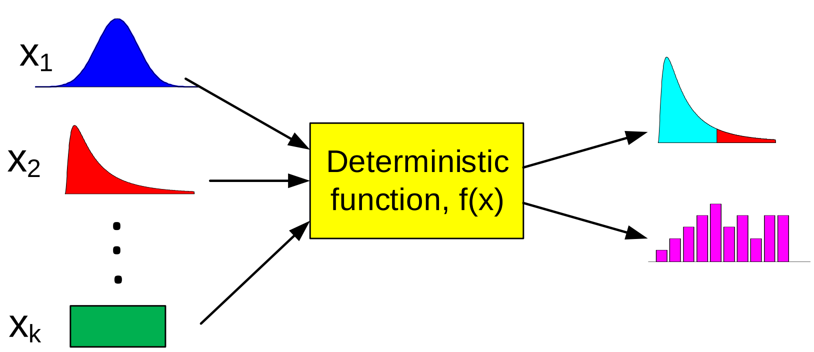
MONTE CARLO SIMULATIONS

The term "Monte Carlo Method" describes a large class of approaches for approximating the solution of a complex problem for which an exact solution is impossible or infeasible to obtain. Monte Carlo methods are especially useful for modeling processes with significant uncertainty in the inputs. In cases where the function is strongly nonlinear or the mean or variance of input random variables are not well-defined, Monte Carlo methods can be used to quantify the uncertainty in the output. A typical Monte Carlo approach consists of three main steps:

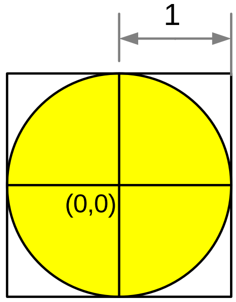
1. Generate a large number of realizations of input random variables $x_1, x_2 \dots, x_n$ using the specified probability distribution (joint, marginal, or conditional).
2. For each realization, compute the output of interest using a deterministic equation $f(x)$. Deterministic means that given a particular input $f(x)$ will always produce the same output.
3. Analyze the output using histograms, confidence intervals, etc.

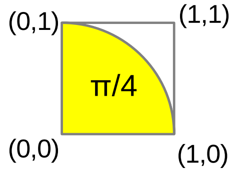
The figure below summarizes the main steps of the Monte Carlo method.



A simple example: Estimation of the value of π

Suppose that we do not know the value of π, and we wish to estimate it based on our knowledge that it represents the area of the unit circle. We also would like to construct a 95% confidence interval for π using sample sizes n =1000, 4000,16000 and 64000. The figure below shows a unit circle circumscribed by a square. Because of symmetry, we consider only one quadrant of the circle. The area of a quadrant of the circle with unit radius is $A_1 = \pi/4$, while the area of the unit square is $A = 1$.





If we generate n points (x, y) uniformly distributed over the square, the expected number of points inside quadrant (n_1) can be computed using the relationship:

$$\frac{E(n_1)}{n} = \frac{A_1}{A} = \frac{\pi}{4}.$$

Therefore, an estimate of π is:

$$\hat{\pi} = \frac{4n_1}{n}.$$

The hat over π indicates that what we compute is an estimate of π, which will get close to the true value as n approaches infinity.

Suppose we create an indicator vector b of length n, which has a value of 1 if the point is inside the circle, and 0 if outside. Since probability that a point falls inside the quarter circle is $\pi/4$, each element of b is a realization of a Bernoulli random variable B with success probability $p = \pi/4$. The expected value and variance of $B \sim Bern(p)$ with $p = \pi/4$ is

$$E(B) = \frac{\pi}{4}, \qquad Var(B) = \frac{\pi}{4}\left(1 - \frac{\pi}{4}\right).$$

Consider the random variable $X = 4B$. The expected value and variance of X are

$$E(X) = 4E(B) = \pi$$

$$Var(X) = 16Var(B) = 4\pi\left(1 - \frac{\pi}{4}\right).$$

The sample mean and sample variance of X are related to those of B as follows:

$$\bar{x} = 4\,\bar{b} \tag{1}$$

$$s^2 = 16 s_B^2 \tag{2}$$

where $\bar{b}$ and s_B^2 denote the sample mean and sample variance of b, respectively.

The Monte Carlo procedure for estimating π using n realizations is then:

1. Generate n realizations of the Bernoulli random variable $B \sim Bern(\frac{\pi}{4})$. Because we are not allowed to use π, we will take the following approach:
    a. Generate n points, uniformly distributed over the unit square: $0 \le x \le 1, \quad 0 \le y \le 1.$
    b. Create an indicator vector b of length n, which takes the value 1 if the point is inside the quadrant, and zero otherwise. The vector b is a realization of the random variable B.
2. Perform a deterministic computation: Compute $\hat{\pi}$ using Eq (5).
3. Analyze the results: Construct a 95% confidence interval for π.

```matlab
% Monte Carlo estimation of pi
nvec=[1000,4000,16000,64000];  % sample size
alpha=0.05;  % for 95% confidence level

for i=1:length(nvec);
n=nvec(i);
rand('seed',0)
xy=rand(n,2);
x=xy(:,1);  y=xy(:,2);
inside=@(x,y)sqrt(x.^2+y.^2)<1;
b=inside(x,y);
mx=4*mean(b);
s2=16*var(b) ;

figure
plot(x,y,'.')
hold on
plot(x(b),y(b),'r.'); % plot the points inside
axis equal
xlabel('x'), xlim([0 1])
ylabel('y'), ylim([0 1])
title(['estimated \pi =',num2str(mx,5)])
tcr=tinv(1-alpha/2, n-1);   % tcr for two-sided CI
me= tcr*sqrt(s2/n) ; % margin of error
lower=mx-me;
upper=mx+me;
result(i,:)=[n/10^3,mx,me,lower,upper];

shg, pause(2)
end

disp('  n/1000     estimate   margin    lower     upper')
disp('---------------------------------------------------')
disp(result)
```