Ann. Inst. Statist. Math.
35 (1983), Part B, 291-302

# STRATIFIED REJECTION AND SQUEEZE METHOD FOR GENERATING BETA RANDOM NUMBERS

H. SAKASEGAWA

(Received Aug. 11, 1982; revised Mar. 24, 1983)

## 1. Introduction

The importance of beta-distributed random variables in statistical simulation experiments lies in the fact that the family of beta distributions has a finite support and has a wide variety of shapes. This paper proposes new algorithms for generating random numbers with beta distributions

$$(1.1)\qquad f(x)=cx^{a-1}(1-x)^{b-1}\,,\qquad \text{for } a>0,\ b>0 \text{ and } 0<x<1\,,$$

where

$$c^{-1}=B(a, b)=\Gamma(a)\Gamma(b)/\Gamma(a, b)=\int_0^1 x^{a-1}(1-x)^{b-1}dx\,.$$

Like other algorithms for generating random numbers with non-uniform distributions, our algorithms generate beta random numbers by transforming uniform random numbers (uniform on the unit interval (0, 1)) and are exact, not approximate, in the sense that generated numbers are truly distributed if 'true' uniform random numbers are supplied.

Algorithms for generating beta random numbers have been proposed and tested by several authors including Jöhnk [6], Ahrens and Dieter [1], Atkinson and Whittaker [2], [3], Cheng [5] and Schmeiser and Babu [8]. Jöhnk used the fact that the ratio of $x$ to $x+y$, where $(x, y)$ is a random point in $\{(x, y); x^{1/a}+y^{1/b}<1\}$, is a beta-distributed random variate. Ahrens et al. considered the normal approximation for the case $a>1$ and $b>1$, and generated beta random numbers using normally distributed ones. Atkinson et al. and Schmeiser et al. considered another functional approximation. Cheng proposed an ingenious algorithm to generate random numbers with modified (second kind) beta distribution, which can be transformed into ordinary beta random numbers by simple calculation.

The main tool in all these algorithms mentioned above is a so-called rejection technique. To generate a random point in some possibly complex region $A$, not necessarily bounded, one finds another simple region $B$ covering $A$, then, generate random points uniformly on $B$ and select those which are contained in $A$. It is evident that selected points are uniformly distributed over $A$. This process to generate random points in $A$ is called rejection method. Squeeze method is an elaborated technique to improve the efficiency of the rejection method. Let $C$ be a region which contains $A$ and let $D$ be an another region which is contained in $A$, that is, $D \subset A \subset C$. When it is time consuming to test if a random point in $B$, say $P$, is also in $A$ ($P \in A$) or not ($P \notin A$), one may save time by testing first if $P \in D$ and/or $P \notin C$ before testing $P \in A$ or not. If $P$ is in $D$, it is also in $A$, and if $P$ is not in $C$, it is not in $A$. This technique has been widely applied to generating algorithms of random numbers with various statistical distributions and the name was given by Marsaglia [7].

In this paper we propose three algorithms corresponding to different shapes of the distributions, that is, $U$-shaped, $J$-shaped and unimodal ones. Each algorithm needs several constants dependent on two shape parameters, $a$ and $b$, and these values must be computed beforehand. Accordingly, our algorithms are not very effective when shape parameters change from time to time, but there are many situations where a sequence of random numbers with fixed shape parameters are required. In such cases our algorithms are superior to other existing algorithms mentioned above.

We give precise descriptions of the algorithms in the next section, and show results of timing tests comparing with other algorithms in Section 3.

## 2. Method

### 2.1. *Stratified rejection method*

Efficiency of the rejection method depends on two things which are, in general, contradictory. One is easiness of generating a random point in $B$, and the other is the expected number of necessary random points in $B$ to get one in $A$. The latter is given by the ratio $|B|/|A|$, where $|A|$ is an area of $A$.

Now we consider a technique to sample easily from $A$ keeping the ratio near to one. Let $B_1, B_2, \cdots$ be a decomposition of $B$ such that sampling from each subset is easier. We call $B_j$ the $j$th stratum of $B$. Let $q_j$ be an area of $B_j$. Our sampling plan is as follows: First, we randomly choose one stratum, say $B_j$, according to the ratio $q_1 : q_2 : \cdots$, then generate a random point uniformly on $B_j$. If the point is not contained in $A$, this sampling is a fail and the same sampling process is tried again, otherwise the point is the required one. We call the above technique as stratified rejection method. As the stratification is used only to simplify generating random points in $B$, correctness of the method is easily verified. It is expected to be more efficient than the old one by designing a fine stratification, since the expected number of random points in $B$ necessary to get one sample from $A$ is the same for two methods and since the additional step to choose one of the strata randomly is not so time-consuming.

In the following, we apply this technique to a beta random number generation. We consider respective algorithms according to different shapes of the distributions:

Case 1) $a, b<1$,
Case 2) $a<1<b$,
Case 2)′ $b<1<a$,
Case 3) $a, b>1$ and
Case 4) $(a-1)(b-1)=0$.

Since a beta distribution is symmetric in $a$ and $b$, case 2)′ is included in case 2): If $x$ is a random variate of case 2), $1-x$ is a random variate of case 2)′. For the last case, the inverse function method seems to be efficient if either $a$ or $b$ is not equal to one. If $a=b=1$, $y=f(x)$ becomes a uniform density. We treat three cases, 1), 2) and 3), in the following.

## 2.2. *Algorithm for Case* 1)

Let $t$ be a real number in (0, 1) and let a function $g(x)$ be defined on (0, 1) by

$$g(x)=\begin{cases} c(1-t)^{b-1}x^{a-1}, & \text{for } 0<x\leqq t, \\ ct^{a-1}(1-x)^{b-1}, & \text{for } t<x<1. \end{cases} \tag{2.1}$$

Let $B$ be a region between $y=g(x)$ and $x$-axis and let $A$ be a region between $y=f(x)$ and $x$-axis, which is completely covered by $B$. Let $B_1$ and $B_2$ be two strata of $B$ divided by the line $x=t$. To sample from each stratum, use inverse function method; $(tu^{1/a}, f(t)u^{1-(1/a)}v)$ is a random point in the left stratum and $(1-(1-t)(1-u)^{1/b}, f(t)(1-u)^{1-(1/b)}v)$ is a random point in the right stratum if $u$ and $v$ are uniform random numbers. If

$$f(x)>f(t)u^{1-(1/a)}v\,(=g(x)v), \qquad \text{where } x=tu^{1/a}, \tag{2.2}$$

in $B_1$ or

$$f(x)>f(t)(1-u)^{1-(1/b)}v\,(=g(x)v), \qquad \text{where } x=1-(1-t)(1-u)^{1/b}, \tag{2.3}$$

in $B_2$ is true, $x$ is a desired random number. (2.2) or (2.3) is equivalent to

$$((1-x)/(1-t))^{b-1}>v \tag{2.4}$$

or

$$(x/t)^{a-1} > v\,, \tag{2.5}$$

respectively. To avoid the time-consuming computation of power operation, we apply the squeeze method to this case. Note that

$$((1-t)^{b-1}-1)x/t+1 > (1-x)^{b-1} > (1-b)x+1\,, \qquad \text{for } 0<x\leqq t\,, \tag{2.6}$$

$$(1-t^{a-1})(x-1)/(1-t)+1 > x^{a-1} > (a-1)(x-1)+1\,, \qquad \text{for } t<x<1\,. \tag{2.7}$$

These inequalities are well used for squeezing steps.

A free parameter $t$ of this algorithm must be determined so that efficiency of the algorithm becomes maximum. Efficiency of stratified rejection method may be measured by the ratio of the expected number of rejection to sampling, $|B|/|A|$, where $|A|$ is equal to one and $|B|$ is calculated as

$$|B| = (c/a)t^a(1-t)^{b-1} + (c/b)t^{a-1}(1-t)^b\,.$$

The optimal value of $t$ is given, solving a quadratic equation, by

$$t_{\text{opt}} = \begin{cases} (a(a-1) \pm \sqrt{ab(1-a)(1-b)})/((b-a)(1-a-b))\,, & \\ & \text{if } a\neq b \text{ and } a+b\neq 1\,, \\ 1/2\,, & \text{if } a=b \text{ or } a+b=1\,. \end{cases} \tag{2.8}$$

It is possible to calculate $t_{\text{opt}}$ by the Newton method to avoid a troublesome problem which sign should be chosen for the first case. Numerical experiments show that practically reasonable approximations can be obtained by the Newton method with single iteration starting from the antimode for any combination of parameters.

Now we give the formal description of the first algorithm below.

Case 1) (Algorithm *B*00).

```
0. $t \leftarrow (1-a)/(2-a-b)$, $s \leftarrow (b-a)(1-a-b)$ and $r \leftarrow a(1-a)$.
   $t \leftarrow t-((st+2r)t-r)/2(st+r)$, $p \leftarrow t/a$, $q \leftarrow (1-t)/b$, $s \leftarrow (1-t)^{b-1}$, $c \leftarrow t^{a-1}$ and $r \leftarrow (c-1)/(t-1)$.
1. $u, v \leftarrow UR(0,1)$ and $u \leftarrow (p+q)u$. If $u>p$, then go to 3.
2. $x \leftarrow t(u/p)^{1/a}$ and $v \leftarrow sv$. If $v<(1-b)x+1$, then deliver $x$. If $v>(s-1)x/t+1$ or $v>(1-x)^{b-1}$, then go to 1, else deliver $x$.
3. $x \leftarrow 1-(1-t)((u-p)/q)^{1/b}$ and $v \leftarrow cv$. If $v<(a-1)(x-1)+1$, then deliver $x$. If $v>r(x-1)+1$ or $v>x^{a-1}$, then go to 1, else deliver $x$.
```

Step 0 should be executed once when parameters $a$ and $b$ are set new. $u \leftarrow UR(0,1)$ means to generate a uniform random number and to set to $u$. These remarks are true for the other algorithm descriptions.

### 2.3. *Algorithm for Case* 2)

Let $t$ be a real number in $(0, 1)$ and let a function $g(x)$ be defined on $(0, 1)$ by

$$g(x)=\begin{cases} cx^{a-1}, & \text{for } 0<x\leqq t, \\ ct^{a-1}(1-x)^{b-1}, & \text{for } t<x<1. \end{cases} \tag{2.9}$$

Same as 2.2, let $B$ be a region between $y=g(x)$ and $x$-axis and $B_1$ and $B_2$ be two strata of $B$ divided by the line $x=t$. Using two uniform random numbers $u$ and $v$, $(tu^{1/a}, g(tu^{1/a})v)$ or $(1-(1-t)(1-u)^{1/b}, f(t)\times(1-u)^{1-(1/b)}v)$ is a random point in $B_1$ or $B_2$, respectively. If

$$f(x)>g(x)v, \qquad \text{where } x=tu^{1/a}, \tag{2.10}$$

in $B_1$ or (2.3) in $B_2$ is satisfied, $x$ is a desired random number. (2.10) is equivalent to

$$(1-x)^{b-1}>v$$

and using

$$m_1x+1>(1-x)^{b-1}>m_2x+1, \qquad \text{for } 0<x\leqq t,$$

where

$$m_1=\max(1-b, ((1-t)^{b-1}-1)/t) \quad \text{and} \quad m_2=\min(1-b, ((1-t)^{b-1}-1)/t),$$

and (2.7) for squeezing steps, we can construct the algorithm in this case. To determine the optimal value of $t$ which minimizes $|B|$ where

$$|B|=(c/a)t^a+(c/b)t^{a-1}(1-t)^b,$$

we solve non-algebraic equation by the Newton method with the initial value $(1-a)/(b-a)$. Single iteration is sufficient to obtain the near-optimal solution.

The formal description of the second algorithm is as follows.

Case 2) (Algorithm $B01$).

```
0. $t\leftarrow(1-a)/(b-a)$, $s\leftarrow(1-t)^{b-2}$ and $r\leftarrow a-(a+b-1)t$.
   $t\leftarrow t-(t-s(1-t)(1-r)/b)/(1-sr)$, $p\leftarrow t/a$, $q\leftarrow(1-t)^{b-1}$, $s\leftarrow\min(1-b, (q-1)/t)$, $r\leftarrow\max(1-b, (q-1)/t)$, $q\leftarrow q(1-t)/b$, $c\leftarrow t^{a-1}$ and $d\leftarrow(c-1)/(t-1)$.
1. $u, v\leftarrow UR(0, 1)$ and $u\leftarrow(p+q)u$. If $u>p$, then go to 3.
2. $x\leftarrow t(u/p)^{1/a}$. If $v<sx+1$, then deliver $x$, else if $v>rx+1$ or $v>(1-x)^{b-1}$, then go to 1, else deliver $x$.
3. $x\leftarrow 1-(1-t)((u-p)/q)^{1/b}$ and $v\leftarrow cv$. If $v<(a-1)(x-1)+1$, then
```

deliver $x$. If $v>d(x-1)+1$ or $v>x^{a-1}$, then go to 1, else deliver $x$.

As stated earlier, the above algorithm is also applicable to Case 2)′: Exchange $a$ and $b$, generate $x$ according to $B01$ and transform $x$ to $1-x$.

## 2.4. *Algorithm for Case* 3)

In this case, the density function is bounded and $B$ can be chosen to be bounded. (1.1) has a single mode at

$$x=x_M=(a-1)/(a+b-2)\,.$$

If $a>2$ $(b>2)$, there is a point of inflection at $x_-$ $(x_+)$, where

$$x_-=x_M(1-\sqrt{(b-1)/(a-1)/(a+b-3)}) \quad \text{and}$$

$$x_+=x_M(1+\sqrt{(b-1)/(a-1)/(a+b-3)})\,,$$

and the left (right) tail of the density decreases faster than the exponential density.

Let $y=g(x)$ be a function defined as follows (see Fig. 1).

$$(2.11)\qquad g(x)=\begin{cases} f(x_1)\exp(r_1(x-x_1))\,, & \text{if } 0<x\leqq x_1\,,\\ m_1(x-x_2)+f(x_2)\,, & \text{if } x_1<x\leqq x_2\,,\\ m_2(x-x_2)+f(x_2)\,, & \text{if } x_2<x\leqq x_3\,,\\ f(x_M)\,, & \text{if } x_3<x\leqq x_5\,,\\ m_3(x-x_6)+f(x_6)\,, & \text{if } x_5<x\leqq x_6\,,\\ m_4(x-x_6)+f(x_6)\,, & \text{if } x_6<x\leqq x_7\,,\\ f(x_7)\exp(-r_2(x-x_7))\,, & \text{if } x_7<x<1\,, \end{cases}$$

where

$$x_2=\begin{cases} x_-\,, & \text{if } a>2\,,\\ x_M/2\,, & \text{if } a\leqq 2\,, \end{cases}$$

$$x_1=\begin{cases} x_2-f(x_2)/f'(x_2)\,, & \text{if } a>2\,,\\ 0\,, & \text{if } a\leqq 2\,, \end{cases}$$

$$m_1=\begin{cases} (f(x_2)-f(x_1))/(x_2-x_1)\,, & \text{if } a>2\,,\\ f'(x_2)\,, & \text{if } a\leqq 2\,, \end{cases}$$

$$m_2=f(x_2)/(x_2-x_1)\,,$$

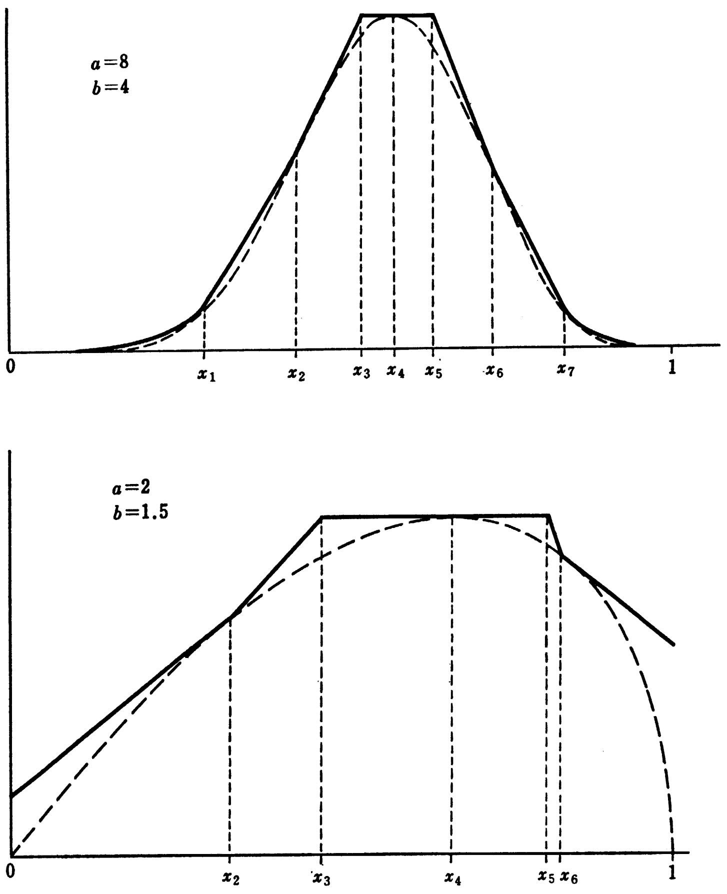

Fig. 1. $y=g(x)$

$$x_3=x_2+(f(x_M)-f(x_2))/m_2\,,$$

$$r_1=f'(x_1)/f(x_1)\,,$$

$$x_6=\begin{cases} x_+\,, & \text{if } b>2\,,\\ (1+x_M)/2\,, & \text{if } b\leqq 2\,,\end{cases}$$

$$x_7=\begin{cases} x_6-f(x_6)/f'(x_6)\,, & \text{if } b>2\,,\\ 1\,, & \text{if } b\leqq 2\,,\end{cases}$$

$$m_3=f(x_6)/(x_6-x_7)\,,$$

$$m_4=\begin{cases} (f(x_6)-f(x_7))/(x_6-x_7)\,, & \text{if } b>2\,,\\ f'(x_6)\,, & \text{if } b\leqq 2\,,\end{cases}$$

$$x_5=x_6+(f(x_M)-f(x_6))/m_3 \quad \text{and}$$

$$r_2 = -f'(x_7)/f(x_7)\,.$$

Let $B$ be a region between $y=g(x)$ and $x$-axis and let $B_j$'s be defined as follows:

$$B_1 = B \cap \{0<x<x_1\}\,,$$

$$B_2 = B \cap \{x>x_1\} \cap \{y>m_2(x-x_2)+f(x_2)\}\,,$$

$$B_3 = B \cap \{x_1<x<x_M\} \cap \{y<m_2(x-x_2)+f(x_2)\}\,,$$

$$B_4 = B \cap \{x_M<x<x_7\} \cap \{y<m_3(x-x_6)+f(x_6)\}\,,$$

$$B_5 = B \cap \{x<x_7\} \cap \{y>m_3(x-x_6)+f(x_6)\} \quad \text{and}$$

$$B_6 = B \cap \{x_7<x<1\}\,.$$

Random points in $B_1$ ($B_6$) are generated by using truncated exponential random variates. The shape of $B_2$ ($B_5$) is triangular and $B_3$ ($B_4$) trapezoid. Sampling from them is executed by using three or two uniform variates, respectively.

A squeeze method is also effective in this case using the following inequalities:

$$f(x) > (f(x_M)-f(x_2))(x-x_M)/(x_M-x_2)+f(x_M) > f(x_2)$$

in $B_3$,

$$f(x) > (f(x_M)-f(x_6))(x-x_M)/(x_M-x_6)+f(x_M) > f(x_6)$$

in $B_4$,

$$f(x) > f'(x_1)(x-x_1)+f(x_1)$$

in $B_1$ and $B_2$ if $a>2$, and

$$f(x) > f'(x_7)(x-x_7)+f(x_7)$$

in $B_5$ and $B_6$ if $b>2$.

Now we give our third algorithm.

Case 3) (Algorithm $B11$).

0. $c \leftarrow a+b-2$, $d \leftarrow c\log(c)$ and $x_4 \leftarrow (a-1)/c$. If $c>1$, then $d_0 \leftarrow \sqrt{(b-1)/(a-1)/(c-1)}$. Set $x_2$, $y_2$, $x_1$, $y_1$, $x_3$, $r_1$, $q_3$, $x_6$, $y_6$, $x_7$, $y_7$, $x_5$, $r_2$ and $q_4$ according as Table 1. $q_1 \leftarrow x_4-(x_3+x_1)/2$, $q_2 \leftarrow q_1+(x_5+x_7)/2-x_4$, $q_3 \leftarrow q_2+q_3$, $q_4 \leftarrow q_3+q_4$, $q_5 \leftarrow q_4+y_1(x_2-x_1)/2$, $q_6 \leftarrow q_5+y_7(x_7-x_6)/2$, $d_1 \leftarrow (1-y_2)/(x_4-x_2)$, $d_2 \leftarrow (1-y_6)/(x_4-x_6)$, $e_1 \leftarrow 0$ and $e_2 \leftarrow 0$.
1. $u, v \leftarrow UR(0,1)$ and $u \leftarrow q_6 u$. If $q_{j-1}<u<q_j$ (where $q_0=0$), then go to $j+1$.

Table 1. Constants for the algorithm $B11$

| | $(a>2)$ | $(a\leqq 2)$ |
|---|---|---|
| $x_2$ | $x_4(1-d_0)$ | $x_4/2$ |
| $y_2$ | $h(x_2)$ | $h(x_2)$ |
| $x_1$ | $x_2(1-d_1)$ | 0 |
| $y_1$ | $h(x_1)$ | $y_2(1-(a-1-cx_2)/(1-x_2))$ |
| $x_3$ | $x_1+x_2d_1/y_2$ | $x_2/y_2$ |
| $r_1$ | $(a-1-cx_1)/x_1/(1-x_1)$ | (not used) |
| $q_3$ | $y_1/r_1$ | 0 |
| | $(b>2)$ | $(b\leqq 2)$ |
| $x_6$ | $x_4(1+d_0)$ | $(1+x_4)/2$ |
| $y_6$ | $h(x_6)$ | $h(x_6)$ |
| $x_7$ | $x_6(1-d_2)$ | 1 |
| $y_7$ | $h(x_7)$ | $y_6(1+(a-1-cx_6)/x_6)$ |
| $x_5$ | $x_7+x_6d_2/y_6$ | $1+(x_6-1)/y_6$ |
| $r_2$ | $(cx_7-a+1)/x_7/(1-x_7)$ | (not used) |
| $q_4$ | $y_7/r_2$ | 0 |

where $h(x)=\exp(d+(a-1)\log(x/(a-1))+(b-1)\log((1-x)/(b-1)))$,
$d_1=(1-x_2)/(a-1-cx_2)$ and
$d_2=(1-x_6)/(a-1-cx_6)$.

2. $x\leftarrow x_4-2u$. If $v>(x-x_1)/(x_3-x_1)$, then $v\leftarrow 1-v$ and $x\leftarrow x_1+x_3-x$. If $v<y_2$ or $v<d_1(x-x_4)+1$, then deliver $x$, else go to 8.
3. $x\leftarrow x_4+2(u-q_1)$. If $v>(x-x_7)/(x_5-x_7)$, then $v\leftarrow 1-v$ and $x\leftarrow x_5+x_7-x$. If $v<y_6$ or $v<d_2(x-x_4)+1$, then deliver $x$, else go to 8.
4. If $e_1=0$, then $e_1\leftarrow\exp(r_1x_1)$. $w\leftarrow 1+(e_1-1)v$, $x\leftarrow(\log(w))/r_1$ and $v\leftarrow wy_1(u-q_2)/(q_3-q_2)/e_1$. Go to 6.1.
5. If $e_2=0$, then $e_2\leftarrow\exp(-r_2(1-x_7))$. $w\leftarrow 1-(1-e_2)v$, $x\leftarrow x_7-(\log(w))/r_2$ and $v\leftarrow wy_7(u-q_3)/(q_4-q_3)$. Go to 7.1.
6. $w\leftarrow UR(0,1)$, $x\leftarrow x_1+(x_2-x_1)\min(w,v)$ and $v\leftarrow(y_2(x-x_1)-y_1(x-x_2)(u-q_4)/(q_5-q_4))/(x_2-x_1)$. If $a\leqq 2$, then go to 8.

6.1. If $v<y_1(r_1(x-x_1)+1)$, then deliver $x$, else go to 8.

7. $w\leftarrow UR(0,1)$, $x\leftarrow x_7-(x_7-x_6)\min(w,v)$ and $v\leftarrow(y_6(x-x_7)-y_7(x-x_6)(u-q_5)/(q_6-q_5))/(x_6-x_7)$. If $b\leqq 2$, then go to 8.

7.1. If $v<y_7(-r_2(x-x_7)+1)$, then deliver $x$.

8. If $v>f(x)$, then go to 1, else deliver $x$.

The area of the region $B$, $|B|$, varying according to two parameters, $a$ and $b$, is a good measure of the efficiency of the algorithm and we give the values for various parameters in all cases in Table 2. According to the table, sampling efficiency is very high in almost all cases except for the case where both values of $a$ and $b$ are very small.

Table 2. Expected number of sampling

| $a \backslash b$ | 0.01 | 0.2 | 0.5 | 0.8 | 1.5 | 5 | 10 |
|---|---|---|---|---|---|---|---|
| 0.01 | 1.973 | 1.402 | 1.249 | 1.121 | 1.004 | 1.008 | 1.008 |
| 0.2 | | 1.595 | 1.365 | 1.169 | 1.063 | 1.131 | 1.145 |
| 0.5 | | | 1.273 | 1.144 | 1.112 | 1.227 | 1.251 |
| 0.8 | | | | 1.087 | 1.098 | 1.178 | 1.194 |
| 1.5 | | | | | 1.089 | 1.064 | 1.068 |
| 5 | | | | | | 1.042 | 1.045 |
| 10 | | | | | | | 1.045 |

## 3. Numerical experiments

We show some timing test results to compare several existing algorithms and to claim the superiority of our algorithms. Compared algorithms are BA, BB and BC by Cheng [5], AS134 by Atkinson et al. [4] and B4PE by Schmeiser et al. [8]. BA and BC may not work for small value(s) of parameter(s), say $\min(a, b) < 0.05$, according to overflow effect. AS134 is only applicable for the case 2) and B4PE is only applicable for the case 3).

All algorithms are coded in FORTRAN and timing tests are executed using FACOM M-200/OS-IV at Tsukuba University. For uniform random numbers, we used in-line generator of multiplicative congruential method to avoid linkage to and from a subroutine: It takes about 4 $\mu$sec. to link a subroutine and about 1 $\mu$sec. to generate one uniform random number.

Results are summarized in Table 3. All numbers are average of 25,000 values. The only algorithm competing with ours is B4PE with both parameters greater than 2. In order to obtain such high performance, we must prepare several constants depending on parameter values before generation. The time to compute these constants is called set-up time and that of each algorithm is listed in Table 4. From these two tables we conclude the followings. Roughly speaking, our new algorithms are recommended for the consecutive generation, of size at least 6 (if $a, b > 1$) or 3 (otherwise), with the same parameter values. For the case where parameter values change from time to time, BA algorithm of Cheng is preferable except for some skew cases, where BC, a time saving modification of BA, becomes efficient. Program length of each algorithm is given in Table 5. Memory requirement is of little interest at present and the difference in the table is not practically significant. Our complete subroutine program of beta random number generation consists of 180 FORTRAN statements including all 5 cases stated in Section 2.1, and it is not too big as a part of a large-scale computer simulation program.

Table 3. Timing tests

| $a$ \ $b$ | 0.01 | 0.2 | 0.5 | 0.8 | 1.5 | 5 | 10 | 100 |
|---|---|---|---|---|---|---|---|---|
| 0.01 | 37 | 27 | 24 | 22 | 20 | 20 | 20 | 20 |
| | * | * | * | * | * | * | * | * |
| | * | * | * | * | * | * | * | * |
| | — | — | — | — | — | — | — | — |
| | — | — | — | — | 32 | 33 | 32 | 33 |
| 0.2 | | 32 | 28 | 24 | 22 | 24 | 25 | 25 |
| | | 56 | 73 | 79 | 83 | 86 | 87 | 88 |
| | | 42 | 41 | 41 | 41 | 40 | 40 | 39 |
| | | — | — | — | — | — | — | — |
| | | — | — | — | 35 | 37 | 38 | 37 |
| 0.8 | | | | 23 | 23 | 26 | 27 | 30 |
| | | | | 40 | 46 | 53 | 55 | 57 |
| | | | | 37 | 38 | 41 | 41 | 42 |
| | | | | — | — | — | — | — |
| | | | | — | 36 | 39 | 40 | 40 |
| 1.5 | | | | | 15 | 13 | 14 | 15 |
| | | | | | 46 | 51 | 52 | 55 |
| | | | | | 32 | 38 | 42 | 47 |
| | | | | | 23 | 16 | 17 | 18 |
| | | | | | — | — | — | — |
| 5 | | | | | | 12 | 12 | 13 |
| | | | | | | 48 | 48 | 50 |
| | | | | | | 33 | 35 | 41 |
| | | | | | | 13 | 13 | 15 |
| | | | | | | — | — | — |
| 10 | | | | | | | 12 | 12 |
| | | | | | | | 49 | 49 |
| | | | | | | | 33 | 39 |
| | | | | | | | 13 | 15 |
| | | | | | | | — | — |
| 100 | | | | | | | | 12 |
| | | | | | | | | 49 |
| | | | | | | | | 34 |
| | | | | | | | | 13 |
| | | | | | | | | — |

(1st row : B00/B01/B11)
(2nd row: BA )
(3rd row : BB/BC )
(4th row : B4PE )
(5th row : AS134 )

Remark 1. * shows that overflow occurred in the "EXP" function in FORTRAN.
2. — shows that the algorithm is not applicable in this case.

Table 4. Set-up time

| | B00/B01/B11 | BB/BC | BA | B4PE | AS134 |
|---|---|---|---|---|---|
| $a,b<1$ | 35 | 11 | 0 | — | — |
| $a<1<b$ | 51 | 11 | 0 | — | 60~130 |
| $1<a,b<2$ | 80 | 17 | 0 | 33 | — |
| $1<a<2<b$ | 108 | 17 | 0 | 80 | — |
| $2<a,b$ | 133 | 17 | 0 | 130 | — |

Remark. — shows that the algorithm is not applicable in this case.

Table 5. Program length (a number of executable statements)

| | B00 | B01 | B11 | BA | BB | BC | B4PE | AS134 |
|---|---|---|---|---|---|---|---|---|
| set-up | 9 | 11 | 46 | 7 | 5 | 0 | 39 | 31 |
| generation | 16 | 15 | 46 | 20 | 15 | 10 | 54 | 9 |

## Acknowledgement

The author gratefully acknowledges the helpful comments of the anonymous referees which led to an improvement of the paper.

UNIVERSITY OF TSUKUBA

## REFERENCES

[1] Ahrens, J. H. and Dieter, U. (1974). Computer methods for sampling from gamma, beta, Poisson and binomial distributions, *Computing*, **12**, 223-246.

[2] Atkinson, A. C. (1979). A family of switching algorithms for the computer generation of beta random variables, *Biometrika*, **66**, 141-145.

[3] Atkinson, A. C. and Whittaker, J. (1976). A switching algorithm for the generation of beta random variables with at least one parameter less than one, *J. R. Statist. Soc.*, A, **139**, 462-467.

[4] Atkinson, A. C. and Whittaker, J. (1979). Algorithm AS134: The generation of beta random variables with one parameter greater than and one parameter less than 1, *Appl. Statist.*, **28**, 90-93.

[5] Cheng, R. C. H. (1978). Generating beta variates with nonintegral shape parameters, *Commun. Ass. Comput. Math.*, **21**, 317-322.

[6] Jöhnk, M. D. (1964). Erzeugung von betaverteilten und gamma verteilten Zufallszahlen, *Metrika*, **8**, 5-15.

[7] Marsaglia, G. (1977). The squeeze method for generating gamma variates, *Comp. Maths. Appls.*, **3**, 321-325.

[8] Schmeiser, B. W. and Babu, A. J. G. (1980). Beta variate generation via exponential majorizing functions, *Operat. Res.*, **28**, 917-926.