

Numerical simulation of random variables

Tuesday, February 10, 2009
12:02 PM

Another reference for last lecture: [Gardiner Sec. 1.2](#)

Any random (stochastic) simulation is typically built up by figuring out how to express the simulated variables through some combination of **independent** random variables.

For the random walk:

$$X_{n+1} = X_n + Z_n$$

↑
i.i.d. r.v.s

Then we just need to separately simulate each independent random variable. How do we do this?

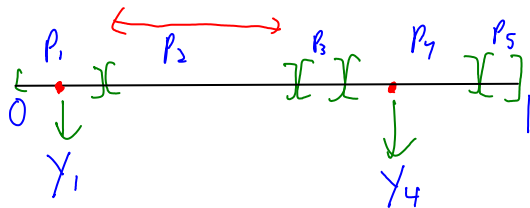
How do computers simulate random variables at all? They don't -- they use "pseudorandom" number generators that are actually deterministic algorithms that generate a sequence of numbers that "look" random and behave randomly according to some basic statistical tests. These pseudorandom number generators are sufficiently complex mappings of one integer to another. Typically these integers are divided by the largest integer the pseudorandom number generator wants to handle and this gives floating point random numbers which appear to be uniformly distributed on [0,1]. These are usually built-in routines in many operating systems and computational software (**rand** or **ran**). If you don't trust your software or operating system to have a good pseudorandom number generator, you can write or use your own (see Pierre L'Ecuyer website).

One practical piece of info: You can "seed" a random number generator at the beginning of a program with a particular integer to start with. If you don't do this the random number generator usually seeds itself by looking at the clock and calendar, etc.

What if you want to simulate a random variable that is not uniformly distributed on [0,1]. These typically involve some sort of transformation of the uniform random variable generator that's built into the system or the software. Some more sophisticated computational packages (MATLAB) actually have many special-purpose algorithms already written to simulate important random variables like exponential, Gaussian, etc,

Reference: [Kloeden & Platen, Numerical Simulation of Stochastic Differential Equations Sec. 1.3](#)

Discrete random variables: $P(Y = y_j) = p_j$



General purpose algorithm for any scalar independent random variable:

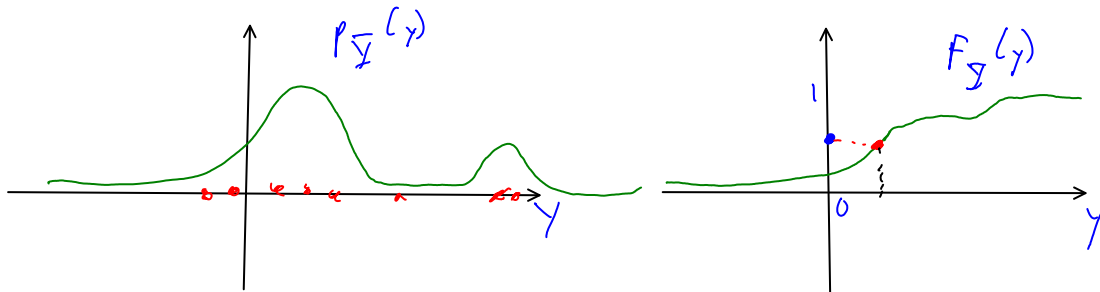
Inverse transform method

Suppose the random variable Y we want to simulate has CDF

$$F_Y(y) = P(Y \leq y)$$

$$= \int_{-\infty}^y p_Y(y') dy'$$

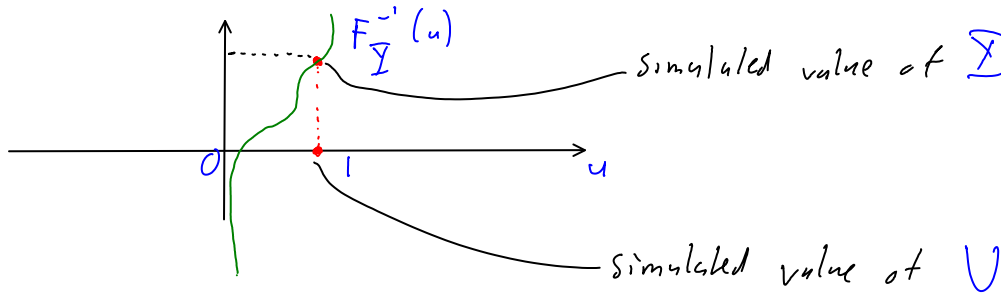
$-\infty$ when PDF exists,



You can simulate Y through the formula:

$$Y \sim F_Y^{-1}(U) \quad \text{where} \quad U \sim U(0,1)$$

↑
inverse function
uniform on $[0,1]$



One technical caveat: the inverse function of the CDF might not be defined in a strict sense when the PDF vanishes over some range of values so to be completely general, define:

$$y = F_Y^{-1}(q) \quad \text{means} \quad y = \min_{y'} \{y' : F_Y(y') \geq q\}$$

This is general purpose but not always practical. It is used, for example, for exponential random variables:

$$p_Y(y) = \begin{cases} \frac{1}{\mu} e^{-y/\mu} & \text{for } y \geq 0 \\ 0 & \text{for } y < 0 \end{cases}$$

$$F_Y(y) = \begin{cases} 1 - e^{-y/\mu} & \text{for } y \geq 0 \\ 0 & \text{for } y < 0 \end{cases}$$

$$F_Y^{-1}(u) = \begin{cases} -\mu \ln(1-u) & \text{for } 0 \leq u < 1 \end{cases}$$

$u = 1 - e^{-y/\mu}$
 $y = -\mu \ln(1-u)$

$$y = -\mu \ln(1-u)$$

$$F_Y^{-1}(U) = -\mu \ln(1-U) \quad \text{simulably}$$

an exp. rand. var. with mean μ

Much more easily to simulate $X \sim U(a,b)$ $X = a + (b-a)U$ where $U \sim U(0,1)$

However, the inverse transform method is not typically used to simulate Gaussian random variables because:

$$p_Y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(y-\mu)^2/\sigma^2}$$

$$F_Y(y) = \int_{-\infty}^y p_Y(y') dy' = \text{erf}\left(\frac{y-\mu}{\sigma}\right) \quad \text{or sth like that}$$

Computing the inverse of erf is not efficient. And this would be annoying to code.

Two special tricks for simulating Gaussian random variables, and they are based on one observation:

- The inverse transform method works wonderfully well when you try to simulate **two** Gaussian random variables in polar coordinates
- The reason is that if you simulate two independent standard Gaussian random variables (mean 0, variance 1), and look at them in polar coordinates, then the angle is uniformly distributed and the square of the radius is exponentially distributed.

$$X_1, X_2 \sim N(0,1)$$

↑ normal (Gaussian)
↑ mean
↑ variance

can be simulated as follows:

Take two independent random variables

$$U_1, U_2 \sim U(0,1)$$

↑ uniform on interval
↑ interval

$$R = \sqrt{-2 \ln U_1}$$

$$\Theta = 2\pi U_2$$

$$X_1 = R \cos \Theta$$

$$X_2 = R \sin \Theta$$

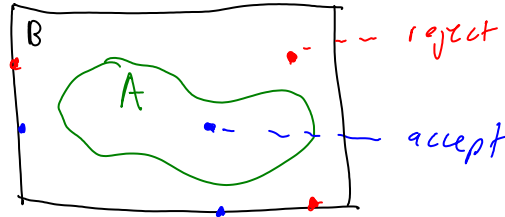
Box-Muller method

$$Y_1 = R \cos \theta \quad (\text{H})$$

$$Y_2 = R \sin \theta \quad (\text{H})$$

Box-Muller
method

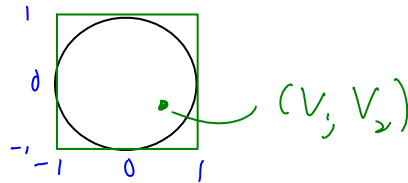
There's actually theoretically an even faster way to simulate Gaussian random variables based on the **rejection method**



To simulate a uniformly distributed random value in A , simulate uniform random values in a containing set B ; keep the first value that falls in A and simply reject those values that do not. Note that simulating uniform random variables over a rectangle is easy because you simply simulate uniform random variables along each side of the rectangle.

The **Polar-Marsaglia** method for simulating Gaussian random variables is the following:

Simulate a uniform point (V_1, V_2) in the unit circle by using the rejection method:



$$\rho^2 = V_1^2 + V_2^2$$

$$h = \sqrt{\frac{-2 \ln \rho^2}{\rho^2}}$$

$$Y_1 = h V_1$$

$$Y_2 = h V_2$$

The basic idea is to think of

$$Y_1 = \sqrt{-2 \ln \rho^2} \cos \alpha$$

$$Y_2 = \sqrt{-2 \ln \rho^2} \sin \alpha$$

where $\alpha \sim U(0, 2\pi)$

but avoid having to compute sin and cos (at the cost of wasting some points by rejection).

$$= t$$

$X(t)$ is also a Gaussian random variable (since it's a linear combination of Gaussian random variables)

with mean $X(0) = X_{in}$

variance $c^2 t$

$$\text{Var}(aX + b) = a^2 \text{Var} X$$

PDE approach:

PDF for $X(t)$ is given by

$$p_X(x; t) = \frac{\exp\left(-\frac{1}{4kt} (x - X_{in})^2\right)}{\sqrt{4\pi kt}}$$

which corresponds to a Gaussian random variable with mean

$$\langle X(t) \rangle = X_{in}$$

$$\text{var} X(t) = 2kt$$

These two viewpoints agree with each other provided that $c^2 = 2k$ ✓

$$c = \lim_{\Delta t \rightarrow 0} \frac{\sigma_z}{\sqrt{\Delta t}}$$

$$k = \lim_{\Delta t \rightarrow 0} \frac{\sigma_z^2}{2\Delta t}$$

This motivates defining the diffusivity for the trajectory-based perspective as:

$$k = \lim_{\Delta t \rightarrow 0} \frac{E(X(t + \Delta t) - X(t))^2}{2\Delta t}$$