UPPSALA
UNIVERSITET

Department of Information Technology

Scientific Computing for Data Analysis

Davoud Mirzaei

Lecture 2: Monte Carlo Method - II

Agenda

- Well-known distributions
- Monte Carlo integration
- Inverse transform method (ITM)

General structure of Monte Carlo (MC) algorithm

$$
\begin{aligned}
&\textit{Input } N: \ (\textit{Number of observations})\\
&\textbf{for } k = 1:N\\
&\quad \textit{perform one stochastic simulation/process}\\
&\quad \textit{result}[k] = \textit{result of the simulation}\\
&\textbf{end}\\
&\textit{FinalResult} = \textit{mean}(\textit{result}) \textit{ or other statistical calculation}
\end{aligned}
$$

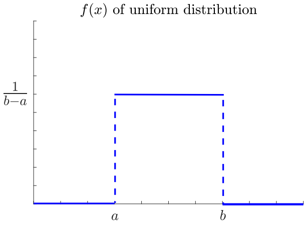
- MC works with random numbers/stochastic (random) processes
- Random numbers are generated (sampled) from different distributions

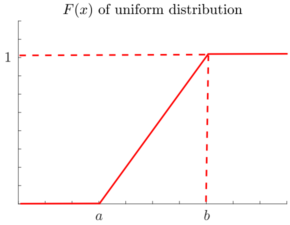
Uniform distribution: We write $X \sim \mathcal{U}(a,b)$, the pdf of X is

$$f(x) = \frac{1}{b-a}\begin{cases}1, & x \in [a,b]\\ 0, & x \notin [a,b]\end{cases}$$

The cdf of X is

$$F(x) = \mathbb{P}(X \leqslant x) = \int_{-\infty}^{x} f(s)ds = \begin{cases}0, & x < a\\ \int_a^x f(s)ds = \frac{x-a}{b-a}, & x \in [a,b]\\ 1, & x > b\end{cases}$$



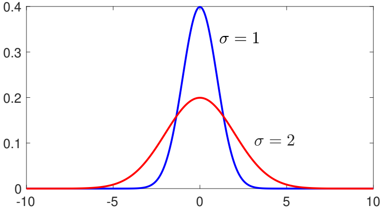


What is the area under the pdf graph?

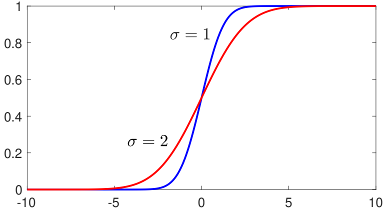
Normal distribution: We write $X \sim \mathcal{N}(\mu, \sigma^2)$, and read X is a normal variable with mean μ and variance σ^2 (standard deviation σ). The pdf of X is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), \quad x \in (-\infty, \infty) \tag{1}$$

Plots of pdf $f(x)$ and cdf $F(x)$ for $\mu = 0$ and two different values for σ:



pdf of normal dist.



cdf of normal dist.

Some well-known probability distributions

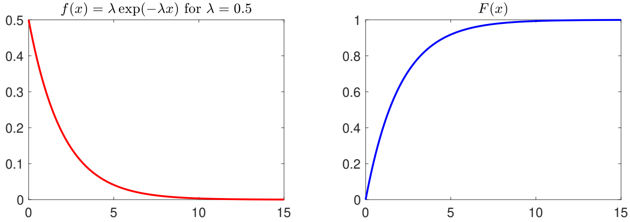
Exponential distribution: We write $X \sim \mathcal{E}xp(\lambda)$, its pdf f is given by

$$f(x) = \lambda \mathrm{e}^{-\lambda x}, \quad x \in [0, \infty), \quad \lambda > 0$$

and its cdf F by

$$F(x) = \int_0^x \lambda \mathrm{e}^{-\lambda s} \mathrm{d}s = 1 - \mathrm{e}^{-\lambda x}, \quad x \geqslant 0.$$

The mean or expectation of X is $\mu = \frac{1}{\lambda}$ and its variance is $\sigma^2 = \frac{1}{\lambda^2}$.

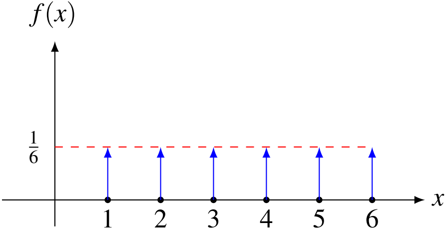


Models things like waiting times, time between earthquakes, time between calls

Discrete distributions:

- Previous distributions are all examples of continuous distributions
- There are also some discrete distributions, Example: dice,

$$f(x) = \begin{cases} \frac{1}{6}, & x \in \{1,2,3,4,5,6\} \\ 0, & \text{otherwise} \end{cases}$$

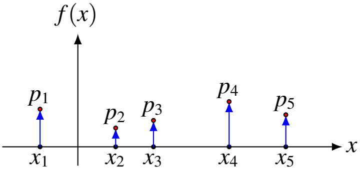


What does the cdf look like?

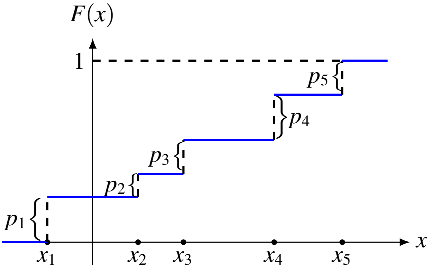
General $\mathcal{D}$**iscrete** $\mathcal{D}$**istributions**: $X \sim \mathcal{DD}([x_1, \ldots, x_m], [p_1, \ldots, p_m])$

$$f(x_k) = p_k, \quad k = 1, 2, \ldots, \quad \sum_{k=1}^{\infty} p_k = 1.$$

x_k	x_1	x_2	$\ldots$	x_m
p_k	p_1	p_2	$\ldots$	p_m



$$F(x) = \sum_{k: x_k \le x} p_k$$



Random numbers in NumPy

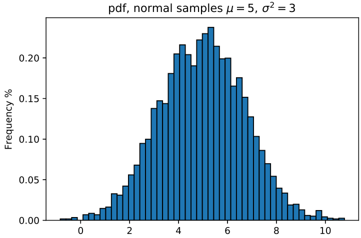
Use `rand` and `randn` to sample from uniform and normal distributions, respectively:

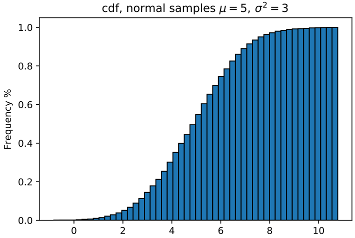
```
# a uniform random number in interval [0,1)
x = numpy.random.rand()

# a (N x 1) array of uniform random numbers in [0,1)
x = numpy.random.rand(N,1)

# a uniform random number in interval [a,b)
x = (b-a)*numpy.random.rand() + a

# a standard normal random number (mu = 0, sigma = 1)
x = numpy.random.randn()

# a (N x 1) array of standard normal numbers (mu = 0, sigma = 1)
x = numpy.random.randn(N,1)

# a normal random number with mean mu and standard deviation s)
x = mu + s*numpy.random.randn()
```

Note: If $X \sim \mathcal{U}(0,1)$ then $a+(b-a)X \sim \mathcal{U}(a,b)$
If $X \sim \mathcal{N}(0,1)$ then $\mu+\sigma X \sim \mathcal{N}(\mu,\sigma^2)$

Example: Normal samples

```python
import numpy as np
import matplotlib.pyplot as plt
N = 5000
mu, s = 5, np.sqrt(3)
X = mu + s*np.random.randn(N,1)
# histogram plot (pdf)
plt.figure()
plt.hist(X, bins=50, histtype='bar',edgecolor='black', density='true')
plt.title('pdf, normal samples $\mu=5,\, \sigma^2 = 3$')
plt.xlabel('$x$'); plt.ylabel('Frequency $\%$')

# histogram plot (cdf)
plt.figure()
plt.hist(X,bins=50,histtype='bar',edgecolor='black',density='true',cumulative ='true')
plt.title('cdf, normal samples $\mu=5,\, \sigma^2 = 3$')
plt.xlabel('$x$'); plt.ylabel('Frequency $\%$')
```





Expectation and Variance of a random variable

Definition: Assume that X is a random variable with pdf f. The *expectation* or *mean* of X is defined by

$$Discrerte: \quad \mu = \mathbb{E}[X] = \sum_k x_k\, p_k = \sum_k x_k\, \mathbb{P}(X = x_k)$$

$$Continuous: \quad \mu = \mathbb{E}[X] = \int x f(x)\, dx$$

The variance of X is defined as

$$\sigma^2 = \mathrm{Var}(X) = \mathbb{E}[(X - \mu)^2]$$

“Expecting how much X is deviated from the mean!"

Connection to integration: If $X \sim f(x)$ is a continuous random variable and g is a function, then $g(X)$ is another random variable. The expectation of $g(X)$ is obtained as

$$\mathbb{E}[g(X)] = \int g(x) f(x)\, dx$$

Exercise

Ex 1: Compute the mean and variance of a 6-sided dice drawing:

x_k	1	2	3	4	5	6
p_k	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$

$$\mu = \mathbb{E}[X] = \sum_{k=1}^{6} x_k p_k = \cdots, \quad \sigma^2 = \sum_{k=1}^{6} (x_k - \mu)^2 p_k = \cdots$$

Ex 2: Compute the mean and variance of a random variable $X \sim \mathcal{U}(a, b)$

$$\mu = \mathbb{E}[X] = \int_a^b x f(x) dx = \cdots, \quad \sigma^2 = \mathbb{E}[(X-\mu)^2] = \int_a^b (x-\mu)^2 f(x) dx = \cdots$$

Ex 3: Compute the mean and variance of a random variable $X \sim \mathcal{E}xp(\lambda)$

$$\mu = \mathbb{E}[X] = \int_0^\infty x f(x) dx = \cdots, \quad \sigma^2 = \int_0^\infty (x - \mu)^2 f(x) dx = \cdots$$

Monte Carlo integration

Assume that we aim to estimate the generic integral

$$\int_a^b g(x)f(x)\,dx$$

where f is a density function for random variable X. The function g is called the *performance* function.

Key point: The above integral is just $\mathbb{E}[g(X)]$, the mean of variable $g(X)$, thus Monte Calro method can be applied to estimate it:

- Generate samples $x_1, \ldots, x_N$ from pdf f
- Compute observations $g(x_k)$ and set

  $$\overline{g}_N = \frac{1}{N}\sum_{k=1}^{N} g(x_k)$$

  as an estimate for the integral.
- Random points are generated from f, only g appears in front of the summation symbol

Monte Carlo integration

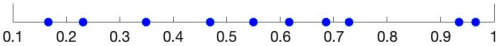
Example: Consider the 1D integral

$$\int_0^1 g(x)dx$$

This integral is indeed the expectation of $g(X)$ for uniform variable X on $[0, 1]$:

$$\mathbb{E}[g(X)] = \int_0^1 g(x) \cdot 1\, dx, \quad X \sim \mathcal{U}(0, 1)$$

- MC method: Choose N uniformly distributed random points $x_1, x_2, \dots, x_N$ in $[0, 1]$



  Then

$$\int_0^1 g(x)dx \approx \frac{1}{N}\left[g(x_1) + g(x_2) + \cdots + g(x_N)\right]$$

Approximate the value of integral $\int_0^1 \sin x \, dx$ using MC method with different number of points $N = 10^k, \; k = 0, 1, 2, 3, 4, 5$

```
def g_fun(x):
    return np.sin(x)

int_exact = 1-np.cos(1)        # exact value of integral for comparison
int_mc = np.zeros([6,1])
for k in range(6):
    N = 10**k
    X = np.random.rand(N,1)    # generate N uniform random points in [0,1]
    g = g_fun(X)
    int_mc[k] = np.sum(g)/N    # take mean
print('int_mc = \n',np.abs(int_exact-int_mc))
```

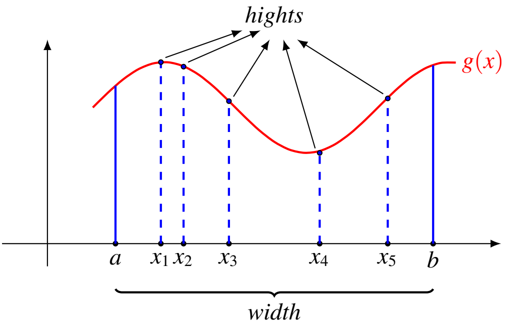
The output for an execution is:

```
int_mc =
 [[0.149659  ]
 [0.07784648]
 [0.0130975 ]
 [0.00131985]
 [0.00172056]
 [0.0005902 ]]
```

A new execution leads to a different result, but still shows a slow convergence

Estimate the integral of g on finite interval $[a,b]$

$$\int_a^b g(x)dx = (b-a)\int_a^b g(x)\frac{1}{b-a}dx \approx \underbrace{(b-a)}_{width}\underbrace{\frac{1}{N}\sum_{k=1}^{N} g(x_k)}_{mean\ hight}, \quad x_k \in \mathcal{U}(a,b)$$



Monte Carlo integration (uniform distr.)

An example in Python: Estimate the value of integral

$$\int_{-\pi/2}^{\pi/2} x^2 \cos x \, dx$$

using MC method and $N = 1000$

```
import numpy as np
def g_fun(x):              # g(x) defined in function g_fun
    return x**2*np.cos(x)

a,b = -np.pi/2,np.pi/2     # integral bounds a and b
N = 1000                   # number of realizations
result = np.empty(N)
for k in range(N):
    u = np.random.rand() # uniform point u in [0,1]
    x = (b-a)*u + a      # uniform point x in [a,b]
    result[k] = g_fun(x) # function evaluation
int_mc = (b-a)*np.mean(result)  # Monte Carlo estimation
```

Note: If $U \sim \mathcal{U}(0,1)$ then $X = (b-a)U + a \sim \mathcal{U}(a,b)$

Monte Carlo integration (example with normal distribution)

To estimate the value of integral

$$I = \int_{-\infty}^{\infty} (x^4 - x + 1)\mathrm{e}^{-x^2/2} dx$$

using MC method, we can write

$$I = \sqrt{2\pi} \int_{-\infty}^{\infty} \underbrace{(x^4 - x + 1)}_{g(x)} \underbrace{\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-x^2/2}}_{f(x)} dx$$

$f(x)$ is the pdf of $\mathcal{N}(0,1)$ on $(-\infty, \infty)$, so

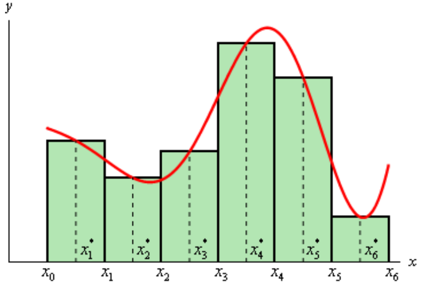
$$I \approx \sqrt{2\pi} \times \frac{1}{N} \sum_{k=1}^{N} (x_k^4 - x_k + 1)$$

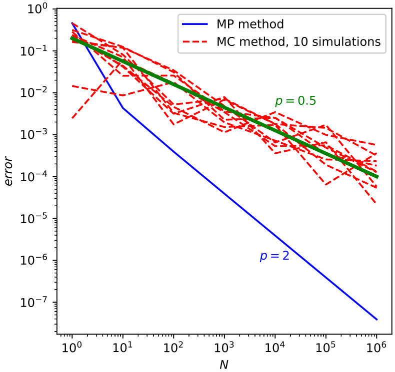
where x_k are generated from $\mathcal{N}(0,1)$.

As a deterministic method, consider the mid-point (MP) rule for integration:

$$\int_0^1 g(x)dx = h\left[g(x_1^*) + g(x_2^*) + \cdots + g(x_N^*)\right] + \mathcal{O}(h^2)$$
$$= \frac{1}{N}\left[g(x_1^*) + g(x_2^*) + \cdots + g(x_N^*)\right] + \mathcal{O}(N^{-2})$$

where $h = 1/N$ and N is the number of integration points in the interval $[0,1]$, and $x_k^* = (x_{k-1} + x_k)/2$ are mid points (equidistance points, not random).





Comparing the error and **order of convergence** in a log-log plot:

- Mid-point method (deterministic): $p = 2$, $e = \mathcal{O}(N^{-2}) = c \cdot N^{-2}$
- Monte Carlo method (stochastic): (ten simulations, different results), $p = 0.5$, $e = \mathcal{O}(N^{-0.5}) = c \cdot N^{-0.5}$

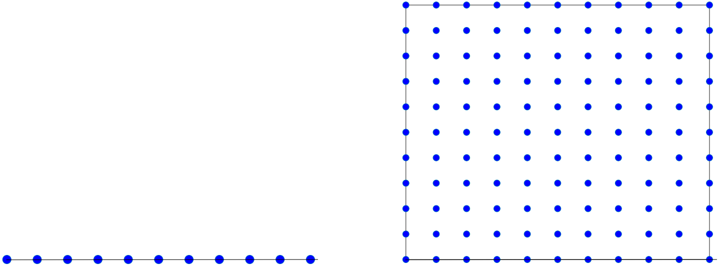
Stochastic vs. Deterministic

- The order of convergence of mid-point method is 2. Error behaves as $c \cdot h^2 = c \cdot N^{-2}$ in the maximum norm.
- The MC method converges in probability with convergence rate 0.5 independent of the dimension, the error behaves as $c \cdot N^{-1/2}$
- Which one is better? It depends on the dimension:
  - In 1D $h = \mathcal{O}(N^{-1})$ so mid-point error is $\mathcal{O}(N^{-2})$
  - In 2D $h = \mathcal{O}(N^{-1/2})$ so mid-point error is $\mathcal{O}(N^{-1})$
  - In 3D $h = \mathcal{O}(N^{-1/3})$ so mid-point error is $\mathcal{O}(N^{-2/3})$
  - In 4D $h = \mathcal{O}(N^{-1/4})$ so mid-point error is $\mathcal{O}(N^{-1/2})$
  - In 5D $h = \mathcal{O}(N^{-1/5})$ so mid-point error is $\mathcal{O}(N^{-2/5})$
- For dimensions greater than 4 the MC has a faster convergence

Answer the question when the mid-point rule is replaced by the Simpson rule (convergence rate h^4)?

Conclusion: Monte Carlo is a better choice for integration in higher dimensions!

Comparing h and N in 1D and 2D



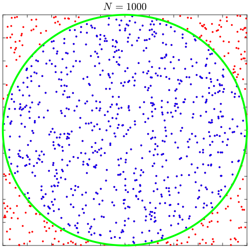
- In 1D $N = 10$ and $h = N^{-1} = 0.1$
- In 2D $N = 100$ and $h = N^{-2} = 0.1$

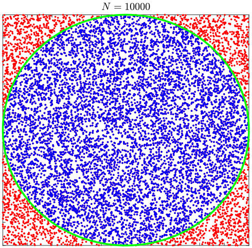
In deterministic methods for integration, to keep the same accuracy in 2D the number of points must be squared. In 3D it must be cubed, ...

Approximate π using a 2D MC method

Since the area of a circle of radius 1 is π so we can generate N uniformly distributed random points in square $[-1, 1]^2$ and count the points inside the circle:

$$\frac{\#\text{points inside circle}}{\#\text{points inside square}} \approx \frac{\text{area of circle}}{\text{area of square}} = \frac{\pi}{4}$$





How is it connected to MC integration?

Inverse transform method (ITM)

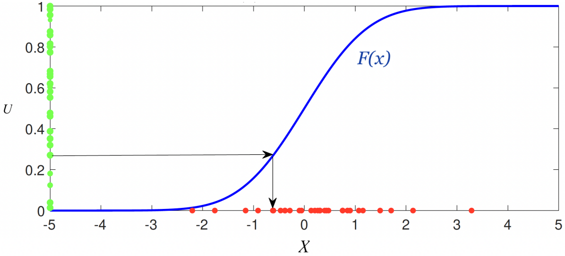
Assume that we want to generate X from a distribution $f(x)$. Let the cdf be denoted by $F(x)$. If F is invertible and $U \sim \mathcal{U}(0,1)$ then

$$X = F^{-1}(U) \sim f$$

Why? Since F is invertible and $\mathbb{P}(U \leqslant u) = u$, we have

$$\mathbb{P}(X \leqslant x) = \mathbb{P}(F^{-1}(U) \leqslant x) = \mathbb{P}(U \leqslant F(x)) = F(x)$$

This means that the cdf of X is F or the pdf of X is f.



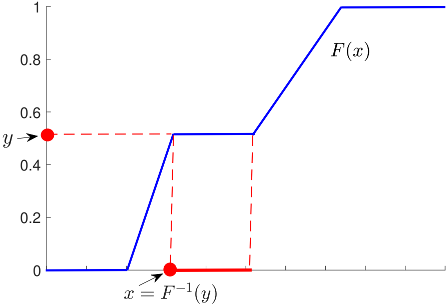
Animation: Watch

The definition of inverse function: If F is continuous and increasing then F^{-1} has the usual definition

$$F^{-1}(y) = \{x : F(x) = y\}$$

To cover all cases including discrete and nondecreasing functions we have the following definition:

$$F^{-1}(y) = \min\{x : F(x) \geqslant y\}, \quad 0 \leqslant y \leqslant 1.$$



Generate random points from the pdf below using ITM:

$$f(x) = \begin{cases} 2x, & x \in [0,1] \\ 0, & \text{otherwise}, \end{cases}$$

Solution: first we compute the corresponding cfd. It is enough to consider f on its support, i.e. for $x \in [0,1]$

$$F(x) = \int_0^x f(s)ds = \int_0^x 2s\,ds = s^2\Big|_0^x = x^2$$

Then we calculate the inverse of F which is $F^{-1}(x) = \sqrt{x}$. (write $y = F(x)$, switch x and y, $F(y) = x$ i.e. $y^2 = x$, or $y = \sqrt{x}$) Then we generate a uniform variable U and finally we set

$$X = F^{-1}(U) = \sqrt{U}$$

```
U = np.random.rand(N,1)
X = np.sqrt(U)
```

Sampling from exponential distribution using ITM

If $X \sim \mathcal{E}xp(\lambda)$, then $f(x) = \lambda e^{-\lambda x}$ for $x \in [0, \infty)$ and its cdf F is computed as

$$F(x) = \int_0^x \lambda e^{-\lambda s} ds = 1 - e^{-\lambda x}, \quad x \geqslant 0.$$

The inverse of F is (why?)

$$F^{-1}(x) = -\frac{1}{\lambda} \ln(1 - x)$$

To sample from the exponential distribution, we assume $U \sim \mathcal{U}(0, 1)$ and set

$$X = -\frac{1}{\lambda} \ln(1 - U) \sim \mathcal{E}xp(\lambda)$$

```
def RandExp(lam,N):
    # lam: distribution parameter, N: number of requested samples
    U =  np.random.rand(N)  # generate N uniform numbers in [0,1)
    X = -1/lam*np.log(1-U)  # use inverse transform to generate X
    return X
```

Sampling from Bernoulli distribution

We say X has Bernoulli distribution with probability $p \in [0, 1]$ and write $\mathcal{B}er(p)$ if

$$\mathbb{P}(X = 0) = p, \quad \mathbb{P}(X = 1) = 1 - p$$

So Bernoulli is the simplest discrete distribution with $x = \{0, 1\}$ and probability vector $\{1 - p, p\}$.

x	0	1
$f(x)$	$1-p$	p

Example: coin fillip, $p = 0.5$ if coin is fair

Sampling: Generate a uniform variable $U \sim \mathcal{U}(0, 1)$. If $U \leqslant p$ set $X = 1$ otherwise $X = 0$.

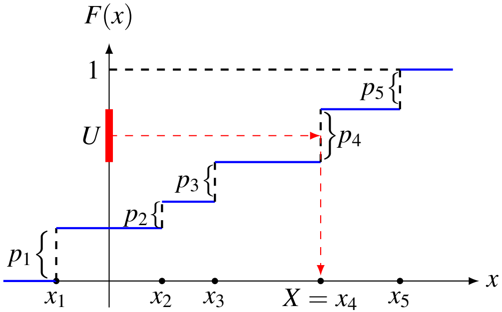
```
def RandBer(p,N):
    X = np.zeros(N)
    U = np.random.rand(N)
    idx = np.where(U <= p)
    X[idx] = 1
    return X
```

Assume that X has a discrete distribution $\mathcal{DD}([x_1, \ldots, x_m], [p_1, \ldots, p_m])$

x_k	x_1	x_2	$\ldots$	x_m
p_k	p_1	p_2	$\ldots$	p_m

Let $x_1 < x_2 < \cdots$. The cdf of X is simply given by

$$F(x) = \sum_{k: x_k \leqslant x} p_k.$$



ITM: according to definition of F^{-1}, first generate a uniform distribution $U \sim \mathcal{U}(0,1)$ and then find the smallest positive integer k such that $U \leqslant F(x_k)$. Finally $X = x_k$ is reported. Equivalently

$$X = \begin{cases} x_1, & 0 \leqslant U < p_1 \\ x_2, & p_1 \leqslant U < p_1 + p_2 \\ x_3, & p_1 + p_2 \leqslant U < p_1 + p_2 + p_3 \\ \vdots & \vdots \end{cases}$$

```python
def RandDisct(x,p,N):
    # x: sorted states
    # p: probabilities
    # N: number of requested samples
    cdf = np.cumsum(p)              # compute the cumulative vector
    U = np.random.rand(N)           # generate N uniform numbers in [0,1)
    idx = np.searchsorted(cdf, U) # search U values in cdf intervals
    return x[idx]                   # return corresponding states
```

Example: A three-dice rolling game

Consider a game where you toss 3 six-sided dice simultaneously.

If there are no doubles or triples then you win the sum of the three dice. Otherwise you lose all you have won so far. Let the random variable X be the value of your winning after 10 tosses. Here comes an example:

toss	outcome	winnings	toss	outcome	winnings
1	3, 2, 1	6	6	5, 4, 6	25
2	4 ,1, 2	13	7	2, 2,2	0
3	3, 5, 3	0	8	3, 5, 1	9
4	1, 6, 1	0	9	1,4, 4	0
5	2, 5, 3	10	10	1, 5, 2	$8 =: x_1$

What is the expected gain? Solve with Python.

Monte Carlo solution to the 3-dice game

```python
# Monte Carlo simulation for the 3-dice game
x = np.array([1,2,3,4,5,6])
p = np.array([1/6,1/6,1/6,1/6,1/6,1/6])
N = 10**4
X = np.empty(N)
for k in range(N):
    win = 0
    for j in range(10):
        Dice = RandDisct(x,p,3)      # 3-dice rolling simulation
        if len(np.unique(Dice)) < 3: # check for double or triple
            win = 0
        else:
            win += np.sum(Dice)
    X[k] = win
print('Expected Win =', np.mean(X))
```

A single execution gives:

```
Expected Win = 13.0534
```

An application of MC to estimate probabilities

Assume that X is a random variable and we want to estimate the probabilities like $\mathbb{P}(X \leqslant a)$ or $\mathbb{P}(X = a)$, ...

Example: In the 3-dice game estimate the probability that your winning will be greater than $\$20$. It means estimation of $\mathbb{P}(X \geqslant 20)$.

We just need to count the the number of outcomes greater than 20 and divide it by total outcomes:

```
pr_atleast20 = np.size(np.where(X >= 20))/N
```

A run of above code for $N = 10^4$ results in $\mathbb{P}(X \geqslant 20) = 0.2603$. So the probability of winning at least $\$20$ is near 0.26%

For grade 3:

Algorithm2_20230819

The task is to approximate the value of integral

$$\int_0^\infty (1+x)\exp(-2x)\,dx$$

using the Monte Carlo method on five random points

$$0.0108,\ 0.0602,\ 0.3568,\ 0.8921,\ 1.7759$$

which are exponentially distributed according to probability density function (pdf) $f(x) = 2\exp(-2x)$. What is the approximate value?

Select one alternative:

- ○ 0.80958
- ○ 1.61916
- ○ 0.59823
- ○ 0.29911

For higher grades:

Grade45_2_20230819

Assume that you are running a lumber mill in Krokom and you are trying to **estimate the production price** of a single piece of your standard framing timber. The cost to produce your standard framing timber includes labor, energy and trees. Assume the cost of labor is constant at 3 SEK per piece of framing timber. The cost of energy needed to make a single piece of framing timber is normally distributed with mean $\mu_E = 0.5$ SEK and standard deviation $\sigma_E = 0.1$ SEK. The price of tree needed to make a single piece of timber is distributed according to Weibull pdf $f_T(x) = 5x^4 \exp(-x^5)$ for $x \in [0, \infty)$.

Design a Monte Carlo algorithm to estimate the mean production price p_{mean} and variance $p_{variance}$ of standard framing timber.

Assume the function `randn()` exists and that it returns one standard normal number (with mean 0 and variance 1) every time that it is called. However, it is necessary to provide a detailed formulation of how we can generate a random number from the given Weibull distribution and incorporate it into the Monte Carlo algorithm.

Fill in your answer here or write in the answer sheet and hand it in.

Help

Format B I U x_2 x^2 I_x Ω Σ