Random numbers - Introduction

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

September 7, 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:

```
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

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

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