ALGORITHMS FOR GENERATING DISCRETE RANDOM VARIABLES WITH A GIVEN GENERATING FUNCTION OR A GIVEN MOMENT SEQUENCE

Luc Devroye
School of Computer Science
McGill University

Abstract. We present and analyze various algorithms for generating positive integer-valued random variables when the distribution is described either through the generating function $\sum_{i=0}^{\infty} p_i s^i$ or via the sequence of moments.

Table of contents.

Keywords and phrases. Generating function, random variate generation, algorithms, moment problem, rejection method, expected time analysis. **AMS Subject Classifications.** Primary 65C10. Secondary 65C05.

Authors' address: School of Computer Science, McGill University, 3480 University Street, Montreal, Canada H3A 2K6. The authors' research was sponsored by NSERC Grant A3456 and FCAR Grant 90-ER-0291.

1. Introduction.

Let X be a nonnegative integer-valued random variable taking the value i with probability p_i. The purpose of this paper is to suggest methods for generating X when the user is only given the probability generating function (or simply generating function)

$$k(s) = E(s^X) \stackrel{\text{def}}{=} \sum_{i=0}^{\infty} p_i s^i (s \in [0,1]).$$

The p_i's are not explicitly given; if they have to be used, they should to be computed from $k(s)$ (which is possible since the generating function uniquely determines the distribution of X).

For many discrete distributions, the explicit form of the p_i's is more complicated than that of the generating function. This has greatly contributed to the popularity and importance of the generating function in probability theory and mathematical statistics. In many situations, distributions are defined in terms of a generating function first. A case in point are the branching processes, where the generating function of the reproduction distribution determines the extinction probability and the Malthusian (or: rate of growth) coefficient in case of a surviving population.

In the table below, some generating functions are shown for well-known distributions.

DISTRIBUTION	GENERATING FUNCTION	p_i
Binomial (n,p)	$(1-p+ps)^n$	$\binom{n}{i} p^i (1-p)^{n-i} (0 \le i \le n)$
Geometric (p)	$\frac{p}{1-(1-p)s}$	$(1-p)^i p (i \ge 0)$
Negative binomial (n,p)	$\left(\frac{p}{1-(1-p)s}\right)^n$	$\binom{n+i-1}{i} (1-p)^i p^n (i \ge 0)$
Poisson (λ)	$e^{-\lambda+\lambda s}$	$\frac{\lambda^i e^{-\lambda}}{i!} (i \ge 0)$
Logarithmic series (p)	$\frac{\log(1-ps)}{\log(1-p)}$	$\frac{-p^i}{i \log(1-p)} (i \ge 1)$
Uniform on $\{0,\ldots,n\}$	$\frac{1-s^{n+1}}{(n+1)(1-s)}$	$\frac{1}{n+1} (0 \le i \le n)$

In the next section, we look at some well-known and not-so-well-known properties of the generating function. In the other sections, we derive and analyze some algorithms for our problem. We are not proposing these algorithms in lieu of more classical algorithms that require the exact knowledge of the p_i's, because they are usually more time-consuming. They are primarily geared towards cases in which the p_i's are not explicitly given. Nevertheless, since the p_i's are approximated in a simple fashion, considerable time savings can be gained for distributions for which evaluations of $k(s)$ take only a fraction of the time of evaluations of p_i.

2. Properties of the generating function.

The generating function $k(s)$ is nondecreasing on $[0,1]$, and takes the value 1 at $s=1$. For every $s \in [0,1]$, it is defined as a convergent Taylor series

$$k(s) = \sum_{i=0}^{\infty} p_i s^i = \sum_{i=0}^{\infty} \frac{k^{(i)}(0)}{i!} s^i,$$

which implies that it is analytic on $[0,1]$. This entails that derivatives can be taken under the summation. Thus, it is easy to see that every derivative $k^{(i)}$ is nondecreasing, and possesses a convergent Taylor series for $s \in [0,1)$.

The generating function of a convex combination (mixture) of distributions is the same convex combination of the individual generating functions. The generating function of a sum of independent random variables is the product of the individual generating functions.

The behavior of k near $s=1$ gives us information regarding the tail behavior of the p_i's. One aspect of this is captured in the well-known property that

$$k^{(r)}(1) = E(X(X-1)\cdots(X-r+1)),$$

the r-th factorial moment of X. To see this, observe that

$$k^{(r)}(s) = \sum_{i=r}^{\infty} i(i-1)\cdots(i-r+1) p_i s^{i-r}.$$

Hence all the moments of X, if they exist, can be derived from the sequence of values $k^{(r)}(1)$. The behavior of $k^{(r)}(0)$ on the other hand provides us with information about the p_i's:

$$k^{(r)}(0) = r! p_r.$$

(This can be obtained from the series for $k^{(r)}(s)$ shown above, or by equating Taylor series expansions for $k(s)$.) Our problem would be solved if we had access to $k^{(r)}(s)$ for all r, but unfortunately we do not. One might propose a solution in which p_r is estimated by the r-th derivative of $k(s)$, and this estimate is treated as the correct probability in some classical algorithm. This is unacceptable since it introduces an error with unknown consequences in the user's application. It is therefore essential that we estimate p_r in a controlled manner (with explicit error bounds), so that we can design an exact algorithm.

To obtain such estimates, we introduce for a function $f(x)$ on the real line the **difference of order r**, defined by

$$\Delta_t^r f(x) \stackrel{\text{def}}{=} \sum_{i=0}^{r} (-1)^{r-i} \binom{r}{i} f(x+it),$$

which can be computed in time proportional to r if all binomial coefficients and evaluations of f are available at unit cost. It can also be obtained recursively from

$$\Delta_t^r f(x) = \Delta_t^{r-1} f(x+t) - \Delta_t^{r-1} f(x).$$

This too requires $r+1$ computations of f, but about $O(r^2)$ subtractions. Differences of order r are commonly used in approximations of functions, see e.g. Lorentz (1986, p. 47). The tool we will need is given in Lemma 1.

LEMMA 1. *Inequalities for the probabilities. Assume that* $tr < 1$, $t > 0$, $r \geq 0$. *Then,*

$$\frac{\Delta_t^r k(0)}{r!t^r} \geq p_r \geq \frac{\Delta_t^r k(0)}{r!t^r} - \left((1-rt)^{-(r+1)} - 1\right) \sup_{i>r} p_i.$$

PROOF. We begin with a very convenient representation for differences of order r (see Lorentz, 1986, p. 47):

$$\Delta_t^r k(x) = \int_0^t \cdots \int_0^t k^{(r)}(x + y_1 + \cdots + y_r) dy_1 \cdots dy_r.$$

Since all $k^{(r)}$'s are *upa*, it is obvious that

$$\Delta_t^r k(x) \geq t^r k^{(r)}(x)$$

for all x, and hence,

$$\Delta_t^r k(0) \geq t^r k^{(r)}(0) = t^r r! p_r.$$

The second half of the proof follows again from the integral representation, applied with $x = 0$:

$$\begin{aligned}
\Delta_t^r k(0)/t^r &\leq k^{(r)}(rt) \\
&= \sum_{i=r}^{\infty} p_i \frac{i!}{(i-r)!} (rt)^{i-r} \\
&= r! p_r + r! \sum_{i=r+1}^{\infty} p_i \binom{i}{r} (rt)^{i-r}
\end{aligned}$$

$$
\begin{aligned}
&= r!p_r + r! \sum_{i=1}^{\infty} p_i + r \binom{i+r}{r} (rt)^i \\
&\leq r!p_r + \sup_i > rp_i r! \sum_{i=1}^{\infty} \binom{i+r}{r} (rt)^i \\
&= r!p_r + \sup_i > rp_i r! \left((1-rt)^{-(r+1)} - 1 \right) . \square
\end{aligned}
$$

As by-products of the representation used in the proof of Lemma 1, we observe that $\Delta_t^r k(x)$ is increasing in t when r, x are kept fixed, and in x when t, r are held fixed (see also Widder (1972, p. 150)).

3. The choice of an algorithm.

Lemma 1 provides us with a method for estimating every p_i to any desired accuracy based upon evaluations of $k(s)$ only, for we note that the error committed when p_r is estimated by $\Delta_t^r k(0)/(r!t^r)$ is not greater than

$$
(1-rt)^{-(r+1)} - 1 \to 0 \text{ as } t \downarrow 0.
$$

Interestingly, this is all we need to be able to design an exact algorithm. If p_r is estimated by $\Delta_t^r k(0)$ for some small t, there may be a serious loss of accuracy on small wordsize machines, as we need an r-times iterated difference. We won't be concerned with this issue here, preferring instead to give machine-invariant algorithms operating on real numbers.

There are basically three (partially overlapping) methods for generating discrete random variates (see e.g. Schmeiser (1983) or Devroye (1986)): table methods, inversion methods and rejection methods. Table methods are generally the most efficient methods (see e.g. Marsaglia (1963), Chen and Asau (1974), Walker (1974,1977), Ahrens and Kohrt (1981) and Peterson and Kronmal (1982)). Unfortunately, they require storage of the probabilities p_i. If they are to be used here, they should be modified in two directions, since the p_i's are known up to some small error, and since the number of possible values of X may not be finite. The way infinite tails are handled is necessarily based upon another method. The inversion and rejection methods are capable of handling infinite tails, and don't require long set-up times. They seem prime candidates for our problem. We begin with the rejection method, noting that the rejection method requires some extra knowledge in the form of a dominating probability vector (a probability vector $q_i (i \geq 0)$ is said to dominate the vector of p_i's when there exists a constant c such that $p_i \leq cq_i$ for all i). The inversion method does not require any additional information, but is (not surprisingly) the slowest of all methods.

4. Rejection algorithms.

If we knew all p_i's without error, we could use the standard rejection method:

```
repeat
       Generate X with probability vector q_0, q_1, ....
       Generate a uniform [0,1] random variate U.
until Accept (U,X)
  (where Accept (U,X) =def [U c q_X ≤ p_X]).
return X.
```

Here, as in the previous section, c is a constant with the property that $p_i \leq cq_i$ for all i. It is known that the expected number of iterations is c. The decision $[Ucq_X \leq p_X]$ does not require exact knowledge of p_X: it suffices to have a series of estimates of p_X approximating p_X. This technique has been developed in Devroye (1981) for example.

The function "Accept (U, X)" needed in the algorithm can be implemented as follows:

```
T ← U c q_X
t ← φ(X)/2 ( φ is defined in the next section;
e.g. the value φ(x) = ε/(x(x+1)^3 ψ(x)
for some ε > 0 and a function ψ(x) = x+1 will do )
while True do
  V ← Δ_t^X k(0)/(r! t^r)
  if T > V then return False
           else if T ≤ V − (1 − Xt)^{−(X+1)} + 1
                   then return True
  t ← t/2
```

A few words of explanation are in order here. A decision is made to accept or reject only when Ucq_X does not fall in the interval around p_X defined by the inequalities of Lemma 1, i.e.

$$Ucq_X not \in [\frac{\Delta_t^X k(0)}{r!t^r} - (1 - Xt) - (X + 1)) + 1, \frac{\Delta_t^X k(0)}{r!t^r}].$$

Here t is varied in a geometric fashion starting with $t = \varphi(X)/2$, so that it travels through the sequence $\varphi(X)/2^i$ $(i = 1, 2, \ldots)$ until the interval is small enough and a decision can be reached. At this point, there is no special reason why we should consider such a sequence.

The function "Accept" remains valid for any sequence $t \downarrow 0$. The proposed geometric sequence of t's leads to easily controllable expected times before halting.

A slight improvement may result in the algorithm if we bound the error not by $(1 - Xt) - (X + 1)) - 1$, but by $\min(1, \sup_{i \geq X} cq_i)$ times this quantity. This is especially useful when the dominating probability vector is monotone $\downarrow$, or when all q_i's are small. For a similar improvement when the p_i's are known to be monotone, we refer to section 13.

The next two sections are devoted to the two key issues, the expected time needed per random variate, and the construction of dominating probability vectors q_i.

5. Expected time per random variate.

The time required before halting can be measured in terms of the number of uniform $[0, 1]$ random variates needed, in terms of the number of evaluations of k, and so forth. It is helpful to isolate a counting method that is representative of the real time taken by the algorithm. One quickly realizes that the evaluations of $\Delta_t^X k(0)$ are at the heart of the matter. However, all evaluations of the difference function are not alike. If old values are not stored, then one evaluation takes time proportional to $X+1$ when binomial coefficients are available in constant time, and time proportional to $(X + 1)^2$ if the difference of order X is computed recursively. To unify the treatment, let $\psi(r)$ be the time required to evaluate $\Delta_t^r k(0)$. The expected time of the algorithm, assuming that all other operations take time zero (without loss of generality) will be called $E(T)$.

For fixed $X = r$, let N be the number of iterations in the function "Accept" before a decision is reached. We observe that

$$P(N > i) \leq \frac{\delta_i}{cq_r}$$

where $\delta_i = (1 - rt) - (r + 1)) - 1$, and t is the i-th value of t. This statement follows from the nature of the decision process and the fact that U is uniform $[0, 1]$. We recall that the i-th value of t is $\varphi(r)/2^i$. Thus, for fixed $X = r$, the expected number of evaluations of Δ_t^r is

$$\begin{aligned} \sum_{i=0}^{\infty} P(N > i) &\leq 1 + \sum_{i=1}^{\infty} \frac{\delta_i}{cq_r}. \\ &\leq 1 + \sum_{i=1}^{\infty} \frac{r(r+1)t}{cq_r(1 - r(r+1)t)} \\ &\text{(use } (1 - rt)^r + 1 \geq 1 - r(r+1)t) \end{aligned}$$

$$\leq 1 + \frac{r(r+1)\varphi(r)}{cq_r(1 - r(r+1)\varphi(r))}.$$

The expected time taken by the function "Accept" for fixed $X = r$ is bounded from above by $\psi(r)$ times the previous upper bound. The expected time, averaged over all X is bounded from above by

$$c \times \sum_{r=0}^{\infty} q_r \psi(r) \left(1 + \frac{r(r+1)\varphi(r)}{cq_r(1 - r(r+1)\varphi(r))}\right) = \sum_{r=0}^{\infty} \left(cq_r \psi(r) + \frac{r(r+1)\varphi(r)\psi(r)}{1 - r(r+1)\varphi(r)}\right)$$

where we used Wald's equation (which explains why we can bring the c outside; see Theorem II.3.5 of Devroye (1986)). The upper bound has two components, one that can't be avoided,

$$cE(\psi(X))(X \text{ has probability vector } \{q_i\})$$

(which can be considered as the absolute cost of the algorithm, regardless of how the sequence of t's is picked), and a t-dependent but distribution-invariant term

$$\sum_{r=0}^{\infty} \frac{r(r+1)\varphi(r)\psi(r)}{1 - r(r+1)\varphi(r)}$$

which can be made as small as desired by picking the t's small enough. Consider for example, for $X = r$,

$$t = \frac{\varphi(r)}{2^i} \stackrel{\text{def}}{=} \frac{\epsilon}{r(r+1)^3\psi(r)2^i}(i \geq 1)$$

for some $\epsilon \in (0,1]$. Then the second contributing term in the expected time is bounded by

$$\sum_{r=0}^{\infty} \frac{\frac{\epsilon}{(r+1)^2}}{1 - \frac{\epsilon}{\psi(r)(r+1)^2}} = \frac{\epsilon pi^2}{6} + O(\epsilon^2).$$

This can be made as small as desired by the proper choice of ϵ.

It should not come as a surprise that even if all Δ_t^r's were precisely equal to the corresponding p_r's (and we would know this), the expected time per variate is equal to $cE(\psi(X))$, the product of the rejection constant c, and a penalty factor indicative of the size of the tail of X. The penalty factor can in some cases be infinite, but this does not invalidate the algorithm, as it still halts with probability one.

6. Dominating probability vectors.

Dominating probability vectors can be found in a variety of manners. We will introduce just a few, based upon the inequalities of the next Lemma. Note that the p_i's should be bounded from above in terms of simple quantities involving the generating function only, as other information should not be assumed to be available.

LEMMA 2. *Upper bounds for probabilities. The following* **exponential** *bounds are valid:*

$$p_i \leq \frac{k(s)}{s^i} \ (s > 0).$$

Furthermore, for all integer $l > 0$, the following **polynomial** *inequality is valid:*

$$\begin{aligned} p_i &\leq \frac{k^{(1)}(1)}{i(i-1)\cdots(i-l+1)}(i \geq l) \\ &\leq \frac{\Delta_t^l k^{(1)}(1)}{i(i-1)\cdots(i-l+1)}(\text{all } t > 0). \end{aligned}$$

PROOF. The first and second inequalities follow from the fact that

$$k^{(1)}(s) = \sum_i = l^\infty i(i-1)\cdots(i-l+1)p_i s^i - l.$$

The first inequality is obtained by setting $l = 1$, and omitting all but the i-th term on the right hand side. The second inequality is obtained by setting $s = 1$. □

The inequalities of Lemma 2 are but a small sampling of simple bounds depending upon the generating function only. Observe for example that the last inequality is a special form of Chebyshev's inequality

$$\begin{aligned} P(X = i) &\leq P(X \geq i) \\ &\leq \frac{E(X(X-1)\cdots(X-l+1)}{i(i-1)\cdots(i-l+1)} \\ &= \frac{k^{(1)}(1)}{i(i-1)\cdots(i-l+1)}(i \geq l). \end{aligned}$$

This simple argument highlights the weakness of the bounding technique employed in Lemma 2. It is usually unacceptably gross to bound individual probabilities by tail probabilities including the individual probability. There is one well-known exception to this: for exponential or subexponential tails, the probabilities are comparable in size to the tails defined from the index in question upwards. In these small-tailed cases, the first inequality of Lemma 2 is useful. Just consider an arbitrary $s > 1$ to be picked further on. Then the bound $p_i \leq k(s)s^{-i}$ when used for all i yields a geometrically decreasing dominating probability vector. It is, of course, only useful when $k(s) < \infty$ for some $s > 1$. The usefulness of the bound to us can be measured in terms of the rejection constant

$$c = \sum_{i=0}^{\infty} k(s)s^{-i} = \frac{s}{s-1k(s)} \ .$$

This should be minimized with respect to s, if this is feasible. A random variate with the dominating probability vector $\left\{\frac{s-1}{s}s^{-i} \ (i \geq 0)\right\}$ can be generated as $\lfloor E/\log(s) \rfloor$, where E is an exponential random variate. The distribution-dependent contribution to the expected time, $cE(\psi(X))$, is bounded by

$$cE(\psi(\frac{E}{\log(s)})) = \begin{cases} \frac{s}{s-1}k(s)(1+\frac{1}{\log(s)}) & (\psi(r) \equiv r+1) \\ \frac{s}{s-1}k(s)(1+\frac{2}{\log(s)}+\frac{2}{\log^2(s)}) & (\psi(r) \equiv (r+1)^2) \end{cases} \ .$$

For distributions with heavier-than-exponential tails, the first bound of Lemma 2 is obviously useless. One should resort to a polynomial Chebyshev-like inequality, or use additional information.

7. Tables of probabilities.

Let us briefly illustrate how tables of inexact probabilities can be kept to reduce the time of the rejection method. One should not confuse this with a table method, in which the table itself is needed for generating X, obviating the need for a dominating probability vector.

The idea is to fix a table size, and to store in the table records with three components, (u, δ, t), where $u = \Delta_t^r k(0)/(r!t^r)$ is an upper bound for p_r, t is the t value used in computing this value, and δ is the bound on the error, $(1-rt)^{-(r+1)} - 1$ (thus, $u - \delta$ bounds p_r from below). In a dynamic table, there is no information at the beginning. As random variates are being generated, the table is gradually filled in. After some time, most δ's in the table will be so small that the expected time per random variate is virtually indistinguishable from that of the rejection method with a priori given p_r's.

For the sake of completeness, we indicate how the function "Accept" should be altered.

```
[An initially empty table of values (u_r, δ_r, t_r), r ≥ 0 is assumed.]
T ← Ucq_X
[Quick table-based decision.]
if table entry at X present
  then if T ≤ u_X − δ_X
          then return True
          else if T > u_X
                  then return False
                  else t ← t_X/2 (new t needed for better estimate of p_X)
  else t ← φ(X)/2 (for definition of φ, see above)
[Standard acceptance based upon converging approximations.]
while True do
  V ← Δ_t^X k(0)/(X!t^X)
  δ ← (1 − Xt)^−(X+1) − 1 (for monotone ↓ p_i's, use V times this quantity)
  Update table:  (u_X, δ_X, t_X) ← (V, δ, t)
  if T > V then return False
          else if T ≤ V − δ then return True
  t ← t/2
```

Dynamic tables like the one suggested here have the pleasing feature that they require work bounded above by the number of random variates generated. Thus, the dilemma of how much time should be spent (or "wasted") on setting up tables is automatically resolved. The tables suggested here improve with time. To see this, assume that $cE(\psi(X)) < \infty$. With $\varphi(r) = \epsilon/(r(r+1)^3\psi(r))$ (as suggested in the previous section), it can be shown that the expected number of evaluations of k per random variate tends to zero as more and more random variates are being generated.

8. The inversion method.

We extend the ideas of the previous sections to design an inversion algorithm that is applicable to **all** generating functions, without exception. Let us recall the raw inversion method for a probability vector of p_i's:

```
X ← 0.
S ← p_0.   (S holds ∑_{i≤X} p_i.)
Generate a uniform [0,1] random variate U.
while U > S do
   X ← X + 1
   S ← S + p_X
return X.
```

The expected number of evaluations of probabilities p_i before halting is $E(1+X)$. This is an unavoidable cost. We note in passing that the inversion method is best suited for small-tailed distributions and distributions with monotonely decreasing probabilities. To show that the inversion method is not different in terms of decision making than what we have seen for the rejection method, let us rewrite the algorithm in terms of a function "Accept (U,X)", similar to the one used for the rejection method. It accepts X when $U \leq \sum_{i \leq X} p_i$ and rejects X otherwise.

```
X ← 0.
Generate a uniform [0,1] random variate U.
while True do
   if Accept (U,X)
      then return X
      else X ← X + 1
```

The tool we will need is given in Lemma 3.

Lemma 3. *Inequalities for cumulative probabilities. Assume that* $r \geq 0$, *and* $t_r = \epsilon_r/((r+1)2^r+1)$ *for some sequence of* ϵ_r*'s with* $0 < \epsilon_r \leq \epsilon < 1$, *all* r. *Then,*

$$\sum_{i=0}^{r} \frac{\Delta_{t_i}^i k(0)}{i! {t_i}^i} \geq \sum_{i=0}^{r} p_i \geq \sum_{i=0}^{r} \frac{\Delta_{t_i}^i k(0)}{i! {t_i}^i} - \frac{2\epsilon}{1-\frac{\epsilon}{4}}.$$

PROOF. The upper bound for the cumulative probability is immediate from Lemma 1. For the lower bound, we again apply Lemma 1, and the fact that

$$\begin{aligned}\sum_{i=0}^{r}((1-it_i)^{-(i+1)}-1) &\leq \sum_{i=0}^{r}\frac{i(i+1)t_i}{1-i(i+1)t_i}\\ &\leq \sum_{i=0}^{r}\frac{i\epsilon/2^{i+1}}{1-i\epsilon/2^{i+1}}\\ &\leq \sum_{i=0}^{\infty}\frac{i\epsilon/2^{i+1}}{1-\frac{\epsilon}{4}}\\ &= \frac{2\epsilon}{1-\frac{\epsilon}{4}}.\square\end{aligned}$$

For the implementation of the function "Accept", we immediately proceed with the dynamic table version. The table can be stored in an array, but is preferably held in a linked list. Since the inversion method increments X by one in each iteration, the linked list structure seems appropriate. Each cell in the linked list holds at least a 7-tuple, (1) the index i of the current p_i; (2) the t-value used in the computation of $\Delta_t^i k(0)$; (3) an upper bound for p_i, $\Delta_t^i k(0)/(i!t^i)$; (4) a lower bound for p_i, $\Delta_t^i k(0)/(i!t^i)\ -(1-it)^{-(i+1)}+1$; (5) an upper bound for $\sum_{j\leq i} p_j$; (6) a lower bound for $\sum_{j\leq i} p_j$; (7) a pointer to the next cell, if not nil.

Items in the linked list are only improved as more variates are generated, i.e. t-values are replaced by smaller ones. One possible implementation is as follows:

```
[X also refers to the cell holding X.]
[p_l(X), p_u(X), P_l(X) and P_U(X) are the individual and cumulative lower and upper bounds referred to above.]
while True do
  if X is not in the list (X has never been considered before)
    then compute all values in its cell, based upon t = ε/((X+1)^3 2^X).
  if U ≤ P_l(X)
    then return True
    else if U > P_u(X) then return False
  Update all cells 0,...,X by halving the t's for each cell.
```

The size of the linked list after n random variates have been generated is precisely equal to $E(\max(X_1,\ldots,X_n))$, where the X_i's are iid with the given generating function k. The function "Accept" stops and returns a value in (random) finite time, since the interval around the cumulative sum shrinks with time. It is possible to generate all the information "on the fly" for each random variate generated. Obviously, it is advantageous to store information as we did in the linked list. Even without a linked list, we will now see that the expected time taken by the algorithm can be controlled very nicely.

9. Expected time per random variate for the inversion method.

Let N be the number of evaluations of k needed in the course of the generation of one random variate by the inversion method. In the algorithms with memory described in the previous sections, $E(N)$ decreases with time as more and more random variates are generated. So we will take the pessimistic viewpoint that N refers to the first random variate only.

LEMMA 4. *In the inversion algorithm, $E(N)$ is bounded from above by $T_1 + T_2 + T_3$, where*

$$T_1 = (2 + \log_2 e)(\mu_3 + 6\mu_2 + 11\mu_1 + 3)/6,$$

$$(\mu_i \stackrel{\text{def}}{=} \sum_{j=0}^{\infty} j^i p_j)$$

$$T_2 = \sum_{i:p_i>\delta} \frac{(i^3 + 6i^2 + 11i + 3)}{6} \delta \log_2 e,$$

$$T_3 = \sum_{i:p_i\leq\delta} \frac{(i^3 + 6i^2 + 11i + 3)}{6} p_i \log_2(\frac{\delta}{p_i}).$$

Here $\delta = 2\epsilon/(1 - \epsilon/4)$, and ϵ is as in Lemma 3.

PROOF. For fixed U in the X-th interval (U is the uniform random variable to be inverted, and X is the generated random variable), the number of evaluations of k is $N_0 + \cdots + N_X$ where N_i refers to the number due to the decision not to accept i. Now, N_i is bounded from above by $1 + 2 + \cdots + (i+1) = (i+1)(i+2)/2 = (i^2 + 3i + 2)/2$ times the number of iterations before a decision is reached. Note that in each iteration, the t-values are halved. If this number of iterations is $n = n(U)$, we note that $n \geq 1$, and, in view of Lemma 3, putting $u = U - \sum_j < ip_j$ for $i < X$ and $u = \sum_{j\leq X} p_j - U$ for $i = X$,

$$\frac{\delta}{2^n - 1} \leq U,$$

where $\delta = 2\epsilon/(1 - \epsilon/4)$. Hence,

$$n \geq \log_2(\frac{\delta}{u}) + 1,$$

and thus, a safe upper bound for n is

$$2+\max(\log_2(\frac{\delta}{u}),0).$$

It is clear that u is uniformly distributed on $[0,p_X]$, so that we obtain the following upper bound for the expected number of evaluations of k:

$$\begin{aligned} E(N) &\leq \sum_{i=0}^{\infty}\int_0^{p_i}\sum_{j=0}^{i}\psi_j n(u)du \\ &= \sum_{i=0}^{\infty}\Psi_i N(p_i) \end{aligned}$$

where $\psi_j = (j^2+3j+2)/2$ (see above), $\Psi_i = \sum_{j=0}^{i}\psi_j = (i^3+6i^2+11i+3)/6$, and $N(z)=\int_0^z n(u)du$. It is easy to verify that

$$N(z)=\begin{cases} 2z+z\log_2(\frac{e\delta}{z}) & (0\leq z\leq \delta) \\ 2z+\delta\log_2 e & (\delta<z) \end{cases}.$$

This gives the bound

$$E(N)\leq (2+\log_2 e)\sum_{i=0}^{\infty}\Psi_i p_i+\sum_{i:p_i>\delta}\Psi_i\delta\log_2 e+\frac{\sum_{i:p_i\leq\delta}\Psi_i p_i\log_2(\delta}{p_i})$$

which was to be shown. □

Lemma 4 reveals that the expected time is finite if the entropy-related term T_3 is finite; note that this implies at the very least that the third moment of the distribution is finite. Roughly speaking, all three terms in the upper bound are indices of the size of the tail of the distribution. The second and third terms can be made as small as desired, if they are finite, by choosing ϵ small enough. Thus, T_1 is an unavoidable cost, inherent in the algorithm; no choice of the parameters affects T_1. The mere fact that T_1 is proportional to the third moment of the distribution renders the inversion method very inefficient; one should recall however that no requirements were imposed on the generating function; the inversion method is a true "black box" method that can be used as a last resort.

10. Finite distributions: extension of the guide table method.

In the case of finite probability vectors with a known bound n for the support (i.e., $p_i = 0$ for $i > n$), it is perhaps best to create a guide table of size n (Chen and Asau, 1974) for generating a random variate with probability vector proportional to $\{\Delta_t^i k(0)/(i!t^i), 0\leq i\leq n\}$ for a given t where t can possibly depend upon i. Since this vector dominates the p_i-vector (see Lemma 1), we can apply the rejection method.

```
[ADJUSTED GUIDE TABLE METHOD.]
[One time initialization.]
FOR i := 0 TO n do u_i ← Δ_t^i k(0)/(i!t^i).
  (t is allowed to depend upon i, see below.)
S ← Σ_{i=0}^n u_i.
Set up the necessary structures for applying the guide table method to the probability vector {u_i/S}.
[Generation.]
repeat Generate X with probability vector {u_i/S} by the guide table method.
       Generate a uniform [0,1] random variate U.
until Accept (U,X) (i.e. U u_X ≤ p_X )
return X.
```

A few words about the efficiency are in order here. The set-up step requires quadratic or cubic time in n, depending upon the manner in which the Δ_t^i's are computed. This is, of course, a one-time cost. For a tight fit, it is important to take t as small as possible. The relationship between t and n is explained below. If a set-up is to be avoided altogether, one should probably start from the only guaranteed dominating probability vector, $u_i \equiv 1, 0 \le i \le n$; or from $u_i = 1/(i+1), 0 \le i \le n$, in the case of nonincreasing probabilities.

The rejection constant is the expected number of uses of the guide table method before halting. It is well-known that the guide table method takes expected time uniformly bounded by a constant over all distributions (see e.g. Devroye (1986)). Hence, the rejection constant is a good index of the expected time contribution of the adjusted guide table method. A simple upper bound for the rejection constant is given below.

Lemma 5. *The rejection constant in the adjusted guide table algorithm with $t = t_i$ is*

$$\frac{\sum_{i=0}^n \Delta_{t_i}^i k(0)}{i! t_i} \le 1 + \sup_i p_i \sum_{i=0}^n \frac{i(i+1)t_i}{1 - i(i+1)t_i}.$$

If we take $t_i \stackrel{\text{def}}{=} \frac{\epsilon}{i(i+1)(n+1)}$ where $\epsilon \in (0,1)$, then the rejection constant is not greater than

$$1 + \sup_i p_i \frac{\epsilon}{1 - \frac{\epsilon}{n+1}} \le 1 + \frac{\epsilon}{1 - \frac{\epsilon}{n+1}}.$$

This is $O(1)$ for fixed $\epsilon > 0$, and converges to 1 as $\epsilon \downarrow 0$.

PROOF. From Lemma 1, the rejection constant is not greater than

$$\sum_{i=0}^{n} \Delta_{t_i}^i k(0) \leq 1 + \sup_i p_i \sum_{i=0}^{n} ((1 - it_i)^{-(i+1)} - 1)$$
$$\leq 1 + \sup_i p_i \sum_{i=0}^{n} \frac{i(i+1)t_i}{1 - i(i+1)t_i}.$$

If we take $t_i \stackrel{\text{def}}{=} \frac{\epsilon}{i(i+1)(n+1)}$ where $\epsilon \in (0,1)$, then the last expression does not exceed

$$1 + \sup_i p_i \frac{\epsilon}{1 - \frac{\epsilon}{n+1}} \leq 1 + \frac{\epsilon}{1 - \frac{\epsilon}{n+1}}. \square$$

Lemma 5 provides us with a simple tool for making the rejection algorithm as efficient as desired; one could for example take $t_i = 0.05/(i(i+1)(n+1))$.

We will see that it is even more to our advantage to take $t_i = \delta/(i(i+1)n\psi(i))$ where $\delta \in (0,1]$ is a constant and $\psi(i)$ is a work function usually equal to $i+1$. To see this, it is necessary to analyze the second component in the total expected time, the expected number of evaluations of k. For this analysis, we give the function "Accept (U, X)" explicitly:

$V \leftarrow u_X$ (upper bound for, and approximation of p_X)
$T \leftarrow U u_X$
$t \leftarrow t_X$ (for definition, see Lemma 5)
while True do
  if $T \leq V - (1 - Xt)^{-(X+1)} + 1$ ($T \leq V - \frac{X(X+1)t}{1-X(X+1)t}$ will do too)
    then return True
    else if $T > V$ then return False
  $t \leftarrow t/2$ (get a better approximation)
  $V \leftarrow \Delta_t^X k(0)/(X! t^X)$

For fixed $X = i$, the expected number of evaluations of Δ_t^i is not greater than

$$\sum_{j=1}^{\infty} \frac{i(i+1)t_i/2^j}{u_i(1 - i(i+1)t_i/2^j)}$$

where we followed an argument as in section 5. Note that in the first iteration, we exit with large probability without evaluating $\Delta_t^i k(0)$. With the choice of $t_i = \delta/(i(i+1)n\psi(i))$ ($\delta \in (0,1]$) suggested in the discussion following Lemma 5, the upper bound becomes

$$\sum_{j=1}^{\infty} \frac{\delta/(n\psi(i)2^j)}{u_i(1 - \delta/(n\psi(i)2^j))}.$$

Each evaluation of the difference of order i takes time $\psi(i)$ (which is either $i+1$ or $(i+1)^2$ depending upon one's model); hence, the overall expected time contribution from the evaluations of the generating function is bounded from above by

$$\begin{aligned}\sum_{i=0}^{n} u_i\psi(i)\left(\sum_{j=1}^{\infty}\frac{\delta/(n\psi(i)2^j)}{u_i(1-\delta/(n\psi(i)2^j))}\right) &= \sum_{i=0}^{n}\psi(i)\left(\sum_{j=1}^{\infty}\frac{\delta/(n\psi(i)2^j)}{(1-\delta/(n\psi(i)2^j))}\right)\\ &\le \sum_{i=0}^{n}\left(\frac{\delta}{n-1/2}\right)\\ &= \delta\frac{n+1}{n-1/2}.\end{aligned}$$

Here too we can control the complexity by δ: the bound can be made as small as desired by picking δ small enough. In fact, for δ very small, the performance of the algorithm should be virtually indistinguishable from that of the guide table method based upon a table of correct probabilities.

It is possible to replace the guide table by Walker tables (Walker (1974, 1977); Peterson and Kronmal (1982)). It is perhaps interesting to note that the set-up of the tables often requires much less time than for the case of known probabilities because most discrete probabilities are of a combinatorial nature and hence involve computing factorials, iterated products, Stirling numbers and the like. Generating functions on the other hand are in many cases surprisingly "simple" functions of s.

11. The moment problem.

Assume that we are not given the generating function, but that the distribution is specified indirectly in the form of values of the factorial moments

$$M_i \stackrel{\text{def}}{=} \sum_{j=0}^{\infty} p_j j(j-1)\cdots(j-i+1) = E(X(X-1)\cdots(X-i+1))(i\ge 0).$$

Under some tail conditions, these moments determine the distribution; they generally don't when one of the moments is infinite. And even if all moments are finite, the distribution is not necessarily uniquely defined: Stoyanov (1987) has constructed two families of distributions on the integers each sharing the same infinite moment sequence. We will show here how one can generate a random integer with a given moment (or factorial moment) sequence under the condition that

$$k(1+u) = E((1+u)^X) < inf$$

for some $u > 1$. (This is also a sufficient condition for the moment sequence to determine the distribution.) The algorithm is not necessarily efficient, but at least it is conceptually

simple.

Since there are trivial linear relationships between the ordinary moments μ_i and the factorial moments, giving the first i factorial moments is equivalent to giving the first i ordinary moments. We have

$$\mu_i = \sum_{j=0}^{i} S_i j M_j$$

where $S_i j$ is Stirling number of the second kind, defined for example through the formula

$$S_i j = \sum_{l} = 0toj \binom{j}{l} (-1)^l (j-l)^i.$$

See Johnson and Kotz (1969). There are similar formulas relating M_i to the μ_j's for $0 \leq j \leq i$ involving Stirling numbers of the first kind $(F_i j)$:

$$M_j = \sum_{i=0}^{j} F_i j \mu_i.$$

For easy reference, it suffices to note that $F_i j = 0$ when $j > 0$ and either $i > j$ or $i \leq 0$. Furthermore, $F_0 0 = 0$ and $F_{ij+1} = F_{i-1j} - jF_{ij}$ for $j \geq i \geq 1$. Hence we will assume without loss of generality that the factorial moments M_i are available.

Under the condition that $k(1+u) < \infty$, we see that k is a convergent power series for $|s| < 1+u$. Thus, we can expand it in a Taylor series around $s = 1$, so that for any point $s \in (-(1+u), 1+u)$,

$$k(s) = \sum_{i=0}^{\infty} \frac{(s-1)^i}{i! M_i}.$$

By the binomial theorem, this is

$$\begin{aligned} k(s) &= \sum_{i=0}^{\infty} \sum_{j=0}^{i} \binom{i}{j} s^j (-1)^i - j \frac{M_i}{i!} \\ &= \sum_{j=0}^{\infty} \sum_{i=j}^{\infty} s^j (-1)^i - j \frac{M_i}{j!(i-j)!} \\ &= \sum_{j=0}^{\infty} \frac{s^j}{j!} \sum_{i=0}^{\infty} (-1)^i \frac{M_{j+i}}{i!} \ . \end{aligned}$$

Under certain conditions, this series can be identified on a term-by-term basis with the original defining power series for $k(s)$. This would yield

$$p_j = \frac{1}{j!} \sum_{i=0}^{\infty} (-1)^i \frac{M_{j+i}}{i!}.$$

For sufficient conditions under which this argument is valid, see Lemma 6 below. The formula for p_j is mentioned for compact support distributions in standard textbooks, such as Kendall and Stuart (1977, p. 96). For it to be useful to us, we require some information about the error committed when the ∞ in the series is replaced by a finite integer n.

LEMMA 6. *Assume that $k(1+u) < \infty$ for some $u > 1$. Let p_nj be an n-term approximation of p_j defined by*

$$p_nj = \frac{1}{j!}\sum_{i=0}^{n}(-1)^i \frac{M_{j+i}}{i!}.$$

Then, for all even n, $p_nj \geq p_j$, while for all odd n, $p_nj \leq p_j$. Furthermore, for all j,

$$\lim_{n\to\infty} p_nj = p_j.$$

PROOF. $k(s)$ is absolutely summable on $A = (-(1+u), 1+u)$, and hence is analytic on A (Dieudonne, 1969, p. 203). Therefore, $k(s)$ can be expanded in a unique absolutely summable power series around every point in the interior of A. The same is true for all derivatives of k. In particular, we have

$$k(s) = \sum_{j=0}^{\infty}(s-1)^j \frac{M_j}{j!}$$

for $s < 1+u$, and

$$\begin{aligned} k^{(j)}(s) &= \sum_{i=0}^{\infty} \frac{(s-1)^i}{i!} k^{(j+i)}(1) \\ &= \sum_{i=0}^{\infty} \frac{(s-1)^i}{i!} M_{j+i}. \end{aligned}$$

We know that $j!p_j = k^{(j)}(0)$. To apply the last series expansion at $s = 0$, we need to require that the radius of convergence be greater than one, and hence that $k(1+u) < \infty$ for $u > 1$. It is also known that if we truncate the last sum to include only the terms $0 \leq i \leq n$, then the remainder is

$$\frac{(s-1)^{n+1}}{(n+1)!} k^{(j+n+1)}(\theta)$$

for some $s \leq \theta \leq 1$. The remainder for $s = 0$ is therefore negative if and only if n is even. Thus, the partial series alternately underestimates and overestimates p_j.

The last statement of Lemma 6 follows from the observation that

$$\sum_{i=0}^{\infty} \frac{u^i}{i!} M_i < \infty,$$

and thus, $M_n \leq Cn!u^{-n}$ for some constant C. From this, for fixed j,

$$\frac{M_j + n}{n!j!} \leq C \binom{n+j}{n} u^{-(n+j)} \leq C(n+j)^j u^{-(n+j)}.$$

This tends to zero when $u > 1$. This implies the convergence of $p_n j$ to p_j as $n \to \infty$. $\square$

The sandwiching inequalities in Lemma 6 are also referred to as Bonferroni's inequalities. They are in fact universally valid. The condition on k in fact implies that for large n the inequalities are also tight. The series representation for p_j given in Lemma 6 can be used for designing general algorithms for generating random variables with a given moment sequence. These algorithms would not involve partial solutions of the well-known moment problem (discussed for example in Widder (1972)). Since this has been an open problem for a long time, it is perhaps worth our while to provide clues as to how one might proceed. Note that the treatment here parallels that of the generating function. It should be noted however that our algorithms can't possibly work when the moments do not uniquely define the distribution. This can be seen as a drawback since it would seem that the additional freedom would make the generation problem simpler. The reader can verify that our algorithms don't halt when for some j, the sequence $M_{j+i}/i!$ does not tend to zero as $i \to \infty$. The algorithms are thus intimately linked to the solution of the moment problem (see e.g. Feller (1971)). In the next two sections, we will take the liberty of referring to our generation problem as "the moment problem".

12. Rejection algorithms in the moment problem.

Rejection can be dealt with as in section 4. We need of course a dominating probability vector. Since we assume that $k(1+u) < \infty$ for some $u > 1$ (Lemma 6), we can use the bound

$$p_j \leq k(1+u)(1+u)^{-j},$$

if the k-value in question, or an upper bound for it, is available. In any case, we always have

$$p_j \leq \min(\mu_r/j^r, 1)$$

for all fixed r, where μ_r is the r-th moment. We should not forget that μ_r can be computed from the first r factorial moments $M_i, 1 \leq i \leq r$. Random variates with the polynomially decreasing probability vector can be generated by the rejection method. To see this, construct a function $g(x)$ via

$$g(x) \stackrel{\text{def}}{=} \begin{cases} 0 & x < -1 \\ 1 & -1 \le x < c \stackrel{\text{def}}{=} \lfloor \mu_r^{1/r} \rfloor \\ \mu_r / x^r & x \ge c \end{cases} .$$

This function has the property that $g(x) \ge p_j$ for all $j-1 < x \le j$ and all $j \ge 0$. Thus, for the rejection algorithm, we can proceed as follows, after noting that the constant portion of g has integral $c+1$, and that the polynomial tail has integral $\mu_r/((r-1)c^{r-1})$:

```
[Rejection algorithm for {min(μ_r/j^r, 1)}.]
while True do
  Generate iid uniform [0,1] random variates U, V.
  if U ≤ (c+1)/(c+1+μ_r/((r−1)c^{r−1}))
    then return X ← ⌈−1 + (c+1)V⌉
      (use fact that p_j = 1 for j ≤ c)
    else Y ← c/V^1/(r − 1) (Y has tail-of-g density)
         X ← ⌈Y⌉
         Generate a uniform [0,1] random variate W.
         if W ≤ (Y/X)^r then return X
```

The expected number of iterations in the algorithm is the area under g, i.e.,

$$\begin{aligned} c + 1 + \frac{\mu_r}{(r-1)c^{r-1}} &= \lfloor \mu_r^{1/r} \rfloor + 1 + \frac{\mu_r}{(r-1)(\lfloor \mu_r^{1/r} \rfloor)^{r-1}} \\ &\le \lceil \mu_r^{1/r} \rceil + 1 + \frac{\mu_r}{(r-1)(\lceil \mu_r^{1/r} \rceil)^{r-1}} \\ &\le 2 + \frac{r}{r-1} \mu_r^{1/r}. \end{aligned}$$

The rejection algorithm for the moment problem is as in section 4, provided that we take $cq_j \stackrel{\text{def}}{=} \min(\mu_r/j^r, 1)$, and that the function "Accept (U, X)" is implemented as follows:

$V \leftarrow U \min(\mu_r/X^r, 1)X!$
  `(get ready for testing whether` $V = Ucq_X X! \le p_X X!$`)`
$n \leftarrow 0$.
$S \leftarrow M_X$
`while True do`
  `if` n `even`
    `then if` $V > S$ `THEN return False (reject)`
    `else if` $V \le S$ `then return True (accept)`
  $n \leftarrow n + 1$
  $S \leftarrow S + (-1)^n M_X + n/n!$ `(next approximation)`

The validity of the algorithm follows from the fact that the approximations of each p_j converge, and that consecutive approximations are underestimates and overestimates respectively.

13. Expected time per random variate in the moment problem.

In this section, we determine the efficiency of the algorithm under the assumption that the j-th factorial moment is available at unit cost, or in the worst case at cost $j+1$ (when it has to be computed from the first j standard moments). For this purpose, let N be the number of evaluations of a factorial moment. We have:

LEMMA 7. *Assume that $k(u) < \infty$ for some $u > 3$ where k is the factorial moment generating function of some random variable. The expected number $E(N)$ of factorial moments required by the rejection algorithm for the moment problem is bounded by*

$$c + k(3)$$

where c is the rejection constant (see previous section).

PROOF. Fix $X = r$, and consider the number of moments N_r needed before a decision is reached whether to reject or accept. It is known that by Wald's theorem, $E(N) = \sum_{r=0}^{\infty} cq_r E(N_r)$ where c is the rejection constant, and q_r is the dominating probability vector.

We have

$$P(N_r - 1 \ge j) \le \frac{M_r + j - 1}{r!(j-1)!cq_r}(j \ge 1)$$

and

$$P(N_r - 1 \geq 0) = 1.$$

Here we use Lemma 6 on alternate over– and underestimates of p_r. Thus,

$$\begin{aligned}
E(N_r - 1) &= \sum_{j=1}^{\infty} P(N_r - 1 \geq j) \\
&\leq \sum_{j=1}^{\infty} \frac{M_r + j - 1}{r!(j-1)!cq_r} \\
&= \sum_{j=0}^{\infty} \frac{M_r + j}{r!j!cq_r} \\
&= \sum_{j=0}^{\infty} \frac{k^{(r+j)}(1)}{r!j!cq_r} \\
&= \frac{k^{(r)}(2)}{r!cq_r}.
\end{aligned}$$

Thus,

$$\begin{aligned}
E(N) &= \sum_{r=0}^{\infty} cq_r \left(1 + E(N_r - 1)\right) \\
&\leq c + \sum_{r=0}^{\infty} \frac{k^{(r)}(2)}{r!} \\
&= c + k(3)
\end{aligned}$$

provided that $k(u) < \infty$ for some $u > 3$. □

In practice, the bound of Lemma 7 is all but useless except for very short-tailed distributions. Indeed, by Jensen's inequality, $k(3) \geq 3^{E(X)}$. Nevertheless, the bound is intuitively appealing since it shows a rejection component (c), and a distributional rejection-less component $k(3)$. The latter component describes a cost we can't do without, since it is present even if we have 100% acceptance probabilities ($c = 1$). In other words, it can be considered as a cost inherent in the series approximation of probabilities by means of moments or factorial moments. On the positive side, it should be mentioned that $k(u) < \infty$ (all $u > 0$) for many important distributions, such as all finite distributions and all distributions with a subexponential tail (i.e., for any $C > 0$, $p_i \leq e^{-Ci}$ for all i large enough).

Even though the algorithm is valid when $k(u) < \infty$ for some $u > 2$, Lemma 7 does not provide a finite expected time guarantee when $k(u) < \infty$ for some $u \in (2, 3]$ but

$k(u) = \infty$ for all $u > 3$.

When the r-th factorial moment is not available at unit cost, but at cost $\psi(r)$, "$N_r + 1$" should be replaced in the argument by $\sum_{i=0}^{r} \psi(i)$. Although we won't pursue this any further, it suffices to mention that for $\psi(r) = r^l$ (l integer), the condition for finite expected time becomes $k^{(1)}(3) < \infty$.

Just as in the first half of this paper, the rejection algorithm can be replaced by a usually less efficient but universally applicable inversion algorithm. In the moment problem, such an algorithm can be based upon the series representation

$$P(X \geq i) = \frac{1}{(i-1)!} \sum_{j=0}^{\infty} (-1)^j \frac{M_i + j}{j!(j+i)} (i \geq 1)$$

(see e.g. Laurent (1965, p. 437)).

The ideas presented here for the moment problem can be applied (with modifications) to continuous random variables as well. It suffices to observe that the distribution function can be represented as a converging series depending upon the moments only, provided that the moments uniquely define the distribution. See for example Theorem 2 on p.227 of Feller (1971). We will present results for the general moment problem elsewhere.

14. Monotone distributions.

If in addition to the generating function $k(s)$, we know that the p_i's are monotonically $\downarrow$, it is possible to improve upon the rejection and inversion algorithms given in sections 4 through 10, basically because we have

LEMMA 8. *Inequalities for monotone probabilities. Assume that $tr < 1$, $t > 0$, $r \geq 0$. Then,*

$$\frac{\Delta_t^r k(0)}{r! t^r} \geq p_r \geq \frac{\Delta_t^r k(0)}{r! t^r} \left(1 - \left((1 - rt)^{-(r+1)} - 1\right)\right).$$

Lemma 8 is an immediate corollary of Lemma 1. The function "Accept (U, X)" used in the rejection algorithm can now be improved as follows:

```
T ← Ucq_X
t ← φ(X)/2
while True do
        V ← Δ_t^X k(0)/(r!t^r)
        if T > V
          then return False
          else if T ≤ V (2 − (1 − Xt)^{−(X+1)}) (improved step)
                then return True
        t ← t/2
```

This modification has some implications on the expected time. We note that in the notation of section 5, $\Delta_t^r k(0)/(r!t^r) \leq p_r/(1-\delta_i)$, so that

$$P(N>i) \leq \frac{\Delta_t^r k(0)\delta_i}{cq_r} \leq \frac{\delta_i p_r}{cq_r(1-\delta_i)}$$

where t is the i-th value of t. A little work, too tedious to reproduce here, shows that the expected time per random variate is bounded from above by

$$cE(\psi(X)) + \sum_{r=0}^{\infty} \frac{r(r+1)\varphi(r)\psi(r)p_r}{1-2r(r+1)\varphi(r)}$$

where X has probability vector $\{q_i\}$. This is nearly always smaller than the corresponding bound for the original rejection algorithm (see section 5).

In terms of a dominating probability vector, we can also slightly improve on what we have seen before, since now

$$p_i \leq \frac{(s-1)k(s)}{s^{i+1}-1}(s>1; i \geq 0).$$

The latter inequality is best obtained by the following argument:

$$p_i \frac{s^{i+1}-1}{s-1} = p_i \sum_{j=0}^{i} s^j \leq \sum_{j=0}^{i} p_j s^j \leq k(s).$$

The area under the bounding curve becomes

$$k(s) \sum_{i=0}^{\infty} \frac{1}{1+s+\cdots+s^i},$$

which is better than the corresponding area obtained without monotonicity (see section 6). Unfortunately, there does not seem to be a simple way of generating a random variable with probability vector proportional to $1/(s^{i+1}-1)$ except via rejection from a geometric random variable with probability vector proportional to s^{-i}.

15. References.

J. H. Ahrens and K. D. Kohrt, "Computer methods for efficient sampling from largely arbitrary statistical distributions," *Computing*, vol. 26, pp. 19–31, 1981.

H. C. Chen and Y. Asau, "On generating random variates from an empirical distribution," *AIIE Transactions*, vol. 6, pp. 163–166, 1974.

L. Devroye, "The series method in random variate generation and its application to the Kolmogorov-Smirnov distribution," *American Journal of Mathematical and Management Sciences*, vol. 1, pp. 359–379, 1981.

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

J. Dieudonne, *Foundations of Modern Analysis*, Academic Press, New York, 1969.

W. Feller, *An Introduction to Probability Theory and its Applications, Vol. 2*, John Wiley, New York, N.Y., 1971.

N. L. Johnson and S. Kotz, *Distributions in Statistics: Discrete Distributions*, John Wiley, New York, N.Y., 1969.

M. Kendall and A. Stuart, *The Advanced Theory of Statistics, Vol. 1, 4th Ed.*, Macmillan, New York, 1977.

A. G. Laurent, "Probability distributions, factorial moments, empty cell test," in: *Classical and Contagious Discrete Distributions*, pp. 437–442, Proceedings of the International Symposium on Classical and Contagious Distributions, Montreal, Pergamon Press, New York, 1965.

G. G. Lorentz, *Approximation of Functions, 2nd Ed.*, Chelsea Publisjing Company, New York, 1986.

G. Marsaglia, "Generating discrete random variables in a computer," *Communications of the ACM*, vol. 6, pp. 37–38, 1963.

A. V. Peterson and R. A. Kronmal, "On mixture methods for the computer generation of random variables," *The American Statistician*, vol. 36, pp. 184–191, 1982.

B. W. Schmeiser, "Recent advances in generating observations from discrete random variables," in: *Computer Science and Statistics: The Interface*, ed. J. E. Gentle, pp. 154–160, North-Holland, 1983.

J. Stoyanov, *Counterexamples in Probability*, John Wiley, Chichester, U.K., 1987.

A. J. Walker, "New fast method for generating discrete random numbers with arbitrary frequency distributions," *Electronics Letters*, vol. 10, pp. 127–128, 1974.

A. J. Walker, "An efficient method for generating discrete random variables with general distributions," *ACM Transactions on Mathematical Software*, vol. 3, pp. 253–256, 1977.

D. V. Widder, *The Laplace Transform*, Princeton University Press, Princeton, N.J., 1972.