ガンマ乱数の生成方法について*
On Gamma Random Number Generators

谷﨑　久志 (神戸大学大学院経済学研究科)

2008 年 1 月

概要：　最近の計量経済学では，マルコフ連鎖モンテカルロ (Markov Chain Monte Carlo または MCMC) という方法を用いて，ベイズによる推定が実証分析に頻繁に取り入れられている。最近のベイズ推定というのは，事後分布から乱数を生成してパラメータの分析を行うという方法となっている。多くの場合，分散の事後分布は逆ガンマ分布になる。ガンマ分布に従う確率変数の逆数が逆ガンマ分布に従うので，ガンマ分布の乱数生成方法を考える必要がある。ガンマ分布はシェイプ・パラメータ α の大きさに依存して分布の形が変化する。$0 < \alpha < 1$ のとき，ガンマ分布は $x > 0$ について右下がりの曲線となる。$\alpha \geq 1$ のとき，ガンマ分布は $x = \alpha - 1$ でモードとなる。したがって，α が 1 より大きいか小さいかで乱数生成のアルゴリズムは大きく異なる。本稿では，いくつかの既存のガンマ乱数生成法を紹介し計算時間を比較する。

キー・ワード：　ガンマ分布，乱数生成，棄却法

1　はじめに

最近の計量経済学では，マルコフ連鎖モンテカルロ (Markov Chain Monte Carlo または MCMC) という方法を用いて，ベイズによる推定が実証分析に頻繁に取り入れられている。最近のベイズ推定というのは，事後分布から乱数を生成してパラメータの分析を行うという方法となっている。多くの場合，分散の事後分布が逆ガンマ分布になる。ガンマ分布に従う確率変数の逆数が逆ガンマ分布に従うので，ガンマ分布の乱数生成方法を考える必要がある。ガンマ分布は，通常，$f(x) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta}$ と表され，2 つのパラメータ (α, β) を含む分布である。確率変数 X が上記のようなパラメータ (α, β) を含むガンマ分布 $f(x)$ に従うことを $X \sim G(\alpha, \beta)$ と表記することにする。$X \sim G(\alpha, 1)$ で $Y = \beta X$ とするとき，$Y \sim G(\alpha, \beta)$ となる。$G(\alpha, 1)$ からの乱数生成方法を考えれば，$G(\alpha, \beta)$ からの乱数も簡単に生成することが出来る。したがっ

て，本稿では，次の $G(\alpha, 1)$ のガンマ分布

$$f(x) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}$$

からの乱数生成法を紹介する。

分布の関数形からガンマ分布はシェイプ・パラメータ α の大きさに依存して分布の形が変化する。$0 < \alpha < 1$ のとき，ガンマ分布は $x > 0$ について右下がりの曲線となる。$\alpha \geq 1$ のとき，ガンマ分布は $x = \alpha - 1$ でモードとなる。したがって，α が 1 より大きいか小さいかで乱数生成のアルゴリズムは大きく異なる。$0 < \alpha < 1$ のときは Ahrens and Dieter (1974) や Best (1983) が用いられることが多く，$\alpha \geq 1$ のときは Cheng (1977)，Marsaglia and Tsang (2001), Schmeiser and Lal (1980) が有名である。本稿では，これらの既存のガンマ乱数生成法を紹介し計算時間を比較する。

まず始めに，本稿で用いられる乱数生成方法の一つである棄却法 (rejection sampling) を紹介しておく。$f(x)$ からの乱数を生成することを考える。$g(x)$ は乱数を生成しやすい密度関数（サンプリング密度関数と呼ばれる）とし，しかも，

$$f(x) \leq cg(x)$$

となる $cg(x)$ を見つける。

$$\omega(x) = \frac{f(x)}{cg(x)}$$

とする。$g(x)$ から乱数 x を生成し，$\omega(x)$ の確率で x は $f(x)$ からの乱数となる。$1/c$ が x の採択確率となる (すなわち，c は $f(x)$ から一つの乱数を発生させるのに必要な $g(x)$ からの乱数の生成数を表す)。以上の乱数生成方法を棄却法と呼ぶ。密度関数 $f(x)$ からの乱数を生成する方法として，他にも一様乱数比法 (ratio-of-uniforms) という方法もあるが，本稿では説明は省略する。乱数に関する日本語文献では伏見 (1989) があげられる。

2　$0 < \alpha \leq 1$ のケース

2.1　Ahrens and Dieter (1974)

Ahrens and Dieter (1974) によると，$f(x) \leq cg(x)$ における $cg(x)$ について，

$$cg(x) = \begin{cases} x^{\alpha-1}/\Gamma(\alpha) & 0 < x < 1 \\ e^{-x}/\Gamma(\alpha) & 1 \leq x \end{cases}$$

を選んだ。このとき，

$$c = \int cg(x)\, \mathrm{d}x = \frac{e + \alpha}{e\alpha\Gamma(\alpha)}$$

となる。サンプリング密度関数は,

$$g(x) = \frac{e}{\alpha + e}\alpha x^{\alpha-1} I_{(0<x<1)} + \frac{\alpha}{\alpha + e} e^{-x+1} I_{(1\leq x<\infty)}$$

と表される。ただし，I_A は，A が真ならば 1，そうでなければ 0 とするインディケータ関数である。$e/(\alpha + e)$ の確率で，$0 < x < 1$ の範囲で密度関数 $\alpha x^{\alpha-1}$ から乱数を発生させ，$\alpha/(\alpha + e)$ の確率で，$1 \geq x$ の範囲で密度関数 e^{-x+1} から乱数を発生させればよい。

$g(x)$ からの乱数の採択確率は,

$$\omega(x) \equiv \frac{f(x)}{cg(x)} = \begin{cases} e^{-x} & 0 < x < 1 \\ x^{\alpha-1} & 1 \leq x \end{cases}$$

となる。このように $g(x)$ から生成した乱数を $\omega(x)$ の確率で $f(x)$ からの乱数とする。

まとめると，アルゴリズムは以下のようになる。

1. $c_1 = e/(\alpha + e)$ とする。
2. 区間 $(0, 1)$ の 2 つの一様乱数 u_1，u_2 を生成する。
3. $u_1 \leq c_1$ の場合：
   $x = (u_1/c_1)^{1/\alpha}$ とする。
   $u_2 \leq e^{-x}$ ならば x を採用して終わり，そうでなければステップ 2 に戻る。
4. $u_1 > c_1$ の場合：
   $x = -\log((1 - u_1)/(c_1\alpha))$ とする。
   $u_2 \leq x^{\alpha-1}$ ならば x を採用して終わり，そうでなければステップ 2 に戻る。

このアルゴリズムはかなり効率の良いアルゴリズムとされている。しかし，サンプリング密度関数が $x = 1$ で変わるという点を改善してより高速な乱数生成方法を提案したのが，次節で紹介する Best (1983) である。本稿を通して，一様乱数の生成方法は L'Ecuyer (1988) による方法を採用した (そのアルゴリズムについては補論を参照せよ)。

2.2 Best (1983)

Best (1983) は Ahrens and Dieter (1974) の x の範囲を修正して，$cg(x)$ を次のように選んだ。

$$cg(x) = \begin{cases} x^{\alpha-1}/\Gamma(\alpha) & 0 < x < t \\ t^{\alpha-1}e^{-x}/\Gamma(\alpha) & t \leq x \end{cases}$$

このとき,

$$c = \int cg(x)\,\mathrm{d}x = \frac{t^\alpha(1 + \alpha t^{-1}e^{-t})}{\alpha\Gamma(\alpha)}$$

となり，さらに，c が最小となるように t を選ぶ。すなわち，

$$1 - \alpha = t(e^t - 1)$$

を満たす t を選べばよい。しかし，t を α の関数として，明示的な形で表すことが出来ないので，近似的に，Best (1983) は，

$$t = 0.07 + 0.75\sqrt{1 - \alpha}$$

を用いた。サンプリング密度関数は，

$$g(x) = \frac{1}{1 + \alpha t^{-1}e^{-t}} \alpha t^{-\alpha} x^{\alpha-1} I_{(0<x<t)} + \frac{\alpha t^{-1}e^{-t}}{1 + \alpha t^{-1}e^{-t}} e^{-x+t} I_{(t \le x<\infty)}$$

となる。$1/(1 + \alpha t^{-1}e^{-t})$ の確率で，$0 < x < t$ の範囲で密度関数 $\alpha t^{-\alpha} x^{\alpha-1}$ から乱数を発生させ，$\alpha t^{-1}e^{-t}/(1 + \alpha t^{-1}e^{-t})$ の確率で，$t \ge x$ の範囲で密度関数 e^{-x+t} から乱数を発生させればよい。

$g(x)$ からの乱数の採択確率は，

$$\omega(x) \equiv \frac{f(x)}{cg(x)} = \begin{cases} e^{-x} & 0 < x < t \\ (x/t)^{\alpha-1} & t \le x \end{cases}$$

となる。このように $g(x)$ から生成した乱数を $\omega(x)$ の確率で $f(x)$ からの乱数とする。

まとめると，アルゴリズムは以下のようになる。

1. $c_1 = 0.07 + 0.75\sqrt{1 - \alpha}$，$c_2 = 1 + \alpha e^{-c_1}/c_1$，$c_3 = 1/\alpha$ とする。
2. 区間 $(0, 1)$ の 2 つの一様乱数 u_1，u_2 を生成し，$v = c_2 u_1$ とする。
3. $v \le 1$ の場合：
   $x = c_1 v^{c_3}$ とする。
   $u_2 \le (2 - x)/(2 + x)$ ならば x を採用して終わり。
   $u_2 \le e^{-x}$ ならば x を採用して終わり。
   上記の 2 条件とも成立しなければステップ 2 に戻る。
4. $v > 1$ の場合：
   $x = -\log(c_1 c_3(c_2 - v))$，$y = x/c_1$ とする。
   $u_2(\alpha + y - \alpha y) \le 1$ ならば x を採用して終わり。
   $u_2 \le y^{\alpha-1}$ ならば x を採用して終わり。
   上記の 2 条件とも成立しなければステップ 2 に戻る。

ステップ 3 と 4 では，対数計算や指数計算を避けて計算スピードの効率を上げるために，それぞれのステップの 3 行目に抉り出し法 (squeeze method) と呼ばれる方法が用いられている。$0 < \alpha < 1$ の場合，この乱数生成法は高速で，IMSL ライブラリに収められている。ただし，前節のアルゴリズムに比べてパラメータのセットアップ数が多くなるので，乱数を一つ生成するたびにパラメータをセットアップし直すような場合には，前節の方が高速なアルゴリズムとなる。

3　$\alpha > 1$ のケース

3.1　Cheng (1977)

ガンマ分布 $f(x)$ から乱数を生成するにあたり，Cheng (1977) は次のような対数ロジスティック分布 $g(x)$ を利用した。

$$g(x) = \lambda\mu\frac{x^{\lambda-1}}{(\mu + x^{\lambda})^2}$$

このとき累積分布関数は $\int_0^x g(t)\mathrm{d}t = x^{\lambda}/(\mu + x^{\lambda})$ となり，区間 $(0, 1)$ の一様乱数 u_1 を発生させると，対数ロジスティック分布からの乱数 x は次のように得られる。

$$x = \left(\frac{\mu u_1}{1 - u_1}\right)^{1/\lambda}$$

μ や λ は c を小さくするように求められ，Cheng (1977) は次のように選んだ。

$$\mu = \alpha^{\lambda} \qquad \lambda = (2\alpha - 1)^{1/2}$$

したがって，c は,

$$c = \frac{4\alpha^{\alpha}e^{-\alpha}}{\Gamma(\alpha)\lambda} = \frac{4\alpha^{\alpha}e^{-\alpha}}{\Gamma(\alpha)(2\alpha - 1)^{1/2}}$$

となる。$1/c$ は $g(x)$ から生成された乱数の採択率を表す。$\alpha = 1,\ 2,\ 5,\ 10,\ \infty$ に対応して，$1/c = 0.68,\ 0.8,\ 0.85,\ 0.87,\ 0.88$ となる。

区間 $(0, 1)$ の一様乱数 u_2 を生成すると,

$$\log(u_1^2 u_2) \le (\alpha + \lambda)\log x - x - \log(c\lambda\mu\Gamma(\alpha))$$

のときに x を $f(x)$ からの乱数とみなす。c を代入して,

$$\log(u_1^2 u_2) \le (\alpha + \sqrt{2\alpha - 1})\log x - x - (\alpha + \sqrt{2\alpha - 1})\log(\alpha) + \alpha - \log(4)$$

のときに x を $f(x)$ からの乱数とみなす。

まとめると，アルゴリズムは以下のようになる。

1. $c_1 = 1/\sqrt{2\alpha - 1}$，$c_2 = \alpha - \log(4)$，$c_3 = \alpha + \sqrt{2\alpha - 1}$，$c_4 = 1 + \log(4.5)$ とする。
2. 区間 $(0, 1)$ の 2 つの一様乱数 u_1，u_2 を生成する。
3. $y_1 = c_1\log(u_1/(1 - u_1))$，$x = \alpha e^{y_1}$，$y_2 = u_1^2 u_2$，$y_3 = c_2 + c_3 y_1 - x$ とおく。
4. $y_3 \ge 4.5y_2 - c_4$ ならば x を採用して終わり。
   $y_3 \ge \log(y_2)$ ならば x を採用して終わり。
   上記の 2 条件とも成立しなければステップ 2 に戻る。

ステップ 4 の 1 行目に，対数計算を避けて計算スピードの効率を上げるために抉り出し法 (squeeze method) が用いられる。

3.2 Marsaglia and Tsang (2001)

まず，$h(y)$ を次のように定義する。

$$h(y) = d(1 + by)^3$$

ただし，$-1/b < y < \infty$，$d = \alpha - 1/3$，$b = 1/\sqrt{9d}$ とする。

確率変数 Y の密度関数が $h(y)^{\alpha-1}e^{-h(y)}h'(y)/\Gamma(\alpha)$ のとき，$X = h(Y)$ はガンマ分布 $f(x)$ に従う。よって，Y の乱数を生成して $X = h(Y)$ と変換してガンマ乱数を生成することを考える。関数 $h(y)$ に関わる部分を指数の形に変形すると，

$$h(y)^{\alpha-1}e^{-h(y)}h'(y) \propto e^{g(y)}$$

が得られる。ただし，

$$g(y) = d\log(1 + by)^3 - d(1 + by)^3 + d$$

とする。$g(0) = 0$ が成り立つように $g(y)$ が求められている。

このとき，次の関係が成り立つ。

$$e^{g(y)} \leq e^{-0.5y^2}$$

ただし，$-1/b < y < \infty$ とする。よって，サンプリング密度関数は $-1/b$ より大きい範囲で切断された標準正規分布とすることになる。

受容確率は

$$\omega(y) = \exp(g(y) + 0.5y^2)$$

となる。

$-1/b$ より大きい範囲で切断された標準正規分布から乱数 y を発生させ，$\omega(y)$ の確率で $e^{g(y)}$ に比例する密度関数からの乱数となる。

まとめると，アルゴリズムは以下のようになる。

1. $c_1 = \alpha - 1/3$，$c_2 = 1/\sqrt{9c_1}$ とする。
2. 標準正規乱数 z を生成する。
3. $c_2 z \leq -1$ ならばステップ 2 に戻る。
   そうでなければ $v = (1 + c_2 z)^3$ とする。
4. 区間 $(0, 1)$ の一様乱数 u を生成する。
5. $u < 1 - 0.0331z^4$ ならば $x = c_1 v$ を採用して終わり。
6. $\log(u) < 0.5z^2 + c_1(1 - v + \log(v))$ ならば $x = c_1 v$ を採用して終わり。
   そうでなければステップ 2 に戻る。

ステップ 5 では，対数計算を避けて計算効率を上げるために抉り出し法 (squeeze method) が用いられている。

このガンマ乱数の生成スピードは標準正規乱数の生成スピードに依存する。高速な標準正規乱数生成法を使えば効率は良いが，Box-Muller 法等の簡単ではあるがスピードの遅い正規乱数生成法を使えば計算に時間がかかることになる。次節のガンマ乱数の計算速度の比較では，Box-Muller 法を用いた場合と Hormann and Derflinger (1990) によって提案された場合とを比較する (Box-Muller 法と Hormann and Derflinger (1990) の方法については，補論を参照せよ)。

3.3 Schmeiser and Lal (1980)

Schmeiser and Lal (1980) によると，ガンマ密度関数 $f(x)$ を，$x_1, x_2, \cdots, x_5$ を用いて，6 つに分割する。それぞれの点は，$x_3 = \alpha - 1$，$x_2 = \max(0, x_3 - x_3^{1/2})$，$x_4 = x_3 + x_3^{1/2}$，$x_1 = x_2(1 - 1/(x_3 - x_2))$，$x_5 = x_4(1 - 1/(x_4 - x_3))$ とする。x_3 は分布のモードに対応し，x_2 と x_4 は $f(x)$ の変曲点を表し，x_1 と x_5 は x_2 と x_4 での $f(x)$ の接線の X 軸との交差する点である。棄却法を用いることが出来るように，$cg(x)$ を次のように設定する。

$$cg(x) \equiv t(x) = \begin{cases} s(x_2)\exp(-\lambda_L(x - x_2)) & 0 < x \leq x_2 \\ 1 & x_2 < x \leq x_4 \\ s(x_4)\exp(-\lambda_R(x - x_4)) & x_4 < x < \infty \end{cases}$$

すなわち，サンプリング密度関数を指数分布と一様分布から成る混合分布としている。$s(x)$ は，$s(x_3) = 1$ となるように $f(x)$ を変形したもので，

$$s(x) = \exp(x_3 \log(x/x_3) + x_3 - x)$$

と表される ($x_3 = \alpha - 1$ に注意)。

また，x_1〜x_5 の各点を結び直線 $b(x)$ を作り，抉り出しに用いる。すなわち，

$$b(x) = \begin{cases} 0 & 0 < x \leq x_1 \\ s(x_2)(x - x_1)/(x_2 - x_1) & x_1 < x \leq x_2 \\ s(x_2) + (1 - s(x_2))(x - x_2)/(x_3 - x_2) & x_2 < x \leq x_3 \\ s(x_4) + (1 - s(x_4))(x - x_4)/(x_4 - x_3) & x_3 < x \leq x_4 \\ s(x_4)(x_5 - x)/(x_5 - x_4) & x_4 < x \leq x_5 \\ 0 & x_5 < x < \infty \end{cases}$$

となる。棄却法と抉り出し法を組み合わせた場合，そのアルゴリズムは一般的に次のようになる。$g(x) = t(x)/\int t(y)\mathrm{d}y$ と区間 $(0, 1)$ の一様分布から乱数 (それぞれ，x と v とする) を発生させ，$v \leq b(x)/t(x)$ なら x を採用して終わり。$v \leq f(x)/t(x)$ なら x を採用して終わり，そうでなければ $g(x)$ から乱数を発生し直す。

まとめると，具体的にはアルゴリズムは以下の通りとなる。

1. $x_3 = \alpha - 1$，$d = \sqrt{x_3}$，$x_1 = 0$，$x_2 = 0$，$f_1 = 0$，$f_2 = 0$，$\lambda_L = 1$ とする。
   $d \geq x_3$ ならばステップ 2 へ。
   そうでなければ $x_2 = x_3 - d$，$x_1 = x_2(1 - 1/d)$，$\lambda_L = 1 - x_3/x_1$，$f_1 = \exp(x_3 \log(x_1/x_3) + x_3 - x_1)$，$f_2 = \exp(x_3 \log(x_2/x_3) + x_3 - x_2)$ とする。

2. $x_4 = x_3 + d$，$x_5 = x_4(1 + 1/d)$，$\lambda_R = 1 - x_3/x_5$，$f_4 = \exp(x_3 \log(x_4/x_3) + x_3 - x_4)$，$f_5 = \exp(x_3 \log(x_5/x_3) + x_3 - x_5)$，$p_1 = f_2(x_3 - x_2)$，$p_2 = f_4(x_4 - x_3) + p_1$，$p_3 = f_1(x_2 - x_1) + p_2$，$p_4 = f_5(x_5 - x_4) + p_3$，$p_5 = (1 - f_2)(x_3 - x_2) + p_4$，$p_6 = (1 - f_4)(x_4 - x_3) + p_5$，$p_7 = (f_2 - f_1)(x_2 - x_1)/2 + p_6$，$p_8 = (f_4 - f_5)(x_5 - x_4)/2 + p_7$，$p_9 = -f_1/\lambda_L + p_8$，$p_{10} = f_5/\lambda_R + p_9$ とする。

3. 区間 $(0, 1)$ の一様乱数 u を生成し，$u = up_{10}$ とする。
   $u > p_4$ ならばステップ 7 へ。
   $u > p_1$ ならばステップ 4 へ。
   そうでなければ $x = x_2 + u/f_2$ を採用して終わり。

4. $u > p_2$ ならばステップ 5 へ。
   そうでなければ $x = x_3 + (u - p_1)/f_4$ を採用して終わり。

5. $u > p_3$ ならばステップ 6 へ。
   そうでなければ $x = x_1 + (u - p_2)/f_1$ を採用して終わり。

6. $x = x4 + (u - p3)/f5$ を採用して終わり。

7. 区間 $(0, 1)$ の一様乱数 w を生成する。
   $u > p_5$ ならばステップ 8 へ。
   $x = x_2 + (x_3 - x_2)w$ とする。
   $(u - p_4)/(p_5 - p_4) \leq w$ ならば x を採用して終わり。
   そうでなければ $v = f_2 + (u - p_4)/(x_3 - x_2)$ とし，ステップ 13 へ。

8. $u > p_6$ ならばステップ 9 へ。
   $x = x_3 + (x_4 - x_3)w$ とする。$(p_6 - u)/(p_6 - p_5) \geq w$ ならば x を採用して終わり。
   そうでなければ $v = f_4 + (u - p_5)/(x_4 - x_3)$ とし，ステップ 13 へ。

9. $u > p_8$ ならばステップ 11 へ。
   そうでなければ区間 $(0, 1)$ の一様乱数 w_2 を生成する。
   $w_2 > w$ ならば $w = w2$ とする。
   $u > p_7$ ならばステップ 10 へ。
   $x = x_1 + (x_2 - x_1)w$，$v = f_1 + 2w(u - p_6)/(x_2 - x_1)$ とする。
   $v \leq f_2 w$ ならば x を採用して終わり。
   そうでなければステップ 13 へ。

10. $x = x_5 - w(x_5 - x_4)$，$v = f_5 + 2w(u - p_7)/(x_5 - x_4)$ とする。
    $v \le f_4 w$ ならば x を採用して終わり。
    そうでなければステップ 13 へ。

11. $u > p_9$ ならばステップ 12 へ。
    $u = (p_9 - u)/(p_9 - p_8)$，$x = x_1 - \log(u)/\lambda_L$ とする。$x \le 0$ ならばステップ 3 に戻る。
    $w < (\lambda_L(x_1 - x) + 1)/u$ ならば x を採用して終わり。
    そうでなければ $v = w f_1 u$ とし，ステップ 13 へ。

12. $u = (p_{10} - u)/(p_{10} - p_9)$，$x = x_5 - \log(u)/\lambda_R$ とする。
    $w < (\lambda_R(x_5 - x) + 1)/u$ ならば x を採用して終わり。
    そうでなければ $v = w f_5 u$ とする。

13. $\log(v) \le x_3 \log(x/x_3) + x_3 - x$ ならば x を採用して終わり。
    そうでなければステップ 3 に戻る。

この乱数生成方法は，ステップ 3〜13 の繰り返しで同時に多くのガンマ乱数を生成するのには非常にスピードが速いが ($\alpha \ge 1$ の場合，この乱数生成法は IMSL ライブラリに収められている)，乱数を一つ生成する度毎にパラメータのセットをやり直す方法 (すなわち，ステップ 1〜13 の繰り返し) で多くの乱数を生成させるには非常にスピードは遅くなる。さらに，ステップ数が 13 ステップとなっていて非常に多く，プログラムが煩雑になることも欠点の一つとなる。

4 計算比較

表 1 において，AD74 は Ahrens and Dieter (1974) の方法 (2.1 節)，B83 は Best (1983) の方法 (2.2 節)，C77 は Cheng (1977) の方法 (3.1 節)，MT01 は Marsaglia and Tsang (2001) の方法 (3.2 節)，SL80 は Schmeiser and Lal (1980) の方法 (3.3 節) をそれぞれ意味する。表中の数字は，10^8 個の乱数を生成するのにかかった時間 (単位は秒) を表す。また，MT01 と MT01* との違いは，標準正規乱数の生成方法の違いによる。MT01 は Box-Muller 法，MT01* は Hormann and Derflinger (1990) によって正規乱数を生成した。計算に用いたパソコンは Opteron 290 (2.8GHz) の Dual CPU，オペレーティング・システムは Windows XP Professional SP2，コンパイラは Open Watcom F77/32 Compiler Ver.1.7a (http://www.openwatcom.org) を用いた。

表中の「パラメータのセットを除く」とは 10^8 個の乱数をベクトルとして生成することであり,「パラメータのセットを含む」とは一つの乱数を生成するたびごとにパラメータをセットし直すことを意味する。それぞれのアルゴリズムの中でパラメータのセットに当たる部分とは，AD74，B83，C77，MT01 ではステップ 1 であり，SL80 ではステップ 1 と 2 である。α 一定のもとで 10^8 個の乱数を生成する場合は「パラメータのセットを除く」を参考にすべきであるが，α の値が乱数を一つ生成するたびごとに変わる場合は「パラメータのセットを

表 1: CPU スピード (単位は秒)

α	パラメータのセットを除く		パラメータのセットを含む	
	AD74 (2.1 節)	B83 (2.2 節)	AD74 (2.1 節)	B83 (2.2 節)
0.01	29.3	20.1	30.1	33.1
0.1	31.4	22.0	32.2	35.9
0.2	34.1	22.0	34.3	38.0
0.3	41.0	30.5	41.7	44.4
0.4	42.9	31.6	43.4	45.6
0.6	44.8	31.8	45.6	45.8
0.8	45.6	29.3	46.1	42.8
0.99	43.7	21.8	45.2	34.5

α	パラメータのセットを除く				パラメータのセットを含む			
	C77 (3.1 節)	MT01 (3.2 節)	MT01* (3.2 節)	SL80 (3.3 節)	C77 (3.1 節)	MT01 (3.2 節)	MT01* (3.2 節)	SL80 (3.3 節)
1.01	36.9	30.7	22.0	22.0	58.1	32.6	23.1	52.3
1.4	33.2	30.2	21.7	21.1	53.4	32.1	22.9	52.4
1.8	31.5	30.0	21.5	21.9	51.3	31.9	22.8	53.9
2.2	30.6	29.8	21.4	21.5	50.2	31.8	22.6	72.1
2.6	30.1	30.5	21.3	20.8	49.5	31.6	22.3	71.0
3	29.7	29.7	21.3	20.4	49.0	31.7	22.3	70.8
4	29.1	29.6	21.2	20.1	48.3	31.5	22.2	72.2
5	28.8	29.5	21.2	19.8	47.9	31.7	22.2	71.7
10	28.1	29.4	21.1	19.6	47.1	31.4	22.3	72.2
20	27.8	29.4	21.0	19.6	46.8	31.3	22.0	58.7
50	27.6	29.4	21.0	19.5	46.6	31.4	22.1	59.7
100	27.6	29.3	21.0	19.5	46.5	31.4	22.1	59.8
400	27.5	29.3	21.0	19.5	46.5	31.4	22.0	59.8

1) 表中の数字は 10^8 個の乱数を発生させるのにかかった時間 (秒) を表す。
2) MT01 と MT01* との違いは，標準正規乱数の生成方法の違いによる。MT01 は Box-Muller 法，MT01* は Hormann and Derflinger (1990) によって正規乱数を生成した。

含む」を見るべきである。後者を用いる場合というのは，最近のベイズ推定で見られるように Gibbs サンプラーを使って分散の乱数を生成する場合に相当する。

結果は次のように要約される。

(1) $0 < \alpha < 1$ で「パラメータのセットを除く」場合，B83 が AD74 よりも 1.3〜2.0 倍程度乱数生成速度が速い。しかし,「パラメータのセットを含む」場合，α が 0.6 までは AD74 が B83 よりも少し速いが，α が 0.8 や 0.99 のときは B83 が速い。

(2) $0 < \alpha < 1$ で,「パラメータのセットを除く」と「パラメータのセットを含む」の差は，AD74 では 0.2〜1.5 秒に対して，B83 では 12.7〜16 秒となっている。B83 のパラメータ・セットにかかる時間は，AD74 のそれよりも圧倒的に長いと言える。

(3) $\alpha \geq 1$ で「パラメータのセットを除く」場合，α が 2.2 より大きいとき，SL80 が一番効率的である。しかし，α が 2.2 より小さいとき，MT01* と SL80 とはほとんど同じくらいのスピードである。

(4) $\alpha \geq 1$ で「パラメータのセットを含む」場合，MT01* が 4 つの中で最速となる。SL80 がもっとも遅いアルゴリズムとなっている。これは「パラメータのセットを除く」場合とは対照的な結果となっている。パラメータ・セットにかかる時間は，C77 で 18.9〜21.2 秒，MT01 では 1.1〜2.2 秒，MT01* では 1〜1.3 秒，SL80 では 30.3〜52.1 秒となっている。SL80 が極端に時間がかかり，MT01 や MT01* はほとんど余分な時間はかからないという結果となっている。

(5) Marsaglia and Tsang (2001) の方法は標準正規乱数の生成方法のアルゴリズムに非常に依存する。MT01 と MT01* を比較すれば明らかである。

アルゴリズムの簡単さとスピードを考えた場合，MT01* が一番推薦できる。また，MT01* における標準正規乱数のアルゴリズムがより高速になれば，SL80 より速くなることもあり得る。一般的に，高速なアルゴリズムは，アルゴリズムのステップ数が多いという傾向がある。例えば，Box-Muller 法と Hormann and Derflinger (1990) とを比較すると，前者は後者よりもプログラムの行数は短いが表 1 を見ると正規乱数生成の時間がかなり長いことが分かる (MT01 と MT01* との違いは正規乱数の生成方法だけである)。

5 おわりに

本稿ではガンマ乱数の生成に関する代表的なアルゴリズムをいくつか紹介した。$0 < \alpha < 1$ について,「パラメータのセットを除く」場合は B83 が AD74 よりもずっと早いが,「パラメータのセットを含む」場合は α の値に依存して一概にどちらが速いとは言えない結果を得た。また，$\alpha \geq 1$ について,「パラメータのセットを除く」場合は SL80 が最速になるが，MT01* とあまり変わらない結果となった。むしろ，標準正規乱数のより高速な生成方法を採用すれ

ば，MT01* が SL80 よりも高速になる可能性は十分にあると言える。しかし,「パラメータのセットを含む」場合，SL80 は最も非効率的なガンマ乱数生成方法となり，パラメータのセット数が少ない MT01* が最も高速なガンマ乱数生成方法となった。

ベイズ推定における MCMC では，一つの乱数を生成するたびごとに，α の値が変化するので,「パラメータのセットを含む」で最速なガンマ乱数の生成方法を採用すべきである。したがって，より効率的な標準正規乱数生成方法を用いた Marsaglia and Tsang (2001) の方法が，$\alpha \geq 1$ の場合，最も良いと言える。

補論： 区間 (0, 1) の一様乱数と標準正規乱数の生成について

一様乱数： L'Ecuyer (1988)

本稿で用いた一様乱数生成方法は L'Ecuyer (1988) よって提案されたものを使う。これは簡単であるが周期が長い擬似乱数を生成出来る方法である。

準備として，次の例を k，m，n を正の整数とする。$[k/n]$ を k を n で割った整数部分と定義する。このとき，k を n で割った余りは,

$$m = k - [k/n]n$$

と表され，同じ意味で,

$$m = k \bmod n$$

と定義する。

L'Ecuyer (1988) は，それぞれの $j = 1, 2, \cdots, k$ について，次のような漸化式で $x_{j,i}$，$i = 1, 2, \cdots$，を求めることを考えた。

$$x_{j,i} = \left(a_j (x_{j,i-1} \bmod q_j) - [x_{j,i-1}/q_j] r_j\right) \bmod n_j$$

ただし，(a_j, n_j) は正の整数であり，$q_j = [n_j/a_j]$，$r_j \equiv n_j \bmod a_j$ とする。この方法は線形合同法と呼ばれる。さらに，$x_{j,i}$，$j = 1, 2, \cdots, k$，から x_i を次のように求める。

$$x_i = \sum_{j=1}^{k} (-1)^{j-1} x_{j,i} \bmod (n_1 - 1)$$

i 番目の区間 $(0, 1)$ の一様乱数 u_i は次のように得られる。

$$u_i = \begin{cases} x_i/n_1 & \text{if } x_i > 0 \\ (n_1 - 1)/n_1 & \text{if } x_i = 0 \end{cases}$$

実際には，L'Ecuyer (1988) は $k = 2$ として，$(a_1, n_1) = (40014, 2147483563)$ と $(a_2, n_2) = (40692, 2147483399)$ を選んだ。このようにして得られた一様乱数の周期は 2.31×10^{18} となり，非常

に長いものとなっている。この周期をより長くするためには，Bays and Durham (1976) により提案されたシャッフリング (shuffling) 法を組み合わせることが考えられる。この方法は Press, Teukolsky, Vetterling and Flannery (1992a, 1992b) にソース・コードが紹介されている。しかし，本稿では単に上記に紹介した L'Ecuyer (1988) の方法を用いた。

標準正規乱数： Box-Muller

標準正規分布 $f(x) = (2\pi)^{-1/2} \exp(-\frac{1}{2}x^2)$ からの乱数は次のようにして得られる。u_1，u_2 を独立な区間 $(0, 1)$ の一様乱数とする。以下のような変換を考える。

$$x_1 = \sin(2\pi u_1) \sqrt{-2 \log(u_2)}$$
$$x_2 = \cos(2\pi u_1) \sqrt{-2 \log(u_2)}$$

このとき，x_1，x_2 は独立な標準正規分布に従う乱数となる。この乱数生成方法を Box-Muller 法と呼ばれる。Box-Muller 法は非常に簡単な標準正規乱数の生成方法であるが，スピード面ではそれほど効率的ではない。より高速に生成できる方法の一つとして Hormann and Derflinger (1990) による方法がある。

標準正規乱数： Hormann and Derflinger (1990)

Box-Muller 法よりより高速な標準正規分布 $f(x) = (2\pi)^{-1/2} \exp(-\frac{1}{2}x^2)$ からの乱数生成法は Hormann and Derflinger (1990) によって提案された。$f(x)$ を 3 つの区間 $|x| \le 1$，$1 < |x| \le 2$，$|x| > 2$ に分割する。そのアルゴリズムの概略は以下のように示される。

アルゴリズム A:

1. x を区間 $(-1, 1)$ の一様乱数，y を区間 $(0, 1/2)$ の一様乱数とする。
2. $f(x) \ge y$ ならば x を採用して終わり。
3. $f(\text{sign}(x)2 - x) \ge 1/2 - y$ ならば $\text{sign}(x)2 - x$ を採用して終わり。
4. その他の場合は $|x| > 2$ についてアルゴリズム B を使う。

アルゴリズム B:

1. u を $u \ne 0$ を除く区間 $(-1/e, 1/e)$ の一様乱数，v を区間 $(0, 0.11328)$ の一様乱数とする。
2. $v \le 2|u|(-1 + (-\log|u|)^{1/2})$ ならば $\text{sign}(u)2 + v/u$ を採用して終わり。
3. そうでなければステップ 1 に戻る。

アルゴリズム B では，一様乱数比法 (ratio-of-uniform method) と呼ばれる方法が用いられている。概略は以上であるが，効率よく乱数を生成できるように様々な工夫がなされている。詳細なアルゴリズムは Hormann and Derflinger (1990) に Fortran のソース・コードが掲載されている。

* 本研究は科学研究費 (C) 18530158 (2006 年～2009 年) と 21 世紀 COE プログラムからの助成を受けた。

参考文献

Ahrens, J.H. and Dieter, U., 1974, “Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions,” *Computing*, Vol.12, pp.223 – 246.

Bays, C. and Durham, S.D., 1976, “Improving a Poor Random Number Generator,” *ACM Transactions on Mathematical Software*, Vol.2, pp.59 – 64.

Best, D.J., 1983, “A Note on Gamma Variate Generators with Shape Parameter Less Than Unity,” *Computing*, Vol.30, pp.185 – 188.

Cheng, R.C.H., 1977, “The Generation of Gamma Variables with Non-Integral Shape Parameter,” *Applied Statistics*, Vol.26, No.1, pp.71 – 75.

Hormann, W. and Derflinger, G., 1990, “The ACR method for generating normal random variables,” *OR Spektrum*, Vol.12, pp.181 – 185.

L’Ecuyer, P., 1988, “Efficient and Portable Combined Random Number Generators,” *Communications of the ACM*, Vol.31, No.6, pp.742 – 749.

Marsaglia, G. and Tsang, W.W., 2001, “A Simple Method for Generating Gamma Variables,” *ACM Transactions on Mathematical Software*, Vol.26, No.3, pp.363 – 372.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., 1992a, *Numerical Recipes in C: The Art of Scientific Computing* (Second Edition), Cambridge University Press.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., 1992b, *Numerical Recipes in Fortran: The Art of Scientific Computing* (Second Edition), Cambridge University Press.

Schmeiser, B. and Lal, R., 1980, “Squeeze Methods for Generating Gamma Variates,” *Journal of the American Statistical Association*, Vol.75, pp.679 – 682.

伏見正則，1989,『乱数』東京大学出版会