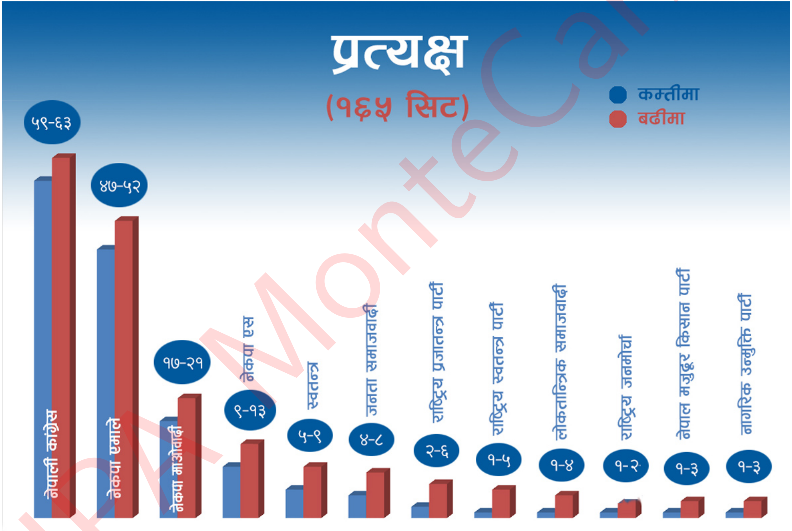
Annapurna Post Survey

Prof. Dr. Narayan Prasad Adhikari
Central Department of Physics
Tribhuvan University Kirtipur, Kathmandu, Nepal

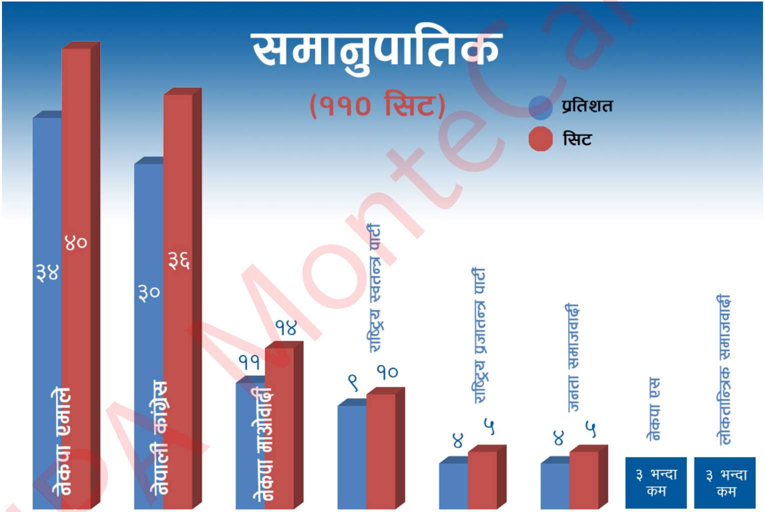
December 6, 2022

■Contents



प्रत्यक्ष
(१६५ सिट)
कम्तीमा
बढीमा
५९-६३
नेपाली कांग्रेस
४७-५२
नेकपा एमाले
१७-२१
नेकपा माओवादी
९-१३
नेकपा एस
५-९
स्वतन्त्र
४-८
जनता समाजवादी
२-६
राष्ट्रिय प्रजातन्त्र पार्टी
१-५
राष्ट्रिय स्वतन्त्र पार्टी
१-४
लोकतान्त्रिक समाजवादी
१-२
राष्ट्रिय जनमोर्चा
१-३
नेपाल मजदुर किसान पार्टी
१-३
नागरिक उन्मुक्ति पार्टी

■Contents



समानुपातिक
(११० सिट)
प्रतिशत
सिट
३४
४०
नेकपा एमाले
३०
३६
नेपाली कांग्रेस
११
१४
नेकपा माओवादी
९
१०
राष्ट्रिय स्वतन्त्र पार्टी
४
५
राष्ट्रिय प्रजातन्त्र पार्टी
४
५
जनता समाजवादी
नेकपा एस
३ भन्दा कम
लोकतान्त्रिक समाजवादी
३ भन्दा कम

■Contents

सर्वेक्षण विधि

- ७७ जिल्लाकै १ सय ६५ निर्वाचन क्षेत्र र तिनका प्रदेश क्षेत्र (क, ख) का १९ हजार ८ सय ५८ जनामा सर्वेक्षण गरिएको हो ।
- असोज ३० देखि कात्तिक १३ सम्म यसका लागि सर्वेक्षकहरु हरेक क्षेत्रमा गरी ६ सय ७९ गणक खटाइएका थिए ।
- प्रमुख दलले घोषणापत्र सार्वजनिक गरेपछि पुनः ३ दिन सर्वेक्षण गरिएको थियो ।
- निर्वाचन क्षेत्रका जनसंख्याका अनुपातमा सर्वेक्षण आकार तय गरिएको थियो ।
- यो 'न्यान्डम स्याम्प्लिङ' विधिमा गरिएको सर्वेक्षण हो ।
- अनुसन्धानका लागि गुगल फर्म प्रयोग गरिएको थियो ।
- यसमा व्यक्तिको गोपनीयता पूर्णतः कायम गरिएको छ ।
- व्यक्ति छनोट गर्दा सार्वजनिक स्थल बढी प्रयोग गरिएको थियो ।
- दल, दलका नेता र उम्मेदवार तीनवटै आधारमा रहेर मत प्रक्षेपण गरिएको हो ।

Monte Carlo Methods - Introduction

Prof. Dr. Narayan Prasad Adhikari
Central Department of Physics
Tribhuvan University Kirtipur, Kathmandu, Nepal

January 28, 2024

■Warm up!!!

- Important Discoveries
- What is it?
- Why do we need it in Data Science?

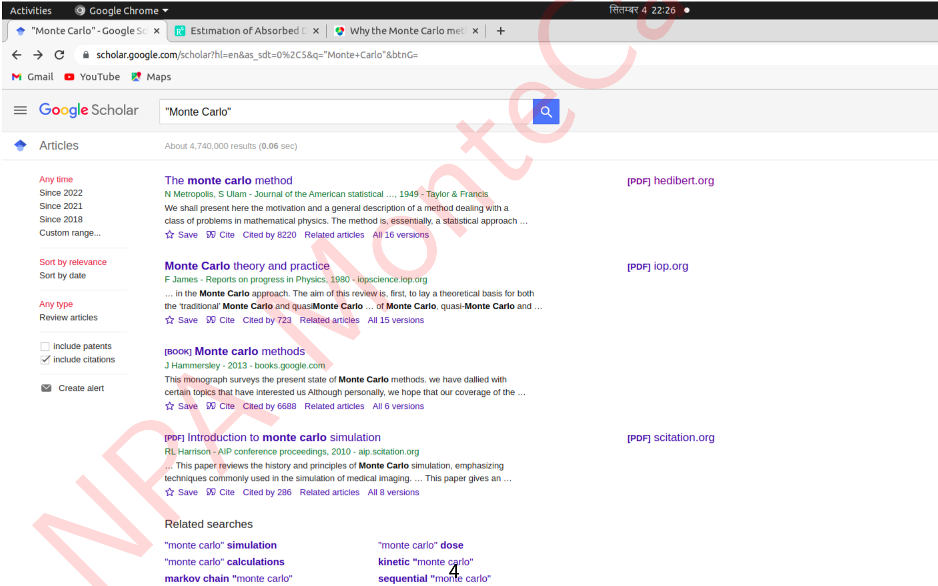
■Warm up - what is really it?!!!

You may face multidimensional integration. It comes in Mathematics as well as in Physics, Statistics ....

$$\int\int.....\int f(x_1)f(x_2).....f(x_n)dx_1dx_2.....dx_n \tag{1}$$

In above equation n may be huge ... 100, 10^4, or 10^6,....?
How to handle such a complex problem? Just think about it....
Does our ways of doing till now work?

■Warm up - what is really it?!!!



Warm up - Health

View PDF Download Full Issue

ELSEVIER

Physica A: Statistical Mechanics and its Applications

Volume 574, 15 July 2021, 126014

A random walk Monte Carlo simulation study of COVID-19-like infection spread

S. Triambak [a], D.P. Mahapatra [b]

Show more

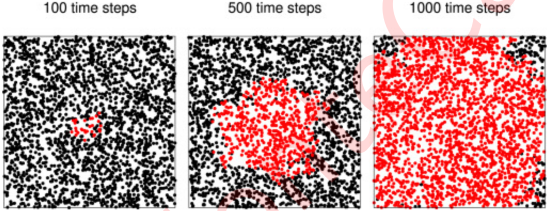
Add to Mendeley Share Cite

https://doi.org/10.1016/j.physa.2021.126014

Abstract

Recent analysis of early COVID-19 data from China showed that the number of confirmed cases followed a subexponential power-law

■Warm up - Health



Download : Download high-res image (1MB)   Download : Download full-size image

Fig. 1. An example of proximity-based infection spread obtained using the random walk Monte Carlo simulations described in this work. Each of the panels shown above has a population of 2.5k over a unit area. The average distance $\langle r \rangle$ between any two points is $= 0.02$ units. In this case every point (walker) takes randomly directed steps of length $l = 0.25 \langle r \rangle$. Further details are described in the text below.

■Warm up - Health

Austin Journal of Radiology

Open Access

Austin Publishing Group

Special Article - Therapeutic Radiology

Estimation of Absorbed Dose Distribution in Different Organs during the CT Scan: Monte Carlo Study

Umit Kara[1]* **and Huseyin Ozan Tekin**[2]
[1]Medical Imaging, Suleyman Demirel University, Turkey
[2]Department of Radiotherapy, Uskudar University, Turkey
***Corresponding author:** Umit Kara, Suleyman Demirel University, Vocational School of Health Services, Medical Imaging, 32100, Isparta, Turkey

Received: March 08, 2017; **Accepted:** March 30, 2017; **Published:** April 04, 2017

Abstract

This work aimed to validate the accuracy of a Monte Carlo source model of the Ge Lightspeed CT scanner using organ doses measured in specific human adult phantoms. The x-ray output of the Ge Lightspeed multidetector CT scanner was simulated within the Monte Carlo code. The resulting source model was able to perform various real simulated scan model helical Computed Tomography (CT) scans of varying scan parameters such as kVp, mAs, filtration, pitch, and beam collimation. This work has been performed by using real Computed Tomography (CT) protocols to patients in public hospitals in Turkey. We used Monte Carlo simulation methods with real height and weight of patients and Computed Tomography (CT) scanners real parameters. Absorbed organ doses have been calculated by using Monte Carlo simulation. The results showed that changes in mAs value are significantly important for obtaining the risk of cancer from dose rates. Additionally, the dose received by each organ has been calculated and the results showed that Monte Carlo is a strong and effective tool in radiological investigations.

Keywords: CT scan; Absorbed dose; Monte carlo simulation

Introduction

Diagnostic radiology is a significant tool for clinical diagnosis and includes general x rays, Computed Tomography (CT), Magnetic Resonance Imaging (MRI), ultrasound, mammography, etc. Diagnostic radiology is part of medicine that uses medical imaging facility and technology to diagnose disease and helps define the structures inside in human body. Diagnostic radiology uses the imaging technologies of X-rays radiography, computed tomography, magnetic resonance imaging, ultrasound, mammography, etc. Common classifications of CT scans are abdominal, bone, head and vascular system. CT scanning is the one of most used units in diagnostic radiology and these units combine the use of X-rays and computer processing to generate tomographic images of the body. The use of CT in radiology and medical imaging has been growing in the world. CT is useful as it allows the radiologist, to view a cross-sectional picture of the entire body. CT is painless, fast, and accurate and CT scan uses x-ray equipment and computers to produce medical images that often can make more detailed image than conventional radiography. CT scans use ionizing radiation as used in other x-rays units. In clinical, the benefits of a correct diagnosis preponderance

most accurate, reliable, and versatile in accomplishing this task [1-8].

In the Monte Carlo method, the patient and CT scanner are simulated using a computational anatomic model of the patient and an x-ray source model representing the scanner's beam output. However, to ensure the accuracy of these calculations, these CT source models must be benchmarked and validated against actual experimental measurements made on the scanners they simulate. In the past, most validation studies were accomplished using standard CT Dose Index (CTDI) phantoms, but in recent years, anthropomorphic phantoms have been increasingly utilized [9,10]. Mathematical phantoms or MIRD phantoms have been produced [11] which were the first models of human phantoms to be widely used in dosimetry studies involving X-ray exposure by the Monte Carlo method.

However, the organs in these phantoms are described by mathematical equations with limited representation of the actual structure of a human body and its chemical and physical characteristics [12]. Voxel-based phantoms created from tomography images present a geometry which adequately represents a patient, including internal organ, displacements, and deformations. These phantoms are recommended for dosimetric studies with the Monte

■Warm up - Health

- Case of Siamese Twins
- Pharmacy
- MC dose calculation in the radiotherapy treatment

Warm up - Artificial intelligence


Outline

Highlights
Abstract
Keywords
1. Introduction
2. Adapting MCTS
3. Examples of usage
4. Quality of decisions
5. Conclusion & future work
Author statement

References

ELSEVIER

Computers in Industry
Volume 128, June 2021, 103433

Monte Carlo Tree Search for online decision making in smart industrial production

Richard Senington, Bernard Schmidt, Anna Syberfeldt

Show more

View PDF

Download Full Issue

Under a Creative Commons license

Open access

Monte Carlo methods are also pervasive in artificial intelligence and machine learning. Many important technologies used to accomplish machine learning goals are based on drawing samples from some probability distribution and using these samples to form a Monte Carlo estimate of some desired quantity.

■Warm up - Risk Analysis

View PDF | Access through your institution | Purchase PDF

ELSEVIER

Reliability Engineering & System Safety
Volume 199, July 2020, 106792

Risk analysis of an underground gas storage facility using a physics-based system performance model and Monte Carlo simulation

Zaki Syed, Yuri Lawryshyn

Show more

Add to Mendeley Share Cite

https://doi.org/10.1016/j.ress.2020.106792

Get rights and content

Highlights

- A risk analysis model of underground gas storage facility operational reliability.
- Physics-based performance model combined with Monte Carlo simulation of disruptions.

e work
g Interest
y materials

Recom
Optimal
Reliability
Purch
An integ
Reliability
Purch
Bayesian
Reliability
Purch

Article
Citation:
Citation I
Capture
Readers:
PLUM

■Warm up - Sensitivity

View PDF | Access through your institution | Purchase PDF | Sear

:view

:ts

)

d articles (6)

ELSEVIER

European Journal of Operational Research

Volume 298, Issue 1, 1 April 2022, Pages 229-242

Stochastics and Statistics

Sensitivity estimation of conditional value at risk using randomized quasi-Monte Carlo

Zhijian He

Show more

Add to Mendeley Share Cite

https://doi.org/10.1016/j.ejor.2021.11.013

Get rights and content

Abstract

Conditional value at risk (CVaR) is a popular measure for quantifying portfolio risk. Sensitivity analysis of CVaR is common in risk management and gradient-based optimization algorithms. In this paper, we study the infinitesimal perturbation analysis estimator for CVaR sensitivity using randomized quasi-Monte Carlo (RQMC) simulation.

■Warm up - powerplant

View PDF | Access through **your institution** | Purchase PDF

ISA Transactions
Volume 100, May 2020, Pages 171-184

Research article

Nonlinear robust fault diagnosis of power plant gas turbine using Monte Carlo-based adaptive threshold approach

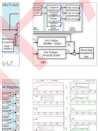
Saeed Amirkhani [a, b], Ali Chaibakhsh [a, b], Ali Ghaffari [c]

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.isatra.2019.11.035 Get rights and content

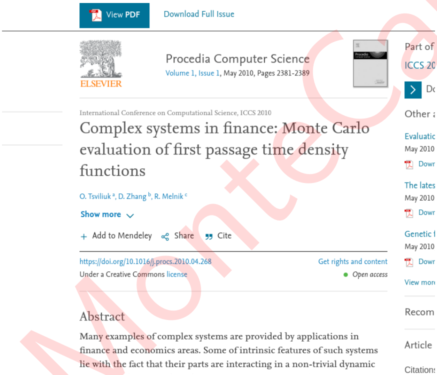
Interest



Highlights

- The adaptive threshold approach is used for the gas turbine fault diagnosis.
- Adaptive threshold bounds are determined based on Monte Carlo simulations.
- The robustness of fault detection is analysed through

■Warm up - Finance



Monte Carlo analysis is useful in risk analysis because many investment and business decisions are made on the basis of one outcome. In other words, many analysts derive one possible scenario and then compare that outcome to the various impediments to that outcome to decide whether to proceed.

■Warm up - Physics

- diffusion Limited Aggregation

■Warm up - what is really it?!!!

- Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle.

- In a Monte Carlo simulation (method) we attempt to follow the 'time dependence' of a model for which change, or growth, does not proceed in some rigorously predefined fashion (e.g. according to Newton's equation of motion) but rather in a stochastic manner which depends on a sequence of random numbers which is generated during the simulation

■Warm up - what is really it?!!!

- Monte Carlo method is used to estimate the possible outcomes of an uncertain event. The Monte Carlo Method was invented by John von Neumann and Stanislaw Ulam during World War II to improve decision making under uncertain conditions. It was named after a well-known casino town, called "Monte Carlo" in Monaco, since the element of chance is core to the modeling approach, similar to a game of roulette.
- Monte Carlo Simulations have assessed the impact of risk in many real-life scenarios, such as in artificial intelligence, stock prices, sales forecasting, project management, and pricing.

■Warm up - What is it?!!!

- When Ulam was in hospital he was playing cards (just for time passing) and got the idea of random sampling. He then applied this idea along with Neuman to solve the problem of "neutron diffusion" in the Manhattan project

- Monte Carlo algorithms are simple, flexible, and scalable. When applied to physical systems, Monte Carlo techniques can reduce complex models to a set of basic events and interactions, opening the possibility to encode model behavior through a set of rules which can be efficiently implemented on a computer.

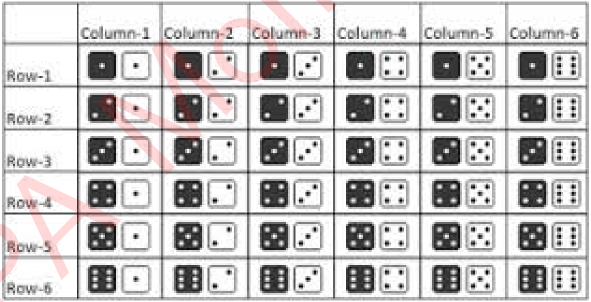
- However he published paper about MC simulation only in 1949 from LANL (Los Alamos National Lab)

■Warm up - What is it?!!!

- Take a random sample of given population. Calculate many different outcomes and their probabilities of occurrence
- This outcome represents the desired results.

■Warm up - What is it?!!!

- Consider a simple example of rolling dice. Assume that you want to determine the probability of rolling a seven using two dice with values one through six. There are 36 possible combinations for the two dice, six of which will total seven, as shown in the following image.



■Warm up - What is it?!!!

- This means that mathematical probability of rolling a seven is six in 36, or 16.67 percent.
- But is the mathematical probability the same as the actual probability? Or are there other factors that might affect the mathematical probability, such as the design of the dice themselves, the surface on which they are thrown, and the technique that is used to roll them?
- To determine the actual probability of rolling a seven, you might physically roll the dice 100 times and record the outcome each time. Assume that you did this and rolled a seven 17 out of 100 times, or 17 percent of the time. Although this result would represent an actual, physical result, it would still represent an approximate result. If you continued to roll the dice again and again, the result would become less and less approximate.

■Warm up - What is it?!!!

- A Monte Carlo simulation is the mathematical representation of this process. It allows you to simulate the act of physically rolling the dice and lets you specify how many times to roll them. Each roll of the dice represents a single iteration in the overall simulation; as you increase the number of iterations, the simulation results become more and more accurate. For each iteration, variable inputs are generated at random to simulate conditions such as dice design, rolling surface, and throwing technique. The results of the simulation would provide a statistical representation of the physical experiment described above.

■Warm up - What is it?!!!

- Bayesian Statistics is fundamentally all about modifying conditional probabilities – it uses prior distributions for unknown quantities which it then updates to posterior distributions using the laws of probability. In fact Bayesian statistics is all about probability calculations!

- MC method is some how based on the Bayesian Statistics

- Bayesian Statistics is a theory in the field of statistics based on the Bayesian interpretation of probability where probability expresses a degree of belief in an event

Random numbers - Introduction

Prof. Dr. Narayan Prasad Adhikari
Central Department of Physics
Tribhuvan University Kirtipur, Kathmandu, Nepal

September 11, 2022

■Warm up!!!

- What are random numbers?
- Can we really get random numbers?
- Pseudorandom numbers
- Generating random numbers in your own laptop

■What are random numbers?

- What is a random number? As the term suggests, a random number is a number chosen by chance i.e., randomly, from a set of numbers.
- Random numbers play vital role in Monte Carlo Methods (simulation).
- The earliest methods for generating random numbers, such as dice, coin flipping and roulette wheels, are still used today, mainly in games and gambling as they tend to be too slow for most applications in statistics and cryptography.

■Who generated first random numbers?

- John von Neuman gave idea to generate random numbers in 1946
- His idea was to start with an initial random seed value, square it, and slice out the middle digits. If you repeatedly square the result and slice out the middle digits, you'll have a sequence of numbers that exhibit the statistical properties of randomness.
- An example: Consider any large numbers say - 2934; square is:8608356; you pick 083 as a random number; square of 83 is 6889; next random number became 88 ...

■Main properties of random numbers

- Good random number generator should be
- (a) random
- (b) reproducible
- (c) portable
- (d) efficient

■Random numbers (RN)- Background

- MC methods are heavily dependent on the fast, efficient production of streams of random numbers.
- Physical processes such as white noise - a random signal having equal intensity at different frequencies, generation from electrical circuits are too slow
- If you are interested in MC you must be able to generate your random numbers (that too in sequence)
- Since such sequences are actually deterministic, the random number sequences we produce in our laptop/computers are only "pseudo-random"
- It is important for you to understand the limitations of pseudo random number generators (PRNG)

■Random numbers (RN)- Background

- In our context - "random numbers— (RN) means "pseudo-random numbers (PRN)"
- These deterministic features of PRN are not always negative.
- For example - for testing a program it is often useful to compare the results with a previous run made using exactly same random numbers.

■Monte Carlo (MC) methods

- MC simulations are subject to both statistical and systematic errors from multiple sources.
- If your RN are of poor quality it leads to systematic errors
- In fact the testing as well as the generation of random numbers remain important problems that have not been fully solved yet. So its for you ....
- As mentioned above RN sequences which are needed in MC should be uniform, uncorrelated, and of extremely long period i.e. do not repeat over quite long intervals.
- Also if you use parallel computing (of course you must to handle large data), you must insure all the random numbers sequences generated are distinct and uncorrelated

■Generation of PRNs- Congruential method

- Most popular method - multiplicative OR congruential method
- Main idea:A fixed number c is chosen along with a given seed and subsequent numbers are generated by simple multiplication

$$X_n = (c \times X_{n-1} + a_0) MOD\ N_{max}$$

where X_n is an integer between 1 and N_{max}.

■Generation of PRNs- Congruential method

- Experience has shown that a good congruential generator is the 32-bit linear congruential (CONG) algorithm:

$$X_n = (16807 \times X_{n-1}) MOD (2^{31} - 1)$$

- Some people call the number "16807" a A Miracle Number
- Even though CONG showed some drawbacks it is still popular being simplest way to generate random numbers

■Generation of PRNs- Congruential method: algorithm

You need to produce random numbers from seed using above formula
Use following algorithm:

1. Start 2. For loop (I mean to produce many random numbers) set count 0
3. Define seed (A large number)
4. start loop (while or any other you like)
5. Calculate ran=16807*seed
6. Set seed equal to ran for the next iteration of the loop
7. Print random number you generated
8. Increase count by 1
9. End program

■Generation of PRNs- Congruential method: algorithm

The code in Python looks like:

```
count=0
seed=1982537
while (count <100):
ran=(16807*seed)%(2**31-1)
seed =ran
print (ran)
count=count+1
```

■Generation of PRNs- Congruential method: algorithm

For following codes each time you must write an algorithm.
You can change the first one to add another one

CWI:Write an algorithm to open a file and write above random numbers in that file.
CWII: Now write two random numbers in the file at a time (say ran1 and ran2)
CWIII: Now convert ran1 and ran2 to lie in the range of (0,1).
CWIV: Plot ran2 vs ran1.
CWV: Check the distribution of random numbers.

■Generation of PRNs-Python random()

Now write a code (python) using following algorithm:

1. Start
2. import random
3. open file
4. start a loop as before
5. get random numbers from python's intrinsic function random()
6. Write them in a file (generate two columns)
7. End loop
8. Close file
9. End program

■Generation of PRNs-Python random()

The code looks like:

```
import random
f = open("rand.dat", "w")
count = 0
while (count < 100):
print (random.random(),random.random())
f.write("{ } { }\n".format(random.random(), random.random()))
count = count + 1
f.close()
```

■Generation of PRNs - Python random()

HWI: Now you compare the distribution of random numbers you generated using congruent method and built in function of python. Compare them and discuss.

I will evaluate this HW for your grading

■Generation of PRNs -Other algorithms

- HW: Now you try to understand at least one more PRNGs algorithm
- Can you convert uniform distribution to gaussian distribution?
- For this: pick any two random numbers x1 and x2 from uniform distribution

$$y_1 = (-2\ln(x_1))^{1/2}\cos(2\pi x_2) \quad (1)$$

$$y_2 = (-2\ln(x_1))^{1/2}\sin(2\pi x_2) \quad (2)$$

HW: Given a sequence of uniformly distributed random numbers y_i show how sequences x_i distributed according to x^2 would be produced.

■HW: Properties of selected RNGs

- The underlying PDF for the generation of random numbers is the uniform distribution, meaning that the probability for finding a number x in the interval $[\ 0, 1)$ is $p(x) = 1.$
- A random number generator should produce numbers which uniformly distributed in this interval.
- Just think about different ways to check this distribution
- One way: by plotting as before.
- Another way: You just find number of random numbers between 0.0 -0.1, 0.1-0.2, ...,0.9-1.0 for say large numbers of random numbers say 100000. You develop a code to read random numbers generated and find RNs between those limits.

■HW: Properties of selected RNGs

- Two additional measures are the s.d. σ and the mean $\mu = \langle x \rangle$.
- For the uniform distribution with N points we have that the average $\langle x^k \rangle$. is

$$\langle x^k \rangle = \frac{1}{N} \sum_{i=1}^{N} x_i^k p(x_i) \tag{3}$$

and taking the limit $N \to \infty$ we have

$$\langle x^k \rangle = \int_0^1 dx\ p(x)\ x^k = \int_0^1\ dx\ x^k = \frac{1}{k+1} \tag{4}$$

as $p(x) = 1$.

$$\therefore \mu = \langle x \rangle = \frac{1}{2} \tag{5}$$

■HW: Properties of selected RNGs

- Similarly standard deviation is

$$\sigma = \sqrt{\langle x^2 \rangle - \mu^2} = \frac{1}{\sqrt{12}} = 0.2886 \tag{6}$$

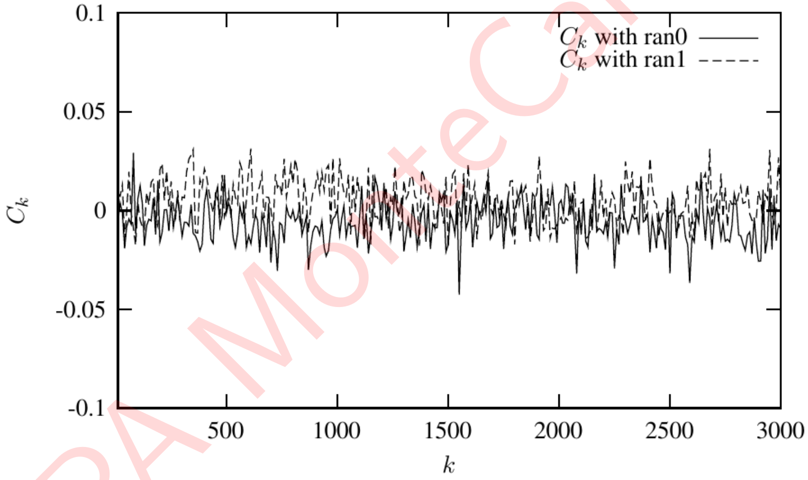
HW: Now you write a code to check your random numbers's distributions. Also evaluate mean and variance of them.

- **Auto-Correlation function:** Since our random numbers, which are typically generated via a linear congruential algorithm, are never fully independent, we can then define an important test which measures the degree of correlation, namely the so-called auto-correlation function C_k

$$C_k = \frac{\langle x_{i+k} x_i \rangle - \langle x_i \rangle^2}{\langle x_i^2 \rangle - \langle x_i \rangle^2} \tag{7}$$

with C_0 =1.The non-vanishing of C_k for $k \neq 0$ means that the random numbers are not independent. The independence of the random numbers is crucial in the evaluation of other expectation values.

■HW: Properties of selected RNGs



HW: Now you calculate auto correlation functions for three different RNs and plot them. Can you explain the fluctuations as shown in above figure.

■HW: Properties of selected RNGs

- The expectation values which enter the definition of C_k are given by

$$\langle x_{i+k} x_i \rangle = \frac{1}{N-k} \sum_{i=1}^{N-k} x_i x_{i+k} \tag{8}$$

Monte Carlo - Basics

Prof. Dr. Narayan Prasad Adhikari
Central Department of Physics
Tribhuvan University Kirtipur, Kathmandu, Nepal

February 4, 2024

■Markov Chain and Master Equation

- Random variables
- Concept of errors
- Estimation of errors
- Markov Chain
- Master Equation

■Random variables

- Consider an elementary event with a countable set of random outcomes, $A_1, A_2, ...A_k$ (e.g. you can consider a rolling dice OR a set of "Khodkhode".)
- You are data scientist so you need to consider this event occurring repeatedly say N times such that $N >>> 1$ and we count how often the outcome A_k is observed (N_k).
- The probabilities p_k for outcome A_k is

$$p_k = \lim_{N\to\infty}\left(\frac{N_k}{N}\right) \tag{1}$$

with $\sum_k p_k = 1$.
Obviously $0 \le p_k \le 1$
You are familiar with conditional probability $P(j/i)$, average of any outcomes of such random events x_i, its variances and so on.

■Statistical errors

- Suppose the quantity A is distributed according to a Gaussian with mean value $\langle A \rangle$ and width σ. We consider n statistically independent observations $\{A_i\}$ of this quantity A.

- An unbiased estimator of the mean $\langle A \rangle$ of this distribution is

$$\bar{A} = \frac{1}{n} \sum_{i=1}^{n} A_i \tag{2}$$

and the standard error of this estimate is

$$\text{error} = \frac{\sigma}{\sqrt{n}} \tag{3}$$

■Statistical errors

- The variance is obtained from mean square deviation

$$\delta \bar{A}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\delta A_i\right)^2 = \bar{A}^2 - \left(\bar{A}\right)^2 \tag{4}$$

The expectation value of this quantity is easily related to $\sigma^2 = \langle A^2 \rangle - \langle A \rangle^2$ as

$$\langle \delta \bar{A}^2 \rangle = \sigma^2 (1 - 1/n) \tag{5}$$

$$\therefore \text{error} = \sqrt{\frac{\delta \bar{A}^2}{(n-1)}} = \sqrt{\frac{\sum_{i=1}^{n} \left(\delta A_i\right)^2}{(n(n-1))}} \tag{6}$$

■Ingredients of MC

- As mentioned before there are at least four ingredients which are crucial in order to understand the basic MC strategy. (i) Random variables
  (ii) Probability distribution functions (PDF),
  (iii) Moments of a PDF (iv) and pertinent variance σ

■Random Variables

- Let us first demistify the somewhat obscure concept of a random variable. The example we choose is the classic one, the tossing of two dice, its outcome and the corresponding probability. In principle, we could imagine being able to determine exactly the motion of the two dice, and with given initial conditions determine the outcome of the tossing. Alas, we are not capable of pursuing this ideal scheme. However, it does not mean that we do not have a certain knowledge of the outcome. This partial knowledge is given by the probablity of obtaining a certain number when tossing the dice. To be more precise, the tossing of the dice yields the following possible values

$$[\,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.] \tag{7}$$

■Random Variables

- These values are called the domain. To this domain we have the corresponding probabilities

$$[\ 1/36, 2/36, 3/36, 4/36, 5/36, 6/36, 5/36, 4/36, 3/36, 2/36, 1/36] \tag{8}$$

- These values are called the domain. To this domain we have the corresponding probabilities
- The numbers in the domain are the outcomes of the physical process tossing the dice. We cannot tell beforehand whether the outcome is 3 or 5 or any other number in this domain. This defines the randomness of the outcome, or unexpectedness or any other synonimous word which encompasses the uncertitude of the final outcome.

Random Variables

- The only thing we can tell beforehand is that say the outcome 2 has a certain probability. If our favorite hobby is to spend an hour every evening throwing dice and registering the sequence of outcomes, we will note that the numbers in the above domain

$$[\ 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.] \quad (9)$$

appear in a random order.
after (say) 11 throws the results may look like

$$[\ 10, 8, 6, 3, 6, 9, 11, 8, 12, 4, 5] \quad (10)$$

- Eleven new attempts may results in a totally different sequence of numbers and so forth. Repeating this exercise the next evening, will most likely never give you the same sequences. Thus, we say that the outcome of this hobby of ours is truely random.

■Random Variables

- *Random variables are hence characterized by a domain which contains all possible values that the random value may take. This domain has a corresponding PDF.*

■MC Illustration - Integration

- Consider an integration

$$I = \int_0^1 f(x)dx \simeq \sum_{i=1}^{N} w_i f(x_i) \tag{11}$$

where w_i are the weights determined by specific integration methods like Trapeziod, Simpson etc. In the crudest approach here in MC integration we set up $w_i = 1$ then above eq becomes

$$I = \int_0^1 f(x)dx \simeq \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{12}$$

Now introduce the concept of the average of the function f for a given PDF $p(x)$ as

$$\langle f \rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i)p(x_i) \tag{13}$$

■MC Illustration - Integration

Now identify $p(x_i) = 1$ with the uniform distribution when $x \in [0, 1)$ and zero for all other values of x. Then

$$I = \int_0^1 f(x)dx \simeq \frac{1}{N}\sum_{i=1}^{N} f(x_i) \simeq \langle f \rangle \tag{14}$$

Similarly the variance (which is also important in MC methods) is

$$\sigma_f^2 = \frac{1}{N}\sum_{i=1}^{N} \left(f(x_i) - \langle f \rangle\right)^2 p(x_i) \tag{15}$$

After inserting value of $p(x_i)$ we get

$$\sigma_f^2 = \frac{1}{N}\sum_{i=1}^{N} f^2(x_i) - \left(\langle f \rangle\right)^2 \tag{16}$$

■MC Illustration - Integration:Algorithm

- Choose the number of Monte Carlo samples N.
- Perform a loop over N and for each step generate a a random number x_i in the interval [0, 1] through a call to a random number generator. Translate the random numbers to other required interval if it needs.
- Use this number to evaluate $f(x_i)$.
- Evaluate the contributions to the mean value and the standard deviation for each loop.
- After N samples calculate the final mean value and the standard deviation.

■MC Illustration - Integration

- As an example evaluate following by MC method:

$$I = \int_0^1 \exp(x)dx \tag{17}$$

and

$$I = \int_1^3 \exp(x)dx \tag{18}$$

Also Compare the final results with the correct and hence estimate the errors.

■MC Illustration - Integration

```
import random
import numpy as np
import math
a = 0.
b = 1.
integral = 0.0
i=0
while i<1000:
x=random.random()
integral += math.exp(x)
i=i+1
ans=integral*(b-a)/float(N)
print ("The value calculated by monte carlo integration is {
}".format(ans))
```

HW: You also estimate it following above algorithm. Further estimate integral for different set of random numbers and hence estimate σ_N.

■MC Illustration - Estimate π

1. Initialize circle points, square points and interval to 0.
2. Generate random point x.
3. Generate random point y.
4. Calculate d = x*x + y*y.
5. If d <= 1, increment circle points.
6. Increment square points.
7. Increment interval.
8. If increment < NOOFITERATIONS, repeat from 2.
9. Calculate pi = 4*(circle points/square points).
10. Terminate.

Also follow the discussion in my lecture

■HW: Estimate π

1. Estimate value of π also from

$$\pi = \int_0^1 4 \frac{dx}{1+x^2} \tag{19}$$

2. What we did in previous slide was to estimate value of π from area of a circle in 2 dimensions. Can you think of similar methods for higher dimensions? You may need volume of a hypersphere of radius R in n dimensions:

$$V_n(R) = \frac{\pi^{n/2}}{\left(\frac{n}{2}\right)!} R^n \tag{20}$$

what you did in 2D is ust a special case of above equation in n=2.

■Concept of Importance Sampling

- Till now we discussed about 'simple sampling' of MC
- In principle, MC integrations and other simulations can be performed using the simple sampling techniques we discussed till now. Unfortunately most of the saples produced in this fashion will contribute relatively little to the equilibrium (time independent) averages and more sophisticated methods are required if we are to obtain results of sufficient accuracy to be useful.
- One of such a methods is "Importance Sampling". For this we need to discuss change of variables.

■Concept of Importance Sampling

- With improvements we think of a smaller variance and the need for fewer Monte Carlo samples, although each new Monte Carlo sample will most likely be more times consuming than corresponding ones of the brute force method (Simple sampling). For this we consider two topics.

- The first topic deals with change of variables, and is linked to the cumulative function $P(x)$ of a PDF $p(x)$. Obviously, not all integration limits go from x $= 0$ to x $= 1$, rather, in DATA Science we are often confronted with integration domains like $x \in [0, \infty]$ or $x \in [-\infty, \infty]$ etc. Since all random number generators give numbers in the interval $x \in [0, 1]$, we need a mapping from this integration interval to the explicit one under consideration.

■Concept of Importance Sampling

- The next topic deals with the shape of the integrand itself. Let us for the sake of simplicity just assume that the integration domain is again from x = 0 to x = 1. If the function to be integrated $f(x)$ has sharp peaks and is zero or small for many values of $x \in [0, 1]$, most samples of $f(x)$ give contributions to the integral I which are negligible. As a consequence we need many N samples to have a sufficient accuracy in the region where $f(x)$ is peaked. What do we do then? We try to find a new PDF $p(x)$ chosen so as to match $f(x)$ in order to render the integrand smooth. The new PDF $p(x)$ has in turn an x domain which most likely has to be mapped from the domain of the uniform distribution.

■ Importance Sampling -Change of variables

- Consider uniform distribution

$$\begin{aligned} p(x)dx =& dx \text{ (for } 0 \leq \text{x} \leq 1) \\ =& 0 \text{ else} \end{aligned} \tag{21}$$

with $p(x) = 1$ and satisfying

$$\int_{-infty}^{\infty} p(x)dx = 1 \tag{22}$$

All random number generators provided in the program library generate numbers in this domain. When we attempt a transformation to a new variable $x \to y$ we have to conserve the probability

$$p(y)dy = p(x)dx \tag{23}$$

which for the uniform distribution implies

$$p(y)dy = dx \tag{24}$$

Importance Sampling -Change of variables

Let us assume that $p(y)$ is a PDF different from the uniform PDF $p(x) = 1$ with $x \in [0, 1]$. If we integrate the last expression we arrive at

$$x(y) = \int_0^y p(y')dy' \tag{25}$$

which is nothing but the cumulative distribution of $p(y)$, i.e.

$$x(y) = P(y) = \int_0^y p(y')dy' \tag{26}$$

This is an important result which has consequences for eventual improvements over the brute force Monte Carlo.

Change of variables- an example

Suppose we have the general uniform distribution

$$\begin{aligned} p(y)dy =& \frac{dy}{b-a} \text{ (for a} \leq \text{y} \leq \text{b)} \\ =& 0 \text{ else} \end{aligned} \tag{27}$$

If we wish to relate this distribution to the one in the interval $x \in [0, 1]$ we have

$$p(y)dy = \frac{dy}{b-a} \text{ (for a} \leq \text{y} \leq \text{b)} = dx \tag{28}$$

and integrating we obtain the cumulative function

$$x(y) = \int_0^y \frac{dy'}{b-a} \tag{29}$$

yielding

$$y = a + (b-a)x \tag{30}$$

Importance Sampling

- With the aid of the above variable transformations we address now one of the most widely used approaches to Monte Carlo integration, namely importance sampling. It will be helpful to sample a function which has peak as we need to consider many more sampling points near the peak.

- Let us assume that $p(y)$ is a PDF whose behavior resembles that of a function F defined in a certain interval $[a, b]$. The normalization condition is

$$\int_a^b p(y)dy = 1 \tag{31}$$

We can rewrite our integral as

Importance Sampling

$$I = \int_a^b F(y)dy = \int_a^b p(y)\frac{F(y)}{p(y)}dy \tag{32}$$

Since random numbers are generated for the uniform distribution $p(x)$ with $x \in [0,1]$, we need to perform a change of variables $x \to y$ through

$$x(y) = \int_a^y p(y')dy' \tag{33}$$

where we used

$$p(x)dx = dx = p(y)dy \tag{34}$$

If we can invert $x(y)$, we find $y(x)$ as well. With this change of variables we can express the integral of Eq. 32 as

$$I = \int_a^b p(y)\frac{F(y)}{p(y)}dy = \int_a^b \frac{F(y(x))}{p(y(x))}dx \tag{35}$$

■ Importance Sampling

meaning that a Monte Carlo evalutaion of the above integral gives

$$\int_a^b \frac{F(y(x))}{p(y(x))} dx = \sum_{i=1}^{N} \frac{F(y(x_i))}{p(y(x_i))} \tag{36}$$

The advantage of such a change of variables in case $p(y)$ follows closely F is that the integrand becomes smooth and we can sample over relevant values for the integrand. It is however not trivial to find such a function p.

The conditions on p which allow us to perform these transformations are

1. p is normalizable and positive definite,
2. it is analytically integrable and
3. the integral is invertible, allowing us thereby to express a new variable in terms of the old one.

Important Note

- The average is over y(x) distribution.

Therefore above equation 35 can be rewritten as

$$I = \int_a^b p(y)\frac{F(y)}{p(y)}dy = \int_a^b p(y)\left\{\frac{F(y)}{p(y)}\right\}dy = E_{p(y)}\left\{\frac{F(y)}{p(y)}\right\} \quad (37)$$

is actually expectation value of

$$\left\{\frac{F(y)}{p(y)}\right\}$$

with distribution $p(y)$.

Please note that the average is over the distribution $p(y)$ not over $p(x)$.

Therefore in the importance sampling integration you first find the p(y) corresponding to $p(x) \in [0, 1]$. Then find average 36 with distribution $p(y)$. See Example below.

■ Importance Sampling - Examples

(1) Consider the integral

$$I = \int_0^1 \exp(-x^2) dx \tag{38}$$

Evaluate I using (i) brute force (simple sampling) MC with $p(x) = 1$ and (ii) importance sampling with $p(x) = a\exp(-x)$.

Improtant Note: You first write Algorithm in each case then write code in python language following the Hints

(a) Obtain average of $\exp(-x^2)$ for $x \in [0, 1]$

(b) Find its variance too.

These are results of Simple sampling.

for importance sampling:

(c) Find $p(y)$ corresponding to $p(x) = a\exp(-x)$ from equation 33 where a is normalization constant.

(d) Then find the expectation value of $\left[\frac{\exp(-x^2)}{a\exp(-x)}\right]$ with distribution $p(y)$. (e) Find variance also. (f)Compare both errors or variances and comments on your results.

- **Solution of above example (Importance sampling):**

$$I = \int_0^1 \exp(-x^2)dx \tag{39}$$

with chosen pdf (probability distribution function) $p(x) = a\exp(-x)$ such that $x \in (0,1)$ and

$$\int_0^1 p(x)dx = \int_0^1 a\exp(-x)dx = 1.$$

resulting $a = \frac{e}{e-1}$.

$$\implies p(x) = \frac{\exp(-x)}{1-\frac{1}{e}} \tag{40}$$

■ Monte Carlo - More on above examples

Also check whether p(x) fulfills the criteria for pdf. for this we find $\frac{F(0)}{p(0)}$ and $\frac{F(1)}{p(1)}$. They have to be equal.
Now

$$\frac{F(0)}{p(0)} = \frac{e}{e-1} = \frac{F(1)}{p(1)} \tag{41}$$

Since our pdf fulfills the criteria lets find $y(x)$. For this we perform then the change of variables (via the Cumulative distribution function)

$$y(x) = \int_0^x p(x')dx' = \int_0^x dx' \exp\left(-x'\right) \times \frac{e}{e-1} \tag{42}$$

$$\Longrightarrow \; y(x) = \left(\frac{e}{e-1}\right)\left(1 - e^{-x}\right) \tag{43}$$

after solving for x we get;

$$\Longrightarrow \; x = -\ln\left(1 - y(1 - e^{-1})\right) \tag{44}$$

which gives $y = 0$ for $x = 0$ and $y = 1$ for $x = 1$ as required for the property of pdf.

Now we need to find expectation value of

$$\left[\frac{\exp(-x^2)}{\exp(-x)}\right]$$

with distribution $y(x)$. That is we need to evaluate

$$\int_0^1 \exp(-x^2)dx = \left\langle \frac{\exp(-y^2(x))}{\exp(-y(x))} \right\rangle$$

■ Monte Carlo - More on above examples

Algorithm:

1. Start
2. import required libraries like random, numpy ..
3. Define n, functions, initialize summ etc
4. start loop over n
5. generate random numbers $x \in (0,1)$
6. define function $y(x)$ using above formula from x as
$y(x) = \left(\frac{e}{e-1}\right)(1 - e^{-x})$
7. Get sum of the function $\frac{\exp(-y^2(x))}{\exp(-y(x))}$
8. close the loop
9. Find the integration value i.e. $\left\langle \frac{\exp(-y^2(x))}{\exp(-y(x))} \right\rangle$
10. You also find variance and hence the error.

The python code is in next page.

Pl note that the code contains the simple sampling also. Compare both results.

■ Monte Carlo - More on above examples

In [3]:
```python
import numpy as np
import random
from scipy.stats import norm
#Define the number of MC steps
n=10000

# Standard (simple sampling Monte Carlo
sum=0.0
i=0
summ=0.
while i<n:
    x = random.random()
    g = np.exp(-x**2)
    sum=sum+g
    summ=summ+g*g
    i=i+1
MC=sum/n
std_MC = summ/n-MC**2
print('Standard Monte-Carlo estimate of given function: ' + str(MC))
print('Standard deviation of simple sampling Monte Carlo: ' + str(std_MC))
print(' ')
#Importance sampling
i=0
sum=0.0
summ=0.0
while i<n:
    x = random.random()
    y=(np.exp(1.)/(np.exp(1.0)-1.)*(1-np.exp(-x)))
    f = np.exp(-y**2)/np.exp(-y)
    sum=sum+f
    summ=summ+f*f
    i=i+1
meanf=sum/n
MCI=meanf*(1.0-np.exp(-1.0))
std_MCI=summ/n-meanf**2
print('Importance Sampling Monte-Carlo estimate of given function: ' + str(MCI))
print('Standard deviation of Impostance sampling  Monte Carlo: ' + str(std_MCI))
print(' ')
```

```
Standard Monte-Carlo estimate of given function: 0.7469875583049324
Standard deviation of simple sampling Monte Carlo: 0.04042059388000929

Importance Sampling Monte-Carlo estimate of given function: 0.7442631266953907
Standard deviation of Impostance sampling  Monte Carlo: 0.00778395938152987
```

■ Importance Sampling - Examples

Now compare the variances OR errors due to simple sampling and importance sampling both.
(2) Consider the integral

$$I = \int_0^{\pi} \frac{1}{x^2 + \cos^2 x} dx \tag{45}$$

Evaluate I using importance sampling with $p(x) = a \exp(-x)$ where a is a constant.
Improtant Note: You first write Algorithm then write code in python language following the Algorithm. Can you find the value of a which minimizes the variance.

■ Monte Carlo Integration - Homework

Consider

$$I = \int_0^1 dx_1 \int_0^1 dx_2 ... \int_0^1 dx_n g(x_1, x_2, ..., x_n) \quad (46)$$

with x_i defined in the interval $[a_i, b_i]$ we would typically need a transformation of variables of the form

$$x_i = a_i + (b_i - a_i) * t_i$$

if we were to use the uniform distribution on the interval $[0, 1]$.
As an example, evaluate

$$I = \int_{-5}^{5} d\mathbf{x} \; d\mathbf{y} \; g(\mathbf{x}, \; \mathbf{y}) \quad (47)$$

with

$$g(\mathbf{x}, \; \mathbf{y}) = \exp\left(-\mathbf{x}^2 - \mathbf{y}^2 - (\mathbf{x} - \mathbf{y})^2/2\right)$$

Again you write Algorithm and code.

■ Monte Carlo Acceptance-Rejection method

It is simple and and appealing method after von Neumann. Assume that we are looking at an interval $x \in [a, b]$, this being the domain of the PDF $p(x)$. Suppose also that the largest value our distribution function takes in this interval is M , that is

$$p(x) \leq M \qquad x \in [a, b] \tag{48}$$

Then we generate a random number x from the uniform distribution for $x \in [a, b]$ and a corresponding number s for the uniform distribution between $[0, M]$. If

$$p(x) \geq s \tag{49}$$

we accept the new value of x, else we generate again two new random numbers x and s and perform the test in the latter equation again.

■ Acceptance-Rejection method: an example

- Actually Acceptance-Rejection sampling is the conceptually simplest way to generate samples of some arbitrary probability function without having to do any transformations.
- No integration, no trickery, you simply trade computational efficiency away to keep everything as simple as possible.
- Consider an example:

$$f(x) = 1.2 - x^4$$

  You want to sample points in the given function for $x \in (0, 1)$. Well, if you integrate $f(x)$ between 0 and 1 you get 1.

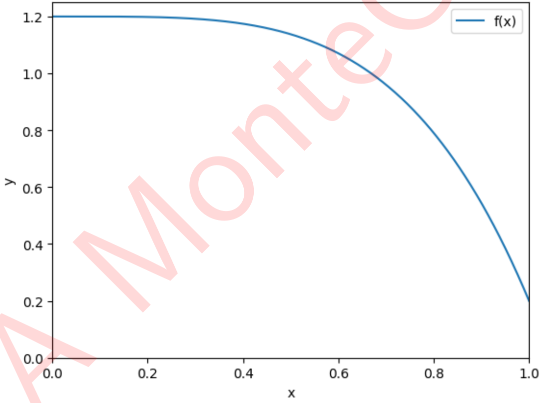
■ Acceptance-Rejection method: an example

Let's first plot the function just to see how does it look like

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
  return 1.2 - x**4
xs = np.linspace(0, 1, 1000)
ys = f(xs)

plt.plot(xs, ys, label="f(x)") plt.xlim(0, 1), plt.ylim(0, 1.25),
plt.xlabel("x"), plt.ylabel("y"), plt.legend();
```

■ Acceptance-Rejection method: an example



■ Acceptance-Rejection method: an example

- **Algorithm**
- Pick two random numbers. One for x (between 0 and 1), one for y (between 0 and 1.2).
- If the y value we randomly picked is less than f(x), keep it, otherwise go back to above step

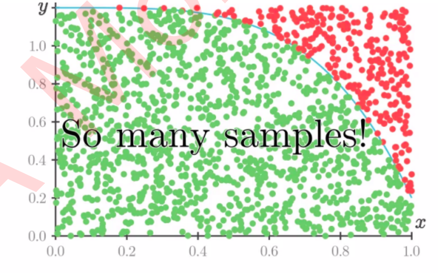


Figure: Green points accepted and red one rejected

■ Acceptance-Rejection method: an example

- So you can see that the reason this is so straightforward is that we get samples according to the function by simply throwing away the right number of samples when the function has a smaller value.

- In our function, this means if we get a small x value, we'd normally keep the sample (and indeed the distribution is pretty flat for $x < 0.5$), but for values close to $x = 1$, we'd throw them out most of the time

Acceptance-Rejection method: an example

```
def sample(function, xmin=0, xmax=1, ymax=1.2):
  while True:
    x = np.random.uniform(low=xmin, high=xmax)
    y = np.random.uniform(low=0, high=ymax)
    if y < function(x):
      return x

samps = [sample(f) for i in range(10000)]
plt.plot(xs, ys, label="f(x)")
plt.hist(samps, density=True, alpha=0.2, label="Sample
distribution")
plt.xlim(0, 1), plt.ylim(0, 1.4), plt.xlabel("x"), plt.ylabel("y"),
plt.legend();
```

■ Acceptance-Rejection method: an example