

## Chapter 1: Simple Sampling.

If  $f(x)$  is a probability density or  $P$  is a probability measure, a Monte Carlo *sample* is a computer generated random variable,  $X$ , with law (i.e. density or measure) given by  $f$  or  $P$ . *Simple sampling* means generating  $X_1, X_2, \dots$ , which is a sequence of independent samples of  $f$  or  $P$ . Simple sampling sometimes is called *static* to distinguish it from *dynamic sampling*, which produces samples that are not independent of each other (more on this later). Few Monte Carlo computations involve only simple sampling. Still, much of the computer time in many sophisticated applications is spent doing simple sampling. Moreover, many basic Monte Carlo ideas are easily illustrated in this case.

As explained in the Introduction, we assume that there is a perfect (pseudo) random number generator that produces a sequence  $U_k$  of independent standard uniforms. The task of a sampler then is to turn a small number of independent uniforms into a random variable  $X \sim f$ . We can repeat this procedure with independent uniforms to get a sequence of independent  $X_k \sim f$ .

Simple sampling generally is used for one dimensional or low dimensional random variables. Multidimensional normals are an exception, see below. Simple sampling strategies for non Gaussian random variables may become very inefficient in high dimensions.

## 1 Mapping methods.

Suppose that  $Y \in R^m$  is a random variable with probability density  $g(y)$ , and that  $h$  is a mapping from  $R^m$  to  $R^n$ . Then  $X = h(Y)$  is another random variable with density  $f(x)$ , which is determined by the density  $g$  and the mapping  $h$ . If we can produce independent  $Y$  samples, then  $X_k = h(Y_k)$  is a simple sampling method for  $X \sim f$ . To create a mapping method that samples  $f$ , we need to find a density  $g$  that can be sampled and a mapping so that the resulting density is the desired  $f(x)$ .

### 1.1 Exponential random variables.

A scalar random variable  $T > 0$  has the *exponential distribution* with *rate parameter*  $\mu > 0$  if its probability density is

$$f(t) = \begin{cases} \mu e^{-\mu t} & \text{for } t \geq 0. \\ 0 & \text{for } t < 0. \end{cases} \quad (1)$$

It is characterized by the property that

$$P(T \in (t, t + dt) \mid T \geq t) = \mu dt. \quad (2)$$

The point of (2) is that  $\mu$  and the probability on the right do not depend on  $t$ . Exponential random variables are used to model the time to breakdown of something, such as a light bulb or hard drive. The formula (2) states that from a statistical point of view, a light bulb that has lasted up to time  $t$  is the same as a new one.

Let  $Y = U$  be a standard uniform random variable and take

$$h(u) = \frac{-1}{\mu} \ln(u) . \quad (3)$$

Let us verify that the random variable  $T = h(U)$  is an exponential with rate  $\mu$ . First, note that  $T > 0$ , since  $0 < U < 1$ . Second,

$$\begin{aligned} f(t)dt &= P(t \leq T \leq t + dt) \\ &= P\left(t \leq \frac{-1}{\mu} \ln(U) \leq t + dt\right) \\ &= P\left(e^{-\mu(t+dt)} \leq U \leq e^{-\mu t}\right) \\ &= P\left(e^{-\mu t}(1 - dt) \leq U \leq e^{-\mu t}\right) \\ &= e^{-\mu t} dt . \end{aligned}$$

The last line is the probability that the standard uniform random variable  $U$  is in an interval of length  $e^{-\mu t} dt$ , which is the length of the interval on the next to last line. This shows that the probability density for  $T$  is (1).

A *Poisson process* with rate  $\mu$  is an increasing sequence of random times  $0 = S_0 < S_1 < S_2 < \dots$ , so that the *inter arrival times*  $T_k = S_k - S_{k-1}$  are independent rate  $\mu$  exponentials. Many arrival processes are modeled as Poisson. For example,  $S_k$  could model the arrival time of the  $k^{\text{th}}$  phone call to a phone bank or the time of the  $k^{\text{th}}$  tick of a Geiger counter. We can simulate a Poisson process by generating the  $U_k$  with our (pseudo) random number generator, then taking  $T_k = \frac{-1}{\mu} \ln(U_k)$  and  $S_0 = 0$  and  $S_k = S_{k-1} + T_k$  for  $k = 1, 2, \dots$ .

Suppose the problem we are studying concerns, say, only the 10<sup>th</sup> arrival time,  $S_{10}$ , with no particular need for  $S_1, \dots, S_9$  or the  $T_k$ . For example, we might be interested in the tenth phone call to arrive at a phone bank. Sampling  $S_{10}$  as

$$S_{10} = \frac{-1}{\mu} (\ln(U_1) + \dots + \ln(U_{10})) \quad (4)$$

may not be the fastest way. Note that (4) uses ten uniforms to make a single  $S_{10}$  sample. Can we get one  $S_{10}$  sample from a single uniform? After all, we do know the probability densities. Let  $f_k(s)$  be the probability density for  $S_k$ . Since  $S_k = T_k + S_{k-1}$ , and the density of  $S_1$  is the same as that of  $T_k$ , we have

$$f_k(s) = \int_0^s f(s-s') f_{k-1}(s') ds' .$$

Doing the integrals gives (for  $s > 0$ )

$$f_2(s) = \mu^2 \int_0^s e^{-\mu(s-s')} e^{-\mu s'} ds' = \mu^2 s e^{-\mu s} ,$$

then

$$f_3(s) = \mu^3 \int_0^s e^{-\mu(s-s')} s' e^{-\mu s'} ds' = \frac{\mu^3 s^2}{2} e^{-\mu s} ,$$

and, eventually,

$$f_k(x) = \frac{\mu^k s^{k-1}}{(k-1)!} e^{-\mu s} .$$

Sampling  $S_{10}$  using (4) might be called direct (or naive) simulation. Sampling  $S_{10}$  a different way is advanced Monte Carlo.

## 1.2 Inverting the distribution function.

For a scalar random variable,  $X$ , the *distribution function* is

$$F(x) = P(X \leq x) = \int_{-\infty}^x f(x') dx' . \quad (5)$$

In this section, we assume that  $X$  has no *atoms*<sup>1</sup> and that there is a single interval,  $(a, b)$ , so that  $f(x) > 0$  if  $x \in (a, b)$  and  $f(x) = 0$  otherwise<sup>2</sup>. Under these hypotheses, for any  $u \in (0, 1)$  there is an  $x$  with  $F(x) = u$ . We call this *inverse distribution function*  $F^{-1}(u)$ . You might worry that  $x$  with  $F(x) = u$  is not unique. You should be able to convince yourself that this *almost never* happens in the technical sense that if  $B \subseteq [0, 1]$  is the set of  $u$  with  $F^{-1}(u)$  not unique, then  $P(B) = 0$ .

The inverse distribution function provides a mapping that converts a standard uniform,  $U$ , into an  $X = F^{-1}(U)$  that has density  $f(x)$ . This is the same as saying (think about this) that if  $X \sim f$  and  $U = F(X)$  then  $U$  is a standard uniform. First note that  $U \in (0, 1)$ . Next, we verify that  $P(F(X) \leq a) = a$  for any  $a \in (0, 1)$ . This is just a restatement of the first part of (5). Indeed, if  $x$  is the unique number so that  $F(x) = a$ , then  $F(X) \leq a$  is the same as  $X \leq x$ , so

$$P(F(X) < a) = P(X \leq x) = F(x) = a .$$

This leads to a simple sampling method for  $X$  if the distribution function is known. Get a  $U$  from the random number generator, then find  $X$  so that

$$F(X) = U . \quad (6)$$

The practicality of this method depends on the practicality of solving (6). In some cases  $F(x)$  is known and there is a simple algebraic formula for  $F^{-1}(u)$ . In other cases, particularly for Gaussians, you will find fast and reliable software to evaluate  $F^{-1}$ . If neither of these is possible, we may be able to solve (6) numerically, say using Newton's method – which often can be engineered to be very fast and reliable.

<sup>1</sup>An *atom* is a number,  $x_0$ , so that  $P(X = x_0) > 0$ . If  $x_0$  is an atom then  $F(x)$  has a jump discontinuity at  $x_0$ . If  $X$  has no atoms, then  $F(x)$  is a continuous function of  $x$ .

<sup>2</sup>This may be restated by saying that the *support* of  $f$  is *connected*. A standard uniform has  $a = 0$ ,  $b = 1$ , an exponential has  $a = 0$ ,  $b = \infty$ , and a Gaussian has  $a = -\infty$ ,  $b = \infty$ .

For example, an exponential random variable with density (always assuming  $t > 0$ )  $f(t) = \mu e^{-\mu t}$  has distribution function  $F(t) = \int_0^t \mu e^{-\mu t'} dt' = 1 - e^{-\mu t}$ . The equation (6) becomes  $U = 1 - e^{-\mu T}$ , whose solution is  $T = \frac{-1}{\mu} \ln(1 - U)$ . This agrees with the method of Subsection 1.1 because  $1 - U$  also is a standard uniform.

## 2 Discrete random variables.

### 2.1 One sample.

A *Bernoulli* random variable, like a coin toss, has two possible values. Let  $X$  be a Bernoulli<sup>3</sup> and call its possible values 0 and 1. The law of  $X$  is given by  $p = P(X = 0)$ . To sample a Bernoulli, let  $U$  be standard uniform and take

$$X = \begin{cases} 0 & \text{if } U \leq p \\ 1 & \text{if } U > p. \end{cases}$$

More generally, suppose  $X$  is a discrete random variable that takes the values  $0, 1, \dots, n$ , with  $p_k = P(X = k)$ , subject to the natural conditions  $p_k \geq 0$  and  $\sum_{k=0}^n p_k = 1$ . Define a discrete version of the distribution function

$$F_k = P(X < k) = \sum_{j < k} p_j, \quad F_0 = 0.$$

Then  $F_{k+1} - F_k = p_k$ , so we can sample  $X$  by taking

$$X = k \quad \text{if } F_k \leq U < F_{k+1}. \quad (7)$$

For any  $U \in (0, 1)$ , there is a unique  $k$  satisfying (7). For large  $n$  it might make sense to find  $k$  using, say, a tree search algorithm from computer science.

### 2.2 Discrete time Markov chains

A discrete time Markov chain<sup>4</sup> is a sequence of discrete random variables,  $X(t)$ , with  $t = 0, 1, \dots$ , with *transition probabilities*  $p_{jk} = P(X(t+1) = k \mid X(t) = j)$ . We assume that the initial state,  $X(0)$ , is known or can be sampled. A *sample path* is a sequence  $X(0), X(1), \dots$  that satisfies the Markov property and has the specified transition probabilities.

We can create a sample path using (7) to find  $X(t+1)$  once  $X(t)$  has been chosen. We must use the appropriate transition probabilities  $p_k = p_{jk}$  with  $j = X(t)$ . A typical Markov chain has a large number of states but a comparatively small number of allowed transitions. That is,  $n$  is large and, for each  $j$ ,  $p_{jk} > 0$  only for a small number of  $k$ . A computer program to

<sup>3</sup>This is short for “Bernoulli random variable”, not one of the Bernoulli mathematicians Johann, Jacob, etc.

<sup>4</sup>This definition omits the *Markov property*. See Stochastic Calculus for the full definition and more background.

generate sample paths (simulate the Markov chain) may be more efficient if at each transition it chooses only among the transitions having positive probability.

As we said in the Introduction, if we simulate a Markov chain in Monte Carlo, we do it for the purpose of evaluating something that is not random. Many such quantities can be calculated explicitly without Monte Carlo. See, for example, the discussion of forward and backward equations in Stochastic Calculus. Whether the deterministic is better than Monte Carlo depends on many factors, such as the size of  $n$  and the quantity being calculated.

It is possible to use simple sampling, i.e. generate independent sample paths, because we specify only *initial conditions*  $X(0)$ . Other sampling problems involving Markov chains may not have efficient simple samplers. One such is the problem of generating paths with both  $X(0)$  and  $X(T)$  specified, for some  $T$ . The naive method of generating many paths with the given  $X(0)$  and keeping the ones with the specified  $X(T)$  may be very slow if  $n$  is large.

### 2.3 Geometric random variables.

How many times do you have to toss a coin before you get the first head? Let  $X_0, X_1, \dots$ , be a sequence of independent Bernoullis with common  $p = P(X_k = 0)$ . The random variable  $N = \min(k \mid X_k = 1)$  is a *geometric* random variable. It is a discrete version of the exponential random variable and is used to model discrete time arrival processes.

One way to generate a sample  $N$  is direct simulation, as in the following C code fragment:

```
int N;
...
while ( rng() < p ) N++;  \\ Keep tossing until you get a head.
```

This uses an unknown and possibly large number of uniforms to generate a single  $N$  sample.

There is a generally faster method that uses a single  $U$  to create an  $N$ . This may be found by calculating the  $F_k$  in (7) and solving for  $k$  explicitly. To carry this out, we calculate  $p_k = P(N = k)$ . Clearly  $p_0 = P(N = 0) = P(X_0 = 1) = 1 - p$ . Next,  $p_1 = P(N = 1) = P(X_1 = 1, X_0 = 0) = (1 - p)p$ . Continuing in this way, we find

$$p_k = P(N = k) = (1 - p)p^k .$$

The random variable is called geometric because the numbers  $p_k$  form a geometric series. The reader should check that  $\sum_{k=0}^{\infty} p_k = (1 - p) \sum_{k=0}^{\infty} p^k = 1$ . More generally,

$$F_k = \sum_{j=0}^{k-1} p_j = 1 - p^k .$$

Note that  $F_0 = 0$  as it should. This, (7) becomes  $(1 - p_k) \leq U < (1 - p^{k+1})$ , which is the same as (remember that  $\ln(p) < 0$  and that  $1 - U$  has the same

distribution as  $U$ )

$$k \leq \frac{\ln(U)}{\ln(p)} < k + 1$$

The method, then, is to compute  $T = \ln(U)/\ln(p)$  and then to truncate to the greatest integer not exceeding  $T$ . This is done by the C code fragment

```
float T; int N;
float lnp = ln(p);  \\ Do not recalculate in the inner loop.
...
T = ln( rng() )/lnp;
N = (int) T;
```

The programmer must be careful not to round;  $T = 2.9$  truncates to  $N = 2$  (the right answer) but rounds to 3.

The reader will notice that  $T$  is an exponential random variable. In retrospect, it is easy to see that we get a geometric random variable with parameter  $p$  from an exponential random variable with rate  $\mu = \ln(p)$  by rounding.

## 2.4 Continuous time Markov chain.

A continuous time Markov chain has a continuous time variable,  $t$ . For each  $t$ , the state,  $X(t)$ , is one of the numbers  $0, 1, \dots, n$ . Transitions between states are modeled by *transition rates*  $\mu_{jk}$  with, for  $j \neq k$ ,

$$\mu_{jk} dt = P(X(t + dt) = k \mid X(t) = j) . \quad (8)$$

The total transition rate out of state  $j$  is

$$\mu_j = \sum_{k \neq j} \mu_{jk} ,$$

so that

$$\mu_j dt = P(X(t + dt) \neq j \mid X(t) = j) . \quad (9)$$

A common convention is to complete the transition rate matrix,  $\mu$ , by taking  $\mu_{jj} = -\mu_j$ . This gives the row sum relations  $\sum_k \mu_{jk} = 0$  and the backward equation  $\partial_t f = \mu f$ , where  $f(t)$  is the  $n$  component column vector with components  $f_j(t) = E_{j,t}[V(X(T))]$ , where  $E_{j,t}[\cdot]$  means to take the expectation under the condition that  $X(t) = j$ . The final condition for  $f$  is determined by the payout function:  $f_j(T) = V(j)$ .

A simple approximate simulation method is to choose a small  $\Delta t$  and use the numbers  $p_{jk} = \mu_{jk} \Delta t$  and  $p_{jj} = 1 - \mu_j \Delta t$  as transition probabilities for a discrete time approximate Markov chain. This has several drawbacks. For one thing, the approximate transition probabilities are not exact and it may be hard to know what  $\Delta t$  will give acceptable accuracy. Also, the method is inefficient; because the transitions probabilities are small, most of the time there will be no transition. This means that we need to use many uniforms to get a single actual transition.

A more accurate and efficient method uses the *embedded* discrete time Markov chain. Let  $S_l$  be the time of the  $l^{\text{th}}$  transition (with  $S_0 = 0$  by convention). The embedded Markov chain states are denoted  $\tilde{X}(l)$  and given by

$$\tilde{X}(l) = X(t) \text{ for } S_l \leq t < S_{l+1} .$$

The formula (9) states that  $S_{l+1} - S_l = T_l$  is an exponential random variable with rate  $\mu_j$ , with  $j = \tilde{X}(l)$ . When a transition occurs, the new state,  $\tilde{X}(l+1)$ , is chosen from the states  $k$  with  $k \neq j$  with transition probabilities

$$p_{jk} = \frac{\mu_{jk}}{\mu_j} . \tag{10}$$

The reader can verify that  $\sum_{k \neq j} p_{jk} = 1$ . The exact simulation method is: once  $\tilde{X}(l)$  is known, choose  $T_{l+1}$  by sampling an exponential with rate  $\mu_j$ , with  $j = \tilde{X}(l)$ . Then choose  $\tilde{X}(l+1)$  using the transition probabilities  $p_{jk}$  given by (10). This method is called SSA, for “stochastic simulation algorithm” in some parts of computational chemistry.

### 3 Event driven simulation.

Models from many fields may be classified as discrete event continuous time Markov chains. It often is cumbersome to describe each possible state by a single integer. The modeler would prefer to specify the state of the system by giving several pieces of information about it. Although the transitions rates  $\mu_{jk}$  are implicitly defined by the model, it might be laborious to calculate them explicitly. *Event driven simulation* is a common way to generate sample paths for such systems.

We describe the method in the context of a simple example from operations research. The reader will recognize the broad applicability of the method. The model describes a telephone help operator with impatient customers phoning in. Customers arrive according to a Poisson process with rate  $\lambda$ . That is,  $\lambda dt$  is the probability that a new customer arrives between time  $t$  and time  $t + dt$ . When a customer arrives, either she or he is connected to the operator, or, if the operator is “busy helping other customers”, the customer is put on hold – put into the *queue*. Once a customer begins service (is connected to the operator), the service time is an exponential random variable with rate  $\mu$ . That is, the customer declares herself or himself done during time  $dt$  with probability  $\mu dt$ . When the customer completes service, the server immediately starts serving another customer in the queue, if there is one. Finally, there is a frustration rate,  $\rho$ . A customer being served or waiting in the queue will become frustrated and leave without completing service in time  $dt$  with probability  $\rho dt$ .

One simulation method has a small  $\Delta t$ . For each size  $\Delta t$  time step, the program:

- Asks whether a new customer has arrived ( $P(\text{arrival}) = \lambda \Delta t$ ).

- Asks each customer whether it has become frustrated ( $P = \rho\Delta t$  per customer).
- Asks the operator whether it is finished serving its customer ( $P = \mu\Delta t$  if there is a customer in service).

Each question uses a standard uniform to answer. Most of these uniforms are wasted because, for small  $\Delta t$ , the answer is “no”. The simulation is not exact.

It is not hard to formulate the “global” Markov chain description of this system and to find the transition rates  $\mu_{jk}$ . Small changes to the model (see below) make the Markov chain description very complicated or impossible.

An event driven simulation maintains and updates two structures. One is a description of the system at time  $t$ . In the present case, that simply is a list of customers in the system (or even just the number of customers). The other structure is an *event list* of events scheduled to happen in the future. If  $t$  is the current time of the system, each element of the event list has a time  $S > t$  and information saying what kind of event it is. In the present model there are three kinds of events: arrival of a new customer, service of customer  $m$ , and frustration of customer  $m$ . Let  $n(t)$  be the number of customers present at time  $t$  and  $m_1(t), \dots, m_{n(t)}(t)$  the customers. Let  $M(t)$  be the number of customers that have arrived up to time  $t$ . The  $m_j$  are numbered so that  $m_1(t)$  is the customer in service at time  $t$ . At time  $t$ , the following events are scheduled and in the event list:

- The completion of the customer currently in service, with the completion time and the number,  $m$ , telling which customer is being served.
- The frustration of each customer present, with the frustration time and the number,  $m$ , of the customer in question.
- The arrival time of the next customer, with the arrival time.

Some of the events in the event list will not happen. For example, if a customer in service is scheduled to become frustrated at time  $S_2$  and scheduled to complete service at time  $S_1$  with  $S_2 > S_1$ , then the frustration event will not occur because the service event at time  $S_1$  will make the frustration event inoperative. In general, an event at time  $S_j$  can change the state of the system in a way that forces us to cancel or reconsider events at tie  $S_k > S_j$ . However, if  $S_j < S_k$  for all  $S_k$  in the event list, then  $S_j$  will happen. This is the basis of event driven simulation.

The data structure used for event driven simulation is a *heap*, also called *priority queue*. A heap is a collection of events together with their times and other information (here: customer number and what kind of event). The allowed operations are<sup>5</sup> `insert`, `delete`, and `deletemin`. The operation `insert(e)` adds event `e` to the heap (event list). The operation `delete(e)` removes `e` from

---

<sup>5</sup>I use the typewriter font to indicate words that could be part of computer programs, particularly C or C++ programs. Almost nothing is syntactically correct C/C++, though a programmer easily could make it real.



the heap. The operation `e = deletemin()` returns (and removes) the event with smallest time. As discussed above, this is the event that is guaranteed to take place. Any book on computer algorithms<sup>6</sup> will describe the heap data structure and its implementation. It is easy to program and easier to find on the web as shareware.

For our model, the inner loop of an event driven simulation is roughly as follows.

```

event e;
customer m;
heap eventList;
int M;          \ \ Number of customers, counting the one in service.
float S, t, T;  \ \ Event time, current time, exponential time.
...
e = eventList.deletemin();
if ( e.eventType == ServiceCompletion ) {
    m = e.customer;
    t = e.Time;
    eventList.delete( m, Frustration);
    M--;
    if ( M == 0 ) break;          \ \ If no there is no customer to put into
                                \ \ service, you're done handling this event.
    m = eventList.GetaCustomer();
    T = (1/mu)*ln( rand() ); \ \ Schedule service time for chosen customer.
    e.customer = m;
    e.eventType = ServiceCompletion;
    e.Time = t + T;
    eventList.insert(e);
    break; }
if ( e.eventType == CustomerArrival ) {
    t = e.Time;
    T = (1/rho) ln( rand() ); \ \ Schedule this customer's frustration.
    m = e.Customer;
    e.Time = t+T;
    e.eventType = Frustration;
    eventList.insert(e);
    m = m++; \ \ A label for the next customer to arrive.
    T = (1/lambda) ln( rand() ); \ \ Schedule the next customer arrivan.
    e.Customer = m;
    e.Time = T + t;
    e.eventType = CustomerArrival;
    eventList.insert(e);
    ... (possibly schedule service)
    break; }
if ( e.eventType == Frustration ) {
    ... (remove customer & possibly schedule another for service completion)

```

The reader will notice that handling one event typically leads to scheduling one or more events and adding them to the event list.

<sup>6</sup>See, for example, the book by Corman, Leiserson, and Rivest.

## 4 Rejection

Suppose  $f(x)$  and  $g(x)$  are two probability densities (or discrete probabilities) related by:

$$f(x) = \frac{1}{Z}p(x)g(x) , \quad (11)$$

where  $0 \leq p(x) \leq 1$  for all  $x$  is an  $x$  dependent *acceptance probability*. *Rejection* sampling uses the mechanism of acceptance/rejection to turn samples from the *trial* or *proposal* density  $g$  into samples from the *target* density  $f$ . The algorithm is:

- A. Generate a sample,  $X$ , from density  $g$  independent of any previous sampling.
- B. Evaluate the acceptance probability  $P = p(X)$ .
- C. Choose to *accept*  $X$  with probability  $P$  using a Bernoulli trial independent of any previous sampling: `if ( rng() < P ) accept;`
- D. If accepted, report  $X$  as the sample. If rejected (not accepted), return to step A.

If `gsamp()` is a simple sampler for  $g$  and  $p(x)$  evaluates the acceptance probability, all this happens in the C code:

```
while( p( X = gsamp() ) > rng() );
```

We can see that this method works by calculating the probability density,  $f(x)$ , of the first accepted sample and checking that it satisfies (11). To get  $X$ , you first have to propose  $X$ , then accept it. Denote by  $Z$  the overall probability of getting an acceptance in a given trial, which is given by

$$Z = \int p(x)g(x)dx . \quad (12)$$

$$\begin{aligned} f(x)dx &= P(x \leq X \leq x + dx) \\ &= P(\text{proposed } X \in (x, x + dx) \mid \text{got an acceptance}) \\ &= \frac{P(\text{proposed } X \in (x, x + dx) \text{ and got an acceptance})}{P(\text{got an acceptance})} \\ &= \frac{P(\text{proposed } X \in (x, x + dx) \text{ and accepted } X \in (x, x + dx))}{P(\text{got an acceptance})} \\ &= \frac{g(x)dxp(x)}{Z} . \end{aligned}$$

Cancelling the common factor  $dx$  gives the desired result (11).

The acceptance probability (12) governs the *efficiency* of the rejection process. The expected number of trials before until acceptance is  $1/Z$ . A practitioner designing a rejection sampling algorithm may take some time to tune

the details to get the largest possible  $Z$ . This is the same as making the acceptance probabilities  $p(x)$  as large as possible consistent with the constraint that  $p(x) \leq 1$  for all  $x$ .

We may use rejection sampling to generate a positive standard normal from a rate one exponential. The target density is

$$f(x) = \begin{cases} \frac{2}{\sqrt{2\pi}} e^{-x^2/2} & \text{if } x \geq 0. \\ 0 & \text{if } x < 0. \end{cases}$$

The trial density is

$$g(x) = \begin{cases} e^{-x} & \text{if } x \geq 0. \\ 0 & \text{if } x < 0. \end{cases}$$

We ignore the possibility that  $X < 0$  since this is impossible in the trial. The rejection sampling formula (11) becomes

$$\frac{2}{\sqrt{2\pi}} e^{-x^2/2} = \frac{1}{Z} e^{-x} p(x),$$

which leads to

$$p(x) = \frac{2Z}{\sqrt{2\pi}} e^{(x-x^2/2)}.$$

To choose the largest possible  $Z$ , we find that  $p_{\max} = \max_x p(x)$  is attained at  $x_* = 1$ . Setting  $p_{\max} = 1$  gives the well known result

$$Z = \sqrt{\frac{\pi}{2e}} \approx 58\%.$$

An efficiency of just over 50% makes this not the best way to generate standard normals. This formula for  $Z$  leads to

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

A C/C++ code fragment for this is

```
float const sqrte = sqrt(e);
float X;
...
while (1) {
    X = - ln( rng() );           \\ an exponential trial
    p = exp( X - X*X/2 )/sqrte; \\ the acceptance probability
    if ( rand() < p ) break; }  \\ leave the loop if accept.
if ( rng() < .5 ) X = -X       \\ Create an actual normal by ...
                               \\ flipping the sign half the time.
```

To make a million standard positive normals this way will take about two million if tests and about five million standard uniforms.

## 5 Testing distributions

We can dream of a world where every sampler would be coded correctly the first time. In the real world, we need ways to verify that samples  $X_k$  actually come from density  $f$ . Like many problems in Monte Carlo, this is a classical problem in statistics that has several solutions. One is through *density estimation*, which is the statistical problem of estimating a probability density,  $f(x)$ , from a collection of samples from  $f$ . To test a sampler, we generate many samples, estimate the density they came from, and compare this (by eye or using statistical tests) to the desired density. The Kolmogorov Smirnov statistic is a different statistical test for equality between distributions that does not use density estimation. We can use it to test whether our samples come from the desired distribution. It is generally more powerful than test based on density estimation but is may be less visual and is restricted to one dimension.

### 5.1 Density estimation.

In simple density estimation we have independent samples  $X_1, \dots, X_L$  of a random variable with density  $f$ . A density estimator is a function  $\hat{f}(x; X_1, \dots, X_L)$  (more properly, a family of functions depending on  $L$ ) that should be close to  $f(x)$  for large  $L$ . There are two simple methods used for density estimation, *histogram* and *kernel* methods. In both cases the user needs to supply a *length scale*,  $\Delta x$ , that should be so large that many sample points fall within an interval of length  $\Delta x$  but not so large that  $f(x)$  varies appreciable between  $x$  and  $x + \Delta x$ . For multivariate problems the criteria are that there should be many samples in a region of size  $\Delta x$  and that  $f(x) \approx f(x')$  when  $|x - x'|$  is of the order of  $\Delta x$ . Although density estimation is not restricted to one dimension, it is restricted to low dimensions, say,  $n < 4$ .

The choice of  $\Delta x$  is a tradeoff between statistical error and bias. The statistical error of  $\hat{f}(x; \dots)$  depends on  $\text{var}(\hat{f}(x; \dots))$ , which, in turn, is roughly the reciprocal of the number of  $X_k$  within a distance  $\Delta x$  of  $x$ . We reduce the statistical error by increasing  $\Delta x$ . The bias is  $E[\hat{f}(x; \dots)] - f(x)$ . It turns out that  $E[\hat{f}(x; \dots)]$  is roughly the average of  $f$  over a region of size  $\Delta x$  about  $x$ . The bias is small when  $f$  does not vary much in this region and is made smaller by reducing  $\Delta x$ . The optimal tradeoff between statistical error and bias depends on the density estimation method and  $f$ .

#### 5.1.1 The histogram method

We start with the histogram method for one dimensional data. For a simple one dimensional histogram, the *bin size* is  $\Delta x$ , the *bin centers* are  $x_j = j\Delta x$ , the *bins* are intervals  $B_j = (x_j - \Delta x/2, x_j + \Delta x/2)$ , and the *bin counts* are  $N_j = \#\{X_k \in B_j\}$ . The bin  $B_j$  is an interval of length  $\Delta x$  with  $x_j$  in the center. The bin count  $N_j$  is the number of data points that fall within  $B_j$ . It is traditional to plot the bin counts as a bar graph, which also is called a histogram.

The histogram density estimator is

$$\widehat{f}(x_j; \dots) = \frac{N_j}{\Delta x L}. \quad (13)$$

We can estimate  $f(x)$  for other  $x$  values either by interpolating (more accurate) or by having an estimator that is constant over each bin (graph is a bar graph). We find the bias of  $\widehat{f}(x_j)$  by evaluating its expected value. Using the bin *indicator function*  $\mathbf{1}_j(x) = 1$  if  $x \in B_j$  and  $\mathbf{1}_j(x) = 0$  if  $x \notin B_j$ , we have

$$N_j = \sum_{k=1}^L \mathbf{1}_j(X_k).$$

For each  $j$ , the random variables  $\mathbf{1}_j(X_k)$  are independent with expected value

$$E[\mathbf{1}_j(X_k)] = P(X_k \in B_j) = \int_{B_j} f(x) dx.$$

Therefore, the expected value of the density estimator literally is the average of the actual density over a region of size  $\Delta x$ :

$$E[\widehat{f}(x_j)] = \frac{1}{\Delta x} \int_{x_j - \Delta x/2}^{x_j + \Delta x/2} f(x) dx = f(x_j) + O(\Delta x^2),$$

if  $f$  is a smooth function of  $x$ . We used  $x_j$  as the center of bin  $B_j$  exactly to make the bias second order in the bin size. Had we taken the left endpoint ( $B_j = [x_j, x_{j+1}]$ ), for example, the bias would have been first order in  $\Delta x$ . Since linear interpolation also is second order accurate, getting  $\widehat{f}(x)$  by piecewise linear interpolation from (13) has bias of order  $\Delta x^2$  for all  $x$ . Warning: Probability densities that are not smooth functions of  $x$  occur in practical problems. For them the bias may well be much larger than  $O(\Delta x)^2$ .

The statistical error in  $\widehat{f}(x_j)$  is of the order of its standard deviation, the square root of its variance. Note that (13) is a sum of iid (independent, identically distributed) terms so the variance is  $L$  times the variance of a typical term. A typical term is  $\frac{1}{\Delta x L}$  times a *Bernoulli*<sup>7</sup> random variable with  $p_j = \int_{B_j} f(x) dx \approx \Delta x f(x_j)$ . The variance of this Bernoulli is  $p_j(1 - p_j)$ , but since  $p_j$  is small, we get essentially the same thing by ignoring the factor  $1 - p_j$ . Altogether,

$$\text{var}(\widehat{f}(x_j)) \approx \frac{f(x_j)}{L \Delta x},$$

so the statistical error is roughly

$$\frac{1}{\sqrt{L \Delta x}} \cdot \sqrt{f(x_j)}. \quad (14)$$

---

<sup>7</sup>A Bounoulli random variable is one that takes only the values 0 or 1, with  $p$  being the probability of 1.

As claimed above, this increases as  $\Delta x$  decreases. For future reference, note that the denominator in (14) is of the order of the number of data points in  $B_j$ , which we often call the number of *hits* in  $B_j$ .

We can choose  $\Delta x$  to roughly optimize the total error

$$\text{total error} = O(\Delta x^2) + O\left(\frac{1}{\sqrt{L\Delta x}}\right).$$

Doing the math (which happens to be the same as setting the two terms on the right equal to each other) gives

$$\Delta x_{\text{opt}} \sim L^{-1/5}, \quad \text{total error}_{\text{opt}} \sim L^{-2/5}.$$

Notice how slowly  $\Delta x_{\text{opt}}$  goes to zero with as  $L \rightarrow \infty$ . If we take  $\Delta x$  much smaller than this, the bias will be so much smaller than the statistical error that we would prefer to reduce the statistical error by increasing  $\Delta x$  because bias is negligibly smaller. Using this optimal  $\Delta x$  gives total error  $= O(L^{-4})$ , which is only slightly worse than the (soon to be) familiar  $O(L^{-5})$ , which is what you would get with no difficulties with small  $\Delta x$ .

In two dimensions the bins would be little squares instead of little intervals. The analysis is analogous and again calls for big bins. In  $n$  dimensions the optimal  $\Delta x$  is of order  $L^{1/(4+n)}$  and the corresponding error in  $\hat{f}$  is of order  $L^{1/(2+n/2)}$ . Clearly the histogram method cannot resolve variations in  $f$  on a scale smaller than  $\Delta x$ . Therefore its resolution for even moderate  $n$  is limited. The fancier kernel methods improve this just a bit, but not enough to make high dimensional density estimation practical.

### 5.1.2 Kernel density estimation

A *kernel function* is a function,  $\phi(x)$ , with  $\phi(x) \rightarrow 0$  (rapidly) as  $|x| \rightarrow \infty$ , and  $\int_{-\infty}^{\infty} \phi(x) dx = 1$ . Most kernel functions are symmetric ( $\phi(-x) = \phi(x)$ ) and continuous. Some have compact support ( $\phi(x) = 0$  if  $|x| > R$ , for some  $R$ ). Some are positive or non-negative. For  $\Delta x > 0$ , we define  $\phi_{\Delta x}(x) = \frac{1}{\Delta x} \phi(\frac{x}{\Delta x})$ . This scaled kernel function has “total mass” equal to one,  $\int_{-\infty}^{\infty} \phi_{\Delta x}(x) dx = 1$ , though it need not be positive. Small  $\Delta x$  concentrates the mass near  $x = 0$  and  $\phi_{\Delta x}$  approximates a delta function.

Kernel functions are used to *regularize*<sup>8</sup> non-smooth functions, which means to find a smooth function that approximates a rough one. If  $g(x)$  is a function, the regularized approximation is given by the convolution integral

$$g_{\Delta x}(x) = \int_{-\infty}^{\infty} \phi_{\Delta x}(x-y)g(y)dy = (\phi_{\Delta x} * g)(x).$$

Suppose, for example, that  $\int_{-\infty}^{\infty} |\phi'(x)| dx = B < L\infty$ . Then, if  $|g(x)| \leq C$  for all  $x$ , then (check this)  $|\partial_x g_{\Delta x}(x)| \leq BC/\Delta x$  for all  $x$ . This says that the

<sup>8</sup>Kernel functions often are called *mollifiers* because they mollify (make less angry) rough functions.

smoothed function  $g_{\Delta x}$  is differentiable even when  $g$  is only bounded (but with possible discontinuities). Moreover, it is an exercise in undergraduate analysis to show that  $g_{\Delta x}(x) \rightarrow g(x)$  if  $g$  is bounded and continuous at  $x$ .

The rate of convergence depends on the smoothness of  $g$  and the number of *moment conditions*  $\phi$  satisfies and its rate of decay. For example, suppose  $\int |x^2 \phi(x)| dx = D < \infty$  (a decay condition) and  $\int x \phi(x) dx = 0$  (a moment condition). Finally, suppose that the first term Taylor approximation of  $g$  at  $x$  has  $g(y) - g(x) = g'(x)(y - x) + R(y, x)$  with the error bound  $|R(y, x)| \leq C(y - x)^2$ . Then  $g_{\Delta x}(x) - g(x) = O(\Delta x^2)$ . To see this, we plug the Taylor series error bound into the convolution integral, using the fact that  $\int \phi_{\Delta x}(x - y)g(x)dy = g(x)$ ,

$$\begin{aligned} g_{\Delta x}(x) - g(x) &= \int_{-\infty}^{\infty} \phi_{\Delta x}(x - y) (g(y) - g(x)) dy \\ &= g'(x) \int_{-\infty}^{\infty} \phi_{\Delta x}(x - y) (y - x) dy \\ &\quad + \int_{-\infty}^{\infty} \phi_{\Delta x}(x - y) R(y, x) dy . \end{aligned}$$

The first term on the right is equal to zero because of the moment condition. To bound the second term, we first scale the moment bound:

$$\int_{-\infty}^{\infty} x^2 \phi_{\Delta x}(x) dx = D \Delta x^2 .$$

Then we change variables to get

$$\begin{aligned} |g_{\Delta x}(x) - g(x)| &= \left| \int_{-\infty}^{\infty} \phi_{\Delta x}(x - y) R(y, x) dy \right| \\ &\leq \int_{-\infty}^{\infty} \phi_{\Delta x}(y) C y^2 dy \\ &\leq DC \Delta x^2 , \end{aligned}$$

Note that the first order moment condition  $\int x \phi(x) dx = 0$  is satisfied whenever  $\phi$  is an even function of  $x$ . In particular, it is satisfied for the step function kernel function used in the histogram method  $\phi(x) = \mathbf{1}_{[-1/2, 1/2]}(x)$ , and the second order accuracy here is the same as the second order accuracy there.

It is possible to construct smooth kernel functions that satisfy more moment conditions. For example,

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \left( e^{-x^2/2} - \frac{1}{2} \partial_x^2 e^{-x^2/2} \right) = \frac{1}{\sqrt{2\pi}} \left( \frac{3}{2} - \frac{1}{2} x^2 \right) e^{-x^2/2} \quad (15)$$

satisfies the second order moment condition  $\int \phi(x) x^2 dx = 0$  and the third order condition (which is free with symmetry)  $\int \phi(x) x^3 dx = 0$ , so  $g_{\Delta x}(x) - g(x) = O(\Delta x^4)$  if  $g$  is five times differentiable. No positive kernel function can satisfy the second order moment condition.

Given samples  $X_1, \dots, X_L$ , the *empirical density* is

$$f_e(x; X_1, \dots, X_L) = \frac{1}{L} \sum_{j=1}^L \delta(x - X_j) .$$

This has the natural property that for any  $V(x)$ , the estimator of  $E_f[V(X)]$  is

$$\frac{1}{L} \sum_{j=1}^L V(X_j) = \int V(x) f_e(x) dx .$$

The kernel function density estimate is the kernel function smoothed empirical density:

$$\hat{f}_{\Delta x}(x; \dots) = (\phi_{\Delta x} * f_e)(x) = \frac{1}{L} \sum_{j=1}^L \phi_{\Delta x}(x - X_j) .$$

As in the discussion of the histogram method, the bias is

$$E \left[ \hat{f}_{\Delta x}(x; \dots) - f(x) \right] = f_{\Delta x}(x) - f(x) .$$

The size of the bias depends on the smoothness of  $f(x)$  and the number of moment conditions satisfied by  $\phi$ . For example, the higher order kernel (15) gives bias of order  $\Delta x^4$ . It is easy to see that the standard deviation is of order  $\frac{1}{L\Delta x}$  as before. Optimizing over  $\Delta x$  gives  $\Delta x_{\text{opt}} = O(L^{-1/9})$  and error of order  $L^{-4/9}$ . The high order kernel increased the power of  $L$  from  $2/5 = .4$  to  $4/9 = .444\dots$ . It is not clear that using a smoothing length as large as  $\Delta x = L^{1/9}$  makes any practical sense.

## 6 Gaussians

Gaussian random variables are important for many reasons. The central limit theorem implies that many random variables have a normal or approximately normal distribution. Gaussian models in statistical physics often are called “free fields”. Besides these situations where Gaussians are the desired end product, they often are used as tools in other Monte Carlo methods because they are easy to generate.

### 6.1 The standard normal

The *standard normal* is a scalar Gaussian with mean zero and variance one. Its density is  $f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ . Its distribution function often is called  $N(x) = P(X \leq x)$ . There is good shareware<sup>9</sup> that computes the inverse function  $N^{-1}(u)$  which can be used to generate a standard normal.

For those too busy to find this software, there is the elegant *Box Muller* method:

<sup>9</sup>As for most simple special functions, there are strikingly simple rational functions that approximate  $N(x)$  and  $N^{-1}(u)$  to nearly full double precision accuracy.



```

R = \sqrt{-2*\log( \rng() );  \ \ the radial component
T = 2*pi*rng();              \ \ the angle
X = R*cos(T);                \ \ a standard normal
Y = R*sin(T);                \ \ an independent standard normal.

```

This turns two independent standard uniforms into two independent standard normals. I don't have time to give the derivation, but it is related to the the trick used to show that  $\int_{-\infty}^{\infty} e^{-x^2/2} = \sqrt{2\pi}$ .

## 6.2 One component normals

We write  $\mathcal{N}(\mu, \sigma^2)$  for the distribution of a one component Gaussian with mean  $\mu$  and variance  $\sigma^2$ . The density of an  $\mathcal{N}(\mu, \sigma^2)$  random variable is  $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2}$ . If  $Z \sim \mathcal{N}(0, 1)$  is a standard normal, then  $X = \sigma Z + \mu \sim \mathcal{N}(\mu, \sigma^2)$ .

## 6.3 Mult-component normals

In general it is hard to find efficient simple samplers for multi-component random variables with many components. The multivariate normal is a fortunate exception. Let  $H$  be an  $n \times n$  symmetric positive definite matrix. Consider the probability density function for an  $n$  component random variable  $X$ :

$$f(x) = \frac{1}{Z} e^{-x^t H x / 2} . \quad (16)$$

Here  $x^t$  is the transpose of  $x$ , which is a row vector. It is common practice, particularly in statistical mechanics, to write unknown normalization constants in the form  $1/Z$ . Here, as we soon will see,

$$Z = \int_{x \in R^n} e^{-x^t H x / 2} dx = \sqrt{\frac{(2\pi)^n}{\det(H)}} . \quad (17)$$

A random variable with density (16) is a non-degenerate multivariate normal with mean zero.

A simple example is Gaussian random walk. Let  $X(t)$  be a standard Brownian motion with  $X(0) = 0$  and  $E[X(t)^2] = t$ . Choose a time step  $\Delta t$ , define times  $t_k = k\Delta t$ , and let  $X$  be the multivariate normal with components  $X_k = X(t_k)$  (some abuse of notation here). Let  $g(y, x, s)$  be the time  $s$  *transition probability* density for standard Brownian motion, which is the probability density for  $X(t+s)$  given that  $X(t) = y$ . It is given by the *heat kernel*  $g(y, x, s) = \frac{1}{\sqrt{2\pi s}} e^{-(x-y)^2/2s}$ . The joint density for the components  $X_k$  is a product of these transition densities:

$$\begin{aligned} f(x_1, \dots, x_n) &= \prod_{k=0}^{n-1} g(x_{k-1}, x_k, \Delta t) \\ &= \frac{1}{(2\pi\Delta t)^{n/2}} \exp\left(\frac{-1}{2\Delta t} \sum_{k=0}^{n-1} (x_{k+1} - x_k)^2\right) . \end{aligned}$$

Note that the right side of this formula involves  $x_0$  which is equal to zero. This simplifies the notation. Comparing this to (16), we see that

$$x^t H x = \frac{1}{\Delta t} \left( x_1^2 + \sum_{k=1}^{n-1} (x_{k+1} - x_k)^2 \right). \quad (18)$$

The reader should check that the matrix  $H$  takes the form

$$H = \frac{1}{\Delta t} \begin{pmatrix} 2 & -1 & 0 & & \cdots & & 0 \\ -1 & 2 & -1 & 0 & & \cdots & 0 \\ & 0 & -1 & 2 & \ddots & & \\ & & 0 & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \vdots \\ \vdots & & & & \ddots & 2 & -1 & 0 \\ & & & & & -1 & 2 & -1 \\ 0 & & & \cdots & & 0 & -1 & 1 \end{pmatrix}$$

The right side of (18) shows that the matrix  $H$  is positive definite. Clearly, none of the terms is negative. Moreover, if  $x^t H x = 0$ , then  $x_1 = 0$ . But if  $x_1 = 0$  then the  $k = 1$  term in the sum is  $(x_2 - x_1)^2 = x_2^2$ , which implies that  $x_2 = 0$  also. Eventually, we see that  $x_k = 0$  for all  $k$ .

We write  $C$  for the covariance matrix of a multivariate normal. If  $EX = 0$ , the  $(j, k)$  entry of  $C$  is  $C_{jk} = \text{cov}(X_j, X_k) = E[X_j X_k]$ . This is expressed succinctly as  $C = E[XX^t]$ . Note that  $X$  is a column vector and  $X^t$  is a row vector, so  $XX^t$  is an  $n \times n$  matrix. Although  $XX^t$  has rank one (and therefore is not positive definite) for any particular  $X$ , the expected value is the average of many rank one matrices so it can have full rank. The relationship between  $C$  and (16) is  $C = H^{-1}$ , as we now show. This implies that the multivariate normal density is determined by the covariance matrix, just as the one dimensional normal is determined by the variance (if the mean is zero).

We verify the integral formula (17) and calculate  $E[XX^t]$  using the Choleski factorization<sup>10</sup>  $H = MM^t$ , where  $M$  is a lower triangular nonsingular matrix. We use  $M$  to change variables in the integral<sup>11</sup> (17):  $M^t x = y$ , so  $x = M^{-t} y$  and<sup>12</sup>

$$dx = \det(M^{-t}) dy = \frac{1}{\sqrt{\det(H)}}$$

the last equality because  $\det(H) = \det(MM^t) = \det(M)\det(M^t) = \det(M)^2$ . Therefore  $x^t H x = (xM^t)^t (M^t x) = y^t y$ , and

$$\int_{R^n} e^{-x^t H x / 2} dx = \frac{1}{\sqrt{\det(H)}} \int_{R^n} e^{-y^t y / 2} dy = \frac{(2\pi)^{n/2}}{\sqrt{\det(H)}}.$$

<sup>10</sup>See most books on numerical analysis, scientific computing, multivariate statistics, or linear algebra.

<sup>11</sup>The inverse of  $M^t$  is the same as the transpose of  $M^{-1}$ , so both are denoted as  $M^{-t}$ .

<sup>12</sup>This is the Jacobian factor in the change of variables formula for multi-dimensional integrals.

In this, we used  $\int_{R^n} e^{-y^t y/2} dy = (\sqrt{2\pi})^n$ . This is true because  $y^t y = \sum_k y_k^2$ , so

$$\begin{aligned} \int_{R^n} e^{-y^t y/2} dy &= \int e^{-(y_1^2/2 + \dots + y_n^2/2)} dy_1 \dots dy_n \\ &= \left( \int_{-\infty}^{\infty} e^{-y_1^2/2} dy_1 \right)^n . \end{aligned}$$

The same change of variable helps us evaluate  $C = E[XX^t]$ . Note that

$$\frac{1}{(2\pi)^{n/2}} \int_{R^n} yy^t e^{-y^t y/2} dy = I ,$$

because the integral represents the covariance matrix of  $n$  independent standard normal random variables. With this, we have

$$\begin{aligned} C &= E[XX^t] \\ &= \frac{\det(M)}{(2\pi)^{n/2}} \int_{R^n} xx^t e^{-x^t Hx/2} dx \\ &= \frac{1}{(2\pi)^{n/2}} \int_{R^n} (M^{-t}y) (y^t M^{-1}) e^{-y^t y/2} dy \\ &= M^{-t} \left( \frac{1}{(2\pi)^{n/2}} \int_{R^n} yy^t e^{-y^t y/2} dy \right) M^{-1} \\ &= M^{-t} I M^{-1} = (MM^t)^{-1} = H^{-1} , \end{aligned}$$

as claimed.

The Choleski factorization can be used to sample a multivariate normal in two ways, depending on whether  $H$  or  $C$  is given. If  $C$  is given we compute it's Choleski factorization  $C = LL^t$ . Then we let  $Z \in R^n$  be the  $n$  component multivariate normal whose entries are independent standard normals. Then  $X = LZ$  has covariance

$$E[XX^t] = E[(LZ)(Z^t L^t)] = L E[ZZ^t] L^t = LL^t = C ,$$

as desired. Note that the Choleski factorizations of  $C$  and  $H$  are related by  $L = M^{-t}$ , since  $(MM^t)^{-1} = M^{-t}M^{-1}$ , and the inverse of a lower triangular matrix is lower triangular. Therefore, if we have  $H$ , we can find  $M$  and take  $X = M^{-t}Z$ , which is the same as solving for  $X$  in the equation  $M^t X = Z$ . This can be done by forward substitution. Of course, we can compute  $C$  from  $H$  or the other way around. The reason to use  $H$  and  $M$  instead of  $C$  and  $L$  is that  $H$  and  $M$  might be sparse, as in the Brownian random walk example above. This happens in many example from the physical sciences but less often in statistics.