$\alpha < 1$ のガンマ乱数生成法のベンチマーク結果

銭谷誠司 (Seiji Zenitani)[1]
オーストリア科学アカデミー 宇宙科学研究所, *8042 Graz, AUSTRIA*[a)]

(Dated: 5 June 2026)

本稿は Zenitani[10] のベンチマーク結果を補足する記事である。

Keywords: ガンマ乱数, モンテカルロ法, 棄却法

[a)]Electronic mail: seiji.zenitani@oeaw.ac.at

ガンマ分布は以下のように定義される。

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

α (>0) は形状パラメーター、λ (>0) はスケールパラメーター、$\Gamma(x)$ はガンマ関数である。ガンマ分布に従って分布する乱数、すなわちガンマ乱数を生成するアルゴリズムはこれまでさまざまなものが提案されている。詳しくは Luengo[6], 谷崎 [9] などのレビューを参照されたい。我々も最近、$\alpha < 1$ のガンマ乱数を生成するアルゴリズムを提案したところである。[10]

本稿では、Zenitani[10] で議論したランマ乱数のベンチマークを再検討する。Zenitani[10] では、以下の 8 つの数値解法の C 言語コードの実行時間を比較した。1) Ahrens & Dieter[1] 法, 2) Best[2] 法, 3) Devroye[3] 法, 4) Kundu & Gupta[5] 法, 5) Zenitani[10] 法のアルゴリズム 1, 6) アルゴリズム 2, 7) アルゴリズム 3 $(s = s^*)$, そして 8) アルゴリズム 3 $(s = 1)$ の 8 つである。一様乱数生成ルーチンには GNU Scientific Library の `gsl_rng_uniform` 関数、コンパイラには Intel oneAPI コンパイラ (`icx`) v2024.1 および LLVM コンパイラ (`clang`) v14.0 を使った。実行環境は Ubuntu Linux 24.04 LTS, CPU は AMD Ryzen 5955 processor であった。本稿では、それぞれのコンパイラのバージョンを Intel oneAPI コンパイラ v2026.0 および LLVM v18.1 にアップデートして同じベンチマークを行った。OS は Ubuntu Linux 24.04 LTS で、過去 2 年分のマイナーアップデートが反映されているが、メジャーバージョンに変更はない。

図 1 に、Zenitani[10] の 2024 年の結果と、今回の 2026 年の結果を示す。図 1 左は、パラメーターのセットアップを含めて 1 個の乱数を生成することを 10^8 回繰り返す “single draw”

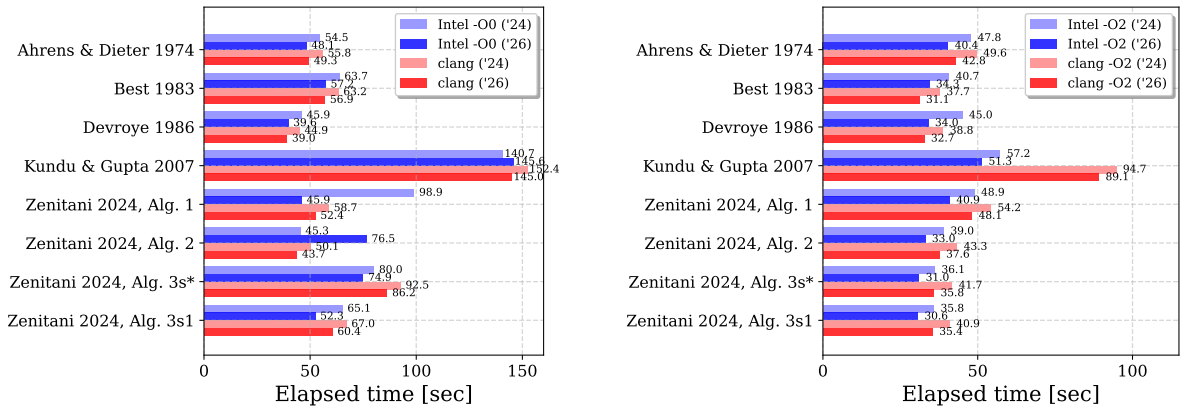

FIG. 1. ガンマ乱数生成法のベンチマーク結果。左）Single draw、右）Multi draw の場合で、′24 が Zenitani[10] の図 2(a), 2(b) の数値、′26 が今回の数値である。

ケースのベンチマーク、図 1 右は、最初にパラメーターのセットアップを行ってから、10^8 個の乱数を一気に生成する “multi draw” ケースのベンチマークである。

Single draw ケースでは、Intel コンパイラの最適化オプションに `-O0` を使っている。これは Intel コンパイラの `-O2` 最適化がアグレッシブで、前回のループと同じ計算結果を出す部分を検知した場合、その箇所の計算を飛ばしてしまうため、“single draw” のベンチマークが成立しないからである。全体として、概ね前回と同じような結果が出ているが、Intel コンパイラでも LLVM コンパイラでも一律、10–15%の高速化が実現している。コードおよびハードウェアは同一であるから、これは 2024 年から 2026 年までの間のソフトウェア（コンパイラおよび OS）の改良によって実現したものである。なお、single draw ケースの Zenitani[10] 法 アルゴリズム 2 の 2026 年の結果のみ、不自然に遅くなっている。この異常値の理由はわからない。アルゴリズム 1, 2 の結果を取り違えたわけではないことは確認済みである。この異常値を脇に置いて全体を眺めると、Devroye[3] 法、Zenitani[10] 法のアルゴリズム 2・アルゴリズム 3 ($s = 1$) がバランス良く優れているように見える。

Intel コンパイラはかなり高性能だが、前述のアグレッシブな最適化を抑制する方法がよくわからない。あとは、GNU C コンパイラ (`gcc`) など LLVM 以外の系統のコンパイラのベンチマークも行って、ベストなガンマ乱数生成法を議論すると良いだろう。その際、Marsaglia & Tsang[7] 法（以降、MT 法）の $\alpha < 1$ 向けのテクニックや、Tanizaki[8] 法などのベンチマーク結果も見てみたい。これからの時代を見据えると、これらの数値解法の GPU でのパフォーマンスも確認すべきであろう。例えば Ericsson[4] は、GPU 上では $\alpha > 1$ であっても MT 法より他の方法（具体的には Cheng GA 法）の方が速い、と報告している。しかし採択効率の良い MT 法は GPU 上ではなおさら有利なはずであり、この結果は我々の予想とは一致しない。さらに、内部で 32 スレッドを同時にコントロールする NVIDIA 製 GPU と 64 スレッドの AMD 製 GPU とでは特性が変わるはずである。こうした状況を踏まえると、各種ガンマ乱数生成法の CPU および GPU（NVIDIA/AMD）での性能を徹底検証する必要があるだろう。

REFERENCES

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

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

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

[4]Ericsson, J., *Gamma Random Variable Generation on GPUs using CUDA*, Master thesis, KTH Royal Institute of Technology (2024), `https://wertysas.github.io/documents/gamma_rng_on_gpus.pdf`

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

[6]Luengo, E. A., "Gamma Pseudo Random Number Generators," *ACM Comput. Surv.* **55**, 85 (2022).

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

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

[9]谷崎 久志、ガンマ乱数の生成方法について、国民経済雑誌, 197, 17–30 (2008), `https://doi.org/10.24546/00056219`.

[10]Zenitani, S., "A gamma variate generator with shape parameter less than unity," *Economics Bulletin* **44**, 1113–1122 (2024), `https://arxiv.org/abs/2411.01415`.