General Sampling Methods

Reference: Glasserman, § 2.2 and § 2.3

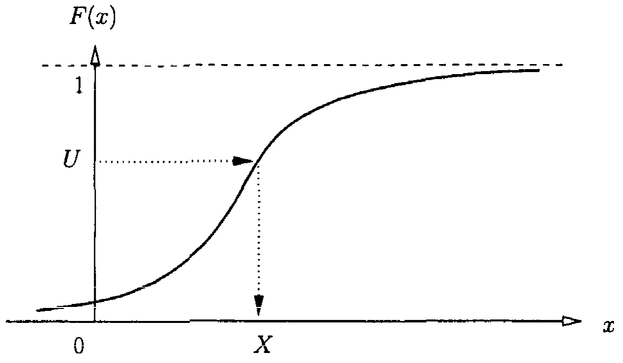
Claudio Pacati

academic year 2012–13

1 Inverse Transform Method

Assume $U \sim U(0,1)$ and let F be the cumulative distribution function of a distribution D. Then

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



Let's check if the distribution of X is D:

$$\begin{aligned}
\mathbb{P}(X \le x) &= \mathbb{P}\big(F^{-1}(U) \le x\big) && \text{(by definition of } X\text{)} \\
&= \mathbb{P}\big(U \le F(x)\big) && \text{(by definition of } F^{-1}\text{)} \\
&= F(x) && \text{(because } \mathbb{P}(U \le y) = y \ \forall y \in [0,1]\text{)} \ .
\end{aligned}$$

All works well if F is *continuous* and *strictly increasing* and hence F^{-1} is well defined.

If F is not increasing, i.e. has some constant sections, then we can resort to a sort of *pseudoinverse*

$$F^{-1}(u) = \inf\{x : F(x) \ge u\} \ .$$

Applications to *discuntinuous distributions* and to *non strictly increasing distributions.*

Drawback: Can be *slow* if the calculation of F^{-1} is slow!

1.1 Examples

Let $U \sim U(0,1)$

- The *exponential distribution* with mean θ has support $x \ge 0$, density

$$f(x) = \frac{1}{\theta} \mathrm{e}^{-x/\theta}$$

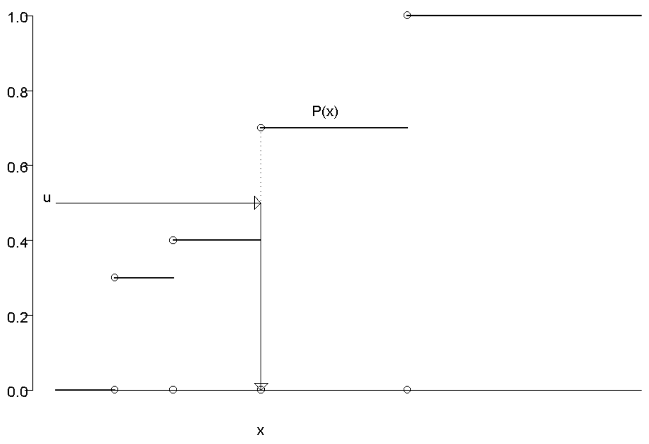


Figure 1: Transformation method: discrete distribution example

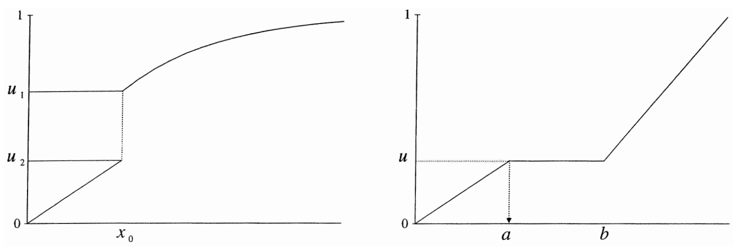


Figure 2: Transformation method: non-invertible continuous distributions examples

and cumulative distribution function

$$F(x) = 1 - \mathrm{e}^{-x/\theta} \qquad \Longrightarrow \qquad F^{-1}(u) = -\theta \log(1-u) \ .$$

Hence $X = -\theta \log(1 - U) \sim -\theta \log(U)$ has the exponential distribution with mean θ (notice that $1 - U \sim U$ by the symmetry of the $U(0, 1)$ distribution).

- The *standard normal* cumulative distribution function $N(x)$ cannot be inverted in closed form. However, every decent computing software implements numerically the inverse of $N(x)$. Using this inverse we have that $X = N^{-1}(U) \sim N(0, 1)$.
- Of course, for every μ and $\sigma > 0$, $Y = \mu + \sigma N^{-1}(U) \sim N(\mu, \sigma^2)$.

2 Multidimensional independent uniform variates

Assume we have a good generator for a $U(0, 1)$.
We want to simulate m independent $U(0, 1)$ variates each of lenght n. How can we do it?
A *wrong* way to solve the problem is to use the $U(0, 1)$ generator m times with *different seeds*. Although we will obtain m variates each with $U(0, 1)$ distribution, we have *no information* on the correlation structure these variates will have!
The *right way* is to use the *exactly the same generator* (i.e. without changing the seed) to produce a unique variate u_1, u_2, ... of length nm and extract from this the m variates using a *deterministic rule*.
Examples:

1. Construct the first variate using u_1, u_{m+1}, u_{2m+1}, ..., $u_{(n-1)m+1}$,

   the second using u_2, u_{m+2}, u_{2m+2}, ..., $u_{(n-1)m+2}$,

$\ldots,$

the last using u_m, u_{2m}, u_{3m}, $\ldots$, u_{nm}.

2. Construct the first variate using the first n us, the second variates using u_{n+1}, $\ldots$, u_{2n}, and so on.

3. $\ldots$.

3 Acceptance-Rejection Method

The *acceptance-rejection method* (ARM) is among the most widely applicable mechanisms for generating random samples.
It is appliable to *multivariate* distributions also.

Idea:
Suppose we want a sample from a distribution with *density* f, defined on some domain $\mathcal{X} \subset \mathbb{R}^d$.
Assume we are able to generate a sample from *another distribution*, with density g and such that

$$f(x) \leq cg(x) \qquad \text{for all } x \in \mathcal{X} \text{ and for some constant } c > 0.$$

$\longrightarrow$ We generate a sample X from g and *accept the sample with probability* $f(X)/[cg(X)]$.

In practice:

1. We generate an x from distribution g.

2. We generate an u from then $U(0,1)$ distribution.

3. If $u \leq f(x)/cg(x)$ we *accept* x as a random number for distribution f.

4. Else we *reject* x and restart from step 1.

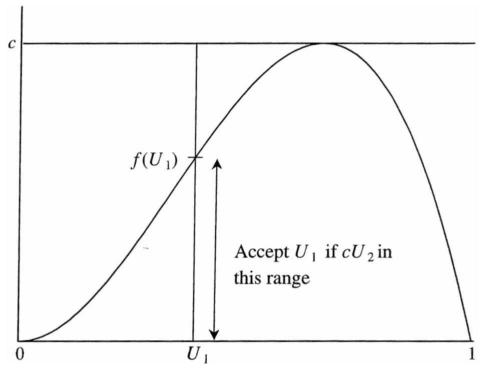


Figure 3: Illustration of the ARM using $g(x) = 1$ (density of a uniform distribution on $[0,1]$.

To verify the validity from the algorithm, assume X *to be a random variable with density* g and Y *to be a random variable returned by the acceptance-rejection method.*
We want to *verify that* Y *has indeed density* f.
First notice that, *conditional to the event* $U \leq f(X)/[cg(X)]$, Y has the same distribution of X.

This is equivalent to: for every Borel set $A \subset \mathcal{X}$

$$\begin{aligned}\mathbb{P}(Y \in A) &= \mathbb{P}\big(X \in A \mid U \le f(X)/[cg(X)]\big) \\ &= \frac{\mathbb{P}\big(X \in A \text{ and } U \le f(X)/[cg(X)]\big)}{\mathbb{P}\big(U \le f(X)/[cg(X)]\big)} \ .\end{aligned}$$

Now, since U is uniform, <u>conditional to X</u> we have that

$$\mathbb{P}\big(U \le f(X)/[cg(X)] \mid X\big) = f(X)/[cg(X)] \ .$$

Hence

$$\begin{aligned}\mathbb{P}\big(U \le f(X)/[cg(X)]\big) &= \int_{\mathcal{X}} \frac{f(x)}{cg(x)} g(x)\,\mathrm{d}x = \int_{\mathcal{X}} \frac{1}{c} f(x)\,\mathrm{d}x \\ &= \frac{1}{c}\int_{\mathcal{X}} f(x)\,\mathrm{d}x = \frac{1}{c} \ .\end{aligned}$$

Similarly

$$\begin{aligned}\mathbb{P}\big(X \in A \text{ and } U \le f(X)/[cg(X)]\big) &= \int_{\mathcal{X}} \mathbb{1}_A(x) \frac{f(x)}{cg(x)} g(x)\,\mathrm{d}x \\ &= \frac{1}{c}\int_A f(x)\,\mathrm{d}x \ .\end{aligned}$$

Hence, by combinig all the steps, we obtain

$$\mathbb{P}(Y \in A) = \int_A f(x)\,\mathrm{d}x \ .$$

Since A is arbitrarly this shows that Y has density f, as desired.

Remarks:

- Since both f and g integrate to 1 over $\mathcal{X}$, c cannot be less than 1.
- It is preferable in practice (for speed) to have c close to 1 (fewer samples rejected by the algorithm).
- The speed of the algorithm depends also on the speed in sampling from g.

3.1 Example: Generating a Normal From a Double Exponential

The *(standard) double exponential density* is defined on the whole $\mathbb{R}$ and is

$$g(x) = \frac{1}{2}\mathrm{e}^{-|x|} \ .$$

We want to use it as a *ARM-candidate* to generate an $N(0,1)$ sample.
Recall the (standard) normal density:

$$f(x) = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-x^2/2} \ .$$

In the ratio

$$\frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}}\mathrm{e}^{|x| - x^2/2}$$

notice that $|x| - x^2/2 \le 1/2$ for each $x \in \mathbb{R}$:

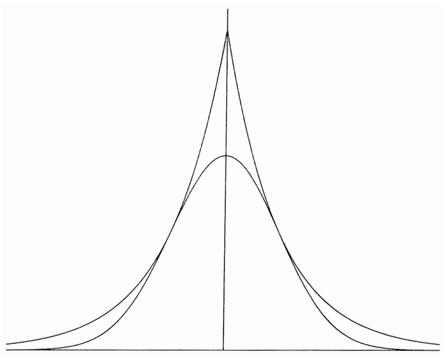


Figure 4: Normal density and scaled double exponential density

If we set $y = |x|$, then $x^2 = y^2$ and the solution of the inequality $y - y^2/2 \leq 1/2$ is $y \in \mathbb{R}$. Hence

$$\frac{f(x)}{g(x)} \leq \sqrt{\frac{2}{\pi}}\mathrm{e}^{1/2} = \sqrt{\frac{2\mathrm{e}}{\pi}} \approx 1.3155 \ .$$

If we set $c = \sqrt{2\mathrm{e}/\pi}$, then for every $x \in \mathbb{R}$ we have $f(x) \leq cg(x)$ and the bound is tigth. The double exponential density can be generated by applying on both sides the algorithm shown as an example for the transformation method.

The *acceptance test* for u becomes

$$u \leq \frac{f(x)}{cg(x)} = \sqrt{\frac{\pi}{2\mathrm{e}}}\sqrt{\frac{2}{\pi}}\mathrm{e}^{|x|-x^2/2} = \mathrm{e}^{|x|-x^2/2-1/2} = \mathrm{e}^{-(|x|-1)^2/2} \ .$$

A clever enhancement is based on the remark that both distributions are *symmetric*: it suffices to generate positive samples and determine the sign only if the sample is accepted. In this case the absolute value is unnecessary in the acceptance test.

The algorithm is therefore:

1. Generate u_1, u_2 and u_3 from a $U(0,1)$.
2. Set $x = -\log(u_1)$
3. If $u_2 > \exp(-(x-1)^2/2)$ (*rejection*) then goto 1.
4. Else (*acceptance*) if $u_3 \leq 0.5$ then set $x = -x$.
5. Return x.

4 Normal Random Variables and Vectors

We have already seen how to generate $N(0,1)$ random samples using the inverse transform method and the acceptance-rejection method.

However those methods are *not particularly fast*.

In this section we will discuss a *faster* method, simple enough to be used as general sampling method for the standard normal distribution.

4.1 The Box-Muller Method

The *Box-Muller method* takes a sample from a bivariate independent standard normal distribution, each component of which is thus a univariate standard normal.
The algorithm is based on the following two properties of the bivariate independent standard normal distribution: if $Z = (Z_1, Z_2)$ has this distribution, then

1. $R = Z_1^2 + Z_2^2$ is exponentially distributed with mean 2, i.e.
$$\mathbb{P}(R \leq x) = 1 - \mathrm{e}^{-x/2} \ .$$
2. Given R, the point (Z_1, Z_2) is uniformly distributed on the circle of radius $\sqrt{R}$ centered at the origin.

We can use these properties to build the algorithm:

1. Generate independent $U_1, U_2 \sim U(0,1)$.
2. Transform U_1 in an exponential sample by $R = -2\log(U_1)$.
3. Transform U_2 in a random uniform angle between 0 and 2π by $\alpha = 2\pi U_2$.
4. The corresponding point on the circle of radius $\sqrt{R}$ centered at the orgin has coordinates $Z_1 = \sqrt{R}\cos\alpha$ and $Z_2 = \sqrt{R}\sin\alpha$.

Remark. If $\{z_{11}, z_{12}, \dots, z_{1n}\}$ and $\{z_{21}, z_{22}, \dots, z_{2n}\}$ are two independent $N(0,1)$ variates (e.g. generated by the Box-Muller algorithm), then by *combining them in a determinstic way* we get a unique variate form a $N(0,1)$ of length $2n$.

In particular:
$$z_{11}, z_{21}, z_{12}, z_{22}, \dots, z_{1n}, z_{2n}$$
is a good sample for a $N(0,1)$.
Therefore, the Box-Muller algorithm can be used in two ways:

- If we need a bivariate $N(0,1)$ independent sample, then we use it in the standard way.
- If we need a univariate $N(0,1)$ sample we combine the output.

4.1.1 VBA Implementation

Box-Muller method:

```
Sub BM(u1 As Double, u2 As Double, n1 As Double, _
    n2 As Double)
    Const Pi As Double = 3.14159265358979#
    Dim sqrtR As Double, alpha As Double
    sqrtR = Sqr(-2 * Log(u1))
    alpha = 2 * Pi * u2
    n1 = sqrtR * Cos(alpha)
    n2 = sqrtR * Sin(alpha)
End Sub
```

Generating *bivariate* $N(0,1)$ independent random numbers:

```
Sub biN(x As Long, n1 As Double, n2 As Double)
    Dim u1 As Double, u2 As Double
    u1 = LCG(x)
    u2 = LCG(x)
    Call BM(u1, u2, n1, n2)
End Sub
```

Generating *univariate* $N(0,1)$ random numbers:

```
Function uniN(x As Long) As Double
    Dim n1 As Double, n2 As Double
    Static hasStored As Boolean
    Static stored As Double
    If hasStored = True Then
        hasStored = False
        uniN = stored
    Else
        Call biN(x,n1,n2)
        stored = n2
        hasStored = True
        uniN = n1
    End If
End Function
```

4.1.2 Application

In file `Monte_Carlo_call_by_inverse_transform_normal.xls` we saw a naive implementation of the inverse transfrom method to simulate the $N(0,1)$ distributed random variable needed to Monte Carlo price a European call option in the Black & Scholes model.

In file `BSEuropean.xls` we use the Box-Muller method (previous functions) in an efficient way to Monte Carlo compute (among other things) the same option (and the corresponding Greeks).

Homework: Modify (a copy of) the previous program to price a Eurepan contingent T-claim D with the following payoff:

$$D(T) = \begin{cases} S(T) & \text{if } S(T) \leq K_1, \\ K_1 + [S(T) - K_1]^2 & \text{if } K_1 < S(T) \leq K_2, \\ K_1 + [K_2 - K_1]^2 & \text{if } S(T) > K_2 \end{cases}$$

Can you find a closed form solution for the price (and for the Greeks) of this contract?

4.2 Generating Multivariate Normals

As for the $U(0,1)$ case, by extracting in a *determinstic* way subsequences of a $N(0,1)$ variate we can construct m *independent* $N(0,1)$ *variates*, i.e. a *variate for an independent standard normal* m*-vector.*

But we could need to simulate m normal variables with *means* $\mu_1, \ldots, \mu_m$ and *covariance matrix* $\mathbf{\Sigma}$.

How to achieve this?

It is *straightforward* to obtain the requirement on means: simply add to each variate the desired mean.

To obtian the requested covariance structure is *more complicated*, but can be solved.

4.2.1 The Linear Transformation Property

A *multivariate normal* random variable $\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is completely defined by its *vector of means* $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)^\top$ (column vector) and its *covariance matrix*

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1m} \\ \sigma_{12} & \sigma_{22} & \dots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1m} & \sigma_{2m} & \dots & \sigma_{mm} \end{pmatrix},$$

where $\sigma_{kk} = \sigma_k^2$ is the *variance* of the kth component and $\sigma_{hk} = \rho_{hk}\sigma_h\sigma_k$ is the *covariance* between the kth component and the hth component; $\rho_{hk} \in [-1, 1]$ is their *correlation.*

Theorem (linear transformation property)**.** *Given a $m \times m$ matrix $\boldsymbol{A}$, the linear transform $\boldsymbol{Y} = \boldsymbol{AX}$ is a multivariate normal, with mean vector $\boldsymbol{A\mu}$ and covariance matrix $\boldsymbol{A\Sigma A}^\top$. I.e.*

$$\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad \Longrightarrow \qquad \boldsymbol{AX} \sim N\left(\boldsymbol{A\mu}, \boldsymbol{A\Sigma A}^\top\right) .$$

We can use the linear transformation property to *transform an independent multivariate normal* $\boldsymbol{Z} \sim N(\boldsymbol{0}, \boldsymbol{I}_m)$ (where $\boldsymbol{I}_m$ is the identity $m \times m$ matrix) into the desired $\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

If $\boldsymbol{A}$ is a $m \times m$ matrix, by the linear transformation property $\boldsymbol{AZ}$ has mean $\boldsymbol{0}$ and covariance matrix $\boldsymbol{AI}_m\boldsymbol{A}^\top = \boldsymbol{AA}^\top$.

So our task is to *find a matrix $\boldsymbol{A}$ satisfying* $\boldsymbol{AA}^\top = \boldsymbol{\Sigma}$.

Once found, we *simultate* Z and *transform* it into the desired $\boldsymbol{X}$ by $\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{AZ}$.

Recall that $\boldsymbol{\Sigma}$ is *symmetric* and *positive semi-defined.*

If $\boldsymbol{\Sigma}$ is *positive defined* then we can use *Cholesky factorization*, that gives us a *lower triangular* matrix $\boldsymbol{A}$ such that $\boldsymbol{AA}^\top = \boldsymbol{\Sigma}$.

4.2.2 Cholesky Factorization

Cholesky factorization is particularly easy to use. Because of lower triangularity of $\boldsymbol{A}$ the final transformation $\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{AZ}$ becomes

$$\begin{aligned} X_1 &= \mu_1 + a_{11}Z_1 \\ X_2 &= \mu_2 + a_{21}Z_1 + a_{22}Z_2 \\ &\vdots \\ X_m &= \mu_m + a_{m1}Z_1 + a_{m2}Z_2 + \dots a_{mm}Z_m \end{aligned}$$

Hence we obtain the component of $\boldsymbol{X}$ by an affine transformation of the first component of $\boldsymbol{Z}$, the second by an affine transformation of the first to components of $\boldsymbol{Z}$ and so on.

Example. If $m = 2$, $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$, the system becomes

$$\begin{aligned} X_1 &= a_{11}Z_1 \\ X_2 &= a_{21}Z_1 + a_{22}Z_2 \end{aligned}$$

and gives us two standard normal variables X_1 and X_2 correlated by ρ from two independent standard normals Z_1 and Z_2.

It is esay to obtain the elements a_{11}, a_{21} and a_{22} of the Cholesky factor $\boldsymbol{A}$ of $\boldsymbol{\Sigma}$, by imposing X_1 and X_2 to have variance 1 and correlation ρ, i.e. by solving the system

$$\begin{aligned} 1 &= \operatorname{var}(a_{11}Z_1) = a_{11}^2 \operatorname{var}(Z_1) = a_{11}^2 \\ 1 &= \operatorname{var}(a_{21}Z_1 + a_{22}Z_2) = a_{21}^2 \operatorname{var}(Z_1) + a_{22}^2 \operatorname{var}(Z_2) = a_{21}^2 + a_{22}^2 \\ \rho &= \mathbb{E}\left[a_{11}Z_1(a_{21}Z_1 + a_{22}Z_2)\right] = a_{11}a_{21}\,\mathbb{E}(Z_1^2) + a_{11}a_{22}\,\mathbb{E}(Z_1Z_2) \\ &= a_{11}a_{21} \end{aligned}$$

We obtain $a_{11} = 1$, $a_{21} = \rho$ *and* $a_{22} = \sqrt{1-\rho^2}$.

4.2.3 VBA Implementation

```
Sub Cholesky(d As Long, s() As Double, a() As Double)
    Dim i As Long, j As Long, k As Long
    Dim v() As Double
    ReDim v(d)
    For i = 1 To d
        For j = 1 To d
            a(i, j) = 0.0
        Next j
    Next i
    For j = 1 To d
        For i = j To d
            v(i) = s(i, j)
            For k = 1 To j - 1
                v(i) = v(i) - a(j, k) * a(i, k)
            Next k
            a(i, j) = v(i) / Sqr(v(j))                (∗)
        Next i
    Next j
End Sub
```

Remark. We can apply Cholesky factorization only to *positive defined* $\boldsymbol{\Sigma}$.

If $\boldsymbol{\Sigma}$ is *not positive defined* but only positive semi-defined, then it can be shown that *the Cholesky algorithm fails*: in line (∗), for some `j` the value of `v(j)` becomes zero.

To solve the problem, notice that it can be shown that if an $m \times m$ covariance matrix $\boldsymbol{\Sigma}$ is not positive defined, then $r = \operatorname{rank} \boldsymbol{\Sigma} < m$. This means that there is a subvector $\tilde{\boldsymbol{X}}$ of $\boldsymbol{X}$ of length r such that every of the $m - r$ components of $\boldsymbol{X}$ not in $\tilde{\boldsymbol{X}}$ is a linear combination of components of $\tilde{\boldsymbol{X}}$.
So we *reduce the sampling problem* to $\tilde{\boldsymbol{X}}$, which has a positive defined covariance matrix, and obtain the other componets by the aforementioned linear combinations.