7.2 Transformation Method: Exponential and Normal Deviates

In the previous section, we learned how to generate random deviates with a uniform probability distribution, so that the probability of generating a number between x and $x + dx$, denoted $p(x)dx$, is given by

$$p(x)dx = \begin{cases} dx & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.2.1}$$

The probability distribution $p(x)$ is of course normalized, so that

$$\int_{-\infty}^{\infty} p(x)dx = 1 \tag{7.2.2}$$

Now suppose that we generate a uniform deviate x and then take some prescribed function of it, $y(x)$. The probability distribution of y, denoted $p(y)dy$, is determined by the fundamental transformation law of probabilities, which is simply

$$|p(y)dy| = |p(x)dx| \tag{7.2.3}$$

or

$$p(y) = p(x)\left|\frac{dx}{dy}\right| \tag{7.2.4}$$

Exponential Deviates

As an example, suppose that $y(x) \equiv -\ln(x)$, and that $p(x)$ is as given by equation (7.2.1) for a uniform deviate. Then

$$p(y)dy = \left|\frac{dx}{dy}\right| dy = e^{-y}dy \tag{7.2.5}$$

which is distributed exponentially. This exponential distribution occurs frequently in real problems, usually as the distribution of waiting times between independent Poisson-random events, for example the radioactive decay of nuclei. You can also easily see (from 7.2.4) that the quantity y/λ has the probability distribution $\lambda e^{-\lambda y}$.

So we have

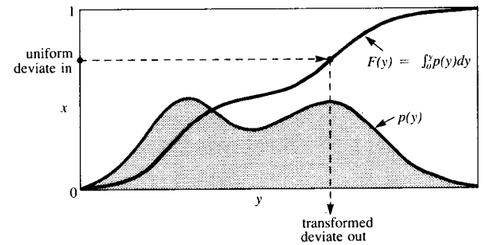

Figure 7.2.1. Transformation method for generating a random deviate y from a known probability distribution $p(y)$. The indefinite integral of $p(y)$ must be known and invertible. A uniform deviate x is chosen between 0 and 1. Its corresponding y on the definite-integral curve is the desired deviate.

```
FUNCTION EXPDEV(IDUM)
    Returns an exponentially distributed, positive, random deviate of unit mean, using
    RAN1(IDUM) as the source of uniform deviates.
EXPDEV=-LOG(RAN1(IDUM))
RETURN
END
```

Let's see what is involved in using the above *transformation method* to generate some arbitrary desired distribution of y's, say one with $p(y) = f(y)$ for some positive function f whose integral is 1. (See Figure 7.2.1.) According to (7.2.4), we need to solve the differential equation

$$\frac{dx}{dy} = f(y) \tag{7.2.6}$$

But the solution of this is just $x = F(y)$, where $F(y)$ is the indefinite integral of $f(y)$. The desired transformation which takes a uniform deviate into one distributed as $f(y)$ is therefore

$$y(x) = F^{-1}(x) \tag{7.2.7}$$

where F^{-1} is the inverse function to F. Whether (7.2.7) is feasible to implement depends on whether the *inverse function of the integral of f(y)* is itself feasible to compute, either analytically or numerically. Sometimes it is, and sometimes it isn't.

Incidentally, (7.2.7) has an immediate geometric interpretation: Since $F(y)$ is the area under the probability curve to the left of y, (7.2.7) is just the prescription: choose a uniform random x, then find the value y that has that fraction x of probability area to its left, and return the value y.

Normal (Gaussian) Deviates

Transformation methods generalize to more than one dimension. If $x_1, x_2, \ldots$ are random deviates with a *joint* probability distribution $p(x_1, x_2, \ldots)\, dx_1 dx_2 \ldots$, and if $y_1, y_2, \ldots$ are each functions of all the x's (same number of y's as x's), then the joint probability distribution of the y's is

$$p(y_1, y_2, \ldots) dy_1 dy_2 \ldots = p(x_1, x_2, \ldots) \left| \frac{\partial(x_1, x_2, \ldots)}{\partial(y_1, y_2, \ldots)} \right| dy_1 dy_2 \ldots \tag{7.2.8}$$

where $|\partial(\ )/\partial(\ )|$ is the Jacobian determinant of the x's with respect to the y's (or reciprocal of the Jacobian determinant of the y's with respect to the x's).

An important example of the use of (7.2.8) is the *Box-Muller* method for generating random deviates with a normal (Gaussian) distribution,

$$p(y)dy = \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy \tag{7.2.9}$$

Consider the transformation between two uniform deviates on (0,1), x_1, x_2, and two quantities y_1, y_2,

$$\begin{aligned} y_1 &= \sqrt{-2 \ln x_1} \cos 2\pi x_2 \\ y_2 &= \sqrt{-2 \ln x_1} \sin 2\pi x_2 \end{aligned} \tag{7.2.10}$$

Equivalently we can write

$$\begin{aligned} x_1 &= \exp\left[-\frac{1}{2}(y_1^2 + y_2^2) \right] \\ x_2 &= \frac{1}{2\pi} \arctan \frac{y_2}{y_1} \end{aligned} \tag{7.2.11}$$

Now the Jacobian determinant can readily be calculated (try it!):

$$\frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix} = -\left[\frac{1}{\sqrt{2\pi}} e^{-y_1^2/2} \right] \left[\frac{1}{\sqrt{2\pi}} e^{-y_2^2/2} \right] \tag{7.2.12}$$

Since this is the product of a function of y_2 alone and a function of y_1 alone, we see that each y is independently distributed according to the normal distribution (7.2.9).

One further trick is useful in applying (7.2.10). Suppose that, instead of picking uniform deviates x_1 and x_2 in the unit square, we instead pick v_1 and v_2 as the ordinate and abscissa of a random point inside the unit circle around the origin. Then the sum of their squares, $R \equiv v_1^2 + v_2^2$ is a uniform deviate, which can be used for x_1, while the angle that (v_1, v_2) defines with respect to the v_1 axis can serve as the random angle $2\pi x_2$. What's the advantage? It's that the cosine and sine in (7.2.10) can now be written as $v_1/\sqrt{R}$ and $v_2/\sqrt{R}$, obviating the trigonometric function calls!

We thus have

```
      FUNCTION GASDEV(IDUM)
        Returns a normally distributed deviate with zero mean and unit variance, using RAN1(IDUM)
        as the source of uniform deviates.
      DATA ISET/0/
      IF (ISET.EQ.0) THEN                 We don't have an extra deviate handy, so
    1   V1=2.*RAN1(IDUM)-1.               pick two uniform numbers in the square extending from -1 to +1 in
          V2=2.*RAN1(IDUM)-1.                 each direction,
          R=V1**2+V2**2                   see if they are in the unit circle,
          IF(R.GE.1.)GO TO 1              and if they are not, try again.
          FAC=SQRT(-2.*LOG(R)/R)          Now make the Box-Muller transformation
          GSET=V1*FAC                     to get two normal deviates. Return one and save the other for next
          GASDEV=V2*FAC                       time.
          ISET=1                          Set flag.
      ELSE                                We have an extra deviate handy,
          GASDEV=GSET                     so return it,
          ISET=0                          and unset the flag.
      ENDIF
      RETURN
      END
```

REFERENCES AND FURTHER READING:

Knuth, Donald E. 1981, *Seminumerical Algorithms*, 2nd ed., vol. 2 of *The Art of Computer Programming* (Reading, Mass.: Addison-Wesley), pp. 116ff.

7.3 Rejection Method: Gamma, Poisson, Binomial Deviates

The *rejection method* is a powerful, general technique for generating random deviates whose distribution function $p(x)dx$ (probability of a value occurring between x and $x + dx$) is known and computable. The rejection method does *not* require that the cumulative distribution function [indefinite integral of $p(x)$] be readily computible, much less the inverse of that function — which was required for the transformation method in the previous section.

The rejection method is based on a simple geometrical argument:

Draw a graph of the probability distribution $p(x)$ that you wish to generate, so that the area under the curve in any range of x corresponds to the desired probability of generating an x in that range. If we had some way of choosing a random point *in two dimensions*, with uniform probability in the

area under your curve, then the x value of that random point would have the desired distribution.

Now, on the same graph, draw any other curve $f(x)$ which has finite (not infinite) area and lies everywhere *above* your original probability distribution. (This is always possible, because your original curve encloses only unit area, by definition of probability.) We will call this $f(x)$ the *comparison function*. Imagine now that you have some way of choosing a random point in two dimensions that is uniform in the area under the comparison function. Whenever that point lies outside the area under the original probability distribution, we will *reject* it and choose another random point. Whenever it lies inside the area under the original probability distribution, we will *accept* it. It should be obvious that the accepted points are uniform in the accepted area, so that their x values have the desired distribution. It should also be obvious that the fraction of points rejected just depends on the ratio of the area of the comparison function to the area of the probability distribution function, not on the details of shape of either function. For example, a comparison function whose area is less than 2 will reject fewer than half the points, even if it approximates the probability function very badly at some values of x, e.g. remains finite in some region where x is zero.

It remains only to suggest how to choose a uniform random point in two dimensions under the comparison function $f(x)$. A variant of the transformation method (§7.2) does nicely: Be sure to have chosen a comparison function whose indefinite integral is known analytically, and is also analytically invertible to give x as a function of "area under the comparison function to the left of x." Now pick a uniform deviate between 0 and A, where A is the total area under $f(x)$, and use it to get a corresponding x. Then pick a uniform deviate between 0 and $f(x)$ as the y value for the two-dimensional point. You should be able to convince yourself that the point (x, y) is uniformly distributed in the area under the comparison function $f(x)$.

An equivalent procedure is to pick the second uniform deviate between zero and one, and accept or reject according to whether it is respectively less than or greater than the ratio $p(x)/f(x)$.

So, to summarize, the rejection method for some given $p(x)$ requires that one find, once and for all, some reasonably good comparison function $f(x)$. Thereafter, each deviate generated requires two uniform random deviates, one evaluation of f (to get the coordinate y), and one evaluation of p (to decide whether to accept or reject the point x, y). Figure 7.3.1 illustrates the procedure. Then, of course, this procedure must be repeated, on the average, A times before the final deviate is obtained.

Gamma Distribution

The gamma distribution of integer order $a > 0$ is the waiting time to the a^{th} event in a Poisson random process of unit mean. For example, when $a = 1$, it is just the exponential distribution of §7.2, the waiting time to the first event.

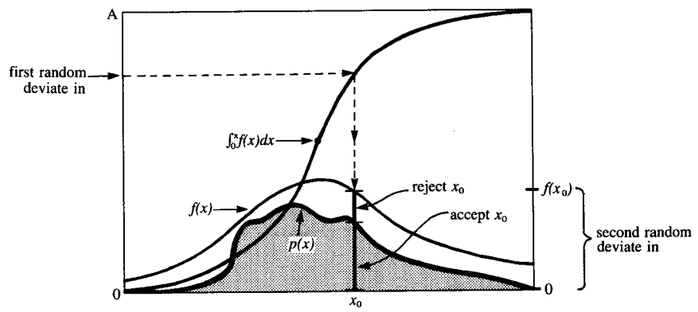

Figure 7.3.1. Rejection method for generating a random deviate x from a known probability distribution $p(x)$ that is everywhere less than some other function $f(x)$. The transformation method is first used to generate a random deviate x of the distribution f (compare Figure 7.2.1). A second uniform deviate is used to decide whether to accept or reject that x. If it is rejected, a new deviate of f is found; and so on. The ratio of accepted to rejected points is the ratio of the area under p to the area between p and f.

A gamma deviate has probability $p_a(x)dx$ of occurring with a value between x and $x + dx$, where

$$p_a(x)dx = \frac{x^{a-1}e^{-x}}{\Gamma(a)}dx \qquad x > 0 \tag{7.3.1}$$

To generate deviates of (7.3.1) for small values of a, it is best to add up a exponentially distributed waiting times, i.e. logarithms of uniform deviates. Since the sum of logarithms is the logarithm of the product, one really has only to generate the product of a uniform deviates, then take the log.

For larger values of a, the distribution (7.3.1) has a typically "bell-shaped" form, with a peak at $x = a$ and a half-width of about $\sqrt{a}$.

We will be interested in several probability distributions with this same qualitative form. A useful comparison function in such cases is derived from the *Lorentzian distribution*

$$p(y)dy = \frac{1}{\pi}\left(\frac{1}{1+y^2}\right)dy \tag{7.3.2}$$

whose inverse indefinite integral is just the tangent function. It follows that the x-coordinate of an area-uniform random point under the comparison function

$$f(x) = \frac{c_0}{1 + (x - x_0)^2/a_0^2} \tag{7.3.3}$$

for any constants a_0, c_0, and x_0, can be generated by the prescription

$$x = a_0 \tan(\pi U) + x_0 \tag{7.3.4}$$

where U is a uniform deviate between 0 and 1. Thus, for some specific "bell-shaped" $p(x)$ probability distribution, we need only find constants a_0, c_0, x_0, with the product $a_0 c_0$ (which determines the area) as small as possible, such that (7.3.3) is everywhere greater than $p(x)$.

Ahrens has done this for the gamma distribution, yielding the following algorithm (as described in Knuth):

```
      FUNCTION GAMDEV(IA,IDUM)
          Returns a deviate distributed as a gamma distribution of integer order IA, i.e. a waiting
          time to the IAth event in a Poisson process of unit mean, using RAN1(IDUM) as the source
          of uniform deviates.
      IF(IA.LT.1)PAUSE
      IF(IA.LT.6)THEN                  Use direct method, adding waiting times.
          X=1.
          DO 11 J=1,IA
              X=X*RAN1(IDUM)
          11 CONTINUE
          X=-LOG(X)
      ELSE                             Use rejection method.
    1     V1=2.*RAN1(IDUM)-1.          These four lines generate the tangent of a random angle, i.e. are
              V2=2.*RAN1(IDUM)-1.          equivalent to Y = TAN(3.14159265 * RAN1(IDUM)).
          IF(V1**2+V2**2.GT.1.)GO TO 1
              Y=V2/V1
              AM=IA-1
              S=SQRT(2.*AM+1.)
              X=S*Y+AM                 We decide whether to reject X:
          IF(X.LE.0.)GO TO 1           Reject in region of zero probability.
              E=(1.+Y**2)*EXP(AM*LOG(X/AM)-S*Y)    Ratio of probability fn. to comparison fn.
          IF(RAN1(IDUM).GT.E)GO TO 1   Reject on basis of a second uniform deviate.
      ENDIF
      GAMDEV=X
      RETURN
      END
```

Poisson Deviates

The Poisson distribution is conceptually related to the gamma distribution. It gives the probability of a certain integer number m of unit rate Poisson random events occurring in a given interval of time x, while the gamma distribution was the probability of waiting time between x and $x + dx$ to the m^{th} event. Note that m takes on only integer values ≥ 0, so that the Poisson distribution, viewed as a continuous distribution function $p_x(m)dm$, is zero everywhere except where m is an integer ≥ 0. At such places, it is infinite, such that the integrated probability over a region containing the integer is some finite number. The total probability at an integer j is

$$\text{Prob}(j) = \int_{j-\epsilon}^{j+\epsilon} p_x(m)dm = \frac{x^j e^{-x}}{j!} \tag{7.3.5}$$

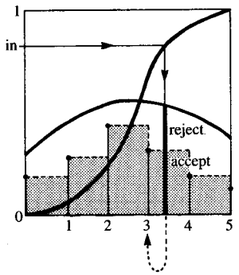

Figure 7.3.2. Rejection method as applied to an integer-valued distribution. The method is performed on the step function shown as a dashed line, yielding a real-valued deviate. This deviate is rounded down to the next lower integer, which is output.

At first sight this might seem an unlikely candidate distribution for the rejection method, since no continuous comparison function can be larger than the infinitely tall, but infinitely narrow, *Dirac delta functions* in $p_x(m)$. However there is a trick that we can do: Spread the finite area in the spike at j uniformly into the interval between j and $j + 1$. This defines a continuous distribution $q_x(m)dm$ given by

$$q_x(m)dm = \frac{x^{[m]} e^{-x}}{[m]!} \tag{7.3.6}$$

where $[m]$ represents the largest integer less than m. If we now use the rejection method to generate a (non-integer) deviate from (7.3.6), and then take the integer part of that deviate, it will be as if drawn from the desired distribution (7.3.5). (See Figure 7.3.2.) This trick is general for any integer-valued probability distribution.

For x large enough, the distribution (7.3.6) is qualitatively bell-shaped (albeit with a bell made out of small, square steps), and we can use the same kind of Lorentzian comparison function as was already used above. For small x, we can generate independent exponential deviates (waiting times between events); when the sum of these first exceeds x, then the number of events which would have occurred in waiting time x becomes known and is one less than the number of terms in the sum.

These ideas produce the following routine:

```
      FUNCTION POIDEV(XM,IDUM)
          Returns as a floating-point number an integer value that is a random deviate drawn from a
          Poisson distribution of mean XM, using RAN1(IDUM) as a source of uniform random deviates.
      PARAMETER (PI=3.141592654)
      DATA OLDM /-1./                  Flag for whether XM has changed since last call.
      IF (XM.LT.12.)THEN               Use direct method.
          IF (XM.NE.OLDM) THEN
              OLDM=XM
              G=EXP(-XM)               If XM is new, compute the exponential.
          ENDIF
          EM=-1
```

```
      T=1.
2     EM=EM+1.                      Instead of adding exponential deviates it is equivalent to multiply
      T=T*RAN1(IDUM)                    uniform deviates.  Then we never actually have to take the
      IF (T.GT.G) GO TO 2               log, merely compare to the pre-computed exponential.
    ELSE                            Use rejection method.
      IF (XM.NE.OLDM) THEN          If XM has changed since the last call, then precompute some functions
        OLDM=XM                         which occur below.
        SQ=SQRT(2.*XM)
        ALXM=ALOG(XM)
        G=XM*ALXM-GAMMLN(XM+1.)       The function GAMMLN is the natural log of the gamma func-
      ENDIF                               tion, as given in §6.2.
1     Y=TAN(PI*RAN1(IDUM))          Y is a deviate from a Lorentzian comparison function.
      EM=SQ*Y+XM                    EM is Y, shifted and scaled.
      IF (EM.LT.0.)  GO TO 1        Reject if in regime of zero probability.
      EM=INT(EM)                    The trick for integer-valued distributions.
      T=0.9*(1.+Y**2)*EXP(EM*ALXM-GAMMLN(EM+1.)-G)     The ratio of the desired distribution to
      IF (RAN1(IDUM).GT.T) GO TO 1      the comparison function; we accept or reject by comparing it
    ENDIF                               to another uniform deviate.  The factor 0.9 is chosen so that
    POIDEV=EM                           T never exceeds 1.
    RETURN
    END
```

Binomial Deviates

If an event occurs with probability q, and we make n trials, then the number of times m that it occurs has the binomial distribution,

$$\int_{j-\epsilon}^{j+\epsilon} p_{n,q}(m)dm = \binom{n}{j} q^j (1-q)^{n-j} \tag{7.3.7}$$

The binomial distribution is integer valued, with m taking on possible values from 0 to n. It depends on *two* parameters, n and q, so is correspondingly a bit harder to implement than our previous examples. Nevertheless, the techniques already illustrated are sufficiently powerful to do the job:

```
FUNCTION BNLDEV(PP,N,IDUM)
    Returns as a floating-point number an integer value that is a random deviate drawn from
    a binomial distribution of N trials each of probability PP, using RAN1(IDUM) as a source
    of uniform random deviates.
PARAMETER (PI=3.141592654)
DATA NOLD /-1/, POLD /-1./    Arguments from previous calls.
IF(PP.LE.0.5)THEN             The binomial distribution is invariant under changing PP to 1.-PP, if
    P=PP                          we also change the answer to N minus itself; we'll remember
ELSE                              to do this below.
    P=1.-PP
ENDIF
AM=N*P                        This is the mean of the deviate to be produced.
IF (N.LT.25)THEN              Use the direct method while N is not too large.  This can require up
    BNLDEV=0.                     to 25 calls to RAN1.
    DO 11 J=1,N
        IF(RAN1(IDUM).LT.P)BNLDEV=BNLDEV+1.
    11 CONTINUE
ELSE IF (AM.LT.1.)  THEN      If fewer than one event is expected out of 25 or more trials, then the
    G=EXP(-AM)                    distribution is quite accurately Poisson.  Use direct Poisson
    T=1.                          method.
```



```
    DO 12 J=0,N
        T=T*RAN1(IDUM)
        IF (T.LT.G) GO TO 1
    12 CONTINUE
    J=N
1   BNLDEV=J
  ELSE                            Use the rejection method.
    IF (N.NE.NOLD) THEN           If N has changed, then compute useful quantities.
      EN=N
      OLDG=GAMMLN(EN+1.)
      NOLD=N
    ENDIF
    IF (P.NE.POLD) THEN           If P has changed, then compute useful quantities.
      PC=1.-P
      PLOG=LOG(P)
      PCLOG=LOG(PC)
      POLD=P
    ENDIF
    SQ=SQRT(2.*AM*PC)             The following code should by now seem familiar: rejection method
2   Y=TAN(PI*RAN1(IDUM))              with a Lorentzian comparison function.
    EM=SQ*Y+AM
    IF (EM.LT.0..OR.EM.GE.EN+1.)  GO TO 2        Reject.
    EM=INT(EM)                    Trick for integer-valued distribution.
    T=1.2*SQ*(1.+Y**2)*EXP(OLDG-GAMMLN(EM+1.)
*     -GAMMLN(EN-EM+1.)+EM*PLOG+(EN-EM)*PCLOG)
    IF (RAN1(IDUM).GT.T) GO TO 2      Reject.  This happens about 1.5 times per deviate, on av-
    BNLDEV=EM                         erage.
  ENDIF
  IF (P.NE.PP) BNLDEV=N-BNLDEV    Remember to undo the symmetry transformation.
  RETURN
  END
```

REFERENCES AND FURTHER READING:

Knuth, Donald E. 1981, *Seminumerical Algorithms*, 2nd ed., vol. 2 of *The Art of Computer Programming* (Reading, Mass.: Addison-Wesley), pp. 120ff.

7.4 Generation of Random Bits

This topic is not very useful for programming in high-level languages, but it can be quite useful when you have access to the machine-language level of a machine or when you are in a position to build special-purpose hardware out of readily available chips.

The problem is how to generate single random bits, with 0 and 1 equally probable. Of course you can just generate uniform random deviates between zero and one and use their first bit (i.e. test if they are greater than or less than 0.5). However this takes a lot of arithmetic; there are special purpose applications, such as real-time signal processing, where you want to generate bits very much faster than that.

One method for generating random bits, with two variant implementations, is based on the theory of "primitive polynomials modulo 2." It is