The Transformed Rejection Method for Generating Poisson Random Variables

W. Hörmann

Abbreviated Title: Poisson random variate generation

Abstract:

The transformed rejection method, a combination of the inversion and the rejection method, which is used to generate non-uniform random numbers from a variety of continuous distributions can be applied to discrete distributions as well. For the Poisson distribution a short and simple algorithm is obtained which is well suited for large values of the Poisson parameter μ, even when μ may vary from call to call. The average number of uniform deviates required is lower than for any of the known uniformly fast algorithms. Timings for a C implementation show that the algorithm needs only half of the code but is - for μ not too small - at least as fast as the current state-of-the-art algorithms.

Keywords:

Rejection method; Inversion; Decomposition; Poisson variate generation; Uniformly fast algorithm

1. Introduction

Several rejection algorithms to generate Poisson variates were suggested in the literature (eg. [3], [4] and [12]). These algorithms are uniformly fast, i.e. they have bounded execution time over the defined range of the Poisson parameter μ. In addition the set-up time to recalculate constants when μ changes is short which makes them especially suited for the varying parameter situation. On the other hand the faster of these algorithms are quite complicated and the expected number of uniform deviates required is greater than two. This is high compared to the inversion and table methods which need only one uniform deviate. To reduce the number of uniform deviates required by a rejection algorithm for continuous densities two main ideas are known: In [5] they are called acceptance-complement and almost-exact inversion methods. The acceptance complement method was used to sample from the Poisson distribution in [1]. The resulting algorithm is fast but very long and complicated and requires a fast normal generator. Therefore we decided to develop an almost-exact inversion algorithm for discrete distributions. Following C. S. Wallace, who suggested the main idea of this method for continuous distributions in [14], we call it transformed rejection method. In [11] it is called exact-approximation.

2. The method

If we use the transformed rejection method to sample from a continuous distribution with inverse distribution function $F^{-1}(u)$ and density $f(x)$ we need an easy to compute transformation $G(u)$ which is close to F^{-1}. Then we use rejection to generate a random number U with density $f(G(u))G'(u)$ for $0 \leq u \leq 1$ and return $X = G(U)$. It is easy to prove that X has the density function f we aimed at. If G is close to F^{-1} then $f(G(u))G'(u)$ is close to the density of the (0,1) uniform distribution. To generate discrete random variates everything remains the same. Only $f(x)$ is no longer the density function but the histogram function of the desired distribution and we have to return $X = \lfloor G(U) \rfloor$ instead of $G(U)$. As we are mainly interested in a simple and short algorithm for the Poisson distribution with large μ we restrict ourselves to the case of a symmetric transformation. Therefore it is more convenient to define $G(u)$ on the interval (-0.5,0.5). $1/\alpha$ denotes a number larger than the maximum of $f(G(u))G'(u)$. Thus α is the acceptance probability of the rejection algorithm. Now we are ready to give the basic transformed rejection algorithm for a discrete distribution X. (f denotes the histogram function of X.)

Algorithm Transformed Rejection (TR):

1: Generate two independent uniform random numbers U and V. Set $U \leftarrow U - 0.5$.

2: If $V \leq \alpha f(G(U))G'(U)$ return $\lfloor G(U) \rfloor$, else go to 1.

To generate one discrete random deviate Algorithm 1 requires $2/\alpha$ uniform random numbers and $1/\alpha$ evaluations of the acceptance condition on average. But if $\alpha f(G(u))G'(u)$ is close to 1 over the interval (-0.5,0.5) it is possible to find a large rectangle below the curve which will be denoted by $(-u_r, u_r) \times (0, v_r)$. Using this rectangle as a simple squeeze yields the following:

Algorithm Transformed Rejection with Squeeze (TRS):

1: Generate two independent uniform random numbers U and V. Set $U \leftarrow U - 0.5$.

2: If $|U| \leq u_r$ and $V \leq v_r$ return $\lfloor G(U) \rfloor$.

3: If $V \leq \alpha f(G(U))G'(U)$ return $\lfloor G(U) \rfloor$, else go to 1.

For Algorithm TRS the expected number of evaluations of the acceptance condition is reduced to $(1 - 2u_r v_r)/\alpha$ whereas the expected number of uniform deviates required remains unchanged. To reduce this number the idea of decomposition is necessary; this means that with probability $2u_r v_r$ we generate a uniform random number over $(-u_r, u_r)$ otherwise we generate a pair (U, V) inside the rectangle $(-0.5, 0.5) \times (0, 1)$ but outside $(-u_r, u_r) \times (0, v_r)$. The technique to reuse the uniform random number over the interval $(0, 2u_r v_r)$ in the next steps of the algorithm is called "recycling" in [5].

Algorithm Transformed Rejection with Decomposition (TRD):

1: Generate a uniform random number V. If $V \leq 2u_r v_r$ return $\lfloor G(V/v_r - u_r) \rfloor$.

2: If $V \geq v_r$ generate a uniform random number U in (-0.5,0.5),

else set $U \leftarrow V/v_r - (u_r + 0.5)$, $U \leftarrow \text{sign}(U)0.5 - U$, generate a uniform random number V in $(0, v_r)$.

3: If $V \leq \alpha f(G(U))G'(U)$ return $\lfloor G(U) \rfloor$, else go to 1.

For Algorithm TRD the expected number of evaluations of the acceptance condition is the same as for Algorithm TRS whereas the expected number of uniform deviates required to generate one discrete deviate is reduced to $(2 - 2u_r v_r)/\alpha$.

The choice of the transformation $G(u)$ of course depends on the desired distribution. The class $G(u) = \left(\frac{2a}{1/2-|u|} + b\right) u$, $-0.5 \le u \le 0.5$, (which was first suggested in [14] for the normal distribution) can be used for any distribution that has a bounded density function with subquadratic tails and yields high acceptance probabilities for a variety of continuous distributions (for the normal distribution over 89 percent). Therefore together with a location parameter the above transformation should be well suited at least for the Poisson the binomial and the hypergeometric distribution.

3. Application to the Poisson distribution

To apply Algorithms TRS and TRD with $G(u) = \left(\frac{2a}{1/2-|u|} + b\right) u + c$, $G'(u) = \frac{a}{(1/2-|u|)^2} + b$ to the Poisson distribution with histogram $f(x) = e^{-\mu}\mu^{\lfloor x \rfloor}/\lfloor x \rfloor!$ we determined optimal values for a, b and c for many values of the Poisson parameter μ by numerical search. As there are simple, short and very fast inversion algorithms for low values of μ we decided to approximate the optimal values of a, b and c by simple functions in the range of $\mu \ge 10$. For these approximations it was necessary to calculate and find easy lower bounds for α, u_r and v_r. Finally we arrived at the values contained in Table 1.

Table 1: Approximations valid for $\mu \geq 10$

c	$\mu + 0.445$
b	$0.931 + 2.53\sqrt{\mu}$
a	$-0.059 + 0.02483b$
$1/\alpha$	$1.1239 + 1.1328/(b - 3.4)$
u_r	0.43
v_r	$0.9277 - 3.6224/(b - 2)$

To demonstrate the good fit of these approximations even for comparable small values of μ Figure 1 shows $\alpha f(G(u))G'(u)$, and the rectangle $(-u_r, u_r) \times (0, v_r)$, Figure 2 presents the Poisson histogram and $1/\alpha$-times the density of the distribution corresponding to G, both for $\mu = 20$.

Figure 1; Figure 2;

The validity of α and v_r in Table 1 was checked numerically. That they are asymptotically correct can be seen easily: As the Poisson histogramm tends to the density function of the normal distribution we must first calculate the limits of the standardized parameters

$$\lim_{\mu\to\infty} b_s = \lim(0.931 + 2.53\sqrt{\mu})/\sqrt{\mu} = 2.53, \quad \lim_{\mu\to\infty} a_s = 2.53 \cdot 0.02483, \quad \lim_{\mu\to\infty} c_s = 0.$$

For the standard normal distribution the transformation G with the limits of a_s, b_s and c_s as parameters yields an acceptance probability α of 0.89004. For $u_r = 0.43$ the maximal v_r/α is 1.04414. The fact that the limit of $\alpha(\mu)$ is 1/1.1239=0.8897 and the limit of $v_r(\mu)/\alpha(\mu)$ is 1.04264 implies that the choice of $\alpha(\mu)$ and $v_r(\mu)$ is asymptotically correct.

Due to the heavy tails of the distribution associated with G the curve $\alpha f(G(u))G'(u)$ is very close to 0 for $|u|$ close to 0.5 for any μ (see also Fig. 1). As most of the time of the Algorithms TRS and TRD is spent in evaluating the acceptance condition Algorithm PTRD below is accelerated by using the very simple outer squeeze $(0.5 - |u|)$ for $|u| \geq 0.487$, which is valid for any $\mu \geq 10$.

The only problem left is the evluation of the Poisson probabilities $f(k)$. Numerically it is more convenient to compute $\log(f(k))$. For $k \le 9$ the values of $\log(k!)$ can be stored in a table. For $k \ge 10$ using the Stirling approximation results in the following acceptance condition: ($\lfloor G(u) \rfloor$ is denoted by k.)

$$\log\left(\frac{V}{\alpha G'(u)}\right) \le -\mu + k\log(\mu) - \log\sqrt{2\pi} - (k+1/2)\log(k) + k - \frac{1}{12k} + \frac{1}{360k^3}$$

For large values of μ the right hand side is a difference of large expressions and thus cancellation can occur. This problem is less serious in the below reformulation as the biggest term is μ or k instead of $k\log(\mu)$ or $(k+1/2)\log(\mu/k)$. As $\sqrt{\mu}$ must be computed to get b in any case the below reformulation is faster as well.

$$\log\left(\frac{V\sqrt{\mu}}{\alpha G'(u)}\right) \le -\mu + (k+1/2)\log(\mu/k) - \log\sqrt{2\pi} + k - \frac{1}{12k} + \frac{1}{360k^3}$$

The truncation error of the Stirling approximation is smaller than $7.9\ 10^{-9}$ for $k \ge 10$, the loss of accuracy due to cancellation is about n decimal digits when $\mu = 10^n$, as μ or k are the biggest terms on the right hand side. Therefore it is justified to say that the last version of the acceptance condition is accurate for $10 \le \mu \le 10^8$, implemented on computers with 64 bit floating point numbers and 32 bit integers and uniform random numbers; in this group fall all PC's with coprocessor and most work-stations. Putting together Algorithm TRD and the above results in the following:

Algorithm PTRD ($\mu \ge 10$)

0: [Set-up] Prepare Table of $\log(k!)$ $(k = 0, \cdots, 9)$;

set $smu \leftarrow \sqrt{\mu}$, $b \leftarrow 0.931 + 2.53 smu$, $a \leftarrow -0.059 + 0.02483b$,

$(1/\alpha) \leftarrow 1.1239 + 1.1328/(b-3.4)$, $v_r = 0.9277 - 3.6224/(b-2)$.

1: Generate a uniform random number V.

If $V \le 0.86 v_r$ set $U \leftarrow V/v_r - 0.43$ and return $\lfloor (2a/(0.5-|U|)+b)U + \mu + 0.445 \rfloor$.

2: If $V \geq v_r$ generate a uniform random number U in (-0.5,0.5),

else set $U \leftarrow V/v_r - 0.93$, $U \leftarrow \text{sign}(U)0.5 - U$ and generate a uniform random number V in $(0, v_r)$.

3.0: Set $us \leftarrow 0.5 - |U|$. If $us < 0.013$ and $V > us$ go to 1.

3.1: Set $k \leftarrow \lfloor (2a/us+b)U+\mu+0.445 \rfloor$, $V \leftarrow V*(1/\alpha)/(a/us^2+b)$. If $k \geq 10$ and $\log(V*smu) \leq (k+0.5)*\log(\mu/k) - \mu - \log\sqrt{2\pi} + k - (1/12 - 1/(360*k*k))/k$ return k.

3.2: If $0 \leq k \leq 9$ and $\log(V) \leq k*\log(\mu) - \mu - \log(k!)$ return k, else go to 1.

If simplicity of the method and readability of the program is the main concern it is better to use Algorithm TRS and to evaluate $\log(k!)$ with the Stirling approximation in a separate function. Then we get.

Algorithm PTRS ($\mu \geq 10$)

0: [Set-up] Prepare function of $\log(k!) = \log\sqrt{2\pi} + (k+1/2)\log(k) - k + (1/12 - 1/(360k^2))/k$ for $k \geq 10$ and a table for $k \leq 9$;

set $b \leftarrow 0.931 + 2.53\sqrt{\mu}$, $a \leftarrow -0.059 + 0.02483b$, $v_r = 0.9277 - 3.6224/(b-2)$.

1: Generate two uniform random numbers U and V. Set $U \leftarrow U - 0.5$, $us \leftarrow 0.5 - |U|$, $k \leftarrow \lfloor (2a/us + b)U + \mu + 0.43 \rfloor$.

2: If $us \geq 0.07$ and $V \leq v_r$ return k. If $k < 0$ go to 1; if $us < 0.013$ and $V > us$ go to 1.

3.0: [Preparation for 3.1] Set $(1/\alpha) \leftarrow 1.1239 + 1.1328/(b-3.4)$, $lnmu \leftarrow \log(\mu)$.

3.1: If $\log(V*(1/\alpha)/(a/us^2+b) \leq -\mu + k*lnmu - \log(k!)$ return k, else go to 1.

To test the accuracy of the two algorithms empirically we programmed the same algorithms without squeezes and with the acceptance conditions in the from: $V \leq f(k)\alpha G'(U)$. $f(k)$ was computed recursively from the exact modal probabilities, which were stored in a table. For

$\mu = 10^n$, $1 \leq n \leq 8$ we compared the results of Algorithms PTRD and PTRS with the "exact" algorithms (samplesize $5 \cdot 10^8$ for every μ). The empirical results given in Table 1 support the considerations above: Algorithm PTRD can be safely used for $10 \leq \mu \leq 10^8$ Algorithm PTRS for $10 \leq \mu \leq 10^7$.

Table 2: Number of differences for $5 \cdot 10^8$ random numbers

μ	10	10^2	10^3	10^4	10^5	10^6	10^7	10^8
PTRD	1	0	0	0	0	0	1	4
PTRS	1	0	0	0	0	0	7	66

It is possible to design a Poisson generator based on the idea of transformed rejection which is faster than PTRD and PTRS especially for small values of μ by using the rescaled Poisson histogram which can be evaluated recursively for k close to the mode of the distribution. As we are convinced that simplicity and readability of a program are as important as speed we do not describe that algorithm which is much longer than PTRD.

4. Comparison of algorithms

In this section we compare our new Algorithms PTRD and PTRS with four known Poisson generators with short set-up times and without large tables. The acceptance-complement algorithm PD (cf. [1]) and the rejection algorithm PTPE (cf. [12]) are known to be the fastest Poisson generators with bounded computation time (cf. [13]). The ratio of uniforms algorithm PRUA (cf. [2]) and [13]) and the search from the mode inversion algorithm KEMPOIS (cf. [9]) are both recent and relatively short. A first important theoretical performance measure is the time complexity of the algorithms (KEMPOIS has $O(\sqrt{\mu})$ all others $O(1)$), a second one is the mean number N of uniforms needed to generate one Poisson random number. The values of N for the six generators and different values of μ are given in Table 3. The normal samples used for PD are generated by the acceptance complement algorithm ACR (cf. [6], $N_{ACR} = 1.49$), the exponential samples by inversion.

Table 3: Number of uniforms required

μ	10	50	100	1000	10000
PTRD	2.19	1.66	1.56	1.41	1.37
PTRS	2.66	2.39	2.35	2.28	2.29
PD	1.88	1.93	1.95	1.97	1.98
PTPE	3.25	2.61	2.38	2.28	2.30
PRUA	3.23	2.95	2.89	2.78	2.75
KEMPOIS	1.00	1.00	1.00	1.00	1.00

Table 3 shows that PTRD needs less uniform deviates than any other uniformly fast algorithm, PTRS has the largest acceptance probabilty among the rejection algorithms. Together with the time complexity of KEMPOIS this means that compared with all known generators the performance of PTRD becomes better for growing μ and for slower (and thus hopefully better) uniform generators. To compare the execution times of the algorithms we coded them in C and measured the speed of these C-implementations on a PC-386 with the UNIX C-compiler, on a PC 386 with Turbo-C and on a DEC station 5200 with the ULTRIX C-compiler and optimization -O3. As the main relations of the execution times did not change much between the computers we tested, Table 4 contains only the results of the DEC station as an example. The times given are the averages of 10^6 replications for each combination of μ and algorithm and are accurate to within 0.2 μ-seconds. Of course comparisons of the execution times are strongly influenced by the speed of the uniform generator. Among the three uniform generators we used (two linear congruential generator with modulus $m = 2^{32}$ multiplier $a = 69069$ and increment $c = 1$, and $m = 2^{31} - 1$, $a = 742938285$, $c = 0$, and one multiple recursive linear congruential generator with modulus $2^{31} - 1$ taking 1.4, 2.4 and 4.5 μ-seconds respectively) we give the results for the best but slowest. The normal generator ACR combined with this uniform generator took 9.3

μ-seconds, 2.5 μ-seconds less than the normal generator suggested in [10] which was used in the comparison of [9]. Influences on the speed of the Poisson algorithms when changing the uniform generator can easily be deduced from Table 3. Column I of Table 4 contains the initialization or set-up times necessary to recalculate constants when μ changes. As a crude measure for the length and complexity of the algorithms the number of C-statements (not counting the normal generator ACR for PD and the function log(k!) for PTRS and PRUA) are given in column L.

Table 4: Execution times in μ-sec

μ	10	50	100	1000	10000	I	L
PTRD	26.1	18.3	16.6	14.2	13.6	9.5	29
PTRS	27.2	21.9	20.5	18.2	17.5	12.1-9.3	19
PD	18.7	16.6	16.3	15.8	15.7	7.6-7.1	68
PTPE	26.1	21.8	18.0	15.9	15.9	14.8	57
PRUA	28.6	30.6	30.1	29.4	29.2	22.3	19
KEMPOIS	10.1	15.1	18.8	46.3	133.2	13.3-12.5	31

Table 4 shows that combined with a relatively slow uniform generator PTRD is fastest for large values of μ whereas KEMPOIS is fastest for low ones. Our experience on different computers and with different uniform generators can be summarized as: For $\mu \geq 100$ PTRD is as fast or faster for $10 \leq \mu \leq 100$ it is not much slower than PD or PTPE although these methods are much more complicated and need about twice the code of PTRD. PTRS is not much slower either; the method is about as simple as PRUA but for large μ it is almost twice as fast. The initialization time for any of the methods is lower than the time to generate a single Poisson random number but PD's is shortest, PTRS and PTRD rank second and third.

An important advantage of PTRS is the fact that it is the only one of the six methods compared in this section which is a monotone transformation of uniform random numbers. Therefore PTRS can be used for correlation induction almost without changes. Only synchronization has to be obtained by using two random number streams similar to the idea described in [7].

5. Conclusions

The transformed rejection method is well suited to design simple algorithms for discrete distributions. For the Poisson distribution Algorithm PTRD needs the fewest uniform random numbers of all uniformly fast algorithms known up to now. Our computing experience indicates that PTRD is at least as fast as much more complicated algorithms for μ not too small. To obtain an algorithm that is very fast for the varying parameter situation for arbitrary μ we recommend to combine PTRD with one of the search algorithms explained in [9] and [8]. Depending on the computer and the uniform generator used the switchpoint should be between μ=30 and μ=80. Algorithm PTRS is very short and simple but not much slower than PTRD. In addition PTRS is especially well suited for correlation induction.

References

[1] J. H. Ahrens and U. Dieter, Computer generation of Poisson deviates from modified normal distributions, ACM Transactions on Mathematical Software 8 (1982) 163-179.

[2] J. H. Ahrens and U. Dieter, A convenient sampling method with bounded computation times for Poisson distributions, in: P. R. Nelson, Ed., The Frontiers of Statistical Computation, Simulation & Modelling (American Sciences Press, 1991) 137-149.

[3] A. C. Atkinson, The computer generation of Poisson random variables, Applied Statistics 28 (1979) 29-35.

[4] L. Devroye, The computer generation of Poisson random variables, Computing 26 (1981) 197-207.

[5] L. Devroye, Non-Uniform Random Variate Generation, (Springer, New York, 1986).

[6] W. Hörmann and G. Derflinger, The ACR method for generating normal random variables, OR Spektrum 12 (1990) 181-185.

[7] V. Kachitvichyanukul, J. Shiow-Wen and B. W. Schmeiser, Fast Poisson and binomial algorithms for correlation induction, Journal of Statistical Computation and Simulation 29 (1988) 17-33.

[8] C. D. Kemp, New algorithms for generating Poisson variates, Journal of Computational and Applied Mathematics 31 (1990) 133-137.

[9] C. D. Kemp and A. W. Kemp, Poisson random variate generation, Applied Statistics 40 (1991) 143-158.

[10] A. J. Kinderman and J. G. Ramage, Computer generation of normal random variables, Journal of the American Statistical Association 71 (1976) 893-896.

[11] G. Marsaglia, The exact-approximation method for generating random variables in a computer, Journal of the American Statistical Association 79 (1984) 218-221.

[12] B. W. Schmeiser and V. Kachitvichyanukul, Poisson random variate generation, Research Memorandum 81-4 Purdue Univesity School of Industrial Engineering, West Lafayette, 1981.

[13] E. Stadlober, Sampling from Poisson, binomial and hypergeometric distributions: Ratio of uniforms as a simple and fast alternative, Math.-Stat. Sektion 303, Forschungsgesellschaft Joanneum, Graz, 1989.

[14] C. S. Wallace, Transformed rejection generators for gamma and normal pseudorandom variables, Australian Computer Journal 8 (1976) 103-105.

Authors Address:

Wolfgang Hörmann

Inst. f. Statistik Wirtschaftuniv. Wien,

Augasse 2-6,

A-1090 Wien, Austria.

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