Chapter 2

Random Variable Generation

From a uniform (pseudo) random generator, one can construct (pseudo) random generators for many other distributions. Indeed, many common distributions and families of distributions are related to each other via simple transformations. Such relations lead to general rules for generating random variables. For example, generating random variables from any location-scale family of distributions can be carried out by generating random variables from the base distribution of the family, followed by an affine transformation. Universal procedures for generating random variables include the inverse-transform method (Sect. 2.1), the composition method (Sect. 2.2), and the acceptance-rejection method (Sect. 2.3). Often, random variables can be generated according to ad-hoc transformations (Sect. 2.4).

2.1 Inverse-transform method

it exploits the cumulative distr. function of a random variable (notice: the cdf has to admit the inverse/generalized inverse)

The inverse-transform method can be applied when the target distribution is one-dimensional, that is, to generate samples from a prescribed target distribution on the real numbers or in the discrete case.

Discrete random variables

Consider a discrete r.v. X taking values $x_1 < x_2 < \ldots < x_n$ with probability

$$P(X = x_i) = p_i, \qquad \sum_{i=1}^{n} p_i = 1.$$

Let us denote by

$$F_i = \sum_{j=1}^{i} p_j = P(X \le x_i);$$

then X can be generated starting from a uniform random variable $U \sim \mathcal{U}(0,1)$ by the following

Algorithm 2 Discrete inverse-transform

1: Generate $U \sim \mathcal{U}(0,1)$
2: Set $X = x_i$ if $F_{i-1} < U \le F_i$.

note: we know F_i since the distribution from which we want to sample is known (we only need a way for obtaining random samples)

Indeed,

$$P(X = x_i) = P(F_{i-1} < U \le F_i) = P(U \subset (F_i - p_i, F_i]) = p_i.$$

Hands-On Problem 2.1 focuses on the implementation in `Matlab`. Graphically:

We sample from $\mathcal{U}[0,1)$ generating U. Then we invert and we obtain X. In this case $X = x_3$ (where $x_1, \dots, x_n$ are the possible values that X can take)

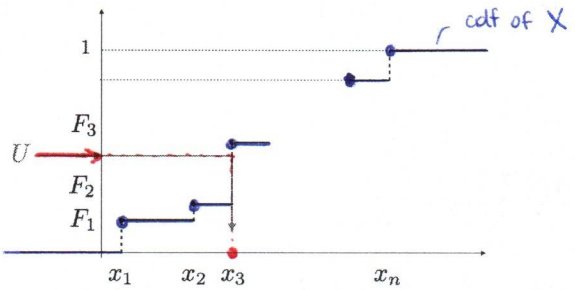


Figure 2.1: Discrete inverse-transform

- **Example 2.1.1.** *We want to sample a Bernoulli random variable* $X \sim Be(p)$*:*

$$P(X = 1) = p, \qquad P(X = 0) = 1 - p.$$

Given $U \sim \mathcal{U}([0,1])$*, we can set* $X = 0$ *if* $U \leq p$*, and* $X = 1$ *otherwise; equivalently,* $X = \mathbf{1}_{\{U>1-p\}}$*. Indeed,*

$$P(X = 1) = P(U > 1 - p) = p, \qquad P(X = 0) = P(U \leq 1 - p) = 1 - p.$$

This automatically follows from Algorithm 2 by setting $x_1 = 0$*,* $x_2 = 1$*,* $p_1 = 1 - p$*,* $p_2 = p$*; in this case,* $F_1 = 1 - p$*,* $F_2 = 1$*.* **Hands-On Problem 1.2** *focuses on the implementation in* `Matlab`*.*

- **Example 2.1.2.** *The fact that a* $Bin(n,p)$*-distributed random variable* X *can be written as the sum* $X = B_1 + \ldots + B_n$ *of independent* $Be(p)$ *random variables* $\{B_i\}$ *leads to the following generator: generate* $X_1, \ldots, X_n \sim Be(p)$*, then return* $X = \sum_{i=1}^{n} X_i$*; however, alternative methods should be used for large* n*.* **Hands-On Problem 2.2** *focuses on the implementation in* `Matlab`*.*

Continuous random variable

Consider now a continuous random variable X taking values in an interval $[a, b]$ with continuous and *strictly increasing* cumulative function

$$F(x) = P(X \leq x) : [a, b] \to [0, 1], \quad F(a) = 0, \; F(b) = 1.$$

In this case the inverse function $F^{-1} : [0, 1] \to [a, b]$ is uniquely defined and X can be generated starting from a uniform random variable $U \sim \mathcal{U}(0, 1)$ by the following

Algorithm 3 Continuous inverse-transform

1: Generate $U \sim \mathcal{U}(0, 1)$
2: Set $X = F^{-1}(U)$

Indeed,

$$P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x),$$

hence X has the right distribution. **Hands-On Problem 1.3** focuses on the implementation in `Matlab`. Graphically:

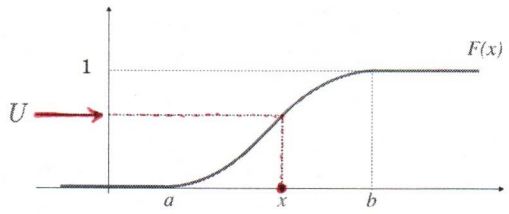


Figure 2.2: Continuous inverse-transform

Example 2.1.3. *Drawing a sample from a uniform variable $X \sim \mathcal{U}(a,b)$ follows immediately from the inverse-transform method, generating $U \sim \mathcal{U}(0,1)$ and returning $X = a + (b-a)U$.*

Example 2.1.4. *We want to sample an exponential variable, $X \sim Exp(\lambda)$, $\lambda > 0$. The PDF and the CDF of X are*

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0, \qquad\qquad F(x) = \int_0^x \lambda e^{-\lambda y} dy = 1 - e^{-\lambda x},$$

respectively. The inverse of $y = F(x)$ is

$$1 - e^{-\lambda x} = y \quad \rightarrow \quad -\lambda x = \log(1-y) \quad \rightarrow \quad x = -\frac{1}{\lambda}\log(1-y)$$

and thus $F^{-1}(y) = -\frac{1}{\lambda}\log(1-y)$ for all $y \in (0,1)$. Now, assume $U \sim \mathcal{U}(0,1)$; we then obtain that

$$X = F^{-1}(U) = -\frac{1}{\lambda}\log(1-U)$$

is $Exp(\lambda)$-distributed. Moreover, since the variable $1-U$ has the same distribution than U, an equivalent formula is

$$X = -\frac{1}{\lambda}\log(U).$$

Hands-On Problem 1.4 *focuses on the implementation in* `Matlab`.

Both discrete and continuous cases can be combined together by defining a proper right inverse of F when F is not continuous or not strictly monotone. Let X be a random variable with CDF F. We can introduce the *generalized inverse* of F as

$$F^-(u) = \inf\{x : F(x) \geq u\};$$

this is the generalized inverse of F for any F: ℝ → ℝ increasing

actually, $F^-(u) = \min\{x : F(x) \geq u\}$ since $F(\cdot)$ is right-continuous. Then, X can be generated as $X = F^-(U)$ with $U \sim \mathcal{U}(0,1)$. Indeed, (proof.)

$$F^-(u) = \min\{z : F(z) \geq u\} \leq x \qquad \text{if and only if} \qquad F(x) \geq u;$$

hence,

$$P(X \leq x) = P(F^-(U) \leq x) = P(U \leq F(x)) = F(x).$$

We remark that with this definition of F^- we recover the discrete inverse-transform in the case of a discrete random variable.

PROBABILITY INTEGRAL TRANSFORM: if $U \sim \mathcal{U}([0,1])$ then the random variable $F^-(U)$ has the distribution F.

Why is this important? To generate $X \sim F$ it's enough to generate $U \sim \mathcal{U}([0,1])$ and then make the transformation $x = F^-(u)$ (or $x = F^{-1}(u)$ if $F^{-1}(\cdot)$ exists)

→ being able to generate from $\mathcal{U}([0,1])$ is of key importance to sample from any other distribution

if we have a distr. that is given by the composition of known distributions then we can put pieces together

2.2 Composition method

Suppose that a random variable X has a mixture distribution CDF

$$F(x) = \sum_{i=1}^{n} p_i F_i(x), \qquad \text{with } \sum_{i=1}^{n} p_i = 1,$$

where F_i is a CDF for all $i = 1, \ldots, n$; equivalently, the PDF of X can be expressed as

$$f(x) = \sum_{i=1}^{n} p_i f_i(x), \qquad \int f_i(x)dx = 1 \quad \forall i = 1, \ldots, n. \tag{2.1}$$

First we decide from which F_i we sample, then we sample from it

Then X can be generated by:

1. generate a discrete random variable Y such that $P(Y = i) = p_i$;
2. generate $X \sim F_Y$ e.g., by inversion.

(e.g. by using inverse transform - discrete)

Hands-On Problem 2.5 focuses on the implementation in `Matlab`.

- **Example 2.2.1.** *Consider $X \sim Laplace(0, 1)$, whose PDF is given by*

$$f(x) = \frac{1}{2}e^{-|x|} = \underbrace{\frac{1}{2}e^{-x}\mathbf{1}_{\{X \geq 0\}}}_{\sim Exp(1)} + \underbrace{\frac{1}{2}e^{x}\mathbf{1}_{\{X < 0\}}}_{\sim -Exp(1)}$$

the two weights are equal so we're going to use a $Be(\frac{1}{2})$

The Laplace distribution is also sometimes called the double exponential distribution, because it can be thought of as two exponential distributions. To sample from X, we can generate $B \sim Be(1/2)$, and $Y \sim Exp(1)$, and set

$$X = \begin{cases} Y & \text{if } B = 1 \\ -Y & \text{if } B = 0 \end{cases} \qquad \text{or, equivalently,} \quad X = (2B - 1)Y.$$

- **Example 2.2.2.** *(Mixture of normals) We wish to draw samples from a mixture of normal PDFs. Specifically, suppose that the PDF from which to draw has the form (2.1) with $n = 3$ and $(p_1, p_2, p_3) = (0.2, 0.4, 0.4)$, and suppose that the means and standard deviations of the normal pdfs are given by $\boldsymbol{\mu} = (-0.5, 1, 2)$ and $\Sigma = (0.5, 0.4, 2)$. A useful shorthand notation for this distribution is $0.2N(-0.5, 0.5^2) + 0.4N(1, 0.42) + 0.4N(2, 2^2)$. The following `Matlab` code implements the composition method and plots a histogram of the generated data.*

```
p = [0.2, 0.4, 0.4];  N = 10^5;
mu = [-0.5, 1, 2];    sigma = [0.5, 0.4, 2];
[dummy,t]=histc(rand(1,N),[0,cumsum(p)]);    % draw from p
x = randn(1,N).*sigma(t) + mu(t);            % draw a normal r.v.
thist(x,200)                                 % make a histogram of the data
```

- **Example 2.2.3.** *Composition methods appear often in Bayesian analysis. Supposing that $\theta \sim f(\theta)$ and $(X|\theta) \sim f(x|\theta)$, the PDF of X is given by $f(x) = \int f(x|\theta)f(\theta)d\theta$. The mechanism for simulating samples from this distribution using the composition method is as follows: first draw θ from the PDF $f(\theta)$, and then, given θ, draw X from $f(x|\theta)$.*

- **Remark 2.2.1.** *The alias method is a family of efficient algorithms for sampling from a discrete probability distribution, exploiting the composition method. Indeed, if n is large, the search fo the interval $(F_{i-1}, F_i]$ in Algorithm 2 such that $U \subset (F_{i-1}, F_i]$ might be costly. In this case, an alternative approach consists in representing the CDF $F(x)$ as a mixture distribution*

$$F(x) = \sum_{i=1}^{n} \frac{1}{n} G_i(x)$$

such that each G_i is a two points distribution (Bernoulli) and apply the composition method.

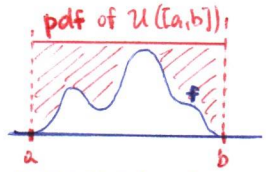
2.3 Acceptance-Rejection method

The acceptance-rejection sampling method is a more advanced, and very popular, method for random number generation. Several aspects make this method different from basic methods such as the inverse-transform method discussed before. First, rejection sampling is not restricted to $\mathcal{U}(0,1)$-distributed input samples. The method is often used in multi-stage approaches where different methods are used to generate samples of approximately the correct distribution and then rejection sampling is used to convert these samples to follow the target distribution exactly. However, the numerical efficiency of the method becomes a concern, since a random (and potentially large) number of input samples is required to generate one output sample.

The intuition

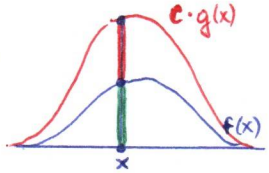
Assume to bound the PDF f of a random variable X (from which we want to draw a sample) onto a large rectangular board and throwing darts at it. Assume that the darts are uniformly distributed around the board. If we now remove all of the darts that are outside the area under the curve, the remaining darts will be distributed uniformly within the area under the curve of f, and the abscissas of these darts will be distributed according to the PDF of f – indeed, there is the most room for the darts to land where the curve is highest and thus the PDF is greatest.

This case is equivalent to a particular form of acceptance-rejection sampling where the *proposal* distribution is uniform (hence its graph is a rectangle). The general form of rejection sampling assumes that the board is not necessarily rectangular but is shaped according to the PDF of some proposal distribution $Cg(\cdot)$ that we know how to sample from (for example, using inversion sampling), and which is at least as high at every point as the distribution we want to sample from – that is, $f(x) \leq Cg(x)$ for any possible x – so that the former completely encloses the latter. Here is the procedure:



here we're choosing as the proposal a uniform distribution, however it can be anything that we can sample from

1. Sample a point on the x-axis from the proposal distribution $Cg(x)$.
2. Draw a vertical line at this x-position, up to the maximum y-value $Cg(x)$ of the proposal distribution.
3. Sample uniformly along this line from 0 to $Cg(x)$. If the sampled value is greater than the value $f(x)$ of the desired distribution at this vertical line – that is, $UCg(x) \geq f(x)$, being $U \sim \mathcal{U}(0,1)$ – reject the x-value and return to step 1; else the x-value is a sample from the desired distribution.



we sample x from $C \cdot g(x)$ and we trace a vertical line. Then we sample uniformly on the line: if we end up higher than $f(x)$ we reject, otherwise we accept the sample x. The condition to check for acceptance is: $U \cdot C \cdot g(x) \leq f(x)$ where $U \sim \mathcal{U}([0,1])$. Remember that we know $f(\cdot)$, we just don't know how to sample from it. In this way the samples that we accept are distributed according to f.

The Acceptance-Rejection algorithm

Consider a continuous random variable X with PDF f and CDF F. In those cases where F is difficult to invert, the inverse-transform method is not viable. Another situation which may arise is when f is known only up to a multiplicative constant, i.e.,

useful for example if we have a **Bayesian context**: we sample from the posterior which is only "$\propto$" to something we know, not "$=$"

$$f(x) = K\tilde{f}(x), \qquad K = \left(\int_{\mathbb{R}} \tilde{f}(x)dx\right)^{-1} = \text{what makes } \tilde{f}(x) \text{ a density}$$

and we only know $\tilde{f}$ whereas k is difficult to compute. In both cases, the acceptance-rejection method might represent a good alternative to sample X. For the sake of generality, we consider along this section the case in which we know $\tilde{f}$; if instead the PDF f is known, simply put $K = 1$.

The idea of the method is to find an auxiliary PDF g – also called the *proposal* – which is easy to sample from, and a constant $C \geq K^{-1}$, such that

$$\tilde{f}(x) \leq Cg(x) \qquad \forall x \in \mathbb{R}.$$

Then, the acceptance-rejection algorithm reads:

Algorithm 4 Acceptance-Rejection

1: Generate Y from the PDF g ;
2: Generate $U \sim \mathcal{U}(0,1)$ independent of Y ;
3: **if** $U \leq \tilde{f}(Y)/Cg(Y)$ **then**
4: set $X = Y$;
5: **else**
6: return to step 1.
7: **end if**

In other words, we generate Y from the PDF g and accept it with probability $\tilde{f}(Y)/Cg(Y)$; otherwise, we reject Y and try again. **Hands-On Problem 2.6** focuses on the implementation in `Matlab`.

Lemma 2.3.1. *The acceptance-rejection algorithm produces a random variable with the desired PDF $f(x) = K\tilde{f}(x)$, even without knowing K.*

Proof. Observe that the distribution of X is the distribution of Y conditional to the event $U \leq \tilde{f}(Y)/Cg(Y)$. Hence,

$$P(X \leq x) = P\left(Y \leq x \mid U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right) = \frac{P\left(Y \leq x\,, U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right)}{P\left(U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right)}.$$

Now,

$$P\left(Y \leq x\,, U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right) = \int_{-\infty}^{x} \int_{0}^{\frac{\tilde{f}(y)}{Cg(y)}} du\, g(y)dy = \int_{-\infty}^{x} \frac{\tilde{f}(y)}{Cg(y)} g(y)dy = \frac{1}{C}\int_{-\infty}^{x} \tilde{f}(y)dy$$

and[1]

$$P\left(U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right) = \int_{\mathbb{R}} \frac{\tilde{f}(y)}{Cg(y)} g(y)dy = \frac{1}{C}\int_{\mathbb{R}} \tilde{f}(y)dy.$$

As a result,

$$P(X \leq x) = \frac{\int_{-\infty}^{x} \tilde{f}(y)dy}{\int_{\mathbb{R}} \tilde{f}(y)dy} = \frac{\frac{1}{K}\int_{-\infty}^{x} f(y)dy}{\frac{1}{K}\int_{\mathbb{R}} f(y)dy} = \int_{-\infty}^{x} f(y)dy = F(x).$$

$\tilde{f} = \frac{1}{K} f$; $= 1$ (density)

■

The probability of acceptance in the A-R algorithm, which is a measure of the *efficiency* of the A-R method, is

$$P\left(U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right) = \frac{1}{C}\int_{\mathbb{R}} \tilde{f}(y)dy = \frac{1}{KC}\int_{\mathbb{R}} f(y)dy = \frac{1}{KC}$$ = rate of acceptance

[1] Indeed, since $P(U \leq u) = u$ when U is uniform on $(0,1)$, we obtain

$$P\left(U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right) = \mathbb{E}\left[\mathbf{1}_{\left\{U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right\}}\right] = \mathbb{E}\left[\mathbb{E}\left[\mathbf{1}_{\left\{U \leq \frac{\tilde{f}(Y)}{Cg(Y)}\right\}} | Y\right]\right] =$$
$$= \mathbb{E}\left[\mathbb{P}\left(U \leq \frac{\tilde{f}(Y)}{Cg(Y)} \middle| Y\right)\right] = \mathbb{E}\left[\frac{\tilde{f}(Y)}{Cg(Y)}\right] = \int_{\mathbb{R}} \frac{\tilde{f}(y)}{Cg(y)} g(y)dy$$

where $U \sim \mathcal{U}(0,1)$, and the value y each time is generated under the PDF g of the proposal distribution Y.

and since the trials (Y, U) are independent, the number of trials required to obtain a successful pair (X, U) –i.e., the time until the first success – has a geometric distribution $Geom(KC)$ with expected value KC. For the algorithm to be efficient, C should be as close as possible to K^{-1}.

Geometric interpretation of the Acceptance-Rejection method

In the one-dimensional case, we can provide a straightforward geometrical interpretation of the A-R method. We start with the following

Lemma 2.3.2. *Given a positive integrable function $h : \mathbb{R} \to \mathbb{R}_+$, and the region*

$$A_h = \{(x, u) \ : \ x \in \mathbb{R}, \ 0 \le u \le h(x)\}$$

then

$$(X, U) \sim \mathcal{U}(A_h) \quad \Leftrightarrow \quad X \sim \frac{h(x)}{\int h(x)dx}, \ \ U|X \sim \mathcal{U}(0, h(X)).$$

Proof. We have that $|A_h| = \int h(x)dx$ so if $(X, U) \sim \mathcal{U}(A_h)$, then

$$f_{(X,U)}(x, u) = \frac{1}{|A_h|}, \qquad f_X(x) = \int_0^{h(x)} f_{(X,U)}(x, u)du = \frac{h(x)}{\int h(x)dx}.$$

On the other hand, if $X \sim \frac{h(x)}{\int h(x)dx}$, $U|X \sim \mathcal{U}(0, h(X))$, then

$$f_{(X,U)}(x, u) = f_{(U|X)}(u|x) f_X(x) = \frac{1}{h(x)} \cdot \frac{h(x)}{\int h(x)dx} = \frac{1}{|A_h|}$$

hence $(X, U) \sim \mathcal{U}(A_h)$. □

This observation leads to the following geometrical interpretation of the A-R algorithm: before the `if` condition in the algorithm, one draws samples (Y, U) uniform in the region

$$A_{Cg} = \{(y, u) \ : \ y \in \mathbb{R}, \ 0 \le u \le Cg(y)\};$$

then, one retains only those samples that fall in the region

$$A_{\tilde{f}} = \{(x, u) \ : \ x \in \mathbb{R}, \ 0 \le u \le \tilde{f}(x)\}.$$

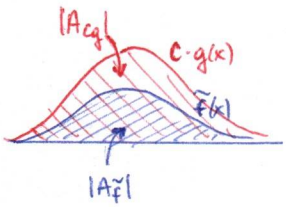
Hence the abscissas of the retained points have density

$$\frac{\tilde{f}(x)}{\int \tilde{f}(x)dx} = K\tilde{f}(x) = f(x).$$

The abscissas of the accepted points (which are uniformly distributed in $A_{\tilde{f}}$) have the desired distribution. The probability that a point is accepted is equal to the ratio $|A_{\tilde{f}}|/|A_{Cg}|$, that is, it is equal to $1/KC$.

In other words, when simulating the pair $(y, u = UCg(y))$, with $U \sim \mathcal{U}(0, 1)$, one produces a uniform simulation over the subgraph of $Cg(y)$. Accepting only pairs such that $UCg(y) < f(y)$ then produces pairs (y, u) uniformly distributed over the subgraph of f and thus, marginally, a simulation from $X \sim f$.

Summary:
$|A_{\tilde{f}}|$ = area below $\tilde{f}$
$|A_{Cg}|$ = area below Cg



$$\frac{|A_{\tilde{f}}|}{|A_{Cg}|} = \frac{1}{K \cdot C}$$

$= \mathbb{P}(\text{accepting})$

we don't want to waste samples, hence we want $\mathbb{P}$(accepting) high
$\Rightarrow$ a good proposal is a proposal that follows goodly $\tilde{f}(x)$

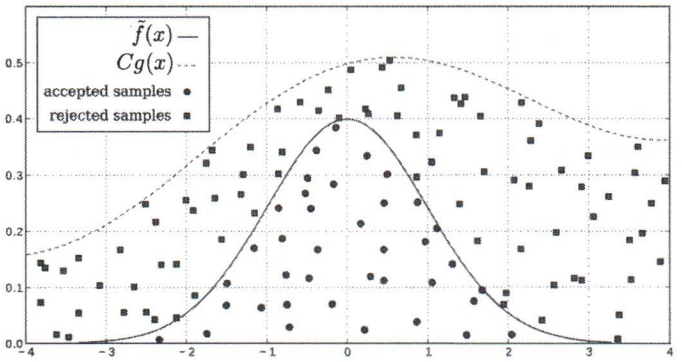
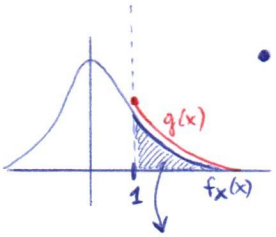


Figure 2.3: Acceptance-Rejection method. The abscissa of the blue points has the right distribution.



we want to sample here knowing how to sample from an exponential

- **Example 2.3.1.** *Suppose we want to sample from the tail of a standard normal distribution* $X \sim N(0,1)$ *with* $X \geq 1$. *In this case,*

$$f_X(x) \propto e^{-x^2}, \quad x \geq 1.$$

We could take as proposal distribution $g(x)$ *an exponential* $Exp(1)$ *translated in 1, that is,*

$$g(x) = e^{-(x-1)} \mathbf{1}_{x \geq 1}.$$

In this case we have $\tilde{f}(x) = e^{-x^2} \mathbf{1}_{x \geq 1}$ *and* $\tilde{f}(x) \leq g(x)\ \forall x \geq 1$, *hence* $C = 1$. *The A-R algorithm reads in this case:*

1. *Generate* Y *from the proposal* $1 + Exp(1)$
2. *Generate* $U \sim \mathcal{U}(0,1)$
3. *if* $U \leq \frac{e^{-Y^2}}{e^{-(Y-1)}} = e^{-Y^2+Y-1}$ *set* $X = Y$, *otherwise return to step 1.*

The acceptance probability is

$$\frac{1}{K} = \int_1^\infty e^{-y^2} dy = \sqrt{2\pi}(1 - \Phi(1)) \approx 0.56$$

being $\Phi(\cdot)$ *the CDF of a standard normal distribution – this value is nothing but* $1/K$. *Notice that if we just sample from* $N(0,1)$ *and reject all samples* < 1, *we would have an acceptance rate of about 0.16.* (however, up to this point we don't know how to sample from a gaussian)

2.4 Transformation of random variables

"ad-hoc" transforms

Samples from a wide variety of distributions can be generated by considering deterministic transformations of random variables. The inverse transform method, introduced in Sect. **??**, is a special case of this technique where we transform a uniformly distributed random variable using the inverse of a CDF.

A typical case is represented by *affine transformations* and location-scale families; other remarkable examples are Box-Muller transformations. Indeed, for specific distributions, such as Normal, Gamma, Poisson, ... there are often much more efficient methods for random variable generation than the general purpose methods introduced so far. which exploit the special structure and probabilistic interpretation of the underlying distribution. Here we show some possible options; other cases are proposed in the *Hands on* section.

Location-scale family

A family of continuous distributions with PDFs $\{f(x;\mu,\sigma),\ \ \mu\in\mathbb{R},\ \ \sigma>0\}$ of the form

$$f(x;\mu,\sigma)=\frac{1}{\sigma}\tilde{f}\left(\frac{x-\mu}{\sigma}\right),\qquad x\in\mathbb{R},$$

is called a *location-scale family* with standard PDF $\tilde{f}(\cdot)$. Parameter μ is called the location and σ is called the scale. Location-scale families of distributions arise from the affine transformation

$$X=\mu+\sigma Z,$$

where Z is distributed according to the standard PDF of the family. In particular, if $Z\sim\tilde{f}\equiv f(\cdot;0,1)$, then

$$\mu+\sigma Z\sim f(\cdot;\mu,\sigma).$$

Thus, to generate a random variable from a location-scale family of pdfs, first generate a random variable from the base pdf and then apply an affine transformation to that random variable.

- **Example 2.4.1.** *The standard* normal *or* Gaussian *distribution $N(0,1)$ has PDF*

$$\tilde{f}(x)=\frac{1}{\sqrt{2\pi}}e^{-x^2/2},\quad x\in\mathbb{R}$$

and the corresponding location-scale family of PDF is therefore

$$f(x;\mu,\sigma^2)=\frac{1}{\sigma}\tilde{f}\left(\frac{x-\mu}{\sigma}\right)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{(x-\mu)}{\sigma}\right)^2},\quad x\in\mathbb{R},$$

yielding a $N(\mu,\sigma^2)$ distribution. Hence, to draw $X\sim N(\mu,\sigma^2)$ first draw $Z\sim N(0,1)$ and then return $X=\mu+\sigma Z$. In `Matlab`*, drawing from the standard normal distribution is implemented via the function* `randn`*. Since $N(\mu,\sigma^2)$ forms a location-scale family, we only consider generation from $N(0,1)$, through the so-called Box–Muller method (see the following subsection).*

For more general transformations, recall this fundamental result, answering the following question: if X is a random variable with PDF $f_X(x)$ and $Y=g(X)$, which is the PDF of Y?

Theorem 2.4.1. *Let X have PDF $f_X(x)$ and let $Y=g(X)$, where g is a monotone function. Let*

$$\mathcal{X}=\{x\ :\ f_X(x)>0\},\qquad \mathcal{Y}=\{y\ :\ y=g(x)\ \text{for some}\ x\in\mathcal{X}\}.$$

Suppose that $f_X(x)$ is continuous on $\mathcal{X}$ and that $g^{-1}(y)$ has a continuous derivative on $\mathcal{Y}$. Then

$$f_Y(y)=\begin{cases} f_X(g^{-1}(y))|(g^{-1})'(y)| & \text{if } y\in\mathcal{Y}\\ 0 & \text{otherwise.}\end{cases}$$

$N(0,1)$ generators

we can use the acceptance-rejection method but it's **not** the best in terms of efficiency

A first option to draw a sample from a standard normal variable is to exploit the so-called *Box-Muller transformation*. Let $X,Y\sim\mathcal{U}([0,1])$ two independent uniform random variables over the interval $[0,1]$, and consider the *Box-Muller transformation*

$$(v,w)=BM(x,y),\qquad \begin{cases} v=\sqrt{-2\log x}\cos(2\pi y)\\ w=\sqrt{-2\log x}\sin(2\pi y).\end{cases}$$

log = $\log_e$

It is possible to show that $(V,W) = BW(X,Y)$ are independent and distributed as $N(0,1)$. Indeed, observing that $v^2+w^2 = -2\log x$ and $w/v = \tan(2\pi y)$, the inverse transformation is

$$(x,y) = BM^{-1}(v,w), \qquad \begin{cases} x = e^{-\frac{v^2+w^2}{2}} \\ y = \frac{1}{2\pi}\arctan(w/v) \end{cases}$$

whose Jacobian is

$$|\det J| = \left| \begin{array}{cc} -ve^{-\frac{v^2+w^2}{2}} & -we^{-\frac{v^2+w^2}{2}} \\ -\frac{1}{2\pi}\frac{w}{v^2+w^2} & \frac{1}{2\pi}\frac{v}{v^2+w^2} \end{array} \right| = \frac{1}{2\pi}e^{-\frac{v^2+w^2}{2}}$$

Since (X,Y) are independent and uniform over $[0,1]$, their joint PDF is $f_{X,Y}(x,y) = 1$. Hence, the joint PDF of (V,W) is

$$f_{V,W}(v,w) = f_{X,Y}(x(v,w),y(v,w))|\det J| = \frac{1}{2\pi}e^{-\frac{v^2+w^2}{2}}$$

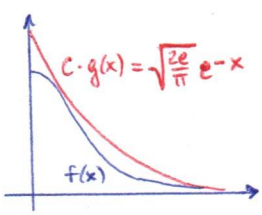
which is the joint PDF of two independent standard Gaussians.

To draw n samples from a standard $N(0,1)$ it is therefore sufficient to draw $n/2$ samples from $X \sim \mathcal{U}([0,1])$ and $n/2$ samples from $Y \sim U([0,1])$ and use the Box Muller transformation.

Algorithm 5 $N(0,1)$ generator , Box-Muller method

1: Generate $U_1, U_2 \sim (0,1)$
2: Return $X = \sqrt{-2\log U_1}\cos(2\pi U_2)$ and $Y = \sqrt{-2\log U_1}\sin(2\pi U_2)$.

Hands-On Problem 2.4 focuses on the `Matlab` implementation.



we do the acceptance-rejection method on this and then we fix the sign according to a Bernoulli(1/2)

Another possible option is to draw samples from $N(0,1)$ using the Acceptance-Rejection algorithm, from an Exponential $Exp(1)$ distribution. Indeed, suppose we wish to generate random variables from the positive normal PDF

$$f(x) = \sqrt{\frac{2}{\pi}}e^{-x^2/2}, \qquad x \geq 0,$$

using acceptance-rejection. We can bound $f(x)$ by $Cg(x)$, where $g(x) = e^{-x}$ is the PDF of the $Exp(1)$ distribution. The smallest constant C such that $f(x) \leq Cg(x)$ is $\sqrt{2e/\pi}$. The efficiency of this method is $1/C = \sqrt{\pi}2e \approx 0.76$.

Since $f(x)$ is the PDF of the absolute value of a standard normal random variable, we can generate $Z \sim N(0,1)$ by first generating $X \sim f$ as above and then returning $Z = XS$, where S is a random sign; for example, $S = 1 - 2\mathbf{1}_{\{U \leq 1/2\}}$ with $U \sim \mathcal{U}(0,1)$. This procedure for generating $N(0,1)$ random variables is summarized in Algorithm 6:

Algorithm 6 $N(0,1)$ generator, Acceptance-Rejection from $Exp(1)$

1: Generate $Y \sim Exp(1)$;
2: Generate $\tilde{U} \sim \mathcal{U}(0,1)$ independent of Y ;
3: **if** $U \leq e^{-(Y-1)^2/2}$ **then**
4: generate $U \sim \mathcal{U}(0,1)$ and output $Z = (1 - 2\mathbf{1}_{\{U \leq 1/2\}})Y$
5: **else**
6: return to step 1.
7: **end if**

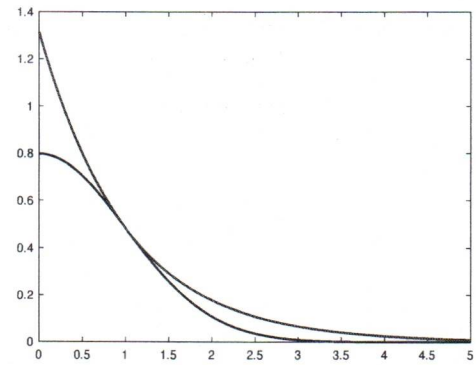


Figure 2.4: Bounding the positive normal density

Example 2.4.2. *Let $X, Y \sim N(0,1)$ be independent standard normal random variables, and (ρ, θ) their representation in polar coordinates. Since*

$$X^2 + Y^2 \sim \chi^2(2) = Exp(1/2),$$

is a chi-square distribution with 2 degrees of freedom, it follows that $\rho^2 \sim Exp(1/2)$. Moreover, by the radial symmetry of the bivariate normal distribution $N(\mathbf{0}, I_2)$, the distribution of (X, Y) given $\rho^2 = X^2 + Y^2$ is uniform on $[0, 2\pi)$. From these considerations, an algorithm to generate $(X, Y) \sim N(\mathbf{0}, I_2)$ is:

Algorithm 7 Box-Muller method, bivariate standard normal $N(\mathbf{0}, I_2)$

1: Generate $U \sim \mathcal{U}(0,1)$ and set $\rho = \sqrt{-2 \ln U}$ (hence $\rho^2 \sim Exp(1/2)$)
2: Generate $V \sim \mathcal{U}(0,1)$ and set $\theta = 2\pi V$ (hence $\theta \sim \mathcal{U}(0, 2\pi)$)
3: Set $X = \rho \cos\theta$, $Y = \rho \sin\theta$.

2.5 Multivariate Random Variable Generation

Let $\mathbf{X} = (X_1, \ldots, X_n)^\top \in \mathbb{R}^n$ be a vector of random variables with joint cumulative distribution function

$$F(\mathbf{z}) = F(z_1, \ldots, z_n) = P(X_1 \leq z_1, \ldots, X_n \leq z_n)$$

and probability density function $f(\mathbf{x}) = f(x_1, \ldots, x_n)$, if it exists, such that

$$F(z_1, \ldots, z_n) = \int_{-\infty}^{z_1} \cdots \int_{-\infty}^{z_n} f(x_1, \ldots, x_n) dx_1 \ldots dx_n.$$

The inverse transform method is not (directly) available in the multivariate case to generate $\mathbf{X}$, since $F : \mathbb{R}^n \to \mathbb{R}$ is not an invertible function. The Acceptance-Rejection method, on the other hand, generalizes in a straightforward way to the multivariate case; however, it is in general very difficult to find an auxiliary function $g(\mathbf{x})$ such that $f(\mathbf{x}) \leq Cg(\mathbf{x})\ \forall \mathbf{x} \in \mathbb{R}^n$ for some $C > 1$, that leads to acceptable acceptance rates.

We discuss in this section two simple cases in which $\mathbf{X}$ can be generated: the case of *independent components* and the case of *conditional distributions*. Then, we consider the use of transformation, and finally the case of multivariate Gaussian distributions.

2.5.1 Independent components

If $X_1, \ldots, X_n$ are independent, each with distribution F_i and density function f_i, we have that

$$F(\mathbf{z}) = F(z_1, \ldots, z_n) = F_1(z_1) \cdot \ldots \cdot F_n(z_n), \quad \text{and} \quad f(\mathbf{x}) = f_1(x_1) \cdot \ldots \cdot f_n(x_n).$$

In this case, each X_i can be generated independently of the others by using any of the techniques described for univariate functions.

Example 2.5.1. *To draw uniformly distributed points $\mathbf{X} = (X_1, X_2)$ in the unit square $(0,1)^2$, one can draw $U_1, U_2 \sim \mathcal{U}(0,1)$ independently and set $\mathbf{X} = (U_1, U_2)$.*

Example 2.5.2. *To draw uniformly distributed points $\mathbf{X} = (X_1, X_2)$ on the unit circle $(0,1)^2$, $B = \{(x,y) : x_1^2 + x_2^2 \leq 1\}$, one can consider a transformation in polar coordinates,*

$$\begin{cases} X_1 = R\cos\Theta \\ X_2 = R\sin\Theta \end{cases} \qquad \text{with } R \in [0,1],\ \Theta \in [0, 2\pi).$$

Since $\mathbf{X}$ is uniformly distributed on B, the PDF will be

$$f_{\mathbf{x}}(x_1, x_2) = \frac{1}{\pi} \qquad \forall (x_1, x_2) \in B$$

and using the transformation,

$$f_{\rho,\theta}(\rho, \theta) = f_{\mathbf{x}}(x_1(\rho,\theta), x_2(\rho,\theta)) \left| \frac{\partial(X_1, X_2)}{\partial(\rho, \theta)} \right| = \frac{1}{\pi}\rho = 2\rho \cdot \frac{1}{2\pi}$$

Hence, (R, Θ) are independent with $\Theta \sim \mathcal{U}(0, 2\pi)$, that is, $f_\Theta(\theta) = \frac{1}{2\pi}$, and $f_R(\rho) = 2\rho$ – the joint distribution is indeed factored into the product of two distributions in the two variables r and θ, respectively. Moreover,

$$F_R(z) = P(R \leq z) = \int_0^z 2\rho d\rho = z^2$$

so that, by inversion, we have $R = \sqrt{U}$ with $U \sim \mathcal{U}(0,1)$ and, finally,

$$\mathbf{X} = (\sqrt{U}\cos\Theta, \sqrt{U}\sin\Theta), \qquad \text{with } U \sim \mathcal{U}(0,1),\ \Theta \sim \mathcal{U}(0, 2\pi)$$

is a uniformly distributed point in the unit circle.

2.5.2 Generation from conditional distributions

Assume now that $\mathbf{X} = (X_1, \ldots, X_n)^\top \in \mathbb{R}^n$ has dependent components, but the conditional distributions of $X_j \,|\, X_1 \ldots X_{j-1}$ are known explicitly. Then the joint cumulative distribution F factorizes as

$$\begin{aligned} F(\mathbf{z}) = &\ P(\mathbf{X} \leq \mathbf{z}) = P(X_1 \leq z_1) P(X_2 \leq z_2 | X_1 \leq z_1) \cdot \ldots \cdot P(X_n \leq z_n | X_i \leq z_i,\ i < n) \\ = &\ F_{X_1}(z_1) F_{X_2|X_1}(z_2|z_1) \cdot \ldots \cdot F_{X_n|X_1,\ldots,X_{n-1}}(z_n | z_1, \ldots, z_{n-1}) \end{aligned}$$

To generate $\mathbf{X}$, we can use Algorithm 8. Again, the generation of X_i from $F_{X_i|X_1,\ldots,X_{i-1}}(z|X_1, \ldots, X_{i-1})$ can be done using any of the techniques available for univariate variables, provided we know the conditional distributions.

Algorithm 8 Generation of $\mathbf{X}$ by conditional distributions

1: Generate $X_1 \sim F_{X_1}(z)$
2: **for** $i = 2, \dots, n$ **do**
3: Generate $X_i \sim F_{X_i|X_1,\dots,X_{i-1}}(z|X_1, \dots, X_{i-1})$;
4: **end for**

2.5.3 Generation by Transformation

with copulas distributions we can decide the distributions on each component and we can also decide a dependence structure among comp.

Consider a vector $\mathbf{X} = (X_1, \dots, X_n)^\top \in \mathbb{R}^n$ with dependent components $\{X_i\}_{i=1}^n$ with marginal distribution $F_i : \mathbb{R} \to [0,1]$. Often the dependency structure is described in terms of *copulas*.

we know only the marginals, not the joint distribution

Definition 2.5.1. *A* copula *is a CDF* $C : [0,1]^n \to [0,1]$ *of* n *dependent uniform random variables* $U_1, \dots, U_n \sim \mathcal{U}(0,1)$,

$$C(u_1, \dots, u_n) = P(U_1 \le u_1, \dots, U_n \le u_n).$$

why uniform? because starting from the uniform we're able to generate other models

Remark 2.5.1. *The simplest copula is the independence copula:*

$$C(u_1, \dots, u_n) = \prod_{i=1}^{n} u_i.$$

We say that the dependency structure of $\mathbf{X}$ is described by the copula C and marginal distributions F_i if

$$F_{\mathbf{X}}(x_1, \dots, x_n) = P(X_1 \le x_1, \dots, X_n \le x_n) = C(F_1(x_1), \dots, F(x_n)).$$

the joint distribution is expressed by a function that links together all the marginals (i.e. the copula). The copula is a choice of the modeler.

The copula C contains all information on the dependence structure between the components of $(X_1, X_2, \dots, X_n)$ whereas the marginal cumulative distribution functions F_i contain all information on the marginal distributions[2].

→ we are giving to the generation of $(U_1, .., U_n)$ the job to make the components dependent among each others

The importance of the above is that the reverse of these steps can be used to sample from general classes of multivariate probability distributions. That is, given a procedure to generate a sample $(U_1, U_2, \dots, U_n)$ from the copula distribution, the required sample can be constructed as:

$$(X_1, X_2, \dots, X_n) = \left(F_1^{-1}(U_1), F_2^{-1}(U_2), \dots, F_n^{-1}(U_n)\right).$$

A typical example is the one of a *Gaussian copula*. In this case, let $\mathbf{Y} = (Y_1, \dots, Y_n) \sim N(\mathbf{0}, \Sigma)$ with $\Sigma \in \mathbb{R}^{n \times n}$ symmetric and positive definite, and denote by $\sigma_i = \sqrt{\mathrm{Var}[Y_i]}$ the standard deviation of Y_i. Then, set

$$\mathbf{U} = (U_1, \dots, U_n) = \left(\Phi\left(\frac{Y_1}{\sigma_1}\right), \dots, \Phi\left(\frac{Y_n}{\sigma_n}\right)\right)$$

being $\Phi(\cdot)$ the CDF of a standard normal random variable, and

$$C_G^\Sigma(u_1, \dots, u_n) = P(U_1 \le u_1, \dots, U_n \le u_n).$$

Such a copula is called a *Gaussian copula* with covariance matrix Σ. To generate a vector $\mathbf{X}$ with copula C_G^Σ and marginals F_i, we can therefore

we still need to clarify this step (next chapter) (2.5.4.)

1. generate $\mathbf{Y} \sim N(\mathbf{0}, \Sigma)$;
2. compute $\mathbf{U} = \left(\Phi\left(\frac{Y_1}{\sigma_1}\right), \dots, \Phi\left(\frac{Y_n}{\sigma_n}\right)\right)$;
3. compute $\mathbf{X} = (F_1^-(U_1), \dots, F_n^-(U_n))$.

$\phi(\cdot)$ brings back to $[0,1]$ and once we're in $[0,1]$ we know how to use the inversion method

[2] Any multivariate joint distribution can be written in terms of univariate marginal distribution functions and a copula which describes the dependence structure between the variables (Sklar's theorem).

Example 2.5.3. *(Taken from* `https://twiecki.io/blog/2018/05/03/copulas/`*. Quite long and verbose, but useful.) Let us measure two variables that are non-normally distributed and correlated. For example, we look at various rivers and for every river we look at the maximum level of that river over a certain time-period. In addition, we also count how many months each river caused flooding. For the probability distribution of the maximum level of the river we can look to Extreme Value Theory which tells us that maximums are Gumbel distributed. How many times flooding occurred will be modeled according to a Beta distribution which just tells us the probability of flooding to occur as a function of how many times flooding vs non-flooding occurred.*

It is pretty reasonable to assume that the maximum level and number of floodings is going to be correlated. However, here we run into a problem: how should we model that probability distribution? *Above we only specified the distributions for the individual variables, irrespective of the other one (i.e. the marginals). In reality we are dealing with a joint distribution of both of these together. Copulas allow us to decompose a joint probability distribution into their marginals (which by definition have no correlation) and a function which couples (hence the name) them together and thus allows us to specify the correlation separately. The copula is that coupling function.*

We know how to convert anything uniformly distributed to an arbitrary probability distribution. So that means we need to generate uniformly distributed data with the correlations we want. To do that, we simulate from a multivariate Gaussian with the specific correlation structure, transform so that the marginals are uniform, and then transform the uniform marginals to whatever we like.

Here are the `Matlab` *commands to sample 1000 values from the joint distribution:*

```
n = 1000; rho = .5;
rng('default') % for reproducibility
mu = 0; beta = 1;
A = 10; B = 2;

% Now "uniformify" the marginals:
U = copularnd('Gaussian',[1 rho; rho 1],n);
                          Σ
% Now we just transform the marginals again to what we want (Gumbel and Beta):
X = [evinv(U(:,1),mu,beta) betainv(U(:,2),A,B)]; ← [F1^-1(U1), F2^-1(U2)]

figure(1)
scatterhist(X(:,1),X(:,2),'Direction','out')
figure(2)
scatterhist(U(:,1),U(:,2),'Direction','out')
```

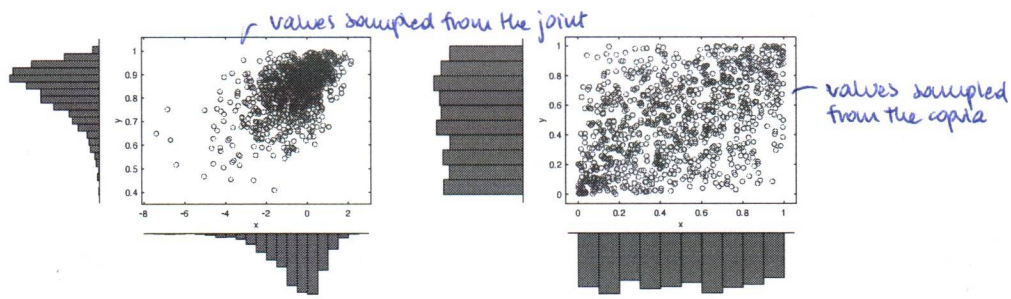


Figure 2.5: Sampled values from $\mathbf{X}$ (left) and from the copula $\mathbf{U}$ (right).

The joint plot in Figure 2.5, left, is the resulting distribution; the joint plot on the right is usually how copulas are visualized. We can also compare this latter with the joint distribution without correlations (see Figure 2.6).

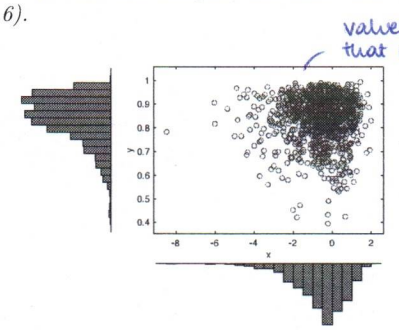


Figure 2.6: Sampled values from the joint distribution, without correlation.

```
Y(:,1) = evrnd(mu,beta,n,1);
Y(:,2) = betarnd(A,B,n,1);
figure(2)
scatterhist(Y(:,1),Y(:,2),'Direction','out')
```

Lesson learned: by using the uniform distribution as our lingua franca *we can easily induce correlations and flexibly construct complex probability distributions.*

2.5.4 Multivariate Gaussian distribution

A multivariate Gaussian random variable $\mathbf{X} \sim N(\boldsymbol{\mu}, \Sigma)$ with mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$ symmetric and positive definite has joint PDF

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}, \qquad \mathbf{x} \in \mathbb{R}^n.$$

Assuming Σ is positive definite, it can always be written as $\Sigma = AA^\top$. There are two common ways to compute the facto A:

1. *Cholesky factorization:* A is then a lower triangular matrix;
2. *Spectral decomposition:* diagonalizing $\Sigma = VDV^\top$ with $D = diag(\lambda_1, \ldots, \lambda_n)$ the matrix of eigenvalues and V the matrix of (orthonormal) eigenvectors, we set $A = VD^{1/2}$ – being $D^{1/2} = diag(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})$.

In either case, $\mathbf{X}$ can be generated by the following algorithm, starting from a vector of independent standard normal variables, and then applying an affine transformation:

Algorithm 9 Generation of $\mathbf{X} \sim N(\boldsymbol{\mu}, \Sigma)$, multivariate Gaussian

1: Given $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\Sigma \in \mathbb{R}^{n \times n}$, compute the Cholesky factorization of $\Sigma = AA^\top$;
2: Generate $\mathbf{Y} \sim N(\mathbf{0}, I)$ – that is, $\mathbf{Y} = (Y_1, \ldots, Y_n)$, $Y_i \sim N(0,1)$
3: output $\mathbf{X} = \boldsymbol{\mu} + A\mathbf{Y}$;

Clearly, $\mathbf{X}$ is multivariate Gaussian (being a linear transformation of a standard normal vector); moreover,

$$\mathbb{E}[\mathbf{X}] = \boldsymbol{\mu}, \qquad Cov[\mathbf{X}] = \mathbb{E}\left[(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^\top\right] = \mathbb{E}\left[AYY^\top A\right] = \Sigma.$$

Indeed, we recall that if $\mathbf{Y} = (Y_1, \dots, Y_n)^\top$ is a random vector, $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, then the random vector

$$\mathbf{X} = A\mathbf{Y} + \mathbf{b} \in \mathbb{R}^m$$

is said to be an *affine transformation* of $\mathbf{Y}$ and in particular:

- If $\mathbf{Y}$ has an expectation vector $\mu_{\mathbf{Y}}$, then the expectation vector of $\mathbf{X}$ is $\mu_{\mathbf{X}} = A\mu_{\mathbf{Y}} + \mathbf{b}$.
- If $\mathbf{Y}$ has a covariance matrix $\Sigma_{\mathbf{Y}}$, then the covariance matrix of $\mathbf{X}$ is $\Sigma_{\mathbf{X}} = A\Sigma_{\mathbf{Y}}A^\top$.
- If $A \in \mathbb{R}^{n\times n}$ is invertible, and $\mathbf{Y}$ has a PDF $f_{\mathbf{Y}}$, then the PDF of $\mathbf{X}$ is given by

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\det(A)} f_{\mathbf{Y}}(A^{-1}(\mathbf{x} - \mathbf{b})), \qquad \mathbf{x} \in \mathbb{R}^n.$$

2.5.5 Conditional Multivariate Gaussian distribution

based on the fact that if we have a gaussian random vector and we condition it to a subset of its components we obtain that the conditioned is still gaussian

Consider a multivariate Gaussian variable $\mathbf{X} \sim N(\boldsymbol{\mu}, \Sigma)$, and split the vector as

$$\mathbf{X} = (\underbrace{X_1, \dots, X_k}_{\mathbf{Y}}, \underbrace{X_{k+1}, \dots, X_n}_{\mathbf{Z}}) = (\mathbf{Y}, \mathbf{Z})$$

in observable ($\mathbf{Z}$) and unobservable ($\mathbf{Y}$) variables. The mean and covariance can also be split accordingly:

$$\mathbb{E}[\mathbf{X}] = (\boldsymbol{\mu}_Y, \boldsymbol{\mu}_Z), \qquad Cov[\mathbf{X}] = \Sigma = \begin{pmatrix} \Sigma_{YY} & \Sigma_{YZ} \\ \Sigma_{ZY} & \Sigma_{ZZ} \end{pmatrix}$$

where

$$\boldsymbol{\mu}_Y = \mathbb{E}[\mathbf{Y}], \qquad \boldsymbol{\mu}_Z = \mathbb{E}[\mathbf{Z}]$$

and

$$\Sigma_{YY} = \mathbb{E}\left[(\mathbf{Y} - \boldsymbol{\mu}_Y)(\mathbf{Y} - \boldsymbol{\mu}_Y)^\top\right], \quad \Sigma_{ZZ} = \mathbb{E}\left[(\mathbf{Z} - \boldsymbol{\mu}_Z)(\mathbf{Z} - \boldsymbol{\mu}_Z)^\top\right],$$
$$\Sigma_{YZ} = \mathbb{E}\left[(\mathbf{Y} - \boldsymbol{\mu}_Y)(\mathbf{Z} - \boldsymbol{\mu}_Z)^\top\right] = \Sigma_{ZY}^\top.$$

The conditional distribution of $\mathbf{Y}$ given $\mathbf{Z}$ is again a multivariate Gaussian given by

$$\mathbf{Y}|\mathbf{Z} = \mathbf{z} \sim N(\boldsymbol{\mu}_{Y|Z}, \Sigma_{Y|Z})$$

where

$$\boldsymbol{\mu}_{Y|Z} = \boldsymbol{\mu}_Y + \Sigma_{YZ}\Sigma_{ZZ}^{-1}(\mathbf{z} - \boldsymbol{\mu}_Z) \quad \text{and} \quad \Sigma_{Y|Z} = \Sigma_{YY} - \Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{ZY}.$$

To generate $\mathbf{Y}|\mathbf{Z} = \mathbf{z}$ one can, of course, factorize the covariance matrix $\Sigma_{Y|Z} = AA^\top$ (by Cholesky or spectral decomposition). This, however, can be expensive or cumbersome, if the size of $\mathbf{Y}$ is large. IN particular, there might be cases (typically in the generation of stationary Gaussian random fields) in which generating $\mathbf{X}$ is easy – even for very large dimensions – but generating $\mathbf{Y}$ e.g. by Cholesky decomposition would be very costly.

Here is an alternative to generate $\mathbf{Y}$:

Algorithm 10 Generation of a conditional multivariate Gaussian distribution $\mathbf{Y}|\mathbf{Z} = \mathbf{z}$

1: Given $\boldsymbol{\mu} \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n\times n}$, and $\mathbf{z} \in \mathbb{R}^{n-k}$, (← observations)
2: Generate $\mathbf{X} \sim N(\boldsymbol{\mu}, \Sigma)$, – that is, $\mathbf{X} = (\underbrace{X_1, \dots, X_k}_{\mathcal{Y}}, \underbrace{X_{k+1}, \dots, X_n}_{\mathcal{Z}})$
3: output $\mathbf{Y} = \mathcal{Y} + \Sigma_{YZ}\Sigma_{ZZ}^{-1}(\mathbf{z} - \mathcal{Z})$;

we need to invert a matrix (Σ_{ZZ}^{-1}) but this is still less comp. complex than Cholesky decomposition

Again, we can easily check that $\mathbf{Y}$ has the correct distribution: indeed,

$$\mathbb{E}[\mathbf{Y}] = \boldsymbol{\mu}_Y + \Sigma_{YZ}\Sigma_{ZZ}^{-1}(\mathbf{z} - \boldsymbol{\mu}_Z)$$

and, by setting $\mathbf{Y}' = \mathbf{Y} - \mathbb{E}[\mathbf{Y}] = (\mathbf{Y} - \boldsymbol{\mu}_Y) - \Sigma_{YZ}\Sigma_{ZZ}^{-1}(\mathbf{z} - \boldsymbol{\mu}_Z)$, we find

$$Cov[\mathbf{Y}] = \mathbb{E}\left[\mathbf{Y}'(\mathbf{Y}')^\top\right] = \Sigma_{YY} - \Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{ZY}.$$

Applications:

- Gaussian process regression (surrogate models, multifidelity techniques)
  observable data = data we have
  un-observable data = data we want to make a prediction on
- Kalman filters

2.6 Gaussian Process Generation

In this section we consider the generation of one among the most relevant random processes used in Monte Carlo simulation, the Gaussian process, for two main reasons: *(i)* often, emulators or surrogate models used to speed up the evaluation of more involved models are based on Gaussian Process regression and *(ii)* other stochastic processes can be generated in similar ways.

Gaussian processes can be thought of as generalizations of Gaussian random vectors. Let $I \subset \mathbb{R}^d$ and X_t, $t \in I$, a collection of random variables indexed by $t \in I$. Then, $\{x_t, t \in I\}$ is called a *stochastic process* (typically when $d = 1$ and t denotes time), or a *random field* (when $d \geq 1$ and t denotes the space variable). A Gaussian process/random field is a stochastic process for which all finite dimensional distributions are Gaussian, i.e., for every $n \in \mathbb{N}$, $\forall t_1, t_2, \ldots, t_n \in I$, the random vector $\mathbf{X} = (X_{t_1}, \ldots, X_{t_n})$ has a multivariate Gaussian distribution. Equivalently, any linear combination $Y_b = \sum_{i=1}^n b_i X_{t_i}$ has a normal distribution.

We'll try to rely on the fact that even if the gaussian process is an infinite-dimensional object, each subset of its components behave like a gaussian random vector (finite subset)

Basic ingredients in the definition of the probability distribution of a Gaussian process/random field $\{X_t, t \in I\}$ are its *mean function*

$$\mu_X : I \to \mathbb{R} \; : \quad \mu_X(t) = \mathbb{E}[X_t] \qquad \forall t \in I,$$

and its *covariance function*

$$C_X : I \times I \to \mathbb{R}, \quad C_X(t,s) = \mathbb{E}[(X_t - \mu_X(t))(X_s - \mu_X(s))] \;\; \forall s,t \in I.$$

To generate a realization of a Gaussian process with mean function μ_X and covariance function C_X at times $t_1, \ldots, t_n$ we can simply sample a multivariate normal random vector

$$\mathbf{X} = (X_{t_1}, \ldots, X_{t_n}) \sim N(\boldsymbol{\mu}, \Sigma)$$

$$\text{with} \qquad \boldsymbol{\mu} = (\mu_X(t_1), \ldots, \mu_X(t_n)), \qquad \Sigma_{ij} = C_X(t_i, t_j).$$

In particular, we say that a function $C : I \times I \to \mathbb{R}$ is non-negative definite if $\forall n \in \mathbb{N}$, $\forall t_1, t_2, \ldots, t_n \in I$, the matrix $\Sigma \in \mathbb{R}^{n \times n}$, $\Sigma_{ij} = C(t_i, t_j)$, is non-negative definite.

A Gaussian process/random field $\{X_t, t \in I\}$ is uniquely determined by the mean function $\mu_X : I \to \mathbb{R}$ and a symmetric and non-negative definite covariance function $C_X : I \times I \to \mathbb{R}$. We use the notation $X \sim N(\mu_X, C_X)$ or $X \sim GP(\mu_X, C_X)$ to denote a Gaussian process $\{X_t, t \in I\}$ with mean function μ_X and covariance function C_X.

A Gaussian process can be generated exactly in a set of points $t_1, \ldots, t_n \in I$ by the following algorithm (in which rows 2-4 indeed generate $\mathbf{X} \sim N(\boldsymbol{\mu}, \Sigma)$):

Algorithm 11 Gaussian process generator

1: Compute $\boldsymbol{\mu} = (\mu_X(t_1), \ldots, \mu_X(t_n))$ and $\Sigma_{ij} = C_X(t_i, t_j)$;
2: Compute the Cholesky decomposition $\Sigma = AA^\top$;
3: Generate $Z_1, \ldots, Z_n \sim N(0,1)$ iid and let $\mathbf{Z} = (Z_1, \ldots, Z_n)$;
4: Return $\mathbf{X} = \boldsymbol{\mu} + A\mathbf{Z}$;

A more efficient algorithm, avoiding the costly Cholesky decomposition of Σ, can be implemented in the case of a stationary Gaussian process/random field.

A Gaussian process $\{X_t, t \in I\}$ is *weakly stationary* if $C_X(t,s)$ only depends on $(s-t)$ and is *(strongly) stationary* if it is weakly stationary and $\mu_X(t)$ does not depend on t.

Assume that now we have generated already $\mathbf{Z} = (X_{t_1}, \ldots, X_{t_n})$ and we want to generate new values $\mathbf{Y} = (X_{t_{n+1}}, \ldots, X_{t_m})$ conditional to the previous generated one. This can be done by Algorithm 10 for conditoned multivariate Gaussian variables.