Computing 26, 197—207 (1981)

Computing

The Computer Generation of Poisson Random Variables

L. Devroye, Montreal

Received May 5, 1980

Abstract — Zusammenfassung

The Computer Generation of Poisson Random Variables. An exact method for the generation of Poisson random variables on a computer is presented. The average time required per random variate decreases as the Poisson parameter tends to infinity.

Key words and Phrases: Random number generation, Poisson distribution, the rejection method, inequalities for Poisson probabilities, average complexity.

Die Erzeugung von Poisson-verteilten Zufallsvariablen im Computer. Es wird eine exakte Computermethode zur Erzeugung von Poisson-verteilten Zufallsvariablen angegeben. Die durchschnittliche pro Zufallsvariable benötigte Zeit nimmt ab, wenn der Poisson-Parameter gegen unendlich strebt.

1. Introduction

In this paper we present yet another algorithm for the computer generation of Poisson random variables. We will say that the random variable X is *Poisson* (λ) when

$$P(X=i)=p_i=\frac{\lambda^i e^{-\lambda}}{i!},\ i\geq 0.$$

Many algorithms for Poisson random variate generation are based on the property that X is distributed as the smallest integer such that

$$\sum_{i=1}^{X+1} E_i > \lambda$$

where $E_1, E_2, \ldots$ is a sequence of i. i. d. exponential random variables. Equivalently, X is distributed as the smallest integer such that

$$\prod_{i=1}^{X+1} U_i < e^{-\lambda}$$

where $U_1, U_2, \ldots$ is a sequence of i. i. d. uniform (0, 1) random variables. Both versions have average complexity $O(\lambda+1)$. The *inversion principle* states that if X is the unique integer for which

$$\sum_{i<X} p_i \leq U_1 < \sum_{i\leq X} p_i, \tag{1}$$

0010-485X/81/0026/0197/$ 02.20

then X is Poisson (λ). Sequential search starting at $i=0$ leads to an algorithm with average complexity $O(\lambda+1)$. A two-level search (determination of one of M intervals, and subsequent sequential search within that interval) may reduce the average computer time but the average complexity remains $O(\lambda+1)$ (Atkinson, 1979a). Knuth (1969) discusses all the early algorithms for Poisson random variate generation. Fishman (1976) pointed out that if the sequential search for the solution of (1) is started near the mode ($\lfloor\lambda\rfloor$), the average complexity of the algorithm becomes $O(\sqrt{\lambda}+1)$. Atkinson (1979a, 1979b) gives an excellent and interesting comparison of most Poisson variate generation algorithms. For moderate values of λ (≤ 200) the celebrated table look-up method yielded the fastest algorithms in his comparison, followed by the two-level search algorithms for the solution of (1). In both cases, the size of the programs is enormous, and the theoretical average complexity is $O(\lambda+1)$.

More recently, several attempts were made at reducing the average complexity for large values for λ. Ahrens and Dieter (1974) proposed a recursive algorithm with average complexity $O(1+\log(\lambda))$. Atkinson (1979a) was the first person to publish an $O(1)$ algorithm. His algorithm uses rejection from the logistic density, and he assumes that $\log x!$ can be computed in time $O(1)$.

Finally, one could generate n Poisson variates at a time as follows: generate X, a Poisson $(n\lambda)$ random variate. Then generate $X_1, \ldots, X_n$, a multinomial $\left(X, \frac{1}{n}, \ldots, \frac{1}{n}\right)$ random vector. The X_i's are i.i.d. Poisson (λ) (see Moran (1951), Bolshev (1965), Patil and Seshadri (1964) and Tadikamalla (1979)).

In this paper, a simple and fast algorithm with average complexity $0(1)$ is developed. It is a rejection algorithm that with probability $1-o(1)$ (as $\lambda\to\infty$) exits with a truncated normal random variable. In other words, it exploits the convergence of the Poisson distribution to the normal distribution.

2. Inequalities for Poisson Probabilities

In this paper we will need several tight bounds for the individual Poisson probabilities p_i. Most of them can be derived from the following collection of inequalities for $\log(1+u)$ and $\log(1-u)$, where $u\geq 0$.

$$
\begin{aligned}
&\text{(i)}\quad \log(1+u)\leq u\\
&\text{(ii)}\quad \log(1+u)\leq u-\frac{u^2}{2}+\frac{u^3}{3}\\
&\text{(iii)}\quad \log(1+u)\geq u-\frac{u^2}{2}\\
&\text{(iv)}\quad \log(1+u)\geq \frac{2u}{2+u}\\
&\text{(v)}\quad \log(1-u)\leq -u-\frac{u^2}{2}\ldots-\frac{u^k}{k},\ k\geq 1,\ u<1\\
&\text{(vi)}\quad \log(1-u)\geq -u-\frac{u^2}{2}-\ldots-\frac{u^k}{k(1-u)},\ k\geq 1,\ u<1.
\end{aligned}
\tag{2}
$$

Most of these inequalities are well-known. The other ones can be derived without difficulty from Taylor's theorem (see Whittaker and Watson, 1963).

Let us define the quantity

$$\begin{aligned} q(y) &= \log p_{\lambda+y} + [\log \lambda! - \lambda \log \lambda + \lambda] \\ &= y \log \lambda - \log\left(\frac{(\lambda+y)!}{\lambda!}\right), \quad y \geq -\lambda, \ \lambda \text{ and } y \text{ integer}. \end{aligned} \tag{3}$$

Clearly,

$$q(y) = \begin{cases} -\log \prod_{i=1}^{y} \left(1+\frac{i}{\lambda}\right), & y>0, \\ 0, & y=0, \\ \log \prod_{i=0}^{-y-1} \left(1-\frac{i}{\lambda}\right), & y<0. \end{cases} \tag{4}$$

Using (2) and the identities

$$\sum_{i=1}^{y} i = \frac{y(y+1)}{2}; \quad \sum_{i=1}^{y} i^2 = \frac{y(y+1)(2y+1)}{6}; \quad \sum_{i=1}^{y} i^3 = \frac{y^2(y+1)^2}{4}, \tag{5}$$

valid for y integer, we have, without further work:

Lemma 1: *If* $y_+ = \max(y, 0)$, *then*

$$q(y) \leq -\frac{y(y+1)}{2\lambda + y_+}, \quad \text{all } \textit{integer } y \geq -\lambda.$$

Proof: Use (2) (iv, v) and (5).

Lemma 2: *For* $y \geq 0$, y *integer*,

$$q(y) + \frac{y(y+1)}{2\lambda} \begin{cases} \geq 0 \\ \leq \dfrac{y(y+1)(2y+1)}{12\lambda^2} \\ \geq \dfrac{y(y+1)(2y+1)}{12\lambda^2} - \dfrac{y^2(y+1)^2}{12\lambda^3} \end{cases}.$$

For $y \leq 0$, y *integer*,

$$q(y) + \frac{y(y+1)}{2\lambda} \begin{cases} \leq 0 \\ \leq \dfrac{y(y+1)(2y+1)}{12\lambda^2} \\ \geq \dfrac{y(y+1)(2y+1)}{12\lambda^2} - \dfrac{y^2(y+1)^2}{12\lambda^2(\lambda+y+1)} \end{cases}.$$

Proof: Use (5) and (2) (i, ii, iii, v, vi).

3. Discretization

One of the most promising techniques for the generation of integer-valued random variables is the truncation of a continuous random variable, because it avoids a time-consuming search. It is very rare however that one can find a simple

continuous density for which the truncation works. One of the few examples is the exponential density: the integer-valued random variable obtained by truncation is geometrically distributed. There is of course always the possibility of applying the following version of the rejection method:

1. Generate a random variate X with density f on R^d. Generate an independent uniform (0, 1) random variate U. Determine the integer Y from $X \in A_Y$ where $\{A_i \mid i \text{ integer}\}$ is a *given partition* of R^d into Borel sets of unit volume each. (This partition is such that

$$\inf_y \inf_{x \in A_y} \frac{cf(x)}{p_y} \geq 1$$

for some constant $c \geq 1$ and a *given probability distribution* $\{p_i\}$ on the integers.)

2. If $U \cdot cf(X) \leq p_Y$, exit with Y. Otherwise, go to 1.

It is clear that the random variate Y generated by this procedure has probability distribution $\{p_i\}$ on the integers. Also, step 1 will be executed c times on the average.

We will apply this procedure to the Poisson distribution when λ is integer. To do so, we need a simple density f that dominates the probabilities $p_{\lambda+y}$, roughly speaking.

Lemma 3: *Consider the following partition of* $[-\lambda, \infty) \times [0, 1) \cup [0, 1) \times [1, 2)$:

$$A_y = \begin{cases} [y, y+1) \times [0, 1) & \text{if } y < 0,\ y \geq -\lambda \\ [y-1, y) \times [0, 1) & \text{if } y > 0 \\ [0, 1) \times [1, 2) & \text{if } y = 0. \end{cases}$$

Then

$$q(y) \leq h(x, z), \text{ all } (x, z) \in A_y, \text{ all integer } y \geq -\lambda,$$

where

$$h(x, z) = \begin{cases} -\left(x + \frac{1}{2}\right)^2 / (2\lambda + \delta) + c_1, & x < \delta,\ 0 \leq z < 1, \\ -(x+1)\frac{\delta}{2\lambda + \delta}, & x \geq \delta,\ 0 \leq z < 1, \\ 0, & 0 \leq x < 1,\ 1 \leq z < 2. \end{cases} \tag{6}$$

Here $\delta > 0$ *is a* given *constant, and*

$$c_1 = \frac{1}{8\lambda}.$$

Proof: By Lemma 1, it suffices to show that for $(x, z) \in A_y$, $h(x, z) \geq -\frac{y(y+1)}{2\lambda + y_+}$. Consider first $y < 0$. Clearly, $-y(y+1) \leq -x(x+1)$ when $(x, z) \in A_y$, i.e., $y \leq x < y+1$. Also, $-\frac{x(x+1)}{2\lambda} \leq -\frac{x(x+1)}{2\lambda + \delta} + \frac{\delta}{4(2\lambda)(2\lambda + \delta)} = -\frac{(x+\frac{1}{2})^2}{2\lambda + \delta} + c_1$. The case $y = 0$ is trivial. Finally, for $y > 0$, we have $-\frac{y(y+1)}{2\lambda + y} \leq -\frac{x(x+1)}{2\lambda + x}$ when $(x, z) \in A_y$, i.e.,

$y-1 \le x < y$. The last expression is not greater than $-\frac{x(x+1)}{2\lambda+\delta}$ when $x<\delta$, and it is not greater than $-\frac{\delta(x+1)}{2\lambda+\delta}$ when $x \ge \delta$. This concludes the proof of Lemma 3.

The function $e^{h(x,z)}$ is proportional to a density:

$$p_1 f_1(x) g(z) + p_2 f_2(x) g(z) + p_3 g(x) g(z-1). \tag{7}$$

Here $g(.)$ is the uniform density on (0, 1), $f_1(.)$ is the normal density with mean $-\frac{1}{2}$ and variance $\lambda+\frac{\delta}{2}$, truncated at δ, and $f_2(.)$ is the right-tailed exponential density with origin at δ and with shape parameter $\frac{\delta}{2\lambda+\delta}$. The p_i's are positive numbers that sum to 1. Unfortunately, the p_i's depend upon the error function in view of the truncation of $f_1(.)$. To eliminate this annoying dependence, we consider a new function $h(x, z)$ which is defined as (6) except that on $x \ge \delta$, $0 \le z < 1$, we set

$$h(x,z) = -(x+1)\frac{\delta}{2\lambda+\delta} - \left(x+\frac{1}{2}\right)^2 (2\lambda+\delta)^{-1} + c_1. \tag{8}$$

Now, $e^{h(x,z)}$ is proportional to the density (7) with the understanding that $f_1(.)$ is the usual normal density with mean $-\frac{1}{2}$ and variance $\lambda+\frac{\delta}{2}$. The p_i's are easily determined: $p_i = a_i/(a_1+a_2+a_3)$, $i=1, 2, 3$, where

$$a_1 = \sqrt{\pi(2\lambda+\delta)}\exp\left(\frac{1}{8\lambda}\right);$$

$$a_2 = \frac{2\lambda+\delta}{\delta}\exp\left(-\frac{\delta(\delta+1)}{2\lambda+\delta}\right);$$

$$a_3 = 1.$$

Using the fact that the logarithm of a uniform (0, 1) random variable is distributed as minus an exponential random variable, we can state the following rough form for our algorithm for Poisson random variate generation.

1. Generate U uniform (0, 1). If $U > p_1$, go to 3.
2. Generate a normal (0, 1) random variable N and set $X \leftarrow N\sqrt{\lambda+\frac{\delta}{2}} - \frac{1}{2}$. If $X > \delta$ or $X < -\lambda$, go to 1. Set $Y \leftarrow \underline{X}$ if $X < 0$, and $Y \leftarrow \overline{X}$ otherwise. Generate E exponential (1), and set $V \leftarrow -E - \frac{N^2}{2} + c_1$. Go to 4.
3. If $U > p_1 + p_2$, exit with $Y \leftarrow \lambda$. Otherwise, generate two independent exponential (1) random variables E_1 and E_2. Set $X \leftarrow \delta + \frac{2\lambda+\delta}{\delta}E_1$, $Y \leftarrow \overline{X}$ and $V \leftarrow -E_2 - \frac{\delta(X+1)}{2\lambda+\delta}$.
4. If $V < q(Y)$, exit with $Y \leftarrow \lambda + Y$. Otherwise, go to 1.

Remark 1: The function (7) is not smaller than $c\, p_{\lambda+y}$ for all $(x, z) \in A_y$. It should be clear now that z is totally artificial, and that it is essentially redundant since it is only used to separate the third component in the mixture from the other two. Taking logarithms and canceling constants on both sides of the basic inequality gives, instead of (6), for integer values of y,

$$q(y) \leq -\frac{(x+\frac{1}{2})^2}{2\lambda+\delta} + c_1, \quad -\lambda \leq x < 0,\ y \leq x < y+1 \text{ or } 0 \leq x < \delta,\ y-1 \leq x < y;$$
$$q(0) = 0; \tag{9}$$
$$q(y) \leq -(x+1)\frac{\delta}{2\lambda+\delta}, \quad x \geq \delta,\ y-1 \leq x < y.$$

The random variable X in the algorithm is only defined if the third component in the mixture (7) is not selected (see steps 2 and 3). When the third component is selected, no test is necessary, and we can immediately exit with the value λ. The random variable V is equal to the right-hand-side of (9) evaluated at X minus an independent exponential random variable. V is then compared to $q(Y)$ in the acceptance/rejection step 4.

Remark 2: The value of the threshold δ is not determined yet. Since $p_{\lambda+y}$ is bounded from above by a constant depending upon λ only, times $\exp(h(x, z))$, $(x, z) \in A_y$, it would be natural to choose δ such that the area under $\exp(h(x, z))$ is minimized. With the modification (8), this area is $a_1 + a_2 + a_3$. Minimizing this expression with respect to δ can only be done numerically. However, since λ is large, $\frac{\delta}{\lambda}$ is small (or so we expect) and δ is large (compared to 1), we may try to minimize the following good approximation of $a_1 + a_2$:

$$\sqrt{2\pi\lambda}\left(1+\frac{\delta}{4\lambda}\right)+\frac{2\lambda}{\delta}\exp\left(-\frac{\delta^2}{2\lambda}\right).$$

The derivative with respect to δ is

$$\frac{\sqrt{2\pi\lambda}}{4\lambda}-2\left(\frac{\lambda}{\delta^2}+1\right)\exp\left(-\frac{\delta^2}{2\lambda}\right).$$

This is 0 when

$$\delta^2 = 8\sqrt{\frac{\lambda}{2\pi}}(\lambda+\delta^2)\exp\left(-\frac{\delta^2}{2\lambda}\right).$$

Again, an explicit solution is not available, but a good approximate solution is $\delta^2 = \lambda \log(b\lambda)$ for some constant b. Resubstitution in the equation gives

$$\lambda \log(b\lambda) = \frac{8}{\sqrt{2\pi b}}\lambda \log(e b \lambda).$$

Equating the coefficients of $\lambda \log(\lambda)$ yields $b = \frac{32}{\pi}$. The value that we suggest for δ is

$$\delta = \sqrt{\lambda \log\left(\frac{32\lambda}{\pi}+1\right)}. \tag{10}$$

It can easily be checked that with δ as in (10),

$$a_1+a_2+a_3=\sqrt{2\pi\lambda}\,(1+o(1)) \quad \text{as} \quad \lambda\to\infty. \tag{11}$$

In fact, one can show that $a_2=O\left(1/\sqrt{\log(\lambda)}\right)$ as $\lambda\to\infty$.

Remark 3: The average number of times step 1 is executed is

$$\begin{aligned}&(a_1+a_2+a_3)/\exp(\log\lambda!-\lambda\log\lambda+\lambda)\\&\sim\sqrt{2\pi\lambda}/\sqrt{2\pi\lambda}=1\end{aligned}$$

as $\lambda\to\infty$ (see (3) and (11)). Here we used Stirling's approximation of the factorial (see Watson and Whittaker (1963, pp. 251—253)). This does not imply that the average complexity of the algorithm is $O(1)$ because the evaluation of $q(y)$ takes time $O(1+|y|)$. If X is the random variable defined in steps 1–3 (with $X=0$ if $U>p_1+p_2$), then the average complexity of the algorithm is $O(1+E(|X|))$. If we choose δ as in (10), then $E(|X|)=O(\sqrt{\lambda})$ as $\lambda\to\infty$. In the next section we will see how we can reduce the average complexity to $O(1)$ by the proper use of the squeeze principle.

4. The Squeeze Principle

In step 4 of the algorithm we need to evaluate $q(Y)$, defined in (4). This requires the multiplication of Y or $-Y-1$ numbers. We would like to avoid this time-consuming step as often as possible. In view of Lemma 2, step 4 can be replaced by the following steps.

1. Let A be the indicator of $X<0$ ($A=1$ if $X<0$, $A=0$ otherwise), and let $T\leftarrow\dfrac{Y(Y+1)}{2\lambda}$.
2. (Quick acceptance.) If $V<-T$ and $A=0$, exit with $Y\leftarrow\lambda+Y$.
3. Compute $q_r=T\left(\dfrac{2Y+1}{6\lambda}-1\right)$, $q_a=q_r-\dfrac{T^2}{3(\lambda+A(Y+1))}$.
4. (Quick acceptance.) If $V<q_a$, exit with $Y\leftarrow\lambda+Y$.
5. (Quick rejection.) If $V>q_r$, go to 1.
6. If $V<q(Y)$, exit with $Y\leftarrow\lambda+Y$. Otherwise, go to 1. Note: q is evaluated using (4).

In view of remark 3, we have E (number of executions of step 6.) $=(1+o(1))$ P (6. executed on first pass). The latter probability is

$$P(C;q_a<V<q_r)$$

where C is the event "$-\lambda\le X<\delta$ and $U<p_1$; or $p_1\le U<p_1+p_2$". Using I_C for the indicator function of the event C, and noting that conditional on X, the random variable V is exponentially distributed, we have

$$P(C;q_a<V<q_r)\le E((q_r-q_a)I_C)=E\left(\frac{Y^2(Y+1)^2}{12\lambda^2(\lambda+(Y+1)I_{Y<0})}I_C\right).$$

The random variable under the expectation sign, say Z, is split up into Z_1+Z_2 where $Z_1=ZI_{p_1<U<p_1+p_2}$, $Z_2=ZI_{U<p_1}$. Now, given that Y is defined as in step 3 of the algorithm (case $p_1<U<p_1+p_2$), we have

$$E\left(Y^2(Y+1)^2\right)=O\left(1+\delta^4+\left(\frac{2\lambda+\delta}{\delta}\right)^4\right)=O(1+\lambda^2\log^2\lambda).$$

Hence,

$$E(Z_1)\leq p_2\ O\left(\frac{\log^2\lambda}{\lambda}\right)=O\left(\frac{1}{\sqrt{\lambda\log\lambda}}\right)O\left(\frac{\log^2\lambda}{\lambda}\right)=O\left(\left(\frac{\log\lambda}{\lambda}\right)^{\frac{3}{2}}\right).$$

Here we used the fact that $a_1+a_2+a_3\sim\sqrt{2\pi\lambda}$ as $\lambda\to\infty$, and that

$$a_2=O\left(\frac{1}{\sqrt{\log\lambda}}\right)\text{ as }\lambda\to\infty.$$

Next, Z_2 can be further split up into Z_3+Z_4 where $Z_3=Z_2I_{Y+1<-\lambda/2}$ and $Z_4=Z_2I_{Y+1\geq-\lambda/2}$. Clearly,

$$E(Z_4)\leq E\left(\frac{Y^2(Y+1)^2}{18\lambda^3}I_C\,I_{U<p_1}\right)=O\left(\frac{1}{\lambda}\right)\text{ as }\lambda\to\infty.$$

Here we used the fact that Y is at most 1 away from a normal $\left(-\frac{1}{2},\lambda+\frac{\delta}{2}\right)$ random variable (see step 2 of algorithm). Finally,

$$E(Z_3)\leq\frac{\lambda^4}{12\lambda^3}P\left(Y+1<-\frac{\lambda}{2},U<p_1,C\right)\leq O(e^{-\lambda/5})\text{ as }\lambda\to\infty.$$

Here we used the fact that $p_1=1-o(1)$, and that for a standard normal random variable N, we have

$$P(N<-x_n)\sim\frac{1}{\sqrt{2\pi}\,x_n}\exp(-x_n^2/2)$$

when x_n is a sequence of numbers with $x_n\to\infty$. Collecting bounds allows us to conclude that

$$E\text{ (number of executions of step 6.)}=O\left(\frac{1}{\lambda}\right)\text{ as }\lambda\to\infty. \tag{12}$$

The average complexity of the complete algorithm is bounded by

$$\begin{aligned}&O\left(1+E\left(|Y|\,I_C\,I_{q_a<V<q_r}\right)\right)\\&\leq O(1)+\left[E\left(Y^2\,I_C\right)P(C;q_a<V<q_r)\right]^{1/2}\\&=O(1)\text{ as }\lambda\to\infty\end{aligned}$$

because $E(Y^2\,I_C)=O(\lambda)$ and $P(C;q_a<V<q_r)=O(\lambda^{-1})$ (12). Here we used the Cauchy-Schwartz inequality. By using an argument similar to the one leading to (12), one can show more: the average complexity due to step 6. is $o(1)$ as $\lambda\to\infty$! In other words, step 6. becomes asymptotically negligible, and the speed with which it is executed has practically no impact upon the overall average speed of the algorithm. Also, p_2 decreases very quickly (it is about 0.003 for $\lambda=1000$ and 0.0007 for $\lambda=10000$!) so that the exponential tail is hardly ever used.

5. Practical Considerations

The algorithm outlined in this paper was coded in FORTRAN and run on McGill University's AMDAHL V7 computer. The supporting random number generators were taken from McGill University's Super-Duper random number generator package. In particular, we used the uniform (0, 1), exponential and normal random number generators from this package.

The algorithm outlined in the previous sections is only valid for integer λ. For fractional λ, we generate a Poisson (λ) random variate Y as $Y_1 + Y_2$ where Y_1 is Poisson ($\underline{\lambda}$), and Y_2 is Poisson ($\lambda - \underline{\lambda}$) and independent of Y_1. For Y_2, the multiplication method is used: Y_2 is distributed as the smallest integer such that

$$\prod_{i=1}^{Y_2+1} U_i < e^{-(\lambda-\underline{\lambda})}$$

where $U_1, U_2, \ldots$ is a sequence of i.i.d. uniform (0, 1) random variables. In Table 1 we give the average time needed per variate for the new algorithm (called IP) and for the multiplication method. The program IP takes 1980 bytes compared to 482 for the code for the multiplication method. For $\lambda > 9$, algorithm IP was faster. IP is especially recommended for very large values of λ.

Table 1. *Average times in microseconds per Poisson variate. Times were obtained by averaging over 5000 runs and rounding off. Star (*) indicates that no experiments were carried out*

λ	Algorithm IP	Multiplication method
1	49	13
2	47	17
4	46	25
8	44	40
16	42	70
32	41	129
64	40	*
128	39	*
256	38	*
512	37	*
1024	36	*
2048	36	*
4096	36	*
8192	36	*
16384	36	*

```
      INTEGER FUNCTION IP(SL)
C
C     THE SUBPROGRAM IP PRODUCES RANDOM POISSON VARIATES WITH PARAMETER
C     SL>0. A REJECTION ALGORITHM WITH SQUEEZING IS USED.
C
C     AUXILIARY SUBPROGRAMS REQUIRED :
C
C       UNI.... UNIFORM (0,1) RANDOM VARIATE GENERATOR
C       RNOR... STANDARD NORMAL RANDOM VARIATE GENERATOR
C       REXP... EXPONENTIAL RANDOM VARIATE GENERATOR
```

```
C
      DATA L/0/,SLM,D,D2,D3,STDEV,PTAIL,PBODY,CON,RL,RI,TWO,EL/12*0./
      IF(SL.EQ.SLM) GO TO 10
      SLM=SL
      L=IFIX(SL)
      RL=FLOAT(L)
      EL=EXP(RL-SL)
      IF(L.EQ.0) GO TO 99
      RI=1./RL
      TWO=RL+RL
      D=SQRT(RL*ALOG(1.+10.18593*RL))
      D2=D+TWO
      D3=D2/D
      STDEV=SQRT(0.5*D2)
      PTAIL=D3*EXP(-(D+1.)/D3)
      CON=0.25/TWO
      PBODY=EXP(CON)*SQRT(3.14159*D2)
      SUM=PTAIL+PBODY+1.
      PBODY=1./SUM
      PTAIL=PBODY+PTAIL/SUM
10    IF(L.EQ.0) GO TO 99
1     A=0.
      IP=0
      U=UNI(0)
      IF(U.LT.PTAIL) GO TO 50
      R=RNOR(0)
      X=R*STDEV-0.5
      IF(X.GT.D.OR.X.LT.-RL) GO TO 1
      IF(X.GT.0.) GO TO 18
      A=1.
      X=X-2.
18    IP=IFIX(X+1.)
      Y=FLOAT(IP)
      V=-REXP(0)-0.5*R**2+CON
19    T=Y*(Y+1.)/TWO
      IF(V.LT.-T.AND.A.EQ.0.) GO TO 100
      QR=T*(-1.+(Y+Y+1.)*0.1666667*RI)
      QA=QR-T**2*0.3333333/(RL+(Y+1.)*A)
      IF(V.LT.QA) GO TO 100
      IF(V.GT.QR) GO TO 1
      RM=RI*(1.-2*A)
      K=-IP-1
      IF(IP.GT.0) K=IP
      PD=1.
      S=0.
      DO 20 J=1,K
      S=S+RM
20    PD=PD*(1.+S)
      IF(V.LT.(2*A-1.)*ALOG(PD)) GO TO 100
      GO TO 1
50    IF(U.LT.PBODY) GO TO 100
      X=D+D3*REXP(0)
      IP=IFIX(X+1.)
      Y=FLOAT(IP)
      V=-REXP(0)-(X+1.)/D3
      IF(V.GT.-Y*(Y+1.)/(TWO+Y)) GO TO 1
      GO TO 19
99    IP=0
100   IP=IP+L
      PD=UNI(0)
110   IF(PD.LT.EL) RETURN
      IP=IP+1
      PD=PD*UNI(0)
      GO TO 110
      END
```

Note added in proof:

When this paper was already in press, the author was informed that J. H. Ahrens and U. Dieter have published two $O(1)$ algorithms for Poisson random variates (see for example, "Sampling from binomial and Poisson distributions: a method with bounded computation times", Computing, 1980). B. Schmeiser (Purdue University) informed me that he also obtained an $O(1)$ Poisson generator (unpublished technical report). Finally the author wishes to thank Dr. Klaus-D. Kohrt for pointing out a bug in the original FORTRAN program.

References

Ahrens, J. H., Dieter, U.: Computer methods for sampling from gamma, beta, Poisson and binomial distributions. Computing *12*, 223—246 (1974).

Atkinson, A. C.: The computer generation of Poisson random variables. Applied Statistics *28*, 29—35 (1979a).

Atkinson, A. C.: Recent developments in the computer generation of Poisson random variables. Applied Statistics *28*, 260—263 (1979b).

Bolshev, L. N.: On a characterization of the Poisson distribution and its statistical applications. Theory of Probability and its Applications *10*, 446—456 (1965).

Fishman, G. S.: Sampling from the Poisson distribution on a computer. Computing *17*, 147—156 (1976).

Knuth, D. E.: The Art of Computer Programming, Vol. 2. Reading, Mass.: Addison-Wesley 1969.

Moran, P. A. P.: A characteristic property of the Poisson distribution. Proceedings of the Cambridge Philosophical Society *48*, 206—207 (1951).

Patil, G. P., Seshadri, V.: Characterization theorems for some univariate probability distributions. Journal of the Royal Statistical Society *B-26*, 286—292 (1964).

Tadikamalla, P. R.: A simple method for sampling from the Poisson distribution. Working Paper 365, Graduate School of Business, University of Pittsburgh, Pittsburgh, Pennsylvania, 1979.

Whittaker, E. T., Watson, G. N.: A Course of Modern Analysis, 4th ed. Cambridge: Cambridge University Press 1963.

Dr. L. Devroye
School of Computer Science
McGill University
805 Sherbrooke Street West
Montreal, Canada H3A 2K6