while $(X > 1)$
  generate $U_1, U_2 \sim$ Unif[0,1]
  $U_1 \leftarrow 2 * U_1 - 1, \quad U_2 \leftarrow 2 * U_2 - 1$
  $X \leftarrow U_1^2 + U_2^2$
end
$Y \leftarrow \sqrt{-2 \log X / X}$
$Z_1 \leftarrow U_1 Y, \quad Z_2 \leftarrow U_2 Y$
return Z_1, Z_2.

Fig. 2.11. Marsaglia-Bray algorithm for generating normal random variables.

Approximating the Inverse Normal

Applying the inverse transform method to the normal distribution entails evaluation of Φ^{-1}. At first sight, this may seem infeasible. However, there is really no reason to consider Φ^{-1} any less tractable than, e.g., a logarithm. Neither can be computed exactly in general, but both can be approximated with sufficient accuracy for applications. We discuss some specific methods for evaluating Φ^{-1}.

Because of the symmetry of the normal distribution,

$$\Phi^{-1}(1-u) = -\Phi^{-1}(u), \quad 0 < u < 1;$$

it therefore suffices to approximate Φ^{-1} on the interval $[0.5, 1)$ (or the interval $(0, 0.5]$) and then to use the symmetry property to extend the approximation to the rest of the unit interval. Beasley and Springer [43] provide a rational approximation

$$\Phi^{-1}(u) \approx \frac{\sum_{n=0}^{3} a_n (u - \frac{1}{2})^{2n+1}}{1 + \sum_{n=0}^{3} b_n (u - \frac{1}{2})^{2n}}, \tag{2.27}$$

for $0.5 \le u \le 0.92$, with constants a_n, b_n given in Figure 2.12; for $u > 0.92$ they use a rational function of $\sqrt{\log(1-u)}$. Moro [271] reports greater accuracy in the tails by replacing the second part of the Beasley-Springer approximation with a Chebyshev approximation

$$\Phi^{-1}(u) \approx g(u) = \sum_{n=0}^{8} c_n [\log(-\log(1-u))]^n, \quad 0.92 \le u < 1, \tag{2.28}$$

with constants c_n again given in Figure 2.12. Using the symmetry rule, this gives

$$\Phi^{-1}(u) \approx -g(1-u) \quad 0 < u \le .08.$$

With this modification, Moro [271] finds a maximum absolute error of 3×10^{-9} out to seven standard deviations (i.e., over the range $\Phi(-7) \le u \le \Phi(7)$). The combined algorithm from Moro [271] is given in Figure 2.13.

$a_0 =$	2.50662823884	$b_0 =$	-8.47351093090
$a_1 =$	-18.61500062529	$b_1 =$	23.08336743743
$a_2 =$	41.39119773534	$b_2 =$	-21.06224101826
$a_3 =$	-25.44106049637	$b_3 =$	3.13082909833

$c_0 = 0.3374754822726147$	$c_5 = 0.0003951896511919$
$c_1 = 0.9761690190917186$	$c_6 = 0.0000321767881768$
$c_2 = 0.1607979714918209$	$c_7 = 0.0000002888167364$
$c_3 = 0.0276438810333863$	$c_8 = 0.0000003960315187$
$c_4 = 0.0038405729373609$	

Fig. 2.12. Constants for approximations to inverse normal.

Input: u between 0 and 1
Output: x, approximation to $\Phi^{-1}(u)$.
$y \leftarrow u - 0.5$
if $|y| < 0.42$
  $r \leftarrow y * y$
  $x \leftarrow y * (((a_3 * r + a_2) * r + a_1) * r + a_0)/$
    $((((b_3 * r + b_2) * r + b_1) * r + b_0) * r + 1)$
else
  $r \leftarrow u$;
  if $(y > 0)$ $r \leftarrow 1 - u$
  $r \leftarrow \log(-\log(r))$
  $x \leftarrow c_0 + r * (c_1 + r * (c_2 + r * (c_3 + r * (c_4 +$
    $r * (c_5 + r * (c_6 + r * (c_7 + r * c_8)))))))$
  if $(y < 0)$ $x \leftarrow -x$
return x

Fig. 2.13. Beasley-Springer-Moro algorithm for approximating the inverse normal.

The problem of computing $\Phi^{-1}(u)$ can be posed as one of finding the root x of the equation $\Phi(x) = u$ and in principle addressed through any general root-finding algorithm. Newton's method, for example, produces the iterates

$$x_{n+1} = x_n - \frac{\Phi(x_n) - u}{\phi(x_n)},$$

or, more explicitly,

$$x_{n+1} = x_n + (u - \Phi(x_n)) \exp(-0.5 x_n \cdot x_n + c), \quad c \equiv \log(\sqrt{2\pi}).$$

Marsaglia, Zaman, and Marsaglia [251] recommend the starting point

$$x_0 = \pm\sqrt{|-1.6 \log(1.0004 - (1-2u)^2)|},$$

the sign depending on whether $u \geq 0$ or $u < 0$. This starting point gives a surprisingly good approximation to $\Phi^{-1}(u)$. A root-finding procedure is useful when extreme precision is more important than speed — for example, in

tabulating "exact" values or evaluating approximations. Also, a small number of Newton steps can be appended to an approximation like the one in Figure 2.13 to further improve accuracy. Adding just a single step to Moro's [271] algorithm appears to reduce the maximum error to the order of 10^{-15}.

Approximating the Cumulative Normal

Of course, the application of Newton's method presupposes the ability to evaluate Φ itself quickly and accurately. Evaluation of the cumulative normal is necessary for many financial applications (including evaluation of the Black-Scholes formula), so we include methods for approximating this function. We present two methods; the first is faster and the second is more accurate, but both are probably fast enough and accurate enough for most applications.

The first method, based on work of Hastings [171], is one of several included in Abramowitz and Stegun [3]. For $x \geq 0$, it takes the form

$$\Phi(x) \approx 1 - \phi(x)(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5), \quad t = \frac{1}{1 + px},$$

for constants b_i and p. The approximation extends to negative arguments through the identity $\Phi(-x) = 1 - \Phi(x)$. The necessary constants and an explicit algorithm for this approximation are given in Figure 2.14. According to Hastings [171, p.169], this method has a maximum absolute error less than 7.5×10^{-8}.

$b_1 = 0.319381530$ $\quad p = 0.2316419$
$b_2 = -0.356563782$ $\quad c = \log(\sqrt{2\pi}) = 0.918938533204672$
$b_3 = 1.781477937$
$b_4 = -1.821255978$
$b_5 = 1.330274429$

Input: x
Output: y, approximation to $\Phi(x)$
$a \leftarrow |x|$
$t \leftarrow 1/(1 + a * p)$
$s \leftarrow ((((b_5 * t + b_4) * t + b_3) * t + b_2) * t + b_1) * t$
$y \leftarrow s * \exp(-0.5 * x * x - c)$
if $(x > 0)$ $y \leftarrow 1 - y$
return y;

Fig. 2.14. Hastings' [171] approximation to the cumulative normal distribution as modified in Abramowitz and Stegun [3].

The second method we include is from Marsaglia et al. [251]. Like the Hastings approximation above, this method is based on approximating the

ratio $(1-\Phi(x))/\phi(x)$. According to Marsaglia et al. [251], as an approximation to the tail probability $1-\Phi(x)$ this method has a maximum *relative* error of 10^{-15} for $0 \le x \le 6.23025$ and 10^{-12} for larger x. (Relative error is much more stringent than absolute error in this setting; a small absolute error is easily achieved for large x using the approximation $1-\Phi(x) \approx 0$.) This method takes about three times as long as the Hastings approximation, but both methods are very fast. The complete algorithm appears in Figure 2.15.

$v_1 = 1.253314137315500$ $\quad v_9 = 0.1231319632579329$
$v_2 = 0.6556795424187985$ $\quad v_{10} = 0.1097872825783083$
$v_3 = 0.4213692292880545$ $\quad v_{11} = 0.09902859647173193$
$v_4 = 0.3045902987101033$ $\quad v_{12} = 0.09017567550106468$
$v_5 = 0.2366523829135607$ $\quad v_{13} = 0.08276628650136917$
$v_6 = 0.1928081047153158$ $\quad v_{14} = 0.0764757610162485$
$v_7 = 0.1623776608968675$ $\quad v_{15} = 0.07106958053885211$
$v_8 = 0.1401041834530502$
$c = \log(\sqrt{2\pi}) = 0.918938533204672$

Input: x between -15 and 15
Output: y, approximation to $\Phi(x)$.
$j \leftarrow \lfloor \min(|x| + 0.5, 14) \rfloor$
$z \leftarrow j, \quad h \leftarrow |x| - z, \quad a \leftarrow v_{j+1}$
$b \leftarrow z * a - 1, \quad q \leftarrow 1, \quad s \leftarrow a + h * b$
for $i = 2, 4, 6, \ldots, 24 - j$
  $a \leftarrow (a + z * b)/i$
  $b \leftarrow (b + z * a)/(i + 1)$
  $q \leftarrow q * h * h$
  $s \leftarrow s + q * (a + h * b)$
end
$y = s * \exp(-0.5 * x * x - c)$
if $(x > 0)$ $y \leftarrow 1 - y$
return y

Fig. 2.15. Algorithm of Marsaglia et al. [251] to approximate the cumulative normal distribution.

Marsaglia et al. [251] present a faster approximation achieving similar accuracy but requiring 121 tabulated constants. Marsaglia et al. also detail the use of accurate approximations to Φ in constructing approximations to Φ^{-1} by tabulating "exact" values at a large number of strategically chosen points. Their method entails the use of more than 2000 tabulated constants, but the constants can be computed rather than tabulated, given an accurate approximation to Φ.

Other methods for approximating Φ and Φ^{-1} found in the literature are often based on the *error function*

$$\mathrm{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$$

and its inverse. Observe that for $x \geq 0$,

$$\mathrm{Erf}(x) = 2\Phi(x\sqrt{2}) - 1, \quad \Phi(x) = \tfrac{1}{2}[\mathrm{Erf}(x/\sqrt{2}) + 1]$$

and

$$\mathrm{Erf}^{-1}(u) = \frac{1}{\sqrt{2}}\Phi^{-1}(\frac{u+1}{2}), \quad \Phi^{-1}(u) = \sqrt{2}\mathrm{Erf}^{-1}(2u-1),$$

so approximations to Erf and its inverse are easily converted into approximations to Φ and its inverse. Hastings [171], in fact, approximates Erf, so the constants in Figure 2.14 (as modified in [3]) differ from his, with p smaller and the b_i larger by a factor of $\sqrt{2}$.

Devroye [95] discusses several other methods for sampling from the normal distribution, including some that may be substantially faster than evaluation of Φ^{-1}. Nevertheless, as discussed in Section 2.2.1, the inverse transform method has some advantages — particularly in the application of variance reduction techniques and low-discrepancy methods — that will often justify the additional computational effort. One advantage is that the inverse transform method requires just one uniform input per normal output: a relevant notion of the *dimension* of a Monte Carlo problem is often the maximum number of uniforms required to generate one sample path, so methods requiring more uniforms per normal sample implicitly result in higher dimensional representations. Another useful property of the inverse transform method is that the mapping $u \mapsto \Phi^{-1}(u)$ is both continuous and monotone. These properties can sometimes enhance the effectiveness of variance reduction techniques, as we will see in later sections.

2.3.3 Generating Multivariate Normals

A multivariate normal distribution $N(\mu, \Sigma)$ is specified by its mean vector μ and covariance matrix Σ. The covariance matrix may be specified implicitly through its diagonal entries σ_i^2 and correlations ρ_{ij} using (2.22); in matrix form,

$$\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_d \end{pmatrix} \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1d} \\ \rho_{12} & \rho_{22} & & \rho_{2d} \\ \vdots & & \ddots & \vdots \\ \rho_{1d} & \rho_{2d} & \cdots & \rho_{dd} \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_d \end{pmatrix}.$$

From the Linear Transformation Property (2.23), we know that if $Z \sim N(0, I)$ and $X = \mu + AZ$, then $X \sim N(\mu, AA^\top)$. Using any of the methods discussed in Section 2.3.2, we can generate independent standard normal random variables $Z_1, \ldots, Z_d$ and assemble them into a vector $Z \sim N(0, I)$. Thus, the problem of sampling X from the multivariate normal $N(\mu, \Sigma)$ reduces to finding a matrix A for which $AA^\top = \Sigma$.

Cholesky Factorization

Among all such A, a lower triangular one is particularly convenient because it reduces the calculation of $\mu + AZ$ to the following:

$$\begin{aligned} X_1 &= \mu_1 + A_{11}Z_1 \\ X_2 &= \mu_2 + A_{21}Z_1 + A_{22}Z_2 \\ &\vdots \\ X_d &= \mu_d + A_{d1}Z_1 + A_{d2}Z_2 + \cdots + A_{dd}Z_d. \end{aligned}$$

A full multiplication of the vector Z by the matrix A would require approximately twice as many multiplications and additions. A representation of Σ as $AA^\top$ with A lower triangular is a *Cholesky factorization* of Σ. If Σ is positive definite (as opposed to merely positive semidefinite), it has a Cholesky factorization and the matrix A is unique up to changes in sign.

Consider a 2×2 covariance matrix Σ, represented as

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{pmatrix}.$$

Assuming $\sigma_1 > 0$ and $\sigma_2 > 0$, the Cholesky factor is

$$A = \begin{pmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sqrt{1-\rho^2}\sigma_2 \end{pmatrix},$$

as is easily verified by evaluating $AA^\top$. Thus, we can sample from a bivariate normal distribution $N(\mu, \Sigma)$ by setting

$$\begin{aligned} X_1 &= \mu_1 + \sigma_1 Z_1 \\ X_2 &= \mu_2 + \sigma_2\rho Z_1 + \sigma_2\sqrt{1-\rho^2}Z_2, \end{aligned}$$

with Z_1, Z_2 independent standard normals.

For the case of a $d \times d$ covariance matrix Σ, we need to solve

$$\begin{pmatrix} A_{11} & & & \\ A_{21} & A_{22} & & \\ \vdots & \vdots & \ddots & \\ A_{d1} & A_{d2} & \cdots & A_{dd} \end{pmatrix} \begin{pmatrix} A_{11} & A_{21} & \cdots & A_{d1} \\ & A_{22} & \cdots & A_{d2} \\ & & \ddots & \vdots \\ & & & A_{dd} \end{pmatrix} = \Sigma.$$

Traversing the Σ_{ij} by looping over $j = 1, \ldots, d$ and then $i = j, \ldots, d$ produces the equations

$$\begin{aligned} A_{11}^2 &= \Sigma_{11} \\ A_{21}A_{11} &= \Sigma_{21} \\ &\vdots \end{aligned}$$

$$\begin{aligned} A_{d1}A_{11} &= \Sigma_{d1} \\ A_{21}^2 + A_{22}^2 &= \Sigma_{22} \\ &\vdots \\ A_{d1}^2 + \cdots + A_{dd}^2 &= \Sigma_{dd}. \end{aligned} \tag{2.29}$$

Exactly one new entry of the A matrix appears in each equation, making it possible to solve for the individual entries sequentially.

More compactly, from the basic identity

$$\Sigma_{ij} = \sum_{k=1}^{j} A_{ik}A_{jk}, \quad j \le i,$$

we get

$$A_{ij} = \left(\Sigma_{ij} - \sum_{k=1}^{j-1} A_{ik}A_{jk} \right) / A_{jj}, \quad j < i, \tag{2.30}$$

and

$$A_{ii} = \sqrt{\Sigma_{ii} - \sum_{k=1}^{i-1} A_{ik}^2}. \tag{2.31}$$

These expressions make possible a simple recursion to find the Cholesky factor. Figure 2.16 displays an algorithm based on one in Golub and Van Loan [162]. Golub and Van Loan [162] give several other versions of the algorithm and also discuss numerical stability.

Input: Symmetric positive definite matrix $d \times d$ matrix Σ
Output: Lower triangular A with $AA^\top = \Sigma$

```
A ← 0  (d × d zero matrix)
for j = 1, ..., d
    for i = j, ..., d
        v_i ← Σ_ij
        for k = 1, ..., j − 1
            v_i ← v_i − A_jk A_ik
        A_ij ← v_i / √v_j
return A
```

Fig. 2.16. Cholesky factorization.