Bayesian Method = Evaluation of Integration (Too much to say?)

- Numerical Integration
- Monte Carlo Integration
- Random Number Generation from $f_{\theta|y}(\theta|y)$

9.5.1 Evaluation of Expectation: Numerical Integration

Univariate Case: Consider integration of a function $f(x)$.

Suppose that x is a scalar.

Let $x_0, x_1, x_2, \cdots, x_n$ be n nodes, which are sorted by order of size but not necessarily equal intervals between x_{i-1} and x_i for $i = 1, 2, \cdots, n$.

Rectangular Approximation:

$$\int f(x)\mathrm{d}x \approx \sum_{i=1}^{n} f(x_i)(x_i - x_{i-1}),$$
$$\text{or } \sum_{i=1}^{n} f(x_{i-1})(x_i - x_{i-1}),$$
$$\text{or } \sum_{i=1}^{n} f(\frac{x_i + x_{i-1}}{2})(x_i - x_{i-1}).$$

Trapezoid Approximation:

$$\int f(x)\mathrm{d}x \approx \sum_{i=1}^{n} \frac{1}{2}(f(x_i) + f(x_{i-1}))(x_i - x_{i-1}).$$

Bivariate Case: Consider integration of a function $f(x, y)$.

Suppose that both x and y are scalars.

Let $x_0, x_1, x_2, \cdots, x_n$ be n nodes, which are sorted by order of size not necessarily equal intervals between x_{i-1} and x_i for $i = 1, 2, \cdots, n$.

Let $y_0, y_1, y_2, \cdots, y_m$ be m nodes.

Rectangular Approximation:

$$\int\int f(x,y)\mathrm{d}x\mathrm{d}y \approx \sum_{i=1}^{n}\sum_{j=1}^{m} f(\frac{x_i + x_{i-1}}{2}, \frac{y_j + y_{j-1}}{2})(x_i - x_{i-1})(y_j - y_{j-1}).$$

Trapezoid Approximation:

$$\int\int f(x.y)\mathrm{d}x\mathrm{d}y$$
$$\approx \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{1}{4}(f(x_i, y_j) + f(x_i, y_{j-1}) + f(x_{i-1}, y_j) + f(x_{i-1}, y_{j-1}))(x_i - x_{i-1})(y_j - y_{j-1}).$$

Applying to Bayes Method (Rectangular Approximation):

$$\mathrm{E}(\theta|y) = \frac{\int \theta f_{y|\theta}(y|\theta) f_\theta(\theta)\mathrm{d}\theta}{\int f_{y|\theta}(y|\theta) f_\theta(\theta)\mathrm{d}\theta} = \frac{\sum_{i=1}^{n} \theta_i f_{y|\theta}(y|\theta_i) f_\theta(\theta_i)(\theta_i - \theta_{i-1})}{\sum_{i=1}^{n} f_{y|\theta}(y|\theta_i) f_\theta(\theta_i)(\theta_i - \theta_{i-1})}$$

$$= \frac{\sum_{i=1}^{n} \theta_i f_{y|\theta}(y|\theta_i) f_\theta(\theta_i)}{\sum_{i=1}^{n} f_{y|\theta}(y|\theta_i) f_\theta(\theta_i)} = \sum_{i=1}^{n} \theta_i \omega_i, \quad \text{for constant } \theta_i - \theta_{i-1},$$

where

$$\omega_i = \frac{f_{y|\theta}(y|\theta_i) f_\theta(\theta_i)}{\sum_{i=1}^{n} f_{y|\theta}(y|\theta_i) f_\theta(\theta_i)}.$$

Problem of Numerical Integration:

1. Choice of initial and terminal values $\Longrightarrow$ Truncation errors
2. Accumulation of computational errors by computer
3. Increase of computational burden for large dimension.

   $\Longrightarrow$ k dimension, and n nodes for each dimension $\Longrightarrow$ n^k

9.5.2 Evaluation of Expectation: Monte Carlo Integration

Univariate Case: Consider integration of a function $f(x)$.

Suppose that x is a scalar.

Let $x_1, x_2, \cdots, x_n$ be n random draws generated from $g(x)$.

$$\int f(x)\mathrm{d}x = \int \frac{f(x)}{g(x)} g(x)\mathrm{d}x = \mathrm{E}\Big(\frac{f(x)}{g(x)}\Big) \approx \frac{1}{n}\sum_{i=1}^{n} \frac{f(x_i)}{g(x_i)}.$$

$\Longrightarrow$ **Importance Sampling (重点的サンプリング)**

Multivariate Case: Consider integration of a function $f(x)$.

Suppose that x is a vector.

Let $x_1, x_2, \cdots, x_n$ be n random draws generated from $g(x)$.

$$\int f(x)\mathrm{d}x = \int \frac{f(x)}{g(x)} g(x)\mathrm{d}x = \mathrm{E}\Big(\frac{f(x)}{g(x)}\Big) \approx \frac{1}{n}\sum_{i=1}^{n} \frac{f(x_i)}{g(x_i)},$$

which is exacly the same as the univariate case.

Computational burden: $\Longrightarrow$ Univariate case: n, Multivariate case: n

Precision of integration ???

Especially, when $g(x)$ is not close to $f(x)$, approximation is prror.

Applying to Bayes Method:

$$\mathrm{E}(\theta|y) = \frac{\int \theta f_{y|\theta}(y|\theta) f_\theta(\theta) \mathrm{d}\theta}{\int f_{y|\theta}(y|\theta) f_\theta(\theta) \mathrm{d}\theta} = \frac{\int \theta \dfrac{f_{y|\theta}(y|\theta) f_\theta(\theta)}{g(\theta)} g(\theta) \mathrm{d}\theta}{\int \dfrac{f_{y|\theta}(y|\theta) f_\theta(\theta)}{g(\theta)} g(\theta) \mathrm{d}\theta} = \frac{(1/n) \sum_{i=1}^n \theta_i \omega(\theta_i)}{(1/n) \sum_{i=1}^n \omega(\theta_i)},$$

where

$$\omega(\theta_i) = \frac{f_{y|\theta}(y|\theta_i) f_\theta(\theta_i)}{g(\theta_i)}.$$

Choice of $g(\theta)$ — One Solution: Define $l(\theta) \equiv f_{y|\theta}(y|\theta) f_\theta(\theta)$.

$$\log l(\theta) \approx \log l(\tilde{\theta}) + \frac{1}{l(\tilde{\theta})}\frac{\partial l(\tilde{\theta})}{\partial \theta}(\theta - \tilde{\theta})$$
$$+\frac{1}{2}(\theta - \tilde{\theta})'\Big(-\frac{1}{l(\tilde{\theta})^2}\frac{\partial l(\tilde{\theta})}{\partial \theta}\frac{\partial l(\tilde{\theta})}{\partial \theta'} + \frac{1}{l(\tilde{\theta})}\frac{\partial^2 l(\tilde{\theta})}{\partial \theta \partial \theta'}\Big)(\theta - \tilde{\theta})$$
$$= -\frac{1}{2}(\theta - \tilde{\theta})'\Big(-\frac{1}{l(\tilde{\theta})}\frac{\partial^2 l(\tilde{\theta})}{\partial \theta \partial \theta'}\Big)(\theta - \tilde{\theta}), \qquad \text{when } \tilde{\theta} \text{ is a mode of } l(\theta).$$

Thus, $N\Big(\tilde{\theta},\ \Big(-\frac{1}{l(\tilde{\theta})}\frac{\partial^2 l(\tilde{\theta})}{\partial \theta \partial \theta'}\Big)^{-1}\Big)$ might be taken as the importance density $g(\theta)$.

9.5.3 Evaluation of Expectation: Random Number Generation

Generate random draws of θ from the posterior distribution $f_{\theta|y}(\theta|y)$.

Then, $(1/n)\sum_{i=1}^{n}\theta_i$ is taken as a consistent estimator of $\mathrm{E}(\theta|y)$, where θ_i indicates the ith random draw generated from $f_{\theta|y}(\theta|y)$.

Note that $(1/n)\sum_{i=1}^{n}\theta_i \longrightarrow \mathrm{E}(\theta|y)$ under the condition $(1/n)\sum_{i=1}^{n}\theta_i < \infty$.

Bayesian confidence interval, median, quntiles and so on are obtained by sorting θ_1, $\theta_2, \cdots, \theta_n$ in order of size.

$\Longrightarrow$ Sampling methods

9.6 Sampling Method I: Random Number Generation

Note that a lot of distribution functions are introduced in Kotz, Balakrishman and Johnson (2000a, 2000b, 2000c, 2000d, 2000e).

The random draws discussed in this section are based on uniform random draws between zero and one.

9.6.1 Uniform Distribution: $U(0, 1)$

Properties of Uniform Distribution: The most heuristic and simplest distribution is uniform.

The **uniform distribution** between zero and one is given by:

$$f(x) = \begin{cases} 1, & \text{for } 0 < x < 1, \\ 0, & \text{otherwise.} \end{cases}$$

Mean, variance and the moment-generating function are given by:

$$\mathrm{E}(X) = \frac{1}{2}, \qquad \mathrm{V}(X) = \frac{1}{12}, \qquad \phi(\theta) = \frac{e^{\theta} - 1}{\theta}.$$

Use L'Hospital's theorem to derive $\mathrm{E}(X)$ and $\mathrm{V}(X)$ using $\phi(\theta)$.

In the next section, we introduce an idea of generating uniform random draws, which in turn yield the other random draws by the transformation of variables, the inverse transform algorithm and so on.

Uniform Random Number Generators: It is no exaggeration to say that all the random draws are based on a uniform random number.

Once uniform random draws are generated, the various random draws such as exponential, normal, logistic, Bernoulli and other distributions are obtained by transforming the uniform random draws.

Thus, it is important to consider how to generate a uniform random number.

However, generally there is no way to generate exact uniform random draws.

As shown in Ripley (1987) and Ross (1997), a deterministic sequence that appears at random is taken as a sequence of random numbers.

First, consider the following relation:

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

where k, m and n are integers.

$[k/n]$ denotes the largest integer less than or equal to the argument.

In Fortran 77, it is written as `m=k-int(k/n)*n`, where $0 \leq m < n$.

m indicates the **remainder (余り)** when k is divided by n.

n is called the **modulus (商)**.

We define the right hand side in the equation above as:

$$k - [k/n]n \equiv k \bmod n.$$

Then, using the modular arithmetic we can rewrite the above equation as follows:

$$m = k \bmod n,$$

which is represented by: `m=mod(k,n)` in Fortran 77 and `m=k%n` in C language.

A basic idea of the uniform random draw is as follows.

Given x_{i-1}, x_i is generated by:

$$x_i = (ax_{i-1} + c) \bmod n,$$

where $0 \leq x_i < n$.

a and c are positive integers, called the **multiplier** and the **increment**, respectively.

The generator above have to be started by an initial value, which is called the **seed**.

$u_i = x_i/n$ is regarded as a uniform random number between zero and one.

This generator is called the **linear congruential generator (線形合同法)**.

Especially, when $c = 0$, the generator is called the **multiplicative linear congruential generator**.

This method was proposed by Lehmer in 1948 (see Lehmer, 1951).

If n, a and c are properly chosen, the period of the generator is n.

However, when they are not chosen very carefully, there may be a lot of serial correlation among the generated values.

Therefore, the performance of the congruential generators depend heavily on the choice of (a, c).

There is a great amount of literature on uniform random number generation.

See, for example, Fishman (1996), Gentle (1998), Kennedy and Gentle (1980), Law and Kelton (2000), Niederreiter (1992), Ripley (1987), Robert and Casella (1999), Rubinstein and Melamed (1998), Thompson (2000) and so on for the other congruential generators.

However, we introduce only two uniform random number generators.

Wichmann and Hill (1982 and corrigendum, 1984) describe a combination of three congruential generators for 16-bit computers.

The generator is given by:

$$x_i = 171x_{i-1} \bmod 30269,$$

$$y_i = 172y_{i-1} \bmod 30307,$$

$$z_i = 170z_{i-1} \bmod 30323,$$

and

$$u_i = \left(\frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323}\right) \bmod 1.$$

We need to set three seeds, i.e., x_0, y_0 and z_0, for this random number generator.

u_i is regarded as a uniform random draw within the interval between zero and one.

The period is of the order of 10^{12} (more precisely the period is 6.95×10^{12}).

The source code of this generator is given by `urnd16(ix,iy,iz,rn)`, where `ix`, `iy` and `iz` are seeds and `rn` represents the uniform random number between zero and one.

`urnd16(ix,iy,iz,rn)`

```
1:       subroutine urnd16(ix,iy,iz,rn)
2: c
3: c  Input:
4: c    ix, iy, iz:  Seeds
5: c  Output:
6: c    rn: Uniform Random Draw U(0,1)
```

```
7: c
8:       1 ix=mod( 171*ix,30269 )
9:         iy=mod( 172*iy,30307 )
10:        iz=mod( 170*iz,30323 )
11:        rn=ix/30269.+iy/30307.+iz/30323.
12:        rn=rn-int(rn)
13:        if( rn.le.0 ) go to 1
14:        return
15:        end
```

We exclude one in Line 12 and zero in Line 13 from rn.

That is, $0 <$ rn < 1 is generated in urnd16(ix,iy,iz,rn).

Zero and one in the uniform random draw sometimes cause the complier errors in programming, when the other random draws are derived based on the transformation of the uniform random variable.

De Matteis and Pagnutti (1993) examine the Wichmann-Hill generator with respect to the higher order autocorrelations in sequences, and conclude that the Wichmann-

Hill generator performs well.

For 32-bit computers, L'Ecuyer (1988) proposed a combination of k congruential generators that have prime moduli n_j, such that all values of $(n_j - 1)/2$ are relatively prime, and with multipliers that yield full periods.

Let the sequence from jth generator be $x_{j,1}, x_{j,2}, x_{j,3}, \cdots$.

Consider the case where each individual generator j is a maximum-period multiplicative linear congruential generator with modulus n_j and multiplier a_j, i.e.,

$$x_{j,i} \equiv a_j x_{j,i-1} \bmod n_j.$$

Assuming that the first generator is a relatively good one and that n_1 is fairly large, we form the ith integer in the sequence as:

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

where the other moduli n_j, $j = 2, 3, \cdots, k$, do not need to be large.

The normalization takes care of the possibility of zero occurring in this sequence:

$$u_i = \begin{cases} \dfrac{x_i}{n_1}, & \text{if } x_i > 0, \\ \dfrac{n_1 - 1}{n_1}, & \text{if } x_i = 0. \end{cases}$$

As for each individual generator j, note as follows.

Define $q = [n/a]$ and $r \equiv n \bmod a$, i.e., n is decomposed as $n = aq + r$, where $r < a$.

Therefore, for $0 < x < n$, we have:

$$\begin{aligned} ax \bmod n &= (ax - [x/q]n) \bmod n \\ &= \Big(ax - [x/q](aq + r)\Big) \bmod n \\ &= \Big(a(x - [x/q]q) - [x/q]r\Big) \bmod n \\ &= \Big(a(x \bmod q) - [x/q]r\Big) \bmod n. \end{aligned}$$

Practically, L'Ecuyer (1988) suggested combining two multiplicative congruential generators, where $k = 2$, $(a_1, n_1, q_1, r_1) = (40014, 2147483563, 53668, 12211)$ and $(a_2, n_2, q_2, r_2) = (40692, 2147483399, 52774, 3791)$ are chosen.

Two seeds are required to implement the generator.

The source code is shown in `urnd(ix,iy,rn)`, where `ix` and `iy` are inputs, i.e., seeds, and `rn` is an output, i.e., a uniform random number between zero and one.

`urnd(ix,iy,rn)`

```
1:        subroutine urnd(ix,iy,rn)
2:  c
3:  c  Input:
4:  c    ix, iy:  Seeds
5:  c  Output:
6:  c    rn: Uniform Random Draw U(0,1)
7:  c
8:       1 kx=ix/53668
9:         ix=40014*(ix-kx*53668)-kx*12211
10:        if(ix.lt.0) ix=ix+2147483563
11: c
```

```
12:         ky=iy/52774
13:         iy=40692*(iy-ky*52774)-ky*3791
14:         if(iy.lt.0) iy=iy+2147483399
15:   c
16:         rn=ix-iy
17:         if( rn.lt.1.) rn=rn+2147483562
18:         rn=rn*4.656613e-10
19:         if( rn.le.0.) go to 1
20:   c
21:         return
22:         end
```

The period of the generator proposed by L'Ecuyer (1988) is of the order of 10^{18} (more precisely 2.31×10^{18}), which is quite long and practically long enough.

L'Ecuyer (1988) presents the results of both theoretical and empirical tests, where the above generator performs well.

Furthermore, L'Ecuyer (1988) gives an additional portable generator for 16-bit computers.

Also, see L'Ecuyer(1990, 1998).

To improve the length of period, the above generator proposed by L'Ecuyer (1988) is combined with the shuffling method suggested by Bays and Durham (1976), and it is introduced as `ran2` in Press, Teukolsky, Vetterling and Flannery (1992a, 1992b). However, from relatively long period and simplicity of the source code, hereafter the subroutine `urnd(ix,iy,rn)` is utilized for the uniform random number generation method, and we will obtain various random draws based on the uniform random draws.

9.6.2 Transforming $U(0,1)$: Continuous Type

In this section, we focus on a continuous type of distributions, in which density functions are derived from the uniform distribution $U(0,1)$ by transformation of variables.

Normal Distribution: $N(0,1)$**:** The normal distribution with mean zero and variance one, i.e, the standard normal distribution, is represented by:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2},$$

for $-\infty < x < \infty$.

Mean, variance and the moment-generating function are given by:

$$\mathrm{E}(X) = 0, \qquad \mathrm{V}(X) = 1, \qquad \phi(\theta) = \exp\Big(\frac{1}{2}\theta^2\Big).$$

The normal random variable is constructed using two independent uniform random variables.

This transformation is well known as the Box-Muller (1958) transformation and is shown as follows.

Let U_1 and U_2 be uniform random variables between zero and one.

Suppose that U_1 is independent of U_2.

Consider the following transformation:

$$X_1 = \sqrt{-2\log(U_1)}\cos(2\pi U_2),$$
$$X_2 = \sqrt{-2\log(U_1)}\sin(2\pi U_2).$$

where we have $-\infty < X_1 < \infty$ and $-\infty < X_2 < \infty$ when $0 < U_1 < 1$ and $0 < U_2 < 1$.

Then, the inverse transformation is given by:

$$u_1 = \exp\left(-\frac{x_1^2 + x_2^2}{2}\right), \qquad u_2 = \frac{1}{2\pi}\arctan\frac{x_2}{x_1}.$$

We perform transformation of variables in multivariate cases.

From this transformation, the Jacobian is obtained as:

$$J = \begin{vmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} \\ \frac{\partial u_2}{\partial x_1} & \frac{\partial u_2}{\partial x_2} \end{vmatrix} = \begin{vmatrix} -x_1 \exp\left(-\frac{1}{2}(x_1^2 + x_2^2)\right) & -x_2 \exp\left(-\frac{1}{2}(x_1^2 + x_2^2)\right) \\ \frac{1}{2\pi}\frac{-x_2}{x_1^2 + x_2^2} & \frac{1}{2\pi}\frac{x_1}{x_1^2 + x_2^2} \end{vmatrix}$$
$$= -\frac{1}{2\pi}\exp\left(-\frac{1}{2}(x_1^2 + x_2^2)\right).$$

Let $f_x(x_1, x_2)$ be the joint density of X_1 and X_2 and $f_u(u_1, u_2)$ be the joint density of U_1 and U_2.

Since U_1 and U_2 are assumed to be independent, we have the following:

$$f_u(u_1, u_2) = f_1(u_1) f_2(u_2) = 1,$$

where $f_1(u_1)$ and $f_2(u_2)$ are the density functions of U_1 and U_2, respectively.

Note that $f_1(u_1) = f_2(u_2) = 1$ because U_1 and U_2 are uniform random variables between zero and one.

Accordingly, the joint density of X_1 and X_2 is:

$$\begin{aligned} f_x(x_1, x_2) &= |J| f_u\Big(\exp(-\frac{x_1^2 + x_2^2}{2}), \frac{1}{2\pi} \arctan \frac{x_2}{x_1}\Big) \\ &= \frac{1}{2\pi} \exp\Big(-\frac{1}{2}(x_1^2 + x_2^2)\Big) \\ &= \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{1}{2}x_1^2\Big) \times \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{1}{2}x_2^2\Big), \end{aligned}$$

which is a product of two standard normal distributions.

Thus, X_1 and X_2 are mutually independently distributed as normal random variables with mean zero and variance one.

See Hogg and Craig (1995, pp.177 – 178).

The source code of the standard normal random number generator shown above is given by `snrnd(ix,iy,rn)`.

snrnd(ix,iy,rn)

```
1:        subroutine snrnd(ix,iy,rn)
2:  c
3:  c  Use "snrnd(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: Standard Normal Random Draw N(0,1)
10: c
11:       pi= 3.1415926535897932385
```

```
12:         call urnd(ix,iy,rn1)
13:         call urnd(ix,iy,rn2)
14:         rn=sqrt(-2.0*log(rn1))*sin(2.0*pi*rn2)
15:         return
16:         end
```

`snrnd(ix,iy,rn)` should be used together with the uniform random number generator `urnd(ix,iy,rn)` shown in Section 9.6.1 (p.219).

`rn` in `snrnd(ix,iy,rn)` corresponds to X_2.

Conventionally, one of X_1 and X_2 is taken as the random number which we use.

Here, X_1 is excluded from consideration.

`snrnd(ix,iy,rn)` includes the sine, which takes a lot of time computationally.

Therefore, to avoid computation of the sine, various algorithms have been invented (Ahrens and Dieter (1988), Fishman (1996), Gentle (1998), Marsaglia, MacLaren and Bray (1964) and so on).

Standard Normal Probabilities When $X \sim N(0,1)$, we have the case where we want to approximate p such that $p = F(x)$ given x, where $F(x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t = P(X < x)$.

Adams (1969) reports that

$$P(X > x) = \int_x^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}\,\mathrm{d}t = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\Big(\frac{1}{x+}\frac{1}{x+}\frac{2}{x+}\frac{3}{x+}\frac{4}{x+}\cdots\Big),$$

for $x > 0$, where the form in the parenthesis is called the continued fraction, which is defined as follows:

$$\frac{a_1}{x_1+}\frac{a_2}{x_2+}\frac{a_3}{x_3+}\cdots = \cfrac{a_1}{x_1 + \cfrac{a_2}{x_2 + \cfrac{a_3}{x_3 + \cdots}}}.$$

A lot of approximations on the continued fraction shown above have been proposed. See Kennedy and Gentle (1980), Marsaglia (1964) and Marsaglia and Zaman (1994).

Here, we introduce the following approximation (see Takeuchi (1989)):

$$P(X > x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5), \qquad t = \frac{1}{1 + a_0 x},$$

$$a_0 = 0.2316419, \qquad b_1 = 0.319381530, \qquad b_2 = -0.356563782,$$

$$b_3 = 1.781477937, \quad b_4 = -1.821255978, \quad b_5 = 1.330274429.$$

In `snprob(x,p)` below, $P(X < x)$ is shown.

That is, `p` up to Line 19 is equal to $P(X > x)$ in `snprob(x,p)`.

In Line 20, $P(X < x)$ is obtained.

`snprob(x,p)`

```
1:        subroutine snprob(x,p)
2:  c
3:  c  Input:
4:  c    x:  N(0,1) Percent Point
5:  c  Output:
6:  c    p:  Probability corresponding to x
```

```
C
      pi= 3.1415926535897932385
      a0= 0.2316419
      b1= 0.319381530
      b2=-0.356563782
      b3= 1.781477937
      b4=-1.821255978
      b5= 1.330274429
C
      z=abs(x)
      t=1.0/(1.0+a0*z)
      pr=exp(-.5*z*z)/sqrt(2.0*pi)
      p=pr*t*(b1+t*(b2+t*(b3+t*(b4+b5*t))))
      if(x.gt.0.0) p=1.0-p
C
      return
      end
```

The maximum error of approximation of p is 7.5×10^{-8}, which practically gives us enough precision.

Standard Normal Percent Points When $X \sim N(0,1)$, we approximate x such that $p = F(x)$ given p, where $F(x)$ indicates the standard normal cumulative distribution function, i.e., $F(x) = P(X < x)$, and p denotes probability.

As shown in Odeh and Evans (1974), the approximation of a percent point is of the form:

$$x = y + \frac{S_4(y)}{T_4(y)} = y + \frac{p_0 + p_1 y + p_2 y^2 + p_3 y^3 + p_4 y^4}{q_0 + q_1 y + q_2 y^2 + q_3 y^3 + q_4 y^4},$$

where $y = \sqrt{-2\log(p)}$.

$S_4(y)$ and $T_4(y)$ denote polynomials degree 4.

The source code is shown in `snperpt(p,x)`, where x is obtained within $10^{-20} < p < 1 - 10^{-20}$.

snperpt(p,x)

```
1:      subroutine snperpt(p,x)
```

```
C
C  Input:
C    p:  Probability
C       (err<p<1-err, where err=1e-20)
C  Output:
C    x:  N(0,1) Percent Point corresponding to p
C
       p0=-0.322232431088
       p1=-1.0
       p2=-0.342242088547
       p3=-0.204231210245e-1
       p4=-0.453642210148e-4
       q0= 0.993484626060e-1
       q1= 0.588581570495
       q2= 0.531103462366
       q3= 0.103537752850
       q4= 0.385607006340e-2
       ps=p
       if( ps.gt.0.5 ) ps=1.0-ps
       if( ps.eq.0.5 ) x=0.0
       y=sqrt( -2.0*log(ps) )
       x=y+((((y*p4+p3)*y+p2)*y+p1)*y+p0)
      &   /((((y*q4+q3)*y+q2)*y+q1)*y+q0)
       if( p.lt.0.5 ) x=-x
       return
       end
```

The maximum error of approximation of x is 1.5×10^{-8} if the function is evaluated in double precision and 1.8×10^{-6} if it is evaluated in single precision.

The approximation of the form $x = y + S_2(y)/T_3(y)$ by Hastings (1955) gives a maximum error of 4.5×10^{-4}.

To improve accuracy of the approximation, Odeh and Evans (1974) proposed the algorithm above.

Normal Distribution: $N(\mu, \sigma^2)$**:** The normal distribution denoted by $N(\mu, \sigma^2)$ is represented as follows:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2},$$

for $-\infty < x < \infty$.

μ is called a **location parameter** and σ^2 is a **scale parameter**.

Mean, variance and the moment-generating function of the normal distribution $N(\mu, \sigma^2)$

are given by:

$$\mathrm{E}(X) = \mu, \qquad \mathrm{V}(X) = \sigma^2, \qquad \phi(\theta) = \exp\left(\mu\theta + \frac{1}{2}\sigma^2\theta^2\right).$$

When $\mu = 0$ and $\sigma^2 = 1$ are taken, the above density function reduces to the standard normal distribution in Section 9.6.2.

$X = \sigma Z + \mu$ is normally distributed with mean μ and variance σ^2, when $Z \sim N(0, 1)$. Therefore, the source code is represented by `nrnd(ix,iy,ave,var,rn)`, where `ave` and `var` correspond to μ and σ^2, respectively.

nrnd(ix,iy,ave,var,rn)

```
      subroutine nrnd(ix,iy,ave,var,rn)
C
C  Use "nrnd(ix,iy,ave,var,rn)"
C  together with "urnd(ix,iy,rn)"
C           and  "snrnd(ix,iy,rn)".
C
```

```
c  Input:
c    ix, iy:  Seeds
c    ave: Mean
c    var: Variance
c  Output:
c    rn: Normal Random Draw N(ave,var)
c
       call snrnd(ix,iy,rn1)
       rn=ave+sqrt(var)*rn1
       return
       end
```

`nrnd(ix,iy,ave,var,rn)` should be used together with `urnd(ix,iy,rn)` and `snrnd(ix,iy,rn)`. It is possible to replace `snrnd(ix,iy,rn)` by `snrnd2(ix,iy,rn)` or `snrnd3(ix,iy,rn)`.

Exponential Distribution: The exponential distribution with parameter β is written as:

$$f(x) = \begin{cases} \frac{1}{\beta} e^{-\frac{x}{\beta}}, & \text{for } 0 < x < \infty, \\ 0, & \text{otherwise,} \end{cases}$$

for $\beta > 0$.

β indicates a scale parameter.

Mean, variance and the moment-generating function are obtained as follows:

$$\mathrm{E}(X) = \beta, \qquad \mathrm{V}(X) = \beta^2, \qquad \phi(\theta) = \frac{1}{1 - \beta\theta}.$$

The relation between the exponential random variable the uniform random variable is shown as follows:

When $U \sim U(0, 1)$, consider the following transformation:

$$X = -\beta \log(U).$$

Then, X is an exponential distribution with parameter β.

Because the transformation is given by $u = \exp(-x/\beta)$, the Jacobian is:

$$J = \frac{\mathrm{d}u}{\mathrm{d}x} = -\frac{1}{\beta}\exp\left(-\frac{1}{\beta}x\right).$$

By transforming the variables, the density function of X is represented as:

$$f(x) = |J| f_u\left(\exp(-\frac{1}{\beta}x)\right) = \frac{1}{\beta}\exp\left(-\frac{1}{\beta}x\right),$$

where $f(\cdot)$ and $f_u(\cdot)$ denote the probability density functions of X and U, respectively.

Note that $0 < x < \infty$ because of $x = -\beta\log(u)$ and $0 < u < 1$.

Thus, the exponential distribution with parameter β is obtained from the uniform random draw between zero and one.

exprnd(ix,iy,beta,rn)

```
      subroutine exprnd(ix,iy,beta,rn)
c
c  Use "exprnd(ix,iy,beta,rn)"
c  together with "urnd(ix,iy,rn)".
c
c  Input:
c    ix, iy:  Seeds
c    beta: Parameter
c  Output:
c    rn: Exponential Random Draw
c        with Parameter beta
c
       call urnd(ix,iy,rn1)
       rn=-beta*log(rn1)
       return
       end
```

exprnd(ix,iy,beta,rn) should be used together with urnd(ix,iy,rn).

When $\beta = 2$, the exponential distribution reduces to the chi-square distribution with

2 degrees of freedom.

Gamma Distribution: $G(\alpha,\beta)$**:** The gamma distribution with parameters α and β, denoted by $G(\alpha,\beta)$, is represented as follows:

$$f(x) = \begin{cases} \dfrac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}}, & \text{for } 0 < x < \infty, \\ 0, & \text{otherwise,} \end{cases}$$

for $\alpha > 0$ and $\beta > 0$, where α is called a **shape parameter** and β denotes a scale parameter.

$\Gamma(\cdot)$ is called the **gamma function**, which is the following function of α:

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x}\, dx.$$

The gamma function has the following features:

$$\Gamma(\alpha+1) = \alpha\Gamma(\alpha), \qquad \Gamma(1) = 1, \qquad \Gamma\left(\frac{1}{2}\right) = 2\Gamma\left(\frac{3}{2}\right) = \sqrt{\pi}.$$

Mean, variance and the moment-generating function are given by:

$$\mathrm{E}(X) = \alpha\beta, \qquad \mathrm{V}(X) = \alpha\beta^2, \qquad \phi(\theta) = \frac{1}{(1-\beta\theta)^{\alpha}}.$$

The gamma distribution with $\alpha = 1$ is equivalent to the exponential distribution shown in Section 9.6.2.

This fact is easily checked by comparing both moment-generating functions.

Now, utilizing the uniform random variable, the gamma distribution with parameters α and β are derived as follows.

The derivation shown in this section deals with the case where α is a positive integer, i.e., $\alpha = 1, 2, 3, \cdots$.

The random variables Z_1, Z_2, $\cdots$, Z_α are assumed to be mutually independently distributed as exponential random variables with parameter β, which are shown in Section 9.6.2.

Define $X = \sum_{i=1}^{\alpha} Z_i$.

Then, X has distributed as a gamma distribution with parameters α and β, where α should be an integer, which is proved as follows:

$$\phi_x(\theta) = \mathrm{E}(e^{\theta X}) = \mathrm{E}(e^{\theta \sum_{i=1}^{\alpha} Z_i}) = \prod_{i=1}^{\alpha} \mathrm{E}(e^{\theta Z_i}) = \prod_{i=1}^{\alpha} \phi_i(\theta) = \prod_{i=1}^{\alpha} \frac{1}{1-\beta\theta}$$
$$= \frac{1}{(1-\beta\theta)^{\alpha}},$$

where $\phi_x(\theta)$ and $\phi_i(\theta)$ represent the moment-generating functions of X and Z_i, respectively.

Thus, sum of the α exponential random variables yields the gamma random variable with parameters α and β.

Therefore, the source code which generates gamma random numbers is shown in `gammarnd(ix,iy,alpha,beta,rn)`.

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

```
      subroutine gammarnd(ix,iy,alpha,beta,rn)
c
c  Use "gammarnd(ix,iy,alpha,beta,rn)"
c  together with "exprnd(ix,iy,beta,rn)"
c            and  "urnd(ix,iy,rn)".
c
c  Input:
c    ix, iy:   Seeds
c    alpha:    Shape Parameter (which should be an integer)
c    beta:     Scale Parameter
c  Output:
c    rn: Gamma Random Draw with alpha and beta
c
       rn=0.0
        do 1 i=1,nint(alpha)
       call exprnd(ix,iy,beta,rn1)
     1 rn=rn+rn1
       return
       end
```

gammarnd(ix,iy,alpha,beta,rn) is utilized together with urnd(ix,iy,rn) and exprnd(ix,iy,rn).

As pointed out above, α should be an integer in the source code.

When α is large, we have serious problems computationally in the above algorithm, because α exponential random draws have to be generated to obtain one gamma random draw with parameters α and β.

When $\alpha = k/2$ and $\beta = 2$, the gamma distribution reduces to the chi-square distribution with k degrees of freedom.

Chi-Square Distribution: $\chi^2(k)$: The chi-square distribution with k degrees of freedom, denoted by $\chi^2(k)$, is written as follows:

$$f(x) = \begin{cases} \dfrac{1}{2^{k/2}\Gamma(\frac{k}{2})} x^{\frac{k}{2}-1} e^{-\frac{1}{2}x}, & \text{for } 0 < x < \infty, \\ 0, & \text{otherwise,} \end{cases}$$

where k is a positive integer.

The chi-square distribution is equivalent to the gamma distribution with $\beta = 2$ and $\alpha = k/2$.

The chi-square distribution with $k = 2$ reduces to the exponential distribution with $\beta = 2$, shown in Section 9.6.2.

Mean, variance and the moment-generating function are given by:

$$\mathrm{E}(X) = k, \qquad \mathrm{V}(X) = 2k, \qquad \phi(\theta) = \frac{1}{(1-2\theta)^{k/2}}.$$

***F* Distribution: $F(m, n)$:** The F distribution with m and n degrees of freedom, denoted by $F(m, n)$, is represented as:

$$f(x) = \begin{cases} \dfrac{\Gamma(\frac{m+n}{2})}{\Gamma(\frac{m}{2})\Gamma(\frac{n}{2})}\left(\dfrac{m}{n}\right)^{\frac{m}{2}} x^{\frac{m}{2}-1}\left(1+\dfrac{m}{n}x\right)^{-\frac{m+n}{2}}, & \text{for } 0 < x < \infty, \\ 0, & \text{otherwise,} \end{cases}$$

where m and n are positive integers.

Mean and variance are given by:

$$\mathrm{E}(X) = \frac{n}{n-2}, \qquad \text{for } n > 2,$$
$$\mathrm{V}(X) = \frac{2n^2(m+n-2)}{m(n-2)^2(n-4)}, \qquad \text{for } n > 4.$$

The moment-generating function of F distribution does not exist.

One F random variable is derived from two chi-square random variables.

Suppose that U and V are independently distributed as chi-square random variables, i.e., $U \sim \chi^2(m)$ and $V \sim \chi^2(n)$.

Then, it is shown that $X = \dfrac{U/m}{V/n}$ has a F distribution with (m, n) degrees of freedom.

***t* Distribution: $t(k)$:** The t distribution (or Student's t distribution) with k degrees of freedom, denoted by $t(k)$, is given by:

$$f(x) = \frac{\Gamma(\frac{k+1}{2})}{\Gamma(\frac{k}{2})} \frac{1}{\sqrt{k\pi}} \left(1 + \frac{x^2}{k}\right)^{-\frac{k+1}{2}},$$

for $-\infty < x < \infty$, where k does not have to be an integer but conventionally it is a positive integer.

When k is small, the t distribution has fat tails.

The t distribution with $k = 1$ is equivalent to the Cauchy distribution.

As k goes to infinity, the t distribution approaches the standard normal distribution, i.e., $t(\infty) = N(0, 1)$, which is easily shown by using the definition of e, i.e.,

$$\left(1 + \frac{x^2}{k}\right)^{-\frac{k+1}{2}} = \left(1 + \frac{1}{h}\right)^{-\frac{hx^2+1}{2}} = \left((1 + \frac{1}{h})^h\right)^{-\frac{1}{2}x^2}\left(1 + \frac{1}{h}\right)^{-\frac{1}{2}} \longrightarrow e^{-\frac{1}{2}x^2},$$

where $h = k/x^2$ is set and h goes to infinity (equivalently, k goes to infinity).

Thus, a kernel of the t distribution is equivalent to that of the standard normal distribution.

Therefore, it is shown that as k is large the t distribution approaches the standard normal distribution.

Mean and variance of the t distribution with k degrees of freedom are obtained as:

$$\mathrm{E}(X) = 0, \qquad \text{for } k > 1,$$
$$\mathrm{V}(X) = \frac{k}{k-2}, \qquad \text{for } k > 2.$$

In the case of the t distribution, the moment-generating function does not exist, because all the moments do not necessarily exist.

For the t random variable X, we have the fact that $\mathrm{E}(X^p)$ exists when p is less than k. Therefore, all the moments exist only when k is infinity.

One t random variable is obtained from chi-square and standard normal random variables.

Suppose that $Z \sim N(0,1)$ is independent of $U \sim \chi^2(k)$.

Then, $X = Z/\sqrt{U/k}$ has a t distribution with k degrees of freedom.

Marsaglia (1984) gives a very fast algorithm for generating t random draws, which is

based on a transformed acceptance/rejection method, which will be discussed later.

9.6.3 Inverse Transform Method

In Section 9.6.2, we have introduced the probability density functions which can be derived by transforming the uniform random variables between zero and one.
In this section, the probability density functions obtained by the inverse transform method are presented and the corresponding random number generators are shown.
The inverse transform method is represented as follows.
Let X be a random variable which has a cumulative distribution function $F(\cdot)$.
When $U \sim U(0,1)$, $F^{-1}(U)$ is equal to X.
The proof is obtained from the following fact:

$$P(X < x) = P(F^{-1}(U) < x) = P(U < F(x)) = F(x).$$

In other words, let u be a random draw of U, where $U \sim U(0,1)$, and $F(\cdot)$ be a

distribution function of X.

When we perform the following inverse transformation:

$$x = F^{-1}(u),$$

x implies the random draw generated from $F(\cdot)$.

The inverse transform method shown above is useful when $F(\cdot)$ can be computed easily and the inverse distribution function, i.e., $F^{-1}(\cdot)$, has a closed form.

For example, recall that $F(\cdot)$ cannot be obtained explicitly in the case of the normal distribution because the integration is included in the normal cumulative distribution (conventionally we approximate the normal cumulative distribution when we want to evaluate it).

If no closed form of $F^{-1}(\cdot)$ is available but $F(\cdot)$ is still computed easily, an iterative method such as the Newton-Raphson method can be applied.

Define $k(x) = F(x) - u$.

The first order Taylor series expansion around $x = x^*$ is:

$$0 = k(x) \approx k(x^*) + k'(x^*)(x - x^*).$$

Then, we obtain:

$$x = x^* - \frac{k(x^*)}{k'(x^*)} = x^* - \frac{F(x^*) - u}{f(x^*)}.$$

Replacing x and x^* by $x^{(i)}$ and $x^{(i-1)}$, we have the following iteration:

$$x^{(i)} = x^{(i-1)} - \frac{F(x^{(i-1)}) - u}{f(x^{(i-1)})},$$

for $i = 1, 2, \cdots$.

The convergence value of $x^{(i)}$ is taken as a solution of equation $u = F(x)$.

Thus, based on u, a random draw x is derived from $F(\cdot)$.

However, we should keep in mind that this procedure takes a lot of time computationally, because we need to repeat the convergence computation shown above as many

times as we want to generate.

9.6.4 Using $U(0, 1)$: Discrete Type

In Sections 9.6.2 and 9.6.3, the random number generators from continuous distributions are discussed, i.e., the transformation of variables in Section 9.6.2 and the inverse transform method in Section 9.6.3 are utilized.

Based on the uniform random draw between zero and one, in this section we deal with some discrete distributions and consider generating their random numbers.

As a representative random number generation method, we can consider utilizing the inverse transform method in the case of discrete random variables.

Suppose that a discrete random variable X can take $x_1, x_2, \cdots, x_n$, where the probability which X takes x_i is given by $f(x_i)$, i.e., $P(X = x_i) = f(x_i)$.

Generate a uniform random draw u, which is between zero and one.

Consider the case where we have $F(x_{i-1}) \leq u < F(x_i)$, where $F(x_i) = P(X \leq x_i)$ and $F(x_0) = 0$.

Then, the random draw of X is given by x_i.

References

Ahrens, J.H. and Dieter, U., 1980, "Sampling from Binomial and Poisson Distributions: A Method with Bounded Computation Times," *Computing*, Vol.25, pp.193 – 208.

Ahrens, J.H. and Dieter, U., 1988, "Efficient, Table-Free Sampling Methods for the Exponential, Cauchy and Normal Distributions," *Communications of the ACM*, Vol.31, pp.1330 – 1337.

Bays, C. and Durham, S.D., 1976, "Improving a Poor Random Number Generator,"

ACM Transactions on Mathematical Software, Vol.2, pp.59 – 64.

Box, G.E.P. and Muller, M.E., 1958, "A Note on the Generation of Random Normal Deviates," *Annals of Mathematical Statistics*, Vol.29, No.2, pp.610 – 611.

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

De Matteis, A. and Pagnutti, S., 1993, "Long-Range Correlation Analysis of the Wichmann-Hill Random Number Generator," *Statistics and Computing*, Vol.3, pp.67 – 70.

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

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

Hastings, C., 1955, *Approximations for Digital Computers*, Princeton University Press.

Hill, I.D and Pike, A.C., 1967, "Algorithm 2999: Chi-Squared Integral," *Communications of the ACM*, Vol.10, pp.243 – 244.

Hogg, R.V. and Craig, A.T., 1995, *Introduction to Mathematical Statistics* (Fifth Edition), Prentice Hall.

Johnson, N.L. and Kotz, S., 1970a, *Continuous Univariate Distributions*, Vol.1, John Wiley & Sons.

Johnson, N.L. and Kotz, S., 1970b, *Continuous Univariate Distributions*, Vol.2, John Wiley & Sons.

Kachitvichyanukul, V. and Schmeiser, B., 1985, "Computer Generation of Hypergeometric Random Variates," *Journal of Statistical Computation and Simulation*, Vol.22, pp.127 – 145.

Kennedy, Jr. W.J. and Gentle, J.E., 1980, *Statistical Computing* (Statistics: Textbooks and Monographs, Vol.33), Marcel Dekker.

Knuth, D.E., 1981, *The Art of Computer Programming, Vol.2: Seminumerical Algorithms* (Second Edition), Addison-Wesley, Reading, MA.

Kotz, S. and Johnson, N.L., 1982, *Encyclopedia of Statistical Sciences*, Vol.2, pp.188 – 193, John Wiley & Sons.

Kotz, S., Balakrishman, N. and Johnson, N.L., 2000a, *Univariate Discrete Distributions* (Second Edition), John Wiley & Sons.

Kotz, S., Balakrishman, N. and Johnson, N.L., 2000b, *Continuous Univariate Distributions, Vol.1* (Second Edition), John Wiley & Sons.

Kotz, S., Balakrishman, N. and Johnson, N.L., 2000c, *Continuous Univariate Distributions, Vol.2* (Second Edition), John Wiley & Sons.

Kotz, S., Balakrishman, N. and Johnson, N.L., 2000d, *Discrete Multivariate Distributions* (Second Edition), John Wiley & Sons.

Kotz, S., Balakrishman, N. and Johnson, N.L., 2000e, *Continuous Multivariate Distributions, Vol.1* (Second Edition), John Wiley & Sons.

Law, A.M. and Kelton, W.D., 2000, *Simulation Modeling and Analysis* (Third Edition), McGraw-Hill Higher Education.

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

L'Ecuyer, P., 1990, "Random Numbers for Simulation," *Communications of the ACM*, Vol.33, No.10, pp.85 – 97.

L'Ecuyer, P., 1998, "Random Number Generation," in *Handbook of Simulation*, Chap. 4, edited by Banks, J., pp.93 – 137, John Wiley & Sons.

Marsaglia, G., 1964, “Generating a Variable from the Tail of the Normal Distribution,” *Technometrics*, Vol.6, pp.101 – 102.

Marsaglia, G., MacLaren, M.D. and Bray, T.A., 1964, “A Fast Method for Generating Normal Random Variables,” *Communications of the ACM*, Vol.7, pp.4 – 10.

Marsaglia, G. and Zaman, A., 1994, “Rapid Evaluation of the Inverse of the Normal Distribution Function,” *Statistics and Probability Letters*, Vol.19, No.2, pp.259 – 266.

Niederreiter, H., 1992, *Random Number Generation and Quasi-Monte Carlo Methods* (CBMS-NFS Regional Conference Series in Applied Mathematics 63), Society for Industrial and Applied Mathematics.

Odeh, R.E. and Evans, J.O., 1974, “Algorithm AS 70: The Percentage Points of the Normal Distribution,” *Applied Statistics*, Vol.23, No.1, pp.96 – 97.

Odell, P.L. and Feiveson, A.H., 1966, “A Numerical Procedure to Generate a Simple

Covariance Matrix," *Journal of the American Statistical Association*, Vol.61, No.313, pp.199 – 203.

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.

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

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

Ross, S.M., 1997, *Simulation* (Second Edition), Academic Press.

Rubinstein, R.Y., 1981, *Simulation and the Monte Carlo Method*, John Wiley & Sons.

Rubinstein, R.Y. and Melamed, B., 1998, *Modern Simulation and Modeling*, John Wiley & Sons.

Schmeiser, B. and Kachitvichyanukul, V., 1990, "Noninverse Correlation Induction: Guidelines for Algorithm Development," *Journal of Computational and Applied Mathematics*, Vol.31, pp.173 – 180.

Shibata, Y., 1981, *Normal Distribution* (in Japanese), Tokyo University Press.

Smith, W.B and Hocking, R.R., 1972, "Algorithm AS53: Wishart Variate Generator," *Applied Statistics*, Vol.21, No.3, pp.341 – 345.

Stadlober, E., 1990, "The Ratio of Uniforms Approach for Generating Discrete Random Variates," *Journal of Computational and Applied Mathematics*, Vol.31, pp.181 – 189.

Takeuchi, K., 1989, *Dictionary of Statistics* (in Japanese), Toyo-Keizai.

Thompson, J.R., 2000, *Simulation: A Modeler's Approach*, Jhon Wiley & Sons.

Wichmann, B.A. and Hill, I.D., 1982, "Algorithm AS183: An Efficient and Portable Pseudo-random Number Generator," *Applied Statistics*, Vol.31, No.2, pp.188 – 190.

Wichmann, B.A. and Hill, I.D., 1984, "Correction of Algorithm AS183: An Efficient and Portable Pseudo-random Number Generator," *Applied Statistics*, Vol.33, No.2, p.123.

Zellner, A., 1971, *An Introduction to Bayesian Inference in Econometrics*, John Wiley & Sons.

9.7 Sampling Method II: Random Number Generation

9.7.1 Rejection Sampling (棄却法)

We want to generate random draws from $f(x)$, called the **target density (目的密度)**, but we consider the case where it is hard to sample from $f(x)$.

Now, suppose that it is easy to generate a random draw from another density $f_*(x)$, called the **sampling density (サンプリング密度)** or **proposal density (提案密度)**. In this case, random draws of X from $f(x)$ are generated by utilizing the random draws sampled from $f_*(x)$.

Let x be the the random draw of X generated from $f(x)$.

Suppose that $q(x)$ is equal to the ratio of the target density and the sampling density, i.e.,

$$q(x) = \frac{f(x)}{f_*(x)}. \tag{19}$$

Then, the target density is rewritten as:

$$f(x) = q(x) f_*(x).$$

Based on $q(x)$, the acceptance probability is obtained.

Depending on the structure of the acceptance probability, we have three kinds of sam-

pling techniques, i.e., **rejection sampling (棄却法)** in this section, **importance resampling (重点的リサンプリング法)** in Section 9.7.2 and the **Metropolis-Hastings algorithm (メトロポリスーハスティング・アルゴリズム)** in Section 9.7.4.

See Liu (1996) for a comparison of the three sampling methods.

Thus, to generate random draws of x from $f(x)$, the functional form of $q(x)$ should be known and random draws have to be easily generated from $f_*(x)$.

In order for rejection sampling to work well, the following condition has to be satisfied:

$$q(x) = \frac{f(x)}{f_*(x)} < c,$$

where c is a fixed value.

That is, $q(x)$ has an upper limit.

As discussed below, $1/c$ is equivalent to the acceptance probability.

If the acceptance probability is large, rejection sampling computationally takes a lot

of time.

Under the condition $q(x) < c$ for all x, we may minimize c.

That is, since we have $q(x) < \sup_x q(x) \leq c$, we may take the supremum of $q(x)$ for c.

Thus, in order for rejection sampling to work efficiently, c should be the supremum of $q(x)$ with respect to x, i.e., $c = \sup_x q(x)$.

Let x^* be the random draw generated from $f_*(x)$, which is a candidate of the random draw generated from $f(x)$.

Define $\omega(x)$ as:

$$\omega(x) = \frac{q(x)}{\sup_z q(z)} = \frac{q(x)}{c},$$

which is called the **acceptance probability (採択確率)**.

Note that we have $0 \leq \omega(x) \leq 1$ when $\sup_z q(z) = c < \infty$.

The supremum $\sup_z q(z) = c$ has to be finite.

This condition is sometimes too restrictive, which is a crucial problem in rejection

sampling.

A random draw of X is generated from $f(x)$ in the following way:

(i) Generate x^* from $f_*(x)$ and compute $\omega(x^*)$.

(ii) Set $x = x^*$ with probability $\omega(x^*)$ and go back to (i) otherwise.

In other words, generating u from a uniform distribution between zero and one, take $x = x^*$ if $u \leq \omega(x^*)$ and go back to (i) otherwise.

The above random number generation procedure can be justified as follows.

Let U be the uniform random variable between zero and one, X be the random variable generated from the target density $f(x)$,

X^* be the random variable generated from the sampling density $f_*(x)$, and x^* be the realization (i.e., the random draw) generated from the sampling density $f_*(x)$.

Consider the probability $P(X \le x | U \le \omega(x^*))$, which should be the cumulative distribution of X, $F(x)$, from Step (ii).

The probability $P(X \le x | U \le \omega(x^*))$ is rewritten as follows:

$$P(X \le x | U \le \omega(x^*)) = \frac{P(X \le x, U \le \omega(x^*))}{P(U \le \omega(x^*))},$$

where the numerator is represented as:

$$\begin{aligned} P(X \le x, U \le \omega(x^*)) &= \int_{-\infty}^{x} \int_{0}^{\omega(t)} f_{u,*}(u,t)\,\mathrm{d}u\,\mathrm{d}t = \int_{-\infty}^{x} \int_{0}^{\omega(t)} f_u(u) f_*(t)\,\mathrm{d}u\,\mathrm{d}t \\ &= \int_{-\infty}^{x} \Big(\int_{0}^{\omega(t)} f_u(u)\,\mathrm{d}u \Big) f_*(t)\,\mathrm{d}t = \int_{-\infty}^{x} \Big(\int_{0}^{\omega(t)} \mathrm{d}u \Big) f_*(t)\,\mathrm{d}t \\ &= \int_{-\infty}^{x} \Big[u \Big]_{0}^{\omega(t)} f_*(t)\,\mathrm{d}t = \int_{-\infty}^{x} \omega(t) f_*(t)\,\mathrm{d}t = \int_{-\infty}^{x} \frac{q(t)}{c} f_*(t)\,\mathrm{d}t = \frac{F(x)}{c}, \end{aligned}$$

and the denominator is given by:

$$P(U \le \omega(x^*)) = P(X \le \infty, U \le \omega(x^*)) = \frac{F(\infty)}{c} = \frac{1}{c}.$$

In the numerator, $f_{u,*}(u, x)$ denotes the joint density of random variables U and X^*. Because the random draws of U and X^* are independently generated in Steps (i) and (ii) we have $f_{u,*}(u, x) = f_u(u) f_*(x)$, where $f_u(u)$ and $f_*(x)$ denote the marginal density of U and that of X^*.

The density function of U is given by $f_u(u) = 1$, because the distribution of U is assumed to be uniform between zero and one.

Thus, the first four equalities are derived.

Furthermore, in the seventh equality of the numerator, since we have:

$$\omega(x) = \frac{q(x)}{c} = \frac{f(x)}{c f_*(x)},$$

$\omega(x) f_*(x) = f(x)/c$ is obtained.

Finally, substituting the numerator and denominator shown above, we have the following equality:

$$P(X \leq x | U \leq \omega(x^*)) = F(x).$$

Thus, the rejection sampling method given by Steps (i) and (ii) is justified.

The rejection sampling method is the most efficient sampling method in the sense of precision of the random draws, because using rejection sampling we can generate mutually independently distributed random draws.

However, for rejection sampling we need to obtain the c which is greater than or equal to the supremum of $q(x)$.

If the supremum is infinite, i.e., if c is infinite, $\omega(x)$ is zero and accordingly the candidate x^* is never accepted in Steps (i) and (ii).

Moreover, as for another remark, note as follows.

Let N_R be the average number of the rejected random draws.

We need $(1 + N_R)$ random draws in average to generate one random number from $f(x)$.

In other words, the acceptance rate is given by $1/(1 + N_R)$ in average, which is equal

to $1/c$ in average because of $P(U \leq \omega(x^*)) = 1/c$.

Therefore, to obtain one random draw from $f(x)$, we have to generate $(1+N_R)$ random draws from $f_*(x)$ in average.

See, for example, Boswell, Gore, Patil and Taillie (1993), O'Hagan (1994) and Geweke (1996) for rejection sampling.

To examine the condition that $\omega(x)$ is greater than zero, i.e., the condition that the supremum of $q(x)$ exists, consider the case where $f(x)$ and $f_*(x)$ are distributed as $N(\mu, \sigma^2)$ and $N(\mu_*, \sigma_*^2)$, respectively.

$q(x)$ is given by:

$$q(x) = \frac{f(x)}{f_*(x)} = \frac{(2\pi\sigma^2)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)}{(2\pi\sigma_*^2)^{-1/2} \exp\left(-\frac{1}{2\sigma_*^2}(x-\mu_*)^2\right)}$$

$$= \frac{\sigma_*}{\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2 + \frac{1}{2\sigma_*^2}(x-\mu_*)^2\right)$$

$$= \frac{\sigma_*}{\sigma} \exp\left(-\frac{1}{2}\frac{\sigma_*^2 - \sigma^2}{\sigma^2\sigma_*^2}\left(x - \frac{\mu\sigma_*^2 - \mu_*\sigma^2}{\sigma_*^2 - \sigma^2}\right)^2 + \frac{1}{2}\frac{(\mu - \mu_*)^2}{\sigma_*^2 - \sigma^2}\right).$$

If $\sigma_*^2 < \sigma^2$, $q(x)$ goes to infinity as x is large.

In the case of $\sigma_*^2 > \sigma^2$, the supremum of $q(x)$ exists, which condition implies that $f_*(x)$ should be more broadly distributed than $f(x)$.

In this case, the supremum is obtained as:

$$c = \sup_x q(x) = \frac{\sigma_*}{\sigma} \exp\left(\frac{1}{2}\frac{(\mu - \mu_*)^2}{\sigma_*^2 - \sigma^2}\right).$$

When $\sigma^2 = \sigma_*^2$ and $\mu = \mu_*$, we have $q(x) = 1$, which implies $\omega(x) = 1$.

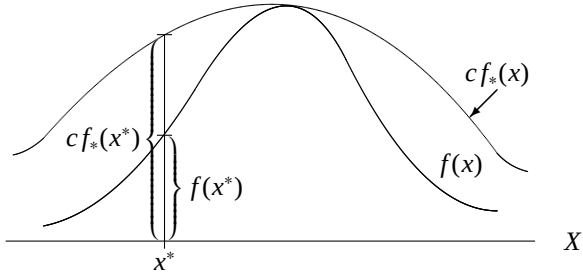
That is, a random draw from the sampling density $f_*(x)$ is always accepted as a random draw from the target density $f(x)$, where $f(x)$ is equivalent to $f_*(x)$ for all x.

If $\sigma^2 = \sigma_*^2$ and $\mu \neq \mu_*$, the supremum of $q(x)$ does not exists.

Accordingly, the rejection sampling method does not work in this case.

From the definition of $\omega(x)$, we have the inequality $f(x) \leq c f_*(x)$.

Figure 1: Rejection Sampling



$cf_*(x)$ and $f(x)$ are displayed in Figure 1.

The ratio of $f(x^*)$ and $cf_*(x^*)$ corresponds to the acceptance probability at x^*, i.e., $\omega(x^*)$.

Thus, for rejection sampling, $cf_*(x)$ has to be greater than or equal to $f(x)$ for all x, which implies that the sampling density $f_*(x)$ needs to be more widely distributed

than the target density $f(x)$.

Finally, note that the above discussion holds without any modification even though $f(x)$ is a kernel of the target density, i.e., even though $f(x)$ is proportional to the target density, because the constant term is canceled out between the numerator and denominator (remember that $\omega(x) = q(x)/\sup_z q(z)$).

Normal Distribution: $N(0, 1)$**:** First, denote the half-normal distribution by:

$$f(x) = \begin{cases} \dfrac{2}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, & \text{for } 0 \le x < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

The half-normal distribution above corresponds to the positive part of the standard normal probability density function.

Using rejection sampling, we consider generating standard normal random draws based on the half-normal distribution.

We take the sampling density as the exponential distribution:

$$f_*(x) = \begin{cases} \lambda e^{-\lambda x}, & \text{for } 0 \le x < \infty, \\ 0, & \text{otherwise,} \end{cases}$$

where $\lambda > 0$. Since $q(x)$ is defined as $q(x) = f(x)/f_*(x)$, the supremum of $q(x)$ is given by:

$$c = \sup_x q(x) = \frac{2}{\lambda\sqrt{2\pi}} e^{\frac{1}{2}\lambda^2}.$$

which depends on parameter λ.

Remember that $P(U \le \omega(x^*)) = 1/c$ corresponds to the acceptance probability.

Since we need to increase the acceptance probability to reduce computational time, we want to obtain the λ which minimizes $\sup_x q(x)$ with respect to λ.

Solving the minimization problem, $\lambda = 1$ is obtained.

Substituting $\lambda = 1$, the acceptance probability $\omega(x)$ is derived as:

$$\omega(x) = e^{-\frac{1}{2}(x-1)^2},$$

for $0 < x < \infty$.

Remember that $-\log U$ has an exponential distribution with $\lambda = 1$ when $U \sim U(0, 1)$. Therefore, the algorithm is represented as follows.

(i) Generate two independent uniform random draws u_1 and u_2 between zero and one.

(ii) Compute $x^* = -\log u_2$, which indicates the exponential random draw generated from the target density $f_*(x)$.

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

x in Step (iii) yields a random draw from the half-normal distribution.

To generate a standard normal random draw utilizing the half-normal random draw above, we may put the positive or negative sign randomly with x.
Therefore, the following Step (iv) is additionally put.

(iv) Generate a uniform random draw u_3 between zero and one, and set $z = x$ if $u_3 \leq 1/2$ and $z = -x$ otherwise.

z gives us a standard normal random draw.
Note that the number of iteration in Step (iii) is given by $c = \sqrt{2e/\pi} \approx 1.3155$ in average, or equivalently, the acceptance probability in Step (iii) is $1/c \approx 0.7602$.
The source code for this standard normal random number generator is shown in `snrnd6(ix,iy,rn)`.

snrnd6(ix,iy,rn)

```
1:        subroutine snrnd6(ix,iy,rn)
2:  c
3:  c  Use "snrnd6(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:      1 call urnd(ix,iy,rn1)
12:        call urnd(ix,iy,rn2)
13:        y=-log(rn2)
14:        if( -2.*log(rn1).lt.(y-1.)**2 ) go to 1
15:        call urnd(ix,iy,rn3)
16:          if(rn3.le.0.5) then
17:        rn= y
18:          else
19:        rn=-y
20:          endif
21:        return
22:        end
```

Note that `snrnd6(ix,iy,rn)` should be used together with `urnd(ix,iy,rn)`. Thus, utilizing rejection sampling, we have the standard normal random number generator, which is based on the half-normal distribution.

Gamma Distribution: $G(\alpha, 1)$ **for** $0 < \alpha \leq 1$ **and** $1 < \alpha$**:** In this section, utilizing rejection sampling we show an example of generating random draws from the gamma distribution with parameters α and $\beta = 1$, i.e., $G(\alpha, 1)$.

When $X \sim G(\alpha, 1)$, the density function of X is given by:

$$f(x) = \begin{cases} \dfrac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}, & \text{for } 0 < x < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

Ahrens and Dieter (1974) consider the case of $0 < \alpha \leq 1$, which is discussed in this section.

The case of $\alpha > 1$ will be discussed later.

Using the rejection sampling, the composition method and the inverse transform method, we consider generating random draws from $G(\alpha, 1)$ for $0 < \alpha \leq 1$.

The sampling density is taken as:

$$f_*(x) = \frac{e}{\alpha + e}\alpha x^{\alpha-1} I_1(x) + \frac{\alpha}{\alpha + e} e^{-x+1} I_2(x),$$

where both $I_1(x)$ and $I_2(x)$ denote the indicator functions defined as:

$$I_1(x) = \begin{cases} 1, & \text{if } 0 < x \leq 1, \\ 0, & \text{otherwise,} \end{cases} \qquad I_2(x) = \begin{cases} 1, & \text{if } 1 < x, \\ 0, & \text{otherwise.} \end{cases}$$

Random number generation from the sampling density above utilizes the composition method and the inverse transform method.

The cumulative distribution related to $f_*(x)$ is given by:

$$F_*(x) = \begin{cases} \frac{e}{\alpha + e} x^{\alpha}, & \text{if } 0 < x \leq 1, \\ \frac{e}{\alpha + e} + \frac{\alpha}{\alpha + e}(1 - e^{-x+1}), & \text{if } x > 1. \end{cases}$$

Note that $0 < \alpha \leq 1$ is required because the sampling density for $0 < x \leq 1$ has to satisfy the property that the integration is equal to one.

The acceptance probability $\omega(x) = q(x)/\sup_z q(z)$ for $q(x) = f(x)/f_*(x)$ is given by:

$$\omega(x) = e^{-x} I_1(x) + x^{\alpha-1} I_2(x).$$

Moreover, the mean number of trials until success, i.e., $c = \sup_z q(z)$ is represented as:

$$c = \frac{\alpha + e}{\alpha e \Gamma(\alpha)},$$

which depends on α and is not greater than 1.39.

Note that $q(x)$ takes a maximum value at $x = 1$.

The random number generation procedure is given by:

(i) Generate a uniform random draw u_1 from $U(0, 1)$, and set $x^* = ((\alpha/e + 1)u_1)^{1/\alpha}$ if $u_1 \leq e/(\alpha + e)$ and $x^* = -\log((1/e + 1/\alpha)(1 - u_1))$ if $u_1 > e/(\alpha + e)$.

(ii) Obtain $\omega(x^*) = e^{-x^*}$ if $u_1 \leq e/(\alpha + e)$ and $\omega(x^*) = x^{*\alpha-1}$ if $u_1 > e/(\alpha + e)$.

(iii) Generate a uniform random draw u_2 from $U(0, 1)$, and set $x = x^*$ if $u_2 \leq \omega(x^*)$ and return to (i) otherwise.

In Step (i) a random draw x^* from $f_*(x)$ can be generated by the inverse transform method discussed in Section 9.6.3.

gammarnd2(ix,iy,alpha,rn)

```
1:       subroutine gammarnd2(ix,iy,alpha,rn)
2: C
3: C  Use "gammarnd2(ix,iy,alpha,rn)"
4: C  together with "urnd(ix,iy,rn)".
5: C
6: C  Input:
7: C    ix, iy: Seeds
8: C    alpha:  Shape Parameter (0<alpha \le 1)
9: C  Output:
10: C   rn: Gamma Random Draw
11: C       with Parameters alpha and beta=1
12: C
```

```
      e=2.71828182845905
    1 call urnd(ix,iy,rn0)
      call urnd(ix,iy,rn1)
            if( rn0.le.e/(alpha+e) ) then
      rn=( (alpha+e)*rn0/e )**(1./alpha)
      if( rn1.gt.e**(-rn) ) go to 1
            else
      rn=-log((alpha+e)*(1.-rn0)/(alpha*e))
      if( rn1.gt.rn**(alpha-1.) ) go to 1
            endif
      return
      end
```

Note that `gammarnd2(ix,iy,alpha,rn)` should be used with `urnd(ix,iy,rn)`.

In `gammarnd2(ix,iy,alpha,rn)`, the case of $0 < \alpha \leq 1$ has been shown.

Now, using rejection sampling, the case of $\alpha > 1$ is discussed in Cheng (1977, 1998).

The sampling density is chosen as the following cumulative distribution:

$$F_*(x) = \begin{cases} \dfrac{x^\lambda}{\delta + x^\lambda}, & \text{for } x > 0, \\ 0, & \text{otherwise,} \end{cases}$$

which is sometimes called the **log-logistic distribution**.

Then, the probability density function, $f_*(x)$, is given by:

$$f_*(x) = \begin{cases} \dfrac{\lambda\delta x^{\lambda-1}}{(\alpha + x^\lambda)^2}, & \text{for } x > 0, \\ 0, & \text{otherwise.} \end{cases}$$

By the inverse transform method, the random draw from $f_*(x)$, denoted by x, is generated as follows:

$$x = \left(\frac{\delta u}{1-u}\right)^{1/\lambda},$$

where u denotes the uniform random draw generated from $U(0, 1)$.

For the two parameters, $\lambda = \sqrt{2\alpha - 1}$ and $\delta = \alpha^{\lambda}$ are chosen, taking into account minimizing $c = \sup_x q(x) = \sup_x f(x)/f_*(x)$ with respect to δ and λ (note that λ and δ are approximately taken, since it is not possible to obtain the explicit solution of δ and λ).

Then, the number of rejections in average is given by:

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

which is computed as:

1.47 when $\alpha = 1$, 1.25 when $\alpha = 2$, 1.17 when $\alpha = 5$,

1.15 when $\alpha = 10$, 1.13 when $\alpha = \infty$.

Thus, the average number of rejections is quite small for all α.

The random number generation procedure is given by:

(i) Set $a = 1/\sqrt{2\alpha - 1}$, $b = \alpha - \log 4$ and $c = \alpha + \sqrt{2\alpha - 1}$.

(ii) Generate two uniform random draws u_1 and u_2 from $U(0, 1)$.

(iii) Set $y = a \log \dfrac{u_1}{1 - u_1}$, $x^* = \alpha e^y$, $z = u_1^2 u_2$ and $r = b + cy - x$.

(iv) Take $x = x^*$ if $r \geq \log z$ and return to (ii) otherwise.

To avoid evaluating the logarithm in Step (iv), we put Step (iii)' between Steps (iii) and (iv), which is as follows:

(iii)' Take $x = x^*$ if $r \geq 4.5z - d$ and go to (iv) otherwise.

d is defined as $d = 1 + \log 4.5$, which has to be computed in Step (i).

Note that we have the relation: $\theta z - (1 + \log \theta) \geq \log z$ for all $z > 0$ and any given $\theta > 0$, because $\log z$ is a concave function of z. According to Cheng (1977), the choice of θ is not critical and the suggested value is $\theta = 4.5$, irrespective of α.

The source code for Steps (i) – (iv) and (iii)' is given by `gammarnd3(ix,iy,alpha,rn)`.

```
      subroutine gammarnd3(ix,iy,alpha,rn)
c
c  Use "gammarnd3(ix,iy,alpha,rn)"
c  together with "urnd(ix,iy,rn)".
c
c  Input:
c    ix, iy: Seeds
c    alpha:  Shape Parameter (1<alpha)
c  Output:
c    rn: Gamma Random Draw
c        with Parameters alpha and beta=1
c
      e=2.71828182845905
      a=1./sqrt(2.*alpha-1.)
      b=alpha-log(4.)
      c=alpha+sqrt(2.*alpha-1.)
      d=1.+log(4.5)
    1 call urnd(ix,iy,u1)
      call urnd(ix,iy,u2)
      y=a*log(u1/(1.-u1))
      rn=alpha*(e**y)
      z=u1*u1*u2
      r=b+c*y-rn
      if( r.ge.4.5*z-d ) go to 2
      if( r.lt.log(z)  ) go to 1
    2 return
```

```
27:         end
```

Note that `gammarnd3(ix,iy,alpha,rn)` requires `urnd(ix,iy,rn)`.

Line 24 corresponds to Step (iii)', which gives us a fast acceptance.

Taking into account a recent progress of a personal computer, we can erase Lines 17 and 24 from `gammarnd3`, because evaluating the `if(...)` sentences in Lines 24 and 25 sometimes takes more time than computing the logarithm in Line 25.

Thus, using both `gammarnd2` and `gammarnd3`, we have the gamma random number generator with parameters $\alpha > 0$ and $\beta = 1$.

9.7.2 Importance Resampling (重点的リサンプリング)

The **importance resampling** method also utilizes the sampling density $f_*(x)$, where we should choose the sampling density from which it is easy to generate random

draws.

Let x_i^* be the ith random draw of x generated from $f_*(x)$.

The acceptance probability is defined as:

$$\omega(x_i^*) = \frac{q(x_i^*)}{\sum_{j=1}^{n} q(x_j^*)},$$

where $q(\cdot)$ is represented as equation (19).

To obtain a random draws from $f(x)$, we perform the following procedure:

(i) Generate x_j^* from the sampling density $f_*(x)$ for $j = 1, 2, \cdots, n$.

(ii) Compute $\omega(x_j^*)$ for all $j = 1, 2, \cdots, n$.

(iii) Generate a uniform random draw u between zero and one and take $x = x_j^*$ when $\Omega_{j-1} \leq u < \Omega_j$, where $\Omega_j = \sum_{i=1}^{j} \omega(x_i^*)$ and $\Omega_0 \equiv 0$.

The x obtained in Step (iii) represents a random draw from the target density $f(x)$.

In Step (ii), all the probability weights $\omega(x_j^*)$, $j = 1, 2, \cdots, n$, have to be computed for importance resampling.

Thus, we need to generate n random draws from the sampling density $f_*(x)$ in advance.

When we want to generate more random draws (say, N random draws), we may repeat Step (iii) N times.

In the importance resampling method, there are n realizations, i.e., x_1^*, x_2^*, $\cdots$, x_n^*, which are mutually independently generated from the sampling density $f_*(x)$.

The cumulative distribution of $f(x)$ is approximated by the following empirical distribution:

$$P(X \le x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t = \int_{-\infty}^{x} \frac{f(t)}{f_*(t)} f_*(t)\,\mathrm{d}t = \frac{\int_{-\infty}^{x} q(t) f_*(t)\,\mathrm{d}t}{\int_{-\infty}^{\infty} q(t) f_*(t)\,\mathrm{d}t}$$
$$\approx \frac{(1/n)\sum_{i=1}^{n} q(x_i^*) I(x, x_i^*)}{(1/n)\sum_{j=1}^{n} q(x_j^*)} = \sum_{i=1}^{n} \omega(x_i^*) I(x, x_i^*),$$

where $I(x, x_i^*)$ denotes the indicator function which satisfies $I(x, x_i^*) = 1$ when $x \geq x_i^*$ and $I(x, x_i^*) = 0$ otherwise.

$P(X = x_i^*)$ is approximated as $\omega(x_i^*)$.

See Smith and Gelfand (1992) and Bernardo and Smith (1994) for the importance resampling procedure.

As mentioned in Section 9.7.1, for rejection sampling, $f(x)$ may be a kernel of the target density, or equivalently, $f(x)$ may be proportional to the target density.

Similarly, the same situation holds in the case of importance resampling.

That is, $f(x)$ may be proportional to the target density for importance resampling, too.

To obtain a random draws from $f(x)$, importance resampling requires n random draws from the sampling density $f_*(x)$, but rejection sampling needs $(1 + N_R)$ random draws from the sampling density $f_*(x)$.

For importance resampling, when we have n different random draws from the sampling density, we pick up one of them with the corresponding probability weight.

The importance resampling procedure computationally takes a lot of time, because we have to compute all the probability weights $\Omega_j,\ j = 1, 2, \cdots, n$, in advance even when we want only one random draw.

When we want to generate N random draws, importance resampling requires n random draws from the sampling density $f_*(x)$, but rejection sampling needs $n(1 + N_R)$ random draws from the sampling density $f_*(x)$.

Thus, as N increases, importance resampling is relatively less computational than rejection sampling.

Note that $N < n$ is recommended for the importance resampling method.

In addition, when we have N random draws from the target density $f(x)$, some of the random draws take the exactly same values for importance resampling, while all the

random draws take the different values for rejection sampling.

Therefore, we can see that importance resampling is inferior to rejection sampling in the sense of precision of the random draws.

Normal Distribution: $N(0,1)$**:** Again, we consider an example of generating standard normal random draws based on the half-normal distribution:

$$f(x) = \begin{cases} \dfrac{2}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, & \text{for } 0 \le x < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

We take the sampling density as the following exponential distribution:

$$f_*(x) = \begin{cases} e^{-x}, & \text{for } 0 \le x < \infty, \\ 0, & \text{otherwise,} \end{cases}$$

which is exactly the same sampling density as in Section 9.7.1.

Given the random draws x_i^*, $i = 1, \cdots, n$, generated from the above exponential density $f_*(x)$, the acceptance probability $\omega(x_i^*)$ is given by:

$$\omega(x_i^*) = \frac{q(x_i^*)}{\sum_{j=1}^n q(x_j^*)} = \frac{f(x_i^*)/f_*(x_i^*)}{\sum_{j=1}^n f(x_j^*)/f_*(x_j^*)} = \frac{\exp(-\frac{1}{2}x_i^{*2} + x_i^*)}{\sum_{j=1}^n \exp(-\frac{1}{2}x_j^{*2} + x_j^*)}.$$

Therefore, a random draw from the half-normal distribution is generated as follows.

(i) Generate uniform random draws $u_1, u_2, \cdots, u_n$ from $U(0, 1)$.

(ii) Obtain $x_i^* = -\log(u_i)$ for $i = 1, 2, \cdots, n$.

(iii) Compute $\omega(x_i^*)$ for $i = 1, 2, \cdots, n$.

(iv) Generate a uniform random draw v_1 from $U(0, 1)$.

(v) Set $x = x_j^*$ when $\Omega_{j-1} \leq v_1 < \Omega_j$ for $\Omega_j = \sum_{i=1}^j \omega(x_i^*)$ and $\Omega_0 = 0$.

x is taken as a random draw generated from the half-normal distribution $f(x)$.

In order to have a standard normal random draw, we additionally put the following step.

(vi) Generate a uniform random draw v_2 from $U(0, 1)$, and set $z = x$ if $v_2 \leq 1/2$ and $z = -x$ otherwise.

z represents a standard normal random draw.

Note that Step (vi) above corresponds to Step (iv) in Section 9.7.1.

Steps (i) – (vi) shown above represent the generator which yields one standard normal random draw.

When we want N standard normal random draws, Steps (iv) – (vi) should be repeated N times.

In Steps (iv) and (v), a random draw from $f(x)$ is generated based on Ω_j for $j = 1, 2, \cdots, n$.

Gamma Distribution: $G(\alpha, 1)$ **for** $0 < \alpha \leq 1$**:** When $X \sim G(\alpha, 1)$, the density function of X is given by:

$$f(x) = \begin{cases} \dfrac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}, & \text{for } 0 < x < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

The sampling density is taken as:

$$f_*(x) = \frac{e}{\alpha + e} \alpha x^{\alpha-1} I_1(x) + \frac{\alpha}{\alpha + e} e^{-x+1} I_2(x),$$

which is the same function as in `gammarnd2` of Section 9.7.1, where both $I_1(x)$ and $I_2(x)$ denote the indicator functions defined in Section 9.7.1.

The probability weights are given by:

$$\omega(x_i^*) = \frac{q(x_i^*)}{\sum_{j=1}^n q(x_j^*)} = \frac{f(x_i^*)/f_*(x_i^*)}{\sum_{j=1}^n f(x_j^*)/f_*(x_j^*)}$$

$$= \frac{x_i^{*\alpha-1} e^{-x_i^*} / (x_i^{*\alpha-1} I_1(x_i^*) + e^{-x_i^*} I_2(x_i^*))}{\sum_{j=1}^{n} x_j^{*\alpha-1} e^{-x_j^*} / (x_j^{*\alpha-1} I_1(x_j^*) + e^{-x_j^*} I_2(x_j^*))},$$

for $i = 1, 2, \cdots, n$.

The cumulative distribution function of $f_*(x)$ is represented as:

$$F_*(x) = \begin{cases} \dfrac{e}{\alpha + e} x^\alpha, & \text{if } 0 < x \leq 1, \\ \dfrac{e}{\alpha + e} + \dfrac{\alpha}{\alpha + e}(1 - e^{-x+1}), & \text{if } x > 1. \end{cases}$$

Therefore, x_i^* can be generated by utilizing both the composition method and the inverse transform method.

Given x_i^*, compute $\omega(x_i^*)$ for $i = 1, 2, \cdots, n$, and take $x = x_i^*$ with probability $\omega(x_i^*)$. Summarizing above, the random number generation procedure for the gamma distribution is given by:

(i) Generate uniform random draws u_i, $i = 1, 2, \cdots, n$, from $U(0, 1)$, and set $x_i^* =$

$((\alpha/e+1)u_i)^{1/\alpha}$ and $\omega(x_i^*) = e^{-x_i^*}$ if $u_i \le e/(\alpha+e)$ and take $x_i^* = -\log((1/e + 1/\alpha)(1-u_i))$ and $\omega(x_i^*) = x_i^{*\alpha-1}$ if $u_i > e/(\alpha+e)$ for $i = 1, 2, \cdots, n$.

(ii) Compute $\Omega_i = \sum_{j=1}^{i} \omega(x_j^*)$ for $i = 1, 2, \cdots, n$, where $\Omega_0 = 0$.

(iii) Generate a uniform random draw v from $U(0,1)$, and take $x = x_i^*$ when $\Omega_{i-1} \le v < \Omega_i$.

As mentioned above, this algorithm yields one random draw.

If we want N random draws, Step (iii) should be repeated N times.

Beta Distribution: The beta distribution with parameters α and β is of the form:

$$f(x) = \begin{cases} \dfrac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}, & \text{for } 0 < x < 1, \\ 0, & \text{otherwise.} \end{cases}$$

The sampling density is taken as:

$$f_*(x) = \begin{cases} 1, & \text{for } 0 < x < 1, \\ 0, & \text{otherwise,} \end{cases}$$

which represents the uniform distribution between zero and one.

The probability weights $\omega(x_i^*)$, $i = 1, 2, \cdots, n$, are given by:

$$\omega(x_i^*) = \frac{q(x_i^*)}{\sum_{j=1}^n q(x_j^*)} = \frac{f(x_i^*)/f_*(x_i^*)}{\sum_{j=1}^n f(x_j^*)/f_*(x_j^*)} = \frac{x_i^{*\alpha-1}(1-x_i^*)^{\beta-1}}{\sum_{j=1}^n x_j^{*\alpha-1}(1-x_j^*)^{\beta-1}}.$$

Therefore, to generate a random draw from $f(x)$, first generate x_i^*, $i = 1, 2, \cdots, n$, from $U(0, 1)$, second compute $\omega(x_i^*)$ for $i = 1, 2, \cdots$,n, and finally take $x = x_i^*$ with probability $\omega(x_i^*)$.

We have shown three examples of the importance resampling procedure in this section.

One of the advantages of importance resampling is that it is really easy to construct a Fortran source code.

However, the disadvantages are that (i) importance resampling takes quite a long time because we have to obtain all the probability weights in advance and (ii) importance resampling requires a great amount of storages for x_i^* and Ω_i for $i = 1, 2, \cdots, n$.

9.7.3 Metropolis-Hastings Algorithm (メトロポリスーハスティングス・アルゴリズム)

This section is based on Geweke and Tanizaki (2003), where three sampling distributions are compared with respect to precision of the random draws from the target density $f(x)$.

The **Metropolis-Hastings algorithm** is also one of the sampling methods to generate random draws from any target density $f(x)$, utilizing sampling density $f_*(x)$, even in the case where it is not easy to generate random draws from the target density.

Let us define the acceptance probability by:

$$\omega(x_{i-1}, x^*) = \min\Big(\frac{q(x^*)}{q(x_{i-1})}, 1\Big) = \min\Big(\frac{f(x^*)/f_*(x^*)}{f(x_{i-1})/f_*(x_{i-1})}, 1\Big),$$

where $q(\cdot)$ is defined as equation (19).

By the Metropolis-Hastings algorithm, a random draw from $f(x)$ is generated in the following way:

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

(ii) Generate x^* from $f_*(x)$ and compute $\omega(x_{i-1}, x^*)$ given x_{i-1}.

(iii) Set $x_i = x^*$ with probability $\omega(x_{i-1}, x^*)$ and $x_i = x_{i-1}$ otherwise.

(iv) Repeat Steps (ii) and (iii) for $i = -M+1, -M+2, \cdots, 1$.

In the above algorithm, x_1 is taken as a random draw from $f(x)$.

When we want more random draws (say, N), we replace Step (iv) by Step (iv)', which is represented as follows:

(iv)' Repeat Steps (ii) and (iii) for $i = -M+1, -M+2, \cdots, N$.

When we implement Step (iv)', we can obtain a series of random draws x_{-M}, x_{-M+1}, $\cdots$, x_0, x_1, x_2, $\cdots$, x_N, where x_{-M}, x_{-M+1}, $\cdots$, x_0 are discarded from further consideration.

The last N random draws are taken as the random draws generated from the target density $f(x)$.

Thus, N denotes the number of random draws.

M is sometimes called the **burn-in period**.

We can justify the above algorithm given by Steps (i) – (iv) as follows.

The proof is very similar to the case of rejection sampling in Section 9.7.1.

We show that x_i is the random draw generated from the target density $f(x)$ under the assumption x_{i-1} is generated from $f(x)$.

Let U be the uniform random variable between zero and one, X be the random vari-

able which has the density function $f(x)$ and x^* be the realization (i.e., the random draw) generated from the sampling density $f_*(x)$.

Consider the probability $P(X \leq x | U \leq \omega(x_{i-1}, x^*))$, which should be the cumulative distribution of X, i.e., $F(x)$.

The probability $P(X \leq x | U \leq \omega(x_{i-1}, x^*))$ is rewritten as follows:

$$P(X \leq x | U \leq \omega(x_{i-1}, x^*)) = \frac{P(X \leq x, U \leq \omega(x_{i-1}, x^*))}{P(U \leq \omega(x_{i-1}, x^*))},$$

where the numerator is represented as:

$$\begin{aligned}
P(X \leq x, U \leq \omega(x_{i-1}, x^*)) &= \int_{-\infty}^{x} \int_{0}^{\omega(x_{i-1},t)} f_{u,*}(u,t)\, \mathrm{d}u\, \mathrm{d}t \\
&= \int_{-\infty}^{x} \int_{0}^{\omega(x_{i-1},t)} f_u(u) f_*(t)\, \mathrm{d}u\, \mathrm{d}t = \int_{-\infty}^{x} \Big(\int_{0}^{\omega(x_{i-1},t)} f_u(u)\, \mathrm{d}u \Big) f_*(t)\, \mathrm{d}t \\
&= \int_{-\infty}^{x} \Big(\int_{0}^{\omega(x_{i-1},t)} \mathrm{d}u \Big) f_*(t)\, \mathrm{d}t = \int_{-\infty}^{x} \Big[u \Big]_{0}^{\omega(x_{i-1},t)} f_*(t)\, \mathrm{d}t \\
&= \int_{-\infty}^{x} \omega(x_{i-1}, t) f_*(t)\, \mathrm{d}t = \int_{-\infty}^{x} \frac{f_*(x_{i-1}) f(t)}{f(x_{i-1})}\, \mathrm{d}t = \frac{f_*(x_{i-1})}{f(x_{i-1})} F(x)
\end{aligned}$$

and the denominator is given by:

$$P(U \leq \omega(x_{i-1}, x^*)) = P(X \leq \infty, U \leq \omega(x_{i-1}, x^*)) = \frac{f_*(x_{i-1})}{f(x_{i-1})} F(\infty) = \frac{f_*(x_{i-1})}{f(x_{i-1})}.$$

The density function of U is given by $f_u(u) = 1$ for $0 < u < 1$.

Let X^* be the random variable which has the density function $f_*(x)$.

In the numerator, $f_{u,*}(u, x)$ denotes the joint density of random variables U and X^*.

Because the random draws of U and X^* are independently generated, we have $f_{u,*}(u, x) = f_u(u) f_*(x) = f_*(x)$.

Thus, the first four equalities are derived.

Substituting the numerator and denominator shown above, we have the following equality:

$$P(X \leq x | U \leq \omega(x_{i-1}, x^*)) = F(x).$$

Thus, the x^* which satisfies $u \leq \omega(x_{i-1}, x^*)$ indicates a random draw from $f(x)$.

We set $x_i = x_{i-1}$ if $u \leq \omega(x_{i-1}, x^*)$ is not satisfied. x_{i-1} is already assumed to be a random draw from $f(x)$.

Therefore, it is shown that x_i is a random draw from $f(x)$.

See Gentle (1998) for the discussion above.

As in the case of rejection sampling and importance resampling, note that $f(x)$ may be a kernel of the target density, or equivalently, $f(x)$ may be proportional to the target density.

The same algorithm as Steps (i) – (iv) can be applied to the case where $f(x)$ is proportional to the target density, because $f(x^*)$ is divided by $f(x_{i-1})$ in $\omega(x_{i-1}, x^*)$.

As a general formulation of the sampling density, instead of $f_*(x)$, we may take the sampling density as the following form: $f_*(x|x_{i-1})$, where a candidate random draw x^* depends on the $(i-1)$th random draw, i.e., x_{i-1}.

For choice of the sampling density $f_*(x|x_{i-1})$, Chib and Greenberg (1995) pointed out

as follows.

$f_*(x|x_{i-1})$ should be chosen so that the chain travels over the support of $f(x)$, which implies that $f_*(x|_{i-1})$ should not have too large variance and too small variance, compared with $f(x)$.

See, for example, Smith and Roberts (1993), Bernardo and Smith (1994), O'Hagan (1994), Tierney (1994), Geweke (1996), Gamerman (1997), Robert and Casella (1999) and so on for the Metropolis-Hastings algorithm.

As an alternative justification, note that the Metropolis-Hastings algorithm is formulated as follows:

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

where $f^*(u|v)$ denotes the transition distribution, which is characterized by Step (iii). x_{i-1} is generated from $f_{i-1}(\cdot)$ and x_i is from $f^*(\cdot|x_{i-1})$.

x_i depends only on x_{i-1}, which is called the **Markov property**.

The sequence $\{\cdots, x_{i-1}, x_i, x_{i+1}, \cdots\}$ is called the **Markov chain**.

The Monte Carlo statistical methods with the sequence $\{\cdots, x_{i-1}, x_i, x_{i+1}, \cdots\}$ is called the **Markov chain Monte Carlo (MCMC)**.

From Step (iii), $f^*(u|v)$ is given by:

$$f^*(u|v) = \omega(v,u) f_*(u|v) + \Big(1 - \int \omega(v,u) f_*(u|v)\, \mathrm{d}u\Big) p(u), \tag{20}$$

where $p(x)$ denotes the following probability function:

$$p(u) = \begin{cases} 1, & \text{if } u = v, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, x is generated from $f_*(u|v)$ with probability $\omega(v,u)$ and from $p(u)$ with probability $1 - \int \omega(v,u) f_*(u|v)\, \mathrm{d}u$.

Now, we want to show $f_i(u) = f_{i-1}(u) = f(u)$ as i goes to infinity, which implies that both x_i and x_{i-1} are generated from the invariant distribution function $f(u)$ for sufficiently large i.

To do so, we need to consider the condition satisfying the following equation:

$$f(u) = \int f^*(u|v) f(v)\, \mathrm{d}v. \tag{21}$$

Equation (21) holds if we have the following equation:

$$f^*(v|u) f(u) = f^*(u|v) f(v), \tag{22}$$

which is called the **reversibility condition**.

By taking the integration with respect to v on both sides of equation (22), equation (21) is obtained.

Therefore, we have to check whether the $f^*(u|v)$ shown in equation (20) satisfies equation (22).

It is straightforward to verify that

$$\omega(v,u) f_*(u|v) f(v) = \omega(u,v) f_*(v|u) f(u),$$
$$\Big(1 - \int \omega(v,u) f_*(u|v)\, \mathrm{d}u\Big) p(u) f(v) = \Big(1 - \int \omega(u,v) f_*(v|u)\, \mathrm{d}v\Big) p(v) f(u).$$

Thus, as i goes to infinity, x_i is a random draw from the target density $f(\cdot)$.

If x_i is generated from $f(\cdot)$, then x_{i+1} is also generated from $f(\cdot)$.

Therefore, all the x_i, x_{i+1}, x_{i+2}, $\cdots$ are taken as random draws from the target density $f(\cdot)$.

The requirement for uniform convergence of the Markov chain is that the chain should be **irreducible** and **aperiodic**.

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

Let $C_i(x_0)$ be the set of possible values of x_i from starting point x_0.

If there exist two possible starting values, say x^* and x^{**}, such that $C_i(x^*) \cap C_i(x^{**}) = \emptyset$ (i.e., empty set) for all i, then the same limiting distribution cannot be reached from both starting points.

Thus, in the case of $C_i(x^*) \cap C_i(x^{**}) = \emptyset$, the convergence may fail.

A Markov chain is said to be **irreducible** if there exists an i such that $P(x_i \in C|x_0) > 0$

for any starting point x_0 and any set C such that $\int_C f(x)\,\mathrm{d}x > 0$.

The irreducible condition ensures that the chain can reach all possible x values from any starting point.

Moreover, as another case in which convergence may fail, if there are two disjoint set C^1 and C^2 such that $x_{i-1} \in C^1$ implies $x_i \in C^2$ and $x_{i-1} \in C^2$ implies $x_i \in C^1$, then the chain oscillates between C^1 and C^2 and we again have $C_i(x^*) \cap C_i(x^{**}) = \emptyset$ for all i when $x^* \in C^1$ and $x^{**} \in C^2$.

Accordingly, we cannot have the same limiting distribution in this case, either.

It is called **aperiodic** if the chain does not oscillate between two sets C^1 and C^2 or cycle around a partition $C^1, C^2, \cdots, C^r$ of r disjoint sets for $r > 2$.

See O'Hagan (1994) for the discussion above.

For the Metropolis-Hastings algorithm, x_1 is taken as a random draw of x from $f(x)$ for sufficiently large M.

To obtain N random draws, we need to generate $M + N$ random draws.

Moreover, clearly we have $\mathrm{Cov}(x_{i-1}, x_i) > 0$, because x_i is generated based on x_{i-1} in Step (iii).

Therefore, for precision of the random draws, the Metropolis-Hastings algorithm gives us the worst random number of the three sampling methods. i.e., rejection sampling in Section 9.7.1, importance resampling in Section 9.7.2 and the Metropolis-Hastings algorithm in this section.

Based on Steps (i) – (iii) and (iv)', under some conditions the basic result of the Metropolis-Hastings algorithm is as follows:

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i) \longrightarrow \mathrm{E}(g(x)) = \int g(x) f(x)\,\mathrm{d}x, \qquad \text{as } N \longrightarrow \infty,$$

where $g(\cdot)$ is a function, which is representatively taken as $g(x) = x$ for mean and $g(x) = (x - \overline{x})^2$ for variance.

$\overline{x}$ denotes $\overline{x} = (1/N) \sum_{i=1}^{N} x_i$.

Thus, it is shown that $(1/N) \sum_{i=1}^{N} g(x_i)$ is a consistent estimate of $\mathrm{E}(g(x))$, even though $x_1, x_2, \cdots, x_N$ are mutually correlated.

As an alternative random number generation method to avoid the positive correlation, we can perform the case of $N = 1$ as in the above procedures (i) – (iv) N times in parallel, taking different initial values for x_{-M}.

In this case, we need to generate $M + 1$ random numbers to obtain one random draw from $f(x)$.

That is, N random draws from $f(x)$ are based on $N(1 + M)$ random draws from $f_*(x|x_{i-1})$.

Thus, we can obtain mutually independently distributed random draws.

For precision of the random draws, the alternative Metropolis-Hastings algorithm should be similar to rejection sampling.

However, this alternative method is too computer-intensive, compared with the above procedures (i) – (iii) and (iv)', which takes more time than rejection sampling in the case of $M > N_R$.

Furthermore, the sampling density has to satisfy the following conditions:

(i) we can quickly and easily generate random draws from the sampling density and

(ii) the sampling density should be distributed with the same range as the target density.

See, for example, Geweke (1992) and Mengersen, Robert and Guihenneuc-Jouyaux (1999) for the MCMC convergence diagnostics.

Since the random draws based on the Metropolis-Hastings algorithm heavily depend on choice of the sampling density, we can see that the Metropolis-Hastings algorithm has the problem of specifying the sampling density, which is the crucial criticism.

Several generic choices of the sampling density are discussed by Tierney (1994) and Chib and Greenberg (1995).

We can consider several candidates for the sampling density $f_*(x|x_{i-1})$, i.e., Sampling Densities I – III.

3.4.1.1 Sampling Density I (Independence Chain) For the sampling density, we have started with $f_*(x)$ in this section.

Thus, one possibility of the sampling density is given by: $f_*(x|x_{i-1}) = f_*(x)$, where $f_*(\cdot)$ does not depend on x_{i-1}.

This sampling density is called the **independence chain**.

For example, it is possible to take $f_*(x) = N(\mu_*, \sigma_*^2)$, where μ_* and σ_*^2 are the hyper-parameters.

Or, when x lies on a certain interval, say (a, b), we can choose the uniform distribution

$f_*(x) = 1/(b-a)$ for the sampling density.

3.4.1.2 Sampling Density II (Random Walk Chain) We may take the sampling density called the **random walk chain**, i.e., $f_*(x|x_{i-1}) = f_*(x - x_{i-1})$.
Representatively, we can take the sampling density as $f_*(x|x_{i-1}) = N(x_{i-1}, \sigma_*^2)$, where σ_*^2 denotes the hyper-parameter.
Based on the random walk chain, we have a series of the random draws which follow the random walk process.

3.4.1.3 Sampling Density III (Taylored Chain) The alternative sampling distribution is based on approximation of the log-kernel (see Geweke and Tanizaki (1999, 2001, 2003)), which is a substantial extension of the **Taylored chain** discussed in Chib, Greenberg and Winkelmann (1998).

Let $p(x) = \log(f(x))$, where $f(x)$ may denote the kernel which corresponds to the target density.

Approximating the log-kernel $p(x)$ around x_{i-1} by the second order Taylor series expansion, $p(x)$ is represented as:

$$p(x) \approx p(x_{i-1}) + p'(x_{i-1})(x - x_{i-1}) + \frac{1}{2}p''(x_{i-1})(x - x_{i-1})^2, \tag{23}$$

where $p'(\cdot)$ and $p''(\cdot)$ denote the first- and second-derivatives.

Depending on the values of $p'(x)$ and $p''(x)$, we have the four cases, i.e., Cases 1 – 4, which are classified by (i) $p''(x) < -\epsilon$ in Case 1 or $p''(x) \geq -\epsilon$ in Cases 2 – 4 and (ii) $p'(x) < 0$ in Case 2, $p'(x) > 0$ in Case 3 or $p'(x) = 0$ in Case 4.

Geweke and Tanizaki (2003) suggested introducing ϵ into the Taylored chain discussed in Geweke and Tanizaki (1999, 2001).

Note that $\epsilon = 0$ is chosen in Geweke and Tanizaki (1999, 2001).

To improve precision of random draws, ϵ should be a positive value, which will be discussed later in detail (see Remark 1 for ϵ).

Case 1: $p''(x_{i-1}) < -\epsilon$: Equation (23) is rewritten by:

$$p(x) \approx p(x_{i-1}) - \frac{1}{2}\Big(\frac{1}{-1/p''(x_{i-1})}\Big)\Big(x - (x_{i-1} - \frac{p'(x_{i-1})}{p''(x_{i-1})})\Big)^2 + r(x_{i-1}),$$

where $r(x_{i-1})$ is an appropriate function of x_{i-1}.

Since $p''(x_{i-1})$ is negative, the second term in the right-hand side is equivalent to the exponential part of the normal density.

Therefore, $f_*(x|x_{i-1})$ is taken as $N(\mu_*, \sigma_*^2)$, where $\mu_* = x_{i-1} - p'(x_{i-1})/p''(x_{i-1})$ and $\sigma_*^2 = -1/p''(x_{i-1})$.

Case 2: $p''(x_{i-1}) \geq -\epsilon$ **and** $p'(x_{i-1}) < 0$: Perform linear approximation of $p(x)$.

Let x^+ be the nearest mode with $x^+ < x_{i-1}$.

Then, $p(x)$ is approximated by a line passing between x^+ and x_{i-1}, which is written as:

$$p(x) \approx p(x^+) + \frac{p(x^+) - p(x_{i-1})}{x^+ - x_{i-1}}(x - x^+).$$

From the second term in the right-hand side, the sampling density is represented as the exponential distribution with $x > x^+ - d$, i.e., $f_*(x|x_{i-1}) = \lambda \exp\Big(-\lambda(x - (x^+ - d))\Big)$ if $x^+ - d < x$ and $f_*(x|x_{i-1}) = 0$ otherwise, where λ is defined as:

$$\lambda = \left|\frac{p(x^+) - p(x_{i-1})}{x^+ - x_{i-1}}\right|.$$

d is a positive value, which will be discussed later (see Remark 2 for d).

Thus, a random draw x^* from the sampling density is generated by $x^* = w + (x^+ - d)$, where w represents the exponential random variable with parameter λ.

Case 3: $p''(x_{i-1}) \geq -\epsilon$ **and** $p'(x_{i-1}) > 0$: Similarly, perform linear approximation of $p(x)$ in this case.

Let x^+ be the nearest mode with $x_{i-1} < x^+$.

Approximation of $p(x)$ is exactly equivalent to that of Case 2.

Taking into account $x < x^+ + d$, the sampling density is written as: $f_*(x|x_{i-1}) = \lambda \exp\Big(-\lambda((x^+ + d) - x)\Big)$ if $x < x^+ + d$ and $f_*(x|x_{i-1}) = 0$ otherwise.

Thus, a random draw x^* from the sampling density is generated by $x^* = (x^+ + d) - w$, where w is distributed as the exponential random variable with parameter λ.

Case 4: $p''(x_{i-1}) \geq -\epsilon$ **and** $p'(x_{i-1}) = 0$: In this case, $p(x)$ is approximated as a uniform distribution at the neighborhood of x_{i-1}.

As for the range of the uniform distribution, we utilize the two appropriate

values x^+ and x^{++}, which satisfies $x^+ < x < x^{++}$.

When we have two modes, x^+ and x^{++} may be taken as the modes.

Thus, the sampling density $f_*(x|x_{i-1})$ is obtained by the uniform distribution on the interval between x^+ and x^{++}, i.e., $f_*(x|x_{i-1}) = 1/(x^{++} - x^+)$ if $x^+ < x < x^{++}$ and $f_*(x|x_{i-1}) = 0$ otherwise.

Thus, for approximation of the kernel, all the possible cases are given by Cases 1 – 4, depending on the values of $p'(\cdot)$ and $p''(\cdot)$.

Moreover, in the case where x is a vector, applying the procedure above to each element of x, Sampling III is easily extended to multivariate cases.

Finally, we discuss about ϵ and d in the following remarks.

Remark 1: ϵ in Cases 1 – 4 should be taken as an appropriate positive number. It may seem more natural to take $\epsilon = 0$, rather than $\epsilon > 0$.

The reason why $\epsilon > 0$ is taken is as follows.

Consider the case of $\epsilon = 0$.

When $p''(x_{i-1})$ is negative and it is very close to zero, variance σ_*^2 in Case 1 becomes extremely large because of $\sigma_*^2 = -1/p''(x_{i-1})$.

In this case, the obtained random draws are too broadly distributed and accordingly they become unrealistic, which implies that we have a lot of outliers.

To avoid this situation, ϵ should be positive.

It might be appropriate that ϵ should depend on variance of the target density, because ϵ should be small if variance of the target density is large.

Thus, in order to reduce a number of outliers, $\epsilon > 0$ is recommended.

Remark 2: For d in Cases 2 and 3, note as follows.

As an example, consider the unimodal density in which we have Cases 2 and 3.

Let x^+ be the mode.

We have Case 2 in the right-hand side of x^+ and Case 3 in the left-hand side of x^+.

In the case of $d = 0$, we have the random draws generated from either Case 2 or 3.

In this situation, the generated random draw does not move from one case to another.

In the case of $d > 0$, however, the distribution in Case 2 can generate a random draw in Case 3.

That is, for positive d, the generated random draw may move from one case to another, which implies that the irreducibility condition of the MH algorithm is guaranteed.

Normal Distribution: $N(0, 1)$**:** As in Sections 9.7.1 and 9.7.2, we consider an example of generating standard normal random draws based on the half-normal dis-

tribution:

$$f(x) = \begin{cases} \dfrac{2}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, & \text{for } 0 \le x < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

As in Sections 9.7.1 and 9.7.2, we take the sampling density as the following exponential distribution:

$$f_*(x) = \begin{cases} e^{-x}, & \text{for } 0 \le x < \infty, \\ 0, & \text{otherwise,} \end{cases}$$

which is the independence chain, i.e., $f_*(x|x_{i-1}) = f_*(x)$.

Then, the acceptance probability $\omega(x_{i-1}, x^*)$ is given by:

$$\begin{aligned} \omega(x_{i-1}, x^*) &= \min\Big(\frac{f(x^*)/f_*(x^*)}{f(x_{i-1})/f_*(x_{i-1})}, 1\Big) \\ &= \min\Big(\exp(-\frac{1}{2}x^{*2} + x^* + \frac{1}{2}x_{i-1}^2 - x_{i-1}), 1\Big). \end{aligned}$$

Utilizing the Metropolis-Hastings algorithm, the standard normal random number generator is shown as follows:

(i) Take an appropriate initial value of x as x_{-M} (for example, $x_{-M} = 0$).

(ii) Set $y_{i-1} = |x_{i-1}|$.

(iii) Generate a uniform random draw u_1 from $U(0, 1)$ and compute $\omega(y_{i-1}, y^*)$ where $y^* = -\log(u_1)$.

(iv) Generate a uniform random draw u_2 from $U(0, 1)$, and set $y_i = y^*$ if $u_2 \leq \omega(y_{i-1}, y^*)$ and $y_i = y_{i-1}$ otherwise.

(v) Generate a uniform random draw u_3 from $U(0, 1)$, and set $x_i = y_i$ if $u_3 \leq 0.5$ and $x_i = -y_i$ otherwise.

(vi) Repeat Steps (ii) – (v) for $i = -M + 1, -M + 2, \cdots, 1$.

y_1 is taken as a random draw from $f(x)$. M denotes the burn-in period.

If a lot of random draws (say, N random draws) are required, we replace Step (vi) by Step (vi)' represented as follows:

(vi)' Repeat Steps (ii) – (v) for $i = -M + 1, -M + 2, \cdots, N$.

In Steps (ii) – (iv), a half-normal random draw is generated.

Note that the absolute value of x_{i-1} is taken in Step (ii) because the half-normal random draw is positive.

In Step (v), the positive or negative sign is randomly assigned to y_i.

Gamma Distribution: $G(\alpha, 1)$ **for** $0 < \alpha \leq 1$**:** When $X \sim G(\alpha, 1)$, the density function of X is given by:

$$f(x) = \begin{cases} \dfrac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}, & \text{for } 0 < x < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

As in `gammarnd2` of Sections 9.7.1 and `gammarnd4` of 9.7.2, the sampling density is taken as:

$$f_*(x) = \frac{e}{\alpha + e}\alpha x^{\alpha-1} I_1(x) + \frac{\alpha}{\alpha + e} e^{-x+1} I_2(x),$$

where both $I_1(x)$ and $I_2(x)$ denote the indicator functions defined in Section 9.7.1. Then, the acceptance probability is given by:

$$\begin{aligned}\omega(x_{i-1}, x^*) &= \min\Big(\frac{q(x^*)}{q(x_{i-1})}, 1\Big) = \min\Big(\frac{f(x^*)/f_*(x^*)}{f(x_{i-1})/f_*(x_{i-1})}, 1\Big) \\ &= \min\Big(\frac{x^{*\alpha-1}e^{-x^*}/(x^{*\alpha-1}I_1(x^*) + e^{-x^*}I_2(x^*))}{x_{i-1}^{\alpha-1}e^{-x_{i-1}}/(x_{i-1}^{\alpha-1}I_1(x_{i-1}) + e^{-x_{i-1}}I_2(x_{i-1}))}, 1\Big).\end{aligned}$$

As shown in Section 9.7.1, the cumulative distribution function of $f_*(x)$ is represented as:

$$F_*(x) = \begin{cases} \dfrac{e}{\alpha + e} x^{\alpha}, & \text{if } 0 < x \le 1, \\ \dfrac{e}{\alpha + e} + \dfrac{\alpha}{\alpha + e}(1 - e^{-x+1}), & \text{if } x > 1. \end{cases}$$

Therefore, a candidate of the random draw, i.e., x^*, can be generated from $f_*(x)$, by utilizing both the composition method and the inverse transform method.
Then, using the Metropolis-Hastings algorithm, the gamma random number generation method is shown as follows.

(i) Take an appropriate initial value as x_{-M}.

(ii) Generate a uniform random draw u_1 from $U(0,1)$, and set $x^* = ((\alpha/e+1)u_1)^{1/\alpha}$ if $u_1 \leq e/(\alpha+e)$ and $x^* = -\log((1/e+1/\alpha)(1-u_1))$ if $u_1 > e/(\alpha+e)$.

(iii) Compute $\omega(x_{i-1}, x^*)$.

(iv) Generate a uniform random draw u_2 from $U(0,1)$, and set $x_i = x^*$ if $u_2 \leq \omega(x_{i-1}, x^*)$ and $x_i = x_{i-1}$ otherwise.

(v) Repeat Steps (ii) – (iv) for $i = -M+1, -M+2, \cdots, 1$.

For sufficiently large M, x_1 is taken as a random draw from $f(x)$. u_1 and u_2 should be

independently distributed.

M denotes the burn-in period. If we need a lot of random draws (say, N random draws), replace Step (v) by Step (v)', which is given by:

(v)' Repeat Steps (ii) – (iv) for $i = -M + 1, -M + 2, \cdots, N$.

Beta Distribution: The beta distribution with parameters α and β is of the form:

$$f(x) = \begin{cases} \dfrac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}, & \text{for } 0 < x < 1, \\ 0, & \text{otherwise.} \end{cases}$$

The sampling density is taken as:

$$f_*(x) = \begin{cases} 1, & \text{for } 0 < x < 1, \\ 0, & \text{otherwise,} \end{cases}$$

which represents the uniform distribution between zero and one.

The probability weights $\omega(x_i^*)$, $i = 1, 2, \cdots, n$, are given by:

$$\omega(x_{i-1}, x^*) = \min\Big(\frac{f(x^*)/f_*(x^*)}{f(x_{i-1})/f_*(x_{i-1})}, 1\Big) = \min\Big(\Big(\frac{x^*}{x_{i-1}}\Big)^{\alpha-1}\Big(\frac{1-x^*}{1-x_{i-1}}\Big)^{\beta-1}, 1\Big).$$

Then, utilizing the Metropolis-Hastings algorithm, the random draws are generated as follows.

(i) Take an appropriate initial value as x_{-M}.

(ii) Generate a uniform random draw x^* from $U(0, 1)$, and compute $\omega(x_{i-1}, x^*)$.

(iii) Generate a uniform random draw u from $U(0, 1)$, which is independent of x^*, and set $x_i = x^*$ if $u \leq \omega(x_{i-1}, x^*)$ and $x_i = x_{i-1}$ if $u > \omega(x_{i-1}, x^*)$.

(iv) Repeat Steps (ii) and (iii) for $i = -M + 1, -M + 2, \cdots, 1$.

For sufficiently large M, x_1 is taken as a random draw from $f(x)$.

M denotes the burn-in period.

If we want a lot of random draws (say, N random draws), replace Step (iv) by Step (iv)', which is represented as follows:

(iv)' Repeat Steps (ii) and (iii) for $i = -M + 1, -M + 2, \cdots, N$.

9.7.4 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 9.7.1), importance resampling (Section 9.7.2) and the Metropolis-Hastings algorithm (Section 9.7.3).

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:

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

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:

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

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 $0 \le U_1 \le \sqrt{h(U_2/U_1)}$, generate a uniform point in the rectangle, and reject the point which does not satisfy $0 \le u_1 \le \sqrt{h(u_2/u_1)}$.

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 $x^2 h(x)$ 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 9.7.4 and 9.7.4, we show two examples as applications of the ratio-of-uniforms method.

Especially, in Section 9.7.4, 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
12:      1 call urnd(ix,iy,rn1)
13:        call urnd(ix,iy,rn2)
14:        rn2=(2.*rn2-1.)*sqrt(2.*e1)
15:        if(-4.*rn1*rn1*log(rn1).lt.rn2*rn2 ) go to 1
16:        rn=rn2/rn1
17:        return
18:        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 9.7.1 – 9.7.3, 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
```

```
14:         b=( (alpha-1.)/e )**(0.5*alpha-0.5)
15:         d=( (alpha+1.)/e )**(0.5*alpha+0.5)
16:       1 call urnd(ix,iy,rn0)
17:         call urnd(ix,iy,rn1)
18:         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:

$$\frac{\int h(x)\,\mathrm{d}x}{2b(d-c)} = \frac{e^{\alpha}\Gamma(\alpha)}{2(\alpha-1)^{(\alpha-1)/2}(\alpha+1)^{(\alpha+1)/2}}. \tag{24}$$

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)

```
1:       subroutine gammarnd7(ix,iy,alpha,rn)
2: c
3: c  Use "gammarnd7(ix,iy,alpha,rn)"
4: c  together with "urnd(ix,iy,rn)".
5: c
6: c  Input:
7: c    ix, iy:   Seeds
8: c    alpha:  Shape Parameter (alpha>1)
9: c  Output:
10: c    rn: Gamma Random Draw
11: c        with Parameters alpha and beta=1
```

```
12: C
13:         e =2.71828182845905
14:         c0=1.857764
15:         c1=alpha-1.
16:         c2=( alpha-1./(6.*alpha) )/c1
17:         c3=2./c1
18:         c4=c3+2.
19:         c5=1./sqrt(alpha)
20:       1 call urnd(ix,iy,u1)
21:         call urnd(ix,iy,u2)
22:         if(alpha.gt.2.5) u1=u2+c5*(1.-c0*u1)
23:         if(0.ge.u1.or.u1.ge.1.) go to 1
24:         w=c2*u2/u1
25:         if(c3*u1+w+1./w.le.c4) go to 2
26:         if(c3*log(u1)-log(w)+w.ge.1.) go to 1
27:       2 rn=c1*w
28:         return
29:         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.278 and `gammarnd7` on p.340, 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.

9.7.5 Gibbs Sampling

The sampling methods introduced in Sections 9.7.1 – 9.7.3 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 (22), 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, gener-

ating 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 $f_*(x|z) = f_*(x)$.

	$\mu_* \backslash \sigma_*$		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	0.000	0.000	0.000	0.000	0.000
			(59.25)	(100.00)	(74.89)	(59.04)	(40.99)	(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	0.000	0.001	0.000	0.000	0.000
			(36.28)	(47.98)	(55.75)	(51.19)	(38.68)	(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	0.005	0.001	0.001	0.000	0.001
			(8.79)	(15.78)	(26.71)	(33.78)	(32.50)	(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	0.073	−0.002	0.000	0.001	−0.001
			(1.68)	(3.53)	(9.60)	(17.47)	(24.31)	(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 $\mathrm{E}(X)$.

$\mathrm{E}(X^3)$ 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 $\mathrm{E}(X) = 0$, from the table a higher acceptance rate generally shows a better performance.

The high acceptance rate implies high randomness of the generated random draws.

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 9.7.2 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

E(X) and 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 sometimes 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 Algorithm," *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, "Illustration 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 Sampler 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.