Inverting the Symmetrical Beta Distribution

PIERRE L'ECUYER and RICHARD SIMARD
Université de Montréal

We propose a fast algorithm for computing the inverse symmetrical beta distribution. Four series (two around $x = 0$ and two around $x = 1/2$) are used to approximate the distribution function and its inverse is found via Newton's method. This algorithm can be used to generate beta random variates by inversion and is much faster than currently available general inversion methods for the beta distribution. It turns out to be very useful for generating gamma processes efficiently via bridge sampling.

Categories and Subject Descriptors: G.4 [**Mathematical Software**]: Algorithm design and analysis; I.6 [**Computing Methodologies**]: Simulation and Modeling

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Random variate generation, inversion method, symmetrical beta distribution, quantiles

1. INTRODUCTION

Conceptually, the simplest method for generating a random variate X from distribution function F is by *inversion*: generate $U \sim U(0,1)$, i.e., from the uniform distribution over the interval $(0,1)$, and return $X = F^{-1}(U)$. This method is also generally preferred to other methods that transform U into X in a non-monotone way, because of its compatibility with variance reduction techniques such as common random numbers, antithetic variates, stratification, randomized quasi-Monte Carlo, etc. [Law and Kelton 2000; L'Ecuyer 2004]. However, F^{-1} is difficult to compute for certain distributions, such as the beta and gamma distributions, whose shapes depend on the parameters in a significant way. Currently available general approximation algorithms that work for all parameters of the distribution, e.g. [Moshier 2000; DiDonato and Morris 1992], are quite slow. For fixed parameters, one may construct a good approximation of F^{-1} for the specific parameters and write the corresponding algorithm, e.g., via Hermite interpolation, as in the *automatic methods* of Hörmann et al. [2004]. But this approach is definitely not convenient when the distribution parameters change for successive calls to the generator, which is often the case.

Authors' addresses: Pierre L'Ecuyer and Richard Simard, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal, H3C 3J7, Canada, e-mail: lecuyer@iro.umontreal.ca, simardr@iro.umontreal.ca.

In this paper we develop precise approximation methods for F and F^{-1} for the case of a *symmetrical beta* distribution with parameter α, over the unit interval $[0, 1]$. This distribution has density

$$f(x) = \frac{(x(1-x))^{\alpha-1}}{B(\alpha,\alpha)} \qquad \text{for } 0 \le x \le 1, \tag{1}$$

where B is the *beta function* defined as $B(\alpha,\nu) = \Gamma(\alpha)\Gamma(\nu)/\Gamma(\alpha+\nu)$, and Γ is the well-known gamma function. A random variable with this distribution can easily be rescaled and shifted to a symmetrical beta over an arbitrary interval $[a, b]$: just multiply by $(b-a)$ and add a. Other methods for generating symmetrical beta variates are described by Devroye [1986, pages 433–437], among which a rejection-based polar method proposed by Ulrich [1984] is the fastest and is quicker than our method when α is large. However, none of these methods is inversion. Moreover, Ulrich's method is valid only for $\alpha \ge 1/2$ and the other ones are also aimed at large values of α. Here we are interested as well in small values of α, because for the application that motivated this paper α is often much smaller than 1, and we rule out anything that is not inversion.

The symmetrical beta is important for a number of practical applications. Consider for example a stationary *gamma process* $\{G(t),\, t \ge 0\}$ with parameters (μ, ν). We have $G(0) = 0$ and the increments of G are independent over disjoint intervals and have the gamma distribution with mean μt and variance νt over any interval of length t. For a given t, let $X_1 = G(t/2)$ and $X_2 = G(t) - G(t/2)$, so $X_1 + X_2 = G(t)$. Since X_1 and X_2 are independent gamma random variables with the same parameters, we know that conditional on $X_1 + X_2$, $X_1/(X_1 + X_2)$ is a symmetrical beta random variable with parameter $\alpha = (\mu^2/\nu)t/2$; see, e.g., Hogg and Craig [1995]. This argument can be used recursively: given $G(0)$, $G(t/2)$, and $G(t)$, we have that $G(t/4)/G(t/2)$ and $G(3t/4)/(G(t) - G(t/2))$ are independent symmetrical beta random variables, and so on. This provides a method for generating a gamma process by successive refinements of the trajectory over some time interval $[0, T]$: generate first $G(T)$ from the appropriate gamma distribution, then $G(T/2)$, $G(T/4)$, $G(3T/4)$, $G(T/8)$, etc., by generating symmetrical beta random variables with parameters $\alpha = (\mu^2/\nu)t/2$, $(\mu^2/\nu)t/4$, $(\mu^2/\nu)t/8$, etc. In Avramidis and L'Ecuyer [2006], this is called *bridge sampling* of the gamma process.

An important advantage of this sampling method is that it concentrates most of the variance on the first few random variates that are generated, because these variates already sketch a good approximation of the trajectory. This opens the way to effective variance reduction by stratifying or applying randomized quasi-Monte Carlo methods for the uniforms used to generate these first few random variates. But for this to work effectively, the random variates must be generated by inversion. This was the original motivation for developing the methodology presented here. The idea of using bridge sampling to improve the effectiveness of quasi-Monte Carlo methods by reducing the effective dimension of the simulation problem was proposed by Caflisch and Moskowitz [1995] and Moskowitz and Caflisch [1996] for generating the path of a Brownian motion at a finite number of points. The method applies (at least in principle) to other types of Lévy processes as well [Fox 1999; Avramidis and L'Ecuyer 2006].

In the next section, we write the distribution function F as infinite series, using a different expansion in different subintervals of $[0,1]$, separating the cases for $\alpha \leq 1$ and $\alpha > 1$. In Section 3, we explain how we compute F^{-1} by finding a root of this expansion. In Section 4, we compare the speed of the new method with that of previously available general methods.

2. APPROXIMATING THE DISTRIBUTION FUNCTION

The symmetrical beta distribution function F over $[0,1]$ satisfies $F(1-x) = 1 - F(x)$, so it suffices to approximate it over the interval $[0,1/2]$ and use symmetry to compute F over $(1/2,1]$. In the remainder of the paper, we denote $v = 1/2 - u$ and $y = 1/2 - x$, where $u = F(x)$. All series used to approximate F at a given point are computed with *relative tolerance* $\epsilon = 10^{-15}$, which means that we neglect all terms smaller than ϵ times the current sum. We will examine the effect of this neglect on the error we make on F. Choosing a larger ϵ increases the speed, but not by much. For example, replacing $\epsilon = 10^{-15}$ by $\epsilon = 10^{-6}$ makes the computation 40% to 50% faster on the average (roughly) for α between 10^{-1} and 10^5, and less than 2% faster for $\alpha < 10^{-3}$.

2.1 $\mathbf{0 < \alpha \leq 1}$

For $0 < \alpha \leq 1$ and $0 \leq x \leq 1/2$, if we replace $(1-t)^{\alpha-1}$ by its binomial series and integrate term by term, we obtain

$$F(x) = \frac{1}{B(\alpha,\alpha)} \int_0^x [t(1-t)]^{\alpha-1}\, dt \tag{2}$$

$$= \frac{1}{B(\alpha,\alpha)} \int_0^x t^{\alpha-1} \sum_{j=0}^{\infty} \frac{(1-\alpha)_j t^j}{j!} dt$$

$$= \frac{1}{B(\alpha,\alpha)} \sum_{j=0}^{\infty} \frac{(1-\alpha)_j}{j!} \frac{x^{j+\alpha}}{(j+\alpha)}, \tag{3}$$

where $(b)_j$ is Pochhammer's symbol defined as $(b)_j = \Gamma(b+j)/\Gamma(b)$ when b is not 0 or a negative integer [Abramowitz and Stegun 1970]. This series has radius of convergence 1 and converges rapidly for x near 0, but more slowly for larger x. For $x \approx 1/2$, each term is approximately half the preceding one. To improve on this, we will use a different expansion when x is close to $1/2$.

Using the variable $y = 1/2 - x$ in Equation (1), we obtain

$$g(y) = \frac{\left(1/4 - y^2\right)^{\alpha-1}}{B(\alpha,\alpha)}$$

for $|y| \leq 1/2$ for the density as a function of y. We approximate $F(x)$ via the following series for $H(y) = 1/2 - F(x) = 1/2 - F(1/2-y)$:

$$H(y) = \frac{1}{B(\alpha,\alpha)} \int_0^y \left(1/4 - \tau^2\right)^{\alpha-1} d\tau,$$

$$= \frac{y}{4^{\alpha-1} B(\alpha,\alpha)} \sum_{j=0}^{\infty} \frac{(1-\alpha)_j}{j!} \frac{(4y^2)^j}{(2j+1)}. \tag{4}$$

This second series has convergence radius 1/2 and it converges rapidly when y is close to 0, i.e., for x close to 1/2. These two series are very similar and both converge fast. To compute $F(x)$ for a given x, we shall use series (3) for $0 \le x \le x_m$ and series (4) for $x_m < x \le 1/2$. The value of x_m is chosen such that when the first is evaluated at x and the second at $y = 1/2 - x$, both converge at the same speed and require approximately the same number of terms to obtain a given accuracy. We find that x_m is often close to 1/4.

To examine this more closely, let T_j and S_j denote the jth term of series (3) and (4), respectively, and let j_0 be the index of the first neglected term. From (3) and (4), we see that $0 < T_j < T_{j-1}$ and $0 < S_j < S_{j-1}$. We have for $x = y = 1/4$

$$\frac{T_j}{F(1/4)} < \frac{T_j}{T_0} = \frac{(1-\alpha)_j}{j!}\frac{\alpha x^j}{(j+\alpha)} < \frac{\alpha}{4^j(j+\alpha)} < \frac{1}{4^j(j+1)}$$

and

$$\frac{S_j}{H(1/4)} < \frac{S_j}{S_0} = \frac{(1-\alpha)_j}{j!}\frac{(4y^2)^j}{(2j+1)} < \frac{1}{4^j(2j+1)}.$$

If j_1 is the smallest j such that $\alpha/(4^j(j+\alpha)) < \epsilon$, then $T_j/T_0 < \epsilon$ for all $j \ge j_1$ and thus $j_0 \le j_1$. For $\epsilon = 10^{-15}$, we have $j_0 \le j_1 = 23$. Similarly, for $S_j/S_0 < \epsilon$, we find $j_0 \le j_1 = 23$. If we use series (3) for x close to 1/2, the same argument gives $j_1 = 45$ while series (4) requires only a few terms. So the use of (4) instead of (3) for x close to 1/2 makes an important difference in terms of speed.

The relative error E_1 on $F(x)$ when we neglect all terms T_j for $j \ge j_0$ in (3) satisfies

$$E_1 = \frac{\sum_{j=j_0}^{\infty} T_j}{F(x)} = \frac{1}{F(x)}\sum_{j=j_0}^{\infty}\frac{(1-\alpha)_j}{B(\alpha,\alpha)j!}\frac{x^{j+\alpha}}{(j+\alpha)} < \frac{1}{F(x)}\sum_{j=j_0}^{\infty}\frac{(1-\alpha)_{j_0}}{B(\alpha,\alpha)j_0!}\frac{x^{j+\alpha}}{(j_0+\alpha)}$$
$$= \frac{T_{j_0}}{F(x)}\sum_{j=0}^{\infty} x^j < \frac{\epsilon}{1-x} \le 2\epsilon \tag{5}$$

for all $x \le 1/2$. Similarly, for series (4), the relative error E_2 on $H(y)$ when we neglect all terms with index $j \ge j_0$ satisfies $E_2 = \sum_{j=j_2}^{\infty} S_j/H(y) < \epsilon/(1-4y^2) \le 4\epsilon/3$ whenever $|y| \le 1/4$. The relative error is thus bounded by 2ϵ in all cases, uniformly in α, for $\alpha \le 1$.

2.2 $\mathbf{1 < \alpha < 100000}$

The above two series are not suitable for $\alpha > 1$ because successive terms alternate in sign and they increase very fast in absolute value when α is large. As a consequence of numerical cancellation, both series very quickly lose precision for even moderately large α. The two series (3) and (4) can be written in terms of Gauss hypergeometric series ${}_2F_1(a,b;c;z)$, defined as [Abramowitz and Stegun 1970]

$${}_2F_1(a,b;c;z) = \sum_{j=0}^{\infty}\frac{(a)_j(b)_j z^j}{(c)_j j!} \tag{6}$$

so that the functions (3) and (4) are given by

$$F(x) = \frac{x^\alpha}{\alpha B(\alpha,\alpha)}\ {}_2F_1(\alpha, 1-\alpha; 1+\alpha; x) \tag{7}$$

and

$$H(y) = \frac{y}{4^{\alpha-1}B(\alpha,\alpha)}\ {}_2F_1(1-\alpha, 1/2; 3/2; 4y^2). \tag{8}$$

The hypergeometric series can be transformed into many mathematically equivalent forms using linear or quadratic transformations. Two forms that we have found useful are

$$F(x) = \frac{x^\alpha(1-x)^{\alpha-1}}{\alpha B(\alpha,\alpha)}\ {}_2F_1\left(1, 1-\alpha; 1+\alpha; \frac{-x}{1-x}\right) \tag{9}$$

and

$$H(y) = \frac{y(1-4y^2)^\alpha}{4^{\alpha-1}B(\alpha,\alpha)}\ {}_2F_1\left(1/2+\alpha, 1; 3/2; 4y^2\right). \tag{10}$$

Equations (9) and (10) are obtained from (7) and (8), respectively, by making use of the following identities for hypergeometric series

$$\begin{aligned} {}_2F_1(a,b;c;z) &= (1-z)^{-b}\ {}_2F_1\left(c-a, b; c; \frac{-z}{1-z}\right), \quad \text{for } |z|<1 \text{ and } \left|\frac{z}{1-z}\right|<1, \\ {}_2F_1(a,b;c;z) &= (1-z)^{c-a-b}\ {}_2F_1(c-a, c-b; c; z), \quad \text{for } |z|<1. \end{aligned}$$

Series (9) converges for $x < 0.5$ and its terms are decreasing (in absolute value), so it can be used to compute the distribution in [0, 0.5). However, for x approaching 0.5, its terms are close to 1 and its convergence becomes very slow, especially for large α; thus we shall use series (10) for x close to 0.5. However, while series (10) converges for all $y < 0.5$ and all its terms T_j are positive, they first grow very quickly as a function of j for small j and start to decrease at $j = j_m$ for some j_m. The larger term T_{j_m} may overflow the capacity of a floating-point number even for moderate α and y. This largest term T_{j_m} becomes larger as either y or α increases. So we use the series (10) only for small y. Thus to compute the cumulative probability for $1 < \alpha \le 100000$, we use (10) for x close to 1/2, and (9) for x close to 0. For x very close to 1/2, the use of series (10) instead of (9) makes a huge difference in terms of speed, since (9) may require more than a thousand terms to compute the cumulative distribution to 15 decimal digits of precision, while (10) needs only a few terms. Because the probability density is sharply peaked around $x = 1/2$ for large α, series (10) will be used for most values of u in $x = F^{-1}(u)$.

The relative error E_3 of series (9), when we neglect all terms T_j with $|T_j/F(x)| < \epsilon$ for $j \ge j_3$ may be bounded by a similar argument as in (5) and we obtain $E_3 = \sum_{j=j_3}^{\infty} |T_j/F(x)| < \epsilon(1-x)/(1-2x)$. As long as $x < 0.44$, we have $E_3 < 5\epsilon$. However, for $\alpha \approx 100000$, we use this series for $x < 0.4967$ and the error bound becomes $E_3 < 76\epsilon$. So for $10000 \le \alpha \le 100000$ and $0.44 < x < 0.4967$, our series may give only 14 decimals digits of precision. The relative error for series (10), when we neglect all terms $|S_j/H(y)| < \epsilon$ for $j \ge j_4$, is bounded similarly as in Subsection 2.1 above: $E_4 = \sum_{j=j_4}^{\infty} S_j/H(y) < \epsilon/(1-4y^2) < 4\epsilon/3$ for all $|y| \le 1/4$.

Computing $4^{\alpha-1}\mathbf{B}(\alpha,\alpha)$. Care must be taken in computing the expression $4^{\alpha-1}B(\alpha,\alpha)$ for $\alpha > 1$ in Equation (10). A naive calculation of the two factors separately (by taking logarithms for example) will give rise to catastrophic loss of precision for large α due to the difference of two large nearly equal quantities. Instead, the expression must be calculated as a whole. From Legendre's duplication formula for the gamma function, $\sqrt{\pi}\Gamma(2\alpha) = 2^{2\alpha-1}\Gamma(\alpha)\Gamma(\alpha+1/2)$, we get

$$4^{\alpha-1}B(\alpha,\alpha) = \frac{4^{\alpha-1}\Gamma(\alpha)\Gamma(\alpha)}{\Gamma(2\alpha)} = \frac{\sqrt{\pi}\,\Gamma(\alpha)}{2\,\Gamma(\alpha+1/2)}.$$

For $\alpha \geq 200$, we make use of the asymptotic expansion, valid for large α [Spanier and Oldham 1987, p. 416]

$$\frac{\Gamma(\alpha+1/2)}{\sqrt{\alpha}\,\Gamma(\alpha)} = 1 - \lambda + \frac{\lambda^2}{2} + \frac{5\lambda^3}{2} - \frac{21\lambda^4}{8} - \frac{399\lambda^5}{8} + \cdots$$

where $\lambda = 1/(8\alpha)$. If we neglect the $O(\lambda^5)$ terms, the relative error is smaller than 5×10^{-15} for $\alpha \geq 200$.

For $10 \leq \alpha < 200$, we use Gauss' hypergeometric series (6) for $z = 1$, which is absolutely convergent when c is neither 0 nor a negative integer and $c - a - b > 0$:

$${}_2F_1(a,b;c;1) = \sum_{j=0}^{\infty} \frac{(a)_j (b)_j}{(c)_j\, j!} = \frac{\Gamma(c)\Gamma(c-a-b)}{\Gamma(c-a)\Gamma(c-b)}.$$

Setting $a = b = -1/2$ and $c = \alpha - 1/2$ in the above equations, we obtain

$$\frac{\Gamma(\alpha+1/2)}{\Gamma(\alpha)} = \sqrt{\left(\alpha - \frac{1}{2}\right) {}_2F_1\left(-\frac{1}{2}, -\frac{1}{2}, \alpha - \frac{1}{2}, 1\right)}.$$

The series converges faster as α increases.

For $\alpha \leq 10$, we use the naive calculation by calling the function `lgamma` from the C standard mathematical library which returns the natural logarithm of the Gamma function. Computing separately the logarithms of $\Gamma(\alpha+1/2)$ and $\Gamma(\alpha)$, we lose at most 1 decimal digit of precision for α close to 10.

2.3 $\alpha >$ **100000**

For large α, the above series for the distribution functions $F(x)$ and $H(y)$ are inefficient and we use instead the normal approximation proposed by Peizer and Pratt [1968] for the beta distribution function $u = F(x) \approx \Phi(z)$, where Φ is the standard normal distribution function and

$$z = (2x-1)\left(\alpha - \frac{1}{3} + \frac{1}{40\alpha}\right)\sqrt{\frac{1 - \xi g(2x) - x g(2\xi)}{(2\alpha - 5/6)\, x\xi}} \tag{11}$$

$$g(x) = \frac{1 - x^2 + 2x\ln(x)}{(1-x)^2}$$

where $\xi = 1 - x$. The relative error estimated in Peizer and Pratt [1968] for $\alpha = 100000$ is smaller than 2.1×10^{-9} for all $x > 0.4912$ ($u > 10^{-15}$) and smaller than 10^{-6} for all $x > 0.4587$ ($u > 10^{-300}$). The error becomes smaller (and the approximation better) as α or x increases.

3. INVERSION

Once we have an efficient way of approximating $F(x)$ at any x, the next step is to select a method that can find a root x of $F(x) = u$ for any u in $[0, 1]$. General root finding methods for that purpose are discussed and compared in Devroye [1986], pages 31–35. The main ones are *bisection* (or *binary search*), the *secant method*, and *Newton's method*. The first two apply under more general conditions, but Newton's method is the most efficient when F is either concave or convex and the density is easy to compute, because it converges at a quadratic rate while the secant and bisection methods converge at a superlinear and linear rates, respectively. In our case, the distribution function in the interval $x \in (0, 0.5)$ is convex for $\alpha > 1$ and concave for $\alpha < 1$, and computing the density takes only a small fraction of the time required to compute the distribution function, so Newton's method is clearly the method of choice.

We thus compute $x = F^{-1}(u)$ at a given u using Newton's iterates

$$x_{n+1} = x_n - \frac{F(x_n) - u}{f(x_n)} \tag{12}$$

with our best guess of x as a starting point x_0 (and similarly for $y = H^{-1}(v)$). Since in the general case, Newton's method converges only when x_0 (or y_0) is close enough to the solution, a good starting point is essential for convergence and efficiency. In our case, since the probability density never vanishes in the open interval $x \in (0, 0.5)$, when x_0 is to the right [to the left] of the solution in $(0, 0.5)$ for $\alpha > 1$ [for $\alpha < 1$], Newton's method is guaranteed to converge to the solution.

In our implementation, we set the maximum number of iterations for Newton's method at 11 and iterate until either the difference of two successive iterates is small enough ($|x_{n+1} - x_n| < \epsilon$), or the maximum number of iterations has been reached (and similarly for y). According to our empirical investigations in all areas of the possible values of α and u, the method rarely needs more than 8 iterations to converge with $\epsilon = 10^{-15}$. If convergence has not occurred after 11 iterations, we simply call a bisection method. This method is much slower but is called extremely rarely (we found a couple of cases in the area where $u \approx 10^{-3}$ and $\alpha \approx 10^5$.

3.1 $\mathbf{0 < \alpha \leq 1}$

We would like to use series (3) for $0 \leq x \leq x_m$ and series (4) for $x_m < x \leq 1/2$, but we don't know the value of x beforehand. As a first guess of $y = 1/2 - x$, just to determine which series to use, we will take

$$\tilde{y}_0 = v\, 4^{\alpha-1} B(\alpha, \alpha), \tag{13}$$

obtained by considering only the first term in (4). This is a reasonable first guess since for $\alpha \leq 1$ and $y \leq 1/4$, the terms in (4) decrease monotonously and the ratio of the first two terms is $4y^2(1 - \alpha)/3 \leq (1 - \alpha)/12$. We will use series (4) when $\tilde{y}_0 \leq 1/4$, and (3) when $\tilde{y}_0 > 1/4$.

Taking the first two terms in (4), we obtain

$$4^{\alpha-1} B(\alpha, \alpha)\, v = y + 4(1 - \alpha)y^3/3.$$

The second term on the right is very small compared to the first and the two-term

approximation

$$y_0 = \frac{\tilde{y}_0}{1 + 4(1-\alpha)\tilde{y}_0^2/3}$$

will be even closer to y than $\tilde{y}_0$ in (13). We shall use it as the starting point for Newton's recursion

$$y_{n+1} = y_n - \frac{H(y_n) - v}{g(y_n)}$$

when $\tilde{y}_0 \le 1/4$.

Similarly, the terms of series (3) decrease monotonously. The ratio of the first two terms is $x\alpha(1-\alpha)/(1+\alpha) \le \alpha(1-\alpha)/4 \le 1/8$ for $x \le 1/4$. Taking the first two terms in (3), we get

$$\alpha B(\alpha,\alpha)\, u = x^\alpha + \frac{\alpha(1-\alpha)x^{\alpha+1}}{(1+\alpha)},$$

whose solution is

$$x_0 = \frac{\tilde{x}_0}{\left(1 + \tilde{x}_0\alpha(1-\alpha)/(1+\alpha)\right)^{1/\alpha}}$$

where $\tilde{x}_0 = (u\,\alpha\,B(\alpha,\alpha))^{1/\alpha}$ is the approximation obtained by taking a single term in (3). This x_0 is a very good estimate of x, especially when x is close to 0 or α is close to 0 or 1. We use it as the starting point for Newton's recurrence (12) to solve for x using (3) when $\tilde{y}_0 > 1/4$.

3.2 $1 < \alpha \le 100000$

Again, we would like to use series (9) for x closer to 0 and series (10) for x closer to 1/2. It turns out that the empirical curve $u_m(\alpha) = 1/(2.5 + 2.25\sqrt{\alpha})$ is a very good separator for the regions where the two series are most useful for $\alpha > 1$, since for $u > u_m(\alpha)$, the maximum term in series (10) is never much larger than 1 and there is no danger of overflow. Thus we use (9) for $0 \le u < u_m(\alpha)$ and (10) for $u_m(\alpha) \le u \le 0.5$.

Consider the first term of series (10) in the form $v\,4^{\alpha-1}B(\alpha,\alpha) = y(1-4y^2)^\alpha$. Since this series is used only for small y, $y_0 = v4^{\alpha-1}B(\alpha,\alpha)$ is a very good starting point for Newton's method.

Similarly, consider the first term of (9) in the form $u\alpha B(\alpha,\alpha) = x^\alpha(1-x)^{\alpha-1}$. For small u, $x_0 = u\alpha B(\alpha,\alpha)$ is a very good starting point, but not for large α and u close to u_m, where Newton's method may sometimes lead to an x outside the interval $[0, 0.5]$. In that case, we take instead $x_0 = 0.5 - \epsilon_1$ where $\epsilon_1 > 0$ is small enough so that x_0 is always to the right of the solution $x = F^{-1}(u)$. Since $F(x)$ is convex in x for $\alpha > 1$ and $x \in (0, 0.5)$, starting to the right of the solution x in (0, 0.5) guarantees convergence.

3.3 $\alpha > 100000$

Given the value of $z = \Phi^{-1}(u)$, where Φ^{-1} is the inverse of the standard normal distribution function, we solve equation (11) for x by fixed-point iterations of the

form $x_{n+1} = h(x_n)$ with the function h defined by

$$h(x) = 0.5 + \frac{z}{(2\alpha - 2/3 + 1/20\alpha)}\sqrt{\frac{(2\alpha - 5/6)\,x\xi}{1 - \xi g(2x) - xg(2\xi)}},$$

where $\xi = 1 - x$. Given an interval $I = [a, b]$ such that $h(x) \in I$ for all $x \in I$, if there exists a nonnegative constant $K < 1$ such that $|h'(x)| \leq K$ for all $x \in I$, then the fixed-point iterations converge to the unique solution $x^* \in I$. In our case, for $x \in [0.1, 0.9]$, we have $h'(x) \leq 0.75|z|/\sqrt{2\alpha - 2/3}$. Since $\alpha > 100000$, the slope $h'(x)$ is extremely small in all cases and convergence is very fast. The probability density having a very narrow peak centered at 0.5, the point $x_0 = 0.5$ is an excellent starting value for this iteration method, on average. For $\alpha \approx 100000$, we estimate that our program returns at least 9 decimal digits of precision for $u > 10^{-15}$, and at least 6.5 digits for $u > 10^{-300}$. For larger α, the normal approximation is even better.

4. SPEED COMPARISONS

4.1 Setting

Our symmetrical version is much faster than general inversion methods for two reasons. Because of the symmetry, we can find a simple expansion of the distribution function around $x = 1/2$ which converges faster than that around $x = 0$ when x is close to $1/2$. Furthermore, for $\alpha < 1$, the terms of the two series (around $x = 0$ and around $x = 1/2$) decrease rapidly and an approximation by the first two terms already gives an x that is very close to the solution. The starting point is in the region of quadratic convergence of Newton's method and a very few iterations suffice to obtain a very precise solution. Choosing a bad starting point for Newton's method may slow down the program by an order of magnitude for $\alpha < 1$ or else it may make the method diverge. For $\alpha \gg 1$, while a good starting point x_0 is hard to find unless u is small or close to $1/2$, it is not so important. For example, choosing $x = 1/2$ as a starting point slows down our program by a factor of 13.5% for $\alpha = 2$, 11% for $\alpha = 10$, 5% for $\alpha = 100$, and 40% for $\alpha = 100000$. However, the use of series (10) instead of (9) makes a huge difference in term of speed, since (9) may require several thousand terms to compute the cumulative distribution to double precision for x close to $1/2$, while (10) needs only a few terms.

We have implemented our algorithm in both C and Java, and made experiments to compare its speed with that of currently available inversion methods. The tests were run on a computer with an AMD Athlon XP 2800+ processor with clock speed of 2088 MHz running Red Hat Linux.

In C, the two best algorithms we know for inverting the general beta distribution are the one implemented in the Cephes math library by Moshier [2000], and the algorithm from DiDonato and Morris [1992] implemented in the DCDFLIB library [Brown et al. 1994]. We use the double precision version of both Cephes and DCDFLIB which returns 14–15 decimal digits of precision for the parameters $\alpha, \beta \leq 10000$. We estimate that our program returns 14–15 decimal digits of precision for $0.05 \leq \alpha \leq 100000$, and slightly less elsewhere. In Java, we have implemented the algorithm of Gautschi [1964a; 1964b] for $\alpha \leq 1000$, and the normal approximation of Peizer and Pratt [1968] for $\alpha > 1000$ to compute the general beta distribution

Table I. CPU time (seconds) to generate one million beta random variates with different methods and different values of α. LS (L'Ecuyer-Simard) refers to the algorithm proposed here.

	C			Java	
α	Cephes	DCDFLIB	LS	General	LS
10^{-9}	81.7	84.3	0.95	145.3	0.75
10^{-7}	83.9	85.5	0.96	146.1	0.75
10^{-5}	85.9	87.3	0.96	149.6	0.75
10^{-3}	84.9	86.7	0.86	172.7	0.86
10^{-1}	66.9	17.5	2.17	141.8	3.57
10^{1}	23.9	23.0	2.35	95.8	3.70
10^{3}	33.0	42.3	2.41	1071.2	3.64
10^{5}	196.1	45.7	2.55	4.08	4.16
10^{7}	851.9	49.2	0.58	3.78	0.83
10^{9}	1354.9	53.0	0.58	3.75	0.77

Table II. CPU time comparisons for simulating a gamma process at m observation points using bridge sampling with the general and symmetrical beta variate generators.

Number of observations m	Time (sec) General beta	Time (sec) Symmetrical beta	Speed ratio
2	1.4	0.12	10.8
4	3.9	0.18	20.6
8	9.5	0.35	26.2
16	21.7	0.63	33.4
32	46.9	1.0	45.9
64	97.7	1.5	62.4
128	206.0	2.4	85.5
256	430.1	3.7	114.0
512	883.7	6.2	142.0
1024	1772.1	10.6	167.2
2048	3488.3	19.1	182.7

function, and the algorithm of Moshier [2000] to compute its inverse.

4.2 Comparisons

Table I gives the CPU time to generate 10^6 beta random variates for various value of α and with different methods, in C and Java. In both languages, our implementation is much faster than the general methods.

4.3 Simulating a gamma process

Table II gives the CPU time it took to generate 10^4 independent copies of a gamma process with parameters $\mu = \nu = 1$ by bridge sampling, in Java. In Avramidis and L'Ecuyer [2006], a combination of this bridge sampling methodology with randomized quasi-Monte Carlo gave significant efficiency improvements when simulating gamma processes involved in pricing certain financial derivatives. The value of the process is generated at $m = 2^k$ equidistant observation times, for $k = 1, 2, \ldots, 11$. For each m, the value of α (the parameter of the symmetrical beta distribution) varies from $1/2$ and $1/2^k$. Thus, none of the methods described by Devroye [1986, pages 433–437] applies in this setting, even if we are ready to give up on inversion. Most of the CPU time here is spent generating the symmetrical beta variates. For each value of m, the table gives the time to generate the paths with the inversion

method for the general beta distribution, the time with our specialized method for the symmetrical beta, and the ratio of these two times (the improvement factor). For $m = 2048$, our method is about 180 times faster. By looking at the CPU time as a function of m, we see that its asymptotic increase appears linear. This is easy to explain: Generating one trajectory of the process requires one call to a gamma variate generator and $m - 1$ calls to the symmetrical beta variate generator. Half of these calls are with $\alpha = 1/2^k$, one quarter with $\alpha = 1/2^{k-1}$, and so on. But since the average time per call converges to a constant, say γ_0, when $\alpha \to 0$, the time for generating the entire process converges to m times γ_0 for large m. A good approximation of this constant γ_0 is given in the first line of Table I.

A computer code implementing the new algorithm in C is available from the author's web pages.

Acknowledgements

This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Grant No. ODGP0110050, NATEQ-Québec grant No. 02ER3218, and a Canada Research Chair to the first author. The authors thank Luc Devroye and two anonymous reviewers whose suggestions helped improve the paper.

REFERENCES

Abramowitz, M. and Stegun, I. A. 1970. *Handbook of Mathematical Functions.* Dover, New York.

Avramidis, T. and L'Ecuyer, P. 2006. Efficient Monte Carlo and quasi-Monte carlo option pricing with the variance-gamma model. *Management Science*. to appear.

Brown, B. W., Lovato, J., and Russell, K. 1994. Library of C routines for cumulative distribution functions, inverses, and other parameters. See `http://odin.mdacc.tmc.edu/anonftp/#DCDFLIB`.

Caflisch, R. E. and Moskowitz, B. 1995. Modified Monte Carlo methods using quasi-random sequences. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, H. Niederreiter and P. J.-S. Shiue, Eds. Number 106 in Lecture Notes in Statistics. Springer-Verlag, New York, 1–16.

Devroye, L. 1986. *Non-Uniform Random Variate Generation.* Springer-Verlag, New York.

DiDonato, A. R. and Morris, A. H. 1992. Significant digit computation of the incomplete beta function ratios. *ACM Transactions on Mathematical Software 18,* 3, 360–377.

Fox, B. L. 1999. *Strategies for Quasi-Monte Carlo.* Kluwer Academic, Boston, MA.

Gautschi, W. 1964a. Algorithm 222: Incomplete beta function ratios. *Communications of the ACM 7,* 3, 143–144.

Gautschi, W. 1964b. Certification of algorithm 222: Incomplete beta function ratios. *Communications of the ACM 7,* 3, 244.

Hogg, R. V. and Craig, A. F. 1995. *Introduction to Mathematical Statistics*, 5th ed. Prentice-Hall.

Hörmann, W., Leydold, J., and Derflinger, G. 2004. *Automatic Nonuniform Random Variate Generation.* Springer-Verlag, Berlin.

Law, A. M. and Kelton, W. D. 2000. *Simulation Modeling and Analysis*, Third ed. McGraw-Hill, New York.

L'Ecuyer, P. 2004. Quasi-Monte Carlo methods in finance. In *Proceedings of the 2004 Winter Simulation Conference*, R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, Eds. IEEE Press, Piscataway, New Jersey.

Moshier, S. L. 2000. Cephes math library. See `http://www.moshier.net`.

Moskowitz, B. and Caflisch, R. E. 1996. Smoothness and dimension reduction in quasi-Monte Carlo methods. *Journal of Mathematical and Computer Modeling 23*, 37–54.

Peizer, D. B. and Pratt, J. W. 1968. A normal approximation for binomial, F, beta, and other common related tail probabilities. *Journal of the American Statistical Association 63*, 1416–1456.

Spanier, J. and Oldham, K. B. 1987. *An Atlas of Functions*. Hemisphere Publishing Corporation, Washington.

Ulrich, G. 1984. Computer generation of distributions on the m-sphere. *Applied Statistics 33*, 158–163.