Volume 44, Issue 3

A gamma variate generator with shape parameter less than unity

Seiji Zenitani
Space Research Institute, Austrian Academy of Sciences

Abstract

Algorithms for generating random numbers that follow a gamma distribution with shape parameter less than unity are proposed. Acceptance-rejection algorithms are developed, based on the generalized exponential distribution. The squeeze technique is applied to our method, and then piecewise envelope functions are further considered. The proposed methods are excellent in acceptance efficiency and promising in speed.

1 Introduction

The gamma distribution is one of the most important probability distributions in statistics. It has applications in a broad range of fields in natural sciences, engineering, and social sciences. The probability density is defined by

$$f_{\text{GA}}(x; \alpha, \lambda) = \frac{1}{\lambda^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\lambda} \quad (x \geq 0) \tag{1}$$

where $\alpha \ (> 0)$ is the shape parameter, $\lambda \ (> 0)$ the scale parameter, and $\Gamma(x)$ the gamma function.

Generating a gamma variate, a random number drawn from a gamma distribution, is an important and fundamental problem in statistical computing. Owing to its importance, various numerical algorithms have been developed since the early days of computer age (Ahrens & Dieter, 1974; Best, 1983; Devroye, 1986; Marsaglia & Tsang, 2000; Kundu & Gupta, 2007; Tanizaki, 2008; Yotsuji, 2010; Kroese et al., 2011).

The nature of the gamma distribution is controlled by the shape parameter α. Accordingly, we need to use different gamma variate generators. When $1 < \alpha$, the distribution function increases from zero at $x = 0$, has a maximum at $x = (\alpha - 1)\lambda$, and then decays as x further increases. To generate a gamma variate with $\alpha > 1$, probably Marsaglia & Tsang (2000)'s algorithm is one of the best, as detailed in literature (Marsaglia & Tsang, 2000; Yotsuji, 2010; Kroese et al., 2011). When $\alpha = 1$, the distribution is reduced to the exponential distribution, and it is easy to generate a variate.

When $0 < \alpha < 1$, which is the range of our interest, the gamma distribution monotonically decreases from positive infinity at $x = 0$ to zero at $x \to \infty$. To generate a gamma variate for $0 < \alpha < 1$, various acceptance-rejection methods have been developed (Ahrens & Dieter, 1974; Best, 1983; Devroye, 1986; Kundu & Gupta, 2007). Combining power-law and exponential distributions, Ahrens & Dieter (1974) proposed a piecewise rejection method. Best (1983) extended the Ahrens & Dieter (1974) method, by adjusting a switching point and introducing a squeeze technique. Devroye (1986) developed a different rejection method, based on the exponential power distribution. As of 2024, it is employed by a Python package, `NumPy` (Baumgarten & Patel, 2022). Using a generalized exponential distribution (Gupta & Kundu, 1999), Kundu & Gupta (2007) developed a rejection method and its piecewise extensions. Finally, Tanizaki (2008) developed a ratio-of-uniforms method, which works either for $0 < \alpha < 1$ and for $1 < \alpha$.

In this note, we propose efficient gamma generators for $0 < \alpha < 1$, advancing an earlier study (Kundu & Gupta, 2007). In Section 2, we develop an acceptance-rejection algorithm, using the generalized exponential distribution. In Section 3, we construct fractional functions for the squeeze technique. In Section 4, we discuss piecewise envelope functions. In Section 5, we evaluate the performance of the proposed and previous methods. Section 6 contains discussions and the summary.

2 Base algorithm

We construct a base algorithm to generate a random variate drawn from the gamma distribution with shape $0 < \alpha < 1$. We focus on the $\lambda = 1$ case, because we can obtain

the results for $\lambda \neq 1$, by multiplying the outputs by λ.

$$f_{\mathrm{GA}}(x; \alpha, 1) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} \tag{2}$$

In line of Kundu & Gupta (2007), we employ the generalized exponential (GE) distribution (Gupta & Kundu, 1999),

$$f_{\mathrm{GE}}(x; \alpha, \lambda) = \frac{\alpha}{\lambda} (1 - e^{-x/\lambda})^{\alpha-1} e^{-x/\lambda} \quad (x \geq 0) \tag{3}$$

where $\alpha \, (> 0)$ and $\lambda \, (> 0)$ are the shape and scale parameters. Note that Gupta & Kundu (1999) and Kundu & Gupta (2007) use different conventions for the scale parameter, and we employ Gupta & Kundu (1999)'s. The cumulative distribution function (CDF) of the GE distribution is

$$F_{\mathrm{GE}}(x; \alpha, \lambda) = (1 - e^{-x/\lambda})^{\alpha} \tag{4}$$

Hereafter we consider the GE distribution with $\lambda = 1$. We generate a uniform random variate U_1 in the $(0, 1)$ range:

$$U_1 \sim U(0, 1) \tag{5}$$

Equating U_1 with (4) with $\lambda = 1$, we find that a random variate x that follows the generalized exponential distribution $f_{\mathrm{GE}(\alpha, 1)}$ is drawn by

$$x \leftarrow -\log(1 - U_1^{1/\alpha}) \tag{6}$$

To help our discussion, we use a parameter β, a function $g(x)$, and a rejection function $R_1(x)$.

$$\beta \equiv 1 - \alpha, \quad g(x) \equiv \frac{1 - e^{-x}}{x} \tag{7}$$

$$R_1(x) \equiv \left(\frac{x}{1 - e^{-x}} \right)^{\alpha-1} = (g(x))^{\beta} \tag{8}$$

The function $g(x)$ is continuous at $x = 0$. When $x \geq 0$, as $g(0) = 1$ and $g'(x) = (1 + x - e^x)/(x^2 e^x) \leq 0$, we find $0 < g(x) \leq 1$. Since $0 < \beta < 1$, we find

$$0 < R_1(x) \leq 1. \tag{9}$$

All the equal signs are met when $x = 0$.

Comparing the GE (3) and gamma distributions (2), we rewrite (2) as follows.

$$f_{\mathrm{GA}}(x; \alpha, 1) = \frac{1}{\Gamma(\alpha + 1)} R_1(x) \, f_{\mathrm{GE}}(x; \alpha, 1) \tag{10}$$

Here, the relation $\alpha \Gamma(\alpha) = \Gamma(\alpha + 1)$ is used. Eq. (10) suggests that we can apply a rejection method to the GE-distributed variate x, to obtain a gamma-distributed variate.

Using another uniform variate $U_2 \sim U(0,1)$, we evaluate the acceptance condition:

$$U_2 \leq R_1(x) \tag{11}$$

or an equivalent inequality:

$$(U_2)^{1/(1-\alpha)}x \leq U_1^{1/\alpha} \tag{12}$$

If either (11) or (12) is satisfied, then we take this number x. If it is not satisfied, we discard the number, and then we go back to the beginning. The final output follows a gamma distribution with shape α. These procedures are summarized in Algorithm 1 in Table 1. We need the total density $1/\Gamma(\alpha+1)$ to generate one gamma distribution.

Table 1: A base algorithm of our generators for $0 < \alpha < 1$.

Algorithm 1
repeat
generate $U_1, U_2 \sim U(0,1)$
$b \leftarrow U_1^{1/\alpha}, \quad x \leftarrow -\log(1-b)$
if $U_2^{1/(1-\alpha)}x \leq b$ **return** x
end repeat

3 Squeeze technique

We improve the speed of the algorithm by the so-called squeeze method. We avoid the power and exponential operations in the acceptance condition (11), by using the following inequalities for $x \geq 0$.

$$\frac{4 - \beta x}{4 + \beta x} \leq R_1(x) \leq \frac{4 + (1-\beta)x}{4 + (1+\beta)x} \tag{13}$$

To show this, we start from several inequalities:

$$e^{-x} \geq \frac{2 - x}{2 + x} \qquad \textbf{if } x \geq 0 \tag{14}$$

$$x^\beta \leq \frac{(1-\beta) + (1+\beta)x}{(1+\beta) + (1-\beta)x} \qquad \textbf{if } 0 < x \leq 1 \text{ and } 0 \leq \beta \leq 1 \tag{15}$$

$$x^\beta \geq \frac{(1-\beta) + (1+\beta)x}{(1+\beta) + (1-\beta)x} \qquad \textbf{if } x \geq 1 \text{ and } 0 \leq \beta \leq 1 \tag{16}$$

$$x^\beta \leq 1 + \beta(x-1) \qquad \textbf{if } 0 < x \text{ and } 0 \leq \beta \leq 1 \tag{17}$$

We can prove them in the following way.

$$p(x) \equiv \frac{2-x}{2+x} e^x \quad \text{which tells } p(0) = 1, \ p'(x) = \frac{-x^2 e^x}{(2+x)^2} < 0 \tag{18}$$

$$q(x) \equiv \frac{(1-\beta) + (1+\beta)x}{(1+\beta) + (1-\beta)x} \, x^{-\beta} \quad \text{which tells } q(1) = 1,$$

$$q'(x) = \frac{\beta(\beta^2 - 1)(1-x)^2 x^{-\beta-1}}{[(1+\beta) + (1-\beta)x]^2} \leq 0 \tag{19}$$

$$r(x) \equiv -x^\beta + 1 + \beta(x-1) \quad \text{which tells } r(1) = 0,$$

$$r'(x) = \beta(1 - x^{\beta-1}) < 0 \text{ for } x < 1, \ r'(x) > 0 \text{ for } x > 1 \tag{20}$$

For the left inequality of (13), we differentiate the rejection function

$$R_1'(x) = \beta \left(\frac{x - e^x + 1}{x(e^x - 1)} \right) R_1(x) \tag{21}$$

With help from the Taylor series $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$, we find

$$\frac{R_1'(x)}{R_1(x)} = -\beta \frac{\sum_{n=2}^{\infty} \dfrac{x^n}{n!}}{x \sum_{n=1}^{\infty} \dfrac{x^n}{n!}} = -\beta \frac{\sum_{n=2}^{\infty} \dfrac{1}{n} \dfrac{x^n}{(n-1)!}}{\sum_{n=2}^{\infty} \dfrac{x^n}{(n-1)!}} \tag{22}$$

In the right-hand side, all the terms in the numerator and in the denominator are positive, but the numerator terms contain additional factor of $1/2, 1/3, \cdots$. This indicates that the ratio of the numerator to the denominator is less than $1/2$. Together with the $x = 0$ case, we find

$$\frac{R_1'(x)}{R_1(x)} \geq -\frac{\beta}{2} \tag{23}$$

This suggests, together with (14),

$$R_1(x) \geq \exp\left(-\frac{\beta}{2} x\right) \geq \frac{4 - \beta x}{4 + \beta x}. \tag{24}$$

For the right inequality of (13), with help from (8), (14) and (15), we obtain

$$R_1(x) \leq \left(\frac{2}{2+x}\right)^\beta \leq \frac{4 + (1-\beta)x}{4 + (1+\beta)x} \tag{25}$$

Thus (24) and (25) prove (13).

In practice, we utilize the inequalities (13) or equivalent logical expressions, before evaluating $R_1(x)$. Translating $\beta \to (1-\alpha)$ again, we show the final algorithm in Algorithm 2 (Table 2).

Table 2: A gamma generator with the squeeze technique.

Algorithm 2
repeat
generate $U_1, U_2 \sim U(0,1)$
$b \leftarrow U_1^{1/\alpha}, \;\; x \leftarrow -\log(1-b)$
if $U_2(4 + (1-\alpha)x) \leq (4 + (\alpha-1)x)$ **return** x
if $U_2(4 + (2-\alpha)x) \leq (4 + \alpha x)$ **then**
if $U_2^{1/(1-\alpha)} x \leq b$ **return** x
end repeat

4 Piecewise envelope functions

Next, we split the envelope function for the rejection method into two parts. Across the switching point at $x = s \; (> 0)$, we consider the GE distribution in the left and an exponential tail in the right. With help from the CDF of the GE distribution (Eq. (4)), we rewrite the gamma distribution (Eq. (2)) as follows.

$$f_{\text{GA}}(x; \alpha, 1) = \begin{cases} S_L \cdot R_1(x) \; \dfrac{f_{\text{GE}}(x; \alpha, 1)}{F_{\text{GE}}(s; \alpha, 1)} & \text{(for } 0 \leq \text{x} \leq \text{s)} \\ S_R \cdot R_2(x) \; e^{-(x-s)} & \text{(for x > s)} \end{cases} \tag{26}$$

Here, S_L and S_R are the densities of the left and right parts,

$$S_L \equiv \frac{(1 - e^{-s})^\alpha}{\Gamma(\alpha+1)}, \quad S_R \equiv \frac{\alpha e^{-s} s^{\alpha-1}}{\Gamma(\alpha+1)}, \tag{27}$$

and $R_2(x)$ is the second rejection function

$$R_2(x) \equiv \left(\frac{x}{s}\right)^{\alpha-1} \tag{28}$$

which satisfies $0 < R_2(x) < 1$ for $x > s$. Equation (26) tells us that we can apply the rejection method to the GE distribution with $R_1(x)$ in the left part, and that we can apply the rejection method to the exponential density with $R_2(x)$ in the right part.

Using a uniform variate U_1, when $U_1 \leq S_L/(S_L + S_R)$, we proceed to the GE part. We generate the GE-distributed number in the range $[0, s]$. Using a uniform variate U_2, the GE variate x can be drawn by

$$x \leftarrow -\log\left(1 - [F_{\text{GE}}(s; \alpha, 1) \; U_2]^{1/\alpha}\right) \tag{29}$$

Then we evaluate the acceptance condition, as discussed in Section 3.

In the right part, we obtain an exponential variate from U_2,

$$x \leftarrow s - \log U_2 \tag{30}$$

and then we apply the rejection method. We can similarly apply the squeeze technique

to (28). For $x \geq 1$, from (16) and (17), we obtain a squeeze relation

$$\frac{1}{1 + \beta(x-1)} \leq x^{-\beta} \leq \frac{(1+\beta) + (1-\beta)x}{(1-\beta) + (1+\beta)x} \tag{31}$$

Finally, we choose s. First, we look for s that minimizes the total density

$$S(\alpha, s) = S_L + S_R = \frac{(1 - e^{-s})^\alpha + \alpha s^{\alpha-1} e^{-s}}{\Gamma(\alpha+1)}. \tag{32}$$

The solution can be found by solving $(1 - e^{-s})^{\alpha-1} + (\alpha - 1)s^{\alpha-2} - s^{\alpha-1} = 0$ by a root finder. In this case, we find an approximation by trial and error,

$$s^* = 1.28 + 0.23\alpha \tag{33}$$

This gives a near-minimum $S(\alpha, s^*)$ within an error of $< 1.5 \times 10^{-6}$. Another choice is $s = 1$, in analogy with Ahrens & Dieter (1974). This simplifies several parameters in the algorithm.

These procedures are shown in Algorithm 3 (Table 3). The first three lines initialize coefficients, and then the lines inside the loop generate the variate. We reuse $(S_L + S_R)/S_L \cdot U_1$ or $(S_L + S_R)/S_R \cdot (U_1 - S_L/(S_L + S_R))$ as a uniform variate, in order to reduce the total number of random variates. When $s = 1$, we can further simplify the code, which is a trivial task.

Table 3: A method with the piecewise envelope functions.

Algorithm 3
$s \leftarrow 1.28 + 0.23\alpha, \ t \leftarrow \exp(-s) \quad // \ s = 1, \ t = 1/e$
$S_L \leftarrow (1 - t)^\alpha, \ S_R \leftarrow \alpha \, t \, s^{\alpha-1}, \ S \leftarrow S_L + S_R$
$p_1 \leftarrow S_L/S, \ d_2 \leftarrow S/S_R$
repeat
generate $U_1, U_2 \sim U(0, 1)$
if $U_1 \leq p_1$ **then**
$b \leftarrow (SU_1)^{1/\alpha}, \ x \leftarrow -\log(1-b)$
if $U_2(4 + (1-\alpha)x) \leq (4 + (\alpha-1)x) \quad$ **return** x
if $U_2(4 + (2-\alpha)x) \leq (4 + \alpha x)$ **then**
if $U_2^{1/(1-\alpha)} x \leq b \quad$ **return** x
else
$x \leftarrow s - \log d_2(U_1 - p_1), \ y \leftarrow x/s$
if $U_2(\alpha + (\alpha-1)y) \leq 1 \quad$ **return** x
if $U_2(\alpha + (2-\alpha)y) \leq (2 - \alpha + \alpha y)$ **then**
if $U_2 \leq y^{\alpha-1} \quad$ **return** x
endif
end repeat

5 Numerical tests

We have carried out several benchmark tests. We wrote C codes for the eight methods: 1) the Ahrens & Dieter (1974) method, 2) the Best (1983) method, 3) the Devroye

(1986) method, 4) the Kundu & Gupta (2007) method (Algorithm 3 in their paper), 5) Algorithm 1 (Table 1), 6) Algorithm 2 (Table 2), 7) Algorithm 3 (Table 3) with $s = s^*$, and 8) Algorithm 3 with $s = 1$. In the last case, we have simplified the algorithm, as mentioned in Sec. 4. We use a uniform random generator (`gsl_rng_uniform`) in the GNU Scientific Library. We use Intel oneAPI Compiler (`icx`) v2024.1 with the `-lgsl -O0 -lm` option and the clang compiler (`clang`) v14.0 with the `-lgsl -lm` option on AMD Ryzen 5955 processor on Ubuntu Linux 24.04.
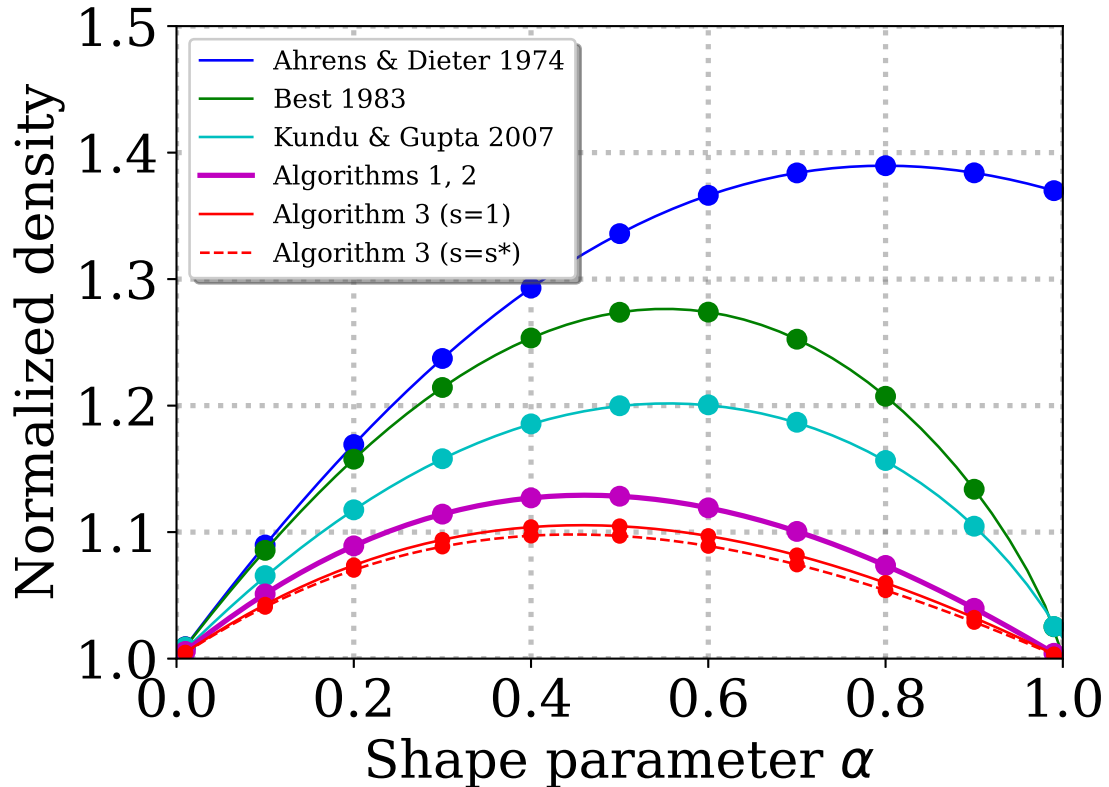

Figure 1: Normalized number densities to generate a gamma distribution. Theoretical predictions (curves) and numerical results (circles) are shown, as a function of α.

Figure 1 shows the normalized number of random variates to generate a gamma distribution, as a function of α. The curves indicate theoretical predictions. Algorithms 1 & 2 require $1/\Gamma(\alpha+1)$, the same number as the most efficient method to date, the Devroye (1986) method (not shown). Algorithm 3's number is given by (32). The circles indicate our Monte Carlo results for eleven shape parameters, $\alpha = 0.01, 0.1, 0.2, 0.3, \cdots 0.9, 0.99$. In each case, 10^8 random numbers are generated. They are in excellent agreement with the theories. The proposed methods, Algorithm 3 in particular, need fewer random numbers than most of the other algorithms. The $s = s^*$ case is better than the $s = 1$ case by $\lesssim 0.8\%$.

Next we compare the speed of the algorithms. Considering practical applications, two settings are considered (Tanizaki, 2008). In the first setting, we repeatedly draw one random variate 10^8 times for all 11 parameters. We measure the elapsed time in seconds three times per method, and then take the average elapsed time. Figure 2(a)
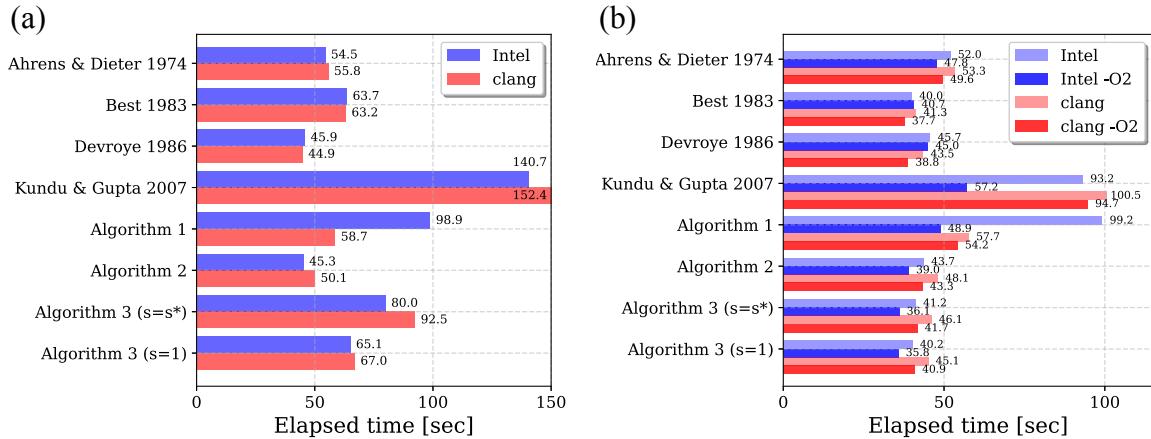
Figure 2: (a) Elapsed time to generate 10^8 gamma variates, including the parameter initialization. (b) Elapsed time to generate 10^8 gamma variates all at once. See the main text for detail.

shows the results. The Devroye (1986) method and Algorithm 2 are the fastest. They outperform the well established methods (Ahrens & Dieter, 1974; Best, 1983). Although the two compilers give different results, from Algorithms 1 and 2, we see that the squeeze technique is effective. Note that the Kundu & Gupta (2007) method does not use the squeeze technique.

In the second setting, we draw 10^8 random variates all at once, and then repeat this procedure for the 11 parameters. Some algorithms initialize internal parameters, before generating many random variates. For example, in Algorithm 3, the first three lines in Table 3 are used only once per parameter, before generating many variates at once. These lines are repeated each time we draw a variate in the first setting. Figure 2(b) shows the elapsed times. For reference, results with high-level compiler optimizations (the `-O2` option) are also displayed. Now the piecewise algorithms such as the Best (1983) method and Algorithm 3 become faster than in Fig. 2(a), suggesting that they are fast, once the parameters are initialized. Together with the Best (1983) method and the Devroye (1986) method, Algorithms 2–3 are close and promising. In Algorithm 3, the $s = 1$ cases are always faster than the $s = s^*$ cases.

6   Discussion and Summary

We have developed the rejection algorithms to generate a gamma variate with the shape parameter $0 < \alpha < 1$. In line of Kundu & Gupta (2007), we have employed the GE distribution (Gupta & Kundu, 1999). The GE distribution is useful, because it resembles the gamma distribution, and because its CDF has a simple form. Kundu & Gupta (2007) employed the GE distribution with the scale parameter 2, that is, $f_{GE}(x; \alpha, 2)$, while we employ the scale parameter 1, $f_{GE}(x; \alpha, 1)$, to obtain better acceptance efficiency. Therefore, the parameters and the rejection function in their study are different from ours. Despite a different strategy, our acceptance efficiency is as good as the best one (Devroye, 1986).

To improve the performance, we have constructed the squeeze functions in (13). As demonstrated, this makes the code faster. One may notice that the left-hand side (the

lower bound) can be negative for $x > 4/\beta$, but logically there is no problem. In addition, since $F_{\mathrm{GE}}(4/\beta; \alpha, 1) \geq F_{\mathrm{GE}}(4; \alpha, 1) = (1 - e^{-4})^\alpha \gtrsim 0.982$, more than 98% of the random numbers are found for $x < 4/\beta$, where the lower bound condition is useful.

We have considered piecewise extensions of our algorithms, with an exponential tail (Ahrens & Dieter, 1974; Best, 1983; Kundu & Gupta, 2007). This makes the acceptance efficiency even better. The squeeze functions are presented in (31). The left inequality (the lower bound) is identical to one in the Best (1983) method, while the right one (the upper bound) is a new addition. This addition actually makes the Best (1983) method few percents faster (not shown). As for Algorithm 3, despite the near-optimum density $S(\alpha, s^*) < S(\alpha, 1)$, the $s = 1$ case is always faster than the $s = s^*$ case, because $s = 1$ simplifies the algorithm. Therefore $s = 1$ is our choice. In the second setting (Fig. 2(b)), Algorithm 3 with $s = 1$ is always faster than Algorithm 2, owing to its good acceptance rate. It is promising when drawing many variates at once. In the first setting (Fig. 2(a)), in agreement with Tanizaki (2008), several methods become slower due to the parameter initialization. This is also the case for Algorithm 3. In contrast, Algorithm 2 remains fast in the first and second settings, because of its simplicity.

Considering several aspects, we conclude that Algorithm 2 is one of the best two generators, together with the Devroye (1986) method. These two often outperform the others, including the well established methods (Ahrens & Dieter, 1974; Best, 1983). With Intel compiler, Algorithm 2 is the best, and the Devroye (1986) method is the best with clang. The proposed methods are simple enough, and easy to implement, as shown in Tables 1–3. In addition to the compiler-dependence, there can be a CPU-dependence (and possibly a GPU-dependence). Considering various runtime environments, we want to have as many good algorithms as possible, and ours are certainly one of them.

In summary, we have developed gamma variate generators for $0 < \alpha < 1$, using the GE distribution, the squeeze technique, and the piecewise envelope functions. The proposed methods are excellent in acceptance efficiency. The numerical tests suggest that Algorithm 2 is one of the best two generators for $0 < \alpha < 1$. Algorithm 3 with $s = 1$ can be a good option when drawing many variates at once. We hope that the proposed methods are useful in practical applications.

Data Availability

Program codes in C language are available from the author upon request.

References

Ahrens, J. H. and Dieter, U. (1974), "Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions," *Computing* **12**, 223–246

Baumgarten C, and Patel, T. (2022), "Automatic random variate generation in Python," *Proc. of the 21st Python in Science Conf. (SCIPY 2022)*, 46–51

Best, D. J. (1983), "A Note on Gamma Variate Generators with Shape Parameter less than Unity," *Computing* **30**, 185–188

Devroye, L. (1986), *Non-Uniform Random Variate Generation*, Springer-Verlag, Chap. VII, Sec. 2.6, p. 304

Gupta, R. D. and Kundu, D. (1999), "Generalized exponential distributions," *Australian and New Zealand Journal of Statistics* **41**, 173–188

Kroese, D. P., Taimre, T., and Botev, Z. I. (2011), *Handbook of Monte Carlo methods*, John Wiley & Sons.

Kundu, D. and Gupta, R. D. (2007), "A convenient way of generating gamma random variables using generalized exponential distribution," *Computing Statistics & Data Analysis* **51**, 2796–2802

Marsaglia, G. and Tsang, W. W. (2000), "A Simple Method for Generating Gamma Variables," *ACM Transactions on Mathematical Software* **26**, 363–372

Tanizaki, H. (2008), "A Simple Gamma Random Number Generator for Arbitrary Shape Parameters," *Economics Bulletin* **3**, 1–10

Yotsuji, T. (2010), *Random number generation of probability distributions for computer simulations,* Pleiades Publishing, Nagano