Chapter 3

Basic Monte Carlo Methods

Simulation and Monte Carlo Methods

Consider as an example the following very simple problem. We wish to price a European call option with exercise price \$22 and payoff function $V(S_T) = (S_T - 22)^+$. Assume for the present that the interest rate is 0% and S_T can take only the following five values with corresponding risk neutral (Q) probabilities

s	20	21	22	23	24
$Q[S_T = s]$	1/16	4/16	6/16	4/16	1/16

In this case, since the distribution is very simple, we can price the call option explicitly;

$$E_Q V(S_T) = E_Q(S_T - 22)^+ = (23 - 22)\frac{4}{16} + (24 - 22)\frac{1}{16} = \frac{3}{8}.$$

However, the ability to value an option explicitly is a rare luxury. An alternative would be to generate a large number (say $n = 1000$) independent simulations of the stock price S_T under the measure Q and average the returns from the option. Say the simulations yielded values for S_T of $22, 20, 23, 21, 22, 23, 20, 24, ...$ then

the estimated value of the option is

$$\begin{aligned}\overline{V(S_T)} &= \frac{1}{1000}[(22-22)^+ + (20-22)^+ + (23-22)^+ + ...]. \\ &= \frac{1}{1000}[0+0+1+....]\end{aligned}$$

The law of large numbers assures us for a large number of simulations n, the average $\overline{V(S_T)}$ will approximate the true expectation $E_Q V(S_T)$. Now while it would be foolish to use simulation in a simple problem like this, there are many models in which it is much easier to randomly generate values of the process S_T than it is to establish its exact distribution. In such a case, simulation is the method of choice.

Randomly generating a value of S_T for the discrete distribution above is easy, provided that we can produce independent random uniform random numbers on a computer. For example, if we were able to generate a random number Y_i which has a uniform distribution on the integers $\{0, 1, 2, ..., 15\}$ then we could define S_T for the i'th simulation as follows:

If Y_i is in set	$\{0\}$	$\{1,2,3,4\}$	$\{5,6,7,8,9,10\}$	$\{11,12,13,14\}$	$\{15\}$
define $S_T =$	20	21	22	23	24

Of course, to get a reasonably accurate estimate of the price of a complex derivative may well require a large number of simulations, but this is decreasingly a problem with increasingly fast computer processors. The first ingredient in a simulation is a stream of uniform random numbers Y_i used above. In practice all other distributions are generated by processing discrete uniform random numbers. Their generation is discussed in the next section.

Uniform Random Number Generation

The first requirement of a stochastic model is the ability to generate "random" variables or something resembling them. Early such generators attached to computers exploited physical phenomena such as the least significant digits in

an accurate measure of time, or the amount of background cosmic radiation as the basis for such a generator, but these suffer from a number of disadvantages. They may well be "random" in some more general sense than are the pseudo-random number generators that are presently used but their properties are difficult to establish, and the sequences are impossible to reproduce. The ability to reproduce a sequence of random numbers is important for debugging a simulation program and for reducing its variance.

It is quite remarkable that some very simple recursion formulae define sequences that behave like sequences of independent random numbers and *appear* to more or less obey the major laws of probability such as the law of large numbers, the central limit theorem, the Glivenko-Cantelli theorem, etc. Although computer generated *pseudo random numbers* have become more and more like independent random variables as the knowledge of these generators grows, the main limit theorems in probability such as the law of large numbers and the central limit theorem still do not have versions which directly apply to dependent sequences such as those output by a random number generator. The fact that certain pseudo-random sequences appear to share the properties of independent sequences is still a matter of observation rather than proof, indicating that many results in probability hold under much more general circumstances than the relatively restrictive conditions under which these theorems have so far been proven. One would intuitively expect an enormous difference between the behaviour of independent random variables X_n and a deterministic (i.e. non-random) sequence satisfying a recursion of the form $x_n = g(x_{n-1})$ for a simple function g. Surprisingly, for many carefully selected such functions g it is quite difficult to determine the difference between such a sequence and an independent sequence. Of course, numbers generated from a simple recursion such as this are neither random, nor are x_{n-1} and x_n independent. We sometimes draw attention to this by referring to such a sequence as *pseudo-random numbers.* While they are in no case independent, we will nevertheless attempt to find

simple functions g which provide behaviour *similar* to that of independent uniform random numbers. The search for a satisfactory random number generator is largely a search for a suitable function g, possibly depending on more than one of the earlier terms of the sequence, which imitates in many different respects the behaviour of independent observations with a specified distribution.

Definition: reduction modulo m. For positive integers x and m, the value $a \bmod m$ is the remainder (between 0 and $m-1$) obtained when a is divided by m. So for example $7 \bmod 3 = 1$ since $7 = 2 \times 3 + 1$.

The single most common class of random number generators are of the form

$$x_n := (ax_{n-1} + c) \mod m$$

for given integers $a, c,$ and m which we select in advance. This generator is initiated with a "seed" x_0 and then run to produce a whole sequence of values. When $c = 0$, these generators are referred to as *multiplicative congruential generators* and in general as *mixed or linear congruential generators*. The "seed", x_0, is usually updated by the generator with each call to it. There are two common choices of m, either m prime or $m = 2^k$ for some k (usually 31 for 32 bit machines).

Example: Mixed Congruential generator

Define $x_n = (5x_{n-1} + 3) \bmod 8$ and the seed $x_0 = 3$. Note that by this recursion

$$x_1 = (5 \times 3 + 3) \bmod 8 = 18 \bmod 8 = 2$$

$$x_2 = 13 \bmod 8 = 5$$

$$x_3 = 28 \bmod 8 = 4$$

and $x_4, x_5, x_{6,} x_7, x_8 = 7, 6, 1, 0, 3$ respectively

and after this point (for $n > 8$) the recursion will simply repeat again the pattern already established, $3, 2, 5, 4, 7, 6, 1, 0, 3, 2, 5, 4, ...$

The above repetition is inevitable for a linear congruential generator. There are at most m possible numbers after reduction mod m and once we arrive back at the seed the sequence is destined to repeat itself. In the example above, the sequence cycles after 8 numbers. The length of one cycle, before the sequence begins to repeat itself again, is called the *period* of the generator. For a mixed generator, the period must be less than or equal to m. For multiplicative generators, the period is shorter, and often considerably shorter.

Multiplicative Generators.

For multiplicative generators, $c = 0$. Consider for example the generator $x_n = 5x_{n-1} \; mod \; 8$ and $x_0 = 3$. This produces the sequence $3, 7, 3, 7, ...$ In this case, the period is only 2, but for general m, it is clear that the maximum possible period is $m-1$ because it generates values in the set $\{1, ..., m-1\}$. The generator cannot generate the value 0 because if it did, all subsequent values generated are identically 0. Therefore the maximum possible period corresponds to a cycle through non-zero integers exactly once. But in the example above with $m = 2^k$, the period is far from attaining its theoretical maximum, $m - 1$. The following Theorem shows that the period of a multiplicative generator is maximal when m is a prime number and a satisfies some conditions.

Theorem 14 *(period of multiplicative generator).*
If m is prime, the multiplicative congruential generator $x_n = ax_{n-1}(mod\, m)$, $a \neq 0$, has maximal period $m - 1$ if and only if $a^i \neq 1(mod\ m)$ for all $i = 1, 2, ..., m - 1$.

If m is a prime, and if the condition $a^{m-1} = 1(mod\ m)$ and $a^i \neq 1(mod\ m)$ for all $i < m - 1$ holds, we say that a is a *primitive root* of m, which means

that the powers of a generate all of the possible elements of the multiplicative group of integers mod m. Consider the multiplicative congruential generator $x_n = 2x_{n-1} \bmod 11$. It is easy to check that $2^i \bmod 11 = 2, 4, 8, 5, 10, 9, 7, 3, 6, 1$ as $i = 1, 2, ...10$. Since the value $i = m-1$ is the first for which $2^i \bmod 11 = 1$, 2 is a primitive root of 11 and this is a maximal period generator having period 10. When $m = 11$, only the values $a = 2, 6, 7, 8$ are primitive roots and produce full period (10) generators.

One of the more common moduli on 32 bit machines is the Mersenne prime $m = 2^{31} - 1$. In this case, the following values of a (among many others) all produce full period generators:

$$a = 7, 16807, 39373, 48271, 69621, 630360016, 742938285, 950706376,$$

$$1226874159, 62089911, 1343714438$$

Let us suppose now that m is prime and a_2 is the multiplicative inverse (mod m) of a_1 by which we mean $(a_1 a_2) \bmod m = 1$. When m is prime, the set of integers $\{0, 1, 2, ..., m-1\}$ together with the operations of addition and multiplication $\bmod m$ forms what is called a *finite field.* This is a finite set of elements together with operations of addition and multiplication such as those we enjoy in the real number system. For example for integers $x_1, a_1, a_2 \in \{0, 1, 2, ..., m-1\}$, the product of a_1 and x_1 can be defined as $(a_1 x_1) \bmod m = x_2$, say. Just as non-zero numbers in the real number system have multiplicative inverses, so too do non=zero elements of this field. Suppose for example a_2 is the muultiplicative inverse of a_1 so that $a_2 a_1 \bmod m = 1$. If we now multiply x_2 by a_2 we have

$$(a_2 x_2) \bmod m = (a_2 a_1 x_1) \bmod m = (a_2 a_1 \bmod m)(x_1 \bmod m) = x_1.$$

This shows that $x_1 = (a_2 x_2) \bmod m$ is equivalent to $x_2 = (a_1 x_1) \bmod m$. In other words, using a_2 the multiplicative inverse of $a_1 \bmod m$, the multiplicative generator with multiplier a_2 generates exactly the same sequence as that with

multiplier a_1 except in reverse order. Of course if a is a primitive root of m, then so is its multiplicative inverse.

Theorem 15 *(Period of Multiplicative Generators with $m = 2^k$)*
If $m = 2^k$ with $k \geq 3$, and if $a \bmod 8 = 3$ or 5 and x_0 is odd, then the multiplicative congruential generator has maximal period $= 2^{k-2}$.

For the proof of these results, see Ripley(1987), Chapter 2. The following simple Matlab code allows us to compare linear congruential generators with small values of m. It generates a total of n such values for user defined a, c, m, x_0 =seed. The efficient implementation of a generator for large values of m depends very much on the architecture of the computer. We normally choose m to be close to the machine precision (e.g. 2^{32} for a 32 bit machine.

```
function x=lcg(x0,a,c,m,n)
y=x0;      x=x0;
for i=1:n   ;      y=rem(a*y+c,m);      x=[x y];      end
```

The period of a linear congruential generator varies both with the multiplier a and the constant c. For example consider the generator

$$x_n = (ax_{n-1} + 1) \bmod 2^{10}$$

for various multipliers a. When we use an even multiplier such as $a = 2, 4, ...$(using seed 1) we end up with a sequence that eventually locks into a specific value. For example with $a = 8$ we obtain the sequence 1,9,73,585,585,....never changing beyond that point. The periods for odd multipliers are listed below (all started with seed 1)

a	1	3	5	7	9	11	13	15	17	19	21	23	25
Period	1024	512	1024	256	1024	512	1024	128	1024	512	1024	256	1024

The astute reader will notice that the only full-period multipliers a are those which are multipliers of 4. This is a special case of the following theorem.

Theorem 16 *(Period of Mixed or Linear Congruential Generators.)*
The Mixed Congruential Generator,

$$x_n = (ax_{n-1} + c) \bmod m \tag{3.1}$$

has full period m *if and only if*
(i) c *and* m *are relatively prime.*
(ii) Each prime factor of m *is also a factor of* $a - 1$.
(iii) If 4 *divides* m *it also divides* $a - 1$.

When m is prime, (ii) together with the assumption that $a < m$ implies that m must divide $a - 1$ which implies $a = 1$. So for prime m the only full-period generators correspond to $a = 1$. Prime numbers m are desirable for long periods in the case of multiplicative generators, but in the case of mixed congruential generators, only the trivial one $x_n = (x_{n-1} + c)(mod\ m)$ has maximal period m when m is prime. This covers the popular Mersenne prime $m = 2^{31} - 1$..

For the generators $x_n = (ax_{n-1} + c) \bmod 2^k$ where $m = 2^k, k \geq 2$, the condition for full period 2^k requires that c is odd, and $a = 4j + 1$ for some integer j.

Some of the linear or multiplicative generators which have been suggested are the following:

m	a	c	
$2^{31}-1$	$7^5=16807$	0	Lewis,Goodman, Miller (1969)IBM,
$2^{31}-1$	630360016	0	Fishman (Simscript II)
$2^{31}-1$	742938285	0	Fishman and Moore
2^{31}	65539	0	RANDU
2^{32}	69069	1	Super-Duper (Marsaglia)
2^{32}	3934873077	0	Fishman and Moore
2^{32}	3141592653	1	DERIVE
2^{32}	663608941	0	Ahrens (C-RAND)
2^{32}	134775813	1	Turbo-Pascal,Version 7(period= 2^{32})
2^{35}	5^{13}	0	APPLE
$10^{12}-11$	427419669081	0	MAPLE
2^{59}	13^{13}	0	NAG
$2^{61}-1$	$2^{20}-2^{19}$	0	Wu (1997)

Table 3.1: Some Suggested Linear and Multiplicative Random Number Generators

Other Random Number Generators.

A generalization of the linear congruential generators which use a k-dimensional vectors X has been considered, specifically when we wish to generate correlation among the components of X. Suppose the components of X are to be integers between 0 and $m-1$ where m is a power of a prime number. If A is an arbitrary $k \times k$ matrix with integral elements also in the range $\{0, 1, ..., m-1\}$ then we begin with a vector-valued seed X_0, a constant vector C and define recursively

$$X_n := (AX_{n-1} + C) \bmod m$$

Such generators are more common when C is the zero vector and called *matrix multiplicative congruential generators*. A related idea is to use a higher order

recursion like

$$x_n = (a_1 x_{n-1} + a_2 x_{n-2} + .. + a_k x_{n-k}) \bmod m, \tag{3.2}$$

called a *multiple recursive generator*. L'Ecuyer (1996,1999) combines a number of such generators in order to achieve a period around $2^{319}$and good uniformity properties. When a recursion such as (3.2) with $m = 2$ is used to generate pseudo-random bits $\{0, 1\}$, and these bits are then mapped into uniform (0,1) numbers, it is called a *Tausworthe* or *Feedback Shift Register* generators. The coefficients a_i are determined from *primitive polynomials* over the *Galois Field*.

In some cases, the uniform random number generator in proprietary packages such as *Splus* and *Matlab* are not completely described in the package documentation. This is a further recommendation of the transparency of packages like **R**. Evidently in *Splus*, the multiplicative congruential generator is used, and then the sequence is " shuffled" using a Shift-register generator (a special case of the matrix congruential generator described above). This secondary processing of the sequence can increase the period but it is not always clear what other effects it has. In general, shuffling is conducted according to the following steps

1. Generate a sequence of pseudo-random numbers x_i using $x_{i+1} = a_1 x_i (mod\, m_1)$

.

2. For fixed k put $(T_1, \ldots, T_k) = (x_1, \ldots, x_k)$.

3. Generate, using a different generator, a sequence $y_{i+1} = a_2 y_i (\bmod m_2)$.

4. Output the random number T_I where $I = \lceil Y_i k/m_2 \rceil$.

5. Increment i, replace T_I by the next value of x, and return to step 3.

One generator is used to produce the sequence x, numbers needed to fill k holes. The other generator is then used select which hole to draw the next number from or to "shuffle" the x sequence.

Example: A shuffled generator

Consider a generator described by the above steps with $k = 4, x_{n+1} = (5x_n)(\mathrm{mod}_1\, 19)$ and $y_{n+1} = (5y_n)(\mathrm{mod}\ 29)$

$x_n =$ 3 15 18 14 13 8 2

$y_n =$ 3 15 17 27 19 8 11

We start by filling four pigeon-holes with the numbers produced by the first generator so that $(T_1, \ldots, T_4) = (3, 15, 18, 14)$. Then use the second generator to select a random index I telling us which pigeon-hole to draw the next number from. Since these holes are numbered from 1 through 4, we use $I = \lceil 4\times 3/29 \rceil = 1$. Then the first number in our random sequence is drawn from box 1, i.e. $z_1 = T_1 = 3$, so $z_1 = 3$. This element T_1 is now replaced by 13, the next number in the x sequence. Proceeding in this way, the next index is $I = \lceil 4\times 15/29 \rceil = 3$ and so the next number drawn is $z_2 = T_3 = 18$. Of course, when we have finished generating the values $z_1, z_2, ...$ all of which lie between 1 and $m_1 = 18$, we will usually transform them in the usual way (e.g. z_i/m_1) to produce something approximating continuous uniform random numbers on [0,1]. When m_1, is large, it is reasonable to expect the values z_i/m_1 to be approximately continuous and uniform on the interval $[0, 1]$. One advantage of shuffling is that the period of the generator is usually greatly extended. Whereas the original x sequence had period 9 in this example, the shuffled generator has a larger period or around 126.

There is another approach, summing pseudo-random numbers, which is also used to extend the period of a generator. This is based on the following theorem (see L'Ecuyer (1988)). For further discussion of the effect of taking linear combinations of the output from two or more random number generators, see Fishman (1995, Section 7.13).

Theorem 17 *(Summing mod m)*

If X is random variable uniform on the integers $\{0, \ldots, m-1\}$ and if Y is

any integer-valued random variable independent of X, then the random variable $W = (X + Y)(mod\ m)$ is uniform on the integers $\{0, \ldots, m-1\}$.

Theorem 18 *(Period of generator summed mod m_1)*
If $x_{i+1} = a_1 x_i \bmod m_1$ has period $m_1 - 1$ and $y_{i+1} = a_2 y_i \bmod m_2$ has period $m_2 - 1$, then $(x_i + y_i) (\bmod m_1)$ has period the least common multiple of $(m_1 - 1, m_2 - 1)$.

Example: summing two generators

If $x_{i+1} = 16807 x_i mod(2^{31} - 1)$ and $y_{i+1} = 40692 y_i mod(2^{31} - 249)$, then the period of $(x_i + y_i) mod(2^{31} - 1)$ is

$$\frac{(2^{31} - 2)(2^{31} - 250)}{2 \times 31} \approx 7.4 \times 10^{16}$$

This is much greater than the period of either of the two constituent generators.

Other generators.

One such generator, the "Mersenne-Twister", from Matsumoto and Nishimura (1998) has been implemented in **R** and has a period of $2^{19937} - 1$. Others use a non-linear function g in the recursion $x_{n+1} = g(x_n) (\bmod\ m)$ to replace a linear one For example we might define $x_{n+1} = x_n^2 (\bmod m)$ (called a *quadratic residue generator*) or $x_{n+1} = g(x_n) \bmod m$ for a quadratic function or some other non-linear function g. Typically the function g is designed to result in large values and thus more or less random low order bits. I*nversive congruential generators* generate x_{n+1} using the $(\bmod\ m)$ inverse of x_n.

Other generators which have been implemented in R include: the *Wichmann-Hill* (1982,1984) generator which uses three multiplicative generators with prime moduli 30269, 30307, 30323 and has a period of $\frac{1}{4}(30268 \times 30306 \times 30322)$. The outputs from these three generators are converted to $[0, 1]$ and then summed mod 1. This is similar to the idea of Theorem 17, but the addition takes

place after the output is converted to [0,1]. See *Applied Statistics* (1984), **33**, 123. Also implemented are *Marsaglia's Multicarry* generator which has a period of more than 2^{60} and reportedly passed all tests (according to Marsaglia), Marsaglia's "Super-Duper", a linear congruential generator listed in Table 1, and two generators developed by Knuth (1997,2002) the *Knuth-TAOCP* and *Knuth-TAOCP-2002*.

Conversion to Uniform $(0,1)$ generators:

In general, random integers should be mapped into the unit interval in such a way that the values 0 and 1, each of which have probability 0 for a continuous distribution are avoided. For a multiplicative generator, since values lie between 1 and $m-1$, we may divide the random number by m. For a linear congruential generator taking possible values $x \in \{0, 1, ..., m-1\}$, it is suggested that we use $(x + 0.5)/m$.

Apparent Randomness of Pseudo-Random Number Generators

Knowing whether a sequence behaves in all respects like independent uniform random variables is, for the statistician, pretty close to knowing the meaning of life. At the very least, in order that one of the above generators be reasonable approximations to independent uniform variates it should satisfy a number of statistical tests. Suppose we reduce the uniform numbers on $\{0, 1, ..., m-1\}$ to values approximately uniformly distributed on the unit interval $[0, 1]$ as described above either by dividing through by m or using $(x + 0.5)/m$. There are many tests that can be applied to determine whether the hypothesis of independent uniform variates is credible (not, of course, whether the hypothesis is *true*. We know by the nature of all of these pseudo-random number generators

that it is not!).

Runs Test

We wish to test the hypothesis H_0 that a sequence$\{U_i, i = 1, 2, ..., n\}$ consists of n independent identically distributed random variables under the assumption that they have a continuous distribution. The *runs test* measures runs, either in the original sequence or in its differences. For example, suppose we denote a positive difference between consecutive elements of the sequence by $+$ and a negative difference by $-$. Then we may regard a sequence of the form $.21, .24, .34, .37, .41, .49, .56, .51, .21, .25, .28, .56, .92, .96$ as unlikely under independence because the corresponding differences $++++++--++++++$ have too few "runs" (the number of runs here is $R = 3$). Under the assumption that the sequence $\{U_i, i = 1, 2, ..., n\}$ is independent and continuous, it is possible to show that $E(R) = \frac{2n-1}{3}$and $var(R) = \frac{3n-5}{18}$. The proof of this result is a problem at the end of this chapter. We may also approximate the distribution of R with the normal distribution for $n \geq 25$. A test at a 0.1% level of significance is therefore: reject the hypothesis of independence if

$$\left| \frac{R - \frac{2n-1}{3}}{\sqrt{\frac{3n-5}{18}}} \right| > 3.29,$$

where 3.29 is the corresponding $N(0, 1)$ quantile. A more powerful test based on runs compares the lengths of the runs of various lengths (in this case one run up of length 7, one run down of length 3, and one run up of length 6) with their theoretical distribution.

Another test of independence is the *serial correlation test*. The runs test above is one way of checking that the pairs $(U_n, U_{n+1})$are approximately uniformly distributed on the unit square. This could obviously be generalized to pairs like (U_i, U_{i+j}). One could also use the sample correlation or covariance as

the basis for such a test. For example, for $j \geq 0$,

$$C_j = \frac{1}{n}(U_1U_{1+j} + U_2U_{2+j} + ..U_{n-j}U_n + U_{n+1-j}U_1 + ... + U_nU_j) \qquad (3.3)$$

The test may be based on the normal approximation to the distribution of C_j with mean $E(C_0) = 1/3$ and $E(C_j) = 1/4$ for $j \geq 1$. Also

$$var(C_j) = \begin{cases} \frac{4}{45n} & \text{for} \quad j = 0 \\ \frac{13}{144n} & \text{for} \quad j \geq 1,\, j \neq \frac{n}{2} \\ \frac{7}{72n} & \text{for} \quad j = \frac{n}{2} \end{cases}$$

Such a test, again at a 0.1% level will take the form: reject the hypothesis of independent uniform if

$$\left| \frac{C_j - \frac{1}{4}}{\sqrt{\frac{13}{144n}}} \right| > 3.29.$$

for a particular preselected value of j (usually chosen to be small, such as $j = 1, ...10$).

Chi-squared test.

The chi-squared test can be applied to the sequence in any dimension, for example $k = 2$. Suppose we have used a generator to produce a sequence of uniform$(0, 1)$ variables, $U_j, j = 1, 2, ...2n$, and then, for a partition $\{A_i; i = 1, ..., K\}$ of the unit square, we count N_i, the number of pairs of the form $(U_{2j-1}, U_{2j}) \in A_i$. See for example the points plotted in Figure 3.1. Clearly this should be related to the area or probability $P(A_i)$ of the set A_i. Pearson's chi-squared statistic is

$$\chi^2 = \sum_{i=1}^{K} \frac{[N_i - nP(A_i)]^2}{nP(A_i)} \qquad (3.4)$$

which should be compared with a chi-squared distribution with degrees of freedom $K-1$ or one less than the number of sets in the partition. Observed values

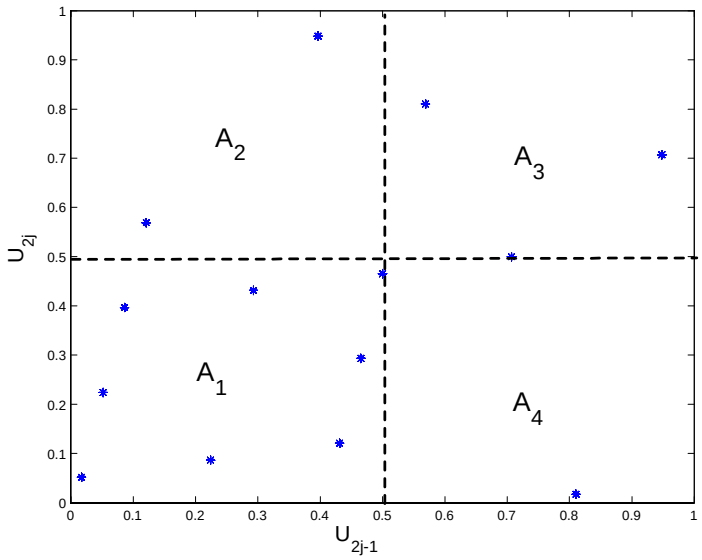

Figure 3.1: The Chi-squared Test

of the statistic that are unusually large for this distribution should lead to rejection of the uniformity hypothesis. The partition usually consists of squares of identical area but could, in general, be of arbitrary shape.

Spectral Test

Consecutive values plotted as pairs (x_n, x_{n+1}), when generated from a multiplicative congruential generator $x_{n+1} = ax_n (\mathrm{mod}\, m)$ fall on a *lattice*. A lattice is a set of points of the form $t_1 e_1 + t_2 e_2$ where t_1, t_2 range over all integers and e_1, e_2 are vectors, (here two dimensional vectors since we are viewing these points in pairs of consecutive values (x_n, x_{n+1})) called the "basis" for the lattice. A given lattice, however, has many possible different bases, and in order to analyze the lattice structure, we need to isolate the most "natural" basis, e.g. the one that we tend to see in viewing a lattice in two dimensions. Consider, for example, the lattice formed by the generator $x_n = 23x_{n-1} \bmod 97$. A plot of adjacent pairs

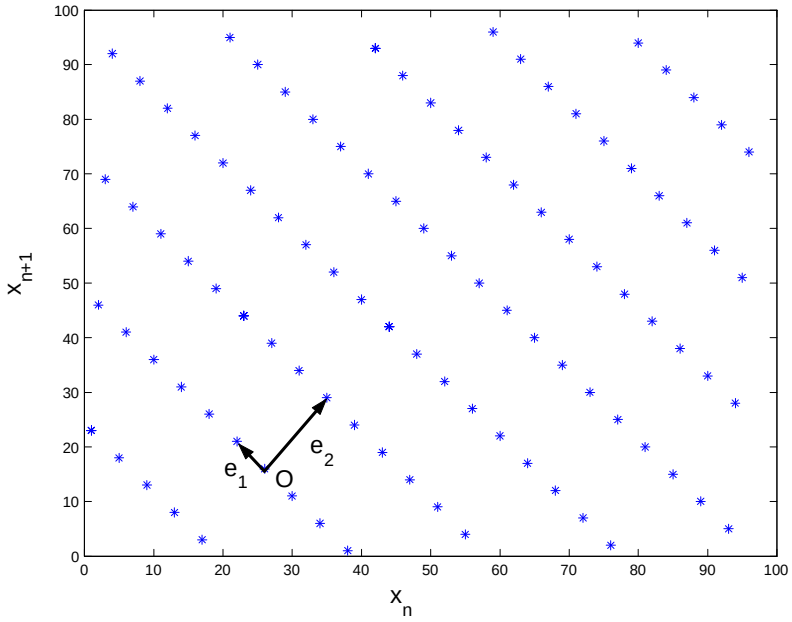

Figure 3.2: The Spectral Test

(x_n, x_{n+1}) is given in Figure 3.2. For basis vectors we could use $e_1 = (1, 23)$ and $e_2 = (4, -6)$, or we could replace e_1 by $(5, 18)$or $(9, 13)$ etc. Beginning at an arbitrary point O on the lattice as origin (in this case, since the original point (0,0) is on the lattice, we will leave it unchanged), we choose an unambiguous definition of e_1 to be the *shortest* vector in the lattice, and then define e_2 as the shortest vector in the lattice which is not of the form te_1 for integer t. Such a basis will be called a *natural basis*. The best generators are those for which the cells in the lattice generated by the 2 basis vectors e_1, e_2 or the parallelograms with sides parallel to e_1, e_2 are as close as possible to squares so that e_1 and e_2 are approximately the same length. As we change the multiplier a in such a way that the random number generator still has period $\simeq m$, there are roughly m points in a region above with area approximately m^2 and so the area of a parallelogram with sides e_1 and e_2 is approximately a constant (m) whatever

the multiplier a. In other words a longer e_1 is associated with a shorter vector e_2 and therefore for an ideal generator, the two vectors of reasonably similar length. A poor generator corresponds to a basis with e_2 much longer than e_1. The *spectral test statistic* ν is the renormalized length of the first basis vector $||e_1||$. The extension to a lattice in k-dimensions is done similarly. All linear congruential random number generators result in points which when plotted as consecutive k-tuples lie on a lattice. In general, for k consecutive points, the spectral test statistic is equal to $min(b_1^2 + b_2^2 + . + . + . + b_k^2)^{1/2}$ under the constraint $b_1 + b_2 a + ...b_k a^{k-1} = mq, q \neq 0$. Large values of the statistic indicate that the generator is adequate and Knuth suggests as a minimum threshold the value $\pi^{-1/2}[(k/2)!m/10]^{1/k}$.

One of the generators that fails the spectral test most spectacularly with $k = 3$ is the generator RANDU, $x_{n+1} = 65539\ x_n (\text{mod}\ 2^{31})$. This was used commonly in simulations until the 1980's and is now notorious for the fact that a small number of hyperplanes fit through all of the points (see Marsaglia, 1968). For RANDU, successive triplets tend to remain on the plane $x_n = 6x_{n-1} - 9x_{n-2}$. This may be seen by rotating the 3-dimensional graph of the sequence of triplets of the form $\{(x_{n-2}, x_{n-1}, x_n); n = 2, 3, 4, ...N\}$ as in Figure 3.3

As another example, in Figure 3.4 we plot 5000 consecutive triplets from a linear congruential random number generator with $a = 383, c = 263, m = 10,000$.

Linear planes are evident from some angles in this view, but not from others. In many problems, particularly ones in which random numbers are processed in groups of three or more, this phenomenon can lead to highly misleading results. The spectral test is the most widely used test which attempts to insure against lattice structure. tABLE 3.2 below is taken from Fishman(1996) and gives some values of the spectral test statistic for some linear congruential random number

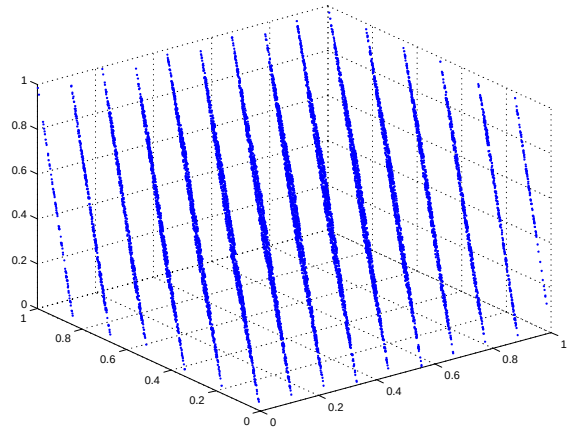

Figure 3.3: Lattice Structure of Uniform Random Numbers generated from RANDU

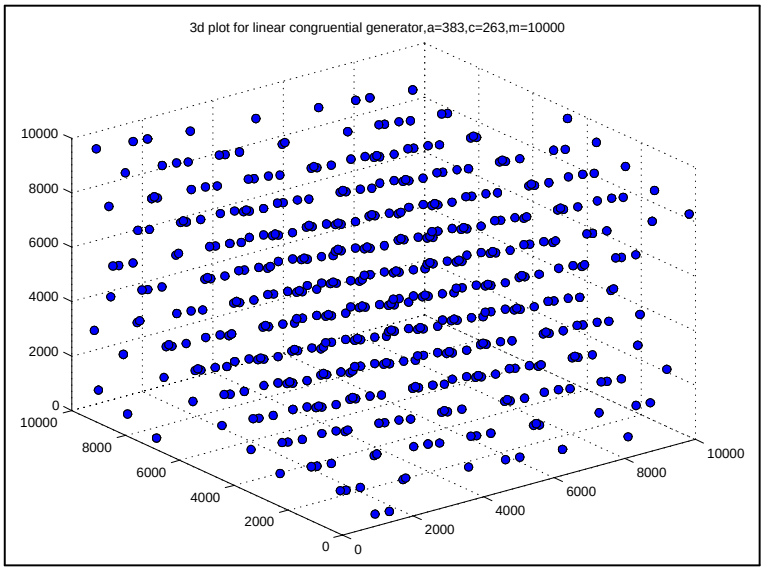

Figure 3.4: The values (x_i, x_{i+1}, x_{i+2}) generated by a linear congruential generator $x_{n+1} = (383x_n + 263) (\text{mod } 10000)$

generators in dimension $k \leq 7$.

m	a	c	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
$2^{31}-1$	7^5	0	0.34	0.44	0.58	0.74	0.65	0.57
$2^{31}-1$	630360016	0	0.82	0.43	0.78	0.80	0.57	0.68
$2^{31}-1$	742938285	0	0.87	0.86	0.86	0.83	0.83	0.62
2^{31}	65539	0	0.93	0.01	0.06	0.16	0.29	0.45
2^{32}	69069	0	0.46	0.31	0.46	0.55	0.38	0.50
2^{32}	3934873077	0	0.87	0.83	0.83	0.84	0.82	0.72
2^{32}	663608941	0	0.88	0.60	0.80	0.64	0.68	0.61
2^{35}	5^{13}	0	0.47	0.37	0.64	0.61	0.74	0.68
2^{59}	13^{13}	0	0.84	0.73	0.74	0.58	0.64	0.52

TABLE 3.2. Selected Spectral Test Statistics

The unacceptably small values for RANDU in the case $k = 3$ and $k = 4$ are highlighted. On the basis of these values of the spectral test, the multiplicative generators

$$x_{n+1} = 742938285 x_n (\text{mod } 2^{31} - 1)$$

$$x_{n+1} = 3934873077 x_n (\text{mod } 2^{32})$$

seem to be recommended since their test statistics are all reasonably large for $k = 2, ..., 7$.

Generating Random Numbers from Non-Uniform Continuous Distributions

By far the simplest and most common method for generating non-uniform variates is based on the inverse cumulative distribution function. For arbitrary

cumulative distribution function $F(x)$, define $F^{-1}(y) = min\{x; F(x) \geq y\}$. This defines a pseudo-inverse function which is a real inverse (i.e. $F(F^{-1}(y)) = F^{-1}(F(y)) = y$) only in the case that the cumulative distribution function is continuous and strictly increasing. However, in the general case of a possibly discontinuous non-decreasing cumulative distribution function the function continues to enjoy some of the properties of an inverse. Notice that $F^{-1}(F(x)) \leq x$ and $F(F^{-1}(y)) \geq y$ but $F^{-1}(F(F^{-1}(y))) = F^{-1}(y)$ and $F(F^{-1}(F(x))) = F(x)$. In the general case, when this pseudo-inverse is easily obtained, we may use the following to generate a random variable with cumulative distribution function $F(x)$.

Theorem 19 *(inverse transform) If F is an arbitrary cumulative distribution function and U is uniform$[0,1]$ then $X = F^{-1}(U)$ has cumulative distribution function $F(x)$.*

Proof. The proof is a simple consequence of the fact that

$$[U < F(x)] \subset [X \leq x] \subset [U \leq F(x)] \text{ for all } x, \tag{3.5}$$

evident from Figure 3.5. Taking probabilities throughout (3.5), and using the continuity of the distribution of U so that $P[U = F(x)] = 0$, we obtain

$$F(x) \leq P[X \leq x] \leq F(x).$$

■

Examples of Inverse Transform

Exponential (θ)

This distribution, a special case of the gamma distributions, is common in most applications of probability. For example in risk management, it is common to model the time between defaults on a contract as exponential (so the default

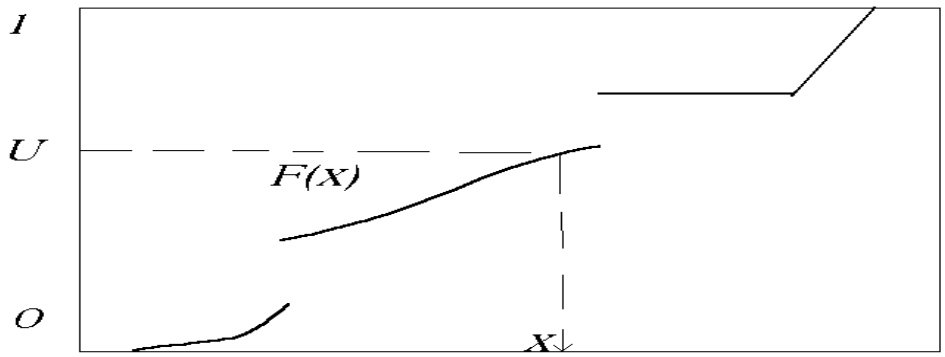

Figure 3.5: The Inverse Transform generator

times follow a Poisson process). In this case the probability density function is $f(x) = \frac{1}{\theta}e^{-x/\theta}, x \geq 0$ and $f(x) = 0$ for $x < 0$. The cumulative distribution function is $F(x) = 1 - e^{-x/\theta}, x \geq 0$. Then taking its inverse,

$$X = -\theta \ln(1 - U) \text{ or equivalently}$$

$$X = -\theta \ln U \text{ since } U \text{ and } 1 - U \text{ have the same distribution.}$$

In *Matlab*, the exponential random number generators is called *exprnd* and in *Splus* or *R* it is *rexp.*

Cauchy (a, b)

This distribution is a member of the *stable family* of distributions which we discuss later. It is similar to the normal only substantially more peaked in the center and with more area in the extreme tails of the distribution. The probability density function is

$$f(x) = \frac{b}{\pi(b^2 + (x - a)^2)}, -\infty < x < \infty.$$

See the comparison of the probability density functions in Figure 3.6. Here we have chosen the second (scale) parameter b for the Cauchy so that the two

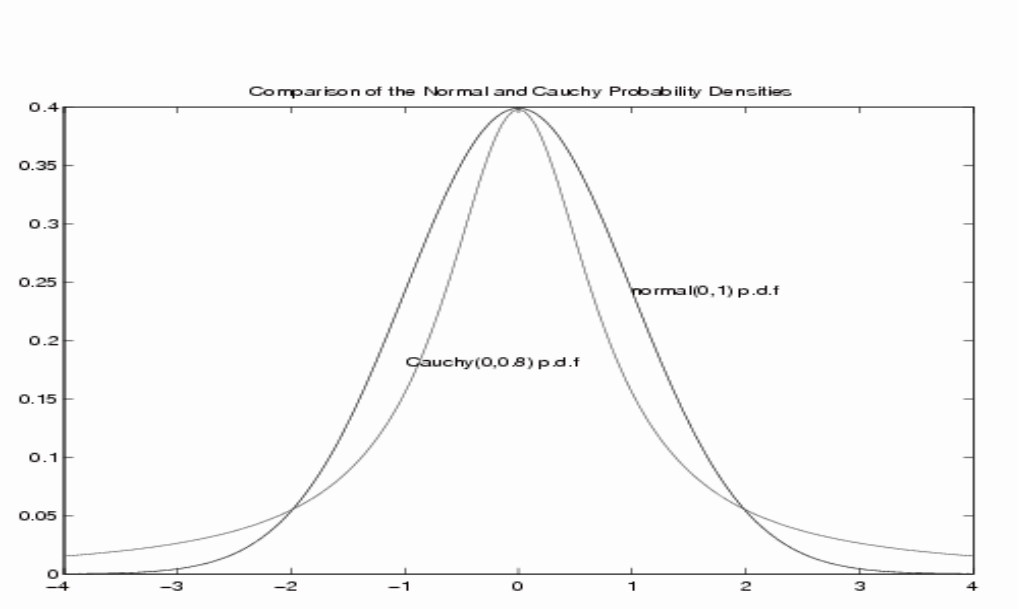

Figure 3.6: The Normal and the Cauchy Probability Density Functions

densities would match at the point $x = a = 0$.

The cumulative distribution function is $F(x) = \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{x-a}{b})$. Then the inverse transform generator is, for U uniform on [0,1],

$$X = a + b\tan\{\pi(U - \frac{1}{2})\} \text{ or equivalently } X = a + \frac{b}{\tan(\pi U)}$$

where the second expression follows from the fact that $\tan(\pi(x-\frac{1}{2})) = (\tan \pi x)^{-1}$.

Geometric (p)

This is a discrete distribution which describes the number of (independent) trials necessary to achieve a single success when the probability of a success on each trial is p. The probability function is

$$f(x) = p(1-p)^x, x = 1, 2, 3, ...$$

and the cumulative distribution function is

$$F(x) = P[X \leq x] = 1 - (1-p)^{[x]}, x \geq 0$$

where $[x]$ denotes the integer part of x. To invert the cumulative distribution function of a discrete distribution like this one, we need to refer to a graph of the cumulative distribution function analogous to Figure 3.5. We wish to output an integer value of x which satisfies the inequalities

$$F(x-1) < U \leq F(x).$$

Solving these inequalities for **integer** x,we obtain

$$1 - (1-p)^{x-1} < U \leq 1 - (1-p)^x$$

$$(1-p)^{x-1} > 1 - U \geq (1-p)^x$$

$$(x-1)\ln(1-p) > \ln(1-U) \geq x\ln(1-p)$$

$$(x-1) < \frac{\ln(1-U)}{\ln(1-p)} \leq x$$

Note that changes of direction of the inequality occurred each time we multiplied or divided by negative quantity. We should therefore choose the smallest integer for X which is greater than or equal to $\frac{\ln(1-U)}{\ln(1-p)}$ or equivalently,

$$X = 1 + [\frac{log(1-U)}{log(1-p)}] \text{ or} 1 + [\frac{-E}{log(1-p)}]$$

where we write $-\log(1-U) = E$, an exponential(1) random variable. In *Matlab*, the geometric random number generators is called *geornd* and in R or *Splus* it is called *rgeom*.

Pareto (a, b)

This is one of the simpler families of distributions used in econometrics for modeling quantities with lower bound b.

$$F(x) = 1 - (\frac{b}{x})^a, \text{ for } x \geq b > 0.$$

Then the probability density function is

$$f(x) = \frac{ab^a}{x^{a+1}}$$

and the mean is $E(X) =$. The inverse transform in this case results in

$$X = \frac{b}{(1-U)^{1/a}} \quad \text{or} \quad \frac{b}{U^{1/a}}$$

The special case $b = 1$ is often considered in which case the cumulative distribution function takes the form

$$F(x) = 1 - \frac{1}{x^a}$$

and the inverse

$$X = (1-U)^{1/a}.$$

Logistic

This is again a distribution with shape similar to the normal but closer than is the Cauchy. Indeed as can be seen in Figure 3.7, the two densities are almost indistinguishable, except that the logistic is very slightly more peaked in the center and has slightly more weight in the tails. Again in this graph, parameters have been chosen so that the densities match at the center.

The logistic cumulative distribution function is

$$F(x) = \frac{1}{1 + exp\{-(x-a)/b\}}.$$

and on taking its inverse, the logistic generator is

$$X = a + b\ln(U/(1-U)).$$

Extreme Value

This is one of three possible distributions for modelling extreme statistics such as the largest observation in a very large random sample. As a result it is relevant

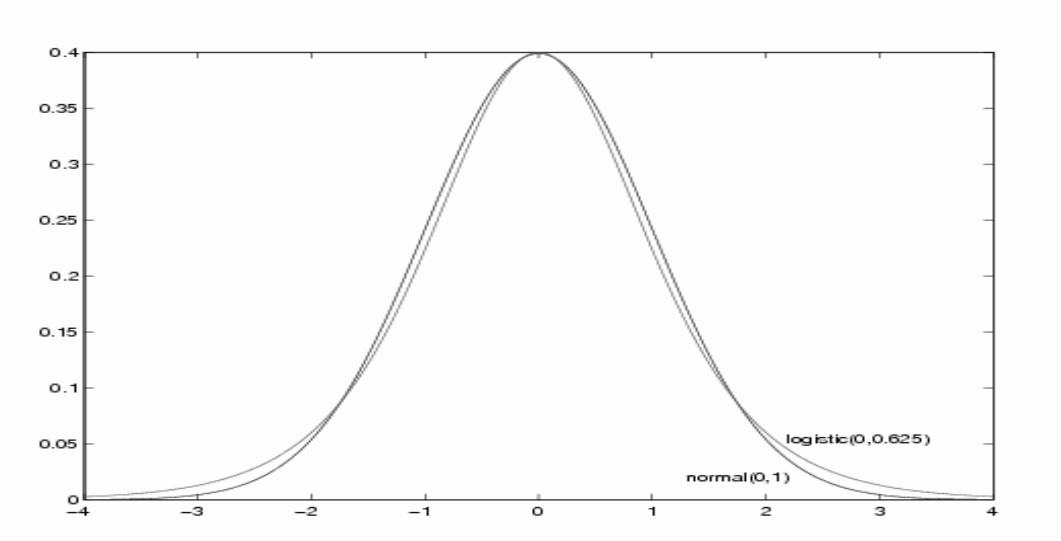

Figure 3.7: Comparison of the Standard Normal and Logistic(0.625) Probability density functions.

to risk management. The cumulative distribution function is for parameters $-\infty < a < \infty$ and $b > 0$,

$$F(x) = 1 - \exp\{-\exp(\frac{x-a}{b})\}.$$

The corresponding inverse is

$$X = a + b\ln(\ln(U)).$$

Weibull Distribution

In this case the parameters a, b are both positive and the cumulative distribution function is

$$F(x) = 1 - exp\{-ax^b\} \text{ for } x \geq 0.$$

The corresponding probability density function is

$$f(x) = abx^{b-1}\exp\{-ax^b\}.$$

Then using inverse transform we may generate X as

$$X = \left\{ \frac{-\ln(1-U)}{a} \right\}^{1/b}.$$

Student's t.

The Student t distribution is used to construct confidence intervals and tests for the mean of normal populations. It also serves as a wider-tailed alternative to the normal, useful for modelling returns which have moderately large outliers. The probability density function takes the form

$$f(x) = \frac{\Gamma((v+1)/2)}{\sqrt{v\pi}\Gamma(v/2)}(1+\frac{x^2}{v})^{-(v+1)/2}, -\infty < x < \infty.$$

The case $v = 1$ corresponds to the Cauchy distribution. There are specialized methods of generating random variables with the Student t distribution we will return to later. In MATLAB, the student's t generator is called *trnd*. In general, *trnd(v,m,n)* generates an $m \times n$ matrix of student's t random variables having v degrees of freedom.

The generators of certain distributions are as described below. In each case a vector of length n with the associated parameter values is generated.

DISTRIBUTION	R and SPLUS	MATLAB
normal	rnorm(n, μ, σ)	normrnd$(\mu, \sigma, 1, n)$ or randn$(1, n)$ if $\mu = 1, \sigma = 1$
Student's t	rt(n, ν)	trnd$(\nu, 1, n)$
exponential	rexp(n, λ)	exprnd$(\lambda, 1, n)$
uniform	runif(n, a, b)	unifrnd$(a, b, 1, n)$ or rand$(1, n)$ if $a = 0, b = 1$
Weibull	rweibull(n, a, b)	weibrnd$(a, b, 1, n)$
gamma	rgamma(n, a, b)	gamrnd$(a, b, 1, n)$
Cauchy	rcauchy(n, a, b)	a+b*trnd$(1, 1, n)$
binomial	rbinom(n, m, p)	binornd$(m, p, 1, n)$
Poisson	rpois(n, λ)	poissrnd$(\lambda, 1, n)$

TABLE: Some Random Number Generators in R,SPLUS and MATLAB

Inversion performs reasonably well for any distribution for which *both the cumulative distribution function and its inverse* can be found in closed form and computed reasonably efficiently. This includes the Weibull, the logistic distribution and most discrete distributions with a small number possible values. However, for other distributions such as the Normal, Student's t, the chi-squared, the Poisson or Binomial with large parameter values, other more specialized methods are usually used, some of which we discuss later.

When the cumulative distribution function is known but not easily inverted, we might attempt to invert it by numerical methods. For example, using the Newton-Ralphson method, we would iterate until convergence the equation

$$X = X - \frac{F(X) - U}{f(X)} \tag{3.6}$$

with $f(X) = F'(X)$, beginning with a good approximation to X. For example we might choose the initial value of $X = X(U)$ by using an easily inverted approximation to the true function $F(X)$. The disadvantage of this approach is that for each X generated, we require an iterative solution to an equation and this is computationally very expensive.

The Acceptance-Rejection Method

Suppose $F(x)$ is a cumulative distribution function and $f(x)$ is the corresponding probability density function. In this case F is continuous and strictly increasing wherever f is positive and so it has a well-defined inverse F^{-1}. Consider the transformation of a point (u, v) in the unit square defined by

$$\begin{aligned} x(u,v) &= F^{-1}(u) \\ y(u,v) &= vf(F^{-1}(u)) = vf(x) \\ &\text{for } 0 < u < 1, \quad 0 < v < 1 \end{aligned}$$

This maps a random point (U, V) uniformly distributed on the unit square into a point (X, Y) uniformly distributed under the graph of the probability density f. The fact that X has cumulative distribution function F follows from its definition as $X = F^{-1}(U)$ and the inverse transform theorem. By the definition of $Y = Vf(X)$ with V uniform on $[0, 1]$ we see that the conditional distribution of Y given the value of X, is uniform on the interval $[0, f(X)]$. Suppose we seek a random number generator for the distribution of X but we are unable to easily invert the cumulative distribution function We can nevertheless use the result that the point (X, Y) is uniform under the density as the basis for one of the simplest yet most useful methods of generating non-uniform variates, the *rejection* or acceptance-rejection method. It is based on the following very simple relationship governing random points under probability density functions.

Theorem 20 *(Acceptance-Rejection)* (X, Y) *is uniformly distributed in the region between the probability density function* $y = f(x)$ *and the axis* $y = 0$ *if and only if the marginal distribution of* X *has density* $f(x)$ *and the conditional distribution of* Y *given* X *is uniform on* $[0, f(X)]$.

Proof. *If a point* (X, Y) *is uniformly distributed under the graph of* $f(x)$ *notice that the probability* $P[a < X < b]$ *is proportional to the area under the*

graph between vertical lines at $x = a$ *and* $x = b$. *In other words* $P[a < X < b]$ *is proportional to* $\int_a^b f(x)dx$. *This implies that* $f(x)$ *is proportional to the probability density function of* X *and provided that* $\int_{-\infty}^{\infty} f(x)dx = 1$, $f(x)$ *is the probability density function of* X. *The converse and the rest of the proof is similar.* ■

Even if the scaling constant for a probability density function is unavailable, in other words if we know $f(x)$ only up to some unknown scale multiple, we can still use Theorem 19 to generate a random variable with probability density f because the X coordinate of a random point uniform under the graph of a constant$\times f(x)$ is the same as that of a random point uniformly distributed under the graph of $f(x)$. The *acceptance-rejection method* works as follows. We wish to generate a random variable from the probability density function $f(x)$. We need the following ingredients:

- A probability density function $g(x)$ with the properties that

  1. the corresponding cumulative distribution function $G(x) = \int_{-\infty}^{x} g(z)dz$ is easily inverted to obtain $G^{-1}(u)$.

  2. $$\sup\{\frac{f(x)}{g(x)}; -\infty < x < \infty\} < \infty. \tag{3.7}$$

For reasonable efficiency we would like the supremum in (3.7) to be as close as possible to one (it is always greater or equal to one).

The condition (3.7) allows us to find a constant $c > 1$ such that $f(x) \leq cg(x)$ for all x. Suppose we are able to generate a point (X, Y) uniformly distributed under the graph of $cg(x)$. This is easy to do using Theorem 19. Indeed we can define $X = G^{-1}(U)$ and $Y = V \times cg(X)$ where U and V are independent $U[0, 1]$. Can we now find a point (X, Y) which is uniformly distributed under the graph of $f(x)$? Since this is a subset of the original region, this is easy. We simple test the point we have already

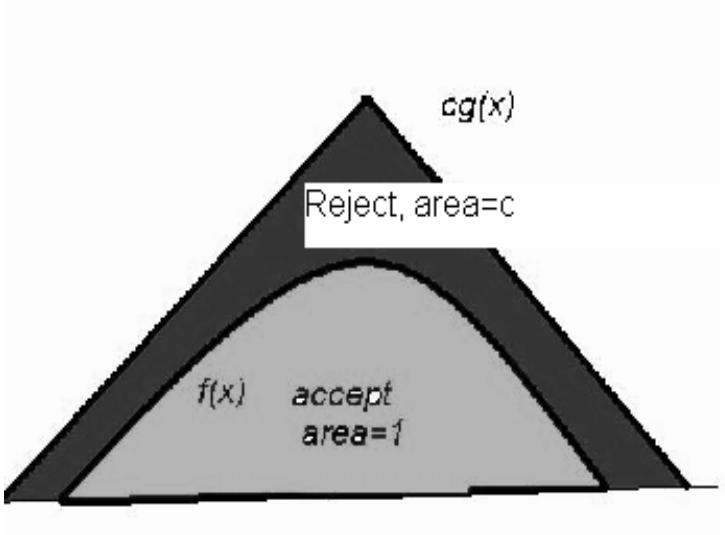

Figure 3.8: The acceptance-Rejection Method

generated to see if it is in this smaller region and if so we use it. If not start over generating a new pair (X, Y), and repeating this until the condition $Y \leq f(X)$ is eventually satisfied, (see Figure **??**).The simplest version of this algorithm corresponds to the case when $g(x)$ is a uniform density on an interval $[a, b]$. In algorithmic form, the acceptance-rejection method is;

1. Generate a random variables $X = G^{-1}(U)$, where U where U is uniform on $[0, 1]$.

2. Generate independent $V \sim U[0, 1]$

3. If $V \leq \frac{f(X)}{cg(X)}$, then return X and exit

4. ELSE go to step 1.

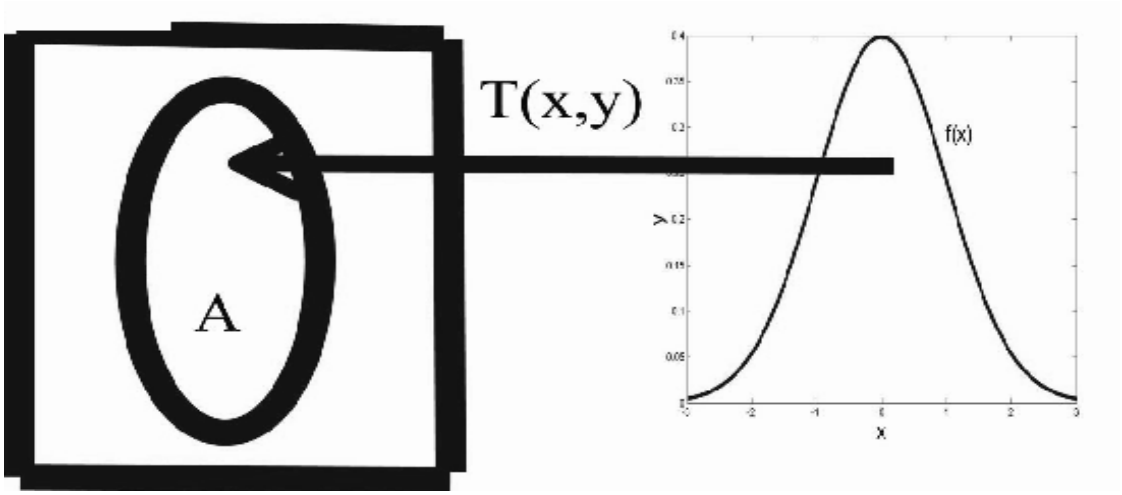

Figure 3.9: $T(x, y)$ is an area Preserving invertible map $f(x, y)$ from the region under the graph of f into the set A, a subset of a rectangle.

The rejection method is useful if the density g is considerably simpler than f both to evaluate and to generate distributions from and if the constant c is close to 1. The number of iterations through the above loop until we exit at step 3 has a geometric distribution with parameter $p = 1/c$ and mean c so when c is large, the rejection method is not very effective.

Most schemes for generating non-uniform variates are based on a transformation of uniform with or without some rejection step. The rejection algorithm is a special case. Suppose, for example, that $T = (u(x, y), v(x, y))$ is a one-one area-preserving transformation of the region $-\infty < x < \infty, 0 < y < f(x)$ into a subset A of a square in R^2 as is shown in Figure 3.9.

Notice that any such transformation defines a random number generator for the density $f(x)$. We need only generate a point (U, V) uniformly distributed in the set A by acceptance-rejection and then apply the inverse transformation T^{-1} to this point, defining $(X, Y) = T^{-1}(U, V)$. Since the transformation is area-preserving, the point (X, Y) is uniformly distributed under the probability

density function $f(x)$ and so the first coordinate X will then have density f. We can think of inversion as a mapping on $[0,1]$ and acceptance-rejection algorithms as an area preserving mapping on $[0,1]^2$.

The most common distribution required for simulations in finance and elsewhere is the *normal distribution*. The following theorem provides the simple connections between the normal distribution in Cartesian and in polar coordinates.

Theorem 21 *If (X,Y) are independent standard normal variates, then expressed in polar coordinates,*

$$(R,\Theta) = (\sqrt{X^2+Y^2}, arctan(Y/X)) \tag{3.8}$$

are independent random variables. $R = \sqrt{X^2+Y^2}$ has the distribution of the square root of a chi-squared(2) or exponential(2) variable. $\Theta = \arctan(Y/X))$ has the uniform distribution on $[0,2\pi]$.

It is easy to show that if (X,Y) *are independent standard normal variates, then* $\sqrt{X^2+Y^2}$ *has the distribution of the square root of a chi-squared(2) (i.e. exponential(2)) variable and* $arctan(Y/X))$*is uniform on* $[0,2\pi]$. The proof of this result is left as a problem.

This observation is the basis of two related popular normal pseudo-random number generators. The *Box-Muller* algorithm uses two uniform$[0,1]$ variates U,V to generate R and Θ with the above distributions as

$$R = \{-2\ln(U)\}^{1/2}, \Theta = 2\pi V \tag{3.9}$$

and then defines two independent normal(0,1) variates as

$$(X,Y) = R(\cos\Theta, \sin\Theta) \tag{3.10}$$

Note that normal variates must be generated in pairs, which makes simulations involving an even number of normal variates convenient. If an odd number are required, we will generate one more than required and discard one.

Theorem 22 *(Box-Muller Normal Random Number generator)*

Suppose (R, Θ) *are independent random variables such that* R^2 *has an exponential distribution with mean* 2 *and* Θ *has a* Uniform$[0, 2\pi]$ *distribution. Then* $(X, Y) = (R\cos\Theta, R\sin\Theta)$ *is distributed as a pair of independent normal variates.*

Proof. Since R^2 has an exponential distribution, R has probability density function

$$\begin{aligned} f_R(r) &= \frac{d}{dr}P[R \le r] \\ &= \frac{d}{dr}P[R^2 \le r^2] \\ &= \frac{d}{dr}(1 - e^{-r^2/2}) \\ &= re^{-r^2/2}, \quad \text{for } r > 0. \end{aligned}$$

and Θ has probability density function $f_\Theta(\theta) = \frac{1}{2\pi}$ for $0 < \theta < 2\pi$. Since $r = r(x, y) = \sqrt{x^2 + y^2}$ and $\theta(x, y) = \arctan(y/x)$, the Jacobian of the transformation is

$$\begin{aligned} \left|\frac{\partial(r,\theta)}{\partial(x,y)}\right| &= \begin{vmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} \end{vmatrix} \\ &= \begin{vmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ \frac{-y}{x^2+y^2} & \frac{x}{x^2+y^2} \end{vmatrix} \\ &= \frac{1}{\sqrt{x^2+y^2}} \end{aligned}$$

Consequently the joint probability density function of (X, Y) is given by

$$\begin{aligned} f_\Theta(\arctan(y/x)) f_R(\sqrt{x^2+y^2}) \left|\frac{\partial(r,\theta)}{\partial(x,y)}\right| &= \frac{1}{2\pi} \times \sqrt{x^2+y^2} e^{-(x^2+y^2)/2} \times \frac{1}{\sqrt{x^2+y^2}} \\ &= \frac{1}{2\pi} e^{-(x^2+y^2)/2} \\ &= \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} \end{aligned}$$

and this is joint probability density function of two independent standard normal random variables. ■

The tails of the distribution of the pseudo-random numbers produced by the Box-Muller method are quite sensitive to the granularity of the uniform generator. For this reason although the Box-Muller is the simplest normal generator it is not the method of choice in most software. A related alternative algorithm for generating standard normal variates is the *Marsaglia polar* method. This is a modification of the Box-Muller generator designed to avoid the calculation of sin or cos. Here we generate a point $(Z_1, Z_2)$from the uniform distribution on the unit circle by rejection, generating the point initially from the square $-1 \leq z_1 \leq 1, -1 \leq z_2 \leq 1$ and accepting it when it falls in the unit circle or if $z_1^2 + z_2^2 \leq 1$. Now suppose that the points (Z_1, Z_2) is uniformly distributed inside the unit circle. Then for $r > 0$,

$$\begin{aligned} P[\sqrt{-2\log(Z_1^2 + Z_2^2)} \leq r] &= P[Z_1^2 + Z_2^2 \geq \exp(-r^2/2)] \\ &\frac{1 - \text{area of a circle of radius } \exp(-r^2/2)}{\text{area of a circle of radius 1}} \\ &= 1 - e^{-r^2/2}. \end{aligned}$$

This is exactly the same cumulative distribution function as that of the random variable R in Theorem 21. It follows that we can replace R^2 by $-2log(Z_1^2+Z_2^2)$. Similarly, if (Z_1, Z_2) is uniformly distributed inside the unit circle then the angle subtended at the origin by a line to the point (X, Y) is random and uniformly$[0, 2\pi]$ distributed and so we can replace $\cos\Theta$, and $\sin\Theta$ by $\frac{Z_1}{\sqrt{Z_1^2+Z_2^2}}$ and $\frac{Z_2}{\sqrt{Z_1^2+Z_2^2}}$ respectively. The following theorem is therefore proved.

Theorem 23 *If the point (Z_1, Z_2) is uniformly distributed in the unit circle $Z_1^2 + Z_2^2 \leq 1$, then the pair of random variables defined by*

$$\begin{aligned} X &= \sqrt{-2log(Z_1^2 + Z_2^2)}\frac{Z_1}{\sqrt{Z_1^2 + Z_2^2}} \\ Y &= \sqrt{-2log(Z_1^2 + Z_2^2)}\frac{Z_2}{\sqrt{Z_1^2 + Z_2^2}} \end{aligned}$$

are independent standard normal variables.

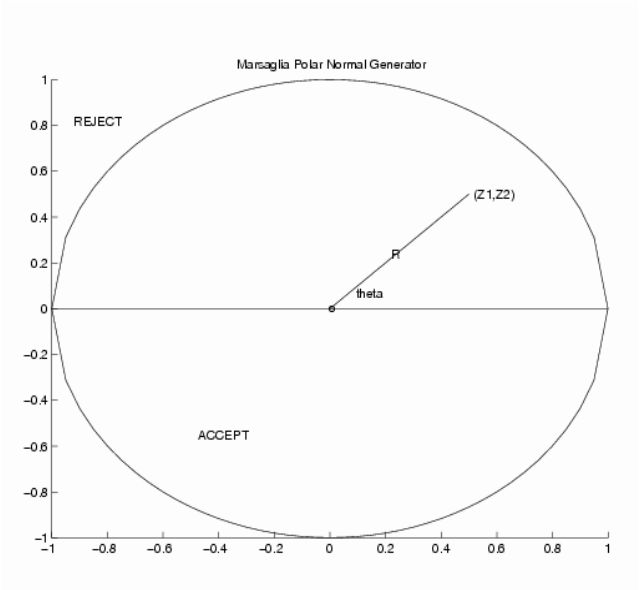

Figure 3.10: Marsaglia's Method for Generating Normal Random Numbers

If we use acceptance-rejection to generate uniform random variables Z_1, Z_2 inside the unit circle, the probability that a point generated inside the square falls inside the unit circle is $\pi/4$,so that on average around $4/\pi \approx 1.27$ pairs of uniforms are needed to generate a pair of normal variates.

The speed of the Marsaglia polar algorithm compared to that of the Box-Muller algorithm depends on the relative speeds of generating uniform variates versus the sine and cosine transformations. The Box-Muller and Marsaglia polar method are illustrated in Figure 3.10:

Unfortunately the speed of these normal generators is not the only consideration. If we run a linear congruential generator through a full period we have seen that the points lie on a lattice, doing a reasonable job of filling the two dimensional rectangle. Transformations like (3.10) are highly non-linear functions of (U, V) stretching the space in some places and compressing it in others. It would not be too surprising if, when we apply this transformation to

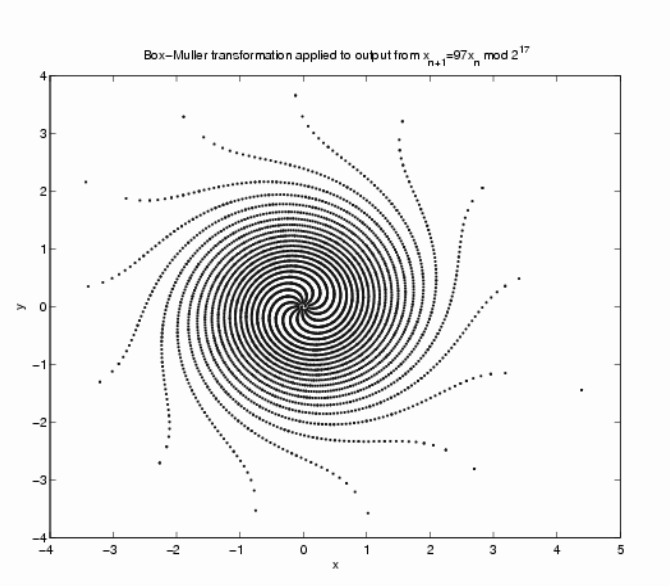

Figure 3.11: Box Muller transformation applied to the output to $x_n = 97x_{n-1} \bmod 2^{17}$

our points on a lattice, they do not provide the same kind of uniform coverage of the space. In Figure 3.11 we see that the lattice structure in the output from the linear congruential generator results in an interesting but alarmingly non-normal pattern, particularly sparse in the tails of the distribution. Indeed, if we use the full-period generator $x_n = 16807x_{n-1} \bmod (2^{31} - 1)$ the smallest possible value generated for y is around -4.476 although in theory there should be around 8,000 normal variates generated below this.

The normal random number generator in *Matlab* is called *normrnd* or for standard normal *randn*. For example $normrnd(\mu, \sigma, m, n)$ generates a matrix of $m \times n$ pseudo-independent normal variates with mean μ and standard deviation σ and *rand(m,n)* generates an $m \times n$ matrix of standard normal random numbers. A more precise algorithm is to use inverse transform and a highly refined rational approximation to normal inverse cumulative distribution func-

tion available from P.J. Acklam (2003). The Matlab implementation of this inverse c.d.f. is called *ltqnorm* after application of a refinement, achieves full machine precision. In R or Splus, the normal random number generator is called *rnorm.* The inverse random number function in Excel has been problematic in many versions. These problems appear to have been largely corrected in *Excel 2002*, although there is still significant error (roughly in the third decimal) in the estimation of lower and upper tail quantiles. The following table provides a comparison of the *normsinv* function in Excel and the Matlab inverse normal *norminv.* The "exact" values agree with the values generated by Matlab *norminv* to the number of decimals shown.

p	Excel 2002	Exact
10^{-1}	-1.281551939	-1.281551566
10^{-2}	-2.326347	-2.326347874
10^{-3}	-3.090252582	-3.090232306
10^{-4}	-3.719090272	-3.719016485
10^{-5}	-4.265043367	-4.264890794
10^{-6}	-4.753672555	-4.753424309
10^{-7}	-5.199691841	-5.199337582
10^{-8}	-5.612467211	-5.612001244
10^{-9}	-5.998387182	-5.997807015
10^{-10}	-6.362035677	-6.361340902

The Lognormal Distribution

If Z is a normal random variable with mean μ and variance σ^2, then we say that the distribution of $X = e^Z$ is lognormal with mean $E(X) = \eta = \exp\{\mu+\sigma^2/2\} > 0$ and parameter $\sigma > 0$. Because a lognormal random variable is obtained by *exponentiating* a normal random variable it is strictly positive, making it a reasonable candidate for modelling quantities such as stock prices, exchange rates, lifetimes, though in a fools paradise in which stock prices and lifetimes

are never zero. To determine the lognormal probability density function, notice that

$$\begin{aligned} P(X \leq x] &= P[e^Z \leq x] \\ &= P[Z \leq \ln(x)] \\ &= \Phi(\frac{\ln(x) - \mu}{\sigma}) \quad \text{with } \Phi \text{ the standard normal c.d.f.} \end{aligned}$$

and differentiating to obtain the probability density function $g(x|\eta, \sigma)$ of X, we obtain

$$\begin{aligned} g(x|\eta, \sigma) &= \frac{d}{dx}\Phi(\frac{\ln(x) - \mu}{\sigma}) \\ &= \frac{1}{x\sigma\sqrt{2\pi}} \exp\{-(\ln(x) - \mu)^2/2\sigma^2\} \\ &= \frac{1}{x\sigma\sqrt{2\pi}} \exp\{-(\ln(x) - \ln(\eta) + \sigma^2/2)^2/2\sigma^2\} \end{aligned}$$

A random variable with a lognormal distribution is easily generated by generating an appropriate normal random variable Z and then exponentiating. We may use either the parameter μ, the mean of the random variable Z in the exponent or the parameter η, the expected value of the lognormal. The relationship is not as simple as a naive first impression might indicate since

$$E(e^Z) \neq e^{E(Z)}.$$

Now is a good time to accommodate to this correction factor of $\sigma^2/2$ in the exponent

$$\eta = E(e^Z) = e^{E(Z)+\sigma^2/2} = e^{\mu+\sigma^2/2} \quad \text{or,}$$
$$E(e^{Z-\mu-\sigma^2/2}) = 1$$

since a similar factor appears throughout the study of stochastic integrals and mathematical finance. Since the lognormal distribution is the one most often used in models of stock prices, it is worth here recording some of its conditional moments used in the valuation of options. In particular if X has a lognormal

distribution with mean $\eta = e^{\mu+\sigma^2/2}$ and volatility parameter σ, then for any p and $l > 0$,

$$\begin{aligned}
E[X^p I(X > l)] &= \frac{1}{\sigma\sqrt{2\pi}} \int_l^\infty x^{p-1} \exp\{-(\ln(x)-\mu)^2/2\sigma^2\} dx \\
&= \frac{1}{\sigma\sqrt{2\pi}} \int_{\ln(l)}^\infty e^{zp} \exp\{-(z-\mu)^2/2\sigma^2\} dz \\
&= \frac{1}{\sigma\sqrt{2\pi}} e^{p\mu+p^2\sigma^2/2} \int_{\ln(l)}^\infty \exp\{-(z-\xi)^2/2\sigma^2\} dz \quad \text{where } \xi = \mu + \sigma^2 p \\
&= e^{p\mu+p^2\sigma^2/2} \Phi(\frac{\xi - \ln(l)}{\sigma}) \\
&= \eta^p \exp\{-\frac{\sigma^2}{2} p(1-p)\} \Phi(\sigma^{-1}\ln(\eta/l) + \sigma(p - \frac{1}{2})) \qquad (3.11)
\end{aligned}$$

where Φ is the standard normal cumulative distribution function.

Application: A Discrete Time Black-Scholes Model

Suppose that a stock price $S_t, t = 1, 2, 3, ...$ is generated from an independent sequence of returns Z_1, Z_2 over non-overlapping time intervals. If the value of the stock at the end of day $t = 0$ is S_0, and the return on day 1 is Z_1 then the value of the stock at the end of day 1 is $S_1 = S_0 e^{Z_1}$. There is some justice in the use of the term "return" for Z_1 since for small values Z_1, $S_0 e^{Z_1} \simeq S_0(1 + Z_1)$ and so Z_1 is roughly $\frac{S_1 - S_0}{S_1}$. Assume similarly that the stock at the end of day i has value $S_i = S_{i-1}\exp(Z_i)$. In general for a total of j such periods (suppose there are n such periods in a year) we assume that $S_j = S_0 \exp\{\sum_{i=1}^j Z_i\}$ for independent random variables Z_i all have the same normal distribution. Note that in this model the returns over non-overlapping independent periods of time are independent. Denote $var(Z_i) = \sigma^2/N$ so that

$$var(\sum_{i=1}^N Z_i) = \sigma^2$$

represents the squared annual volatility parameter of the stock returns. Assume that the annual interest rate on a risk-free bond is r so that the interest rate per period is r/N.

Recall that the risk-neutral measure Q is a measure under which the stock price, discounted to the present, forms a martingale. In general there may be many such measures but in this case there is only one under which the stock price process has a similar lognormal representation $S_j = S_0 \exp\{\sum_{i=1}^{j} Z_i\}$ for independent normal random variables Z_i. Of course under the risk neutral measure, the normal random variables Z_i may have a different mean. Full justification of this model and the uniqueness of the risk-neutral distribution really relies on the continuous time version of the Black Scholes described in Section 2.6. Note that if the process

$$e^{-rt/N} S_j = S_0 \exp\{\sum_{i=1}^{j} (Z_i - \frac{r}{N})\}$$

is to form a martingale under Q, it is necessary that

$$\begin{aligned} E_Q[S_{j+1}|H_t] &= S_j \quad \text{or} \\ E_Q[S_j \exp\{Z_{j+1} - \frac{r}{N}\}|H_j] &= S_j E_Q[\exp\{Z_{j+1} - \frac{r}{N}\}] \\ &= S_j \end{aligned}$$

and so $\exp\{Z_{j+1} - \frac{r}{N}\}$ must have a lognormal distribution with expected value 1. Recall that, from the properties of the lognormal distribution,

$$E_Q[\exp\{Z_{t+1} - \frac{r}{N}\}] = \exp\{E_Q(Z_{t+1}) - \frac{r}{N} + \frac{\sigma^2}{2N}\}$$

since $var_Q(Z_{t+1}) = \frac{\sigma^2}{N}$. In other words, for each i the expected value of Z_i is, under Q, equal to $\frac{r}{N} - \frac{\sigma^2}{2N}$. So under Q, S_j has a lognormal distribution with mean

$$S_0 e^{rj/N}$$

and volatility parameter $\sigma\sqrt{j/N}$.

Rather than use the Black-Scholes formula of Section 2.6, we could price a call option with maturity $j = NT$ periods from now by generating the random path $S_i, i = 1, 2, ...j$ using the lognormal distribution for S_j and then averaging

the returns discounted to the present. The value at time $j = 0$ of a call option with exercise price K is an average of simulated values of

$$e^{-rj/N}(S_j - K)^+ = e^{-rj/N}(S_0 \exp\{\sum_{i=1}^{T} Z_i\} - K)^+,$$

with the simulations conducted under the risk-neutral measure Q with initial stock price the current price S_0. Thus the random variables Z_i are independent $N(\frac{r}{N} - \frac{\sigma^2}{2N}, \frac{\sigma^2}{N})$. The following *Matlab* function simulates the stock price over the whole period until maturity and then values a European call option on the stock by averaging the discounted returns.

Example 24 *(simulating the return from a call option)*

Consider simulating a call option on a stock whose current value is $S_0 =$ \$1.00. The option expires in j days and the strike price is $K =$ \$1.00. We assume constant spot (annual) interest rate r and the stock price follows a lognormal distribution with annual volatility parameter σ. The following Matlab function provides a simple simulation and graph of the path of the stock over the life of the option and then outputs the discounted payoff from the option.

function z=plotlogn(r,sigma,T, K)

% outputs the discounted simulated return on expiry of a call option (per dollar pv of stock).

% Expiry =T years from now, $(T = j/N)$

% current stock price=\$1. $(= S_0)$, r = annual spot interest rate, sigma=annual volatility $(=\sigma)$,

% K= strike price.

N=250 ; % N is the assumed number of business days in a year.

j=N*T; % the number of days to expiry

s = sigma/sqrt(N); % s is volatility per period

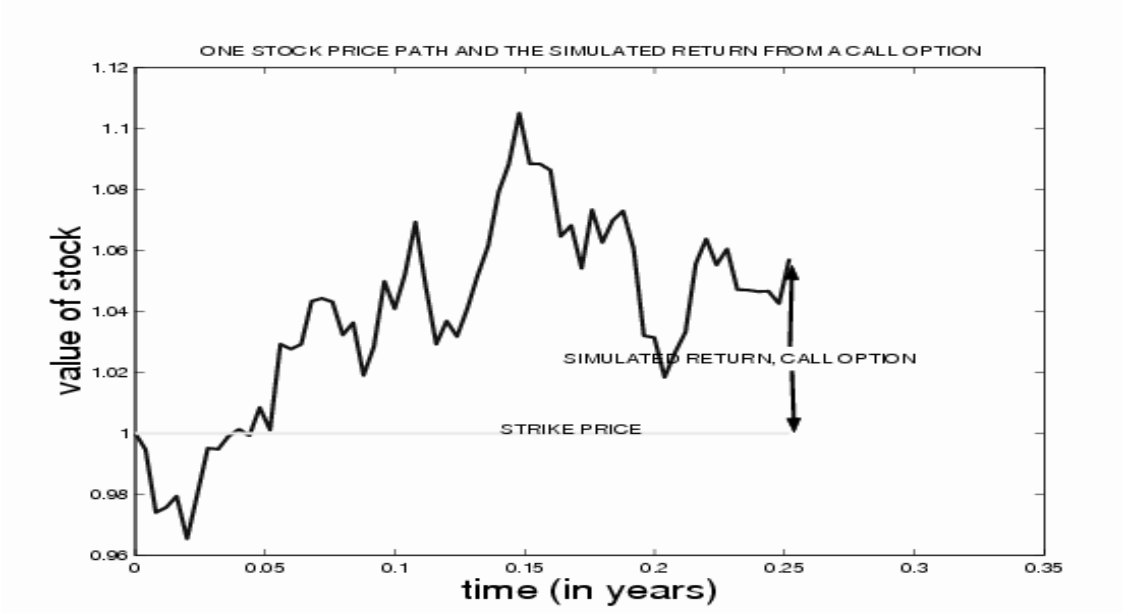

Figure 3.12: One simulation of the return from a call option with strike price $1.00

```
mn = r/N - s^2/2;                % mn= mean of the normal increments per period
y=exp(cumsum(normrnd(mn,s,j,1)));
y=[1 y'];                        % the value of the stock at times 0,...,
x = (0:j)/N;                     % the time points i
plot(x,y,'-',x,K*ones(1,j+1),'y')
xlabel('time (in years)')
ylabel('value of stock')
title('SIMULATED RETURN FROM CALL OPTION')
z = exp(-r*T)*max(y(j+1)-K, 0);          % payoff from option  discounted to
present
```

Figure 3.12 resulted from one simulation run with $r = .05, j = 63$ (about 3 months), $\sigma = .20, K = 1$.

The return on this run was the discounted difference between the terminal value of the stock and the strike price or around 0.113. We may repeat this many times, averaging the discounted returns to estimate the present value of the option.

For example to value an at the money call option with exercise price=the initial price of the stock=\$1, 5% annual interest rate, 20% annual volatility and maturity 0.25 years from the present, we ran this function 100 times and averaged the returns to estimate the option price as *0.044978.* If we repeat the identical statement, the output is different, for example *option val= 0.049117* because each is an average obtained from only 100 simulations. Averaging over more simulations would result in greater precision, but this function is not written with computational efficiency in mind. We will provide more efficient simulations for this problem later. For the moment we can compare the price of this option as determined by simulation with the exact price according to the Black-Scholes formula. This formula was developed in Section 2.6. The price of a call option at time $t = 0$ given by

$$V(S_T, T) = S_T \Phi(d_1) - K e^{-rT/N} \Phi(d_2)$$

where

$$d_1 = \frac{\log(S_T/K) + (r + \frac{\sigma^2}{2})T/N}{\sigma\sqrt{T/N}} \quad \text{and} \quad d_2 = \frac{\log(S_T/K) + (r - \frac{\sigma^2}{2})T/N}{\sigma\sqrt{T/N}}$$

and the Matlab function which evaluates this is the function *blsprice* which gives, in this example, and exact price on entering *[CALL,PUT] =BLSPRICE(1,1,.05,63/250,.2,0)* which returns the value *CALL=0.0464.* With these parameters, 4.6 cents on the dollar allows us to lock in any anticipated profit on the price of a stock (or commodity if the lognormal model fits) for a period of about three months. The fact that this can be done cheaply and with ease is part of the explanation for the popularity of derivatives as tools for hedging.

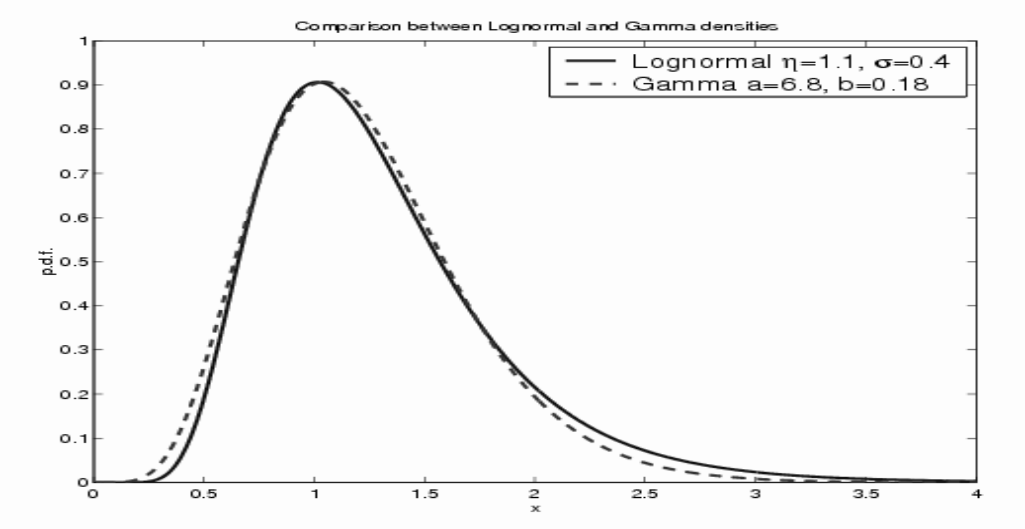

Figure 3.13: Comparison between the Lognormal and the Gamma densities

Algorithms for Generating the Gamma and Beta Distributions

We turn now to algorithms for generating the *Gamma distribution* with density

$$f(x|a,b) = \frac{x^{a-1}e^{-x/b}}{\Gamma(a)b^a}, \text{ for } x > 0, a > 0, b > 0. \tag{3.12}$$

The exponential distribution ($a = 1$) and the chi-squared (corresponding to $a = \nu/2, b = 2$, for ν integer) are special cases of the Gamma distribution. The gamma family of distributions permits a wide variety of shapes of density functions and is a reasonable alternative to the lognormal model for positive quantities such as asset prices. In fact for certain parameter values the gamma density function is very close to the lognormal. Consider for example a typical lognormal random variable with mean $\eta = 1.1$ and volatility $\sigma = 0.40$.

The probability density functions can be quite close as in Figure **??**. Of course the lognormal, unlike the gamma distribution, has the additional attractive feature that a product of independent lognormal random variables also has

a lognormal distribution.

Another common distribution closely related to the gamma is the *Beta distribution* with probability density function defined for parameters $a, b > 0$,

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, 0 \le x \le 1. \tag{3.13}$$

The beta density obtains for example as the distribution of order statistics in a sample from independent uniform $[0,1]$ variates. This is easy to see. For example if $U_1, ..., U_n$ are independent uniform random variables on the interval $[0,1]$ and if $U_{(k)}$ denotes the $k'th$ largest of these n values, then

$$\begin{aligned} P[U_{(k)} < x] &= P[\text{there are } k \text{ or more values less than } x] \\ &= \sum_{j=k}^{n} \binom{n}{j} x^j (1-x)^{n-j}. \end{aligned}$$

Differentiating we find the probability density function of $U_{(k)}$ to be

$$\begin{aligned} \frac{d}{dx} \sum_{j=k}^{n} \binom{n}{j} x^j (1-x)^{n-j} &= \sum_{j=k}^{n} \binom{n}{j} \{j x^{j-1}(1-x)^{n-j} + (n-j)x^j(1-x)^{n-j-1}\} \\ &= k\binom{n}{k} x^{k-1}(1-x)^{n-k} \\ &= \frac{\Gamma(n+1)}{\Gamma(k)\Gamma(n-k+1)} x^{k-1}(1-x)^{n-k} \end{aligned}$$

and this is the beta density with parameters $a = k-1$, $b = n-k+1$. Order statistics from a Uniform sample therefore have a beta distribution with the k'th order statistic having the Beta$(k-1, n-k+1)$ distribution. This means that order statistics from more general continuous distributions can be easily generated using the inverse transform and a beta random variable. For example suppose we wish to simulate the largest observation in a normal(μ, σ^2) sample of size 100. Rather than generate a sample of 100 normal observations and take the largest, we can simulate the value of the largest uniform order statistic $U_{(100)} \sim Beta(99, 1)$ and then $\mu + \sigma\Phi^{-1}(U_{(100)})$ (with Φ^{-1} the standard normal inverse cumulative distribution function) is the required simulated value. This may be used to render simulations connected with risk management more efficient.

The following result lists some important relationships between the Gamma and Beta distributions. For example it allows us to generate a Beta random variable from two independent Gamma random variables.

Theorem 25 *(Gamma distribution) If X_1, X_2 are independent Gamma (a_1, b) and Gamma (a_2, b) random variables, then $Z = \frac{X_1}{X_1+X_2}$ and $Y = X_1 + X_2$ are independent random variables with Beta (a_1, a_2) and Gamma $(a_1 + a_2, b)$ distributions respectively. Conversely, if (Z, Y) are independent variates with Beta (a_1, a_2) and the Gamma $(a_1 + a_2, b)$ distributions respectively, then $X_1 = YZ$, and $X_2 = Y(1 - Z)$ are independent and have the Gamma (a_1, b) and Gamma (a_2, b) distributions respectively.*

Proof. *Assume that X_1, X_2 are independent Gamma (a_1, b) and Gamma (a_2, b) variates. Then their joint probability density function is*

$$f_{X_1X_2}\ (x_1, x_2) = \frac{1}{\Gamma(a_1)\Gamma(a_2)} x_1^{a_1-1} x_2^{a_2-1} e^{-(x_1+x_2)/b},\ \text{for } x_1 > 0, x_2 > 0.$$

Consider the change of variables $x_1(z, y) = zy, x_2(z, y) = (1 - z)y$. Then the Jacobian of this transformation is given by

$$\begin{vmatrix} \frac{\partial x_1}{\partial z} & \frac{\partial x_1}{\partial y} \\ \frac{\partial x_2}{\partial z} & \frac{\partial x_2}{\partial y} \end{vmatrix} = \begin{vmatrix} y & z \\ -y & 1-z \end{vmatrix}$$

$$= y.$$

Therefore the joint probability density function of (z, y) is given by

$$f_{z,y}(z, y) = f_{X_1X_2}\ (zy, (1 - z)y) \begin{vmatrix} \frac{\partial x_1}{\partial z} & \frac{\partial x_1}{\partial y} \\ \frac{\partial x_2}{\partial z} & \frac{\partial x_2}{\partial y} \end{vmatrix}$$

$$= \frac{1}{\Gamma(a_1)\Gamma(a_2)} z^{a_1-1}(1 - z)^{a_2-1} y^{a_1+a_2-1} e^{-y/b}, \text{for } 0 < z < 1, y > 0$$

$$= \frac{\Gamma(a_1 + a_2)}{\Gamma(a_1)\Gamma(a_2)} z^{a_1-1}(1 - z)^{a_2-1} \times \frac{1}{\Gamma(a_1 + a_2)} y^{a_1+a_2-1} e^{-y/b}, \text{for } 0 < z < 1, y > 0$$

and this is the product of two probability density functions, the Beta(a_1, a_2) density for Z and the Gamma$(a_1 + a_2, b)$ probability density function for Y. The converse holds similarly. ■

This result is a basis for generating gamma variates with integer value of the parameter a (sometimes referred to as the shape parameter). According to the theorem, if a is integer and we sum a independent Gamma(1,b) random variables the resultant sum has a Gamma(a, b) distribution. Notice that $-b\log(U_i)$ for uniform$[0, 1]$ random variable U_i is an exponential or a Gamma$(1, b)$ random variable. Thus $-b\log(\prod_{i=1}^{n} U_i)$ generates a gamma (n, b) variate for independent uniform U_i. The computation required for this algorithm, however, increases linearly in the parameter $a = n$, and therefore alternatives are required, especially for large a. Observe that the *scale parameter* b is easily handled in general: simply generate a random variable with scale parameter 1 and then multiply by b. Most algorithms below, therefore, are only indicated for $b = 1$.

For large a Cheng (1977) uses acceptance-rejection from a density of the form

$$g(x) = \lambda\mu\frac{x^{\lambda-1}}{(\mu + x^{\lambda})^2}dx \quad , x > 0 \tag{3.14}$$

called the *Burr XII distribution.* The two parameters μ and λ of this density (μ is not the mean) are chosen so that it is as close as possible to the gamma distribution. We can generate a random variable from (3.14) by inverse transform as $G^{-1}(U) = \{\frac{\mu U}{1-U}\}^{1/\lambda}$.

A much simpler function for dominating the gamma densities is a minor extension of that proposed by Ahrens and Dieter (1974). It corresponds to using as a dominating probability density function

$$g(x) = \begin{cases} \frac{x^{a-1}}{k^{a-1}(\frac{k}{a}+\exp(-k))} & 0 \le x \le k \\ \frac{k^{a-1}e^{-x}}{k^{a-1}(\frac{k}{a}+\exp(-k))} & x > k \end{cases} \quad , x > k \tag{3.15}$$

Other distributions that have been used as dominating functions for the Gamma are the Cauchy (Ahrens and Dieter), the Laplace (Tadakamalla), the exponential (Fishman), the Weibull, the relocated and scaled t distribution with 2 degrees of freedom (Best), a combination of normal density (left part) and exponential density (right part) (Ahrens and Dieter), and a mixture of two

Erlang distributions (Gamma with integral shape parameter α).

Best's algorithm generates a Student's t_2 variate as

$$Y = \frac{\sqrt{2}(U - 1/2)}{\sqrt{U(1-U}} \tag{3.16}$$

where $U \sim U[0,1]$. Then Y has the Students t distribution with 2 degrees of freedom having probability density function

$$g(y) = \frac{1}{(2+y^2)^{3/2}}. \tag{3.17}$$

We then generate a random variable $X = (a-1) + Y\sqrt{3a/2 - 3/8}$ and apply a rejection step to X to produce a Gamma random variable. See Devroye (p. 408) for details.

Most of the above algorithms are reasonably efficient only for $a > 1$ with the one main exception being the combination of power of x and exponential density suggested by Ahrens and Dieter above. Cheng and Feast (1979) also suggest a ratio of uniforms algorithm for the gamma distribution, $a > 1$.

A final fast and simple procedure for generating a gamma variate with $a > 1$ is due to Marsaglia and Tsang (2000) and generates a gamma variate as the cube of a suitably scaled normal. Given a fast generator of the Normal to machine precision, this is a highly efficient rejection technique. We put $d = a - \frac{1}{3}$ and generate a standard normal random variable X and a uniform variate U until, with $V = (1 + \frac{X}{\sqrt{9d}})^3$, the following inequality holds:

$$\ln(U) < \frac{X^2}{2} + d - dV + d\ln(V).$$

When this inequality is satisfied, we accept the value $d \times V$ as obtained from the Gamma$(a, 1)$ distribution. As usual multiplication by b results in a Gamma(a, b) random variable. The efficiency of this algorithm appears to be very high (above 96% for $a > 1$).

In the case $0 < a < 1$, Stuart's theorem below allows us to modify a Gamma variate with $a > 1$ to one with $a < 1$. We leave the proof of the theorem as an exercise.

Theorem 26 *(Stuart) Suppose U is uniform $[0,1]$ and X is Gamma $(a+1,1)$ independent of U. Then $XU^{1/a}$ has a gamma $(a,1)$ distribution*

The Matlab function *gamrnd* uses Best's algorithm and acceptance rejection for $\alpha > 1$. For $\alpha < 1$, it uses Johnk's generator, which is based on the following theorem.

Theorem 27 *(Johnk) Let U and V be independent Uniform[0,1] random variables. Then the conditional distribution of*

$$X = \frac{U^{1/\alpha}}{U^{1/\alpha} + V^{1/(1-\alpha)}}$$

given that the denominator $U^{1/\alpha} + V^{1/(1-\alpha)} < 1$ is Beta$(\alpha, 1-\alpha)$.

Multiplying this beta random variable by an independent exponential (1) results in a Gamma$(\alpha, 1)$ random variable.

Toward generating the *beta distribution,* use of Theorem 24 and the variable $Z = \frac{X_1}{X_1+X_2}$ with X_1, X_2 independent gamma variates is one method of using a gamma generator to produce beta variates, and this is highly competitive as long as the gamma generator is reasonably fast. The MATLAB generator is *betarnd(a,b,1,n)* Alternatives are, as with the gamma density, rejection from a Burr XII density (Cheng, 1978) and use of the following theorem as a generator (due to Johnk). This a more general version of the theorem above.

Theorem 28 *(Beta distribution)*

Suppose U, V are independent uniform$[0,1]$ variates. Then the conditional distribution of

$$X = \frac{U^{1/a}}{U^{1/a} + V^{1/b}} \tag{3.18}$$

given that $U^{1/a} + V^{1/b} \leq 1$ is Beta (a,b). Similarly the conditional distribution of $U^{1/a}$ given that $U^{1/a} + V^{1/b} \leq 1$ is Beta $(a+1,b)$.

Proof. *Define a change of variables*

$$X = \frac{U^{1/a}}{U^{1/a} + V^{1/b}}, Y = U^{1/a} + V^{1/b}$$

$$\text{or } U = (YX)^a \text{ and } V = [(1-X)Y]^b$$

so that the joint probability density function of (X, Y) *is given by*

$$f_{X,Y}(x,y) = f_{U,V}((yx)^a, [(1-x)y]^b) \begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{vmatrix}$$

$$= aby^{a+b-1}x^{a-1}(1-x)^{b-1}$$

provided either $(0 < x < 1$ and $y < 1)$ or $(1 - \frac{1}{y} < x < \frac{1}{y}$ and $1 < y < 2)$. *Notice that in the case* $y < 1$, *the range of values of* x *is the unit interval and does not depend on* y *and so the conditional probability density function of* X *given* $Y = y$ *is a constant times* $x^{a-1}(1-x)^{b-1}$, *i.e. is the Beta*(a, b) *probability density function. The rest of the proof is similar.* ■

A generator exploiting this theorem produces pairs (U, V) until the condition is satisfied and then transforms to the variable X. However, the probability that the condition is satisfied is $\frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+1)}$ which is close to 0 unless a, b are small, so this procedure should be used only for small values of both parameters. Theorems 24 and 25 together provide an algorithm for generating Gamma variates with non-integral a from variates with integral ones. For example if X is Gamma$(4, 1)$and Z is independent Beta $(3.4, .6)$then XZ is Gamma $(3.4, 1)$.

There are various other continuous distributions commonly associated with statistical problems. For example the *Student's t-distribution* with ν degrees of freedom is defined as a ratio $\sqrt{\frac{2\nu}{X}}Z$ where Z is standard normal and X is gamma $(\frac{\nu}{2}, 2)$. Alternatively, we may use $\sqrt{\nu}\frac{X-1/2}{\sqrt{X(1-X)}}$where X is generated as a symmetric beta$(\nu/2, \nu/2)$ variate.

Example 29 *(some alternatives to lognormal distribution)*

The assumption that stock prices, interest rates, or exchange rates follow a lognormal distribution is a common exercise in wishful thinking. The lognormal

distribution provides a crude approximation to many financial time series, but other less theoretically convenient families of distributions sometimes provide a better approximation. There are many possible alternatives, including the students t distribution and the stable family of distributions discussed later. Suppose, for the present, we modify the usual normal assumption for stock returns slightly by assuming that the log of the stock price has a distribution "close" to the normal but with somewhat more weight in the tails of the distribution. Specifically assume that under the Q measure, $S_T = S_0 \exp\{\mu + cX\}$ where X has cumulative distribution function $F(x)$. Some constraint is to be placed on the constant c if we are to compare the resulting prices with the Black-Scholes model and it is natural to require that both models have identical volatility, or identical variance of returns. Since the variance of the return in the Black Scholes model over a period of length T is $\sigma^2 T$ where σ is the annual volatility, we therefore require that

$$var(cX) = \sigma^2 T \text{ or } \quad c = \sqrt{\frac{\sigma^2 T}{var(X)}}.$$

The remaining constraint is required of all option pricing measures is the martingale constraint and this implies that the discounted asset price is a martingale, and in consequence

$$e^{-rT} E_Q S_T = S_0. \tag{3.19}$$

Letting the moment generating function of X be

$$m(s) = Ee^{sX},$$

the constraint (3.19) becomes

$$e^{\mu - rT} m(c) = 1$$

and solving for μ, we obtain

$$\mu = rT - \ln(m(c)).$$

Provided that we can generate from the cumulative distribution function of X, the price of a call option with strike price K under this returns distribution can be estimated from N simulations by the average discounted return from N options,

$$\begin{aligned} e^{-rT}\frac{1}{N}\sum_{i=1}^{N}(S_{Ti}-K)^{+} &= e^{-rT}\frac{1}{N}\sum_{i=1}^{N}(S_0e^{\mu+cX_i}-K)^{+} \\ &= e^{-rT}\frac{1}{N}\sum_{i=1}^{N}(S_0e^{rT-\ln(m(c))+cX_i}-K)^{+} \\ &= \frac{1}{N}\sum_{i=1}^{N}(S_0\frac{e^{cX_i}}{m(c)}-e^{-rT}K)^{+} \end{aligned}$$

A more precise calculation is the difference between the option price in this case and the comparable case of normally distributed returns. Suppose we use inverse transform together with a uniform[0,1] variate to generate both the random variable $X_i = F^{-1}(U_i)$ and the corresponding normal return $Z_i = rT + \sigma\sqrt{T}\Phi^{-1}(U_i)$. Then the difference is estimated by

$$\begin{aligned} &\text{option price under } F - \text{option price under } \Phi \\ &\simeq \frac{1}{N}\sum_{i=1}^{N}\{(S_0\frac{e^{cF^{-1}(U_i)}}{m(c)}-e^{-rT}K)^{+}-(S_0e^{\sigma\sqrt{T}\Phi^{-1}(U_i)-\sigma^2T/2}-e^{-rT}K)^{+}\} \end{aligned}$$

If necessary, in case the moment generating function of X is unknown, we can estimate it and the variance of X using sample analogues over a large number N of simulations. In this case c is estimated by

$$\sqrt{\frac{\sigma^2T}{\widehat{var}(X)}}$$

with $\widehat{var}$ representing the sample variance and $m(c)$ estimated by

$$\frac{1}{N}\sum_{i=1}^{N}e^{cF^{-1}(U_i)}.$$

To consider a specific example, the logistic$(0, 0.522)$ distribution is close to the normal, except with slightly more weight in the tails. The scale parameter in

this case was chosen so that the logistic has approximate unit variance. The cumulative distribution function is $F(x) = \frac{1}{1+exp\{-x/b\}}$ and its inverse is $X = b\ln(U/(1-U))$. The moment generating function is $m(s) = \Gamma(1-bs)\Gamma(1+bs)$, $s < 1/b$. The following function was used to compare the price of a call option when stock returns have the logistic distribution(i.e. stock prices have the "loglogistic" distribution) with the prices in the Black-Scholes model.

```
function [re,op1,opbs]=diffoptionprice(n,So,strike,r,sigma,T)
%estimates the relative error in the BS option price and price under
% logistic returns distribution . Runs n simulations.
u=rand(1,n);
x=log(u./(1-u));                                    % generates standard
logistic*
z=sigma*sqrt(T)*norminv(u)-sigma^2*T/2;
c=sigma*sqrt(T/var(x));
mc=mean(exp(c*x));
re=[]; op1=[]; opbs=[];
for i=1:length(strike)
op1=[op1 mean(max(exp(c*x)*So/mc-exp(-r*T)*strike(i),0))];  % price under F
opbs=[opbs mean(max(So*exp(z)-exp(-r*T)*strike(i),0))];      % price under BS
end
dif=op1-opbs;
re=[re dif./(dif+BLSPRICE(So,strike,r,T,sigma,0))];
plot(strike/So,re)
xlabel('Strike price/initial price')
ylabel('relative error in Black Scholes formula')
```

The relative error in the Black-Scholes formula obtained from a simulation of 100,000 is graphed in Figure 3.14. The logistic distribution differs only slightly

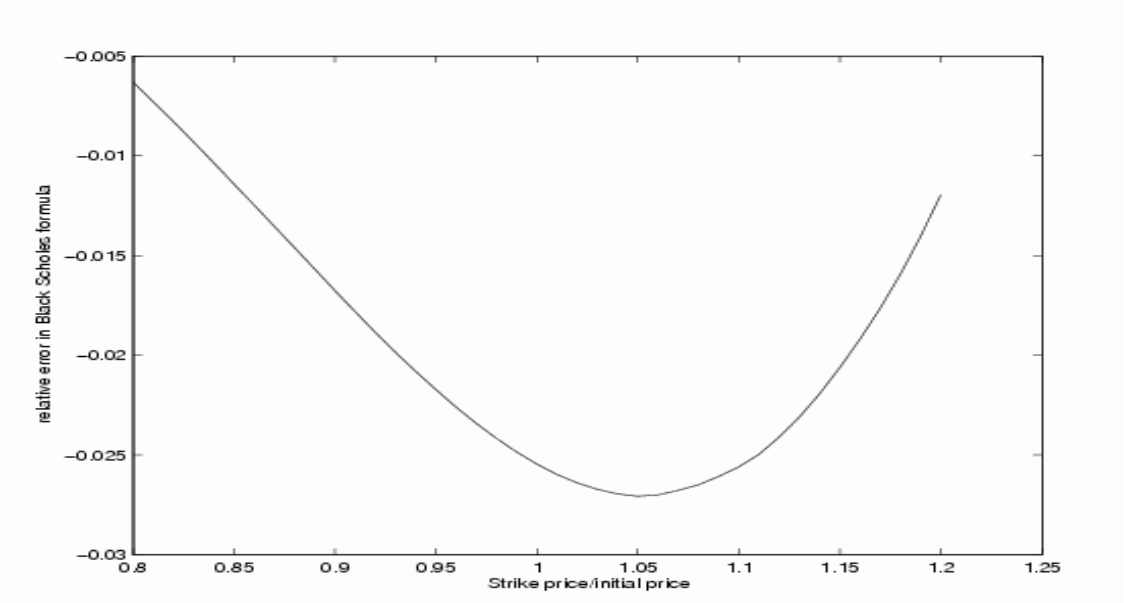

Figure 3.14: Relative Error in Black-Scholes price when asset prices are loglogistic, $\sigma = .4$, $T = .75$, $r = .05$

from the standard normal, and the primary difference is in the larger kurtosis or weight in the tails. Indeed virtually any large financial data set will differ from the normal in this fashion; there may be some skewness in the distribution but there is often substantial kurtosis. How much difference does this slightly increased weight in the tails make in the price of an option? Note that the Black-Scholes formula overprices all of the options considered by up to around 3%. The differences are quite small, however and there seems to be considerable robustness to the Black-Scholes formula at least for this type of departure in the distribution of stock prices.

A change in the single line x=log(u./(1-u)) in the above function permits revising the returns distribution to another alternative. For example we might

choose the double exponential or Laplace density

$$f(x) = \frac{1}{2}\exp(-|x|)$$

for returns, by replacing this line by $x = (u < .5)\log(2 * u) - (u > .5)\log(2 * (1 - u))$. The resulting Figure 3.15 shows a similar behaviour but more substantial pricing error, in this case nearly 10% for an at-the-money option.

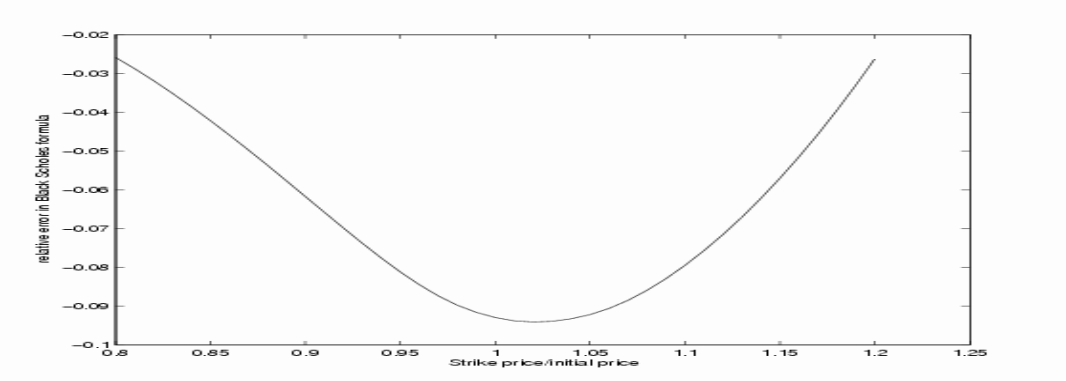

Figure 3.15: Relative pricing error in Black Scholes formula when returns follow the Laplace distribution

Another possible distribution of stock returns which can be used to introduce some skewness to the returns distribution is the loggamma or extreme value distribution whose probability density function takes the form

$$f(x) = \frac{1}{\Gamma(a)}\exp\{-e^{(x-c)} + (x-c)a\}, -\infty < x < \infty.$$

We can generate such a distribution as follows. Suppose Y is a random variable with gamma(a, e^c) distribution and probability density function

$$g(y) = \frac{y^{a-1}e^{-ca}}{\Gamma(a)}e^{-ye^{-c}}.$$

and define $X = \ln(Y)$. Then X has probability density function

$$f(x) = g(e^x)|\frac{d}{dx}e^x| = \frac{1}{\Gamma(a)}\exp\{(x(a-1) - ca - e^{x-c}\}e^x$$
$$= \frac{1}{\Gamma(a)}\exp\{-e^{x-c} + (x-c)a\}, -\infty < x < \infty.$$

As an example in Figure 3.16 we plot this density in the case $a = 2, c = 0$ This distribution is negatively skewed, a typical characteristic of risk-neutral distributions of returns. The large left tail in the risk-neutral distribution of returns reflects the fact that investors have an aversion to large losses and consequently the risk-neutral distribution inflates the left tail.

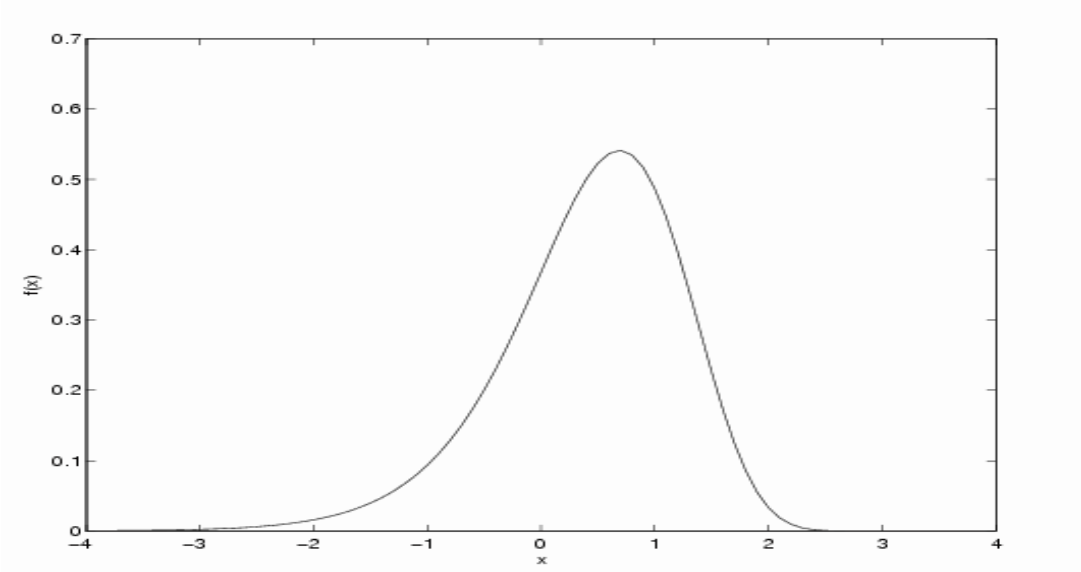

Figure 3.16: The probability density function e^{-e^x+2x}

Introducing a scale parameter ν, the probability density function of $\nu\ln(Y) = \ln(Y^\nu)$ where Y has a Gamma(2,1) distribution is

$$f(x) = \nu e^{-e^{(\nu x-c)}+2(\nu x-c)}.$$

The mean is approximately 0 and variance approximately σ^2 when we choose $c = -.42278$ and $\nu = .80308/\sigma$ and so this distribution is analogous to the

standard normal. However, the skewness is -0.78 and this negative skewness is more typical of risk neutral distributions of stock returns. We might ask whether the Black-Scholes formula is as robust to the introduction of skewness in the returns distribution as to the somewhat heavier tails of the logistic distribution. For comparison with the Black-Scholes model we permitted adding a constant and multiplying the returns by a constant which, in this case, is equivalent to assuming under the risk neutral distribution that

$$S_T = S_0 e^{\alpha} Y^{\nu}, Y \text{ is Gamma(2,1)}$$

where the constants α and ν are chosen so that the martingale condition is satisfied and the variance of returns matches that in the lognormal case. With some integration we can show that this results in the equations

$$\alpha = -\ln(E(Y^{\nu})) = -\ln(\Gamma(2+\nu))$$
$$\nu^2 var(\ln(Y)) = \nu^2 \psi'(2) = \sigma^2 T$$

where $\psi'(\alpha)$ is the *trigamma function* defined as the second derivative of $\ln(\Gamma(\alpha))$, and evaluated fairly easily using the series $\psi'(\alpha) = \sum_{k=0}^{\infty} \frac{1}{(k+\alpha)^2}$. For the special cases required here, $\psi'(2) \approx .6449$ so $\nu \approx \sigma\sqrt{T}/.8031$ and $\alpha = -\log(\Gamma(2+\sigma\sqrt{T}/.8031))$. Once again replacing the one line marked with a * in the function diffoptionprice by x=log(gaminf(u,2,1); permits determining the relative error in the Black-Scholes formula. There is a more significant pricing error in the Black-Scholes formula now, more typical of the relative pricing error that is observed in practice. Although the graph can be shifted and tilted somewhat by choosing different variance parameters, the shape appears to be a consequence of assuming a symmetric normal distribution for returns when the actual risk-neutral distribution is skewed. It should be noted that the practice of obtaining implied volatility parameters from options with similar strike prices and maturities is a partial, though not a compete, remedy to the substantial pricing errors caused by using a formula derived from a frequently ill-fitting

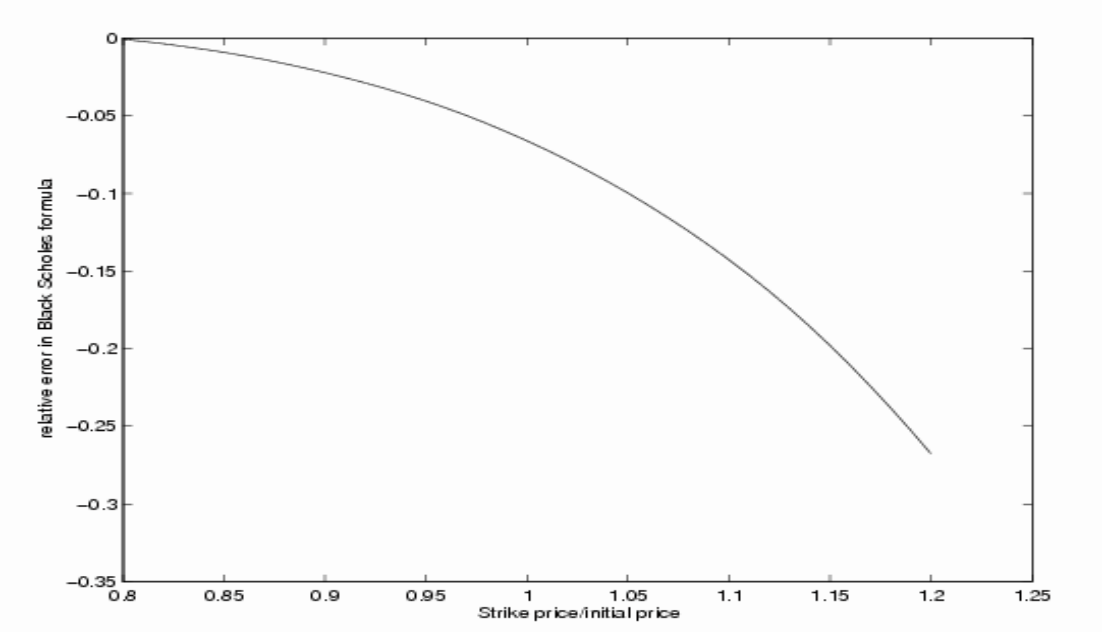

Figure 3.17: Relative Error in Black-Scholes formula when Asset returns follow extreme value

Black_Scholes model.

The Symmetric Stable Laws

A final family of distributions of increasing importance in modelling is the *stable family* of distributions. The stable cumulative distribution functions F are such that if two random variables X_1 and X_2 are independent with cumulative distribution function $F(x)$ then so too does the sum $X_1 + X_2$ after a change in location and scale. More generally the cumulative distribution function F of independent random variables X_1, X_2 is said to be *stable* if for each pair of constants a and b, there exist constants c and m such that

$$\frac{a\,X_1 + b\,X_2 - m}{c}$$

has the same cumulative distribution function F. A stable random variable X is most easily characterized through its characteristic function

$$Ee^{iuX} = \begin{cases} \exp(iu\theta - |u|^\alpha c^\alpha (1 - i\beta(sign\ u)\tan\frac{\pi\alpha}{2}) & \text{for} \quad \alpha \neq 1 \\ \exp(iu\theta - |u|c(1 + i\beta(sign\ \mathrm{u})\ln|u|)\frac{2}{\pi}) & \text{if} \quad \alpha = 1 \end{cases}$$

where i is the complex number $i^2 = -1$, θ is a location parameter of the distribution, and c is a scale parameter. The parameter $0 < \alpha \leq 2$ is the index of the stable distribution and governs the tail behavior and $\beta \in [-1, 1]$ governs the skewness of the distribution. In the case $\beta = 0$, we obtain the symmetric stable family of distributions, all unimodal densities, symmetric about their mode, and roughly similar in shape to the normal or Cauchy distribution (both special cases). They are of considerable importance in finance as an alternative to the normal distribution, in part because they tend to fit observations better in the tail of the distribution than does the normal, and in part because they enjoy theoretical properties similar to those of the normal family: sums of independent stable random variables are stable. Unfortunately, this is a more complicated family of densities to work with; neither the density function nor the cumulative

distribution function can be expressed in a simple closed form. Both require a series expansion. The parameter $0 < \alpha \leq 2$ indicates what moments exist. Except in the special case $\alpha = 2$ (the normal distribution) or the case $\beta = -1$, moments of order less than α exist while moments of order α or more do not. This is easily seen because the tail behaviour is, when $\alpha < 2$,

$$\lim_{x\to\infty} x^\alpha P[X > x] = K_\alpha \frac{1+\beta}{2} c^\alpha$$
$$\lim_{x\to\infty} x^\alpha P[X < -x] = K_\alpha \frac{1-\beta}{2} c^\alpha$$

for constant K_α depending only on α. Of course, for the normal distribution, moments of all orders exist. The stable laws are useful for modelling in situations in which variates are thought to be approximately normalized sums of independent identically distributed random variables. To determine robustness against heavy-tailed departures from the normal distribution, tests and estimators can be computed with data simulated from a symmetric stable law with α near 2. The probability density function does not have a simple closed form except in the case $\alpha = 1$ (Cauchy) and $\alpha = 2$ (Normal) but can be expressed as a series expansion of the form

$$f_c(x) = \frac{1}{\pi\,\alpha\,c} \sum_{k=0}^{\infty} (-1)^k \frac{\Gamma\left(\frac{2k+1}{\alpha}\right)}{(2k)!} (\frac{x}{c})^k$$

where c is the scale parameter (and we have assumed the mode is at 0). Especially for large values of x, this probability density function converges extremely slowly. However, Small (2003) suggests using an Euler transformation to accelerate the convergence of this series, and this appears to provide enough of an improvement in the convergence to meet a region in which a similar tail formula (valid for large x) provides a good approximation. According to Chambers, Mallows and Stuck, (1976), when $1 < \alpha < 2$, such a variate can be generated as

$$X = c\sin(\alpha U) \left[\frac{\cos(U(1-\alpha))}{E}\right]^{\frac{1}{\alpha}-1} (\cos U)^{-1/\alpha} \tag{3.20}$$

where U is uniform $[-\pi/2, \pi/2]$ and E, standard exponential are independent. The case $\alpha = 1$ and $X = \tan(U)$ is the Cauchy. It is easy to see that the

Cauchy distribution can also be obtained by taking the ratio of two independent standard normal random variables and $\tan(U)$ may be replaced by Z_1/Z_2 for independent standard normal random variables Z_1, Z_2 produced by Marsaglia's polar algorithm. Equivalently, we generate $X = V_1/V_2$ where $V_i \sim U[-1,1]$ conditional on $V_1^2 + V_2^2 \leq 1$ to produce a standard Cauchy variate X.

Example: Stable random walk.

A stable random walk may be used to model a stock price but the closest analogy to the Black Scholes model would be a logstable process S_t under which the distribution of $\ln(S_t)$ has a symmetric stable distribution. Unfortunately, this specification renders impotent many of our tools of analysis, since except in the case $\alpha = 2$ or the case $\beta = -1$, such a stock price process S_t has no finite moments at all. Nevertheless, we may attempt to fit stable laws to the distribution of $\ln(S_t)$ for a variety of stocks and except in the extreme tails, symmetric stable laws with index $\alpha \simeq 1.7$ often provide a reasonably good fit. To see what such a returns process looks like, we generate a random walk with 10,000 time steps where each increment is distributed as independent stable random variables having parameter 1.7. The following *Matlab* function was used

```
function s=stabrnd(a,n)
u=(unifrnd(0,1,n,1)*pi)-.5*pi;
e = exprnd(1,n,1);
s=sin(a*u).*(cos((1-a)*u)./e).^(1/a-1).*(cos(u)).^(-1/a)
```

Then the command

```
plot(1:10000, cumsum(stabrnd(1.7,10000)));
```

resulted in the Figure 3.18. Note the occasional very large jump(s) which dominates the history of the process up to that point, typical of random walks generated from the stable distributions with $\alpha < 2$.

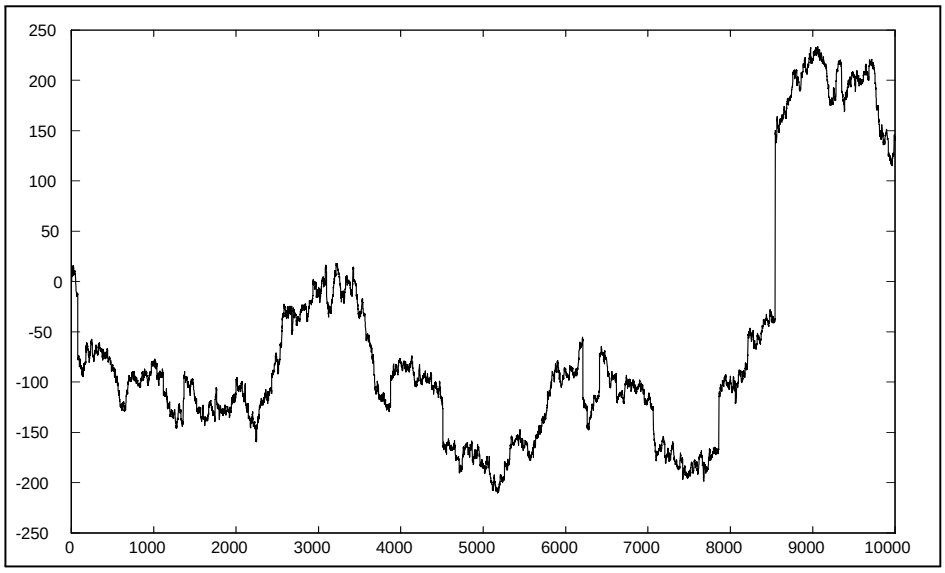

Figure 3.18: A Symmetric Stable Random Walk with index $\alpha = 1.7$

The Normal Inverse Gaussian Distribution

There is a very substantial body of literature that indicates that the normal distribution assumption for returns is a poor fit to data, in part because the observed area in the tails of the distribution is much greater than the normal distribution permits. One possible remedy is to assume an alternative distribution for these returns which, like the normal distribution, is infinitely divisible, but which has more area in the tails. A good fit to some stock and interest rate data has been achieved using the Normal Inverse Gaussian (NIG) distribution (see for example Prausse, 1999). To motivate this family of distributions, let us suppose that stock returns follow a Brownian motion process but with respect to a random time scale possibly dependent on volume traded and other external factors independent of the Brownian motion itself. After one day, say, the return on the stock is the value of the Brownian motion process at a random time, τ, independent of the Brownian motion. Assume that this random time

has the *Inverse Gaussian* distribution having probability density function

$$g(t) = \frac{\theta}{c\sqrt{2\pi t^3}} \exp\{-\frac{(\theta - t)^2}{2c^2 t}\} \tag{3.21}$$

for parameters $\theta > 0, c > 0$. This is the distribution of a first passage time for Brownian motion. In particular consider a Brownian motion process $B(t)$ having drift 1 and diffusion coefficient c. Such a process is the solution to the stochastic differential equation

$$dB(t) = dt + cdW(t), B(0) = 0.$$

Then the first passage of the Brownian motion to the level θ is $T = \inf(t; B(t) = \theta\}$ and this random variable has probability density function (3.21). The mean of such a random variable is θ and with variance θc^2. These can be obtained from the moment generating function of the distribution with probability density function (3.21),

$$g^*(s) = \exp\{-\theta(\frac{-1 + \sqrt{1 - 2sc}}{c^2})\}.$$

Expanding this locally around $c = 0$ we obtain

$$g^*(s) = \exp\{\theta s + \frac{1}{2}\theta s^2 c^2 + O(c^4)\}$$

and by comparing this with the moment generating function of the normal distribution, as $c \to 0$, the distribution of

$$\frac{T - \theta}{c\sqrt{\theta}}$$

approaches the standard normal or, more loosely, the distribution (3.21) approaches Normal$(\theta, \theta c^2)$.

Lemma 30 *Suppose $X(t)$ is a Brownian motion process with drift β and diffusion coefficient* 1*, hence satisfying*

$$dX_t = \beta dt + dW_t, \quad X(0) = \mu.$$

Suppose a random variable T has probability density function (3.21) and is independent of X_t. Then the probability density function of the randomly stopped Brownian motion process is given by

$$f(x;\alpha,\beta,\delta,\mu) = \frac{\alpha\delta}{\pi}\exp(\delta\sqrt{\alpha^2-\beta^2}+\beta(x-\mu))\frac{K_1(\alpha\sqrt{\delta^2+(x-\mu)^2})}{\sqrt{\delta^2+(x-\mu)^2}} \quad (3.22)$$

with

$$\delta = \frac{\theta}{c}, \quad \textit{and } \alpha = \sqrt{\beta^2+\frac{1}{c^2}}$$

and the function $K_\lambda(x)$ is the modified Bessel function of the second kind defined by

$$K_\lambda(x) = \frac{1}{2}\int_0^\infty y^{\lambda-1}\exp(-\frac{x}{2}(y+y^{-1}))dy, \textit{ for } x>0.$$

Proof. The distribution of the randomly stopped variable $X(T)$ is the same as that of the random variable

$$X = \mu + \beta T + \sqrt{T}Z$$

where Z is $N(0,1)$ independent of T. Conditional on the value of T the probability density function of X is

$$f(x|T) = \frac{1}{\sqrt{2\pi T}}\exp(-\frac{1}{2T}(x-\mu-\beta T)^2)$$

and so the unconditional distribution of X is given by

$$\begin{aligned}
&\int_0^\infty \frac{1}{\sqrt{2\pi t}}\exp(-\frac{1}{2t}(x-\mu-\beta t)^2)\frac{\theta}{c\sqrt{2\pi t^3}}\exp(-\frac{(\theta-t)^2}{2c^2t})dt \\
&= \frac{\theta}{2\pi c}\int_0^\infty t^{-2}\exp(-\frac{1}{2t}(x-\mu-\beta t)^2 - \frac{(\theta-t)^2}{2c^2t})dt \\
&= \frac{\theta}{2\pi c}\int_0^\infty t^{-2}\exp(-\frac{1}{2t}(x^2-2x\mu+\mu^2+\theta^2)+(\beta(x-\mu)+\frac{\theta}{c^2}) - \frac{t}{2}(\beta^2+\frac{1}{c^2}))dt \\
&= \frac{\theta}{2\pi c}\exp(\beta(x-\mu)+\frac{\theta}{c^2})\int_0^\infty t^{-2}\exp(-\frac{1}{2t}((x-\mu)^2+\theta^2) - \frac{t}{2}(\beta^2+\frac{1}{c^2}))dt \\
&= \frac{\alpha\delta}{\pi}\exp(\delta\sqrt{\alpha^2-\beta^2}+\beta(x-\mu))\frac{K_1(\alpha\sqrt{\delta^2+(x-\mu)^2})}{\sqrt{\delta^2+(x-\mu)^2}}.
\end{aligned}$$

■

The modified Bessel function of the second kind $K_\lambda(x)$ is given in MATLAB by *besselk*(ν, x) and in **R** by besselK(x,ν,expon.scaled=FALSE). The distribution with probability density function given by (3.22) is called the *normal inverse Gaussian* distribution with real-valued parameters $x, \mu,\ 0 \leq \delta$ and $\alpha \geq |\beta|$. The tails of the normal inverse Gaussian density are substantially heavier than those of the normal distribution. In fact up to a constant

$$f(x; \alpha, \beta, \delta, \mu) \sim |x|^{-3/2} \exp((\mp\alpha + \beta)x) \text{ as } x \to \pm\infty.$$

The moments of this distribution can be obtained from the moment generating function

$$M(s) = e^{\mu s} \left[\frac{\alpha^2 - (\beta + s)^2}{\alpha^2 - \beta^2}\right]^{1/4} \exp\{\delta(\alpha^2 - \beta^2)^{1/2} - \delta(\alpha^2 - (s+\beta)^2)^{1/2}\} \text{ for } |\beta + s| < \alpha. \tag{3.23}$$

These moments are:

$$E(X) = \mu + \delta\beta(\alpha^2 - \beta^2)^{-1/2}$$
$$var(X) = \delta\alpha^2(\alpha^2 - \beta^2)^{-3/2}$$

and the skewness and kurtosis:

$$\text{skew} = 3\beta\alpha^{-1}\delta^{-1/2}(\alpha^2 - \beta^2)^{-1/4}$$
$$\text{kurtosis} = 3\delta^{-1}\alpha^{-2}(\alpha^2 + 4\beta^2)(\alpha^2 - \beta^2)^{-1/2}.$$

One of the particularly attractive features of this family of distributions, shared by the normal and the stable family of distributions, is that it is closed under convolutions. This is apparent from the moment generating function (3.23) since

$$M^N(s)$$

gives a moment generating function of the same form but with μ replaced by μN and δ by δN. In Figure 3.19 we plot the probability density function of a member of this family of distributions.

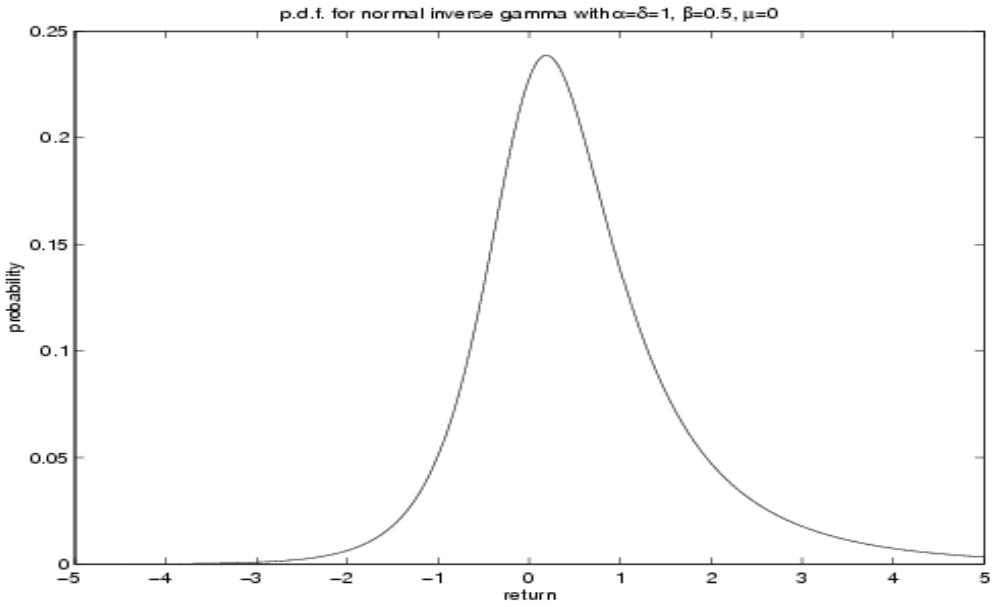

Figure 3.19: Normal Inverse Gaussian probability density function with $\alpha = \delta = 1$, $\beta = \frac{1}{2}, \mu = 0$

Note the similarity to the normal density but with a modest amount of skewness and increased weight in the tails. We can generate random variables from this distribution as follows:

Sample T from an inverse Gaussian distribution (3.21)

Return $X = \mu + \beta T + N(0, T)$

where $N(0, T)$ is a normal random variable with mean 0 and variance T.

We sample from the inverse Gaussian by using a property of the distribution that if T has density of the form (3.21) then

$$\frac{(T-\theta)^2}{c^2 T} \tag{3.24}$$

has a chi-squared distribution with one degree of freedom (easily generated as the square of a standard normal random variable). The algorithm is (see Michael, Shucany, Hass (1976));

1. For
$$c = \frac{1}{\sqrt{\alpha^2 - \beta^2}}, \text{ and } \theta = \delta c,$$
generate G_1 from the Gamma$(\frac{1}{2}, \frac{c}{\delta})$ distribution. Define
$$Y_1 = 1 + G_1(1 - \sqrt{1 + \frac{2}{G_1}})\}.$$

2. Generate $U_2 \sim U[0,1]$. If $U_2 \leq \frac{1}{1+Y_1}$ then output $T = \theta Y_1$

3. Otherwise output $T = \theta Y_1^{-1}$.

The two values θY_1, and θY_1^{-1} are the two roots of the equation obtained by setting (3.24) equal to a chi-squared variate with one degree of freedom and the relative values of the probability density function at these two roots are $\frac{1}{1+Y_1}$ and $1 - \frac{1}{1+Y_1}$.

Finally to generate from the normal inverse Gaussian distribution (3.22) we generate an inverse gamma random variable above and then set $X = \mu + \beta T +$

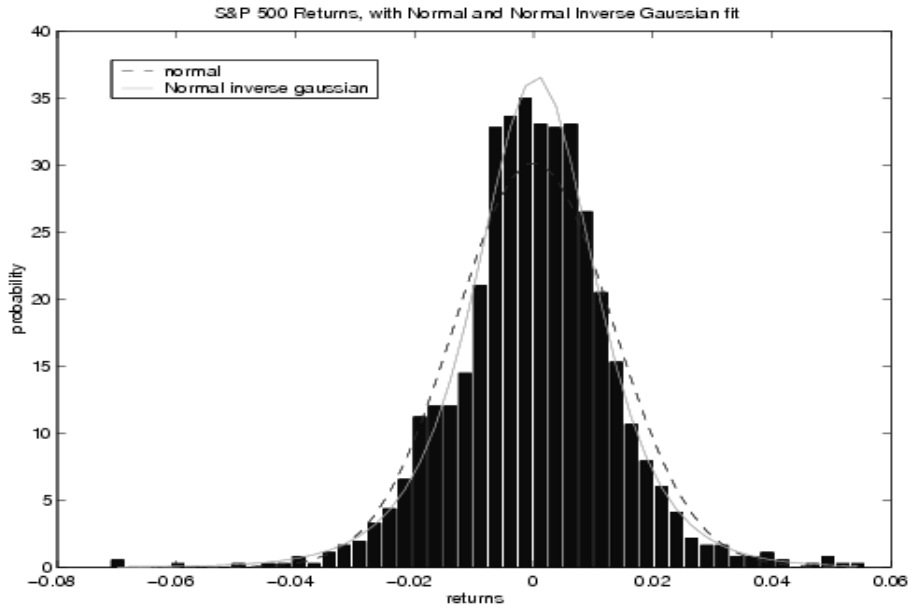

Figure 3.20: The Normal and the Normal inverse Gaussian fit to the S&P500 Returns

$N(0,T)$. Prause (1999) provides a statistical evidence that the Normal Inverse Gaussian provides a better fit than does the normal itself. For example we fit the normal inverse gamma distribution to the S&P500 index returns over the period Jan 1, 1997-Sept 27, 2002. There were a total of 1442 values over this period. Figure 3.20 shows a histogram of the daily returns together with the normal and the NIG fit to the data. The mean return over this period is 8×10^{-5} and the standard deviation of returns 0.013. If we fit the normal inverse Gaussian distribution to these returns we obtain parameter estimates

$$\alpha = 95.23, \beta = -4.72, \delta = 0.016, \mu = 0.0009$$

and the Q-Q plots in Figure 3.21 . Both

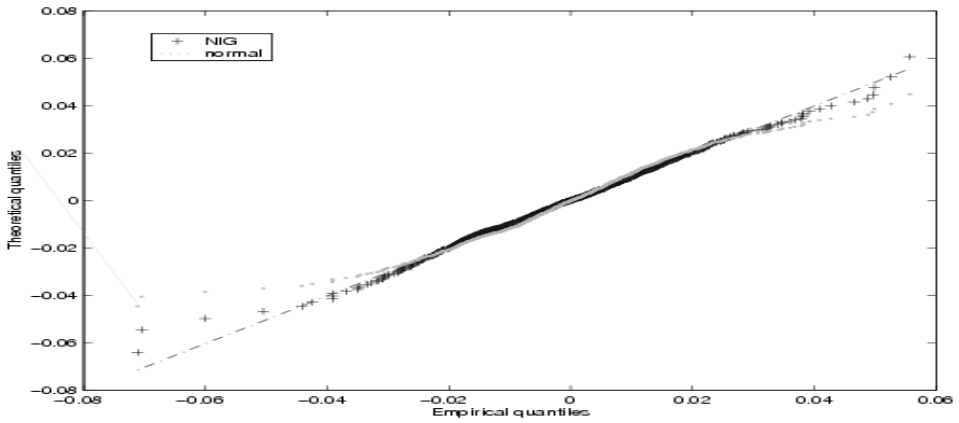

Figure 3.21: QQ plots showing the Normal Inverse Gaussian and the Normal fit to S&P 500 data, 1997-2002

indicate that the normal approximation fails to properly fit the tails of the distribution but that the NIG distribution is a much better fit. This is similar to the conclusion in Prause using observations on the Dow Jones Industrial Index.

Generating Random Numbers from Discrete Distributions

Many of the methods described above such as inversion and acceptance-rejection for generating continuous distributions work as well for discrete random variables. Suppose for example X is a discrete distribution taking values on the integers with probability function $P[X = x] = f(x)$, for $x = 0, 1, 2, ...$ Suppose we can find a continuous random variable Y which has exactly the same value of its cumulative distribution function at these integers so that $F_Y(j) = F_X(j)$ for all $j = 1, 2, ...$ Then we may generate the continuous random variable Y, say by inversion or acceptance-rejection and then set $X = \lfloor Y \rfloor$ the integer part

of Y. Clearly X takes integer values and since $P[X \leq j] = P[Y \leq j] = F_X(j)$ for all $j = 0, 1, ...,$ then X has the desired distribution. The continuous exponential distribution and the geometric distribution are linked in this way. If X has a geometric(p) distribution and Y has the exponential distribution with parameter $\lambda = -\ln(1-p)$, then X has the same distribution as $\lceil Y \rceil$ or $\lfloor Y \rfloor + 1$.

Using the inverse transform method for generating discrete random variables is usually feasible but for random variables with a wide range of values of reasonably high probability, it often requires some setup costs to achieve reasonable efficiency. For example if X has cumulative distribution function $F(x), x = 0, 1, ...$inversion requires that we output an integer $X = F^{-1}(U)$,an integer X satisfying $F(X-1) < U \leq F(X)$. The most obvious technique for finding such a value of X is to search sequentially through the potential values $x = 0, 1, 2, ...$ Figure 3.22 is the search tree for inversion for the distribution on the integers $0, \ldots 4$ given by

x	0	1	2	3	4
$f(x)$	0.11	0.30	0.25	0.21	0.13

We generate an integer by repeatedly comparing a uniform [0,1] variate U with the value at each node, taking the right branch if it is greater than this threshold value, the left if it is smaller. If X takes positive integer values $\{1, 2, ..., N\}$,the number of values searched will average to $E(X)$ which for many discrete distributions can be unacceptably large.

An easy alternative is to begin the search at a value m which is near the median (or mode or mean) of the distribution. For example we choose $m = 2$ and search to the left or right depending on the value of U in Figure 3.23.

If we assume for example that we root the tree at m then this results in searching roughly an average of $E[|X - m + 1|]$ before obtaining the generated variable. This is often substantially smaller than $E(X)$ especially when $E(X)$ is large but still unacceptably large when the distribution has large variance.

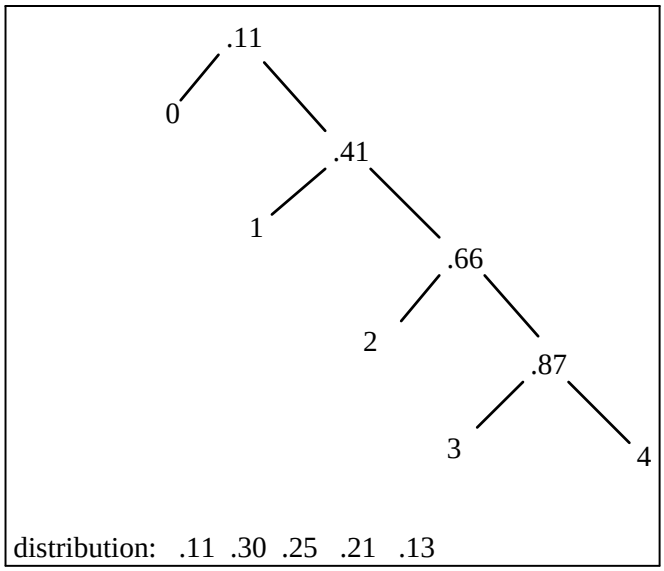

Figure 3.22: Sequential Search tree for Inverse Transform with root at $x = 0$

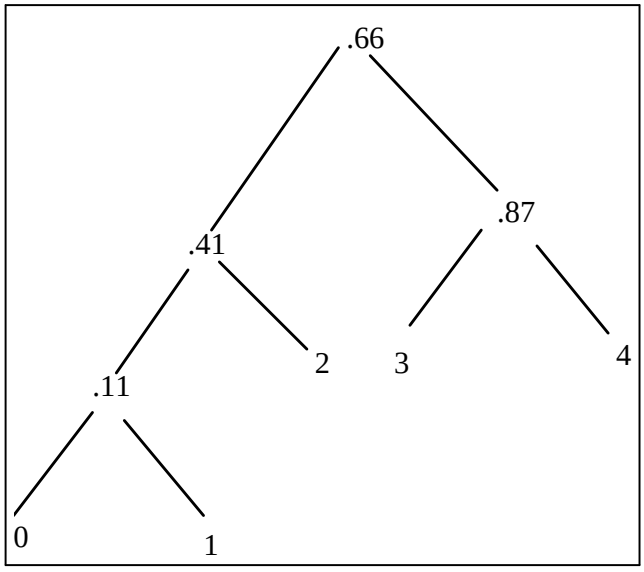

Figure 3.23: Search tree rooted near the median

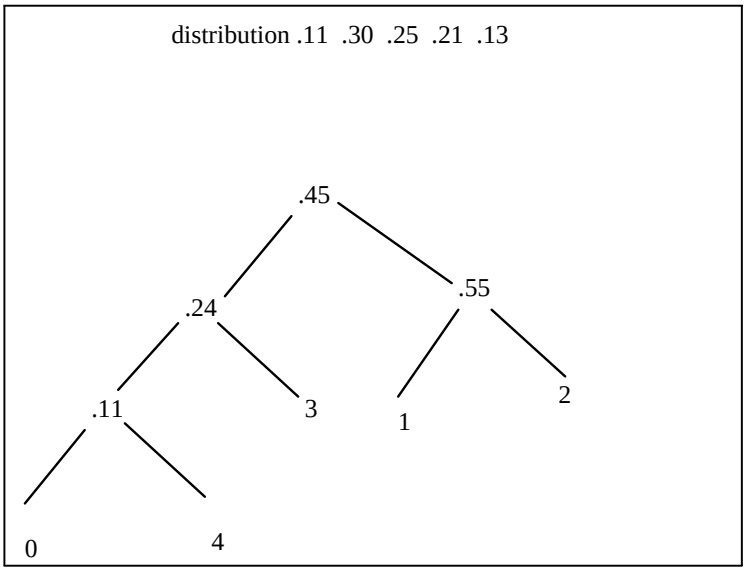

Figure 3.24: Optimal Binary Search Tree

An optimal binary search tree for this distribution is graphed in Figure 3.24. This tree has been constructed from the bottom up as follows. We begin by joining the two smallest probabilities $f(4)$ and $f(0)$ to form a new node with weight $f(0) + f(4) = 0.24$. Since we take the left path (towards $X = 0$ rather than towards $X = 4$) if U is smaller than the value .11 labelling the node at the intersection of these two branches. We now regard this pair of values as a unit and continue to work up the tree from the leaves to the root. The next smallest pair of probabilities are $\{0, 1\}$ and $\{3\}$ which have probabilities 0.24 and 0.21 respectively so these are the next to be joined hence working from the leaves to the root of the tree. This optimal binary search tree provides the minimum expected number of comparisons and is equivalent to sorting the values in order of largest to smallest probability, in this case $1, 2, 3, 4, 0$, relabelling them or coding them $\{0, 1, 2, 3, 4\}$ and then applying the inverse transform method starting at 0.

The leaves of the tree are the individual probabilities $f(j)$ and the internal

nodes are sums of the weights or probabilities of the "children", the values $f(j)$ for j on paths below this node. Let D_i represents the depth of the $i'th$ leaf so for example the depth of leaf 0 in Figure 3.24 is $D_0 = 3$. Then the average number of comparisons to generate a single random variable Xstarting at the root is $\sum_i f(i)D_i$. The procedure for constructing the last tree provides an optimal algorithm in the sense that this quantity is minimized. It is possible to show that an optimal binary search tree will reduce the average number of comparisons from $E(X)$ for ordinary inversion to less than $1 + 4\,[log_2(1 + E(X))]$.

Another general method for producing variates from a discrete distribution was suggested by Walker (1974, 1977) and is called the *alias method*. This is based on the fact that *every discrete distribution is a uniform mixture of two-point distributions.* Apart from the time required to set up an initial table of aliases and aliasing probabilities, the time required to generate values from a discrete distribution with K supporting points is bounded in K, whereas methods such as inverse transform have computational time which increase proportionally with $E(X)$.

Consider a discrete distribution of the form with probability function $f(j)$ on K integers $j = 1, 2, ...K$. We seek a table of values of $A(i)$ and associated "alias" probabilities $q(i)$ so that the desired discrete random variable can be generated in two steps, first generate one of the integers $\{1, 2, ..., K\}$ at random and uniformly, then if we generated the value I, say, replace it by an "alias" value $A(I)$ with alias probability $q(I)$. These values $A(I)$ and $q(I)$ are determined below. The algorithm is:

GENERATE I UNIFORM ON $\{1, ...K\}$.

WITH PROBABILITY $q(I)$, OUTPUT $X = I$, OTHERWISE, $X = A(I)$.

An algorithm for producing these values of $(A(i), q(i)), i = 1, ..., K\}$ is suggested by Walker(1977) and proceeds by reducing the number of non-zero probabilities one at a time.

1. Put $q(i) = Kf(i)$ for all $i = 1, ..., K$.

2. LET m be the index so that $q(m) = \min\{q(i); q(i) > 0\}$ and let $q(M) = \max\{q(i); q(i) > 0\}$.

3. SET $A(m) = M$ and fix $q(m)$ (it is no longer is subject to change).

4. Replace $q(M)$ by $q(M) - (1 - q(m))$

5. Replace $(q(1), ...q(K))$by $(q(1), ..., q(m-1), q(m+1), .q(M)$ (so the component with index m is removed).

6. Return to 2 unless all remaining $q_i = 1$ or the vector of q_i's is empty.

Note that on each iteration of the steps above, we fix one of components $q(m)$ and remove it from the vector and adjust one other, namely $q(M)$. Since we always fix the smallest $q(m)$ and since the average $q(i)$ is one, we always obtain a probability, i.e. fix a value $0 < q(m) \leq 1$. Figure 3.25 shows the way in which this algorithm proceeds for the distribution

$x =$	1	2	3	4
$f(x) =$	.1	.2	.3	.4

We begin with $q(i) = 4 \times f(i) = .4, .8, 1.2, 1.6$ for $i = 1, 2, 3, 4$. Then since $m = 1$ and $M = 4$ these are the first to be adjusted. We assign $A(1) = 4$ and $q(1) = 0.4$. Now since we have reassigned mass $1 - q(1)$ to $M = 4$ we replace $q(4)$ by $1.6 - (1 - 0.4) = 1$. We now fix and remove $q(1)$ and continue with $q(i) = .8, 1.2, 1.0$ for $i = 2, 3, 4$. The next step results in fixing $q(2) = 0.8$, $A(2) = 3$ and changing $q(3)$ to $q(3) - (1 - q(2)) = 1$. After this iteration, the remaining $q(3)$, $q(4)$ are both equal to 1, so according to step 6 we may terminate the algorithm. Notice that we terminated without assigning a value to $A(3)$ and $A(4)$. This assignment is unnecessary since the probability the alias $A(i)$ is used is $(1 - q(i))$ which is zero in these two cases. The algorithm therefore results in aliases $A(i) = 4, 3, i = 1, 2$ and $q(i) = .4, .8, 1, 1$, respectively for $i = 1, 2, 3, 4$. Geometrically, this method iteratively adjusts a probability histogram to form a rectangle with base K as in Figure 3.25.

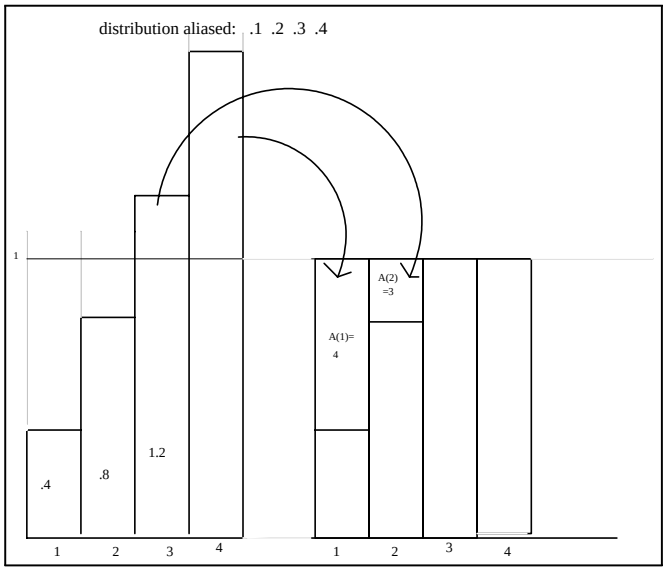

Figure 3.25: The alias method for generating from the distribution 0.1 0.2 0.3 0.4

Suppose I now wish to generate random variables from this discrete distribution. We simply generate a random variable uniform on the set $\{1, 2, 3, 4\}$ and if 1 is selected, we replace it by $A(1) = 4$ with probability $1 - q(1) = 0.6$. If 2 is selected it is replaced by $A(2) = 3$ with probability $1 - q(2) = 0.2$.

Acceptance-Rejection for Discrete Random Variables

The acceptance-rejection algorithm can be used both for generating discrete and continuous random variables and the geometric interpretation in both cases is essentially the same. Suppose for example we wish to generate a discrete random variable X having probability function $f(x)$ using as a dominating function a multiple of $g(x)$ the probability density function of a *continuous* random variable. Take for example the probability function

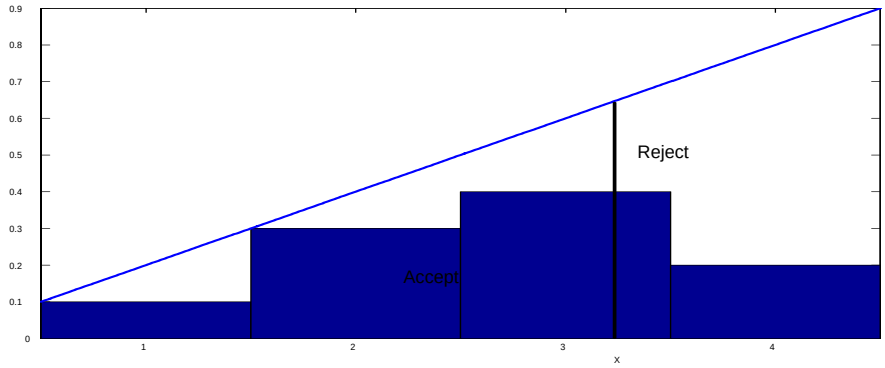

Figure 3.26: Acceptance-Rejection with for Discrete Distribution with continuous dominating function.

$x =$	1	2	3	4
$f(x) =$	.1	.3	.4	.2

using the dominating function $2g(x) = 0.1 + 0.2(x - 0.5)$ for $0.5 < x < 4.5$. It is easy to generate a continuous random variable from the probability density function $g(x)$ by inverse transform. Suppose we generate the value X. Then if this value is under the probability histogram graphed in Figure 3.26 we accept the value (after rounding it to the nearest integer to conform the discreteness of the output distribution) and otherwise we reject and repeat.

We may also dominate a discrete distribution with another discrete distribution in which case the algorithm proceeds as in the continuous case but with the probability density functions replaced by probability functions.

The Poisson Distribution.

Consider the probability function for a *Poisson distribution with parameter* λ

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!}, x = 0, 1, ... \tag{3.25}$$

The simplest generator is to use the Poisson process. Recall that a Poisson process with rate 1 on the real line can be described in two equivalent ways:

1. Points are distributed on the line in such a way that the spacings between consecutive points are independent exponential(λ) random variables. Then the resulting process is a Poisson process with rate λ.

2. The number of points in an interval of length h has a Poisson (λh) distribution. Moreover the numbers of points in non-overlapping intervals are independent random variables.

The simplest generator stems from this equivalence. Suppose we use the first specification to construct a Poisson process with rate parameter 1 and then examine X = the number of points occurring in the interval $[0, \lambda]$. This is the number of partial sums of exponential(1) random variables that are less than or equal to λ

$$X = \inf\{n; \sum_{i=1}^{n+1}(-lnU_i) > \lambda\}$$

or equivalently

$$X = \inf\{n; \prod_{i=1}^{n+1} U_i < e^{-\lambda}\} \tag{3.26}$$

This generator requires CPU time which grows linearly with λ since the number of exponential random variables generated and summed grows linearly with λ and so an alternative for large λ is required. Various possibilities of acceptance-rejection algorithms have been suggested including dominating the Poisson probability function with multiples of the logistic probability density function (Atkinson (1979)), the normal density with exponential right tail (cf. Devroye, lemma 3.8, page 509). A simple all-purpose dominating function is the so-called table-mountain function (cf. Stadlober (1989)), essentially a function with a flat top and tails that decrease as $1/x^2$. Another simple alternative for generating Poisson variates that is less efficient but simpler to implement is to use the *Lorentzian*, or truncated Cauchy distribution with probability density function

$$g(x|a, b) = \frac{c_0}{b^2 + (x - a)^2}, x > 0 \tag{3.27}$$

where c_0 is the normalizing constant. A random variable is generated from this distribution using the inverse transform method; $X = a + b\tan(\pi U),$, where $U \sim U[0,1]$. Provided that we match the modes of the distribution $a = \lambda$ and put $b = \sqrt{2\lambda}$, this function may be used to dominate the Poisson distribution and provide a simple rejection generator. The *Matlab* Poisson random number generator is $poissrnd(\lambda, m, n)$ which generates an $m \times n$ matrix of Poisson(λ) variables. This uses the simple generator (3.26) and is not computationally efficient for large values of λ.In **R** the command *rpois(n,*λ) generates a vector of n Poisson variates.

The Binomial Distribution

For the *Binomial* distribution, we may use any one of the following alternatives:

(1) $X = \sum_{i=1}^{n} I(U_i < p), U_i \sim$ independent uniform$[0,1]$

(2) $X = \inf\{x; \sum_{i=1}^{x+1} G_i > n\}$, where $G_i \sim$independent Geometric(p)

(3) $X = \inf\{x; \sum_{i=1}^{x+1} \frac{E_i}{n-i+1} > -log(1-p)\}$, where $E_i \sim$independent Exponential(1).

Method (1) obtains from the definition of the sum of independent Bernoulli random variables since each of the random variables $I(U_i < p)$ are independent, have values 0 and 1 with probabilities $1 - p$ and p respectively. The event $(U_i < p)$ having probability p is typically referred to as a "success". Obviously this method will be slow if n is large. For method (2), recall that the number of trials necessary to obtain the first success, G_1, say, has a geometric distribution. Similarly, G_2 represents the number of additional trials to obtain the second success. So if $X = j$, the number of trials required to obtain $j + 1$ successes was greater than n and to obtain j successes, less than or equal to n. In other words there were exactly j successes in the first n trials. When n is large but np fairly small, method (2) is more efficient since it is proportional to the number os successes rather than the total number of trials. Of course for large n and

np sufficiently small (e.g. <1), we can also replace the Binomial distribution by its Poisson ($\lambda = np$) approximation. Method (3) is clearly more efficient if $-log(1-p)$ is not too large so that p is not too close to 1, because in this case we need to add fewer exponential random variables.

For large mean np and small $n(1-p)$ we can simply reverse the role of successes and failures and use method (2) or (3) above. But if both np and $n(1-p)$ are large, a rejection method is required. Again we may use rejection beginning with a Lorentzian distribution, choosing $a = np$, and $b = \sqrt{2np(1-p)}$ in the case $p < 1/2$. When $p > 1/2$, we simply reverse the roles of "failures" and "successes". Alternatively, a dominating table-mountain function may be used (Stadlober (1989)). The binomial generator in *Matlab* is the function *binornd(n,p,j,k)* which generates an $n \times k$ matrix of binomial(n,p) random variables. This uses the simplest form (1) of the binomial generator and is not computationally efficient for large n. In **R**, *rbinom(m,n,p)* will generate a vector of length m of Binomial(n,p) variates.

Random Samples Associated with Markov Chains

Consider a finite state *Markov Chain,* a sequence of (discrete) random variables $X_1, X_2, \ldots$each of which takes integer values $1, 2, \ldots N$ (called *states*). The number of states of a Markov chain may be large or even infinite and it is not always convenient to label them with the positive integers and so it is common to define the *state space* as the set of all possible states of a Markov chain, but we will give some examples of this later. For the present we restrict attention to the case of a finite state space. The *transition probability matrix* is a matrix P describing the conditional probability of moving between possible states of the

chain, so that

$$P[X_{n+1} = j | X_n = i] = P_{ij}, i = 1, \ldots N, j = 1, \ldots N.$$

where $P_{ij} \geq 0$ for all i, j and $\sum_j P_{ij} = 1$ for all i. A *limiting distribution* of a Markov chain is a vector ($\underline{\pi}$ say) of long run probabilities of the individual states with the property that

$$\pi_i = lim_{t \to \infty} P[X_t = i].$$

A *stationary distribution* of a Markov chain is the column vector ($\underline{\pi}$ say) of probabilities of the individual states such that

$$\underline{\pi}' P = \underline{\pi}'. \tag{3.28}$$

$\underline{\pi}' P = \underline{\pi}'$. For a Markov chain, every limiting distribution is in fact a stationary distribution. For the basic theory of Markov Chains, see the Appendix. Roughly, a Markov chain which eventually "forgets" the states that were occupied in the distant path, in other words for which the probability of the current states does not vary much as we condition on different states in the distant past, is called ergodic. A Markov chain which simply cycles through three states $1 \to 2 \to 3 \to 1 \to ...$ is an example of a periodic chain, and is not ergodic.

It is often the case that we wish to simulate from a finite ergodic Markov chain when it has reached equilibrium or stationarity, which is equivalent to sampling from the distribution of X_n assuming that the distribution of X_0 is given by the stationary distribution $\underline{\pi}$. In a few cases, we can obtain this stationary distribution directly from (3.28) but when N is large this system of equations is usually not feasible to solve and we need to find another way to sample from the probability vector π. Of course we can always begin the Markov chain in some arbitrary initial state and run it waiting for Hele to freeze over (it does happen since Helle is in Devon) until we are quite sure that the chain has

essentially reachedequilibrium, and then use a subsequent portion of this chain, discarding this initial period, sometimes referred to as the "initial transient".

Clearly this is often not a very efficient method, particularly in cases in which the chain mixes or forgets its past very slowly for in this case the required initial transient is long. On the other hand if we shortened it, we run the risk of introducing bias into our simulations because the distribution generated is too far from the equilibrium distribution π. There are a number of solutions to this problem proposed in a burgeoning literature. Here we limit ourselves to a few of the simpler methods.

Metropolis-Hastings Algorithm

The Metropolis-Hastings Algorithm is a method for generating random variables from a distribution π that applies even in the case of an infinite number of states or a continuous distribution π. It is assumed that π is known up to some multiplicative constant. Roughly, the method consists of using a convenient "proposal" Markov chain with transition matrix Q to generate transitions, but then only "accept" the move to these new states with probability that depends on the distribution π. The idea resembles that behind importance sampling. The basic result on which the Metropolis-Hastings algorithm is pinned is the following theorem.

Theorem 31 *Suppose Q_{ij} is the transition matrix of a Markov chain. Assume that g is a vector of non-negative values such that $\sum_{i=1}^{N} g_i = G$ and*

$$|\frac{g_j}{Q_{ij}}| \leq K < \infty \text{ for all } i, j$$

for some finite value K. Define

$$\rho_{ij} = \min(1, \frac{g_j Q_{ji}}{g_i Q_{ij}})$$

Then the Markov Chain with transition probability matrix

$$P_{ij} = Q_{ij}\rho_{ij}, \text{ for } i \neq j \tag{3.29}$$

has stationary distribution $\pi_i = \frac{g_i}{G}$.

Proof. The proof consists of showing that the so-called "detailed balance condition" is satisfied, i.e. with $\pi_i = \frac{g_i}{G}$, that

$$\pi_i P_{ij} = \pi_j P_{ji}, \text{ for all } i, j. \tag{3.30}$$

This condition implies that when the chain is operating in equilibrium,

$$P[X_n = i, X_{n+1} = j] = P[X_n = j, X_{n+1} = i]$$

reflecting a cavalier attitude to the direction in which time flows or reversibility of the chain. Of course (3.30) is true automatically if $i = j$ and for $i \neq j$,

$$\begin{aligned} \pi_i P_{ij} &= \frac{g_i}{G} Q_{ij} \min(1, \frac{g_j Q_{ji}}{g_i Q_{ij}}) \\ &= \frac{1}{G} \min(g_i Q_{ij}, g_j Q_{ji}) \\ &= \pi_j P_{ji} \end{aligned}$$

by the symmetry of the function $\frac{1}{G} \min(g_i Q_{ij}, g_j Q_{ji})$. Now the detailed balance condition (3.30) implies that π is a stationary distribution for this Markov chain since

$$\begin{aligned} \sum_{i=1}^{N} \pi_i P_{ij} &= \sum_{i=1}^{N} \pi_j P_{ji} \\ &= \pi_j \sum_{i=1}^{N} P_{ji} \\ &= \pi_j \text{ for each } j = 1, ..., N. \end{aligned}$$

■

Provided that we are able to generate transitions for the Markov Chain with transition matrix Q, it is easy to generate a chain with transition matrix P in (3.29). If we are currently in state i, generate the next state with probability Q_{ij}. If $j = i$ then we stay in state i. If $j \neq i$, then we "accept" the move to state j with probability ρ_{ij}, otherwise we stay in state i. Notice that the Markov

Chain with transition matrix P tends to favour moves which increase the value of π. For example if the proposal chain is as likely to jump from i to j as it is to jump back so that $Q_{ij} = Q_{ji}$, then if $\pi_j > \pi_i$ the move to j is always accepted whereas if $\pi_j < \pi_i$ the move is only accepted with probability $\frac{\pi_j}{\pi_i}$. The assumption $Q_{ij} = Q_{ji}$ is a common and natural one, since in applications of the Metropolis-Hastings algorithm, it is common to choose j "at random" (i.e. uniformly distributed) from a suitable neighborhood of i.

The above proof only provides that π is a stationary distribution of the Markov Chain associated with P, not that it is necessarily the limiting distribution of this Markov chain. For this to follow we need to know that the chain is ergodic. Various conditions for ergodicity are given in the literature. See for example Robert and Casella (1999, Chapter 6) for more detail.

Gibbs Sampling

There is one simple special case of the Metropolis-Hastings algorithm that is particularly simple, common and compelling. To keep the discussion simple, suppose the possible states of our Markov Chain are points in two-dimensional space (x, y). We may assume both components are discrete or continuous. Suppose we wish to generate observations from a stationary distribution which is proportional to $g(x, y)$ so

$$\pi(x, y) = \frac{g(x, y)}{\sum_x \sum_y g(x, y)} \tag{3.31}$$

defined on this space but that the form of the distribution is such that directly generating from this distribution is difficult, perhaps because it is difficult to obtain the denominator of (3.31). However there are many circumstances where it is much easier to obtain the value of the conditional distributions

$$\pi(x|y) = \frac{\pi(x, y)}{\sum_z \pi(z, y)} \text{ and}$$
$$\pi(y|x) = \frac{\pi(x, y)}{\sum_z \pi(x, z)}$$

Now consider the following algorithm: begin with an arbitrary value of y_0 and generated x_1 from the distribution $\pi(x|y_0)$ followed by generating y_1 from the distribution $\pi(y|x_1)$. It is hard to imagine a universe in which iteratively generating values x_{n+1} from the distribution $\pi(x|y_n)$ and then y_{n+1} from the distribution $\pi(y|x_{n+1})$ does not, at least asymptotically as $n \to \infty$, eventually lead to a draw from the joint distribution $\pi(x, y)$. Indeed that is the case since the transition probabilities for this chain are given by

$$P(x_{n+1}, y_{n+1}|x_n, y_n) = \pi(x_{n+1}|y_n)\pi(y_{n+1}|x_{n+1})$$

and it is easy to show directly from these transition probabilities that

$$\begin{aligned}
&\sum_{(x,y)} P(x_1, y_1|x, y)\pi(x, y) \\
&= \pi(y_1|x_1) \sum_y \pi(x_1|y) \sum_x \pi(x, y) \\
&= \pi(y_1|x_1) \sum_y \pi(x_1, y) \\
&= \pi(x_1, y_1).
\end{aligned}$$

Of course the real power of Gibbs Sampling is achieved in problems that are not two-dimensional such as the example above, but have dimension sufficiently high that calculating the sums or integrals in the denominator of expressions like (3.31) is not computationally feasible.

Coupling From the Past: Sampling from the stationary distribution of a Markov Chain

All of the above methods assume that we generate from the stationary distribution of a Markov chain by the "until Hele freezes over" method, i.e. wait until run the chain from an arbitrary starting value and then delete the initial transient. An alternative elegant method that is feasible at least for some finite

state Markov chains is the method of "coupling from the past" due to Propp and Wilson (1996).

We assume that we are able to generate transitions in the Markov Chain. In other words if the chain is presently in state i at time n we are able to generate a random variable X_{n+1} from the distribution proportional to $P_{ij}, j = 1, ...K$. Suppose $F(x|i)$ is the cumulative distribution function $P(X_{n+1} \leq x|X_n = i)$ and let us denote its inverse by $F^{-1}(y|i)$. So if we wish to generate a random variable X_{n+1} conditional on X_n, we can use the inverse transform $X_{n+1} = F^{-1}(U_{n+1}|X_n)$ applied to the Uniform[0,1] random variable U_{n+1}. Notice that a starting value say X_{-100} together with the sequence of uniform[0,1] variables $(U_{-99}, ..., U_0)$ determines the chain completely over the period $-100 \leq t \leq 0$. If we wish to generated the value of X_t given $X_s, s < t$, then we can work this expression backwards

$$\begin{aligned} X_t &= F^{-1}(U_{t-1}|X_{t-1}) \\ &= F^{-1}(U_{t-1}|F^{-1}(U_{t-2}|X_{t-2})) \\ &= F^{-1}(U_{t-1}|F^{-1}(U_{t-2}|...F^{-1}(U_{t-1}|F^{-1}(U_s|i)))) \\ &= F_s^t(X_s), \text{ say.} \end{aligned}$$

Now imagine an infinite sequence $\{U_t, t = ..., -3, -2, -1\}$ of independent uniform[0,1] random variables that was used to generate the state X_0 of a chain at time 0. Let us imagine for the moment that there is a value of M such that $F_{-M}^0(i)$ is a constant function of i. This means that *for this particular draw of uniform random numbers,* whatever the state i of the system at time $-M$, the same state $X_0 = F_{-M}^0(i)$ is generated to time 0. All chains, possibly with different behaviour prior to time $-M$ are "coupled" at time $-M$ and identical from then on. In this case we say that *coalescence* has occurred in the interval $[-M, 0]$. No matter where we start the chain at time $-M$ it ends up in the same state at time 0, so it is quite unnecessary to simulate the chain over the whole infinite time interval $-\infty < t \leq 0$. No *matter what state is occupied*

at time $t = -M$, *the chain ends up in the same state at time* $t = 0$. When coalescence has occurred, we can safely consider the common value of the chain at time 0 to be generated from the stationary distribution since it is exactly the same value as if we had run the chain from $t = -\infty$.

There is sometimes an easy way to check whether coalescence has occurred in an interval, if the state space of the Markov chain is suitably ordered. For example suppose the states are numbered $1, 2, ..., N$. Then it is sometimes possible to relabel the states so that the conditional distribution functions $F(x|i)$ are stochastically ordered, or equivalently that $F^{-1}(U|i)$ is monotonic (say monotonically increasing) in i for each value of U. This is the case for example provided that the partial sums $\sum_{l=1}^{j} P_{il}$ are increasing functions of i for each $j = 1, 2, ..., N$. If follows that the functions $F^0_{-M}(i)$ are all monotonic functions of i and so

$$F^0_{-M}(1) \leq F^0_{-M}(2) \leq ...F^0_{-M}(N).$$

Therefore, if $F^0_{-M}(1) = F^0_{-M}(N)$, then $F^0_{-M}(i)$ must be a constant function. Notice also that if there is any time in an interval $[s, t]$ at which coalescence occurs so that $F^t_s(i)$ is a constant function of i, then for any interval $[S, T]$ containing it $[S, T] \supset [s, t]$, $F^T_S(i)$ is also a constant function of i.

It is easy to prove that coalescence occurs in the interval $[-M, 0]$ for sufficiently large M. For an ergodic finite Markov chain, there is some step size τ such that every transition has positive probability $P[X_{t+\tau} = j|X_t = i] > \epsilon$ for all i, j. Consider two independent chains, one beginning in state i and the other in state i' at time $t = 0$. Then the probability that they occupy the same state j at time $t = \tau$ is at least ϵ^2. It is easy to see that if we use inverse transform to generate the transitions and if they are driven by common random numbers then this can only increase the probability of being in the same state, so the probability these two chains are coupled at time τ is at least ϵ^2. Similarly for N possible states, the probability of coalescence in an interval of length τ

is at least $\varepsilon^N > 0$. Since there are infinitely many intervals disjoint of length τ in $[-\infty, 0]$ and the events that there is a coalescence in each interval are independent, the probability that coalescence occurs somewhere in $[-\infty, 0]$ is 1.

We now detail the Propp Wilson algorithm

1. Set $M = 1, X_U = N, X_L = 1$
2. Generate $U_{-M}....U_{-M/2+1}$ all independent $Uniform[0, 1]$.
3. For $t = -M$ to -1 repeat
   (a) obtain $X_L = F^{-1}(U_{t-1}|X_L)$ and $X_U = F^{-1}(U_{t-1}|X_U)$.
   (b) If $X_L = X_U$ stop and output $X(0) = X_L$
4. Otherwise, set $M = 2M$ and go to step 2.

This algorithm tests for coalescence repeatedly by starting on the intervals

$$[-1, 0], [-2, -1], [-4, -2], [-8, -4].$$

We are assured that with probability one, the process will terminate with coalescence after a finite number of steps. Moreover, in this algorithm that the random variable U_t once generated is NOT generated again on a subsequent pass when M is doubled. The generated U_t is reused at each pass until coalescence occurs. If U_t were regenerated on subseuqent passes, this would lead to bias in the algorithm.

It may well be that this algorithm needs to run for a very long time before achieving coalescence and an impatient observer who interrupts the algorithm prior to coalescence and starts over will bias the results. Varous modifications have been made to speed up the algorithm (e.g. Fill, 1998).

Sampling from the Stationary Distribution of a Diffusion Process

A basic Ito process of the form

$$dX_t = a(X_t)dt + \sigma(X_t)dW_t$$

is perhaps the simplest extension of a Markov chain to continuous time, continuous state-space. It is well-known that under fairly simple conditions, there is a unique (strong) solution to this equation and that the limiting distribution of X_T as $T \to \infty$ has stationary distribution with probability density function

$$f(x) = c\frac{1}{\sigma^2(x)} \exp\{2\int_0^x \frac{a(z)}{\sigma^2(z)}dz\}$$

where the constant c is chosen so that the integral of the density is 1. To be able to do this we need to assume that

$$\int_{-\infty}^{\infty} \frac{1}{\sigma^2(x)} \exp\{2\int_0^x \frac{a(z)}{\sigma^2(z)}dz\}dx < \infty. \tag{3.32}$$

In order to generate from this stationary distribution, we can now start the process at some arbitrary value X_0 and run it for a very long time T, hoping that this is sufficiently long that the process is essentially in its stationary state, or try to generate X_0 more directly from (3.32) in which case the process is beginning (and subsequently running) with its stationary distribution.

For an example, let us return to the CIR process

$$dX_t = k(b - X_t)dt + \sigma X_t^{1/2} dW_t. \tag{3.33}$$

In this case

$$a(x) = k(b - x), \text{ for } x > 0,$$
$$\sigma^2(x) = \sigma^2 x, \text{ for } x > 0.$$

Notice that

$$\frac{1}{\sigma^2 x}\exp\{2\int_{\varepsilon}^{x}\frac{k(b-z)}{\sigma^2 z}dz\} = \frac{1}{\sigma^2}x^{-1}\exp\{\frac{2kb}{\sigma^2}\ln(x/\varepsilon) - \frac{k}{\sigma^2}(x-\varepsilon)\}$$

is proportional to

$$x^{2kb/\sigma^2-1}\exp\{-kx/\sigma^2\}$$

and the integral of this function, a Gamma function, will fail to converge unless $2kb/\sigma^2 - 1 > -1$ or $2kb > \sigma^2$. Under this condition the stationary distribution of the CIR process is Gamma$(2kb/\sigma^2, \frac{\sigma^2}{k})$. If this condition fails and $2kb < \sigma^2$, then the process X_t is absorbed at 0. If we wished to simulate a CIR process in equilibrium, we should generate starting values of X_0 from the Gamma distribution. More generally for a CEV process satisfying

$$dX_t = k(b - X_t)dt + \sigma X_t^{\gamma/2}dW_t \tag{3.34}$$

a similar calculation shows that the stationary density is proportional to

$$x^{-\gamma}\exp\{-\frac{2kb}{\sigma^2}\frac{1}{x^{\gamma-1}(\gamma-1)} - \frac{k}{\sigma^2\gamma}x^{\gamma}\},\ \text{for } \gamma > 1.$$

Simulating Stochastic Partial Differential Equations.

Consider a derivative product whose underlying asset has price X_t which follows some model. Suppose the derivative pays an amount $V_0(X_T)$ on the maturity date T. Suppose that the value of the derivative depends only on the current time t and the current value of the asset S, then its current value is the discounted future payoff, an expectation of the form

$$V(S,t) = E[V_0(X_T)exp\{-\int_t^T r(X_v, v)dv\}|X_t = S] \tag{3.35}$$

where $r(X_t, t)$ is the current spot interest rate at time t. In most cases, this expectation is impossible to evaluate analytically and so we need to resort to

numerical methods. If the spot interest rate is function of *both arguments* (X_v, v) and not just a function of time, then this integral is over the *whole joint distribution of the process* $X_v, 0 < v < T$ and simple one-dimensional methods of numerical integration do not suffice. In such cases, we will usually resort to a Monte-Carlo method. The simplest version requires simulating a number of sample paths for the process X_v starting at $X_t = S$, evaluating $V_0(X_T)exp\{-\int_t^T r(X_v, v)dv\}$ and averaging the results over all simulations. We begin by discussing the simulation of the process X_v required for integrations such as this.

Many of the stochastic models in finance reduce to simple diffusion equation (which may have more than one *factor* or dimension). Most of the models in finance are Markovian in the sense that at any point t in time, the future evolution of the process depends only on the current state X_t and not on the past behaviour of the process $X_s, s < t$. Consequently we restrict to a "Markov diffusion model" of the form

$$dX_t = a(X_t, t)dt + \sigma(X_t, t)dW_t \tag{3.36}$$

with some initial value X_0 for X_t at $t = 0$. Here W_t is a driving standard Brownian motion process. Solving deterministic differential equations can sometimes provide a solution to a specific problem such as finding the arbitrage-free price of a derivative. In general, for more complex features of the derivative such as the distribution of return, important for considerations such as the *Value at Risk*, we need to obtain a solution $\{X_t, 0 < t < T\}$to an equation of the above form which is a stochastic process. Typically this can only be done by simulation. One of the simplest methods of simulating such a process is motivated through a crude interpretation of the above equation in terms of discrete time steps, that is that a small increment $X_{t+h} - X_t$in the process is approximately normally distributed with mean given by $a(X_t, t)h$and variance given by $\sigma^2(X_t, t)h$. We generate these increments sequentially, beginning with an assumed value for

X_0, and then adding to obtain an approximation to the value of the process at discrete times $t = 0, h, 2h, 3h, \ldots$. Between these discrete points, we can linearly interpolate the values. Approximating the process by assuming that the conditional distribution of $X_{t+h} - X_t$ is $N(a(X_t, t)h, \sigma^2(X_t, t)h)$ is called *Euler's method* by analogy to a simple method by the same name for solving ordinary differential equations. Given simulations of the process satisfying (3.36) together with some initial conditions, we might average the returns on a given derivative for many such simulations, (provided the process is expressed with respect to the risk-neutral distribution), to arrive at an arbitrage-free return for the derivative.

In this section we will discuss the numerical solution, or simulation of the solution to stochastic differential equations.

Letting $t_i = i\Delta x$, Equation (3.36) in integral form implies

$$X_{t_{i+1}} = X_{t_i} + \int_{t_i}^{t_{i+1}} a(X_s, s)ds + \int_{t_i}^{t_{i+1}} \sigma(X_s, s)dW_s \tag{3.37}$$

For the following lemma we need to introduce O_p or "order in probability", notation common in mathematics and probability. A sequence indexed by Δt, say $Y_{\Delta t} = O_p(\Delta t)^k$ means that when we divide this term by $(\Delta t)^k$ and then let $\Delta t \to 0$, the resulting sequence is bounded in probability or that for each ε there exists $K < \infty$ so that

$$P[|\frac{Y_{\Delta t}}{\Delta t^k}| > K] < \varepsilon$$

whenever $|\Delta t| < \varepsilon$. As an example, if W is a Brownian motion, then $\Delta W_t = W(t+\Delta t) - W(t)$ has a Normal distribution with mean 0 and standard deviation $\sqrt{\Delta t}$ and is therefore $O_p(\Delta t)^{1/2}$. Similarly Then we have two very common and useful approximations to a diffusion given by the following lemma.

Lemma 32 *If X_t satisfies a diffusion equation of the form (3.36) then*

$X_{t_{i+1}} = X_{t_i} + a(X_{t_i}, t_i)\Delta t + \sigma(X_{t_i}, t_i)\Delta W_t + O_p(\Delta t)$ *(Euler approximation)*

$X_{t_{i+1}} = X_{t_i} + a(X_{t_i}, t_i)\Delta t + \sigma(X_{t_i}, t_i)\Delta W_t + \frac{\sigma(X_{t_i}, t_i)\frac{\partial}{\partial x}\sigma(X_{t_i}, t_i)}{2}[(\Delta W_t)^2 - \Delta t] + O_p(\Delta t)^{3/2}$ *(Milstei*

Proof. Ito's lemma can be written in terms of two operators on functions f for which the derivatives below exist;

$$df(X_t,t) = L^o f dt + L^1 f dW_t \quad \text{where}$$

$$L^0 = a\frac{\partial}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2}{\partial x^2} + \frac{\partial}{\partial t}, \text{ and}$$
$$L^1 = \sigma\frac{\partial}{\partial x}.$$

Integrating, this and applying to twice differentiable functions a and σ and $s > t_i$,

$$a(X_s,s) = a(X_{t_i},t_i) + \int_{t_i}^{s} L^0 a(X_u,u)du + \int_{t_i}^{s} L^1 a(X_u,u)dW_u$$
$$\sigma(X_s,s) = \sigma(X_{t_i},t_i) + \int_{t_i}^{s} L^0 \sigma(X_u,u)du + \int_{t_i}^{s} L^1 \sigma(X_u,u)dW_u.$$

By substituting in each of the integrands in 3.37 using the above identity and iterating this process we arrive at the Ito-Taylor expansions (e.g. Kloeden and Platen, 1992). For example,

$$\int_{t_i}^{t_{i+1}} a(X_s,s)ds = \int_{t_i}^{t_{i+1}} \{a(X_{t_i},t_i) + \int_{t_i}^{s} L^0 a(X_u,u)du + \int_{t_i}^{s} L^1 a(X_u,u)dW_u\}ds$$
$$\approx a(X_{t_i},t_i)\Delta t + L^0 a(X_{t_i},t_i)\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} duds + L^1 a(X_{t_i},t_i)\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} dW_u ds$$

The first term $a(X_{t_i},t_i)\Delta t$, is an initial approximation to the desired integral and the rest is a lower order correction that we may regard as an error term for the moment. For example it is easy to see that the second term $L^0 a(X_{t_i},t_i)\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} duds$ is $O_p(\Delta t)^2$ because the integral $\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} duds = (\Delta t)^2/2$ and $L^0 a(X_{t_i},t_i)$ is bounded in probability. The third term $L^1 a(X_{t_i},t_i)\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} dW_u ds$ is $O_p(\Delta t)^{3/2}$ since $\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} dW_u ds = \int_{t_i}^{t_{i+1}}(t_{i+1}-u)dW_u$ and this is a normal random variable with mean 0 and variance $\int_{t_i}^{t_{i+1}}(t_{i+1}-u)^2 du = (\Delta t)^3/3$. We can write such a normal random variable as $3^{-1/2}(\Delta t)^{3/2}Z$ for Z a standard

normal random variable and so this is obviously $O_p(\Delta t)^{3/2}$. Thus the simplest *Euler approximation* to the distribution of the increment assumes that ΔX has conditional mean $a(X_{t_i}, t_i)\Delta t$. Similarly

$$\begin{aligned}\int_{t_i}^{t_{i+1}} \sigma(X_s, s)dW_s &= \int_{t_i}^{t_{i+1}} \{\sigma(X_{t_i}, t_i) + \int_{t_i}^{s} L^0\sigma(X_u, u)du + \int_{t_i}^{s} L^1\sigma(X_u, u)dW_u\}dW_s \\ &\approx \sigma(X_{t_i}, t_i)\Delta W_t + L^0\sigma(X_{t_i}, t_i)\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} dudW_s + L^1\sigma(X_{t_i}, t_i)\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} dW_u dW_s \\ &= \sigma(X_{t_i}, t_i)\Delta W_t + \frac{\sigma(X_{t_i}, t_i)\frac{\partial}{\partial x}\sigma(X_{t_i}, t_i)}{2}[(\Delta W_t)^2 - \Delta t] + O_p(\Delta t)^{3/2}\end{aligned}$$

since $\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} dW_u dW_s = \frac{1}{2}[(\Delta W_t)^2 - \Delta t]$, $L^0\sigma(X_u, u) = \sigma(X_{t_i}, t_i) + O_p(\Delta t)^{1/2}$, $L^1\sigma(X_u, u) = \sigma(X_u, u)\frac{\partial}{\partial x}\sigma(X_u, u)$ and $\int_{t_i}^{t_{i+1}}\int_{t_i}^{s} dudW_s = O_p(\Delta t)^{3/2}$. Putting these terms together, we arrive at an approximation to the increment of the form

$$\Delta X_t = a(X_{t_i}, t_i)\Delta t + \sigma(X_{t_i}, t_i)\Delta W_t + \frac{\sigma(X_{t_i}, t_i)\frac{\partial}{\partial x}\sigma(X_{t_i}, t_i)}{2}[(\Delta W_t)^2 - \Delta t] + O_p(\Delta t)^{3/2} \tag{3.38}$$

which allow an explicit representation of the increment in the process X in terms of the increment of a Brownian motion process $\Delta W_t \sim N(0, \Delta t)$. ■

The approximation (3.38) is called the *Milstein approximation,* a refinement of the first, the *Euler approximation.* It is the second Ito-Taylor approximation to a diffusion process. Obviously, the increments of the process are quadratic functions of a normal random variable and are no longer normal. The error approaches 0 at the rate $O_p(\Delta t)^{3/2}$ *in probability* only. This does not mean that the trajectory is approximated to this order but that the difference between the Milstein approximation to a diffusion and the diffusion itself is bounded in probability when divided by $(\Delta t)^{3/2}$ and as we let $\Delta t \to 0$. Higher order Taylor approximations are also possible, although they grow excessively complicated very quickly. See the book by Kloeden and Platten(1992) for details.

There remains the question of how much difference it makes which of these approximations we employ for a particular diffusion. Certainly there is no difference at all between the two approximations in the case that the diffusion

coefficient $\sigma(X_t, t)$ does not depend at all on X_t. In general, the difference is hard to assess but in particular cases we can at least compare the performance of the two methods. The approximations turn out to be very close in most simple cases. For example consider the stock price path in Figure 3.27. The dashed line corresponds to a Milstein approximation whereas the piecewise continuous line corresponds to the Euler approximation. In this case the Milstein appears to be a little better, but if I run a number of simulations and compare the sum of the squared errors (i.e. squared differences between the approximate value of X_t and the true value of X_t) we find that the improvement is only about two percent of the difference. The same is true even if I change the value of Δt from 1/12 (i.e. one month) to 1/52 (i.e. one week). Unlike the behaviour of higher order approximations to deterministic functions, there appears to be little advantage in using a higher order approximation, at least in the case of diffusions with smooth drift and diffusion coefficients.

We can compare using Milstein approximation on the original process and using Euler's approximation on a transformation of the process in the case that the diffusion term depends only on the state of the process (not time). In other words, suppose we have an Ito process of the form

$$dX_t = a(X_t, t)dt + \sigma(X_t)dW_t \tag{3.39}$$

where W_t is an ordinary Wiener measure. A simple transformation reduces this to a problem with constant diffusion term. Suppose $\sigma(x) > 0$ for all x and let

$$s(x) = \int_0^x \frac{1}{\sigma(z)}dz, \text{ for } x \geq 0$$
$$s(x) = -\int_x^0 \frac{1}{\sigma(z)}dz \text{ for } x < 0$$

where we assume these integrals are well defined. Let g be the inverse function of s. This inverse exists since the function is continuous monotonically increasing.

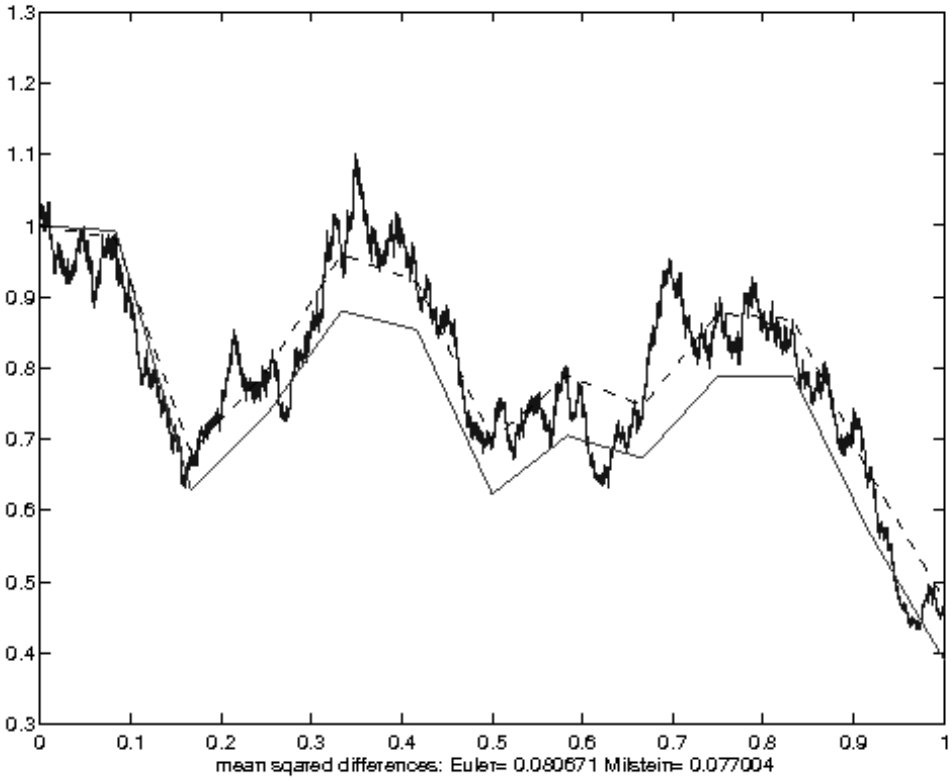

Figure 3.27: Comparison of Milstein and Euler approximation to stock with $\Delta t = 1/12$ year.

Suppose we apply Ito's lemma to the transformed process $Y_t = s(X_t)$. We obtain

$$\begin{aligned} dY_t &= \{a(X_t,t)s'(X_t) + \frac{1}{2}\sigma^2(X_t)s''(X_t)\}dt + \sigma(X_t)s'(X_t)dW_t \\ &= \{\frac{a(X_t,t)}{\sigma(X_t)} - \frac{1}{2}\sigma^2(X_t)\frac{\sigma'(X_t)}{\sigma^2(X_t)}\}dt + dW_t \\ &= \mu(Y_t,t)dt + dW_t \end{aligned}$$

where

$$\mu(Y_t,t) = \frac{a(g(Y_t),t)}{\sigma(g(Y_t))} - \frac{1}{2}\sigma'(g(Y_t)).$$

In other words, Yt satisfies an Ito equation with constant diffusion term. Suppose we generate an increment in Y_t using Euler's method and then solve for the corresponding increment in X_t.Then using the first two terms in the Taylor series expansion of g,

$$\begin{aligned} \Delta X_t &= g'(Y_t)\Delta Y_t + \frac{1}{2}g''(Y_t)(\Delta Y_t)^2 \\ &= g'(Y_t)(\mu(Y_{t_i},t_i)\Delta t + \Delta W_t) + \frac{1}{2}\sigma'(g(Y_t))\sigma(g(Y_t))(\Delta Y_t)^2 \\ &= \{a(g(Y_t),t) - \frac{1}{2}\sigma(g(Y_t))\sigma'(g(Y_t))\}\Delta t + \sigma(g(Y_t))\Delta W_t + \frac{1}{2}\sigma'(g(Y_t))\sigma(g(Y_t))(\Delta Y_t)^2 \end{aligned}$$

since

$$g'(Y_t) = \frac{1}{s'(g(Y_t))} = \sigma(g(Y_t)) \text{ and}$$

$$g''(Y_t) = \sigma'(g(Y_t))\sigma(g(Y_t)).$$

But since $(\Delta Y_t)^2 = (\Delta W_t)^2 + o(\Delta t)$ it follows that

$$\Delta X_t = \{a(X_t,t) - \frac{1}{2}\sigma(X_t)\sigma'(X_t)\}\Delta t + \sigma(X_t)\Delta W_t + \frac{1}{2}\sigma'(X_t)\sigma(X_t)(\Delta W_t)^2 + o(\Delta t)$$

and so the approximation to this increment is identical, up to the order considered, to the Milstein approximation. For most processes, it is preferable to apply a diffusion stabilizing transformation as we have here, prior to discretizing the process. For the geometric Brownian motion process, for example, the diffusion-stabilizing transformation is a multiple of the logarithm, and this transforms to a Brownian motion, for which the Euler approximation gives the exact distribution.

Example: Down-and-out-Call.

Consider an asset whose price under the risk-neutral measure Q follows a constant elasticity of variance (CEV) process

$$dS_t = rS_t dt + \sigma S_t^{\gamma} dW_t \tag{3.40}$$

for a standard Brownian motion process W_t. A down-and-out call option with exercise price K provides the usual payment $(S_T - K)^+$ of a European call option on maturity T if the asset never falls below a given *out barrier* b. The parameter $\gamma > 0$ governs the change in the diffusion term as the asset price changes. We wish to use simulation to price such an option with current asset price S_0, time to maturity T, out barrier $b < S_0$ and constant interest rate r and compare with the Black-Scholes formula as $b \to 0$.

A geometric Brownian motion is most easily simulated by taking logarithms. For example if S_t satisfies the risk-neutral specification

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{3.41}$$

then $Y_t = \log(S_t)$ satisfies

$$dY_t = (r - \sigma^2/2)dt + \sigma dW_t. \tag{3.42}$$

This is a Brownian motion and is simulated with a normal random walk. Independent normal increments are generated $\Delta Y_t \sim N((r - \sigma^2/2)\Delta t, \sigma^2 \Delta t)$ and their partial sums used to simulate the process Y_t. The return for those options that are *in the money* is the average of the values of $(e^{Y_T} - E)^+$ over those paths for which $\min\{Y_s; t < s < T\} \geq \ln(b)$. Similarly the transformation of the CEV process which provides a constant diffusion term is determined by

$$\begin{aligned} s(x) &= \int_0^x \frac{1}{\sigma(z)} dz \\ &= \int_0^x z^{-\gamma} dz = \begin{cases} \frac{x^{1-\gamma}}{1-\gamma} & \text{if} \quad \gamma \neq 1 \\ \ln(x) & \text{if} \quad \gamma = 1 \end{cases} . \end{aligned}$$

Assuming $\gamma \neq 1$, the inverse function is

$$g(y) = cy^{1/(1-\gamma)}$$

for constant c and the process $Y_t = (1-\gamma)^{-1} S_t^{1-\gamma}$ satisfies an Ito equation with constant diffusion coefficient;

$$dY_t = \{\frac{r}{\sigma} S_t^{1-\gamma} - \frac{1}{2}\gamma\sigma S_t^{\gamma-1}\}dt + dW_t$$
$$dY_t = \{\frac{r}{\sigma}(1-\gamma)Y_t - \frac{\gamma\sigma}{2(1-\gamma)Y_t}\}dt + dW_t. \qquad (3.43)$$

After simulating the process Y_t we invert the relation to obtain $S_t = ((1-\gamma)Y_t)^{1/(1-\gamma)}$. There is one fine point related to simulating the process (3.43) that we implemented in the code below. The equation (3.40) is a model for a non-negative asset price S_t but when we simulate the values Y_t from (3.43) there is nothing to prevent the process from going negative. Generally if $\gamma \geq 1/2$ and if we increment time in sufficiently small steps Δt, then it is *unlikely* that a negative value of Y_t will obtain, but when it does, we assume absorption at 0 (analogous to default or bankruptcy). The following *Matlab* function was used to simulate sample paths from the CEV process over the interval $[0, T]$.

```
function s=simcev(n,r,sigma,So,T,gam)
% simulates n sample paths of a CEV process on the interval [0,T] all with
% the same starting value So. assume gamma != 1.
Yt=ones(n,1)*(So^(1-gam))/(1-gam);  y=Yt;
dt=T/1000; c1=r*(1-gam)/sigma; c2=gam*sigma/(2*(1-gam));
dw=normrnd(0,sqrt(dt),n,1000);
for i=1:1000
    v=find(Yt);   % selects positive components of Yt for update
    Yt=max(0,Yt(v)+(c1.*Yt(v)-c2./Yt(v))*dt+dw(v,i));
    y=[y Yt];
end
```

```
s=((1-gam)*max(y,0)).^(1/(1-gam));  %transforms to St
```

For example when $r = .05, \sigma = .2, \Delta t = .00025, T = .25$, $\gamma = 0.8$ we can generate 1000 sample paths with the command

```
s=simcev(1000,.05,.2,10,.25,.8);
```

In order to estimate the price of a barrier option with a down-and-out barrier at b and exercise price K, capture the last column of s,

```
ST=s(:,1001);
```

then value a European call option based on these sample paths

```
v=exp(-r*T)*max(ST-K,0);
```

finally setting the values equal to zero for those paths which breached the lower barrier and then averaging the return from these 1000 replications;

```
v(min(s')<=9)=0;
mean(v);
```

which results in an estimated value for the call option of around \$0.86. Although the standard error is still quite large (0.06), we can compare this with the Black-Scholes price with similar parameters. [CALL,PUT] = BLSPRICE(10,10,.05,.25,.2,0) which gives a call option price of \$0.4615. Why such a considerable difference? Clearly the down-and-out barrier can only reduce the value of a call option. Indeed if we remove the down-and-out feature, the European option is valued closer to \$1.28 so the increase must be due to the differences betwen the CEV process and the geometric Brownian motion. We can confirm this by simulating the value of a barrier option in the Black_Scholes model later on.

Problems

1. Consider the mixed generator $x_n = (ax_{n-1} + 1)mod(m)$ with $m = 64$. What values of a results in the maximum possible period. Can you indicate which generators appears more and less random?

2. Consider the multiplicative generator with $m = 64$. What values of a result in the maximum possible period and which generators appears more random?

3. Consider a shuffled generator described in Section 3.2 with $k = 3, m_1 = 7, m_2 = 11$.

   Determine the period of the shuffled random number generator above and compare with the periods of the two constituent generators.

4. Prove: *If X is random variable uniform on the integers $\{0, \ldots, m-1\}$ and if Y is any integer-valued random variable independent of X, then the random variable $W = (X + Y)\text{mod } m$ is uniform on the integers $\{0, \ldots, m-1\}$.*

5. Consider the above quadratic residue generator $x_{n+1} = x_n^2 \bmod m$ with $m = 4783 \times 4027$. Write a program to generate pseudo-random numbers from this generator. Use this to determine the period of the generator starting with seed $x_0 = 196$, and with seed $x_0 = 400$?

6. Consider a multiplicative generator of the form $x_{n+1} = ax_n(\text{mod m})$. Prove that if $m = m_1 m_2$ is the product of relative primes (i.e. $\gcd(m_1, m_2) = 1$), then the period of the generator $x_{n+1} = ax_n(\text{mod } m)$ is the least common multiple of the generators with moduli m_1 and m_2.

7. Consider a sequence of independent $U[0, 1]$ random variables $U_1, ..., U_n$.

Define indicator random variables

$S_i = 1$ if $U_{i-1} < U_i$ and $U_i > U_{i+1}$ for $i = 2, 3, ..., n-1$, otherwise $S_i = 0$,

$T_i = 1$ if $U_{i-1} > U_i$ and $U_i < U_{i+1}$ for $i = 2, 3, ..., n-1$, otherwise $T_i = 0$.

Verify the following:

(a)

$$R = 1 + \sum (S_i + T_i)$$

(b)

$$E(T_i) = E(S_i) = \frac{1}{3} \text{ and } E(R) = \frac{2n-1}{3}$$

(c) $cov(T_i, T_j) = cov(S_i, S_j) = -\frac{1}{9}$.if $|i-j| = 1$ and it equals 0 if $|i-j| > 1$.

(d) $cov(S_i, T_j) = \begin{cases} \frac{5}{24} - \frac{1}{9} := \frac{7}{72} & \text{if} \quad |i-j| = 1 \\ -\frac{1}{9} & \text{if} \quad i = j \\ 0 & \text{if} \quad |i-j| > 1 \end{cases}$.

(e) $var(R) = 2(n-2)\frac{1}{3}(\frac{2}{3}) + 4(n-3)(-\frac{1}{9}) + 4(n-3)(\frac{7}{72}) + 2(n-2)(-\frac{1}{9}) = \frac{3n-5}{18}$.

(f) Confirm these formulae for mean and variance of R in the case $n = 3, 4$.

8. Verify that for the serial correlation statistic C_j,

$$var(C_j) = \begin{cases} 4/45n & \text{for} \quad j = 0 \\ \frac{7}{72n} & \text{for} \quad j \geq 2, \text{ even} \\ \frac{13}{144n} & \text{for} \quad j \geq 1, \text{ odd} \end{cases}$$

9. Consider the turbo-pascal generator $x_{n+1} = (134775813 x_n + 1) \bmod 2^{32}$. Generate a sequence of length 5000 and apply the serial correlation test. Is there evidence of dependence?

10. Generate 1000 daily "returns" $X_i, i = 1, 2, ..., 1000$ from each of the two distributions, the Cauchy and the logistic. In each case, assume that $a = 0, b = 0.1$. Graph the total return over an n day period versus n. Is there a qualitative difference in the two graphs? Repeat with a graph of the average daily return.

11. Briefly indicate an efficient algorithm for generating one random variable from each of the following distributions and generate such a random variable using one or more of the uniform[0,1] random numbers.

U_i:	*0.794*	*0.603*	*0.412*	*0.874*	*0.268*	*0.990*	*0.059*	*0.112*	*0.395*

(a) X$\sim U[-1, 2]$.

(b) a random variable X with probability density function $f(x) = \frac{3}{16}x^{1/2}, 0 < x < 4$

(c) A discrete random number X having probability function $P[X = x] = (1-p)^x p, x = 0, 1, 2, ..., p = 0.3$.

(d) A random variable X with the normal distribution, mean 1 and *variance* 4.

(e) A random variable X with probability density function

$$f(x) = cx^2 e^{-x}, 0 \leq x < 1$$

for constant $c = 1/(2 - 5e^{-1})$.

(f) A random variable X with the following probability function:

x	0	1	2	3
$P[X = x]$	0.1	0.2	0.3	0.4

12. Consider the multiplicative pseudo-random number generator

$$x_{n+1} = ax_n \bmod 150$$

starting with seed $x_0 = 7$. Try various values of the multiplier a and determine for which values the period of the generator appears to be maximal.

13. Consider the linear congruential generator

$$x_{n+1} = (ax_n + c) \bmod 2^8$$

What is the maximal period that this generator can achieve when $c = 1$ and for what values of a does this seem to be achieved? Repeat when $c = 0$.

14. Evaluate the following integral by simulation:

$$\int_0^1 (1 - x^2)^{3/2} dx.$$

15. Let U be a uniform random variable on the interval [0,1]. Find a function of U which is uniformly distributed on the interval [0,2]. The interval $[a, b]$?

16. Evaluate the following integral by simulation:

$$\int_0^2 x^{3/2}(4 - x)^{1/2} dx.$$

17. Evaluate the following integral by simulation:

$$\int_{-\infty}^{\infty} e^{-x^2} dx.$$

(Hint: Rewrite this integral in the form $2\int_0^{\infty} e^{-x^2} dx$ and then change variables to $y = x/(1 + x)$)

18. Evaluate the following integral by simulation:

$$\int_0^1 \int_0^1 e^{(x+y)^2} dxdy.$$

(Hint: Note that if U_1, U_2 are independent Uniform[0,1] random variables, $E[g(U_1, U_2)] = \int_0^1 \int_0^1 g(x, y)dxdy$ for any function g).

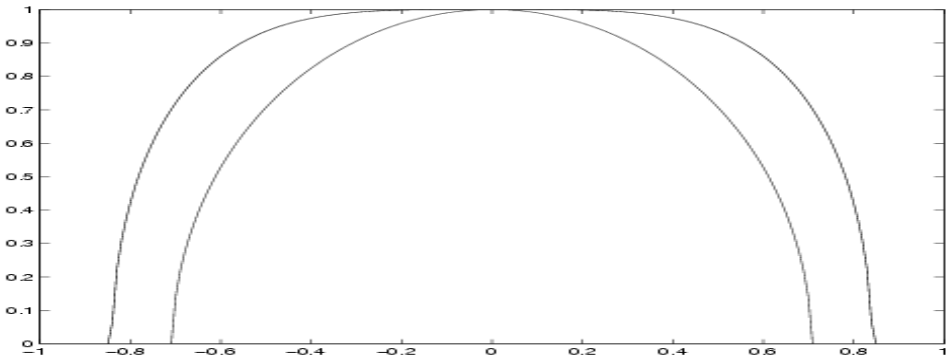

Figure 3.28: The region $\{(x,y); -1 < x < 1,\ y > 0, \sqrt{1-2x^2} < y < \sqrt{1-2x^4}\}$

19. Find the covariance $cov(U, e^U)$ by simulation where U is uniform[0,1] and compare the simulated value to the true value.

20. Find by simulation the area of the region $\{(x,y); -1 < x < 1,\ y > 0, \sqrt{1-2x^2} < y < \sqrt{1-2x^4}\}$. The boundaries of the region are graphed in Figure 3.28.

21. For independent uniform random numbers $U_1, U_{2,....}$ define the random variable $N = \min\{n; \sum_{i=1}^{n} U_i > 1\}$.

    Estimate $E(N)$ by simulation. Repeat for larger and larger numbers of simulations. Guess on the basis of these simulations what is the value of $E(N)$. Can you prove this?

22. Assume that you have available a Uniform[0,1] random number generator. Give a precise algorithm for generating observations from a distribution with probability density function

$$f(x) = \frac{(x-1)^3}{4}$$

    for $1 \leq x \leq 3$. Record the time necessary to generate the sample mean of 5,000 random variables with this distribution. .

23. Assume that you have available a Uniform[0,1] random number generator. Give a precise algorithm for generating observations from a distribution with probability density function $\frac{(x-20)}{200}$ for $20 \leq x \leq 40$. Record the time necessary to generate the sample mean of 5,000 random variables with this distribution. .

24. Assume that you have available a Uniform[0,1] random number generator. Give a precise algorithm for generating observations from a distribution with a density function of the form $f(x) = cx^3e^{-x/2}$ for $x > 0$ and appropriate constant c. Record the time necessary to generate the sample mean of 100,000 random variables with this distribution.

25. Assume that you have available a Uniform[0,1] random number generator. Give a precise algorithm for generating observations from a discrete distribution with $P[X = j] = (2/3)(1/3)^j; j = 0, 1, ...$Record the time necessary to generate the sample mean of 100,000 random variables with this distribution. .

26. Assume that you have available a Uniform[0,1] random number generator. Give a precise algorithm for generating observations from a distribution with probability density function $f(x) = e^{-x}, 0 \leq x < \infty$. Record the time necessary to generate the sample mean of 100,000 random variables with this distribution. Compute as well the sample variance and compare withe the sample mean. How large would the simulation need to be if we wanted to estimate the mean within 0.01 with a 95% confidence interval?

27. Assume that you have available a Uniform[0,1] random number generator. Give a precise algorithm for generating observations from a distribution which has probability density function $f(x) = x^3, 0 < x < \sqrt{2}$. Record the time necessary to generate the sample mean of 100,000 random variables

with this distribution. Determine the standard error of the sample mean. How large would the simulation need to be if we wanted to estimate the mean within 0.01 with a 95% confidence interval?

28. Assume that you have available a Uniform[0,1] random number generator. Give a precise algorithm for generating observations from a discrete distribution with probability function

x=	0	1	2	3	4	5
P[X=x]=	0.1	0.2	0.25	0.3	0.1	0.05

Record the time necessary to generate the sample mean of 100,000 random variables with this distribution. Compare the sample mean and variance with their theoretical values. How large would the simulation need to be if we wanted to estimate the mean within 0.01 with a 95% confidence interval? How much difference does it make to the time if we use an optimal binary search? If we use the alias method?

29. Give an algorithm for generating observations from a distribution which has cumulative distribution function $F(x) = \frac{x+x^3+x^5}{3}, 0 < x < 1$. Record the time necessary to generate the sample mean of 100,000 random variables with this distribution. (Hint: Suppose we generate X_1 with cumulative distribution function $F_1(x)$ and X_2 with cumulative distribution function $F_2(x)$, X_3 with cumulative distribution function $F_3(x)$ We then generate $J = 1, 2,$ or 3 such that $\mathrm{P}[J = j] = p_j$ and output the value X_J. What is the cumulative distribution function of the random variable output?)

30. Consider independent random variables X_i $i = 1, 2, 3$ with cumulative distribution function

$$F_i(x) = \begin{cases} x^2, & i = 1 \\ \frac{e^x - 1}{e - 1} & i = 2 \\ xe^{x-1}, & i = 3 \end{cases}$$

for $0 < x < 1$. Explain how to obtain random variables with cumulative distribution function $G(x) = \Pi_{i=1}^{3} F_i(x)$ and $G(X) = 1 - \Pi_{i=1}^{3}(1 - F_i(x))$.

(Hint: consider the cumulative distribution function of the minimum and maximum).

31. Suppose we wish to estimate a random variable X having cumulative distribution function $F(x)$ using the inverse transform theorem, but the exact cumulative distribution function is not available. We do, however, have an unbiased estimator $\widehat{F}(x)$ of $F(x)$ so that $0 \leq \widehat{F}(x) \leq 1$ and $E\,\widehat{F}(x) = F(x)$ for all x. Show that provided the uniform variate U is independent of $\widehat{F}(x)$, the random variable $X = \widehat{F}^{-1}(U)$ has cumulative distribution function $F(x)$.

32. Give an algorithm for generating a random variable with probability density function

$$f(x) = 30(x^2 - 2x^3 + x^4), \quad 0 < x < 1$$

Discuss the efficiency of your approach.

33. The interarrival times between consecutive buses at a certain bus stop are independent uniform$[0, 1]$ random variables starting at clock time $t = 0$. You arrive at the bus stop at time $t = 1$. Determine by simulation the expected time that you will have to wait for the next bus. Is it more than $1/2$? Explain.

34. What is the probability density function of $X = a(1 - \sqrt{U})$ where $U \sim U[0, 1]$?

35. Develop an algorithm for generating variates from the density:

$$f(x) = 2/\sqrt{\pi} e^{2a - x^2 - a^2/x^2}, x > 0$$

36. Develop an algorithm for generating variates from the density:

$$f(x) = \frac{2}{e^{\pi x} + e^{-\pi x}}, \text{for } -\infty < x < \infty$$

37. Explain how the following algorithm works and what distribution is generated.

    (a) Let $I = 0$

    (b) Generate $U \sim U[0,1]$and set $T = U$.

    (c) Generate U^*. IF $U \leq U^*$return $X = I + T$.

    (d) Generate U. If $U \leq U^*$go to c.

    (e) $I = I + 1$. Go to b

38. Obtain generators for the following distributions:

    (a) *Rayleigh*

$$f(x) = \frac{x}{\sigma^2} e^{-x^2/2\sigma^2}, x \geq 0 \tag{3.44}$$

    (b) *Triangular*

$$f(x) = \frac{2}{a}(1 - \frac{x}{a}), 0 \leq x \leq a \tag{3.45}$$

39. Show that if (X, Y) *are independent standard normal variates, then* $\sqrt{X^2 + Y^2}$ *has the distribution of the square root of a chi-squared(2) (i.e. exponential(2)) variable and* $arctan(Y/X)$ *is uniform on* $[0, 2\pi]$.

40. Generate the pair of random variables (X, Y)

$$(X, Y) = R(cos\Theta, sin\Theta) \tag{3.46}$$

where we use a random number generator with poor lattice properties such as the generator $x_{n+1} = (383x_n + 263) \text{ mod}(10{,}000)$ to generate

our uniform random numbers. Use this generator together with the Box-Mueller algorithm to generate 5000 pairs of independent random normal numbers. Plot the results. Do they appear independent?

41. Assume that a option has payoff at expiry one year from now ($T = 1$) given by the function $g(S_T) = 0, S_T < 20$, and $g(S_T) = \frac{S_T - 20}{S_T}, S_T > 20$. What is the approximate present value of the option assuming that the risk-neutral interest rate is 5 percent, the current price of the stock is 20, and the annual volatility is 20 percent. Determine this by simulating 1000 stock prices S_T and averaging the discounted return from a corresponding option. Repeat with 100000 simulations. What can you say about the precision?

42. (*Log-normal generator*) Describe an algorithm for generating log-normal random variables with probability density function given by

$$g(x|\eta, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} exp\{-(logx - log\eta + \sigma^2/2)^2/2\sigma^2\}. \tag{3.47}$$

43. (*hedging with futures*). I need to buy 1000 barrels of heating oil on November 1 1998. On June 1, I go long a December futures contract which allows me to purchase 1000 barrels of heating oil on December 1 for $20 per barrel. Suppose we have observed that the price of heating oil is log-normally distributed with monthly volatility 2 percent. The spot interest rate is presently 5 percent per annum

    (a) What is the value of the oil future on November 1 as a function of the current price of oil?

    (b) Determine by simulation what is the standard deviation of the value of my portfolio on November 1 assuming I sell the futures contract at that time.

    (c) How much difference would it have made if I had purchased the optimal number of futures rather than 1000?

44. (*Multivariate Normal generator*) Suppose we want to generate a multivariate normal random vector $(X_1, X_2, ..., X_N)$ having mean vector $(\mu_1, ..., \mu_N)$ and covariance matrix the $N \times N$ matrix Σ. The usual procedure involves a decomposition of Σ into factors such that $A'A = \Sigma$. For example, A could be determined from the Cholesky decomposition, in Matlab, *A=chol(sigma),* or in **R**, A= chol(sigma, pivot = FALSE, LINPACK = pivot) which provides such a matrix A which is also upper triangular, in the case that Σ is positive definite. Show that if $Z = (Z_1, ..., Z_N)$ is a vector of independent standard normal random variables then the vector $X = (\mu_1, ..., \mu_N) + ZA$ has the desired distribution.

45. (Ahrens-Dieter) Show that the rejection algorithm of Ahrens and Dieter ($b = 1$) has rejection constant c that is bounded for all $\alpha \epsilon (0, 1]$ and approaches 1 as $\alpha \to 0$.

46. What distribution is generated by the following algorithm where U is uniform$[0, 1]$ and V is uniform $[-\sqrt{2/e}, \sqrt{2/e}]$?

    (a) GENERATE U, V

    (b) PUT $X = V/U$

    (c) IF $-ln(U) < X^2/4$,GO TO a.; ELSE RETURN X.

    (Hint: First prove the following result due to Kinderman and Monahan, 1977: Suppose (U, V) are uniformly distributed on the region $\{(u, v); 0 \leq v \leq 1,\ 0 \leq u \leq \sqrt{f(\frac{v}{u})}\}$ for some integrable function $1 \geq f(x) \geq 0$. Then V/U has probability density function $cf(x)$ with $c = 1/\int f(x)dx$.)

47. *(Euler vs. Milstein Approximation)* Use the Milstein approximation with step size .001 to simulate a geometric Brownian motion of the form

$$dS_t = .07S_t dt + .2S_t dW_t$$

Compare both the Euler and the Milstein approximations using different step sizes, say $\Delta t = 0.01, 0.02, 0.05, 0.1$ and use each approximation to price an at-the-money call option assuming $S_0 = 50$ and expiry at $T = 0.5$. How do the two methods compare both for accurately pricing the call option and for the amount of computing time required?

48. *(Cox, Ingersoll, Ross model for interest rates)* Use the Milstein approximation to simulate paths from a CIR model of the form

$$dr_t = k(b - r_t) + \sigma\sqrt{r_t}dW_t$$

and plot a histogram of the distribution of r_1 assuming that $r_0 = .05$ for $b = 0.04$. What are the effects of the parameters k and b?

49. Simulate independent random variables from the Normal Inverse Gamma distribution using parameter values so that the expected value, the variance, the skewness and the kurtosis of daily returns are respectively 0, 0.004, 0.5 and 4 respectively. Evaluate an at the money call option with time to maturity 250 days using these simulated values and compare the price of the option with the Black-Scholes price. Repeat with an option whose strike is 10% over the initial stock price and one 10% under.

50. Suppose interest rates follow the constant elasticity of variance process of the form

$$dr_t = k(b - r_t) + \sigma|r_t|^\gamma dW_t$$

for parameters value $\gamma, b, k > 0$. For various values of the parameters k, γ and for $b = 0.04$ use both Euler and Milsten to generate paths from this process. Draw conclusions about the following:

    (a) When does the marginal distribution of r_t appear to approach a steady state solution. Plot the histogram of this steady state distribution.

(b) Are there simulations that result in a negative value of r? How do you rectify this problem?

(c) What does the parameter σ represent? Is it the annual volatility of the process?

51. Consider a sequence of independent random numbers $X_1, X_2, ...$with a continuous distribution and let M be the first one that is less than its predecessor:

$$M = \min\{n; X_1 \leq X_2 \leq ... \leq X_{n-1} > X_n\}$$

(a) Use the identity $E(M) = \sum_{n=0}^{\infty} P[M > n\}$ to show $E(M) = e$.

(b) Use 100,000 simulation runs and part a to estimate e with a 95% confidence interval.

(c) How many simulations are required if you wish to estimate e within 0.005 (using a 95% confidence interval)?

52. Daily relative losses from a portfolio are assumed to follow the NIG distribution with parameters

$$\alpha = 100, \delta = .02, \beta = 0, \mu = 0.$$

In other words if the portfolio is worth S_t at the end of day t, then

$$\frac{S_t - S_{t+1}}{S_t}$$

has the above NIG distribution. Assume daily relative losses are independent of one another. Assume $S_0 = 100000$. Use simulation to determine the weekly 99% VAR, i.e. the value x such that

$$P[S_5 - S_0 \leq x] = 0.99$$

Compare this result with the VAR if we replace the NIG distribution with the Normal having the same mean and variance.

53. The interarrival times between consecutive buses at a certain bus stop are independent Beta(1/2,1/2) random variables with probability density function

$$f(x) = \frac{1}{\pi\sqrt{x(1-x)}}, \quad 0 < x < 1.$$

starting at clock time $t = 0$. You arrive at the bus stop at time $t = 10$. Determine by simulation the expected time that you will have to wait for the next bus. Is it more than 1/2 ? Repeat when the interarrival times have the distribution of Y^2 where Y is an exponential with expected value $\frac{1}{2}$. Compare your average wait times with the expected time between buses and explain how your results are possible.