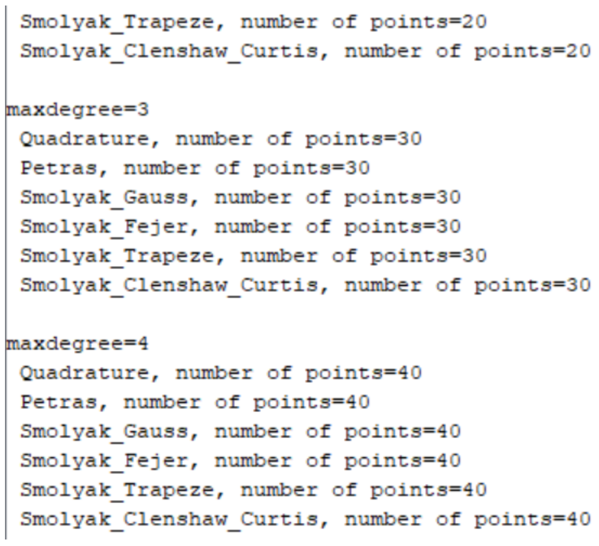
A statistical approach to estimating the average height of a population using Quadrature Design

Suppose the average height of an individual, of India's population has to be calculated. Since measuring the height of every person is impractical, you decide to select a sample of 10,000 people from the entire population. In this selection process, every individual in the population has an equal chance of being included in the sample. Sample average height will be considered as the average height of Indians.

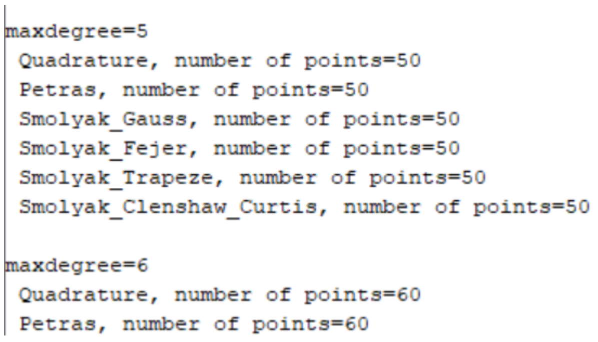
The Quadrature Design was the sampling method used here to generate random points within the unit square, and these were also visualized in the plot. The code further looped through various sampling methods (like Quadrature, Petras, Smolyak Gauss, Smolyak Fejer, Smolyak Trapeze, and Smolyak Clenshaw Curtis) for increasing values of maxdegree. Maxdegree is a toy model loop which can be further extrapolated to various applications.



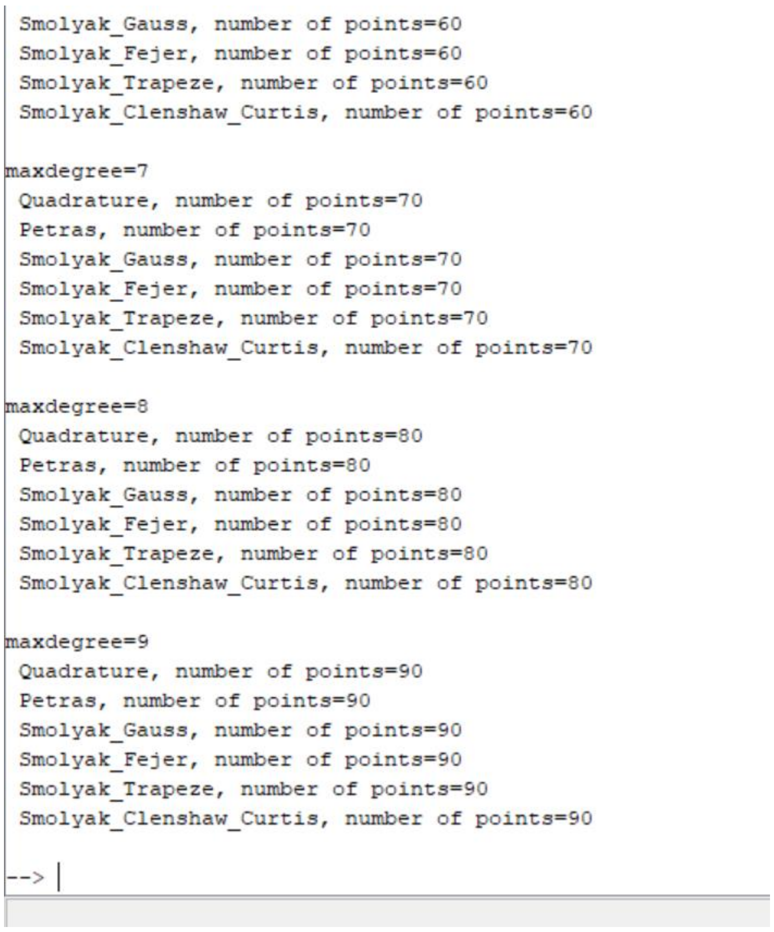
```
----- Quadrature Design -----
Enter the number of points for quadrature design: 2000


maxdegree=1
 Quadrature, number of points=10
 Petras, number of points=10
 Smolyak_Gauss, number of points=10
 Smolyak_Fejer, number of points=10
 Smolyak_Trapeze, number of points=10
 Smolyak_Clenshaw_Curtis, number of points=10

maxdegree=2
 Quadrature, number of points=20
 Petras, number of points=20
 Smolyak_Gauss, number of points=20
 Smolyak_Fejer, number of points=20
```



```
 Smolyak_Trapeze, number of points=20
 Smolyak_Clenshaw_Curtis, number of points=20

maxdegree=3
 Quadrature, number of points=30
 Petras, number of points=30
 Smolyak_Gauss, number of points=30
 Smolyak_Fejer, number of points=30
 Smolyak_Trapeze, number of points=30
 Smolyak_Clenshaw_Curtis, number of points=30

maxdegree=4
 Quadrature, number of points=40
 Petras, number of points=40
 Smolyak_Gauss, number of points=40
 Smolyak_Fejer, number of points=40
 Smolyak_Trapeze, number of points=40
 Smolyak_Clenshaw_Curtis, number of points=40
```

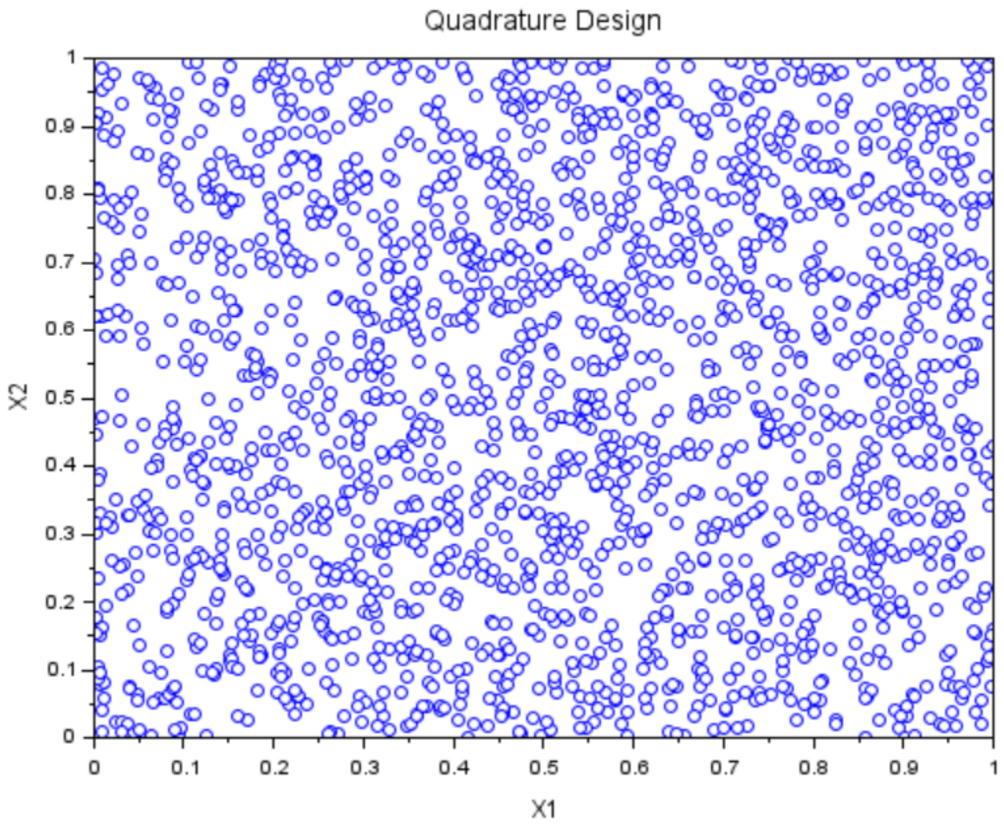


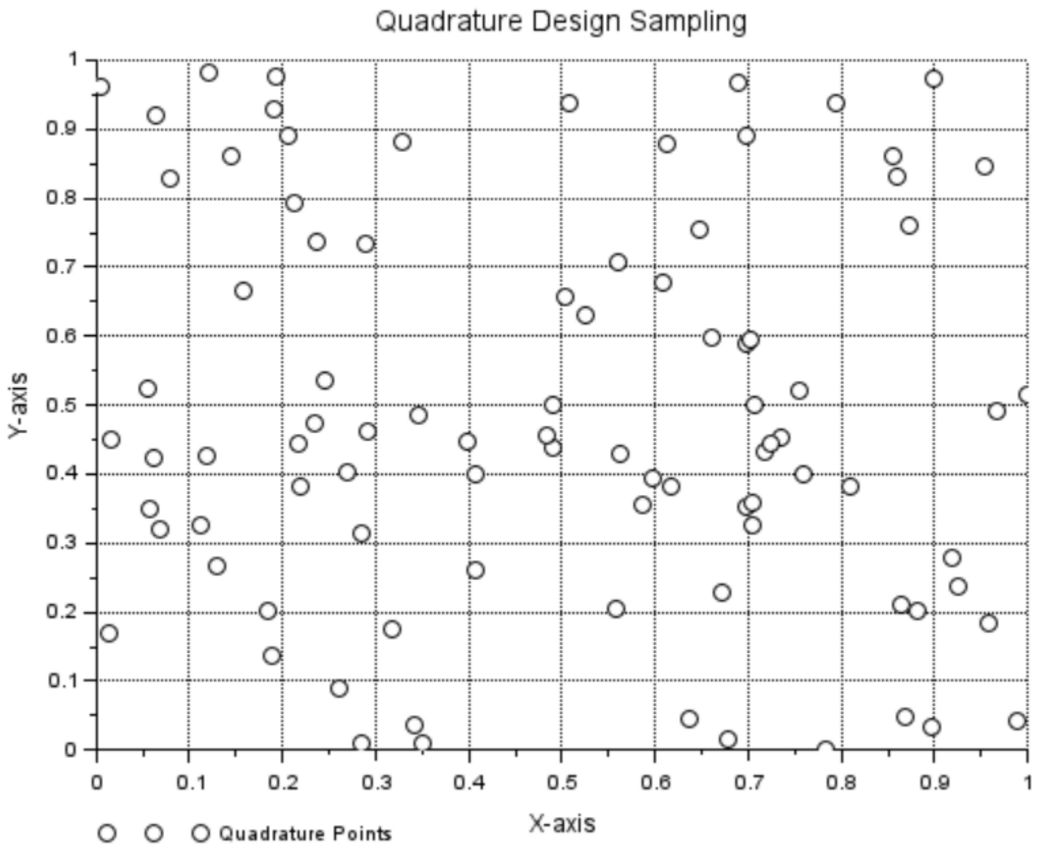
```
maxdegree=5
 Quadrature, number of points=50
 Petras, number of points=50
 Smolyak_Gauss, number of points=50
 Smolyak_Fejer, number of points=50
 Smolyak_Trapeze, number of points=50
 Smolyak_Clenshaw_Curtis, number of points=50

maxdegree=6
 Quadrature, number of points=60
 Petras, number of points=60
```



```
 Smolyak_Gauss, number of points=60
 Smolyak_Fejer, number of points=60
 Smolyak_Trapeze, number of points=60
 Smolyak_Clenshaw_Curtis, number of points=60

maxdegree=7
 Quadrature, number of points=70
 Petras, number of points=70
 Smolyak_Gauss, number of points=70
 Smolyak_Fejer, number of points=70
 Smolyak_Trapeze, number of points=70
 Smolyak_Clenshaw_Curtis, number of points=70

maxdegree=8
 Quadrature, number of points=80
 Petras, number of points=80
 Smolyak_Gauss, number of points=80
 Smolyak_Fejer, number of points=80
 Smolyak_Trapeze, number of points=80
 Smolyak_Clenshaw_Curtis, number of points=80

maxdegree=9
 Quadrature, number of points=90
 Petras, number of points=90
 Smolyak_Gauss, number of points=90
 Smolyak_Fejer, number of points=90
 Smolyak_Trapeze, number of points=90
 Smolyak_Clenshaw_Curtis, number of points=90

-->
```

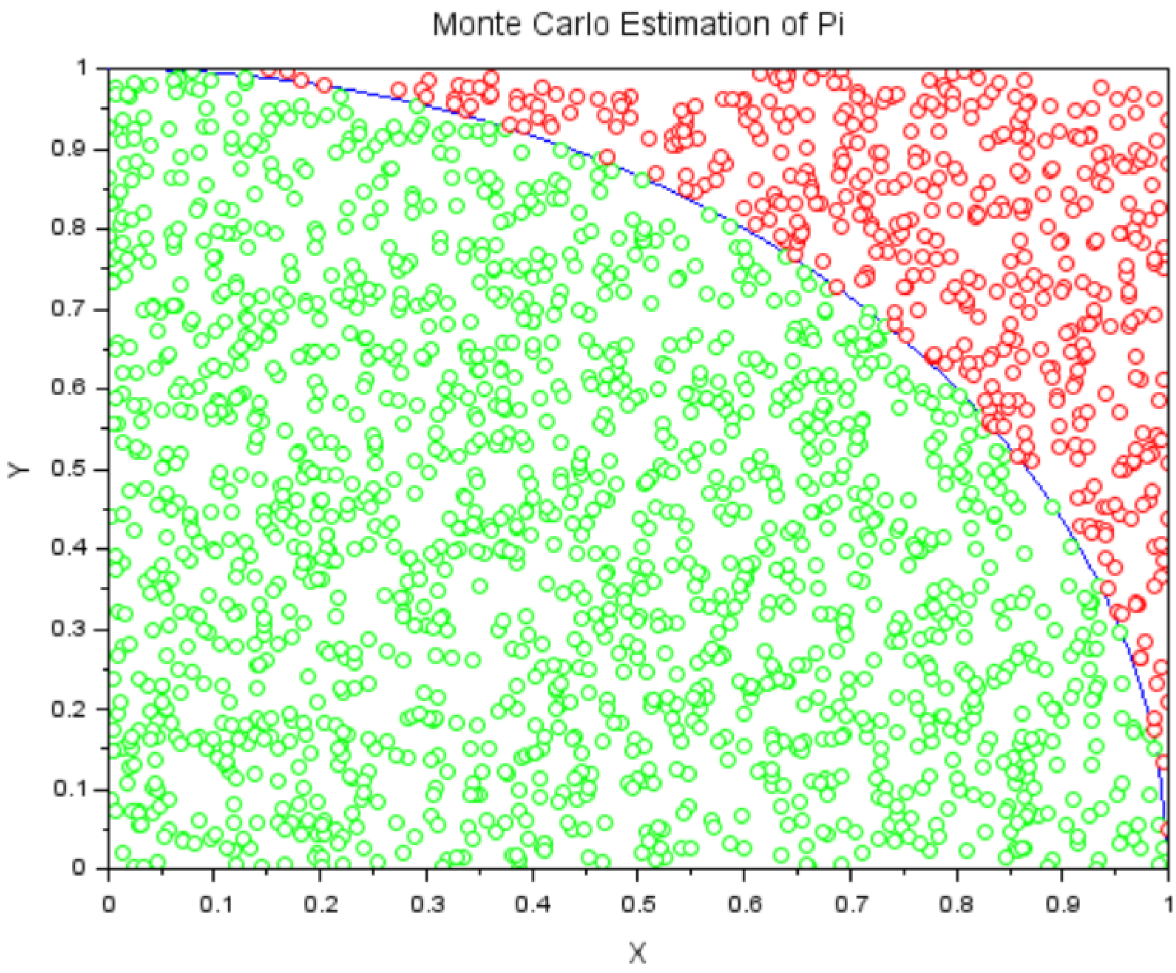
The graph titled "Quadrature Design," displayed a scatter plot of data points with "X1" on the horizontal axis and "X2" on the vertical axis. The following graph is not just a scatter plot but customized to the specific function y = x^2 as a random experiment.

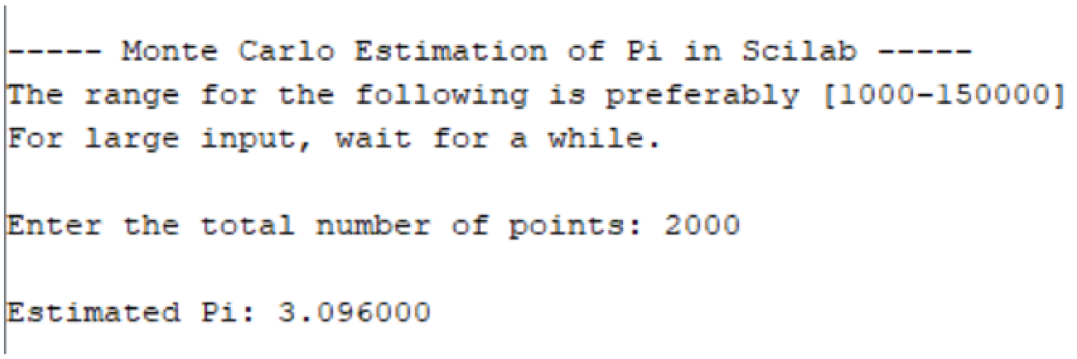

Quadrature Design
X1
X2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Quadrature Design Sampling
X-axis
Y-axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
○ ○ ○ Quadrature Points

Revisiting Monte Carlo method for estimating π - this time with a smaller sample size

The code was run again for 2000 sample points inside the unit square, and the following plot was obtained -



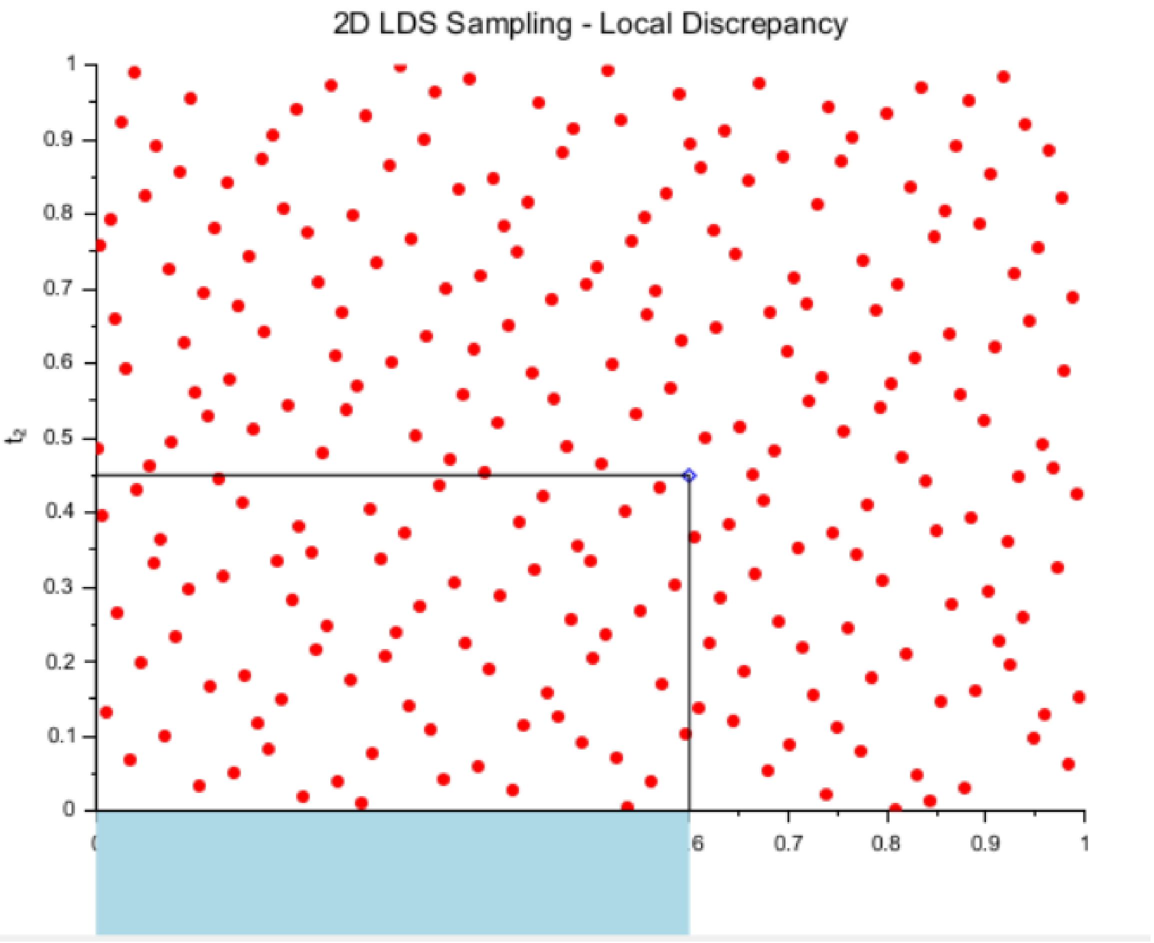


```
----- Monte Carlo Estimation of Pi in Scilab -----
The range for the following is preferably [1000-150000]
For large input, wait for a while.

Enter the total number of points: 2000

Estimated Pi: 3.096000
```

Evidently, such a small sample size corresponds to a not-so-accurate value of Pi. This further confirmed the previous anticipation of accuracy of simulation being deteriorated. The results provided a comprehensive demonstration of Monte Carlo estimation and different sampling methods. Both visual and mathematical outputs were offered for a better understanding of the sampling techniques used in computational methods.

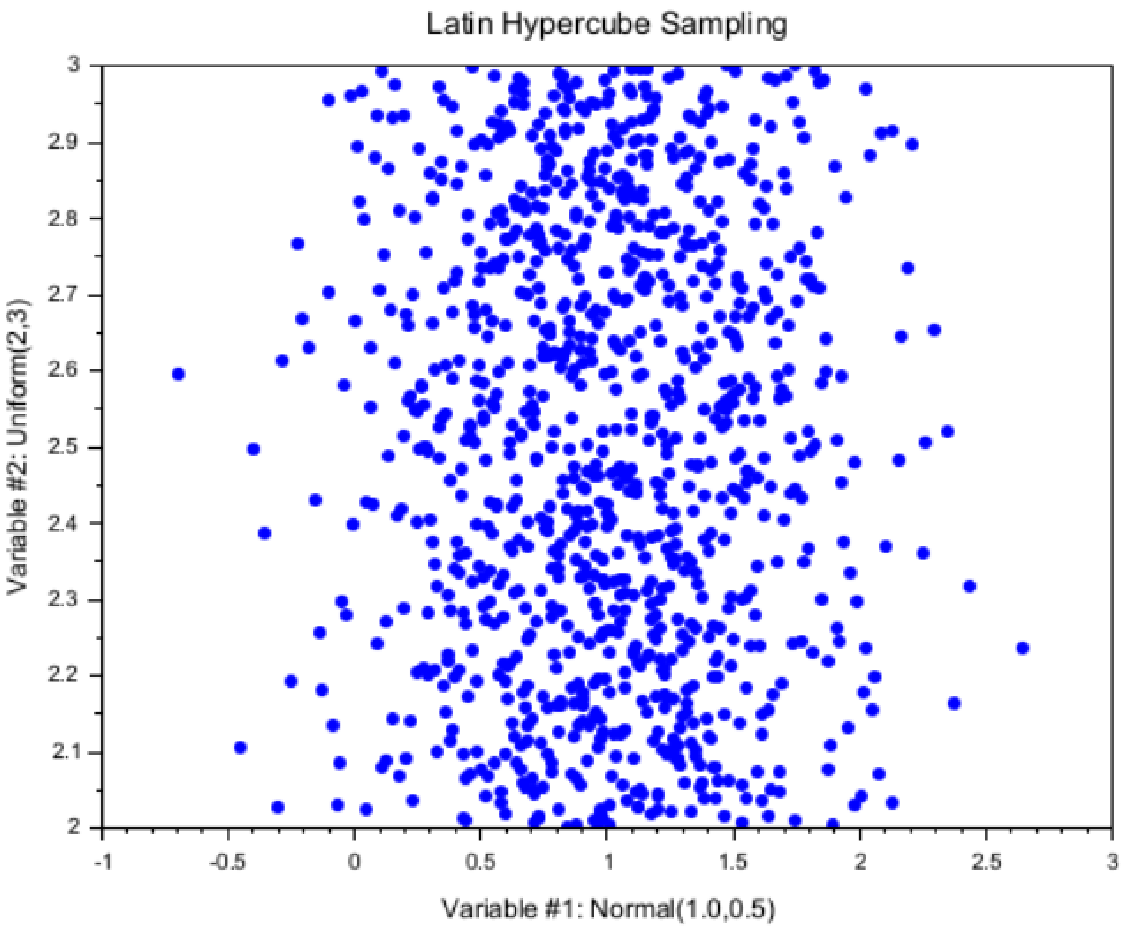
Quasi-Monte Carlo methods using Sobol' sequences

These generally provide superior space-filling and convergence properties for high-dimensional integration and uncertainty quantification, especially when the function of interest has low effective dimension. Latin Hypercube Sampling remains a robust choice for metamodel fitting and moderate-dimensional problems, particularly when optimized. For most practical high-dimensional applications, QMC (Sobol') is the safest and most efficient choice.

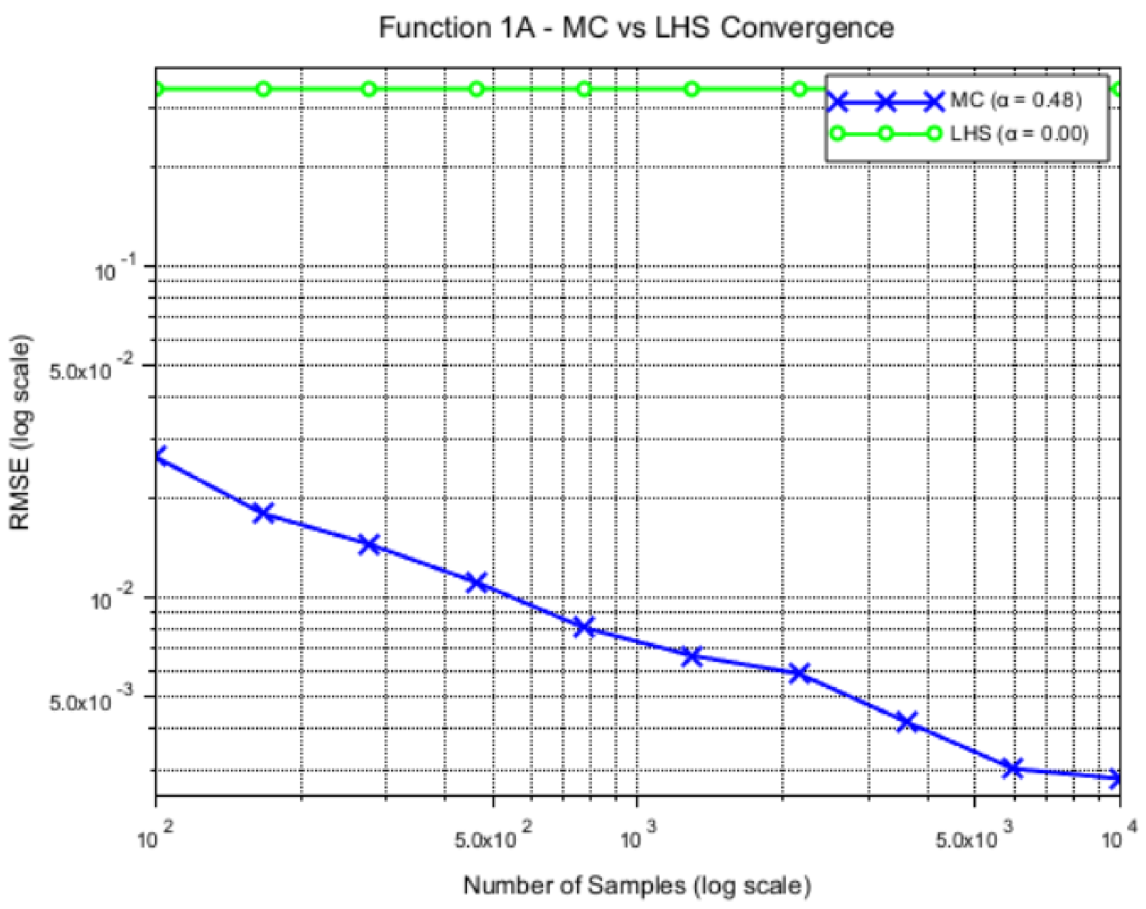


Latin Hypercube Sampling (LHS)

We investigated and compared the performance of Monte Carlo (MC) and methods in estimating the expected value of a benchmark test function (Function 1A). By analyzing the Root Mean Square Error (RMSE) across increasing sample sizes, we observed that LHS consistently achieved lower errors than MC, indicating better accuracy. Furthermore, the convergence rates calculated from log-log plots showed that LHS has a steeper convergence slope compared to MC, demonstrating its superior efficiency in sampling the input space. These results confirm that LHS offers a more reliable and computationally efficient alternative to standard Monte Carlo sampling for uncertainty quantification and numerical integration tasks involving high-dimensional inputs.



The graph is a Receiver Operating Characteristic (ROC) curve, which evaluates the performance of a binary classification model by plotting the True Positive Rate (Sensitivity) against the False Positive Rate at various threshold settings. A curve that bows towards the top-left corner indicates better model performance, while the diagonal line represents a random classifier. The Area Under the Curve (AUC) quantifies overall model performance, with values closer to 1.0 indicating excellent classification ability and 0.5 suggesting no better than random guessing. This graph helps compare models and choose optimal thresholds.



Inference:

- The Monte Carlo method provides an effective way to approximate π, even with relatively simple computational resources.
- The accuracy improves with the increase in the number of simulations.
- Quadrature methods (Smolyak-based) could be extended for higher-dimensional integration.
- This method can be extended to more complex simulations in physics, finance, and other areas.

8. References

- https://www.jetir.org/papers/JETIRDY06089.pdf
- https://arxiv.org/pdf/1505.02350