Lecture 6
CH-4114
Molecular Simulation

"Everything that living things do can be understood in terms of the jigglings and wigglings of atoms."

- Richard P. Feynman

By
Dr. Susmita Roy
IISER-Kolkata

Central Limit Theorem

- The Central Limit Theorem, as the name signifies, is central to the entire subject of probability theory. We recommend the students consult the books by Feller (Volume 1) and Kai Lai Chung for a detailed description. In short if S is a sum of n number of random variable x_i and if x_i are weakly correlated then the probability distribution P(S) has a Gaussian nature.
- In Statistical Mechanics often the above condition of weak correlation among random variables is satisfied. For example, the total energy of the system, E, is the sum of kinetic and potential energies of individual particles and the individual energies are weakly correlated among themselves. Thus, the total energy E is almost invariably Gaussian with the standard deviation being given by the specific heat.

Mathematically, if

$$S = \sum_{i=1}^{N} x_i,$$

$$< S >= \mu$$

then,

$$P_{\mu,\sigma}(S) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where, $P_{\mu,\sigma}(S)$ = probability distribution function of S

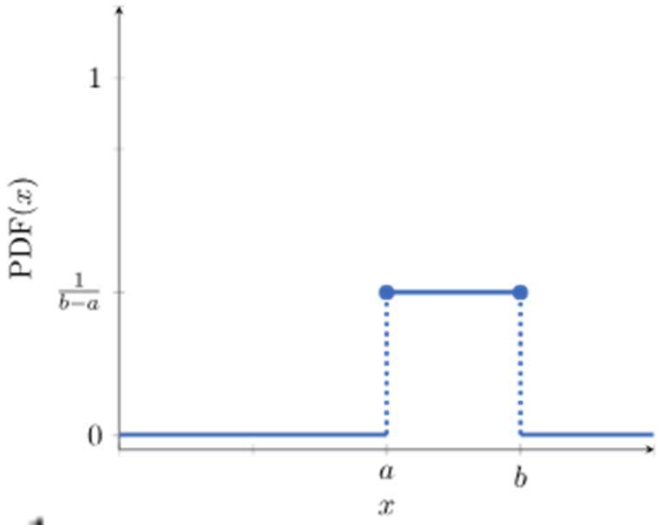
σ = standard deviation = $\left(\sqrt{\langle S^2 \rangle - \langle S \rangle^2}\right)$

Uniformity and Normality

The Uniform Distribution

The uniform distribution also takes the name of the rectangular distribution, because of the peculiar shape of its probability density function:

Within any continuous interval [a, b], which may or not include the extremes, we can define a uniform distribution U(a, b). This is the distribution for which all possible arbitrarily small intervals ε Є [a, b], with or without extremes, have the same probability P(ε) = 1/ (b-a) of occurrence. We can verify that this is a proper probability desity function by calculating the area under the curve:



$$Pr[a \leq x \leq b] = \int_a^b \mathrm{PDF}(x)dx = (b-a) \times \frac{1}{b-a} = 1$$

The Normal Distribution

The normal distribution, instead, is a distribution characterized by this probability density function:

$$\mathrm{PDF}(x|\sigma,\mu) = \frac{1}{\sigma\sqrt{2\pi}} \times e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

here, σ and μ indicate, respectively, the standard deviation and the mean of the distribution. This is its corresponding chart, for σ=1 and μ=0:

We can see that the values contained in a normal distribution aren't equally likely, as the corresponding values of the uniform distribution were. This is due to two reasons.

The first reason is that the uniformly-distributed values with non-zero probability are all contained in a finite interval, while those of the normal distributions aren't. We noted that the values with a non-zero probability in the normal distribution occupy the whole domain (-∞, +∞). This isn't true for the uniform distribution, so we'll need some kind of transformation to convert the finite interval [a, b] to the other.

The second reason is that all values in discrete uniform distributions have the same probability of being drawn. For discrete normal distributions, instead, any two values $x_1 \neq x_2$ have corresponding probabilities $P(x_1) \neq \mathrm{P}(x_2)$ different from one another. Of course, with the exception of the case in which $(x_1 + x_2)/2 = \mu$.

Approach-1: CLT

Old-school hack for generating Gaussian-distributed random numbers using uniform random numbers.

little snippet

```
def gauss(self):
   return np.sum(np.random.random(12)) - 6.0
```

np.random.random(12)This generates 12 independent random numbers.Each number comes from a uniform distribution between 0 and 1.Example output:[0.45, 0.12, 0.89, 0.33, ...]

```
np.sum(...)
```

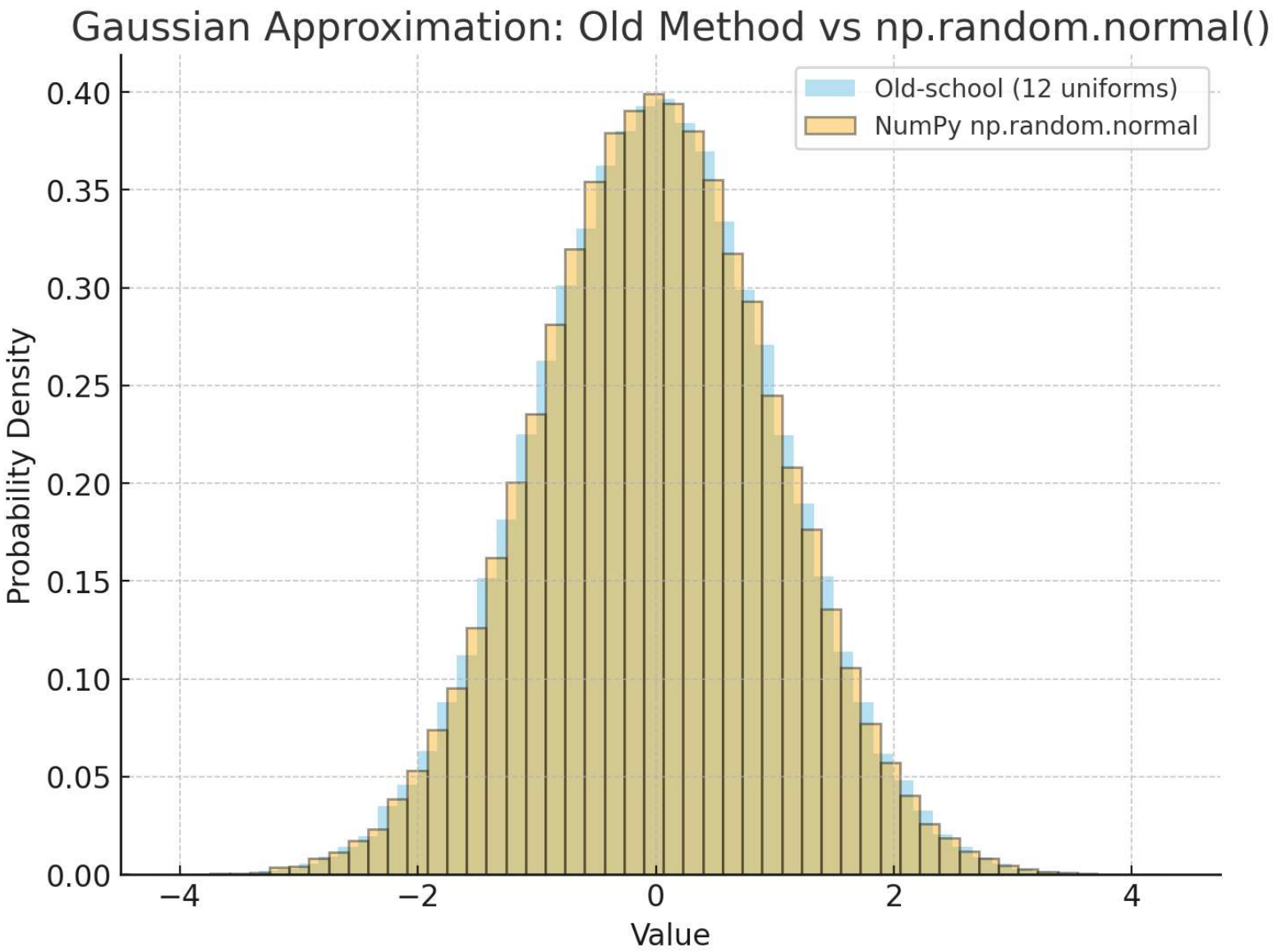
All 12 numbers are added together.If each number is uniform between 0 and 1, the expected mean of one number is 0.5.So, the expected sum of 12 numbers is:E[sum]=12×0.5=6.0

Without subtraction, the distribution would be centered around 6.0.Subtracting 6.0 shifts it so the mean is 0.0, giving a zero-centered distribution.

Why does this approximate a Gaussian? This statistics magic kicks in because of The Central Limit Theorem (CLT)

Summary→

This is a *cheap* Gaussian generator:
1.Make 12 uniform(0,1) numbers
2.Sum them → CLT makes it "bell-shaped"
3.Subtract mean to center at zero


Gaussian Approximation: Old Method vs np.random.normal()
Old-school (12 uniforms)
NumPy np.random.normal
Probability Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4
Value

How to plot a histogram: Demo

```
# --------------------------
# 2. Define bin width and bin edges
# --------------------------
bin_width = 2                          # width of each bin
data_min, data_max = min(data), max(data)         # range of data
num_bins = int((data_max - data_min) / bin_width)    # number of bins
bin_edges = np.linspace(data_min, data_max, num_bins + 1) # Create bin edges from min to max
```

data = [21, 22, 23, 25, 28, 29, 31, 33, 34, 36]
bin_width = 5

- **Minimum (data_min) = 21**
- **Maximum (data_max) = 36**
- Range = 36 − 21 = 15
- Number of bins = 15 / 5 = 3

So we have **3 bins**:

- Bin 0 → [21, 26)
- Bin 1 → [26, 31)
- Bin 2 → [31, 36] (last bin includes max)

Final bin counts

- Bin 0 (21–25.9): **4 values** → [21, 22, 23, 25]
- Bin 1 (26–30.9): **2 values** → [28, 29]
- Bin 2 (31–36): **4 values** → [31, 33, 34, 36]

Approach-2

Box-Muller Transform

The Definition of the Algorithm

We can therefore identify an algorithm that maps the values drawn from a uniform distribution into those of a normal distribution. The algorithm that we describe here is the Box-Muller transform. **This algorithm is the simplest one to implement in practice**, and it performs well for the pseudorandom generation of normally-distributed numbers.

The algorithm is very simple. We first start with two random samples of equal length, u_1 and u_2, drawn from the uniform distribution $U(0, 1)$. Then, we generate from them two normally-distributed random variables z_1 and z_2. Their values are:

- $z_1 = \sqrt{-2\ln(u_1)}\cos(2\pi u_2)$
- $z_2 = \sqrt{-2\ln(u_1)}\sin(2\pi u_2)$

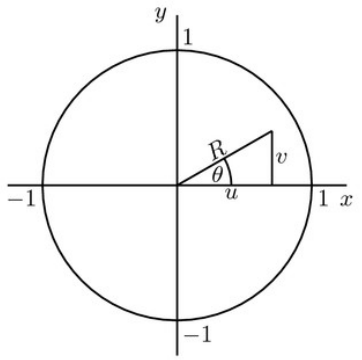
Geometric Interpretation

Suppose U_1 and U_2 are independent samples chosen from the uniform distribution on the unit interval $(0, 1)$. Let

$$Z_0 = R\cos(\Theta) = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

and

$$Z_1 = R\sin(\Theta) = \sqrt{-2\ln U_1}\sin(2\pi U_2).$$



$$R^2 = u^2 + v^2$$
$$\cos\theta = \frac{u}{R}$$
$$\sin\theta = \frac{v}{R}$$

Then Z_0 and Z_1 are independent random variables with a standard normal distribution.

The derivation[5] is based on a property of a two-dimensional Cartesian system, where X and Y coordinates are described by two independent and normally distributed random variables, the random variables for R^2 and Θ (shown above) in the corresponding polar coordinates are also independent and can be expressed as

$$R^2 = -2\cdot\ln U_1$$

and

$$\Theta = 2\pi U_2.$$

Because R^2 is the square of the norm of the standard bivariate normal variable (X, Y), it has the

Random Velocity Generation for our MD code

1. Generating initial velocities from a Gaussian distribution
2. Later comparing to the Maxwell–Boltzmann distribution

MD Initialization goal

- At initialization, you only care that each velocity component follows the right Gaussian statistics so that the total kinetic energy matches the target temperature.
- During the simulation, you want to verify that your evolving particle velocities actually follow the correct equilibrium distribution — which in 3D means Maxwell–Boltzmann for speeds.
- Step 1: Set each vx, vy, vz ~ Gaussian(0, σ^2) with
- $\sigma^2 = k_B T/m$
- Step 2: Run MD and measure the speed distribution → check against Maxwell–Boltzmann.