3
Generation of non-uniform random numbers: non-correlated values

3.1
General method

We have already described in section 2.4 a method that allows one, in principle, to generate values of a random variable distributed according to a given probability distribution function $f_{\hat{\mathbf{x}}}(x)$ based upon the relation $\hat{\mathbf{x}} = F^{-1}_{\hat{\mathbf{x}}}(\hat{\mathbf{u}})$ being $\hat{\mathbf{u}}$ a $\hat{\mathbf{U}}(0,1)$ random variable uniformly distributed in the interval $(0,1)$.

The main problem with this general method is the technical difficulty in finding a good implementation of the inverse cumulative distribution function $F^{-1}_{\hat{\mathbf{x}}}$. When this happens, a possibility is to use the Newton-Raphson method to find the solution of $F_{\hat{\mathbf{x}}}(x) - u = 0$. As $F'_{\hat{\mathbf{x}}}(x) = f_{\hat{\mathbf{x}}}(x)$, the algorithm proceeds by iteration as described in (2.39). After setting an initial value x_0, the recursion proceeds until $|x_{n+1} - x_n| < \epsilon$, a prefixed accuracy, for instance $\epsilon = 10^{-8}$. However, this method might be slow and it is not guaranteed that the algorithm always converges to the solution as it is known that the Newton-Raphson algorithm might get trapped in endless loops. Therefore, one needs a good deal of testing before the algorithm can actually be used with warranty of success. Of course, this method needs a good routine for the calculation of $F_{\hat{\mathbf{x}}}(x)$ which is not always the case. Imagine we want to apply it to generate random numbers distributed according to the the gamma family of distributions[1)]. The pdf of the $\hat{\mathbf{\Gamma}}(\alpha)$ distribution is given in (1.59). The cumulative function is:

$$F_{\hat{\mathbf{x}}}(x;\alpha) = \int_{-\infty}^{x} dx' f_{\hat{\mathbf{x}}}(x') = \begin{cases} 0, & x < 0, \\ \dfrac{\gamma(\alpha;x)}{\Gamma(\alpha)}, & x \geq 0, \end{cases} \tag{3.1}$$

being $\gamma(\alpha;x)$ the lower incomplete gamma function[2)]. To generate random numbers x distributed according to the gamma distribution using the inverse cdf, $x = F^{-1}_{\hat{\mathbf{x}}}(u)$, all we need, then, is a good algorithm to compute the inverse lower incomplete gamma function, $\gamma^{-1}(\alpha;x)$. Alas, this function is not one of the standard functions

1) We are simplifying notation and denote $\hat{\mathbf{\Gamma}}(\alpha) \equiv \hat{\mathbf{\Gamma}}(\alpha, \theta = 1)$. To generate a random variable distributed according to $\hat{\mathbf{\Gamma}}(\alpha,\theta)$ we simply multiply by θ a random variable distributed according to $\hat{\mathbf{\Gamma}}(\alpha)$.

2) This is defined precisely as $\gamma(\alpha;x) = \int_0^x ds\, s^{\alpha-1}\mathrm{e}^{-s}$.

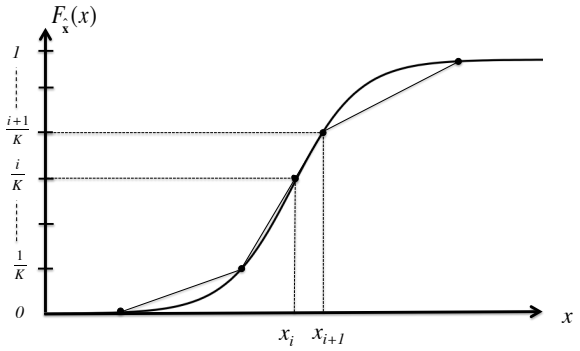

Figure 3.1 $F_{\hat{\mathbf{x}}}(x)$ and the piece-wise linear approximation used in the numerical inversion method.

you get "for free" in the most popular compilers and one might need to shop around to find a good library that implements this function. The same technical difficulty arises when finding the numerical solution of $F_{\hat{\mathbf{x}}}(x) = u$ using, for instance, Newton-Raphson method as the lower incomplete gamma function itself, $\gamma(\alpha; x)$, is not standard. Even if we manage to find suitable implementations for $\gamma(\alpha; x)$ or $\gamma^{-1}(\alpha; x)$ (or if we write those ourselves) it is more likely that the resulting algorithm will be slow as it will require a lot of calculations.

A good alternative to solving $F_{\hat{\mathbf{x}}}(x) = u$ every time we need a random number x might be to use a numerical inversion algorithm. The trick is to divide the $[0, 1]$ interval in K subintervals $\left[\frac{i}{K}, \frac{i+1}{K}\right]_{i=0,1,\ldots,K-1}$ and tabulate the inverse cumulative distribution function at the points $u_i = i/K$, i.e. compute $x_i = F^{-1}(i/K)$ for $i = 0, ..., K$. However painful this might be, the good news is that you need to do it only once. The set of numbers x_i can be kept in a file and read whenever are needed. The next step is to replace the true $F_{\hat{\mathbf{x}}}(x)$ by its piecewise-linear approximation in the interval $[\frac{i}{K}, \frac{i+1}{K}]$, see figure 3.1. Given a number $u \in [0, 1]$ it is straightforward to invert the linear interpolation to $x = F^{-1}_{\hat{\mathbf{x}}}(u)$ by means of

$$x = (Ku - i)x_{i+1} + (i + 1 - Ku)x_i, \tag{3.2}$$

where i is such that the number u belongs to the interval $[\frac{i}{K}, \frac{i+1}{K})$, or $i = [Ku]$, (integer part of Ku). A possible program could be:

```
double precision function ran_f(x,K)
implicit double precision (a-h,o-z)
```

```
dimension x(0:K)
```

```
u=ran_u()
i=K*u
ran_f=(K*u-i)*x(i+1)-(i+1-K*u)*x(i)
```

```
end function ran_f
```

Recall that in the Fortran language, there is an automatic truncation for integer numbers. So the line `i=K*u` is equivalent to `i=int(K*u)`. Before the first call to this routine, we need to initialize it by choosing a value for K and filling the entries of `x(i)` for $i = 0, \ldots, K$ with the values $x_i = F_{\hat{\mathbf{x}}}^{-1}(i/K)$. A technical problem arises if either $x_0 = -\infty$ or $x_K = +\infty$, as it happens in common distributions. The usual procedure is then set $x_0 = -\Gamma_0$ and $x_K = +\Gamma_1$ being Γ_0 and Γ_1 cut-off values. One then has to check that these cut-off values correspond to events of very low probability and can be safely discarded. Another possibility is not to cut off the distribution, but to use the numerical inversion only for the part of the distribution $f_{\hat{\mathbf{x}}}(x)$ limited to $x \in (-\Gamma_0, \Gamma_1)$ and to use another, exact or approximate method, for values of x outside this interval.

Whether this numerical solution to the inverse cumulative function will yield good quality random numbers depends on the goodness of the piecewise approximation to $F_{\hat{\mathbf{x}}}(x)$. This is, in turn, determined by the smoothness of $F_{\hat{\mathbf{x}}}(x)$ and by the number of subdivisions K of the $[0, 1]$ interval. On the other hand, once the table of x_i's has been generated, the method is very quick as it only involves sums and multiplications.

3.2 Change of variables

Sometimes a random variable which is difficult to generate can become easy after a change of variables. We gave in (1.23) the relation between the pdf's of two random variables $\hat{\mathbf{y}}$ and $\hat{\mathbf{x}}$ related by a known function $\hat{\mathbf{y}} = y(\hat{\mathbf{x}})$. A very simple case is the linear change $\hat{\mathbf{y}} = a\hat{\mathbf{x}} + b$ with $a \neq 0$. The only solution of $y = ax + b$ is $x = (y - b)/a$ and the pdf of $\hat{\mathbf{y}}$ is:

$$f_{\hat{\mathbf{y}}}(y) = \frac{1}{|a|} f_{\hat{\mathbf{x}}}\left(\frac{y-b}{a}\right). \tag{3.3}$$

For example, we have already used that if $\hat{\mathbf{x}}$ is a $\hat{\mathbf{U}}(0, 1)$, then $\hat{\mathbf{y}}$ is $\hat{\mathbf{U}}(b, a + b)$ for $a > 0$ or $\hat{\mathbf{U}}(a + b, b)$ for $a < 0$, as well as the fact that if $\hat{\mathbf{x}}$ is a Gaussian $\hat{\mathbf{G}}(0, 1)$ variable, then the change $\hat{\mathbf{y}} = \sigma\hat{\mathbf{x}} + \mu$ converts $\hat{\mathbf{y}}$ into a $\hat{\mathbf{G}}(\mu, \sigma)$ Gaussian variable.

It is possible to derive useful algorithms by using changes of variables in more than one dimension. An interesting example appears when we consider two independent Gaussian $\hat{\mathbf{G}}(0, 1)$ random variables $\hat{\mathbf{x}}_1,\ \hat{\mathbf{x}}_2$, such that the joint probability density function is:

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) = f_{\hat{\mathbf{x}}_1}(x_1) f_{\hat{\mathbf{x}}_2}(x_2) = \frac{1}{2\pi} e^{-\frac{x_1^2+x_2^2}{2}}. \tag{3.4}$$

Consider the change to polar coordinates $\hat{\mathbf{r}}$ and $\hat{\theta}$:

$$\begin{aligned} \hat{\mathbf{x}}_1 &= \hat{\mathbf{r}}\cos(\hat{\theta}), \\ \hat{\mathbf{x}}_2 &= \hat{\mathbf{r}}\sin(\hat{\theta}). \end{aligned} \tag{3.5}$$

We now apply (1.79) to this change of variables. As for given x_1, x_2 there is a unique pair of values (r, θ), the joint pdf of $\hat{\mathbf{r}}$ and $\hat{\theta}$ is:

$$f_{\hat{\mathbf{r}},\hat{\theta}}(r,\theta) = \frac{f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2)}{J\left(\frac{r,\theta}{x_1,x_2}\right)} \tag{3.6}$$

The Jacobian is $1/r$ and, after replacing $r^2 = x_1^2 + x_2^2$, we obtain:

$$f_{\hat{\mathbf{r}},\hat{\theta}}(r,\theta) = \frac{1}{2\pi} r e^{-\frac{r^2}{2}}. \tag{3.7}$$

This can be written in the form:

$$f_{\hat{\mathbf{r}},\hat{\theta}}(r,\theta) = f_{\hat{\mathbf{r}}}(r) f_{\hat{\theta}}(\theta) \tag{3.8}$$

with

$$f_{\hat{\mathbf{r}}}(r) = r e^{-\frac{r^2}{2}}, \tag{3.9}$$

$$f_{\hat{\theta}}(\theta) = \frac{1}{2\pi}, \tag{3.10}$$

which shows that $\hat{\mathbf{r}}$ and $\hat{\theta}$ are also independent random variables. More specifically, $\hat{\theta}$ is uniformly distributed in the interval $[0, 2\pi]$ that can be obtained simply as $\hat{\theta} = 2\pi\hat{\mathbf{u}}$ being $\hat{\mathbf{u}}$ a uniform $\hat{\mathbf{U}}(0,1)$ variable. $\hat{\mathbf{r}}$ follows a Rayleigh distribution and we already showed in section 2.4 that it can be generated by $\hat{\mathbf{r}} = \sqrt{-2\log(\hat{\mathbf{v}})}$. This means that the variables $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ can be generated by

$$\begin{aligned} \hat{\mathbf{x}}_1 &= \sqrt{-2\log(\hat{\mathbf{v}})}\cos(2\pi\hat{\mathbf{u}}), \\ \hat{\mathbf{x}}_2 &= \sqrt{-2\log(\hat{\mathbf{v}})}\sin(2\pi\hat{\mathbf{u}}). \end{aligned} \tag{3.11}$$

This way of obtaining two independent Gaussian $\hat{\mathbf{G}}(0,1)$ variables $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$ starting from two independent uniform $\hat{\mathbf{U}}(0,1)$ variables $\hat{\mathbf{u}}, \hat{\mathbf{v}}$ is the celebrated Box-Muller-Wiener algorithm. The main advantage of this algorithm, besides being exact and not relying upon approximations to the inverse error function, is that it uses only elementary functions (sine, cosine, log, square root) that can be found in (almost) any computing language. At the same time, it turns out to be somewhat slow as these elementary functions are computed by complicated routines. It all depends on what you need. If you just need a few thousands of Gaussian random numbers, then use the Box-Muller-Wiener algorithm. If your needs are in the millions of Gaussian random numbers you might want to implement a more efficient routine based upon numerical inversion, for example. We give now a possible implementation of the Box-Muller-Wiener algorithm. We need to know if the function is being called for an even or an odd number of times, since the second time of a pair we do not need to generate again the uniform random numbers u, v.

```
double precision function ran_gbmw()
  implicit none
  double precision ran_u
  double precision, parameter :: pi2=6.283185307179586d0
  double precision, save :: u,v
  integer, save :: icount=1
```

```
   if (icount.eq.1) then
            u=dsqrt(-2.0d0*log(ran_u()))
            v=pi2*ran_u()
            ran_gbmw=u*cos(v)
            icount=2
    else
            ran_gbmw=u*sin(v)
            icount=1
    endif
```

```
end function ran_gbmw
```

The change of variables does not need to lead from a set of n random variables to another set of exactly the same number of variables. For instance, we could consider the change $\hat{\mathbf{x}} = \hat{\mathbf{x}}_1 + \hat{\mathbf{x}}_2$ that takes from two independent variables $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$ to only one, $\hat{\mathbf{x}}$. It is known that the pdf of the sum is:

$$f_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{\infty} dx_1 f_{\hat{\mathbf{x}}_1}(x_1) f_{\hat{\mathbf{x}}_2}(x - x_1). \tag{3.12}$$

Imagine, for example, that $\hat{\mathbf{x}}_1$ follows a gamma distribution $\hat{\mathbf{\Gamma}}(\alpha_1)$ and $\hat{\mathbf{x}}_2$ follows a gamma distribution $\hat{\mathbf{\Gamma}}(\alpha_2)$. The pdf of the sum is:

$$\begin{aligned} f_{\hat{\mathbf{x}}}(x) &= \int_0^x dx_1 \frac{x^{\alpha_1 - 1} e^{-x_1}}{\Gamma(\alpha_1)} \frac{(x - x_1)^{\alpha_2 - 1} e^{-(x - x_1)}}{\Gamma(\alpha_2)} \\ &= \frac{x^{\alpha_1 + \alpha_2 - 1} e^{-x}}{\Gamma(\alpha_1 + \alpha_2)}, \end{aligned} \tag{3.13}$$

for $x \geq 0$. This is a $\hat{\mathbf{\Gamma}}(\alpha_1 + \alpha_2)$ distribution! This is called the α-addition property. A simple iteration of this rule tells us that to generate a gamma distribution whose index α is an integer number, all we have to do is to add up α independent $\hat{\mathbf{\Gamma}}(1)$ variables, i.e. variables following an exponential distribution generated, as we know, as $-\log(u)$ being u obtained from a uniform $\hat{\mathbf{U}}(0, 1)$ distribution. A simple program is:

```
double precision function ran_gamma(alpha)
  implicit none
  integer i,alpha
```

```
   ran_gamma=0.0d0
   do i=1,alpha
     ran_gamma=ran_gamma-dlog(ran_u())
   enddo
```

```
end function ran_gamma
```

If α is not an integer number, this α-additivity property allows us to reduce the problem to the case $\alpha \in (0,1)$. For this case, it is particularly appropriate the rejection methods that will be the topic of next sections (see also problem 9).

Other transformations lead also to interesting results. For instance, take 2 independent random variables $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$ with cumulative distribution functions $F_{\hat{\mathbf{x}}_1}(x_1), F_{\hat{\mathbf{x}}_2}(x_2)$ and let us define a new random variable $\hat{\mathbf{z}}$:

$$\hat{\mathbf{z}} = \max(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2). \tag{3.14}$$

The cdf of $\hat{\mathbf{z}}$ is defined as $F_{\hat{\mathbf{z}}}(z) = P(\hat{\mathbf{z}} \leq z)$, but $\hat{\mathbf{z}} \leq z$ requires $\hat{\mathbf{x}}_1 \leq z$ and $\hat{\mathbf{x}}_2 \leq z$. As $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ are independent variables, we have

$$P(\hat{\mathbf{z}} \leq z) = P(\hat{\mathbf{x}}_1 \leq z, \hat{\mathbf{x}}_2 \leq z) = P(\hat{\mathbf{x}}_1 \leq z)P(\hat{\mathbf{x}}_2 \leq z), \tag{3.15}$$

or

$$F_{\hat{\mathbf{z}}}(z) = F_{\hat{\mathbf{x}}_1}(z)F_{\hat{\mathbf{x}}_2}(z). \tag{3.16}$$

Take, for instance, the case that $F_{\hat{\mathbf{x}}_1}(x) = x^n$, $F_{\hat{\mathbf{x}}_2}(x) = x^m$ for $x \in (0,1)$. Then $\hat{\mathbf{z}}$ has a cdf $F_{\hat{\mathbf{z}}}(z) = z^{n+m}$, a sort of α-additivity property. Iterating this procedure, we can see that taking $\hat{\mathbf{z}} = \max(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ where $\hat{\mathbf{x}}_i, i = 1, \ldots, n$ are independent $\hat{\mathbf{U}}(0,1)$ uniform variables with $F_{\hat{\mathbf{x}}_i}(x) = x$, then

$$F_{\hat{\mathbf{z}}}(z) = z^n, \quad z \in [0,1], \tag{3.17}$$

and the corresponding pdf is:

$$f_{\hat{\mathbf{z}}}(z) = nz^{n-1}. \tag{3.18}$$

Hence a generation based on $\hat{\mathbf{z}} = \max(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ is an alternative to the formula $\hat{\mathbf{z}} = \hat{\mathbf{u}}^{1/n}$ that we can derive by a direct application of the inversion of the cdf. If n is not too a large number, it is faster to compute the maximum of n independent uniform random numbers than to exponentiate u to $1/n$ as this is a slow operation.

The same ideas can be used with discrete distributions. For instance, if $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$ are two discrete distributions taking the value $i = 0, 1, 2, \ldots$ with probability $p_i^{(1)}$ and $p_i^{(2)}$, respectively, then $\hat{\mathbf{x}} = \hat{\mathbf{x}}_1 + \hat{\mathbf{x}}_2$ adopts the value i with probability:

$$p_i = \sum_{k=0}^{i} p_k^{(1)} p_{i-k}^{(2)}. \tag{3.19}$$

For example, if $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ both follow a geometrical distribution with respective probabilities p_1 and p_2, i.e. $p_i^{(1)} = p_1(1-p_1)^i$ and $p_i^{(2)} = p_2(1-p_2)^i$, then

$$p_i = \frac{p_1 p_2}{p_2 - p_1}((1-p_1)^{i+1} - (1-p_2)^{i+1}). \tag{3.20}$$

It is the time now to mention another algorithm for the generation of the binomial distribution $\hat{\mathbf{B}}(N,p)$. We have shown how to implement the inversion of the cumulative function in section 2.4. We use here the fact that a binomial distribution appears as the sum of Bernoulli distributions. Therefore, we could generate N repetitions of an event with probability p and add up the positive results. The algorithm would be:

```
integer function iran_binomial2(N,p)
  implicit none
   iran_binomial2=0
   do i=1,N
        if (ran_u().lt.p) iran_binomial2=iran_binomial2+1
   enddo
end function iran_binomial2
```

This algorithm is slower that the one developed in 2.4 for moderate values of N as it implies the generation of N uniform random numbers, but it is more efficient for large values of N.

The Poisson distribution can be generated using an interesting ad hoc method. We have shown when discussing the exponential distribution in section 1.3 a relation between the Poisson and the exponential distribution. This relation says that if $\hat{\mathbf{t}}$ follows an exponential distribution with pdf:

$$f_{\hat{\mathbf{t}}}(t) = \lambda e^{-\lambda t}, \tag{3.21}$$

then the number of events occurring in a time interval $(0, 1)$ follows a Poisson $\hat{\mathbf{P}}(\lambda)$ distribution. As values t_i of this random variable can be obtained from

$$t_i = \frac{-1}{\lambda} \log u_i, \tag{3.22}$$

being u_i independent values of a $\hat{\mathbf{U}}(0, 1)$ distribution, we need to count emissions until the total time exceeds $t = 1$. Hence, if m are integer values distributed according to the Poisson distribution $\hat{\mathbf{P}}(\lambda)$, then the times t_i verify

$$\sum_{i=0}^{m} t_i \leq 1 < \sum_{i=0}^{m+1} t_i, \tag{3.23}$$

which, using (3.22), can be written as:

$$-\frac{1}{\lambda} \sum_{i=0}^{m} \log u_i \leq 1 < -\frac{1}{\lambda} \sum_{i=0}^{m+1} \log u_i, \tag{3.24}$$

or

$$\prod_{i=0}^{m} u_i \geq e^{-\lambda} > \prod_{i=0}^{m+1} u_i, \tag{3.25}$$

as the condition to verify by the number m. This can be implemented by the following program:

```
integer function iran_poisson(lambda)
  implicit none
  double precision u, lambda,ran_u
   u=ran_u()
   iran_poisson=0
   do while(u.gt.exp(-lambda))
       u=u*ran_u()
       iran_poisson=iran_poisson+1
   enddo
```

```
end function iran_poisson
```

The examples could continue forever, but our intention has been just to show that a little of thinking combined with some intuition can simplify the problem of the generation of random numbers for many complicated distributions. We will just explain a last trick that can be useful in a variety of occasions.

3.3 Combination of variables

Imagine that the pdf $f_{\hat{\mathbf{x}}}(x)$ of the random variable $\hat{\mathbf{x}}$ whose values we want to generate can be split as

$$f_{\hat{\mathbf{x}}}(x) = \sum_i p_i f_i(x), \tag{3.26}$$

where the functions $f_i(x)$ can also be considered as the pdf of some random variables $\hat{\mathbf{x}}_i$. All we need for that is non-negativity $f_i(x) \geq 0$ and normalization $\int dx f_i(x) = 1$. From the latter we derive easily

$$\sum_i p_i = 1. \tag{3.27}$$

If, furthermore, it is $p_i > 0 \; \forall i$, we can interpret p_i as the different probabilities of the outcomes of a discrete random variable $\hat{\mathbf{z}}$ that takes integer values, with pdf:

$$f_{\hat{\mathbf{z}}}(z) = \sum_i p_i \delta(z - i). \tag{3.28}$$

The generation of a value for $\hat{\mathbf{x}}$ proceeds in two steps: first we generate a value of $\hat{\mathbf{z}}$, say $\hat{\mathbf{z}} = i$ and then we generate a value of the random variable distributed according to $f_i(x)$. Using (1.135) the resulting pdf of this two-step process,

$$f_{\hat{\mathbf{x}}}(x) = \sum_i \mathrm{Prob}(\hat{\mathbf{z}} = i) f(x|\hat{\mathbf{z}} = i) = \sum_i p_i f_i(x), \tag{3.29}$$

is the pdf $f_{\hat{\mathbf{x}}}(x)$ we wanted to sample.

Let us see an example. Imagine we want to generate values of a random variable $\hat{\mathbf{x}}$ with pdf:

$$f_{\hat{\mathbf{x}}}(x) = \frac{5}{6}(1 + x^4), \qquad x \in (0, 1), \tag{3.30}$$

that we write in the form:

$$f_{\hat{\mathbf{x}}}(x) = \frac{5}{6} 1 + \frac{1}{6}(5x^4) \equiv p_1 f_1(x) + p_2 f_2(x). \tag{3.31}$$

$f_1(x) = 1$ is the pdf of a uniform $\hat{\mathbf{U}}(0, 1)$ variable. This occurs with probability $p_1 = 5/6$. Using the inverse cdf, $f_2(x) = 5x^4$ can be generated using $x = u^{1/5}$, with u a value from a $\hat{\mathbf{U}}(0, 1)$ variable. This second distribution should be used with probability $p_2 = 1/6$. The algorithm can be programmed as:

```
double precision function ran_f()
  implicit none
  double precision ran_u
```

```
  if (ran_u().gt.1.0d0/6.0d0) then
    ran_f=ran_u()
  else
    ran_f=ran_u()**0.2d0
  endif
```

```
end function ran_f
```

Some reflexion shows that this program is equivalent to the following

```
double precision function ran_f()
  implicit none
  double precision ran_u
```

```
  ran_f=ran_u()
  if (ran_u().lt.1.0d0/6.0d0) ran_f=ran_f**0.2d0
```

```
end function ran_f
```

Sometimes, the splitting given by (3.26) is not so obvious. Remember that it is essential that $p_i > 0$. Take, for instance, the pdf

$$f_{\hat{\mathbf{x}}}(x) = \frac{4}{7}(1 + 3x^2 - x^3), \qquad x \in (0, 1), \tag{3.32}$$

that we split as

$$f_{\hat{\mathbf{x}}}(x) = \frac{1}{7}(4(1-x)^3) + \frac{6}{7}(2x) \equiv p_1 f_1(x) + p_2 f_2(x) \tag{3.33}$$

so $p_1 = 1/7,\ p_2 = 6/7,\ f_1(x) = 4(1-x)^3,\ f_2(x) = 2x$, $f_1(x)$ and $f_2(x)$ can be sampled easily as $1-(1-u)^{1/4} \equiv 1-u^{1/4}$ and $u^{1/2}$, respectively. The program is:

```
double precision function ran_f()
  implicit none
  double precision ran_u
```

```
  ran_f=sqrt(ran_u())
  if (ran_u().lt.1.0d0/7.0d0) ran_f=1.0d0-dsqrt(ran_f)
```

```
end function ran_f
```

Here comes an interesting example. Imagine we need to sample the pdf:

$$f_{\hat{\mathbf{x}}}(x) = e^{-x-1/4} I_0(\sqrt{x}),\ \ x \geq 0, \tag{3.34}$$

We expand in a Taylor series the modified Bessel function I_0 to obtain:

$$f_{\hat{\mathbf{x}}}(x) = e^{-x-1/4} \sum_{i=0}^{\infty} \frac{\left(\frac{\sqrt{x}}{2}\right)^{2i}}{(i!)^2} = \sum_{i=0}^{\infty} e^{-1/4} \frac{\left(\frac{1}{4}\right)^i}{i!} e^{-x} \frac{x^i}{i!} \tag{3.35}$$

which is of the form (3.26) with $p_i = e^{-1/4}\frac{\left(\frac{1}{4}\right)^i}{i!}$, a Poisson distribution of parameter $\lambda = 1/4$ and $f_i(x) = e^{-x}\frac{x^i}{i!}$ a gamma distribution with integer parameter $\alpha = i+1$. All we need to do to generate the distribution $f_{\hat{\mathbf{x}}}(x)$ is to get an integer number i from a Poisson distribution of parameter $1/4$ and then add $i+1$ random numbers distributed according to an exponential distribution. A possible program would be:

```
double precision function ran_I0()
  implicit none
  double precision ran_gamma
  integer k,iran_poisson
  k=iran_poisson(0.25d0)+1
  ran_I0=ran_gamma(k)
end function ran_I0
```

This method of combination of variables has an interesting twist. It suffices to consider the case of two variables for which (3.29) reduces to

$$\begin{aligned} f_{\hat{\mathbf{x}}}(x) &= \text{Prob}(\hat{\mathbf{z}}=1) f(x|\hat{\mathbf{z}}=1) + \text{Prob}(\hat{\mathbf{z}}=2) f(x|\hat{\mathbf{z}}=2) \\ &= p_1 f_{\hat{\mathbf{x}}_1}(x) + p_2 f_{\hat{\mathbf{x}}_2}(x). \end{aligned} \tag{3.36}$$

We consider now the case in which the random variable $\hat{\mathbf{x}}$ can be easily generated, but not so $\hat{\mathbf{x}}_1$ (we do not care much about $\hat{\mathbf{x}}_2$ in this process). If we generated a value x from $f_{\hat{\mathbf{x}}}(x)$ using the method of combination of variables, we know that it could have come from the generation of $\hat{\mathbf{x}}_1$ or of $\hat{\mathbf{x}}_2$. Let us be more precise using Bayes theorem. Given a value x of the random variable $\hat{\mathbf{x}}$, the probability $P(\hat{\mathbf{z}}=1|x)$ that it comes from the variable $\hat{\mathbf{x}}_1$ is:

$$P(\hat{\mathbf{z}}=1|x) = \frac{f(x|\hat{\mathbf{z}}=1)P(\hat{\mathbf{z}}=1)}{f_{\hat{\mathbf{x}}}(x)} = \frac{f_{\hat{\mathbf{x}}_1}(x)p_1}{f_{\hat{\mathbf{x}}}(x)}. \tag{3.37}$$

The idea now is to generate a value of x from $f_{\hat{\mathbf{x}}}(x)$ (which is supposed to be easy) and keep this value with probability $P(\hat{\mathbf{z}}=1|x)$. In this way, we keep it only when x corresponds to $\hat{\mathbf{x}}_1$ and, effectively, we generate values of $\hat{\mathbf{x}}_1$ (which was supposed to be difficult). The price to pay is that not all values x are valid, we only keep a fraction of them. If we do not keep a value, we need to repeat the process until we generate a valid number.

Let us give an example. Let us consider $f_{\hat{\mathbf{x}}}(x) = 2x$, this can be generated as $x = \sqrt{u}$; $f_{\hat{\mathbf{x}}_1}(x) = 6x(1-x)$, this is the difficult one to generate as the cumulative function is $F_{\hat{\mathbf{x}}_1} = 3x^2 - 2x^3$ and the inverse function of a third-degree polynomial is not so easy to compute; $f_{\hat{\mathbf{x}}_2} = 3x^2$, this is also easy to generate, but we do not care. We write then the identity $f_{\hat{\mathbf{x}}}(x) = p_1 f_{\hat{\mathbf{x}}_1}(x) + p_2 f_{\hat{\mathbf{x}}_2}(x)$ in the form:

$$2x = \frac{1}{3}6x(1-x) + \frac{2}{3}3x^2 \tag{3.38}$$

which identifies $P(\hat{\mathbf{z}}=1) = p_1 = \frac{1}{3}$. Bayes theorem then tells us:

$$P(\hat{\mathbf{z}}=1|x) = \frac{f_{\hat{\mathbf{x}}_1}(x)p_1}{f_{\hat{\mathbf{x}}}(x)} = \frac{6x(1-x)\frac{1}{3}}{2x} = 1-x. \tag{3.39}$$

Then, in order to generate $f_{\hat{\mathbf{x}}_1}(x) = 6x(1-x)$ we generate a value x according to $f_{\hat{\mathbf{x}}}(x) = 2x$ using $x = \sqrt{u}$ and then accept it with probability $1-x$. The acceptance is a Bernoulli process (either we accept or not) and we do it by comparing the acceptance probability $1-x$ with a second uniform random number v: if $v < 1-x$ we accept it. Or equivalently, if $x < 1-v \equiv v$ as $1-v$ and v are both uniform random numbers. In other words, if $x > v$ we do not accept (we reject) the proposed number x as it is not representative of $f_{\hat{\mathbf{x}}_1}(x)$. When we reject, we need to repeat the process again until we accept the proposed number. The final algorithm to sample $f_{\hat{\mathbf{x}}_1}(x) = 6x(1-x)$ is:

```
double precision function ran_f()
  implicit none
  double precision ran_u

  do
    ran_f=sqrt(ran_u())
    if (ran_f.lt.ran_u()) exit
  enddo

end function ran_f
```

This is a very simple algorithm. Its main problem is that the average acceptance probability is $p_1 = 1/3$, so we discard 2 out of every 3 random numbers generated. This is an example of a *rejection* algorithm, where not all proposed values are kept. General rejection algorithms will be analyzed later on and they constitute one of the most powerful algorithms to generate very general probability distributions.

3.4 Multidimensional distributions

It is not easy to extend the previous techniques to the case of multidimensional distributions. The problem is to generate the values of a vector or random variables $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, ..., \hat{\mathbf{x}}_n)$ with joint probability density function $f_{\hat{\mathbf{x}}_1,...,\hat{\mathbf{x}}_n}(x_1, ..., x_n)$. The equivalent method to the inversion formula $\hat{\mathbf{x}} = F^{-1}_{\hat{\mathbf{x}}}(u)$ for one variable is based on the splitting of the joint pdf in conditional probabilities:

$$f_{\hat{\mathbf{x}}_1,...,\hat{\mathbf{x}}_n}(x_1, ..., x_n) = f_{\hat{\mathbf{x}}_1}(x_1) f_{\hat{\mathbf{x}}_2}(x_2|x_1) \cdots f_{\hat{\mathbf{x}}_n}(x_n|x_1, ..., x_{n-1}). \tag{3.40}$$

So what we do is to generate first an independent value of $\hat{\mathbf{x}}_1$ according to $f_{\hat{\mathbf{x}}_1}(x_1)$; given this value x_1 we generate a value for $\hat{\mathbf{x}}_2$ based on the conditional distribution $f_{\hat{\mathbf{x}}_2}(x_2|x_1)$; given x_1 and x_2 we generate a value for $\hat{\mathbf{x}}_3$ based on $f_{\hat{\mathbf{x}}_3}(x_3|x_1, x_2)$ and so on. The process starts by generating a value of the n-dimensional vector $(u_1, ..., u_n)$ of independent uniform $\hat{\mathbf{U}}(0, 1)$ variables, and then find $(x_1, ..., x_n)$ as

the solution of the following system of equations:

$$\begin{aligned} F_{\hat{\mathbf{x}}_1}(x_1) &= \int_{-\infty}^{x_1} dx_1' \, f_{\hat{\mathbf{x}}_1}(x_1') = u_1, \\ F_{\hat{\mathbf{x}}_2}(x_2|x_1) &= \int_{-\infty}^{x_2} dx_2' \, f_{\hat{\mathbf{x}}_2}(x_2'|x_1) = u_2, \\ \cdots\cdots\cdots\cdots &\quad \cdots \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ F_{\hat{\mathbf{x}}_n}(x_n|x_1,...,x_{n-1}) &= \int_{-\infty}^{x_n} dx_n' \, f_{\hat{\mathbf{x}}_n}(x_n'|x_1,...,x_{n-1}) = u_n. \end{aligned} \tag{3.41}$$

Neither the calculation of n cumulative distribution functions nor the solution of this system of equations are easy to carry out in most cases of interest. Still we can give some examples of the application of this procedure. We consider $n = 2$, and let the random variables $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$ with joint pdf:

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2) = \begin{cases} 2x_1x_2e^{-x_1(1+x_2^2)}, & \text{if } x_1 \geq 0, x_2 \geq 0, \\ 0, & \text{else.} \end{cases} \tag{3.42}$$

We first compute $f_{\hat{\mathbf{x}}_1}(x_1)$ and the corresponding cumulative function $F_{\hat{\mathbf{x}}_1}(x_1)$:

$$f_{\hat{\mathbf{x}}_1}(x_1) = \int_{-\infty}^{\infty} dx_2 f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2) = e^{-x_1}, \; x_1 \geq 0, \tag{3.43}$$

$$F_{\hat{\mathbf{x}}_1}(x_1) = \int_{-\infty}^{x_1} dx_1' f_{\hat{\mathbf{x}}_1}(x_1') = 1 - e^{-x_1}, \tag{3.44}$$

an exponential distribution, generated by $x_1 = -\log(u_1)$. The conditional probability $f_{\hat{\mathbf{x}}_2}(x_2|x_1)$ is

$$f_{\hat{\mathbf{x}}_2}(x_2|x_1) = \frac{f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2)}{f_{\hat{\mathbf{x}}_1}(x_1)} = 2x_1x_2e^{-x_1x_2^2} \tag{3.45}$$

and the cumulative conditional probability:

$$F_{\hat{\mathbf{x}}_2}(x_2|x_1) = \int_{-\infty}^{x_2} dx_2' f_{\hat{\mathbf{x}}_2}(x_2'|x_1) = 1 - e^{-x_1x_2^2}. \tag{3.46}$$

The solution of $F_{\hat{\mathbf{x}}_2}(x_2|x_1) = u_2$ is $x_2 = \sqrt{-\frac{\log(1-u_2)}{x_1}} \equiv \sqrt{-\frac{\log u_2}{x_1}}$. We implement this in the following program listing.

```
double precision function ran_f()
  implicit none
  double precision :: ran_u
  double precision, dimension(2):: ran_f

    ran_f(1)=-log(ran_u())
    ran_f(2)=sqrt(-log(ran_u())/ran_f(1))

end function ran_f
```

Of course, one can interchange the role of $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ and compute first $f_{\hat{\mathbf{x}}_2}(x_2)$ and then $f_{\hat{\mathbf{x}}_1}(x_1|x_2)$. In this case this would give:

$$f_{\hat{\mathbf{x}}_2}(x_2) = \int_{-\infty}^{\infty} dx_2 f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2) = \frac{2x_2}{(1+x_2^2)^2},\ x_1 \geq 0, \tag{3.47}$$

$$F_{\hat{\mathbf{x}}_2}(x_2) = \int_{-\infty}^{x_2} dx_2' f_{\hat{\mathbf{x}}_2}(x_2') = \frac{x_2^2}{1+x_2^2}. \tag{3.48}$$

Hence the solution of $F_{\hat{\mathbf{x}}_2}(x_2) = u$ is $x_2 = \sqrt{\frac{1-u}{u}}$. For $\hat{\mathbf{x}}_1$ we compute:

$$f_{\hat{\mathbf{x}}_1}(x_1|x_2) = \frac{f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2)}{f_{\hat{\mathbf{x}}_2}(x_2)} = (1+x_2^2)^2 x_1 e^{-x_1(1+x_2^2)}. \tag{3.49}$$

To make things clearer, let us define $a = 1+x_2^2$. The distribution is $f_{\hat{\mathbf{x}}_1}(x_1|x_2) = a^2 x_1 e^{-a x_1}$. So $a x_1$ is a $\hat{\mathbf{\Gamma}}(2)$ variable that we already know that can be generated as the sum of two independent exponential variables: $a x_1 = -\log(u_1) - \log(u_2)$. As $a = 1+x_2^2 = 1/u$, the final algorithm can be written as:

```
double precision function ran_f()
    implicit none
    double precision :: u,ran_u,a
    double precision, dimension(2):: ran_f

     u=ran_u()
     ran_f(2)=dsqrt((1.0d0-u)/u)
     ran_f(1)=u*(-log(ran_u())-log(ran_u()))

end function ran_f
```

In this example, it was possible to find first $f_{\hat{\mathbf{x}}_1}(x_1)$ and then $f_{\hat{\mathbf{x}}_2}(x_2|x_1)$ or to reverse the order and find first $f_{\hat{\mathbf{x}}_2}(x_2)$ and then $f_{\hat{\mathbf{x}}_1}(x_1|x_2)$. Sometimes the order in which we consider the variables does matter. Consider the distribution

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2) = \begin{cases} C\frac{x_1}{x_1^2+x_2^2+1}, & 0 \leq x_1 \leq 1, x_2 \geq 0, \\ 0, & \text{else.} \end{cases} \tag{3.50}$$

The normalization constant is $C = \frac{2(\sqrt{2}+1)}{\pi}$. We first compute $f_{\hat{\mathbf{x}}_1}$:

$$f_{\hat{\mathbf{x}}_1}(x_1) = \int_{-\infty}^{\infty} dx_2 f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2) = (\sqrt{2}+1)\frac{x_1}{\sqrt{1+x_1^2}}, \tag{3.51}$$

and the cumulative distribution:

$$\begin{aligned} F_{\hat{\mathbf{x}}_1}(x_1) &= \int_{-\infty}^{x_1} dx_1' f_{\hat{\mathbf{x}}_1}(x_1') \\ &= \int_0^{x_1} (\sqrt{2}+1)\frac{x_1'}{\sqrt{1+x_1'^2}} = (\sqrt{2}+1)(\sqrt{1+x_1^2}-1), \end{aligned} \tag{3.52}$$

and the solution of $F_{\hat{\mathbf{x}}_1}(x_1) = u_1$ is $x_1 = \left(\frac{u_1}{\sqrt{2}+1} + 1\right)^2 - 1$. Once x_1 has been found, we find $f_{\hat{\mathbf{x}}_2}(x_2|x_1)$. Indeed it is not necessary to actually compute the ratio $f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2)/f_{\hat{\mathbf{x}}_1}(x_1)$. All we effectively need to do is to consider that x_1 is now known. The distribution of $\hat{\mathbf{x}}_2$ given that $\hat{\mathbf{x}}_1$ takes the value x_1 is

$$f_{\hat{\mathbf{x}}_2}(x_2|x_1) = \frac{2A}{\pi}\frac{1}{x_2^2 + A^2}, \qquad x_2 \geq 0, \tag{3.53}$$

with $A^2 = x_1^2 + 1$. The cumulative function is $F_{\hat{\mathbf{x}}_2}(x_2|x_1) = \frac{2}{\pi}\arctan\left(\frac{x_2}{A}\right)$ and the solution of $F_{\hat{\mathbf{x}}_2}(x_2|x_1) = u_2$ is $x_2 = A\tan\left(\frac{\pi}{2}u_2\right)$. If, on the other hand we compute first $f_{\hat{\mathbf{x}}_2}$ we obtain:

$$f_{\hat{\mathbf{x}}_2}(x_2) = \int_{-\infty}^{\infty} dx_1 f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2) = \frac{\sqrt{2}+1}{2}\log\left[\frac{2+x_2^2}{1+x_2^2}\right], \tag{3.54}$$

and the cumulative distribution is:

$$F_{\hat{\mathbf{x}}_2}(x_2) = \frac{\sqrt{2}+1}{2}\left(-2\arctan(x_2) + 2\sqrt{2}\arctan\left(\frac{x_2}{\sqrt{2}}\right) + x_2\log\left[\frac{2+x_2^2}{1+x_2^2}\right]\right), \tag{3.55}$$

and it is not easy to solve $F_{\hat{\mathbf{x}}_2}(x_2) = u_2$ and find x_2.

As a final example, let us consider an innocent-looking distribution again in two dimensions:

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2) = \begin{cases} \frac{1}{\pi}, & \text{if } x_1^2 + x_2^2 \leq 1, \\ 0, & \text{else.} \end{cases} \tag{3.56}$$

This is an uniform distribution in the circle of radius 1. If we follow our general setup, we first compute $f_{\hat{\mathbf{x}}_1}(x_1)$ and the corresponding cumulative function $F_{\hat{\mathbf{x}}_1}(x_1)$:

$$\begin{aligned} f_{\hat{\mathbf{x}}_1}(x_1) &= \int_{-\infty}^{\infty} dx_2 f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2) \\ &= \int_{-\sqrt{1-x_1^2}}^{+\sqrt{1-x_1^2}} dx_2 \frac{1}{\pi} = \frac{2}{\pi}\sqrt{1-x_1^2}, \text{ for } -1 \leq x_1 \leq 1, \end{aligned} \tag{3.57}$$

$$\begin{aligned} F_{\hat{\mathbf{x}}_1}(x_1) &= \int_{-\infty}^{x_1} dx_1' f_{\hat{\mathbf{x}}_1}(x_1') = \int_{-\infty}^{x_1} dx_1' \frac{2}{\pi}\sqrt{1-x_1'^2} \\ &= \frac{1}{2} + \frac{1}{\pi}\left(x_1\sqrt{1-x_1^2} + \arcsin x_1\right), \end{aligned} \tag{3.58}$$

and solving $F_{\hat{\mathbf{x}}_1}(x_1) = u_1$ might require some work (maybe use Newton-Raphson algorithm). However, once x_1 has been found the pdf for $\hat{\mathbf{x}}_2$ is not difficult:

$$f_{\hat{\mathbf{x}}_2}(x_2|x_1) = \frac{f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2)}{f_{\hat{\mathbf{x}}_1}(x_1)} = \frac{1}{2\sqrt{1-x_1^2}}, \text{ for } |x_2| \leq \sqrt{1-x_1^2}, \tag{3.59}$$

nothing but a uniform distribution in the interval $(-\sqrt{1-x_1^2}, \sqrt{1-x_1^2})$ obtained by $x_2 = (2u_2 - 1)\sqrt{1-x_1^2}$. We will not write the corresponding program as there are more efficient methods to generate this distribution. For example, we might go to polar coordinates, as given by (3.5) to find that the joint pdf for $(\hat{\mathbf{r}}, \hat{\theta})$ is particularly simple (if we do not forget the Jacobian of the transformation):

$$f_{\hat{\mathbf{r}},\hat{\theta}}(r,\theta) = r f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2} = r\frac{1}{\pi}, \qquad \text{if } r \leq 1 \tag{3.60}$$

This is now the product of two independent distributions: $f_{\hat{\mathbf{r}},\hat{\theta}}(r,\theta) = f_{\hat{\mathbf{r}}}(r) f_{\hat{\theta}}(\theta)$, with $f_{\hat{\mathbf{r}}}(r) = 2r$ and $f_{\hat{\theta}}(\theta) = \frac{1}{2\pi}$. Hence $\hat{\theta}$ is uniformly distributed in $(0, 2\pi)$ or $\theta = 2\pi u_1$, and r is obtained from the solution of $F_{\hat{\mathbf{r}}}(r) = r^2 = u_2$ or $r = \sqrt{u_2}$. The algorithm can be written as:

```
function ran_f()
implicit double precision(a-h,o-z)
double precision,dimension(2)::ran_f
data /pi/ 3.14159265359
theta=2.0*pi*ran_u()
r=dsqrt(ran_u())
ran_f(1)=r*cos(theta)
ran_f(2)=r*sin(theta)
end function ran_f
```

The same idea can be applied to the generation of points distributed uniformly in a 3-dimensional sphere of radius 1. The pdf is:

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3}(x_1,x_2,x_3) = \begin{cases} \frac{3}{4\pi}, & \text{if } x_1^2 + x_2^2 + x_3^2 \leq 1, \\ 0, & \text{else.} \end{cases} \tag{3.61}$$

We change to spherical coordinates (r, ϕ, θ), with $\phi \in (0, \pi)$, $\theta \in (0, 2\pi)$ defined by:

$$\begin{aligned} x_1 &= r\sin\phi\sin\theta, \\ x_2 &= r\cos\phi\sin\theta, \\ x_3 &= r\cos\theta. \end{aligned} \tag{3.62}$$

The Jacobian is $J = r^2\sin\phi$, and the pdf in polar coordinates is:

$$f_{\hat{\mathbf{r}},\hat{\phi},\hat{\theta}}(r,\phi,\theta) = \frac{3}{4\pi} r^2 \sin\phi \tag{3.63}$$

Which can be written as product of independent distributions:

$$\begin{aligned} f_{\hat{\mathbf{r}},\hat{\phi},\hat{\theta}}(r,\phi,\theta) &= f_{\hat{\mathbf{r}}}(r) f_{\hat{\phi}}(\phi) f_{\hat{\theta}}(\theta) \\ &= 3r^2 \times \frac{1}{2}\sin(\phi) \times \frac{1}{2\pi}. \end{aligned} \tag{3.64}$$

It is simple now to obtain (r, ϕ, θ) from values (u_1, u_2, u_3) independently taken from an uniform $\hat{\mathbf{U}}(0,1)$ random variable:

$$r = u_1^{1/3}, \tag{3.65}$$

$$\phi = \arccos(2u_2 - 1) \Rightarrow \cos\phi = 2u_2 - 1, \sin\phi = 2\sqrt{u_2(1-u_2)}, \tag{3.66}$$

$$\theta = 2\pi u_3. \tag{3.67}$$

However, the change to spherical coordinates is not very useful for an n-dimensional sphere for $n > 3$. The pdf is:

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1,\cdots,x_n) = \begin{cases} 1/V_n, & x_1^2+\cdots+x_n^2 \leq 1 \\ 0, & \text{otherwise,} \end{cases} \tag{3.68}$$

with $V_n = \frac{\pi^{n/2}}{\Gamma(n/2+1)}$ the volume of the n-dimensional sphere of radius 1. In this case, one can use a trick that can be applied to any multidimensional distribution which only depends on the modulus $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1,\cdots,x_n) = f(r)$ with $r^2 = x_1^2 + \cdots + x_n^2$. In the previous example it is:

$$f(r) = \begin{cases} 1/V_n, & r \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.69}$$

Using the change to spherical coordinates: modulus r and $n-1$ angles $(\phi_1,\ldots,\phi_{n-1})$ we can write the pdf in these coordinates as:

$$f_{\hat{\mathbf{r}},\hat{\phi}_{\mathbf{1}},\ldots,\hat{\phi}_{\mathbf{n-1}}}(r,\phi_1,\ldots,\phi_{n-1}) = \\ J\left(\frac{r,\phi_1,\ldots,\phi_{n-1}}{x_1,\ldots,x_n}\right) f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1,\cdots,x_n). \tag{3.70}$$

As the Jacobian is $nV_n r^{n-1}\Omega(\phi_1,\ldots,\phi_{n-1})$ with $\Omega(\phi_1,\ldots,\phi_{n-1})$ the solid angle, and $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1,\cdots,x_n) = f(r)$, the pdf can be split as

$$f_{\hat{\mathbf{r}},\hat{\phi}_{\mathbf{1}},\ldots,\hat{\phi}_{\mathbf{n-1}}}(r,\phi_1,\ldots,\phi_{n-1}) = \Omega(\phi_1,\ldots,\phi_{n-1})nV_n r^{n-1} f(r) \tag{3.71}$$

$$\equiv f_{\hat{\mathbf{r}}}(r) f_{\hat{\phi}_{\mathbf{1}},\ldots,\hat{\phi}_{\mathbf{n-1}}}(\phi_1,\ldots,\phi_{n-1}), \tag{3.72}$$

with $f_{\hat{\mathbf{r}}}(r) = Cr^{n-1}f(r)$, being C a normalization constant required to make $f_{\hat{\mathbf{r}}}(r)$ a true pdf verifying $\int_0^\infty dr f_{\hat{\mathbf{r}}}(r) = 1$. All we have to do now is to sample this one-dimensional distribution $f_{\hat{\mathbf{r}}}(r)$ to obtain the modulus r. Concerning the generation of the angles $(\phi_1,\ldots,\phi_{n-1})$, we notice that their pdf $f_{\hat{\phi}_{\mathbf{1}},\ldots,\hat{\phi}_{\mathbf{n-1}}}(\phi_1,\ldots,\phi_{n-1}) = A\Omega(\phi_1,\ldots,\phi_{n-1})$ (with A another normalization constant) is the same for all distributions of the form $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1,\cdots,x_n) = f(r)$. Then, if we can find one pdf $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1,\cdots,x_n)$ which depends only on the modulus, and generate points $(x_1,\cdots,x_n)$ according to this distribution then the obtained angles can be used in any other pdf of the same form. One pdf that depends only on the modulus r and that can be sampled easily is the product of Gaussian pdf's:

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1,\cdots,x_n) = \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-x_i^2/2} = \frac{1}{(2\pi)^{n/2}} e^{-r^2/2}. \tag{3.73}$$

Once the x_i's have been generated using this distribution, all that remain to do is to correct for the modulus by generating a value of r with the correct distribution $f_{\hat{\mathbf{r}}}(r)$.

In summary, the procedure to generate an n-th dimensional distribution which only depends on the modulus, $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1,\cdots,x_n) = f(r)$, is the following:

(i) Generate a value of the modulus r using the pdf $f_{\hat{\mathbf{r}}}(r) = Cr^{n-1}f(r)$, being C a normalization constant found by imposing $\int_0^\infty dr f_{\hat{\mathbf{r}}}(r) = 1$.
(ii) Generate n independent Gaussian $\hat{\mathbf{G}}(0,1)$ variables $(x_1, \ldots, x_n)$.
(iii) Use the modulus of (i) and the angles of (ii).

This las step can be done simply by rescaling all variables $(x_1, \ldots, x_n)$ obtained in (ii) such that the resulting modulus is r, i.e setting $x_i \to \frac{x_i}{\sqrt{x_1^2+\cdots+x_n^2}} r$. Let us give a computer program to implement this algorithm:

```
double precision function ran_f(x,n)
implicit none
dimension x(n)
```

```
q=0.0d0
do i=1,n
  x(i)=ran_g()
  q=q+x(i)*x(i)
enddo
q=dsqrt(q)
r=ran_fr()
x=x/q*r
```

```
end function ran_f
```

In this program, the line `r=ran_fr()` produces a values of the modulus according to the distribution $f_{\hat{\mathbf{r}}}(r)$. For example, in the distribution (3.68) it is $f_{\hat{\mathbf{r}}}(r) = Cr^{n-1}$ if $r \leq 1$ and the normalization constant is $C = n$. The cumulative distribution is $F_{\hat{\mathbf{r}}}(r) = r^n$ and this can be sampled by $r = u^{1/n}$, being u a value of an uniform $\hat{\mathbf{U}}(0,1)$ distribution.

3.5 Gaussian distribution

One of the few n-dimensional distributions that can be generated without using the rejection methods that we will explain later is the Gaussian distribution. The joint pdf is (1.80) that we repeat here,

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1, x_2, ..., x_n) = \sqrt{\frac{|\,\mathbf{A}\,|}{(2\pi)^n}} \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i-\mu_i)A_{ij}(x_j-\mu_j)\right] \tag{1.79}$$

where μ_i is the average of $\hat{\mathbf{x}}_i$ and the symmetric matrix $\mathbf{A}$ is the inverse of the correlation matrix $\mathbf{C}$:

$$\mu_i = \langle \hat{\mathbf{x}}_i \rangle, \tag{3.74}$$

$$C_{ij} = \langle (\hat{\mathbf{x}}_i - \mu_i)(\hat{\mathbf{x}}_j - \mu_j) \rangle. \tag{3.75}$$

We take the set of numbers μ_i and C_{ij} as the given parameters and we want to generate n Gaussian distributed random variables $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, ..., \hat{\mathbf{x}}_n)$ whose averages and correlations are given by (3.74) and (3.75).

The method resembles somehow Schmidt's ortonormalization procedure and it is based on the property that a linear combination of Gaussian random variables is also a Gaussian random variable. We start from n independent Gaussian random variables $\hat{\mathbf{z}}_1, ..., \hat{\mathbf{z}}_n$ of zero mean and variance one, i.e. $\langle \hat{\mathbf{z}}_i \rangle = 0$, $\langle \hat{\mathbf{z}}_i \hat{\mathbf{z}}_j \rangle = \delta_{ij}$. They are generated by our favorite Gaussian random generator (for instance, Box-Muller-Wiener). Then we start defining new random variables $\hat{\mathbf{x}}_i$ as linear combinations of the $\hat{\mathbf{z}}_i$'s in such a way that (3.74) and (3.75) are satisfied.

The first variable $\hat{\mathbf{x}}_1$ is obtained as:

$$\hat{\mathbf{x}}_1 = b_{11}\hat{\mathbf{z}}_1 + \mu_1. \tag{3.76}$$

As linear combination of a Gaussian variable, $\hat{\mathbf{x}}_1$ is also Gaussian. Also, it is clear that $\langle \hat{\mathbf{x}}_1 \rangle = \mu_1$. To find b_{11} we use $C_{11} = \langle (\hat{\mathbf{x}}_1 - \mu_1)^2 \rangle = b_{11}^2 \langle \hat{\mathbf{z}}_1^2 \rangle = b_{11}^2$ or $b_{11} = C_{11}^{1/2}$.

Next we write the second variable as:

$$\hat{\mathbf{x}}_2 = b_{21}\hat{\mathbf{z}}_1 + b_{22}\hat{\mathbf{z}}_2 + \mu_2. \tag{3.77}$$

Again, it is a Gaussian random variable which, trivially, satisfies $\langle \hat{\mathbf{x}}_2 \rangle = \mu_2$. Constants b_{21} and b_{22} are determined by imposing the two conditions:

$$\begin{aligned} C_{12} &= \langle (\hat{\mathbf{x}}_1 - \mu_1)(\hat{\mathbf{x}}_2 - \mu_2) \rangle = b_{11} b_{21}, \\ C_{22} &= \langle (\hat{\mathbf{x}}_2 - \mu_2)^2 \rangle = b_{21}^2 + b_{22}^2, \end{aligned} \tag{3.78}$$

which leads to

$$b_{21} = \frac{C_{12}}{b_{11}}, \tag{3.79}$$

$$b_{22} = \left(C_{22} - b_{21}^2 \right)^{1/2}. \tag{3.80}$$

The process is iterated. To generate the j-th random variable $\hat{\mathbf{x}}_j$ we write:

$$\hat{\mathbf{x}}_j = \sum_{i=1}^{j} b_{ji}\hat{\mathbf{z}}_i + \mu_j \tag{3.81}$$

and compute constants b_{ij} by the recurrence relations

$$b_{ji} = \frac{C_{ji} - \sum_{k=1}^{i-1} b_{ik} b_{jk}}{b_{ii}}, \qquad j = 1 \ldots, n, \;\; i = 1, \ldots, j-1, \tag{3.82}$$

$$b_{jj} = \sqrt{C_{jj} - \sum_{k=1}^{j-1} b_{jk}^2}. \tag{3.83}$$

A possible program implementing this algorithm is:

```
subroutine generab(b,c,n)
implicit none
dimension c(n,n),b(n,n)
do j=1,n
```

```
  do i=1,j
     z=c(j,i)
     do  k=1,i-1
        z=z-b(j,k)*b(i,k)
     enddo
     if (i.lt.j) then
        b(j,i)=z/b(i,i)
     else
        b(j,j)=sqrt(z)
     endif
   enddo
 enddo
 end subroutine generab

function ran_gn(n,mu,b)
implicit double precision(a-h,o-z)
double precision,dimension(n)::ran_gn
double precision,dimension(n)::mu
double precision,dimension(n)::z
double precision,dimension(n,n)::b
```

```
 do j=1,n
   z(j)=ran_g()
   ran_gn(j)=mu(j)
   do i=1,j
       ran_gn(j)=ran_gn(j)+b(j,i)*z(i)
    enddo
 enddo
```

```
end function ran_gn
```

Where we first call the subroutine `genera(b,c,n)` to generate the matrix `b` from the correlation matrix `c` and then we can call the vector function `ran_gn(n,mu,b)` which returns the desired vector of Gaussian random numbers.

Note that the number of operations needed to carry on this process of generation of n Gaussian random variables increases as n^2. There are specific cases in which the efficiency of the generation can be greatly improved. This is discussed in appendix 13.

3.6 Rejection methods

We have already seen in section 3.3 a rejection method at work. The procedure we used was to propose a value of the random variable $\hat{\mathbf{x}}$ we wanted to sample and then decide whether we keep this proposed value or not. We now study in more detail some details of this kind of methods. We advance that rejection methods are the ones that best suit the generation of highly (and not so highly) dimensional sets of random variables. Still, for the sake of clarity, we analyze first an example with only

one variable. Let us consider a random variable $\hat{\mathbf{x}}$ whose pdf is

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} C\exp(-\frac{x^2}{2}), & -1 \leq x \leq 1, \\ 0, & x \notin (-1,1). \end{cases} \tag{3.84}$$

A cut-off Gaussian distribution limited to the interval $(-1,1)$. It is possible to obtain the normalization constant from:

$$1 = \int_{-\infty}^{\infty} dx\, f_{\hat{\mathbf{x}}}(x) = C\int_{-1}^{1} dx\, \exp(-\frac{x^2}{2}) = C\,\mathrm{erf}(1/\sqrt{2})\sqrt{2\pi}, \tag{3.85}$$

or $C = (\mathrm{erf}(1/\sqrt{2})\sqrt{2\pi})^{-1} = 0.584369\ldots$. However, we do not need to know the value of the normalization constant C. As we will see, this is one of the nice features that make rejection methods so useful. Remember now what a pdf means: $f_{\hat{\mathbf{x}}}(x)$ is a measure of the probability of finding a value of the random variable around x. Hence, if e.g. $f_{\hat{\mathbf{x}}}(x_1) = 2f_{\hat{\mathbf{x}}}(x_2)$, then x_1 is twice as probable to appear as x_2. What we will do is to generate values uniformly in the interval $(-1,1)$ and accept the proposed value, x, with a probability proportional to its pdf $f_{\hat{\mathbf{x}}}(x)$. It seems obvious that those values that make $f_{\hat{\mathbf{x}}}(x)$ larger will be accepted more often than those that make $f_{\hat{\mathbf{x}}}(x)$ small, what is, at least qualitatively, what we want to achieve. Summing up, we propose the following steps:

1.- Propose a value x sampled from the $\hat{\mathbf{U}}(-1,1)$, uniform distribution, i.e. $x = 2u - 1$.
2.- Accept that value with a probability, $h(x)$, proportional to $f_{\hat{\mathbf{x}}}(x) \propto \exp(-\frac{x^2}{2})$.

In step 2, it is important to stress that the acceptance probability, $h(x)$, must be a number between 0 and 1. Recall that $f_{\hat{\mathbf{x}}}(x)$ is non-negative and normalized but, otherwise, it might take any value between 0 and ∞. The acceptance probability is $\alpha f_{\hat{\mathbf{x}}}(x)$ with α a real number carefully chosen to ensure that the resulting probability indeed does not exceed the value 1. Otherwise, α is arbitrary. In the example we are considering, the resulting acceptance probability is $\alpha C\exp(-\frac{x^2}{2})$. As $\exp(-\frac{x^2}{2}) \leq 1$, $\forall x$, all we need is that $\alpha C \leq 1$ and we take the simplest (and, as we will se, also the more convenient) choice $\alpha = 1/C$ so the acceptance probability of a proposed value x is $h(x) = \exp(-\frac{x^2}{2})$.

How is the acceptance process performed? This is a Bernoulli process: either we accept the proposed value with probability $\exp(-\frac{x^2}{2})$ or we do not. This decision is done by comparing a uniform $\hat{\mathbf{U}}(0,1)$ random number with the acceptance probability. If the random number is smaller that this probability, the proposed value is accepted. Otherwise, if the random number is larger than the acceptance probability, it is rejected. What do we do if we reject the value? Answer: propose a new one. This algorithm can be programmed as:

```
do
    x=2*ran_u()-1
    if(ran_u().lt.exp(-0.5*x*x)) exit
enddo
```

We see in this simple example, the power of the rejection method. The algorithm is really simple and requires only two lines of code. Try to do the same by using the method based on the inversion of the cumulative function or $x = F_{\hat{\mathbf{x}}}^{-1}(u)$. The only problem with the rejection method could be that the rejection step is taken too often, requiring a large number of trial proposals before a number is actually generated. We will learn how to compute this average acceptance probability, or the average number of proposals we must make before accepting one value.

Let us now define what we mean by a general rejection method. It is based upon the two following steps:
1.- Propose a value x for the random variable distributed according to a given probability density function $g(x)$.
2.- Accept that value x with a probability $h(x)$. Recall that this probability must satisfy $0 \leq h(x) \leq 1,\ \forall x$.

To connect with the previous example, we had taken there:

$$g(x) = \begin{cases} \frac{1}{2}, & x \in (-1,1) \\ 0, & x \notin (-1,1) \end{cases} \tag{3.86}$$

$$h(x) = \exp\left(-\frac{x^2}{2}\right) \tag{3.87}$$

For the acceptance step, we consider a Bernoulli random variable $\hat{\mathbf{B}}$ which takes the value $\hat{\mathbf{B}} = 1$ (acceptance) with probability $h(x)$ and the value $\hat{\mathbf{B}} = 0$ (rejection) with probability $1 - h(x)$. During the proposal and acceptance/rejection we generate a two-dimensional random variable $(\hat{\mathbf{x}}, \hat{\mathbf{B}})$ with joint distribution $f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x, n)$. The proposal $g(x)$ can give rise to two values of $\hat{\mathbf{B}}$, this implies: $g(x) = f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x, \hat{\mathbf{B}} = 0) + f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x, \hat{\mathbf{B}} = 1)$. We are interested on those values x which result after acceptance. They are distributed according to $f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{B}} = 1)$ which we want to be identical to the given $f_{\hat{\mathbf{x}}}(x)$ we wish to sample. According to the definition of conditional pdf, this distribution is given by:

$$f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{B}} = 1) = \frac{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}{\text{Prob}(\hat{\mathbf{B}} = 1)} = \frac{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}{\int_{-\infty}^{\infty} f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)\,dx}. \tag{3.88}$$

According to our review for conditional pdf's, we have:

$$h(x) = \text{prob}(\hat{\mathbf{B}} = 1|x) = \frac{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,0) + f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)} = \frac{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}{g(x)}. \tag{3.89}$$

Replacing $f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1) = h(x)g(x)$ in (3.88) we obtain:

$$f_{\hat{\mathbf{x}}}(x) = f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{B}} = 1) = \frac{h(x)g(x)}{\int_{-\infty}^{\infty} h(x)g(x)\,dx}, \tag{3.90}$$

as the pdf resulting of the double proposal/acceptance steps. As it should be, this pdf is properly normalized.

Summing up, to sample $f_{\hat{\mathbf{x}}}(x)$ we need to find two functions $h(x)$ and $g(x)$ such that $f_{\hat{\mathbf{x}}}(x)$ can be split as $f_{\hat{\mathbf{x}}}(x) = Ag(x)h(x)$, being A a constant. Note that this implies:

$$g(x)h(x)f_{\hat{\mathbf{x}}}(y) = g(y)h(y)f_{\hat{\mathbf{x}}}(x), \qquad \forall x, y, \tag{3.91}$$

the so-called "detailed-balance condition"[3)]

The efficiency of a rejection method depends on the average probability of acceptance, ϵ. If ϵ is very small, then we need a large average number of trials to generate a single value of $\hat{\mathbf{x}}$. The average acceptance probability can be computed using:

$$\epsilon = \int_{-\infty}^{\infty} P(\hat{\mathbf{B}} = 1|x)g(x)\,dx = \int_{-\infty}^{\infty} h(x)g(x)\,dx = \langle h \rangle. \tag{3.92}$$

This implies that, given $g(x)$ the efficiency increases with increasing $h(x)$ and, hence, it is convenient to take $h(x)$ as large as possible, always keeping the condition $0 \leq h(x) \leq 1$. The average acceptance probability is related to the normalization constant. In the previous example, we have

$$\epsilon = \int_{-1}^{1} \frac{1}{2} \exp(-\frac{x^2}{2})\,dx = \frac{1}{2C}. \tag{3.93}$$

If we know $C = (\mathrm{erf}(1/\sqrt{2})\sqrt{2\pi})^{-1} = 0.584369\ldots$ we get $\epsilon = 0.855624\ldots$, a large average acceptance probability. If we did not know C we could derive from the numerical estimation of ϵ.

Once we have understood how rejection methods work, we could have generated (3.84) using the splitting:

$$g(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}), \quad x \in (-\infty, \infty), \tag{3.94}$$

$$h(x) = \begin{cases} 1, & \text{if } x \in (-1, 1), \\ 0, & \text{if } x \notin (-1, 1). \end{cases} \tag{3.95}$$

If we multiply these two functions we obtain $g(h)h(x) \propto f_{\hat{\mathbf{x}}}(x)$, which is all that we need. To implement this option, we follow the next steps: (1) generate x according to the usual (not truncated) Gaussian distribution $\hat{\mathbf{G}}(0, 1)$ and accept it with probability 1 (i.e. accept it) if the proposed value for x belongs to the interval $[-1, 1]$. The algorithm can be programmed as:

```
do
  x=ran_g()
  if(abs(x).lt.1.0d0) exit
enddo
```

3) The name is not by accident. As the generation of the different values of the random variable are done independently of each other, it can be thought, formally, as a homogeneous Markov chain in which the proposal pdf $f(x|y) = \frac{h(x)g(x)}{\int_{-\infty}^{\infty} h(x)g(x)\,dx}$ is independent on y. The stationarity condition (1.141) then leads trivially to the detailed balance condition.

The average acceptance probability of this alternative method is:

$$\epsilon = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})\, dx = \mathrm{erf}\left(1/\sqrt{2}\right) = 0.682698\ldots, \tag{3.96}$$

less than the previous algorithm. However, to see which method is more efficient we would need to compute the average time needed to compute one random number and to perform the comparison step.

Let us see another example. We want to generate values of a random variable $\hat{\mathbf{x}}$ whose pdf is:

$$f_{\hat{\mathbf{x}}}(x) = C \exp(-\frac{x^2}{2} - x^4), \qquad x \in (-\infty, \infty), \tag{3.97}$$

being C an irrelevant normalization constant given by: $C = 2\sqrt{2}e^{-1/32}K_{1/4}(1/32) = 0.643162\ldots$. We need to split $f_{\hat{\mathbf{x}}}(x) \propto g(x)h(x)$, being $g(x)$ a pdf we know how to sample numerically and $0 \leq h(x) \leq 1$. The obvious choice is:

$$g(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}), \quad x \in (-\infty, \infty), \tag{3.98}$$

$$h(x) = \exp(-x^4). \tag{3.99}$$

So we propose a value x drawn from a $\hat{\mathbf{G}}(0,1)$ Gaussian distribution and accept it with probability $\exp(-x^4)$. The algorithm can be programmed as:

```
do
  x=ran_g()
  if (ran_u().lt.exp(-x**4) exit
enddo
```

The acceptance probability is

$$\epsilon = \int_{-\infty}^{\infty} dx\, g(x)h(x) = \frac{1}{C\sqrt{2\pi}} \approx 0.620283\ldots \tag{3.100}$$

Let us see now an example with the two-dimensional distribution (3.56). We consider the following functions:

$$g(x_1, x_2) = 1/4 \text{ if } x_1 \in (-1,1), x_2 \in (-1,1), \tag{3.101}$$

$$h(x_1, x_2) = \begin{cases} 1, & \text{if } x_1^2 + x_2^2 \leq 1, \\ 0, & \text{if } x_1^2 + x_2^2 > 1. \end{cases} \tag{3.102}$$

$g(x_1, x_2)$ is the pdf of a two-dimensional point (x_1, x_2) distributed uniformly in the square $x_1 \in (-1,1), x_2 \in (-1,1)$. This can be generated from two independent random variables u_1, u_2 uniformly distributed in the $(0,1)$ interval: $x_1 = 2u_1 - 1, x_2 = 2u_2 - 1$. The proposed point is accepted (with probability 1) if it belongs to the circle $x_1^2 + x_2^2 \leq 1$ and rejected otherwise. The algorithm can be implemented as:

```
do
  x=2*ran_u()-1
  y=2*ran_u()-1
  r2=x*x+y*y
  if (r2.lt.1.0d0) exit
enddo
r=sqrt(r2)
c=x/r
s=y/r
```

We have added the last three lines that allow the calculation of the sine and cosine of the angle θ in polar coordinates. We know that this angle is uniformly distributed in the $(0, 2\pi)$ interval. So these lines compute the sine and cosine of an angle uniformly distributed in the $(0, 2\pi)$ interval without ever calling the `sin` or `cos` functions. This trick is sometimes used to replace the lines

```
v=pi2*ran_u()
ran_gbmw=u*cos(v)
ran_gbmw=u*cos(v)
```

of the Box-Muller-Wiener algorithm by

```
ran_gbmw=u*x/r
ran_gbmw=u*y/r
```

Let us now consider an n-dimensional example. Imagine we need to compute the integral:

$$I(n) = \frac{\int_{-\infty}^{\infty} dx_1 \cdots \int_{-\infty}^{\infty} dx_n e^{-(x_1^2+\cdots+x_n)^2/2-(x_1\cdots x_n)^4} \cos(x_1 \cdots x_n)}{\int_{-\infty}^{\infty} dx_1 \cdots \int_{-\infty}^{\infty} dx_n e^{-(x_1^2+\cdots+x_n)^2/2-(x_1\cdots x_n)^4}} \quad (3.103)$$

Obviously we interpret it as the average value $\langle G(x_1, \ldots, x_n)\rangle$ with respect to the pdf $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1, \ldots, x_n)$, with

$$G(x_1, \ldots, x_n) = \cos(x_1 \cdots x_n), \quad (3.104)$$

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1, \ldots, x_n) = Ce^{-(x_1^2+\cdots+x_n)^2/2-(x_1\cdots x_n)^4}, \quad (3.105)$$

C is the normalization constant of the pdf. To generate values of the n-dimensional random variable $(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{s}}_n)$ we split the pdf as:

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1, \ldots, x_n) \propto g(x_1, \ldots, x_n)h(x_1, \ldots, x_n) \quad (3.106)$$

$$g(x_1, \cdots, x_n) = \prod_{i=1}^{n} \left[\frac{1}{\sqrt{2\pi}} e^{-x_i^2/2}\right] \quad (3.107)$$

$$h(x_1, \cdots, x_n) = e^{-(x_1\cdots x_n)^4}. \quad (3.108)$$

Note that indeed $0 \leq h(x_1, \ldots, x_n) \leq 1$. The proposal $g(x_1, \ldots, x_n)$ corresponds to n independent Gaussian $\hat{\mathbf{G}}(0, 1)$ distributions. The program to implement the rejection algorithm to compute this integral is:

```
begin program rejection
  n=4
```

```
data pi/3.14159265/
call ran_ini(12345)
M=10000000
r=0.0
s=0.0
na=0
  do ij=1,M
    do b=1.0d0
      do i=1,n
          b=b*ran_g()
      enddo
      na=na+1
      if (ran_u().lt.exp(-b**4)) exit
    enddo
    g=cos(b)
    r=r+g
    s=s+g*g
  enddo
  p=dble(M)/na
  r=r/M
  s=sqrt((s/M-r*r)/M)
  write(6,*)n, r,s,p
end program rejection
```

Note that, besides the value of the integral and its error, we also write the average acceptance probability. This program runs without problems for arbitrary value of n. Some results: $I(n=2) = 0.922467 \pm 0.000037$, $I(n=10) = 0.993885 \pm 0.000011$, $I(n=40) = 0.99999666 \pm 0.00000024$, strongly suggest $\lim_{n\to\infty} I(n) = 1$, a result that can be confirmed analytically. The average acceptance probability increases with n as $\epsilon(n=2) = 0.748$, $\epsilon(n=10) = 0.977$, $\epsilon(n=40) = 0.999987$. This increase of ϵ with n, unfortunately, is not a typical feature of rejection methods. On the contrary, usually the average acceptance probability decreases with n and becomes very small for the values of n of interest. For instance, consider the n-dimensional random variable $(\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_n)$ uniformly distributed inside the n-dimensional sphere of radius 1 whose pdf is given in 3.68. If we use a simple rejection method where we propose a value for the i-th coordinate x_i taken independently from a uniform distribution in $(-1, 1)$, and accept the proposed vector $(x_1, \cdots, x_n)$ only if it satisfies $x_1^2 + \cdots + x_n^2 \leq 1$, then the average acceptance probability is $\epsilon(n) = V_n/2^n$ as 2^n is the volume of the n-dimensional hypercube $[-1, 1]^n$. This takes values $\epsilon(n=2) = 0.7854\ldots$, $\epsilon(n=10) = 2.49 \times 10^{-3}$, $\epsilon(n=100) = 1.87 \times 10^{-70}$. We see that, for $n = 100$, we need to make, on average, 10^{70} proposals to accept one! This is clearly not useful. The method of rejection needs to be modified in order to be able to deal with distributions like this one.

Let us see another example for which the performance of rejection methods greatly decreases with the number of variables. We consider the so-called ϕ^4 distribution of interest in the field of statistical mechanics of phase transitions as we will discuss in detail in chapter 5. In the particular version we consider here, we have n random

variables $(\hat{\mathbf{x}}, \ldots, \hat{\mathbf{x}}_n)$ whose joint pdf is:

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1, \cdots, x_n) = C \exp[-V(x_1, \cdots, x_n)] \tag{3.109}$$

with[4)]

$$V(x_1, \cdots, x_n) = \sum_{i=1}^{n} \left[\frac{1}{2} \left((x_{i+1} - x_i)^2 + a x_i^2 \right) + b x_i^4 \right]. \tag{3.110}$$

To sample this distribution one might be tempted to split the pdf as:

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n}(x_1, \cdots, x_n) = C \exp[-\mathcal{L}_0(x_1, \cdots, x_n)] \times \exp[-b \sum_{i=1}^{n} x_i^4], \tag{3.111}$$

and then generate a set of correlated Gaussian variables with pdf

$$g(x_1, \ldots x_n) = C \exp[-\mathcal{L}_0(x_1, \cdots, x_n)] \tag{3.112}$$

using the quadratic form (called the "free Lagrangian" in some contexts):

$$\mathcal{L}_0(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i=1}^{n} \left[(x_{i+1} - x_i)^2 + a x_i^2 \right]. \tag{3.113}$$

using the method explained in subsection 3.5 (or the more sophisticated one developed in the Appendix 13), and then accept that proposal with probability

$$h(x_1, \ldots, x_n) = \exp[-b \sum_{i=1}^{n} x_i^4]. \tag{3.114}$$

The problem with this procedure is that the average acceptance probability greatly decreases exponentially with n (see exercise 18). For moderate values of n (say $n = 100$), this acceptance probability is so small that not a single value is effectively accepted in the lifetime of the person running the simulation.

As in previous cases, it is possible to generalize the rejection methods to discrete distributions. If we propose values of a random variable distributed according to a discrete pdf:

$$g(x) = \sum_i g_i \delta(x - x_i) \tag{3.115}$$

and then accept the proposed value x_i with probability h_i (fulfilling, of course, the condition $0 \leq h_i \leq 1, \ \forall i$), then the resulting distribution of this proposal/acceptance method is:

$$f_{\hat{\mathbf{x}}}(x) = \frac{\sum_i h_i g_i \delta(x - x_i)}{\sum_i h_i g_i}. \tag{3.116}$$

So if we want to sample a discrete distribution $f_{\hat{\mathbf{x}}}(x) = \sum_i p_i \delta(x - x_i)$ we can split $p_i \propto h_i g_i$ with the conditions $0 \leq h_i \leq 1$ and $\sum_i g_i = 1$. We then propose a value x_i according to the distribution $g(x)$ and accept it with probability h_i.

4) In the sum we use the "periodic boundary conditions" convention, by which x_{n+1} is equivalent to x_1.

Let us see an example. We want to generate values of a discrete random variable $\hat{\mathbf{x}}$ that takes integer values with probability:

$$\text{Prob}(\hat{\mathbf{x}} = i) = p_i = \frac{-\log(1-a)}{a}\frac{a^i}{1+i}, \qquad i = 0, 1, 2, \ldots \tag{3.117}$$

being $0 < a < 1$ a given number. The obvius split in this case is $g_i = (1-a)a^i$, a geometric distribution $\hat{\mathbf{N}}_G(p)$ of parameter $p = 1-a$, and $h_i = (1+i)^{-1}$ which fulfills $0 \leq h_i \leq 1$. The algorithm would be

```
integer function iran_f(a)
do
   iran_f=dlog(ran_u())/dlog(a)
   if (ran_u().lt.1.0d0/(1.0d0+iran_f)) exit
enddo
end function iran_f
```

Unfortunately, not many discrete distributions g_i can be generated easily and serve as proposal for the rejection step. A simple idea, though, allows us to use a continuous random variable as a proposal suitable for the generation of a discrete distribution. The point is to relate the discrete variable $\hat{\mathbf{x}}$ we want to sample to a suitable continuous variable $\hat{\mathbf{x}}_c$. For the sake of concreteness we assume that $\hat{\mathbf{x}}$ takes only non-negative integer values $0, 1, 2, \ldots$ and let $p_i = \text{Prob}(\hat{\mathbf{x}} = i)$. We define a continuous random variable $\hat{\mathbf{x}}_c$ via the pdf defined as $f_{\hat{\mathbf{x}}_c}(x) = p_{[x]}$, being, as usual, $[x]$ the integer part of x. For instance, if $p_i = \frac{6}{\pi^2}(1+i)^{-2}$ for $i = 0, 1, 2, \ldots$, we define $f_{\hat{\mathbf{x}}_c}(x) = \frac{6}{\pi^2}(1+[x])^{-2}, x \geq 0$. It should be clear now that $f_{\hat{\mathbf{x}}_c}(x)$ is properly normalized and that the corresponding cumulative function $F_{\hat{\mathbf{x}}_c}(x)$ satisfies $F_{\hat{\mathbf{x}}_c}(i+1) - F_{\hat{\mathbf{x}}_c}(i) = p_i$. The relation between the two random variables is simply $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_c]$, as can be seen by checking that $\text{Prob}(\hat{\mathbf{x}} = i) = \text{Prob}(i \leq \hat{\mathbf{x}}_c < i+1) = F_{\hat{\mathbf{x}}_c}(i+1) - F_{\hat{\mathbf{x}}_c}(i) = p_i$, see figure 3.2

Therefore, in order to obtain values of the discrete random variable $\hat{\mathbf{x}}$, all we need to do is to generate values of the continuous random variable $\hat{\mathbf{x}}_c$ and set $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_c]$. If we use a rejection method for the generation of $\hat{\mathbf{x}}_c$, the process proceeds as for any standard continuous distribution: propose a value x from a pdf $g(x)$ and accept it with probability $h(x) = Cf_{\hat{\mathbf{x}}_c}(x)/g(x)$, choosing C such that $0 \leq h(x) \leq 1$. If accepted, the integer value $[x]$ of the obtained variable will yield us values of the discrete random variable $\hat{\mathbf{x}}$.

Let us give an example. Consider again the integer distribution $p_i = \frac{6}{\pi^2}(1+i)^{-2}$ for $i = 0, 1, 2, \ldots$. This was done before using a direct method. We introduce the random variable $\hat{\mathbf{x}}_c$ whose pdf is $f_{\hat{\mathbf{x}}_c}(x) = \frac{6}{\pi^2}(1+[x])^{-2}, x \geq 0$. For the proposal distribution we choose a pdf $g(x)$ as close as possible to $f_{\hat{\mathbf{x}}_c}(x)$ but such that it can be easily generated. A natural choice (but not the only possible one) is $g(x) = (1+x)^{-2}, x \geq 0$. The cdf is $G_{\hat{\mathbf{x}}_c}(x) = \dfrac{x}{1+x}$ and the solution of $G_{\hat{\mathbf{x}}_c}(x) = u$ is $x = \frac{u}{1-u}$. This proposed value is accepted with a probability $h(x) = Cf_{\hat{\mathbf{x}}_c}(x)/g(x)$. It turns out that $C = 1/4$ keeps the limits $0 \leq h(x) \leq 1$, so we choose:

$$h(x) = \frac{1}{4}\left(\frac{1+x}{1+[x]}\right)^2,$$

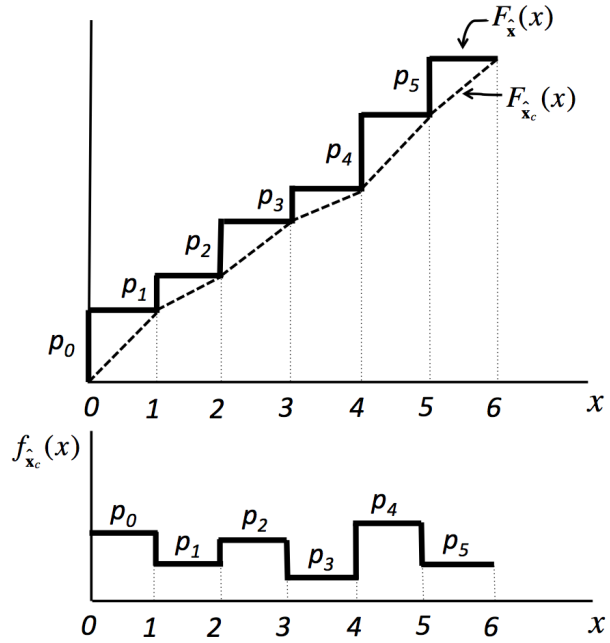

Figure 3.2 Top: piece-wise linear approximation $F_{\hat{\mathbf{x}}_c}(x)$ (dotted line) to the true cdf (solid line) of the discrete variable $F_{\hat{\mathbf{x}}_c}(x)$. Bottom: corresponding $f_{\hat{\mathbf{x}}_c}(x)$

if x is accepted, then its integer part $i = [x]$ is distributed according to p_i. The function can be implemented as:

```
integer function iran_pot2()
implicit none
double precision :: ran_u,u,x,h,acc
do
  u=ran_u()
  x=u/(1.0d0-u)
  h=0.25d0*((1.0d0+x)/(1.0d0+int(x)))**2
  if (ran_u().lt.h) exit
enddo
iran_pot2=int(x)
end function iran_pot2
```

Let us apply this method to the discrete Poisson distribution $p_i = e^{-\lambda}\frac{\lambda^i}{i!}$. We begin by defining a continuous random variable $\hat{\mathbf{x}}_c$ whose pdf is

$$f_{\hat{\mathbf{x}}_c}(x) = e^{-\lambda}\frac{\lambda^{[x]}}{[x]!}. \tag{3.118}$$

Now we need a good proposal $g(x)$. We know that $g(x)$ has to be as close as possible to $f_{\hat{\mathbf{x}}_c}(x)$. We focus now on the case of large λ for which previous algorithms were not so efficient. In this case, $f_{\hat{\mathbf{x}}_c}(x)$ comes close to a Gaussian distribution of mean and variance equal to λ. It would seem that a good choice would be the Gaussian distribution

$$g(x) = \frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{(x-\lambda)^2}{2\lambda}}. \tag{3.119}$$

However, it turns out that the acceptance probability $h(x) = C f_{\hat{\mathbf{x}}_c}(x)/g(x)$ can not satisfy the condition $h(x) \leq 1$ as the ratio $f_{\hat{\mathbf{x}}_c}(x)/g(x)$ tends to infinity with x. Instead, and based on the shape of $f_{\hat{\mathbf{x}}_c}(x)$ we propose the Cauchy distribution

$$g(x) = \frac{1}{\pi\sqrt{\lambda}} \frac{1}{1 + \left(\frac{x-\lambda}{\sqrt{\lambda}}\right)^2}. \tag{3.120}$$

which has a maximum at $x = \lambda$ and a width of the order of $\lambda^{1/2}$. The acceptance probability is

$$h(x) = C e^{-\lambda} \frac{\lambda^{[x]}}{[x]!} \pi\sqrt{\lambda} \left(1 + \left(\frac{x-\lambda}{\sqrt{\lambda}}\right)^2\right), \qquad x \geq 0. \tag{3.121}$$

The constant C is chosen under the condition $h(x) \leq 1, \forall x$. Based on the Stirling approximation at $x = \lambda$, $\lambda! \approx e^{-\lambda}\lambda^{\lambda}\sqrt{2\pi\lambda}$, we adopt $C = 1/\sqrt{\pi}$ which a simple graphical analysis shows that indeed fulfills the condition $h(x) \leq 1$ if $\lambda \geq 20$. In the cases of $\lambda < 20$ we can use other methods for the Poisson distribution. To generate a value x from the above written Cauchy distribution we set $x = \lambda + \lambda^{1/2} z$ with $z = \tan(\pi u)$. It is convenient to write the acceptance probability as:

$$h(x) = (1 + z^2)\sqrt{\pi\lambda} \exp\left[-\lambda + [x]\log\lambda - \log([x]!)\right], \tag{3.122}$$

since good routines exist for the calculation of the logarithm of the factorial[5)]. The acceptance probability of this algorithm is $C \approx 0.564$. A possible implementation is:

```
integer function iran_poisson(lambda)
implicit none
double precision :: ran_u,lambda,h,z
  do
    z=dtan(3.141592653589793d0*ran_u())
    iran_poisson=lambda+sqrt(lambda)*z
    if (iran_poisson.ge.0) then
      h=(1.0d0+z*z)*sqrt(pi*lambda)*&
          dexp(-lambda+iran_poisson*dlog(lambda)-&
          gammln(dble(iran_poisson+1)))
      if (ran_u().lt.h) exit
    endif
  enddo
end function iran_poisson
```

5) For example, one could use the function `gammln(x+1)` from the Numerical Recipes book.

Further Reading

An explicit algorithm to generate Gaussian random numbers using a numerical inversion method was provided in [5].

Zigurat

A distribution similar to (3.34) was actually needed in a problem related to field theories for systems with absorbing states[6].

Exercises

1) Check that the transformation $x = -\log(u)$ generates random numbers with pdf $f_{\hat{\mathbf{x}}}(x) = \exp(-x),\ x \geq 0$, if u is uniformly distributed in the $(0,1)$ interval. Do this by generating M random numbers and approximating the pdf $f_{\hat{\mathbf{x}}}(x)$ by a histogram of width Δx. Use $M = 10^4$ and $\Delta x = 0.01$ and check the dependence on M and Δx.
2) Repeat the previous exercise for the Gaussian $\hat{\mathbf{G}}(0,1)$ distribution using (i) the Box-Muller-Wiener algorithm (ii) the approximate algorithm inverse error function and (iii) the linear interpolation approximation. Check in each case the quality of the obtained distribution and the computer time needed.
3) Use the numerical inversion method to generate random numbers according to the pdf $f_{\hat{\mathbf{x}}}(x) = \frac{3}{4}(1-x^2),\ x \in (-1,1)$. Compare the pdf with the histogram of the obtained numbers.
4) For the pdf $f_{\hat{\mathbf{x}}}(x) = nx^{n-1},\ x \in (0,1)$ compare the efficiency of the algorithms $x = \max(u_1, \ldots, u_n)$ and $x = u^{1/n}$ as a function of n.
5) Use a rejection method to sample the pdf $f_{\hat{\mathbf{x}}}(x) = C\mathrm{e}^{-\frac{1}{2}x^2},\ x \in (0,1)$ and use the result to evaluate the integral of exercise 2.10 with the sampling method.
6) Let $x_1, \ldots, x_6$ be Gaussian random numbers of zero mean and correlations $C_{ij} = \frac{1}{2}\delta_{i-1,j} + \delta_{i,j} + \frac{1}{2}\delta_{i+1,j}$ (assume periodic boundary conditions, i.e. replace 7 by 1 and 0 by 6 whenever these numbers appear). Generate 6-dimensional number using this distribution and use those numbers to compute the average value $\langle x_1 x_2 x_3 x_4 x_5 x_6 \rangle$, comparing with the exact result given by Wick's theorem.
7) Plot pairs of numbers (x, y) obtained as $x = r\cos(\theta)$, $y = r\sin(\theta)$ being r and θ randoms variables uniformly distributed in the $(0,1)$ and $(0, 2\pi)$ intervals respectively and check graphically that these numbers are not uniformly distributed in the unit circle of center $(0,0)$ and radius 1. Design a rejection algorithm that does produce points uniformly distributed in the unit circle.
8) Design a rejection algorithm that yields random numbers uniformly distributed in the surface of an n-dimensional sphere of radius 1. Compute the rejection probability of the algorithm as a function of n.
9) Design a rejection method to generate values of a random variable $\hat{\mathbf{x}}$ distributed according to the pdf

$$f_{\hat{\mathbf{x}}}(x) = \frac{x^{\alpha-1}\mathrm{e}^{-x}}{\Gamma(\alpha)}, \qquad , x \geq 0$$

valid for $0 < \alpha < 1$.
10) Use a rejection method to generation values of the random variable $\hat{\mathbf{x}}$ distributed according the pdf $f_{\hat{\mathbf{x}}}(x) = C\exp(-\frac{1}{2}x^2 - x^4)$. Computer numerically, from the acceptance probability, the normalization constant C.
11) Repeat the previous problem for the pdf $f_{\hat{\mathbf{x}}}(x) = C\exp(\frac{1}{2}x^2 - x^4)$.
12) Use the numerical inversion of the cumulative distribution function to generate values of a discrete Poisson random variable of parameter λ. How long does it take to generate 10^6 numbers for λ=0.5, 1, 2, 5 and 10?

13) Repeat the previous problem using a rejection method using a convenient geometric distribution as proposal.
14) Generate the binomial distribution using (i) the numerical inversion of the distribution function and (ii) a rejection method.
15) Implement a numerical inversion method to generate values of the two-dimensional random variable $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ distributed according to

$$f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y) = \begin{cases} 6x, & \text{if } x+y \leq 1, x \geq 0, y \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

16) Repeat the previous exercise using a rejection method.
17) Use a rejection technique with the proposal choice

$$g_{\hat{\mathbf{x}}_c}(x) = \frac{1}{\sqrt{2a\lambda}} e^{-\sqrt{\frac{2}{a\lambda}}|x-\lambda|},$$

and a suitable value of a to generate Poisson distributed random numbers. Compare its efficiency with the Cauchy proposal given in the main text.
18) Measure the acceptance probability of the rejection method proposed to sample the multidimensional pdf (3.109-3.110) and study its dependence with the number of variables n. Get a rough estimate of that dependence by replacing $\sum_{i=1}^{n} x_i^4$ by its average value $\sum_{i=1}^{n} \langle x_i^4 \rangle = n\langle x^4 \rangle = 3n\langle x^2 \rangle^2$ assuming a Gaussian approximation.