Rejection-Inversion to Generate Variates from Monotone Discrete Distributions

W. HÖRMANN and G. DERFLINGER

University of Economics and Business Administration Vienna

Department of Statistics, Augasse 2-6, A-1090 Vienna, Austria

e-mail: whoer@statrix2.wu-wien.ac.at

For discrete distributions a variant of rejection from a continuous hat function is presented. The main advantage of the new method, called *rejection-inversion*, is that no extra uniform random number to decide between acceptance and rejection is required which means that the expected number of uniform variates required is halved. Using rejection-inversion and a squeeze, a simple universal method for a large class of monotone discrete distributions is developed. It can be used to generate variates from the tails of most standard discrete distributions. Rejection-inversion applied to the Zipf (or zeta) distribution results in algorithms that are short and simple and at least twice as fast as the fastest methods suggested in the literature.

Categories and Subject Descriptors: G.3 [**Mathematics of Computing**]: Probability and Statistics–*random number generation*

General Term: Algorithms

Additional Terms: Rejection method, Zipf distribution, tail of Poisson distribution, universal algorithm, T-concave

---

1. INTRODUCTION

In random-variate generation literature (for good monographs see Devroye [1986] and Dagpunar [1988]) general methods for discrete distributions include two table methods (i.e. inversion by sequential or table-aided search and the alias method) and rejection from continuous or discrete dominating distributions. The advantages of rejection compared with the others are that no tables are necessary, that the method can cope with infinite tails, and that the generation time is not affected by the size of the domain of the distribution. On the other hand, rejection algorithms are often quite slow because a continuous majorizing function (also called *hat*) does not fit so well and that there is little choice between discrete hats. The relatively large expected number of uniform random variates required (above two for any standard rejection algorithm) is an additional drawback.

When developing a fast generator for the Zipf distribution we found a general principle, called *rejection-inversion*, that can improve the fit of the continuous hat of rejection algorithms. It can be applied to distributions with heavy tails. This is important because table methods are useless for this case and rejection-inversion halves the expected number of uniform random numbers required to generate one discrete variate, as no extra uniform random number to decide between acceptance and rejection is necessary. This is possible as the regions of acceptance and rejection are not separated by a horizontal line as in ordinary rejection algorithms but by a vertical line. Technically, it is necessary that the hat function $h(x)$ of the rejection algorithm has a known integral $H(y) = -\int_y^\infty h(x)\, dx$ and inverse integral $H^{-1}(y)$. For each integer k, let $p_k = P(X = k)$ where X denotes the discrete random variable we want to generate. Then Figure 1 shows the hat h and for an arbitrary point k, the probability p_k (illustrated by the height of the thick line), the region of rejection (between $k - 1/2$ and x_k), and the shaded region of acceptance (between x_k and $k + 1/2$). The main idea can be sketched as follows:

Algorithm Rejection-Inversion

Repeat

Generate a random variate X with density proportional to the hat $h(x)$.

Set $K \leftarrow \lfloor X + 1/2 \rfloor$.

Until $X \geq x_K$;

Return K.

Section 2 presents the technical details of rejection-inversion for arbitrary convex hat functions; Section 3 discusses the choice of optimal hats for a large class of distributions whereas Section 4 proves the validity of the squeeze introduced in Section 2. After the theoretical development of the main ideas, given in Sections 2 to 4, Sections 5 to 7, as an application of the theory, present descriptions and performance characteristics of rejection-inversion algorithms for a large class of monotone discrete distributions (Section 5), for Poisson tails (Section 6) and for the Zipf distribution (Section 7).

2. REJECTION-INVERSION

To generate random variates from a monotone discrete distribution with probabilities proportional to p_k for $k \geq m$ is of practical importance as this allows the generation from the tails of standard distributions. Using standard rejection from a continuous hat $h(x)$, it is suggested in the literature to choose the hat such that the histogram of the discrete distribution is below $h(x)$ i.e. $p_k \leq h(k-1/2)$ with $h(x)$ defined for $x \geq m-1/2$. As the evaluation of the probabilities p_k is time consuming in most cases, the speed of the rejection algorithm is improved if we can find an easy-to-compute squeeze s_k with $s_k \leq p_k$ for $k \geq m$. The standard rejection algorithm then works in the following way. The construct "While true do" here means that the indented set of instructions that follow is repeated indefinitely often. The statement "return K" exits this loop and the algorithm then outputs the value K.

Algorithm SR: Standard Rejection

While true do

Generate a random variate X with density proportional to h;

Generate a uniform random number U;

Set $K \leftarrow \lfloor X + 1/2 \rfloor$;

If $Uh(X) \leq s_K$ then return K;

else if $Uh(X) \leq p_K$ then return K.

An important measure for the quality of the rejection algorithm is the expected number of iterations α. For the above algorithm we have

$$\alpha = \frac{\int_{m-1/2}^{\infty} h(x)\, dx}{\sum_{m}^{\infty} p_k}.$$

The acceptance probability is $1/\alpha$.

Rejection-inversion lowers α and halves the number of uniform random numbers required. We need a hat $h(x)$ defined for $x \geq m - 1/2$ with $\int_{k-1/2}^{k+1/2} h(x)\, dx \geq p_k$ for all $k \geq m$. It is obvious that an easy to check sufficient condition for this is h convex and $p_k \leq h(k)$. Now we decide between acceptance and rejection by testing if the random variate X, which lies in the interval $(k - 1/2, k + 1/2)$, is left or right of the point x_k. This point, separating the region of acceptance (between x_k and $k + 1/2$) from the region of rejection (between $k - 1/2$ and x_k), can be computed easily. It is only necessary to know $H(x) = -\int_x^{\infty} h(x)\, dx$ and the inverse function H^{-1} (which is in any case necessary if we want to generate X by inversion). From the equation

$$\int_{x_k}^{k+1/2} h(x)\, dx = H(k + 1/2) - H(x_k) = p_k$$

we get $x_k = H^{-1}(H(k+1/2) - p_k)$. This gives the acceptance condition $X \geq H^{-1}(H(k+1/2) - p_k)$.

As m is the left end of the domain, it is possible for this point to make rejection impossible by just defining $h(x) = 0$ for $x < x_m$. Because then we have $X = H^{-1}(U)$ where U is uniformly distributed in the interval $(H(x_m), H(\infty) = 0)$, and the acceptance condition simplifies to $U \geq H(k+1/2) - p_k$. Figure 1 shows the curve $h(x)$, the region of acceptance with its probability represented by the shaded area, and the probability p_k represented by the thick line. If we can find a k_o such that $k - x_k$ is non-decreasing for $m \leq k \leq k_o$, then $m - x_m$ is a simple lower bound on $k - x_k$ for these values of k and can be used as a squeeze. It is also possible to use $m + 1 - x_{m+1}$ as a (second) squeeze for $k - x_k$ if $m + 1 \leq k \leq k_o$. For $k = m$ it is not a squeeze but this does not matter as rejection is impossible in this case. The advantage of the second squeeze is the better fit, but on the other hand it requires additional computations for the set-up. We use the second squeeze only for the Zipf distribution. The details of the new algorithm are as follows. The choice of the hat $h(x)$ and its integral $H(x)$ are discussed in Section 3.

Algorithm RI: Rejection-Inversion

[Set-up]

Compute and store $y_m \leftarrow H(m+1/2) - p_m$, $x_m \leftarrow H^{-1}(y_m)$ and determine k_o.

[Generator]

While true do

  Generate a uniform random number U;

  Set $U \leftarrow U y_m$;

  Set $X \leftarrow H^{-1}(U)$;

  Set $K \leftarrow \lfloor X + 1/2 \rfloor$;

If ($K \leq k_o$ and $K - X \leq m - x_m$) then return K;

Else if $U \geq H(K + 1/2) - p_K$ then return K.

What are the performance characteristics of Algorithm RI? For fixed h, the expected number of iterations

$$\alpha = \frac{p_m + \int_{m+1/2}^{\infty} h(x)\,dx}{\sum_m^{\infty} p_k}$$

is lower than for Algorithm SR as there is no rejection possible for m. In addition the condition $p_k \leq h(k)$ allows a better fit for h than $p_k \leq h(k - 1/2)$. The most important advantage of Algorithm RI is that the number of uniforms required for one iteration is one compared with two for Algorithm SR. This property is the reason why we suggest the name rejection-inversion for the new method. With no squeeze, the numbers of evaluations of p_k and of H^{-1} per iteration equal 1 for both methods if the generation of X is done by inversion in Algorithm SR. For Algorithm RI one additional evaluation of H, and for Algorithm SR one of h are necessary. These theoretical performance measures show that Algorithm RI can result in simple and comparably fast generators as long as we can use the simple squeeze and can find a class of convex hat functions with simple and invertible integrals H. Of practical importance is $h(x) = a/(b + x)^q$ with $q > 1$ (with the simplest case $q = 2$). Considering the equivalent form $\tilde{a}/(1 - \tilde{b}x/q)^q$, we can take the limit as $q \to \infty$ to get the hat $h(x) = \tilde{a}\exp(\tilde{b}x)$. Table 1 contains the information necessary to use these functions as hat functions in Algorithm RI, only the choice of (a, b) and the last two columns will be explained below.

Table 1: Classes of hat functions

q	$h(x)$	$H(x)$	$H^{-1}(x)$	T_c	c
$1 < q < \infty$	$\frac{a}{(b+x)^q}$	$\frac{a/(-q+1)}{(b+x)^{q-1}}$	$-b + \left(\frac{a}{x(1-q)}\right)^{(1/(q-1))}$	$-x^c$	$-1 < c < 0$
2	$\frac{a}{(b+x)^2}$	$\frac{-a}{b+x}$	$-b - \frac{a}{x}$	$-1/\sqrt{x}$	$-1/2$
∞	ae^{bx}	ae^{bx}/b	$\log(xb/a)/b$	$\log(x)$	0

3. THE CHOICE OF AN OPTIMAL HAT

To use Algorithm RI with one of the hat functions given in Table 1, it is necessary to choose the parameters a and b. As this is easier in the case of continuous random variables with density $f(x)$ we shall discuss this case first. The problem becomes much clearer if we use the transformations T_c, defined by $T_c(x) = -x^c$ for $-1 < c < 0$ and $T_0(x) = \log(x)$, given in Table 1. For $c = -1/q$, T_c transforms the corresponding hat function into a straight line. Therefore – in a transformed scale – the hat function is a tangent line touching the transformed density $T_c(f(x))$ in a single point of contact. Choosing an arbitrary point of contact, h remains a hat function for $f(x)$ as long as the transformed density $T_c(f(x))$ is concave. For a fixed c the class of all densities with this property is called T_c-concave and are discussed in detail in Hörmann [1995]. To optimize the choice of a and b for a class of hat functions, it is then enough to optimize the choice of the point of contact. Surprisingly, it turns out that – although the transformations and corresponding hat functions are totally different regarding their behavior at zero and towards infinity – the optimal point of contact does not depend on the choice of the transformation T_c. Using this optimal point of contact, α is uniformly bounded for all distributions that are T_c-concave for an arbitrary but fixed c. We have the following:

THEOREM 1: Given a transformation $T_c = -x^c$ for $-1 < c < 0$ or $T_0 = \log(x)$ with the corresponding hat function $h(x) = a/(b + x)^q$ with $q = -1/c$ or $h(x) =$

$a \exp(bx)$ for $c = 0$, and a monotone continuous T_c-concave density $f(x)$ defined for $x \geq m$, which is twice differentiable with the possible exception of countably many isolated jumps of f', the area below the hat (i.e. α) is minimized if the touching point x_o is chosen such that

$$f(x_o)(x_o - m) = \max_{x \geq m} (f(x)(x - m)) \tag{1}$$

This means that we take the parameters a, b of $h(x)$ such that

$$h(x_o) = f(x_o) \quad \text{and} \quad h'(x_o) = \frac{f(x_o)}{m - x_o} \tag{2}$$

This implies that $h'(x_o) = f'(x_o)$ for the case that $f'(x_o)$ exists.

For all T_c-concave densities with the hat chosen according to (1) and (2) α is bounded by $\alpha \leq (1 + c)^{1/c}$ for $c < 0$ and $\alpha \leq e$ for $c = 0$.

Proof: In a first step we assume that f is twice differentiable. It is convenient to write the hat function touching in the point x_1 in the form: $T_c^{-1}(g(x_1) + g'(x_1)(x - x_1))$ where $g(x) = T_c(f(x))$. We start with $T_c(x) = -x^c$ and minimize the area below the hat

$$A(x_1) = \int_m^\infty (-g(x_1) - g'(x_1)(x - x_1))^{1/c} \, dx = \frac{(-g(x_1) - g'(x_1)(m - x_1))^{1/c+1}}{(1/c + 1)g'(x_1)}.$$

Differentiation and some simplification give

$$\frac{dA}{dx_1} = \frac{g''(x_1)(-g(x_1) - g'(x_1)(m - x_1))^{1/c}}{g'(x_1)^2(1/c + 1)} \left[g(x_1) - g'(x_1)(m - x_1)/c\right].$$

As g is concave it is clear that the fraction is always greater or equal 0. Differentiation of the part in square brackets yields $g'(x_1)(1/c + 1) - g''(x_1)(m - x_1)/c \geq 0$ which shows that A has its global minimum at the point x_o satisfying $m - x_o = \frac{cg(x_o)}{g'(x_o)}$. Substituting the definition of g gives

$$m - x_o = \frac{f(x_o)}{f'(x_o)} \tag{3}$$

Considering the argument above together with differentiation of $(x-m)f(x)$ shows that $(x-m)f(x)$ has for $x \geq m$ one global maximum and thus (3) and (1) are equivalent.

To compute the value of $A(x_o)$ it is best to substitute (3) into the formula for A which results in $A(x_o) = (1+c)^{1/c} f(x_o)(x_o - m)$. A simple consideration shows that for any monotone density $f(x)(x-m) \leq 1$ which proves the bound of α.

For $T_c(x) = \log(x)$ a similar computation results in the same condition for x_o. For $A(x_o)$ we obtain $A(x_o) = \lim_{c \to 0}(1+c)^{1/c} f(x_o)(x_o - m) = e\, f(x_o)(x_o - m)$, which completes the proof for twice differentiable f.

If f' is discontinuous in isolated points, the choice of x_o according to (1) follows from the above proof as any f is the limit of a uniformly convergent sequence f_n of twice differentiable functions with optimal point of contact $(x_o)_n$. Obviously $x_o = \lim_{n \to \infty}(x_o)_n$. Therefore (2) can be used to determine the parameters a and b. Using the sequence f_n it is easily seen that the bound for the optimal α remains valid. $\square$

Now we can return to our main topic: The generation of discrete random variates. First it is necessary to define $\tilde{g}(x)$ as the function which has as graph the polygon with vertices $(k, T_c(p_k))$. Then a discrete random variate is called T_c-concave if this polygon $\tilde{g}$ is concave. It is possible to apply the result of Theorem 1 to $\tilde{f} = T_c^{-1}(\tilde{g})$ which is a continuous interpolation of the points (k, p_k). Thus it is not difficult to prove:

Corollary 1: Given a transformation $T_c = -x^c$ for $-1 < c < 0$ or $T_0 = \log(x)$ with the corresponding hat function $h(x) = a/(b+x)^q$ with $q = -1/c$ or $h(x) = a\exp(bx)$ for $c = 0$, a discrete T_c-concave random variate with probabilities p_k and the minimal T_c-concave continuous interpolation $\tilde{f}$ (for the construction see the definition above), the area below the hat of Algorithm RI (i.e. α) is minimized if the touching point x_o is chosen such that

$$\tilde{f}(x_o) * (x_o - (m + 1/2)) = \max_{x \geq m+1/2} \tilde{f}(x) * (x - (m + 1/2)). \tag{4}$$

This means that we take the parameters a, b of $h(x)$ such that

$$h(x_o) = \tilde{f}(x_o) \quad \text{ and } h'(x_o) = \frac{\tilde{f}(x_o)}{m + 1/2 - x_o}. \tag{5}$$

For all T_c-concave discrete distributions with h chosen according to (4) and (5) we have: $\alpha \leq (1+c)^{1/c}$ for $c < 0$ and $\alpha \leq e$ for $c = 0$.

PROOF: As in Algorithm RI no rejection is possible for the case $k = m$ we are only interested in $\tilde{f}(x)$ for $x \geq m + 1/2$. $\tilde{f}$ is the infimum over all possible continuous, monotone and T_c-concave interpolations of the p_k. It is easy to see that $\tilde{f}$ fulfills the conditions of Theorem 1 and that $p_m + \int_{m+1/2}^{\infty} \tilde{f}(x)\, dx \leq 1$. Applying Theorem 1 thus completes the proof. □

REMARK 1: Corollary 1 can be used in a simple way to determine the optimal hat function for a discrete distribution: First use a search procedure to determine the point of contact k_o such that $p_{k_o}(k_o - (m + 1/2)) = \max_k(p_k(k - (m + 1/2)))$. This is simple, as the argument below (3) implies that $p_k(k - (m + 1/2))$ has only one local maximum. Then compute the optimal $h'(k_o)$ according to Corollary 1 and the parameters a and b from the equations for $h'(k_o)$ and $h(k_o)$. As k_o need not be equal to x_o it is necessary to test if $h(k_o - 1) \leq p_{k_o - 1}$ (in this case x_o is in the interval $(k_o - 1, k_o)$ and a and b must be chosen such that h connects the points $(k_o - 1, p_{k_o-1})$ and (k_o, p_{k_o})) or if $h(k_o + 1) \leq p_{k_o+1}$ (in this case x_o is in the interval $(k_o, k_o + 1)$ and a and b must be chosen such that h connects the points (k_o, p_{k_o}) and $(k_o + 1, p_{k_o+1})$).

REMARK 2: It is obvious that, following the guidelines of Remark 1, the determination of the optimal hat function can become very time consuming. Therefore the following simple idea to find a hat very close to the optimal hat is of practical importance. Based on (2) in Theorem 1 the optimal point of contact for a continuous distribution x_o must have the property that $f'(x_o) = f(x_o)/(m - x_o)$. For discrete distributions it is therefore possible to find a point x_o close to the optimal k_o if we take some natural continuous interpolation of the points (k, p_k) and solve the equation $p(x+1) - p(x) = p(x)/(m - x)$ or equivalently

$$\frac{p(x+1)}{p(x)} = 1 + \frac{1}{m-x}.$$

For discrete distributions where the p_k have a simple recurrence the above equation can be solved analytically (as is the case, e.g., for Poisson, binomial and hypergeometric tails) or by some numerical approximation. To find a hat close to optimal it is enough to compute this (possibly approximate) solution x_o of the equation and to take $k_o = \lfloor x_o + 1 \rfloor$. The hat h is then chosen such that $h(k_o) = p_{k_o}$ and $h(k_o - 1) = p_{k_o - 1}$

4. THE SQUEEZE

To show the validity of the squeeze used in Algorithm RI we need the following:

THEOREM 2: Given a discrete distribution, which for a fixed c is T_c-concave, and a hat function $h(x) = a/(b+x)^q$ with $q = -1/c$ or $h(x) = a \exp(bx)$ for $c = 0$ touching the distribution in an arbitrary integer point k_o we have:

$$k - x_k \text{ is non-decreasing for } m \le k \le k_o$$

where $x_k = H^{-1}(H(k+1/2) - p_k)$ denotes the point that separates the region of acceptance and rejection for the integer k.

PROOF: First we consider the special case that $h(k) = p_k$ for $k \geq m$:

$\sigma = \sigma(k) = k - x_k$ is defined implicitly by

$$\int_{k-\sigma}^{k+1/2} h(u)\, du - h(k) = 0. \tag{6}$$

In a more complicated but equivalent form we can rewrite (6) as

$$-\int_{k-1/2}^{k-\sigma} \int_0^{h(k)} dt\, du + \int_{k-\sigma}^{k} \int_{h(k)}^{h(u)} dt\, du - \int_k^{k+1/2} \int_{h(u)}^{h(k)} dt\, du = 0. \tag{7}$$

Differentiating (6) implicitly after k (which is no longer restricted to integer values) gives

$$h(k-\sigma)\frac{d\sigma}{dk} = -h(k+1/2) + h(k-\sigma) + h'(k) = \int_{k-\sigma}^{k+1/2} \phi(u)\, du - \phi(k), \tag{8}$$

with $\phi(k) = -h'(k) > 0$. The right side of (8) can be rewritten analogously to the left side of (6) as

$$h(k-\sigma)\frac{d\sigma}{dk} = -\int_{k-1/2}^{k-\sigma} \int_0^{\phi(k)} dt\, du + \int_{k-\sigma}^{k} \int_{\phi(k)}^{\phi(u)} dt\, du - \int_k^{k+1/2} \int_{\phi(u)}^{\phi(k)} dt\, du.$$

The substitution $t = \phi(h^{-1}(s)) = -1/(h^{-1})'(s)$, $dt = (-1/(h^{-1})'(s))'ds$ together with the notation $\lambda(s) = (-1/(h^{-1})'(s))'$ implies that

$$\begin{aligned} \frac{d\sigma}{dk} &= \frac{1}{h(k-\sigma)} \left[-\int_{k-1/2}^{k-\sigma} \int_0^{h(k)} \lambda(s)\, ds\, du + \right. \\ & \left. \int_{k-\sigma}^{k} \int_{h(k)}^{h(u)} \lambda(s)\, ds\, du - \int_k^{k+1/2} \int_{h(u)}^{h(k)} \lambda(s)\, ds\, du \right]. \end{aligned} \tag{9}$$

A standard calculation shows that $\lambda'(t) \geq 0$ for all classes of hat functions of Table 1. Thus it is easy to see that replacing $\lambda(s)$ by $\lambda(h(k))$ in (9) gives a lower bound for σ'. Together with (7) we have $\sigma' \geq 0$ which completes the proof for the special case.

For the general case we define $D(k) = h(k) - p(k)$ where $p(k)$ is a continuous, differentiable interpolation of the points (k, p_k), and have instead of (6)

$$\int_{k-\sigma}^{k+1/2} h(u)\, du - h(k) + D(k) = 0.$$

Following the argumentation above we get

$$\frac{d\sigma}{dk} = (9) - \frac{D'(k)}{h(k-\sigma)}$$

which completes the proof as $D'(k) \le 0$ for $m \le k \le k_o$. □

5. UNIVERSAL ALGORITHMS

Putting together the results of Sections 2 to 4 it is now no problem to describe a uniformly fast universal algorithm for the tail of a log-concave discrete distribution with probabilities proportional to p_k and mode m. ($p(x)$ denotes a simple continuous interpolation of the points (k, p_k)). For the functions needed for Algorithm RI we take $h(x) = a\exp(bx)$, $H(x) = a\exp(bx)/b$, $H^{-1}(x) = \log(xb/a)/b$. Remark 2 of Section 3 is used to compute nearly optimal parameters a and b for the hat.

Algorithm RILC: Rejection-Inversion for log-concave distributions

[Set-up]

Find an (approximate) solution x_o for $p(x+1)/p(x) = 1 + 1/(m-x)$.

Compute $k_o \leftarrow \lfloor x_o + 1 \rfloor$, $b \leftarrow \log(p_{k_o}) - \log(p_{k_o-1})$, $a \leftarrow p_{k_o}\exp(-bk_o)$,

$y_m \leftarrow a\exp(b(m+1/2))/b - p_m$, $x_m \leftarrow \log(y_m b/a)/b$.

[Generator]

While true do

  Generate a uniform random number U;

  Set $U \leftarrow U y_m$;

  Set $X \leftarrow \log(Ub/a)/b$;

  Set $K \leftarrow \lfloor X + 1/2 \rfloor$;

  If ($K \le k_o$ and $K - X \le m - x_m$) then return K;

  Else if $U \ge a\exp(b(K+1/2))/b - p_K$ then return K.

For the tail of an arbitrary discrete T_c-concave distribution everything is very similar but with $h(x) = a/(b+x)^q$, $H(x) = (a/(-q+1))/(b+x)^{q-1}$ and $H^{-1}(x) = -b + (a/(x(1-q)))^{(1/(q-1))}$. Due to the easy-to-compute forms of H and H^{-1} the algorithm executes fastest if we take $q = 2$ which is possible if the distribution is at least $T_{-1/2}$-concave.

Algorithm RITC: Rejection-Inversion for T_{c_o}-concave distributions

[Set-up]

Take a fixed q with $1 < q \le -1/c_0$.

Define $H(x) = (a/(-q+1))/(b+x)^{q-1}$ and $H^{-1}(x) = -b + (a/(x(1-q)))^{(1/(q-1))}$ as in-line functions or macros.

Find an (approximate) solution x_o for $p(x+1)/p(x) = 1 + 1/(m-x)$.

Compute $k_o \leftarrow \lfloor x_o + 1 \rfloor$, $\tilde{p} \leftarrow (p_{k_o-1}/p_{k_o})^{1/q}$, $b \leftarrow (\tilde{p}(k_o-1) - k_o)/(1-\tilde{p})$,
$a \leftarrow p_{k_o}(b+k_o)^q$, $y_m \leftarrow H(m+1/2) - p_m$, $x_m \leftarrow H^{-1}(y_m)$.

[Generator]

While true do

Generate a uniform random number U;

Set $U \leftarrow U y_m$;

Set $X \leftarrow H^{-1}(U)$;

Set $K \leftarrow \lfloor X + 1/2 \rfloor$;

If ($K \le k_o$ and $K - X \le m - x_m$) then return K;

Else if $U \ge H(K+1/2) - p_K$ then return K.

Using Algorithm RILC or RITC we can generate the tails of most of the classical discrete distributions including the Poisson, binomial, hypergeometric and negative binomial distributions which are all log-concave. Even more important is the fact that heavy tails can be generated as well using Algorithm RITC with $q = 2$ for subquadratic tails and with $q < 2$ for heavier ones. This is of practical importance as for the modeling of the input distributions of simulation models the sub-exponential tails of the classical discrete distributions are often too thin and little advice is given in simulation literature how to generate random variates from discrete distributions with thicker tails.

The two algorithms could also be used as the main building block of a program that generates random variates from arbitrary discrete T_c-concave distributions (for a universal algorithm for discrete log-concave distributions see for example Hörmann [1994]) as these distributions are unimodal and can thus be decomposed at the mode into two monotone parts.

6. POISSON TAILS

We tested Algorithm RILC for the right tail of the Poisson distribution with mean μ and cut-off point m (i.e. $X \geq m$). The computation of x_o reduces to the solution of a quadratic equation: $x_o = (m+\mu)/2 + \sqrt{(m-\mu)^2/4 + m + 1}$. It is easy to see that for the Poisson distribution α is bounded by the α for the positive standard normal distribution, which is $\sqrt{2e/\pi} = 1.315\ldots$.

We know that comparative timings are only of limited evidence as they depend heavily on the speed of the uniform generator used and on the implementation of the evaluation of the p_k's. Nevertheless, we include in Table 2 the comparison of the speed of our C implementations of Algorithm RILC optimized for the tail of the Poisson distribution (using a multiple recursive linear congruential generator with module $2^{31} - 1$ as uniform generator) with the speed of a simple tail generation method for the Poisson distribution using rejection from a geometric hat, which was suggested by Dagpunar [1988] (p. 148). There the cut-off point is called m as well. The execution times given in Table 2 are the averages from 10^6 calls to the generator (the overhead of the measurement program was subtracted). The first decimal place of the given numbers is meaningful but not necessarily correct.

We think that this comparison demonstrates the potential of the rejection-inversion method. The good speed is due to the use of the squeeze, to the reduced number of uniform variates required, and to the good fit of the nearly optimal hat computed in the set-up. This set-up is slower than the set-up of Dagpunar's algorithm but still quite fast.

7. A NEW GENERATOR FOR THE ZIPF DISTRIBUTION

The Zipf distribution – also called (Riemann) zeta or discrete Pareto distribution – is frequently used in linguistics and other social sciences to model the number of occurrences of certain events. (See Johnson et al. [1992], Dagpunar [1988], and references given there.) In simulation studies it could be well used to model the tail part of discrete input distributions, at least to carry out a sensitivity analysis of how strongly the tail shape influences the simulation results. We will consider the two parameter generalization as defined in Dagpunar [1988] with the unnormalized probability function

$$p_k = \frac{1}{(v+k)^q} \qquad (k = 0, 1, \ldots),$$

where $q > 1$, $v > 0$. It is clear that $h(x) = 1/(v+x)^q$ is a hat function that touches at every integer and that the distribution is T_c-concave with respect to this hat function. Therefore Algorithm RI and the second version of the squeeze described before the algorithm can be used. Theorem 2 implies that the squeeze is valid for all $k \geq m+1$ as ∞ is a touching point as well. We include a formal description of our Zipf-generator as this facilitates its application. One practical problem can occur for small values of q if the random number X generated by inversion is larger than the largest representable integer i_{max}. Therefore X is not generated in the interval

(x_m, ∞) but in the interval $(x_m, i_{max} + 1/2)$ which means that U is uniformly distributed over $(H(x_m), H(i_{max} + 1/2))$.

ALGORITHM ZRI: (Zipf-distribution generated with rejection-inversion)

[Set-up]

Let q and v be the parameters of the Zipf distribution. Compute and store $1 - q$, $1/(1-q)$ and define $H(x) \leftarrow \exp((1-q)*\log(v+x))*(1/(1-q))$ and $H^{-1}(x) \leftarrow -v+\exp((1/(1-q))*\log((1-q)*x))$ as in-line functions or macros. Compute and store $H(x_0) \leftarrow H(1/2)-\exp(\log(v)*(-q))$, $s \leftarrow 1-H^{-1}(H(3/2)-\exp(\log(v+1)*(-q)))$ and $H(i_{max} + 1/2)$.

[Generator]

While true do

Generate a uniform random number U;

Set $U \leftarrow H(i_{max} + 1/2) + U * (H(x_0) - H(i_{max} + 1/2))$;

Set $X \leftarrow H^{-1}(U)$;

Set $K \leftarrow \lfloor X + 1/2 \rfloor$;

If $(K - X \leq s)$ then return K;

Else if $(U \geq H(K + 1/2) - \exp(-\log(v + K) * q))$ then return K.

To evaluate the quality of Algorithm ZRI it is important to discuss the value of α and the expected number of operations necessary to generate one random number from the Zipf distribution. We have:

THEOREM 3: The expected number $\alpha(q, v)$ of iterations for Algorithm ZRI is uniformly bounded for all values of $q > 1$ and of $v > 0$. The smallest upper bound is given by the maximum of the function

$$\phi(t) = (1 + e^{-t/2}/t)(1 - e^{-t}), \qquad t > 0.$$

REMARK: By numerical methods one finds that $\phi(t)$ takes its maximum value 1.023775 at $t = 2.111114$. Therefore $\alpha < 1.023775$ for all $q > 1$, $v > 0$.

PROOF: First we show that

$$\lim_{v\to 0} \alpha(q,v) = 1, \qquad q > 1, \tag{10}$$

$$\lim_{v\to \infty} \alpha(q,v) = 1, \qquad q > 1. \tag{11}$$

For (10) consider $\lim_{v\to 0} \alpha(q,v) = \lim_{v\to 0} \frac{1+v^q \int_{v+1/2}^{\infty} x^{-q}\, dx}{1+v^q \zeta(q,v+1)} = 1$.

To show (11) we use $\int_{v+1/2}^{\infty} x^{-q}\, dx < \zeta(q, v+1/2) < \zeta(q,v)$ twice:

$$\alpha(q,v) = 1 + \frac{\int_{v+1/2}^{\infty} x^{-q}\, dx - \zeta(q,v+1)}{\zeta(q,v)} < 1 + \frac{1}{v^q \zeta(q,v)} <$$

$$< 1 + \frac{1}{v^q \int_{v+1/2}^{\infty} x^{-q}\, dx} = 1 + \frac{(q-1)(v+1/2)^{q-1}}{v^q}.$$

For $v \to \infty$ the fraction converges towards zero which proves (11).

Because of (10) and (11), for fixed $q > 1$ the function $\alpha(q,v)$, which is obviously continuous, must have a conditional maximum at a point $v = v_m(q)$: $\max_v \alpha(q,v) = \alpha(q, v_m(q))$. As $\alpha(q,v)$ is differentiable with respect to v we have $\frac{\partial}{\partial v}\alpha(q,v)\big|_{v=v_m(q)} = 0$. An easy calculation shows the equivalence of this condition to $\alpha(q, v_m(q)) = \alpha(q+1, v_m(q))$ which implies $\alpha(q, v_m(q)) \le \alpha(q+1, v_m(q+1))$. Thus for any starting value the sequence

$$\max_v \alpha(q+k, v), \qquad k = 0, 1, 2, \ldots, \tag{12}$$

is monotone increasing.

Now, using the inequalities $\exp(1/(1+x)) < 1 + 1/x < \exp(1/x)$, for $x > 0$, which can be easily verified, we construct an upper bound of α:

$$\alpha = \frac{1 + \frac{v}{q-1}\left(1+\frac{1}{2v}\right)^{1-q}}{\sum_{j=0}^{\infty}\left(1+\frac{j}{v}\right)^{-q}} < \frac{1 + \frac{v}{q-1}\exp\left(\frac{1-q}{2v+1}\right)}{\sum_{j=0}^{\infty}\exp\left(\frac{-jq}{v}\right)} =$$

$$= \left(1 + \frac{v}{q-1}\exp\left(\frac{1-q}{2v+1}\right)\right)\left(1 - \exp\left(\frac{-q}{v}\right)\right).$$

We substitute $v = (q-1)/t$ and consider α as a function of q and t:

$$\alpha(q,t) < \left(1 + \frac{1}{t}\exp\left(\frac{-t}{2+\frac{t}{q-1}}\right)\right)\left(1 - \exp\left(-t\left(1+\frac{1}{q-1}\right)\right)\right)$$

Specifying this for the sequence (12) of the conditional maxima and taking the limit for $k \to \infty$ gives

$$\lim_{k\to\infty} \max_t \alpha(q+k,t) \le \max \phi(t).$$

It is evident that $\lim_{t\to\infty} \phi(t) = 1$. That $\lim_{t\to 0} \phi(t) = 1$ can be easily shown. Thus, as $\phi(t)$ is continuous, it is bounded. It follows that $\max \phi(t)$ is an upper bound of α. In order to prove that it is the smallest upper bound we consider

$$\lim_{q\to\infty} \alpha(q,t) = \lim_{q\to\infty} \frac{1 + \frac{1}{t}\left(1+\frac{t}{2(q-1)}\right)^{1-q}}{\sum_{j=0}^{\infty}\left(1+\frac{jt}{q-1}\right)^{-q}} = \frac{1+\exp(-\frac{t}{2})/t}{\sum_{j=0}^{\infty}\exp(-jt)} = \phi(t).$$

This completes the proof of the theorem. □

The expected number of power operations (implemented with one call to exp and one to log) is well below 1.1 for all parameter combinations we tried, due to the good fit of the squeeze. The theoretical properties (especially the very low α) and the simple and short code show that Algorithm ZRI is really well suited for the generation of random variates from the Zipf distribution. This fact becomes even more apparent if we compare the performance of ZRI with algorithms suggested in the literature. First, it is important to note that the two standard methods for the generation of discrete random variates (inversion and the alias method) have several disadvantages for the Zipf distribution. It is necessary for both methods to compute the normalization constant which includes the evaluation of the ζ-function which is time consuming and requires care. For small values of q both standard methods are useless. As the tails of the distribution are too heavy the expected number of comparisons when using inversion by sequential search is unbounded for $q \le 2$. For the alias method heavy-tailed distributions cause problems as the cut off point where the tail probability is smaller than a threshold (e.g. 2^{-32} for uniform random number generators working with 32 bit integers) can be very large which means that millions of probabilities must be computed and stored in a set-up step. For the Zipf distribution with $v = 1$ and $q = 2$ the cut off point is about $2 \cdot 10^9$ which shows that the alias method is out of question for low values of q.

Table 2: Average execution times in μ-seconds

tail of Poisson distribution with mean μ							
μ	10		100		1000		set-
m	12	20	102	130	1010	1050	up
DAGP	18.4	11.5	47.3	14.6	44.1	19.3	10
RILC	11.6	8.8	15.7	10.4	15.9	12.5	30
Zipf distribution							
q	1.1		2		10		set-
v	1	10	1	10	1	10	up
DAGP	18.8	14.6	31.2	13.8	36.6	21.9	3.3
DEVR	21.4	-	16.8	-	13.8	-	1.0
ZRI	7.2	7.1	7.0	6.9	6.6	7.1	32.1

In the literature we found only two algorithms especially designed for the Zipf distribution. One, by Devroye [1986], (with a better bound for α proved by Baringhaus [1989]) is based on rejection from a discrete hat and works for the case $v = 1$ only. The second, by Dagpunar [1988], is based on Algorithm SR. Both have a larger α than ZRI, need two uniform random numbers per iteration and use no squeeze. The code length of the three generators are about the same but Algorithm ZRI is – depending on the parameter combinations – between two and four times faster than the algorithms suggested by Devroye [1986] and by Dagpunar [1988]. For the results of our timings see Table 2.

For the case $q = 1.1$ the execution times are influenced by the fact that algorithms DAGP and DEVR reject values larger than i_{max}. Additional timings showed that the use of this rejection would slow down Algorithm ZRI by about ten percent for the case $q = 1.1$.

8. CONCLUSION

The rejection-inversion method developed in this paper has several advantages over standard rejection methods to generate discrete random variates. Especially important is the fact that it requires only half of the uniform random numbers. Also concerning simplicity and speed the presented algorithms for the Zipf distribution and for a large class of monotone discrete distributions compare well with the algorithms presented in literature. Rejection-inversion can be used to sample from the tails of classical discrete distributions and to generate variates from discrete distributions with heavy tails which is important as for these distributions the very fast table methods cannot be used. Rejection-inversion can also be used to design universal algorithms for T_c-concave discrete distributions by applying the idea presented in Hörmann [1995] to discrete distributions. The details of this development will be given in a subsequent paper.

ACKNOWLEDGEMENT: The authors thank the referees and the area editor for their useful hints that substantially improved the presentation of the paper.

REFERENCES

BARINGHAUS, L. 1989. A note on a rejection method for generating random variates from the discrete pareto distribution. *Utilitas Mathematica 35*, 65–66.

DAGPUNAR, J. 1988. *Principles of Random Variate Generation.* Clarendon Press, Oxford.

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

HÖRMANN, W. 1994. A universal generator for discrete log-concave distributions. *Computing 52*, 89–96.

HÖRMANN, W. 1995. A rejection technique for sampling from T-concave distributions. *ACM Transactions on Mathematical Software 21*, 182–193.

JOHNSON, N. L., KOTZ, S., AND KEMP, A. W. 1992. *Univariate Discrete Distributions.* John Wiley, New York, 2 edition, 1992.