Chapter 17
Monte Carlo Methods

59 A taste of Monte Carlo method

Monte Carlo methods is a class of numerical methods that relies on random sampling. For example, the following Monte Carlo method calculates the value of π:

1. Uniformly scatter some points over a unit square $[0,1] \times [0,1]$, as in Figure ??.
2. For each point, determine whether it lies inside the unit circle, the red region in Figure ??.
3. The percentage of points inside the unit circle is an estimate of the ratio of the red area and the area of the square, which is $\pi/4$. Multiply the percentage by 4 to estimate π.

The following Matlab script performs the Monte Carlo calculation:

```
n = 1000000;                      # number of Monte Carlo samples
x = rand(n, 1);                   # sample the input random variable x
y = rand(n, 1);                   # sample the input random variable y
isInside = (x.^2 + y.^2 < 1);     # is the point inside a unit circle?
percentage = sum(isInside) / n;   # compute statistics: the inside percentage
piEstimate = percentage * 4
```

Exercise 1. For each value of $n = 100$, $n = 10000$ and $n = 1000000$, run the script 3 times. How accurate is the estimated *pi*?

This example represent a general procedure of Monte Carlo methods: First, the input random variables (x and y) are sampled. Second, for each sample, a calculation is performed to obtain the outputs (whether the point is inside or not). Due to the randomness in the inputs, the outputs are also random variables. Finally, the statistics of the output random variables (the precentage of points inside the circle) are computed, which estimates the output.

Exercise 2. Write a Matlab script to use Monte Carlo method to estimate the volume of a 3-dimensional ball.

Exercise 3. Write a Matlab script to use Monte Carlo method to estimate the volume of a 10-dimensional hyper-ball.

60 Monte Carlo method in Engineering: Colloid thruster

In many engineering problems, the inputs are inheriently random. As an example of Monte Carlo method for these engineering applications, we study a space propulsion device, the colloid thruster. It use electrostatic acceleration of charged particles for propulsion. The charged particles are produced by an electrospray process, and have random initial velocity distribution.

The trajectory of the particle inside the electrical field $0 \le x \le L$ is governed by the set of ODEs

$$\frac{dx}{dt} = v_x \,, \quad \frac{dr}{dt} = v_r \,, \quad \frac{dv_x}{dt} = A_x(x) \,, \quad \frac{dv_r}{dt} = 0 \,, \tag{141}$$

where $A_x = \frac{e}{m}\frac{d\Phi(x)}{dx}$ is the acceleration induced by the electrical field. Here we use the polynomial approximation

$$A_x = \begin{cases} A_{x0}(1 - 3x^2 + 2x^3) & x \le L \\ 0 & x > L \end{cases} \tag{142}$$

The initial position of the charged particle is at $x(0) = r(0) = 0$. The initial velocity is

$$v_x(0) = V_0 \cos\alpha_0 \,, \quad v_r(0) = V_0 \sin\alpha_0 \tag{143}$$

In this equation, V_0 is the initial speed of the charged particle. It is assumed to be a fixed value in this section. α_0 is the initial angle, and is **random**. Here, we assume that α_0 is a **uniform random variable** $U(0, \alpha_{\max})$. We are interested in the **statistical distribution** of velocity (v_x, v_r) after the charged particle leave the electrical field to $x > L$.

A simple simulation of charge particle acceleration can be performed using the following Matlab function. For given initial velocity V_0, initial angle α_0, Coulumb acceleration A_x, length of acceleration L and time step size Δt, the function returns the final velocity v_x and v_r.

```
function [vx, vr] = thruster(V0, a0, Ax0, L, dt)
    % initial condition
    x = 0; r = 0;
    vx = V0 * cos(a0); vr = V0 * sin(a0);
    % time integration using Forward Euler
    while (x ≤ L)
        x = x + dt * vx;
        r = r + dt * vr;
        vx = vx + dt * Ax0 * (1 - 3 * x.^2 + 2 * x.^3);
    end
```

In our problem, there are large amounts of charged particles, and the initial angle of each charged particle is random. The role of probabilistic methods is to quantify the impact of this type of randomness on properties of interest (e.g. the terminal velocity). The results of the probabilistic analysis can take many forms depending on the specific application. In the example of the thurster where the terminal velocity is critical to the performance and efficiency of the thruster, the following information might be desired from a probabilistic analysis:

- The distribution of the terminal velocity v_x, v_r that would be observed in the population of charged particles.
- The probability that v_x or v_r is above some critical value (e.g., indicating the particle will hit part of the thruster).
- Instead of determining the entire distribution of terminal velocity, sometimes knowing the mean values $\mu v_x, \mu v_r$ is sufficient, e.g. for calculating the thrust.
- To have some indication of the variability of v_x and v_y without requiring accurate estimation of the entire distribution, the standard deviation, σv_x and σv_y, can be used.

The Monte Carlo method is based on the idea of taking a small, randomly-drawn sample from a population and estimating the desired outputs from this sample. For the outputs described above, this would involve:

- Replacing the distribution of v_x, v_r that would be observed over the entire population of particles with the distribution (i.e. histogram) of those observed in the random sample.

- Replacing the probability that v_x or v_r is above a critical value for the entire population of charged particles with the fraction of particles in the random sample that have v_x or v_r greater than the critical value.
- Replacing the mean value of v_x and v_r for the entire population with the mean value of the random sample.
- Replacing the standard deviation of v_x and v_r for the entire population with the standard deviation of the random sample.

Since this exactly what is done in the field of statistics, the analysis of the Monte Carlo method is a direct application of statistics.

In summary, the Monte Carlo method involves essentially three steps:

1. Generate a random sample of the input parameters according to the (assumed) distributions of the inputs.
2. Analyze (deterministically) each set of inputs in the sample.
3. Estimate the desired probabilistic outputs, and the uncertainty in these outputs, using the random sample.

The following Matlab code performs the Monte Carlo simulation for our thruster

```
% Deterministic (non-random) parameters
V0 = 0.1; Ax = 1.0; L = 1.0; dt = 0.001;
a0Max = 60. * pi / 180.;
% 1. Generate a random sample of the input parameters
a0MC = rand(10000, 1) * a0Max;
% Array to store Monte Carlo outputs
vxMC = [];
vrMC = [];
% 2. Analyze (deterministically) each set of inputs in the sample
for i = 1:10000
    [vx, vr] = thruster(V0, a0MC(i), Ax, L, dt);
    vxMC = [vxMC; vx];
    vrMC = [vrMC; vr];
end
% 3. Estimate the desired probabilistic outputs
% histogram
figure; hist(vxMC);
figure; hist(vrMC);
% mean
muVx = mean(vxMC)
muVr = mean(vrMC)
% standard deviation
sigmaVx = std(vxMC)
sigmaVr = std(vrMC)
```

Exercise 1. Use Monte Carlo method to estimate the probability $\mathbf{P}(v_r > C)$.

61 Uniform and Non-Uniform Random Variables

In the previous examples, the random input parameters have uniform distribution. A uniform distribution is defined by the two parameters, a and Ib, which are the minimum and maximum values the random variable can possibly take. Within the interval (a,b), all values are equally probable. The distribution is often abbreviated $U(a,b)$: Its probability density function (pdf) is

$$f(x) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & x < a \text{ or } x > b \end{cases} \tag{144}$$

It's cumulative distribution function (cdf), the integral of its pdf, is

$$F(x) = \begin{cases} \frac{x-a}{b-a} & a \le x \le b \\ 0 & x < a \\ 1 & x > b \end{cases} \tag{145}$$

Uniform random variable is special in Monte Carlo methods and in computation – most psuedo random number generators are designed to generate uniform random numbers. In MATLAB, for example, the following command generates an m by m array of $U(0,1)$ uniform random numbers. `x=rand(m,n);` To generate an $U(a,b)$ uniform random numbers, one can simply scale the $U(0,1)$ random numbers by `x=rand(m,n)*(b-a)+a;` Almost all other languages used for scientific computation have similar random number generators.

Exercise 2. What is the **mean, variance** and **standard deviation** of a $U(a,b)$ random variable?

Non-uniform distributions are those whose probability density functions are not constant. Several simple but important non-uniform distributions are

- Triangular distribution. It is characterized by three parameters a, b, c. The probability density function is

$$f(x) = \begin{cases} \dfrac{2(x-a)}{(b-a)(c-a)} & a \le x \le c \\ \dfrac{2(b-x)}{(b-a)(c-a)} & c \le x \le b \\ 0 & x < a \text{ or } x > b \end{cases} \tag{146}$$

- Exponential distribution. It is characterized by a single parameter λ. The probability density function is

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \ge 0 \\ 0 & x < 0 \end{cases} \tag{147}$$

- Normal distribution, also known as Gaussian distribution

61.1 Sampling non-uniform random variables

61.1.1 Rejection method

Rejection for triangular distribution

61.1.2 Transformation method

Uniform sampling for square root of α_0. Triangular distribution

Inverse cummulative density function.

Application to exponential random variable

62 Risk assessment

62.1 Computing probability of failure

62.2 Charged particle breakup and errosion in a Colloid thruster

63 Error Estimation for Monte Carlo Method

63.1 Law of Large Number of Central Limit Theorem

63.2 Error Estimator via Central Limit Theorem

63.3 Application to Mean and Variance

63.4 Application to Risk Assessment

63.5 Error Estimator via Bootstrap

64 Importance Sampling

65 Linear Sensitivity Method (Delta Method)