5.7.4 Ratio-of-Uniforms Method

As an alternative random number generation method, in this section we introduce the **ratio-of-uniforms method**.

This generation method does not require the sampling density utilized in rejection sampling (Section 5.7.1), importance resampling (Section 5.7.2) and the Metropolis-Hastings algorithm (Section 5.7.3).

Suppose that a bivariate random variable (U_1, U_2) is uniformly distributed, which satisfies the following inequality:

$$0 \leq U_1 \leq \sqrt{h(U_2/U_1)},$$

for any nonnegative function $h(x)$. Then, $X = U_2/U_1$ has a density function $f(x) = h(x)/\int h(x)\,\mathrm{d}x$.

Note that the domain of (U_1, U_2) will be discussed below.

The above random number generation method is justified in the following way.

The joint density of U_1 and U_2, denoted by $f_{12}(u_1, u_2)$, is given by:

$$f_{12}(u_1, u_2) = \begin{cases} k, & \text{if } 0 \le u_1 \le \sqrt{h(u_2/u_1)}, \\ 0, & \text{otherwise,} \end{cases}$$

where k is a constant value, because the bivariate random variable (U_1, U_2) is uniformly distributed.

Consider the following transformation from (u_1, u_2) to (x, y):

$$x = \frac{u_2}{u_1}, \qquad y = u_1,$$

i.e.,

$$u_1 = y, \qquad u_2 = xy.$$

The Jacobian for the transformation is:

$$J = \begin{vmatrix} \dfrac{\partial u_1}{\partial x} & \dfrac{\partial u_1}{\partial y} \\ \dfrac{\partial u_2}{\partial x} & \dfrac{\partial u_2}{\partial y} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ y & x \end{vmatrix} = -y.$$

Therefore, the joint density of X and Y, denoted by $f_{xy}(x, y)$, is written as:

$$f_{xy}(x, y) = |J| f_{12}(y, xy) = ky,$$

for $0 \leq y \leq \sqrt{h(x)}$.

The marginal density of X, denoted by $f_x(x)$, is obtained as follows:

$$f_x(x) = \int_0^{\sqrt{h(x)}} f_{xy}(x, y)\, \mathrm{d}y = \int_0^{\sqrt{h(x)}} ky\, \mathrm{d}y = k\left[\frac{y^2}{2}\right]_0^{\sqrt{h(x)}} = \frac{k}{2} h(x) = f(x),$$

where k is taken as: $k = 2/\int h(x)\, \mathrm{d}x$.

Thus, it is shown that $f_x(\cdot)$ is equivalent to $f(\cdot)$.

This result is due to Kinderman and Monahan (1977).

Also see Ripley (1987), O'Hagan (1994), Fishman (1996) and Gentle (1998).

Now, we take an example of choosing the domain of (U_1, U_2).

In practice, for the domain of (U_1, U_2), we may choose the rectangle which encloses the area $0 \leq U_1 \leq \sqrt{h(U_2/U_1)}$, generate a uniform point in the rectangle, and reject the point which does not satisfy $0 \leq u_1 \leq \sqrt{h(u_2/u_1)}$.

That is, generate two independent uniform random draws u_1 and u_2 from $U(0, b)$ and $U(c, d)$, respectively.

The rectangle is given by:

$$0 \leq u_1 \leq b, \qquad c \leq u_2 \leq d,$$

where b, c and d are given by:

$$b = \sup_x \sqrt{h(x)}, \qquad c = -\sup_x x\sqrt{h(x)}, \qquad d = \sup_x x\sqrt{h(x)},$$

because the rectangle has to enclose $0 \leq u_1 \leq \sqrt{h(u_2/u_1)}$, which is verified as follows:

$$\begin{aligned} &0 \leq u_1 \leq \sqrt{h(u_2/u_1)} \leq \sup_x \sqrt{h(x)}, \\ &-\sup_x x\sqrt{h(x)} \leq -x\sqrt{h(x)} \leq u_2 \leq x\sqrt{h(x)} \leq \sup_x x\sqrt{h(x)}. \end{aligned}$$

The second line also comes from $0 \leq u_1 \leq \sqrt{h(u_2/u_1)}$ and $x = u_2/u_1$.

We can replace $c = -\sup_x x\sqrt{h(x)}$ by $c = \inf_x x\sqrt{h(x)}$, taking into account the case of $-\sup_x x\sqrt{h(x)} \leq \inf_x x\sqrt{h(x)}$.

The discussion above is shown in Ripley (1987).

Thus, in order to apply the ratio-of-uniforms method with the domain $\{0 \leq u_1 \leq b,\ c \leq u_2 \leq d\}$, we need to have the condition that $h(x)$ and $x^2h(x)$ are bounded.

The algorithm for the ratio-of-uniforms method is as follows:

(i) Generate u_1 and u_2 independently from $U(0, b)$ and $U(c, d)$.

(ii) Set $x = u_2/u_1$ if $u_1^2 \leq h(u_2/u_1)$ and return to (i) otherwise.

As shown above, the x accepted in Step (ii) is taken as a random draw from $f(x) =$

$h(x)/\int h(x)\,\mathrm{d}x$.

The acceptance probability in Step (ii) is $\int h(x)\,\mathrm{d}x/(2b(d-c))$.

We have shown the rectangular domain of (U_1, U_2).

It may be possible that the domain of (U_1, U_2) is a parallelogram.

In Sections 5.7.4 and 5.7.4, we show two examples as applications of the ratio-of-uniforms method.

Especially, in Section 5.7.4, the parallelogram domain of (U_1, U_2) is taken as an example.

Normal Distribution: $N(0,1)$**:** The kernel of the standard normal distribution is given by: $h(x) = \exp(-\frac{1}{2}x^2)$.

In this case, b, c and d are obtained as follows:

$$b = \sup_x \sqrt{h(x)} = 1,$$
$$c = \inf_x x\sqrt{h(x)} = -\sqrt{2e^{-1}},$$
$$d = \sup_x x\sqrt{h(x)} = \sqrt{2e^{-1}}.$$

Accordingly, the standard normal random number based on the ratio-of-uniforms method is represented as follows.

(i) Generate two independent uniform random draws u_1 and v_2 from $U(0, 1)$ and define $u_2 = (2v_2 - 1)\sqrt{2e^{-1}}$.

(ii) Set $x = u_2/u_1$ if $u_1^2 \leq \exp(-\frac{1}{2}u_2^2/u_1^2)$, i.e., $-4u_1^2 \log(u_1) \geq u_2^2$, and return to (i) otherwise.

The acceptance probability is given by:

$$\frac{\int h(x)\,\mathrm{d}x}{2b(d-c)} = \frac{\sqrt{\pi e}}{4} \approx 0.7306,$$

which is slightly smaller than the acceptance probability in the case of rejection sampling, i.e., $1/\sqrt{2e/\pi} \approx 0.7602$.

The Fortran source code for the standard normal random number generator based on the ratio-of-uniforms method is shown in snrnd9(ix,iy,rn).

snrnd9(ix,iy,rn)

```
1:        subroutine snrnd9(ix,iy,rn)
2:  c
3:  c  Use "snrnd9(ix,iy,rn)"
4:  c  together with "urnd(ix,iy,rn)".
5:  c
6:  c  Input:
7:  c    ix, iy:   Seeds
8:  c  Output:
9:  c    rn: Normal Random Draw N(0,1)
10: c
11:       e1=1./2.71828182845905
```

```
      1 call urnd(ix,iy,rn1)
        call urnd(ix,iy,rn2)
        rn2=(2.*rn2-1.)*sqrt(2.*e1)
        if(-4.*rn1*rn1*log(rn1).lt.rn2*rn2 ) go to 1
        rn=rn2/rn1
        return
        end
```

Gamma Distribution: $G(\alpha,\beta)$**:** When random variable X has a gamma distribution with parameters α and β, i.e., $X \sim G(\alpha,\beta)$, the density function of X is written as follows:

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

for $0 < x < \infty$.

When $X \sim G(\alpha, 1)$, we have $Y = \beta X \sim G(\alpha, \beta)$.

Therefore, first we consider generating a random draw of $X \sim G(\alpha, 1)$.

Since we have discussed the case of $0 < \alpha \leq 1$ in Sections 5.7.1 – 5.7.3, now we consider the case of $\alpha > 1$.

Using the ratio-of-uniforms method, the gamma random number generator is introduced.

$h(x)$, b, c and d are set to be:

$$h(x) = x^{\alpha-1}e^{-x},$$

$$b = \sup_x \sqrt{h(x)} = \left(\frac{\alpha - 1}{e}\right)^{(\alpha-1)/2},$$

$$c = \inf_x x\sqrt{h(x)} = 0,$$

$$d = \sup_x x\sqrt{h(x)} = \left(\frac{\alpha + 1}{e}\right)^{(\alpha+1)/2}.$$

Note that $\alpha > 1$ guarantees the existence of the supremum of $h(x)$, which implies $b > 0$.

See Fishman (1996, pp.194 – 195) and Ripley (1987, pp.88 – 89).

By the ratio-of-uniforms method, the gamma random number with parameter $\alpha > 1$ and $\beta = 1$ is represented as follows:

(i) Generate two independent uniform random draws u_1 and u_2 from $U(0, b)$ and $U(c, d)$, respectively.

(ii) Set $x = u_2/u_1$ if $u_1 \leq \sqrt{(u_2/u_1)^{\alpha-1}e^{-u_2/u_1}}$ and go back to (i) otherwise.

Thus, the x obtained in Steps (i) and (ii) is taken as a random draw from $G(\alpha, 1)$ for $\alpha > 1$.

Based on the above algorithm represented by Steps (i) and (ii), the Fortran 77 program for the gamma random number generator with parameters $\alpha > 1$ and $\beta = 1$ is shown in `gammarnd6(ix,iy,alpha,rn)`.

gammarnd6(ix,iy,alpha,rn)

```
      subroutine gammarnd6(ix,iy,alpha,rn)
c
c  Use "gammarnd6(ix,iy,alpha,rn)"
c  together with "urnd(ix,iy,rn)".
c
c  Input:
c    ix, iy:   Seeds
c    alpha:  Shape Parameter (alpha>1)
c  Output:
c    rn: Gamma Random Draw
c        with Parameters alpha and beta=1
c
      e=2.71828182845905
      b=( (alpha-1.)/e )**(0.5*alpha-0.5)
      d=( (alpha+1.)/e )**(0.5*alpha+0.5)
    1 call urnd(ix,iy,rn0)
      call urnd(ix,iy,rn1)
      u=rn0*b
```

```
19:         v=rn1*d
20:         rn=v/u
21:         if( 2.*log(u).gt.(alpha-1.)*log(rn)-rn ) go to 1
22:         return
23:         end
```

`gammarnd6(ix,iy,alpha,rn)` should be used together with `urnd(ix,iy,rn)`.

b and *d* are obtained in Lines 14 and 15.

Lines 16 –19 gives us two uniform random draws `u` and `v`, which correspond to u_1 and u_2.

`rn` in Line 20 indicates a candidate of the gamma random draw.

Line 21 represents Step (ii).

To see efficiency or inefficiency of the generator above, we compute the acceptance probability in Step (ii) as follows:

$$\frac{\int h(x)\,\mathrm{d}x}{2b(d-c)} = \frac{e^{\alpha}\Gamma(\alpha)}{2(\alpha-1)^{(\alpha-1)/2}(\alpha+1)^{(\alpha+1)/2}}. \tag{6}$$

It is known that the acceptance probability decreases by the order of $O(\alpha^{-1/2})$, i.e., in other words, computational time for random number generation increases by the order of $O(\alpha^{1/2})$.

Therefore, as α is larger, the generator is less efficient.

See Fishman (1996) and Gentle (1998).

To improve inefficiency for large α, various methods have been proposed, for example, Cheng and Feast (1979, 1980), Schmeiser and Lal (1980), Sarkar (1996) and so on.

As mentioned above, the algorithm `gammarnd6` takes a long time computationally by the order of $O(\alpha^{1/2})$ as shape parameter α is large.

Chen and Feast (1979) suggested the algorithm which does not depend too much on shape parameter α.

As α increases the acceptance region shrinks toward $u_1 = u_2$.

Therefore, Chen and Feast (1979) suggested generating two uniform random draws

within the parallelogram around $u_1 = u_2$, rather than the rectangle.

The source code is shown in gammarnd7(ix,iy,alpha,rn).

gammarnd7(ix,iy,alpha,rn)

```
      subroutine gammarnd7(ix,iy,alpha,rn)
c
c  Use "gammarnd7(ix,iy,alpha,rn)"
c  together with "urnd(ix,iy,rn)".
c
c  Input:
c    ix, iy:   Seeds
c    alpha:  Shape Parameter (alpha>1)
c  Output:
c    rn: Gamma Random Draw
c        with Parameters alpha and beta=1
```

```
C
      e =2.71828182845905
      c0=1.857764
      c1=alpha-1.
      c2=( alpha-1./(6.*alpha) )/c1
      c3=2./c1
      c4=c3+2.
      c5=1./sqrt(alpha)
    1 call urnd(ix,iy,u1)
      call urnd(ix,iy,u2)
      if(alpha.gt.2.5) u1=u2+c5*(1.-c0*u1)
      if(0.ge.u1.or.u1.ge.1.) go to 1
      w=c2*u2/u1
      if(c3*u1+w+1./w.le.c4) go to 2
      if(c3*log(u1)-log(w)+w.ge.1.) go to 1
    2 rn=c1*w
      return
      end
```

See Fishman (1996, p.200) and Ripley (1987, p.90).

In Line 22, we use the rectangle for $1 < \alpha \leq 2.5$ and the parallelogram for $\alpha > 2.5$ to give a fairly constant speed as α is varied.

Line 25 gives us a fast acceptance to avoid evaluating the logarithm.

From computational efficiency, `gammarnd7(ix,iy,alpha,rn)` is better.

Gamma Distribution: $G(\alpha, \beta)$ **for** $\alpha > 0$ **and** $\beta > 0$**:** Combining `gammarnd2` on p.353 and `gammarnd7` on p.441, we introduce the gamma random number generator in the case of $\alpha > 0$.

In addition, utilizing $Y = \beta X \sim G(\alpha, \beta)$ when $X \sim G(\alpha, 1)$, the random number generator for $G(\alpha, \beta)$ is introduced as in the source code `gammarnd8(ix,iy,alpha,beta,rn`

gammarnd8(ix,iy,alpha,beta,rn)

```
1:        subroutine gammarnd8(ix,iy,alpha,beta,rn)
2:  c
3:  c  Use "gammarnd8(ix,iy,alpha,beta,rn)"
4:  c  together with "gammarnd2(ix,iy,alpha,rn)",
5:  c                "gammarnd7(ix,iy,alpha,rn)"
6:  c           and  "urnd(ix,iy,rn)".
7:  c
8:  c  Input:
9:  c    ix, iy:   Seeds
10: c    alpha:  Shape Parameter
11: c    beta:   Scale Parameter
```

```
12: c  Output:
13: c    rn: Gamma Random Draw
14: c        with Parameters alpha and beta
15: c
16:          if( alpha.le.1. ) then
17:        call gammarnd2(ix,iy,alpha,rn1)
18:          else
19:        call gammarnd7(ix,iy,alpha,rn1)
20:          endif
21:        rn=beta*rn1
22:        return
23:        end
```

Lines 16 – 20 show that we use `gammarnd2` for $\alpha \leq 1$ and `gammarnd7` for $\alpha > 1$.

In Line 21, $X \sim G(\alpha, 1)$ is transformed into $Y \sim G(\alpha, \beta)$ by $Y = \beta X$, where X and Y

indicates `rn1` and `rn`, respectively.

Chi-Square Distribution: $\chi^2(k)$: The gamma distribution with $\alpha = k/2$ and $\beta = 2$ reduces to the chi-square distribution with k degrees of freedom.

5.7.5 Gibbs Sampling

The sampling methods introduced in Sections 5.7.1 – 5.7.3 can be applied to the cases of both univariate and multivariate distributions.

The Gibbs sampler in this section is the random number generation method in the

multivariate cases.

The Gibbs sampler shows how to generate random draws from the unconditional densities under the situation that we can generate random draws from two conditional densities.

Geman and Geman (1984), Tanner and Wong (1987), Gelfand, Hills, Racine-Poon and Smith (1990), Gelfand and Smith (1990), Carlin and Polson (1991), Zeger and Karim (1991), Casella and George (1992), Gamerman (1997) and so on developed the Gibbs sampling theory.

Carlin, Polson and Stoffer (1992), Carter and Kohn (1994, 1996) and Geweke

and Tanizaki (1999, 2001) applied the Gibbs sampler to the nonlinear and/or non-Gaussian state-space models.

There are numerous other applications of the Gibbs sampler.

The Gibbs sampling theory is concisely described as follows.

We can deal with more than two random variables, but we consider two random variables X and Y in order to make things easier.

Two conditional density functions, $f_{x|y}(x|y)$ and $f_{y|x}(y|x)$, are assumed to be known, which denote the conditional distribution function of X given Y and that of Y given X, respectively.

Suppose that we can easily generate random draws of X from $f_{x|y}(x|y)$ and those of Y from $f_{y|x}(y|x)$.

However, consider the case where it is not easy to generate random draws from the joint density of X and Y, denoted by $f_{xy}(x, y)$.

In order to have the random draws of (X, Y) from the joint density $f_{xy}(x, y)$, we take the following procedure:

(i) Take the initial value of X as x_{-M}.

(ii) Given x_{i-1}, generate a random draw of Y, i.e., y_i, from $f(y|x_{i-1})$.

(iii) Given y_i, generate a random draw of X, i.e., x_i, from $f(x|y_i)$.

(iv) Repeat the procedure for $i = -M+1, -M+2, \cdots, 1$.

From the convergence theory of the Gibbs sampler, as M goes to infinity, we can regard x_1 and y_1 as random draws from $f_{xy}(x, y)$, which is a joint density function of X and Y.

M denotes the **burn-in period**, and the first M random draws, (x_i, y_i) for $i = -M+1, -M+2, \cdots, 0$, are excluded from further consideration.

When we want N random draws from $f_{xy}(x, y)$, Step (iv) should be replaced by Step (iv)', which is as follows.

(iv)' Repeat the procedure for $i = -M+1, -M+2, \cdots, N$.

As in the Metropolis-Hastings algorithm, the algorithm shown in Steps (i) – (iii) and (iv)' is formulated as follows:

$$f_i(u) = \int f^*(u|v) f_{i-1}(v)\, \mathrm{d}v.$$

For convergence of the Gibbs sampler, we need to have the invariant distribution $f(u)$ which satisfies $f_i(u) = f_{i-1}(u) = f(u)$. If we have the reversibility condition shown in equation (4), i.e.,

$$f^*(v|u) f(u) = f^*(u|v) f(v),$$

the random draws based on the Gibbs sampler converge to those from the invariant

distribution, which implies that there exists the invariant distribution $f(u)$.

Therefore, in the Gibbs sampling algorithm, we have to find the transition distribution, i.e., $f^*(u|v)$.

Here, we consider that both u and v are bivariate vectors.

That is, $f^*(u|v)$ and $f_i(u)$ denote the bivariate distributions. x_i and y_i are generated from $f_i(u)$ through $f^*(u|v)$, given $f_{i-1}(v)$.

Note that $u = (u_1, u_2) = (x_i, y_i)$ is taken while $v = (v_1, v_2) = (x_{i-1}, y_{i-1})$ is set.

The transition distribution in the Gibbs sampler is taken as:

$$f^*(u|v) = f_{y|x}(u_2|u_1) f_{x|y}(u_1|v_2)$$

Thus, we can choose $f^*(u|v)$ as shown above.

Then, as i goes to infinity, (x_i, y_i) tends in distribution to a random vector whose joint density is $f_{xy}(x, y)$.

See, for example, Geman and Geman (1984) and Smith and Roberts (1993).

Furthermore, under the condition that there exists the invariant distribution, the basic result of the Gibbs sampler is as follows:

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i, y_i) \longrightarrow \mathrm{E}(g(x, y)) = \iint g(x, y) f_{xy}(x, y)\, \mathrm{d}x\, \mathrm{d}y, \quad \text{as } N \longrightarrow \infty,$$

where $g(\cdot, \cdot)$ is a function.

The Gibbs sampler is a powerful tool in a Bayesian framework.

Based on the conditional densities, we can generate random draws from the joint density.

Remark 1: We have considered the bivariate case, but it is easily extended to the multivariate cases.

That is, it is possible to take multi-dimensional vectors for x and y.

Taking an example, as for the tri-variate random vector (X, Y, Z), if we generate the ith random draws from $f_{x|yz}(x|y_{i-1}, z_{i-1})$, $f_{y|xz}(y|x_i, z_{i-1})$ and $f_{z|xy}(z|x_i, y_i)$, sequentially, we can obtain the random draws from $f_{xyz}(x, y, z)$.

Remark 2: Let X, Y and Z be the random variables.

Take an example of the case where X is highly correlated with Y.

If we generate random draws from $f_{x|yz}(x|y,z)$, $f_{y|xz}(y|x,z)$ and $f_{z|xy}(z|x,y)$, it is known that convergence of the Gibbs sampler is slow.

In this case, without separating X and Y, random number generation from $f(x,y|z)$ and $f(z|x,y)$ yields better random draws from the joint density $f(x,y,z)$.

Rejection Sampling, Importance Resampling and the Metropolis-Hastings Algorithm: We compare rejection sampling, importance resampling and the Metropolis-

Hastings algorithm from precision of the estimated moments and CPU time.

All the three sampling methods utilize the sampling density and they are useful when it is not easy to generate random draws directly from the target density.

When the sampling density is too far from the target density, it is known that rejection sampling takes a lot of time computationally while importance resampling and the Metropolis-Hastings algorithm yields unrealistic random draws.

In this section, therefore, we investigate how the sampling density depends on the three sampling methods.

For simplicity of discussion, consider the case where both the target and sampling

densities are normal.

That is, the target density $f(x)$ is given by $N(0, 1)$ and the sampling density $f_*(x)$ is $N(\mu_*, \sigma_*^2)$.

$\mu_* = 0, 1, 2, 3$ and $\sigma_* = 0.5, 1.0, 1.5, 2.0, 3.0, 4.0$ are taken.

For each of the cases, the first three moments $\mathrm{E}(X^j)$, $j = 1, 2, 3$, are estimated, generating 10^7 random draws.

For importance resampling, $n = 10^4$ is taken, which is the number of candidate random draws.

The Metropolis-Hastings algorithm takes $M = 1000$ as the burn-in period and the

initial value is $x_{-M} = \mu_*$.

As for the Metropolis-Hastings algorithm, note that is the independence chain is taken for $f_*(x)$ because of $f_*(x|z) = f_*(x)$.

Comparison of Three Sampling Methods

	μ_* \ σ_*		0.5	1.0	1.5	2.0	3.0	4.0
E(X) = 0	0	RS	—	—	0.000	0.000	0.000	0.000
		IR	0.060	0.005	0.000	0.005	0.014	0.014
		MH	−0.004 (59.25)	0.000 (100.00)	0.000 (74.89)	0.000 (59.04)	0.000 (40.99)	0.000 (31.21)
	1	RS	—	—	0.000	0.000	0.000	0.000
		IR	0.327	0.032	0.025	0.016	0.011	0.011
		MH	0.137 (36.28)	0.000 (47.98)	0.001 (55.75)	0.000 (51.19)	0.000 (38.68)	0.000 (30.23)
	2	RS	—	—	0.000	0.000	0.000	0.000
		IR	0.851	0.080	0.031	0.030	0.003	0.005
		MH	0.317 (8.79)	0.005 (15.78)	0.001 (26.71)	0.001 (33.78)	0.000 (32.50)	0.001 (27.47)
	3	RS	—	—	0.000	0.000	0.000	−0.001
		IR	1.590	0.337	0.009	0.029	0.021	−0.007
		MH	0.936 (1.68)	0.073 (3.53)	−0.002 (9.60)	0.000 (17.47)	0.001 (24.31)	−0.001 (23.40)

Comparison of Three Sampling Methods

	$\mu_* \backslash \sigma_*$		0.5	1.0	1.5	2.0	3.0	4.0
$E(X^2) = 1$	0	RS	—	—	1.000	1.000	1.000	0.999
		IR	0.822	0.972	0.969	0.978	0.994	1.003
		MH	0.958	1.000	1.000	1.000	1.001	1.001
	1	RS	—	—	1.000	1.000	1.000	1.000
		IR	0.719	0.980	0.983	0.993	1.010	1.004
		MH	0.803	1.002	0.999	0.999	1.001	1.002
	2	RS	—	—	1.000	1.000	1.001	1.001
		IR	1.076	0.892	1.014	0.984	1.000	1.012
		MH	0.677	0.992	1.001	0.999	1.001	1.002
	3	RS	—	—	1.000	1.000	1.000	1.000
		IR	2.716	0.696	1.013	1.025	0.969	1.002
		MH	1.165	0.892	1.005	1.001	0.999	0.999

Comparison of Three Sampling Methods

	$\mu_* \backslash \sigma_*$		0.5	1.0	1.5	2.0	3.0	4.0
$E(X^3) = 0$	0	RS	—	—	0.000	0.000	0.000	−0.001
		IR	0.217	0.034	−0.003	−0.018	0.018	0.036
		MH	−0.027	0.001	0.001	−0.001	−0.002	−0.004
	1	RS	—	—	0.002	−0.001	0.000	0.001
		IR	0.916	0.092	0.059	0.058	0.027	0.032
		MH	0.577	−0.003	0.003	0.000	0.002	−0.001
	2	RS	—	—	−0.001	0.002	0.001	0.001
		IR	1.732	0.434	0.052	0.075	0.040	0.001
		MH	0.920	0.035	0.003	0.004	0.004	0.004
	3	RS	—	—	0.000	0.001	0.001	−0.001
		IR	5.030	0.956	0.094	0.043	0.068	0.020
		MH	1.835	0.348	−0.002	0.003	0.001	−0.001

Comparison of Three Sampling Methods: CPU Time (Seconds)

$\mu_* \backslash \sigma_*$		0.5	1.0	1.5	2.0	3.0	4.0
0	RS	—	—	15.96	20.50	30.69	39.62
	IR	431.89	431.40	431.53	432.58	435.37	437.16
	MH	9.70	9.24	9.75	9.74	9.82	9.77
1	RS	—	—	23.51	24.09	32.77	41.03
	IR	433.22	427.96	426.41	426.36	427.80	430.39
	MH	9.73	9.54	9.81	9.75	9.83	9.76
2	RS	—	—	74.08	38.75	39.18	45.18
	IR	435.90	432.23	425.06	423.78	421.46	422.35
	MH	9.71	9.52	9.83	9.77	9.82	9.77
3	RS	—	—	535.55	87.00	52.91	53.09
	IR	437.32	439.31	429.97	424.45	422.91	418.38
	MH	9.72	9.48	9.79	9.75	9.81	9.76

RS, IR and MH denotes rejection sampling, importance resampling and the Metropolis-Hastings algorithm, respectively.

In each table, "—" in RS implies the case where rejection sampling cannot be applied because the supremum of $q(x)$, $\sup_x q(x)$, does not exist.

As for MH in the case of $\mathrm{E}(X) = 0$, the values in the parentheses represent the acceptance rate (percent) in the Metropolis-Hastings algorithm.

The results obtained from each table are as follows.

$\mathrm{E}(X)$ should be close to zero because we have $\mathrm{E}(X) = 0$ from $X \sim N(0, 1)$.

When $\mu_* = 0.0$, all of RS, IR and MH are very close to zero and show a good

performance.

When $\mu_* = 1, 2, 3$, for $\sigma_* = 1.5, 2.0, 3.0, 4.0$, all of RS, IR and MH perform well, but IR and MH in the case of $\sigma_* = 0.5, 1.0$ have the case where the estimated mean is too different from zero.

For IR and MH, we can see that given σ_* the estimated mean is far from the true mean as μ_* is far from mean of the target density.

Also, it might be concluded that given μ_* the estimated mean approaches the true value as σ_* is large.

$\mathrm{E}(X^2)$ should be close to one because we have $\mathrm{E}(X^2) = \mathrm{V}(X) = 1$ from $X \sim N(0, 1)$.

The cases of $\sigma_* = 1.5, 2.0, 3.0, 4.0$ and the cases of $\mu_* = 0, 1$ and $\sigma_* = 1.0$ are very close to one, but the other cases are different from one.

These are the same results as the case of $\mathrm{E}(X)$. $\mathrm{E}(X^3)$ in Table **??** should be close to zero because $\mathrm{E}(X^3)$ represents skewness.

For skewness, we obtain the similar results, i.e., the cases of $\sigma_* = 1.5, 2.0, 3.0, 4.0$ and the cases of $\mu_* = 0, 1$ and $\sigma_* = 0.5, 1.0$ perform well for all of RS, IR and MH.

In the case where we compare RS, IR and MH, RS shows the best performance of the three, and IR and MH is quite good when σ_* is relatively large.

We can conclude that IR is slightly worse than RS and MH.

As for the acceptance rates of MH in $E(X) = 0$, from the table a higher acceptance rate generally shows a better performance.

The high acceptance rate implies high randomness of the generated random draws. In Section **??**, the sampling density in the MH algorithm will be discussed in detail. From Table **??**, for variance of the sampling density, both too small variance and too large variance give us the relatively low acceptance rate, which result is consistent with the discussion in Chib and Greenberg (1995).

MH has the advantage over RS and IR from computational point of view.

IR takes a lot of time because all the acceptance probabilities have to be computed

in advance (see Section 5.7.2 for IR).

That is, 10^4 candidate random draws are generated from the sampling density $f_*(x)$ and therefore 10^4 acceptance probabilities have to be computed.

For MH and IR, computational CPU time does not depend on μ_* and σ_*.

However, for RS, given σ_* computational time increases as μ_* is large.

In other words, as the sampling density is far from the target density the number of rejections increases.

When σ_* increases given μ_*, the acceptance rate does not necessarily increase.

However, from the table a large σ_* is better than a small σ_* in general.

Accordingly, as for RS, under the condition that mean of $f(x)$ is unknown, we can conclude that relatively large variance of $f_*(x)$ should be taken.

Finally, the results are summarized as follows.

(1) For IR and MH, depending on choice of the sampling density $f_*(x)$, we have the cases where the estimates of mean, variance and skewness are biased.

For RS, we can always obtain the unbiased estimates without depending on choice of the sampling density.

(2) In order to avoid the biased estimates, it is safe for IR and MH to choose the sampling density with relatively large variance.

Furthermore, for RS we should take the sampling density with relatively large variance to reduce computational burden.

But, note that too large variance leads to an increase in computational disadvantages.

(3) MH is the least computational sampling method of the three.

For IR, all the acceptance probabilities have to be computed in advance and therefore

IR takes a lot of time to generate random draws.

In the case of RS, the amount of computation increases as $f_*(x)$ is far from $f(x)$.

(4) For the sampling density in MH, it is known that both too large variance and too small variance yield slow convergence of the obtained random draws.

The slow convergence implies that a great amount of random draws have to be generated from the sampling density for evaluation of the expectations such as $\mathrm{E}(X)$ and $\mathrm{V}(X)$.

Therefore, choice of the sampling density has to be careful,

Thus, RS gives us the best estimates in the sense of unbiasedness, but RS some-

times has the case where the supremum of $q(x)$ does not exist and in this case it is impossible to implement RS.

As the sampling method which can be applied to any case, MH might be preferred to IR and RS in a sense of less risk.

However, we should keep in mind that MH also has the problem which choice of the sampling density is very important.

References

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

Bernardo, J.M. and Smith, A.F.M., 1994, *Bayesian Theory*, John Wiley & Sons.

Besag, J., Green, P., Higdon, D. and Mengersen, K., 1995, “Bayesian Computation and Stochastic Systems,” *Statistical Science*, Vol.10, No.1, pp.3 – 66 (with discussion).

Boswell, M.T., Gore, S.D., Patil, G.P. and Taillie, C., 1993, “The Art of Computer

Generation of Random Variables," in *Handbook of Statistics, Vol.9*, edited by Rao, C.R., pp.661 – 721, North-Holland.

Carlin, B.P. and Louis, T.A., 1996, *Bayes and Empirical Bayes Methods for Data Analysis*, Chapman & Hall.

Carlin, B.P. and Polson, N.G., 1991, "Inference for Nonconjugate Bayesian Models Using the Gibbs Sampler," *Canadian Journal of Statistics*, Vol.19, pp.399 – 405.

Carlin, B.P., Polson, N.G. and Stoffer, D.S., 1992, "A Monte Carlo Approach to Nonnormal and Nonlinear State Space Modeling," *Journal of the American*

Statistical Association, Vol.87, No.418, pp.493 – 500.

Carter, C.K. and Kohn, R., 1994, "On Gibbs Sampling for State Space Models," *Biometrika*, Vol.81, No.3, pp.541 – 553.

Carter, C.K. and Kohn, R., 1996, "Markov Chain Monte Carlo in Conditionally Gaussian State Space Models," *Biometrika*, Vol.83, No.3, pp.589 – 601.

Casella, G. and George, E.I., 1992, "Explaining the Gibbs Sampler," *The American Statistician*, Vol.46, pp.167 – 174.

Chen, M.H., Shao, Q.M. and Ibrahim, J.G., 2000, *Monte Carlo Methods in Bayesian Computation*, Springer-Verlag.

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

Cheng, R.C.H., 1998, “Random Variate Generation,” in *Handbook of Simulation*, Chap.5, edited by Banks, J., pp.139 – 172, John Wiley & Sons.

Cheng, R.C.H. and Feast, G.M., 1979, “Some Simple Gamma Variate Generators,” *Applied Statistics*, Vol.28, No.3, pp.290 – 295.

Cheng, R.C.H. and Feast, G.M., 1980, “Gamma Variate Generators with Increased Shape Parameter Range,” *Communications of the ACM*, Vol.23, pp.389 – 393.

Chib, S. and Greenberg, E., 1995, “Understanding the Metropolis-Hastings Algo-

rithm," *The American Statistician*, Vol.49, No.4, pp.327 – 335.

Chib, S., Greenberg, E. and Winkelmann, R., 1998, "Posterior Simulation and Bayes Factors in Panel Count Data Models," *Journal of Econometrics*, Vol.86, No.1, pp.33 – 54.

Fishman, G.S., 1996, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer-Verlag.

Gamerman, D., 1997, *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, Chapman & Hall.

Gelfand, A.E., Hills, S.E., Racine-Poon, H.A. and Smith, A.F.M., 1990, "Illustra-

tion of Bayesian Inference in Normal Data Models Using Gibbs Sampling," *Journal of the American Statistical Association*, Vol.85, No.412, pp.972 – 985.

Gelfand, A.E. and Smith, A.F.M., 1990, "Sampling-Based Approaches to Calculating Marginal Densities," *Journal of the American Statistical Association*, Vol.85, No.410, pp.398 – 409.

Gelman, A., Roberts, G.O. and Gilks, W.R., 1996, "Efficient Metropolis Jumping Rules," in *Bayesian Statistics, Vol.5*, edited by Bernardo, J.M., Berger, J.O., Dawid, A.P. and Smith, A.F.M., pp.599 – 607, Oxford University Press.

Geman, S. and Geman D., 1984, “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.Pami-6, No.6, pp.721 – 741.

Gentle, J.E., 1998, *Random Number Generation and Monte Carlo Methods*, Springer-Verlag.

Geweke, J., 1992, “Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments,” in *Bayesian Statistics, Vol.4*, edited by Bernardo, J.M., Berger, J.O., Dawid, A.P. and Smith, A.F.M., pp.169 – 193 (with discussion), Oxford University Press.

Geweke, J., 1996, “Monte Carlo Simulation and Numerical Integration,” in *Handbook of Computational Economics, Vol.1*, edited by Amman, H.M., Kendrick, D.A. and Rust, J., pp.731 – 800, North-Holland.

Geweke, J. and Tanizaki, H., 1999, “On Markov Chain Monte-Carlo Methods for Nonlinear and Non-Gaussian State-Space Models,” *Communications in Statistics, Simulation and Computation*, Vol.28, No.4, pp.867 – 894.

Geweke, J. and Tanizaki, H., 2001, “Bayesian Estimation of State-Space Model Using the Metropolis-Hastings Algorithm within Gibbs Sampling,” *Computational Statistics and Data Analysis*, Vol.37, No.2, pp.151-170.

Geweke, J. and Tanizaki, H., 2003, "Note on the Sampling Distribution for the Metropolis-Hastings Algorithm," *Communications in Statistics, Theory and Methods*, Vol.32, No.4, pp.775 – 789.

Kinderman, A.J. and Monahan, J.F., 1977, "Computer Generation of Random Variables Using the Ratio of Random Deviates," *ACM Transactions on Mathematical Software*, Vol.3, pp.257 – 260.

Liu, J.S., 1996, "Metropolized Independent Sampling with Comparisons to Rejection Sampling and Importance Sampling," *Statistics and Computing*, Vol.6, pp.113 – 119.

Mengersen, K.L., Robert, C.P. and Guihenneuc-Jouyaux, C., 1999, "MCMC Convergence Diagnostics: A Reviewww," in *Bayesian Statistics, Vol.6*, edited by Bernardo, J.M., Berger, J.O., Dawid, A.P. and Smith, A.F.M., pp.514 – 440 (with discussion), Oxford University Press.

O'Hagan, A., 1994, *Kendall's Advanced Theory of Statistics*, Vol.2B (Bayesian Inference), Edward Arnold.

Ripley, B.D., 1987, *Stochastic Simulation*, John Wiley & Sons.

Robert, C.P. and Casella, G., 1999, *Monte Carlo Statistical Methods*, Springer-Verlag.

Sarkar, T.K., 1996, “A Composition-Alias Method for Generating Gamma Variates with Shape Parameter Greater Than 1,” *ACM Transactions on Mathematical Software*, Vol.22, pp.484 – 492.

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

Smith, A.F.M. and Gelfand, A.E., 1992, “Bayesian Statistics without Tears: A Sampling-Resampling Perspective,” *The American Statistician*, Vol.46, No.2, pp.84 – 88.

Smith, A.F.M. and Roberts, G.O., 1993, “Bayesian Computation via Gibbs Sam-

pler and Related Markov Chain Monte Carlo Methods," *Journal of the Royal Statistical Society*, Ser.B, Vol.55, No.1, pp.3 – 23.

Tanner, M.A. and Wong, W.H., 1987, "The Calculation of Posterior Distributions by Data Augmentation," *Journal of the American Statistical Association*, Vol.82, No.398, pp.528 – 550 (with discussion).

Tierney, L., 1994, "Markov Chains for Exploring Posterior Distributions," *The Annals of Statistics*, Vol.22, No.4, pp.1701 – 1762.

Zeger, S.L. and Karim, M.R., 1991, "Generalized Linear Models with Random Effects: A Gibbs Sampling Approach," *Journal of the American Statistical*

Association, Vol.86, No.413, pp.79 – 86.