PAPER

Two Algorithms for Random Number Generation Implemented by Using Arithmetic of Limited Precision

Tomohiko UYEMATSU[†], *Member and* **Yuan LI**[†], *Student Member*

SUMMARY This paper presents two different algorithms for random number generation. One algorithm generates a random sequence with an arbitrary distribution from a sequence of pure random numbers, i.e. a sequence with uniform distribution. The other algorithm generates a sequence of pure random numbers from a sequence of a given i.i.d. source. Both algorithms can be regarded as an implementation of the interval algorithm by using the integer arithmetic with limited precision. We analyze the approximation error measured by the variational distance between probability distributions of the desired random sequence and the output sequence generated by the algorithms. Further, we give bounds on the expected length of input sequence per one output symbol, and compare it with that of the original interval algorithm.

key words: *random number generation, interval algorithm, arithmetic code, variational distance, limited precision*

1. Introduction

Random number generation (RNG) is one of the key issues in information theory, closely related to source coding and data compression [1],[2]. Han and Hoshi [3] proposed an extremely flexible and efficient algorithm for random number generation. Their algorithm is called *the interval algorithm*, and generates a random sequence with an arbitrary distribution from a sequence of a given independent and identically distributed (i.i.d.) source. They gave some bounds on the expected length of input sequence per one output symbol for the interval algorithm. Although the interval algorithm is general and efficient, it is based on the successive refinement of partitions of the unit interval $[0,1)$. This implies that as the length of the output or input sequences gets longer, the required precision for partitioning the unit interval becomes higher. Hence, the interval algorithm requires unlimited precision of the arithmetic, and is hard to be implemented.

The aim of this paper is to present algorithms which provide implementations of the interval algorithm by using the integer arithmetic with limited precision. We propose two algorithms for RNG. One algorithm generates a random sequence with an arbitrary distribution from a sequence of pure random numbers, i.e. i.i.d. with uniform distribution. The other algorithm generates a sequence of pure random numbers using a sequence from a given i.i.d. source. Both algorithms can be regarded as an implementation of the interval algorithm by using the integer arithmetic with limited precision. We also analyze the performance of the proposed algorithms for finite output length. First, we show bounds on the approximation error measured by the variational distance between probability distributions of the desired random sequence and the output sequence generated by the algorithms. These bounds enable us to determine a sufficient precision (word length) for a given maximum allowable approximation error. Second, we give bounds on the expected length of input sequence per one output symbol, and compare them with that of the original interval algorithm.

Manuscript received January 15, 2003.
Manuscript revised April 18, 2003.

[†]The authors are with the Department of Communications and Integrated Systems, Tokyo Institute of Technology, Tokyo 152-8552 Japan.

2. RNG using binary i.i.d. sequences with uniform distribution

In this section, we consider to generate a random sequence with a specified distribution using a sequence of pure random bits, i.e. a binary i.i.d. sequence with uniform distribution.

2.1 Algorithm I

Consider the generation of an i.i.d. sequence of fixed length n over $\mathcal{Y} = \{1, 2, \cdots, N\}$ with a specified probability distribution Q using a binary i.i.d. sequence with uniform distribution $P(1) = P(2) = 1/2$. For this RNG problem, we propose our first algorithm.

Algorithm I (generate arbitrary i.i.d. sequences)

1. Set

$$\begin{aligned} u &= 2^{w-1}, \\ F_i &= \left\lfloor \frac{1}{2} + u \sum_{j=1}^{i} Q(j) \right\rfloor \qquad i = 1, \cdots N, \\ F_0 &= 0, \\ R &= 2^{w-1}, \end{aligned}$$

and partition the interval $[0, u)$ into N disjoint subintervals $J(1), \cdots, J(N)$ such that

$$J(b) = \left[\left\lfloor \frac{2^{w-1} F_{b-1}}{u} + \frac{1}{2} \right\rfloor, \left\lfloor \frac{2^{w-1} F_b}{u} + \frac{1}{2} \right\rfloor \right),$$

$$b \in \{1, \cdots, N\},$$

where $\lfloor x \rfloor$ denotes the maximum integer smaller than or equal to x.

2. Set $m = 1$, $s = t = \lambda$ (null string), $\alpha_s = \gamma_t = 0$, $\beta_s = \delta_t = u$, $I(s) = [\alpha_s, \beta_s)$, and $J(t) = [\gamma_t, \delta_t)$.
3. Get a symbol from the input sequence. If the obtained symbol is $a \in \{1, 2\}$, then generate the subinterval $I(sa)$ of $I(s)$ such that

$$\begin{aligned} I(sa) &= [\alpha_{sa}, \beta_{sa}), \\ \alpha_{sa} &= \alpha_s + (a-1)R/2, \\ \beta_{sa} &= \alpha_s + aR/2. \end{aligned}$$

Let $R = R/2$.

4. If $I(sa)$ is entirely contained in some $J(tb)$ ($b \in \{1, \cdots, N\}$), namely,

$$\gamma_{tb} \le \alpha_{sa} < \beta_{sa} \le \delta_{tb}$$

holds for some b, then output b as the value of the m-th output Y_m, and let $t = tb$. Otherwise, go to step 6.

5. If $m = n$, the algorithm stops. Otherwise, we find the value v such that

$$2^{w-1} \le (\delta_t - \gamma_t)2^v < 2^w,$$

where $J(t) \equiv [\gamma_t, \delta_t)$. Then, let

$$Z = (\delta_t - \gamma_t)2^v,$$

and partition the interval $[0, Z)$ into N disjoint subintervals such that

$$\begin{aligned} J(tb) = \left[\left\lfloor \frac{ZF_{b-1}}{u} + \frac{1}{2} \right\rfloor, \left\lfloor \frac{ZF_b}{u} + \frac{1}{2} \right\rfloor \right), \\ b \in \{1, \cdots, N\}. \end{aligned}$$

Furthermore, update the interval $I(sa)$ and R such that (recall $\gamma_t \le \alpha_{sa} < \beta_{sa}$)

$$\begin{aligned} I(sa) &= [\alpha_{sa}, \beta_{sa}), \\ \alpha_{sa} &= (\alpha_{sa} - \gamma_t)2^v, \\ \beta_{sa} &= (\beta_{sa} - \gamma_t)2^v, \\ R &= R2^v. \end{aligned}$$

Lastly, let $m = m + 1$, and return to step 4.

6. Set $s = sa$, and return to step 3.

This algorithm is a straight-forward implementation of the interval algorithm [3]. All the variables such as $\alpha_s, \beta_s, \gamma_t, \delta_t$, and R take integer values in the range $[0, 2^w)$. Hence, the proposed algorithm can be executed by using the integer arithmetic with at most $2w+1$ bits. In this sense, we call w as the *word length*. Since the distribution of the input sequence is uniform, we can partition the interval $I(s)$ corresponding to the input sequence without error. On the contrary, the partition corresponding to the output sequence $J(t)$ is done by the same method developed by Jones [4] for the arithmetic code by employing the integer arithmetic with limited precision.

It is easy to extend the proposed algorithm for an M-ary input sequence with uniform distribution. Further, one output symbol is always generated by reading at most w symbols from an input sequence.

Remark 1: The variable u denotes the scaling factor. In the prescribed algorithm, we set $u = 2^{w-1}$. However, if possible, we should choose u such that $uQ(b)$ becomes integer for all $b \in \{1, \cdots, N\}$. If we choose such a value for u, we can decreased the upper bound on the approximation error shown in Theorem 1 by half.

Remark 2: The proposed algorithm can generate not only i.i.d. sequences but also sequences with an arbitrary probability distribution. In such a case, the partition at step 5 should be based on the conditional probability $Q(b|t)$ depending on the output sequence t generated so far (cf. [3, Remark 12]).

Example 1: Consider the case with $w = 12$ (the word length is 12 bits), $N = 3$, $Q(1) = Q(2) = Q(3) = 1/3$ (output sequences have a uniform distribution over $\{1, 2, 3\}$), $n = 2$ (the length of output sequence).
(step.1) Set $u = R = 2^{w-1} = 2048$, and

$$F_0 = 0,\ F_1 = 683,\ F_2 = 1365,\ F_3 = 2048.$$

Then, partition the interval $[0, 2048)$ into subintervals, and obtain

$$\begin{aligned} J(1) = [0, 683),\ J(2) = [683, 1365), \\ J(3) = [1365, 2048). \end{aligned}$$

(step.2) Set $m = 1$, $I(\lambda) = J(\lambda) = [0, 2048)$.
(step.3) Get a symbol from an input sequence, and suppose that we have a value 2. Then, partition the interval $I(\lambda)$, and obtain $I(2) = [1024, 2048)$. Set $R = 1024$.
(step.4) $I(2) \not\subset J(b)$ for $b \in \{1, 2, 3\}$. Go to step.6.
(step.6) Set $s = 2$ and return to step.3.
(step.3) Get a symbol from the input sequence again, and suppose that we have a value 2. Partition the interval $I(2)$ into $I(22) = [1536, 2048)$, then set $R = 512$.
(step.4) Since $I(22) \subset J(3)$, output 3 as the first output number.
(step.5) For $\gamma_3 = 1365$ and $\delta_3 = 2048$, find a value v which satisfies

$$2^{11} \le (2048 - 1365)2^v < 2^{12},$$

and obtain $v = 2$. Therefore, we have $Z = 2732$. Next, partition $[0, 2732)$ into subintervals

$$\begin{aligned} J(31) = [0, 911),\ J(32) = [911, 1821), \\ J(33) = [1821, 2732). \end{aligned}$$

Set $I(22) = [684, 2732)$, $R = 2048$, and $m = 2$.
(step.4) $I(22) \not\subset J(3b)$ for $b \in \{1, 2, 3\}$. Go to step.6.
(step.6) Let $s = 22$, and return to step.3.

(step.3) Get a symbol from the input sequence, and suppose that we have a value 1. Partition the interval $I(22)$ into $I(221) = [684, 1708)$, then set $R = 1024$.
(step.4) We have $I(221) \not\subset J(3b)$ for $b \in \{1, 2, 3\}$. Go to step.6.
(step.6) Let $s = 221$, and return to step.3.
(step.3) Get a symbol from the input sequence again, and suppose to have a value 2. Partition the interval $I(221)$ into $I(2212) = [1196, 1708)$, and let $R = 512$.
(step.4) Since $I(2212) \subset J(32)$, output 2 as the second output number.
(step.5) Since $m = n$, stop the algorithm.

2.2 Performance of Algorithm I

The performance of the RNG algorithm can be measured by two criteria [6]. One is the approximation error of output statistics measured by the variational distance between the desired distribution and the distribution of output sequences obtained by the algorithm. The other is the efficiency of the algorithm measured by the expected length of input sequence per one output symbol.

Consider to generate a sequence of fixed length n by our algorithm I, and let $\hat{Q}^n(\boldsymbol{y})$ denote the probability that the sequence $\boldsymbol{y} \in \mathcal{Y}^n$ is generated by the algorithm. Define the approximation error of the algorithm I as

$$\theta \stackrel{\triangle}{=} \sum_{\boldsymbol{y} \in \mathcal{Y}^n} |\hat{Q}^n(\boldsymbol{y}) - Q^n(\boldsymbol{y})|, \tag{1}$$

where $Q^n(\boldsymbol{y})$ is the desired probability of the sequence $\boldsymbol{y}$ given by

$$Q^n(\boldsymbol{y}) = \prod_{i=1}^{n} Q(y_i) \quad \text{for} \quad \boldsymbol{y} = y_1 \cdots y_n.$$

Then, the next theorem shows a bound on the approximation error θ.

Theorem 1: For any fixed $n > 0$, we have

$$\theta \le nN2^{-w+2}. \tag{2}$$

This theorem indicates that the approximation error is an exponential function of the word length w. Further, the approximation error θ vanishes as w tends to infinity.

Define the upper bound on the approximation error as

$$\Theta \stackrel{\triangle}{=} nN2^{-w+2}. \tag{3}$$

By taking the logarithm of both sides of (3), we obtain

$$w = 2 - \log_2 \Theta + \log_2 n + \log_2 N.$$

For a given maximum allowable approximation error, this equation enables us to determine a sufficient word length w for the algorithm I.

Proof of Theorem 1: Let $\boldsymbol{y} \in \mathcal{Y}^*$ be a finite sequence (may be null string). Just after the algorithm I outputs the last symbol of the sequence $\boldsymbol{y}$, the algorithm partitions $[0, Z)$ into N subintervals $J(\boldsymbol{y}b)$ $(b \in \mathcal{Y})$ at step 5. In this case, the conditional probability that the algorithm I outputs b as the next symbol is given as

$$\hat{Q}(b|\boldsymbol{y}) = \frac{1}{Z}\left\{\left\lfloor \frac{ZF_b}{u} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{ZF_{b-1}}{u} + \frac{1}{2} \right\rfloor\right\}.$$

Then, using the definitions of F_b and $\lfloor \cdot \rfloor$, and $2^{w-1} \le Z < 2^w$, for any $b \in \mathcal{Y}$

$$\begin{aligned}\hat{Q}(b|\boldsymbol{y}) &< \frac{F_b - F_{b-1}}{u} + 2^{-w+1} \\ &< \frac{1 + uQ(b)}{u} + 2^{-w+1} \\ &= Q(b) + 2^{-w+2}. \end{aligned} \tag{4}$$

In the same manner, for any $b \in \mathcal{Y}$

$$\hat{Q}(b|\boldsymbol{y}) > Q(b) - 2^{-w+2}. \tag{5}$$

Combining (4) and (5) yields

$$|\hat{Q}(b|\boldsymbol{y}) - Q(b)| < 2^{-w+2} \quad \text{for any } b \in \mathcal{Y}. \tag{6}$$

By using (6), we obtain

$$\begin{aligned}\theta &= \sum_{\boldsymbol{y} \in \mathcal{Y}^n} |\hat{Q}^n(\boldsymbol{y}) - Q^n(\boldsymbol{y})| \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}^{n-1}} \sum_{b \in \mathcal{Y}} |\hat{Q}^n(\boldsymbol{y}b) - Q^n(\boldsymbol{y}b)| \\ &\le \sum_{\boldsymbol{y} \in \mathcal{Y}^{n-1}} \sum_{b \in \mathcal{Y}} \Big\{ |\hat{Q}(b|\boldsymbol{y})\hat{Q}^{n-1}(\boldsymbol{y}) - Q(b)\hat{Q}^{n-1}(\boldsymbol{y})| \\ &\quad + |Q(b)\hat{Q}^{n-1}(\boldsymbol{y}) - Q(b)Q^{n-1}(\boldsymbol{y})| \Big\} \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}^{n-1}} \sum_{b \in \mathcal{Y}} |\hat{Q}(b|\boldsymbol{y}) - Q(b)|\hat{Q}^{n-1}(\boldsymbol{y}) \\ &\quad + \sum_{\boldsymbol{y} \in \mathcal{Y}^{n-1}} \sum_{b \in \mathcal{Y}} Q(b)|\hat{Q}^{n-1}(\boldsymbol{y}) - Q^{n-1}(\boldsymbol{y})| \\ &\le N2^{-w+2} \sum_{\boldsymbol{y} \in \mathcal{Y}^{n-1}} \hat{Q}^{n-1}(\boldsymbol{y}) \\ &\quad + \sum_{\boldsymbol{y} \in \mathcal{Y}^{n-1}} |\hat{Q}^{n-1}(\boldsymbol{y}) - Q^{n-1}(\boldsymbol{y})| \\ &= N2^{-w+2} + \sum_{\boldsymbol{y} \in \mathcal{Y}^{n-1}} |\hat{Q}^{n-1}(\boldsymbol{y}) - Q^{n-1}(\boldsymbol{y})|. \end{aligned} \tag{7}$$

By repeating the same derivation for the second term in the right hand side of (7), we obtain (2). Q.E.D.

The next theorem shows bounds on the expected length of the input sequence to generate a sequence with length n by the proposed algorithm I.

Theorem 2: Let $E(L)$ be the expected length L of input sequence to generate a sequence with length n by using the proposed algorithm I. If θ defined in (1) satisfies $\theta < 1/2$, then

$$nH(Q) - \delta(\theta, n) \leq E(L) \leq nH(Q) + 3 + \delta(\theta, n), \quad (8)$$

where

$$\delta(\theta, n) \stackrel{\triangle}{=} -\theta \log_2 \theta + n\theta \log_2 N, \quad (9)$$

and $H(Q)$ is the entropy of the desired distribution Q.

Before we prove Theorem 2, we show some useful lemmas.

Lemma 1:[5, Lemma 2.7] If two distributions Q_1 and Q_2 over a finite alphabet $\mathcal{Z}$ satisfies

$$\hat{\theta} \stackrel{\triangle}{=} \sum_{a \in \mathcal{Z}} |Q_1(a) - Q_2(a)| \leq 1/2,$$

then we have

$$|H(Q_1) - H(Q_2)| \leq -\hat{\theta} \log_2(\hat{\theta}/|\mathcal{Z}|), \quad (10)$$

where $|\mathcal{Z}|$ denotes the cardinality of $\mathcal{Z}$.

Lemma 2:[3, Theorem 1 and Remark 8] If we employ the original interval algorithm to generate one symbol with the distribution Q using the sequence of pure random bits, the expected length $E(\hat{L})$ of the input sequence can be bounded as

$$H(Q) \leq E(\hat{L}) \leq H(Q) + 3.$$

Proof of Theorem 2: Since $\theta < 1/2$, we have

$$\sum_{\boldsymbol{y} \in \mathcal{Y}^n} |\hat{Q}^n(\boldsymbol{y}) - Q^n(\boldsymbol{y})| = \theta < 1/2.$$

Using Lemma 1 for the distributions $\hat{Q}^n$ and Q^n over $\mathcal{Y}^n$, we have

$$|nH(Q) - H(\hat{Q}^n)| \leq -\theta \log_2(\theta/N^n) = \delta(\theta, n), \quad (11)$$

where $H(\hat{Q}^n)$ denotes the entropy of the distribution $\hat{Q}^n$. Since our proposed algorithm I can be regarded as an ideal interval algorithm for the distribution $\hat{Q}^n$, according to Lemma 2, we have

$$H(\hat{Q}^n) \leq E(L) \leq H(\hat{Q}^n) + 3.$$

Substituting (11) into the above inequalities, we obtain (8). Q.E.D.

Comparing (8) and Lemma 2, the difference of expected lengths of input sequences between the proposed algorithm I and the original interval algorithm is at most $3 + \delta(\theta, n)$. Further, according to Theorem 1, for fixed n, (8) tends to

$$nH(Q) \leq E(L) \leq nH(Q) + 3, \quad \text{as } w \to \infty.$$

These bounds are identical with the bounds on the expected length $E(\hat{L})$ of the input sequence for the original interval algorithm described in Lemma 2 (substituting Q^n instead of Q).

From Theorem 1 and Theorem 2, we can conclude that the performance of the algorithm I approaches to that of the original interval algorithm, as the word length w tends to infinity.

Remark 3: Close look of the proofs shows that even if the output sequence has any probability distribution, i.e. Markov or stationary distribution, both Theorem 1 and Theorem 2 are valid.

3. RNG of binary i.i.d. sequences with uniform distribution

In this section, we deal with the inverse problem treated in Section 2. Namely, consider the generation of an i.i.d. sequence of length n with a uniform distribution using an i.i.d. sequence with a given distribution P. We propose our second algorithm, and analyze its performance.

3.1 Algorithm II

Consider the generation of an i.i.d. sequence of uniform distribution of length n over $\mathcal{Y} = \{1, 2\}$ using an i.i.d. input sequence over $\mathcal{X} = \{1, 2, \cdots, M\}$ with probability P. For this RNG problem, we propose our second algorithm which can also be implemented by using the integer arithmetic with finite precision.

Algorithm II (generate i.i.d. sequences with uniform distribution)

1. Set

$$\begin{aligned} u &= 2^{w-1}, \\ F_0 &= 0, \\ F_i &= \left\lfloor \frac{1}{2} + u \sum_{j=1}^{i} P(j) \right\rfloor, \ i = 1, \cdots, M \\ Z &= R = 2^{w-1}, \end{aligned}$$

and partition the interval $[0, u)$ into two disjoint subintervals $J(1)$ and $J(2)$ such that

$$J(1) = [0, R/2), \ J(2) = [R/2, u).$$

2. Set $m = 1$, $s = t = \lambda$ (null string), $\alpha_s = \gamma_t = 0$, $\beta_s = \delta_t = u$, $I(s) = [\alpha_s, \beta_s)$, and $J(t) = [\gamma_t, \delta_t)$. Partition the interval $I(s)$ such that

$$\begin{aligned} I(sa) &= [\alpha_{sa}, \beta_{sa}), \\ \alpha_{sa} &= \left\lfloor \frac{ZF_{a-1}}{u} + \frac{1}{2} \right\rfloor, \ \beta_{sa} = \left\lfloor \frac{ZF_a}{u} + \frac{1}{2} \right\rfloor, \\ &\qquad \text{for} \quad a \in \{1, \cdots, M\}. \end{aligned}$$

Let $R = R/2$.

3. Get a symbol from the input sequence. If we obtain a value $a \in \mathcal{X}$, then choose the subinterval $I(sa)$ of $I(s)$.
4. If $I(sa)$ is entirely contained in some $J(tb) = [\gamma_{tb}, \delta_{tb})$ $(b \in \{1,2\})$, namely,

$$\gamma_{tb} \le \alpha_{sa} < \beta_{sa} \le \delta_{tb}$$

holds for some b, then output b as the m-th output Y_m, and let $t = tb$. Otherwise, go to step 6.

5. If $m = n$, the algorithm stops. Otherwise, let $m = m + 1$, and partition the interval $J(t)$ as follows.

a. $\delta_t - \gamma_t = R$:
Partition $J(t)$ into $J(t1)$ and $J(t2)$ such that

$$J(t1) = [\gamma_t, \gamma_t + R/2)\,,$$
$$J(t2) = [\gamma_t + R/2, \delta_t)\,.$$

b. $R/2 < \delta_t - \gamma_t < R$:
i If $\gamma_t = 0$, the lower bound of the subinterval $J(t1)$ may be smaller than 0. Hence, partition $J(t)$ into $J(t1)$ and $J(t2)$ such that

$$J(t1) = [0, \delta_t - R/2)\,,$$
$$J(t2) = [\delta_t - R/2, \delta_t)\,.$$

ii If $\delta_t = Z$, the upper bound of the subinterval $J(t2)$ may be greater than Z. Hence, partition $J(t)$ into $J(t1)$ and $J(t2)$ such that

$$J(t1) = [\gamma_t, \gamma_t + R/2)\,,$$
$$J(t2) = [\gamma_t + R/2, \delta_t)\,.$$

c. $0 < \delta_t - \gamma_t \le R/2$:
i If $\gamma_t = 0$, the upper bound of the subinterval $J(t1)$ is smaller than 0. Hence, we omit the partition and let

$$J(t2) = [\gamma_t, \delta_t)\,.$$

ii If $\delta_t = Z$, the lower bound of the subinterval $J(t2)$ is greater than Z. Hence, we omit the partition and let

$$J(t1) = [\gamma_t, \delta_t)\,.$$

Let $R = R/2$ and return to step 4.

6. Set $s = sa$, and find the value v such that

$$2^{w-1} \le (\beta_s - \alpha_s)2^v < 2^w,$$

where $I(s) \equiv [\alpha_s, \beta_s)$. Then, let

$$Z = (\beta_s - \alpha_s)2^v,$$

and partition the interval $[0, Z)$ into M disjoint subintervals such that

$$I(sa) = [\alpha_{sa}, \beta_{sa}),$$
$$\alpha_{sa} = \left\lfloor \frac{ZF_{a-1}}{u} + \frac{1}{2} \right\rfloor,\ \beta_{sa} = \left\lfloor \frac{ZF_a}{u} + \frac{1}{2} \right\rfloor,$$
$$\text{for} \quad a \in \{1, \cdots, M\}.$$

Further, update the subinterval $J(tb)$ and R such that

$$J(tb) = [\gamma_{tb}, \delta_{tb}), \quad \text{for} \quad b = 1, 2,$$
$$\gamma_{tb} = \begin{cases} (\gamma_{tb} - \alpha_s)2^v & \text{if} \quad \alpha_s < \gamma_{tb} \\ 0 & \text{if} \quad \gamma_{tb} < \alpha_s \le \delta_{tb} \end{cases}$$
$$\beta_{tb} = \begin{cases} (\delta_{tb} - \alpha_s)2^v & \text{if} \quad \delta_{tb} < \beta_s \\ Z & \text{if} \quad \gamma_{tb} < \beta_s \le \delta_{tb} \end{cases}$$
$$R = 2^v R,$$

and return to step 3.

It should be noted here that in step 5, lower bound or upper bound of the subinterval $J(t1)$ or $J(t2)$ may become outside the interval $[0, Z)$. Hence, in step 5, we need complicated partitioning of the interval.

Example 2: Consider the generation of random sequence of length $n = 3$ with uniform distribution from the output sequence of the source with the distribution $P(1) = 1/5$, $P(2) = P(3) = 2/5$. We assume the word length $w = 5$.
(step.1) Set $u = R = Z = 2^{w-1} = 16$. Then, we have

$$F_0 = 0, \quad F_1 = 3, \quad F_2 = 10, \quad F_3 = 16.$$

We also have

$$J(1) = [0, 8)\,, \quad J(2) = [8, 16)\,.$$

(step.2) Set $I(\lambda) = J(\lambda) = 16$, and $m = 1$. Partition the interval $I(\lambda)$, and obtain

$$I(1) = [0, 3)\,, \quad I(2) = [3, 10)\,, \quad I(3) = [10, 16)\,.$$

Let $R = 8$.
(step.3) Get a symbol from an input sequence and suppose that we have a value 2.
(step.4) $I(2) \not\subset J(b)$ for $b \in \{1, 2\}$. Go to step 6.
(step.6) For $\alpha_2 = 3$ and $\beta_2 = 10$, we find an integer v such that

$$2^4 \le (\beta_2 - \alpha_2)2^v < 2^5,$$

and obtain $v = 2$. Then, let $Z = (\beta_2 - \alpha_2)2^v = 28$, partition the new interval $[0, 28)$, and obtain

$$I(21) = [0, 6)\,, \quad I(22) = [6, 17)\,, \quad I(23) = [17, 28)\,.$$

Update the interval $J(b)$ $(b \in \{1, 2\})$ such that

$$J(1) = [0, 20)\,, \quad J(2) = [20, 28)\,,$$

and set $R = 32$. (Here, the true interval should be $J(1) = [-12, 20)$ and $J(2) = [20, 52)$.)
(step.3) Get a symbol from the input sequence, and suppose that we have a value 2.
(step.4) Since $I(22) \subset J(1)$, output 1 as the first output number. Let $t = 1$.
(step.5) Let $m = 2$ and partition the interval $J(1)$ into

subintervals

$$J(11) = [0, 4), \quad J(12) = [4, 20),$$

and let $R = 16$.
(step.4) Since $I(22) \subset J(12)$, output 2 as the second output number. Let $t = 12$.
(step.5) Let $m = 3$ and partition the interval $J(12)$ into subintervals

$$J(121) = [4, 12), \quad J(122) = [12, 20).$$

Let $R = 8$.
(step.4) $I(22) \not\subset J(12b)$ for $b \in \{1, 2\}$. Go to step 6.
(step.6) For $\alpha_{22} = 6$, and $\beta_{22} = 17$, find an integer such that

$$2^4 \leq (\beta_{22} - \alpha_{22})2^v < 2^5,$$

and obtain $v = 1$. Then, let $Z = (\beta_{22} - \alpha_{22})2^v = 22$, and partition the new interval $[0, 22)$ into subintervals

$$I(221) = [0, 5), \quad I(222) = [5, 13),$$
$$I(223) = [13, 22).$$

Update the interval $J(12b)$ $(b \in \{1, 2\})$ such that

$$J(121) = [0, 12), \quad J(122) = [12, 22),$$

Let $R = 16$.
(step.3) Get a symbol from the input sequence, and suppose that we have a value 1.
(step.4) Since $I(221) \subset J(121)$, output 1 as the third output number. Let $t = 121$.
(step.5) Since $m = n = 3$, stop the algorithm.

3.2 Performance of Algorithm II

The next theorem shows a bound on the approximation error of the proposed algorithm due to the arithmetic of limited precision.

Theorem 3: For any fixed $n > 0$, let $\hat{Q}^n(\boldsymbol{y})$ denote the probability that the output sequence $\boldsymbol{y} \in \mathcal{Y}^n$ is generated by the proposed algorithm II. Then,

$$\sum_{\boldsymbol{y} \in \mathcal{Y}^n} |\hat{Q}^n(\boldsymbol{y}) - 2^{-n}| \leq \left(\frac{1}{1 - p_{max}} + 2 \left\lceil \frac{-n}{\log_2(p_{max} + 2^{-w+2})} \right\rceil \right) M 2^{-w+2}, \quad (12)$$

where $\lceil x \rceil$ denotes the minimum integer greater than or equal to x, and

$$p_{max} = \max_{a \in \mathcal{X}} P(a). \quad (13)$$

Similar to the proposed algorithm I, this theorem indicates that the approximation error of the proposed algorithm II is an exponential function of the word length w. Further, the approximation error vanishes as the word length w tends to infinity. For a given maximum allowable approximation error, (12) enables us to determine a sufficient word length w for the algorithm II.

Before we prove the theorem, it is convenient to describe the algorithm II in terms of a generating tree T introduced in [3]. The generating tree T is a M-ary tree (possibly of infinite size), and can be characterized by the following properties, where the terms *nodes* and *leaves* denote internal nodes and terminal nodes, respectively.

1. The tree T is complete, i.e. every node has M children and those M branches that connect the node to the M children are labeled as $1, 2, \cdots, M$ in the order from left to right.
2. Each leaf of the tree T is labeled with one of the sequence in $\mathcal{Y}^n$ (the same label may be assigned to several leaves).

We can describe the algorithm II in terms of a generating tree T as follows. Every node (or leaf) is uniquely denoted by the sequence $\boldsymbol{x}(= x_1 x_2 \cdots x_k)$ where $x_1, \cdots, x_k$ are the labels of branches on the path from the root to the node (or leaf) $\boldsymbol{x}$. Each node (or leaf) $\boldsymbol{x}$ of the tree T corresponds one-to-one to the subinterval $I(\boldsymbol{x})$. A node $\boldsymbol{x}$ associated with the subinterval $I(\boldsymbol{x})$ has M children $\boldsymbol{x}a$ $(a \in \mathcal{X})$ that are associated with the subintervals $I(\boldsymbol{x}a)$, respectively. Moreover, a subinterval $I(\boldsymbol{x})$ is terminating, i.e. the algorithm II stops just after reading the input sequence $\boldsymbol{x}$, if and only if the corresponding $\boldsymbol{x}$ is a leaf of the tree T.

Given a generating tree T as above, the algorithm for random number generation is carried out as follows. Starting at the root of the tree we see the input symbol and let the result be $a \in \{1, \cdots, M\}$. Then we proceed along the branch labeled a to reach a new node or a leaf. If it is a node we continue to see the next input symbol and repeat the same process; otherwise, we output the label assigned to the leaf and stop the algorithm. It is easy to see this RNG algorithm is equivalent to the algorithm II.

Define $\mathcal{T}$ as the set of leaves, and $\mathcal{T}(\boldsymbol{y})$ as the set of leaves with its label $\boldsymbol{y} \in \mathcal{Y}^n$. Hence, $\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y})$ implies that the algorithm II generates the output $\boldsymbol{y}$ from any input sequence with its prefix $\boldsymbol{x}$. Further, define $\hat{\mathcal{T}}$ as the set of nodes.

Next, for any integer $t \geq 0$, any sequence $\boldsymbol{x} = x_1 \cdots x_t \in \mathcal{X}^t$, and any $a \in \mathcal{X}$, define the conditional probability $\hat{P}(a|\boldsymbol{x})$ and the probability $\hat{P}(\boldsymbol{x})$ as

$$\hat{P}(a|\boldsymbol{x}) \stackrel{\triangle}{=} \frac{||I(\boldsymbol{x}a)||}{\sum_{a' \in \mathcal{X}} ||I(\boldsymbol{x}a')||}, \quad (14)$$

$$\hat{P}(\boldsymbol{x}) \stackrel{\triangle}{=} \prod_{i=1}^{t} \hat{P}(x_i|x_1 \cdots x_{i-1}), \quad (15)$$

where $||I(\boldsymbol{x}a)||$ denotes the width of the interval $I(\boldsymbol{x}a)$

determined at step.2 or step.6 of the algorithm. $\hat{P}(\boldsymbol{x})$ means the approximated occurrence probability of $\boldsymbol{x}$ in the algorithm II. It should be noted that even if the input sequence $\boldsymbol{x} = x_1 \cdots x_t$ is an output of an i.i.d. source, $\hat{P}(\boldsymbol{x})$ is in general a distribution with finite state. This makes the analysis of the algorithm II hard.

Here, we describe some lemmas needed to prove Theorem 3.

Lemma 3: For any integer $t > 0$, and a generating tree T of the proposed algorithm II, we have

$$\sum_{\boldsymbol{y} \in \mathcal{Y}^n} \Big| \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ \ell(\boldsymbol{x}) \le t}} (\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})) \Big| \le \sum_{\boldsymbol{x} \in \mathcal{X}^t} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})|, \tag{16}$$

where $\ell(\boldsymbol{x})$ denotes the length of the sequence $\boldsymbol{x} \in \mathcal{X}^*$, and $P(\boldsymbol{x})$ is the occurrence probability of the input sequence $\boldsymbol{x}$, i.e.

$$P(\boldsymbol{x}) = \prod_{i=1}^{t} P(x_i) \quad \text{for} \quad \boldsymbol{x} = x_1 \cdots x_t.$$

Proof of Lemma 3: First, by using the triangular inequality, we have

$$\begin{aligned} &\sum_{\boldsymbol{y} \in \mathcal{Y}^n} \Big| \sum_{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \ell(\boldsymbol{x}) \le t} (\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})) \Big| \\ &\le \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \sum_{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \ell(\boldsymbol{x}) \le t} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\ &= \sum_{\boldsymbol{x} \in \mathcal{T}: \ell(\boldsymbol{x}) \le t} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})|. \end{aligned} \tag{17}$$

According to (14), we have

$$\sum_{a \in \mathcal{X}} \hat{P}(a|\boldsymbol{x}) = 1.$$

Hence, for any finite sequence $\boldsymbol{x} \in \mathcal{X}^*$, we have

$$\begin{aligned} &|\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\ &= \Big| \hat{P}(\boldsymbol{x}) \sum_{a \in \mathcal{X}} \hat{P}(a|\boldsymbol{x}) - P(\boldsymbol{x}) \sum_{a \in \mathcal{X}} P(a) \Big| \\ &\le \sum_{a \in \mathcal{X}} |\hat{P}(\boldsymbol{x}a) - P(\boldsymbol{x}a)|. \end{aligned} \tag{18}$$

By using (18) repeatedly, we obtain

$$\begin{aligned} \sum_{\substack{\boldsymbol{x} \in \mathcal{T}: \\ \ell(\boldsymbol{x}) \le t}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| &= \sum_{k=1}^{t} \sum_{\substack{\boldsymbol{x} \in \mathcal{T}: \\ \ell(\boldsymbol{x}) = k}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\ &\le \sum_{\boldsymbol{x} \in \mathcal{X}^t} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})|. \end{aligned} \tag{19}$$

Combining (17) and (19), we obtain (16). Q.E.D.

Lemma 4: For any sequence $\boldsymbol{x} \in \mathcal{X}^*$ and any $a \in \mathcal{X}$,

$$|\hat{P}(a|\boldsymbol{x}) - P(a)| \le 2^{-w+2}. \tag{20}$$

Further, for any integer $t > 0$, we have

$$\sum_{\boldsymbol{x} \in \mathcal{X}^t} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \le tM2^{-w+2}. \tag{21}$$

This lemma can be proven in a similar manner as the proof of Theorem 1. So, we omit the proof.

Lemma 5: For any sequence $\boldsymbol{x} \in \mathcal{X}^*$,

$$\begin{aligned} &\sum_{a \in \mathcal{X}} |\hat{P}(\boldsymbol{x}a) - P(\boldsymbol{x}a)| \\ &\le |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| + M2^{-w+2}(p_{max})^{\ell(\boldsymbol{x})}, \end{aligned} \tag{22}$$

where p_{max} is defined in (13).

Proof of Lemma 5:

$$\begin{aligned} &\sum_{a \in \mathcal{X}} |\hat{P}(\boldsymbol{x}a) - P(\boldsymbol{x}a)| \\ &\le \sum_{a \in \mathcal{X}} \hat{P}(a|\boldsymbol{x}) |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\ &\quad + \sum_{a \in \mathcal{X}} |\hat{P}(a|\boldsymbol{x}) - P(a)| P(\boldsymbol{x}) \\ &\le |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| + M2^{-w+2}(p_{max})^{\ell(\boldsymbol{x})}, \end{aligned} \tag{23}$$

where the last inequality comes from (20) and $P(\boldsymbol{x}) \le (p_{max})^{\ell(\boldsymbol{x})}$. Q.E.D.

Lemma 6: For any sequence $\boldsymbol{y} \in \mathcal{Y}^n$,

$$\sum_{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y})} \hat{P}(\boldsymbol{x}) = 2^{-n}, \tag{24}$$

where the distribution $\hat{P}$ is defined in (14).

Proof of Lemma 6: Consider the original interval algorithm which generates i.i.d. sequence with uniform distribution over $\mathcal{Y}^n$ by using the input sequence with the distribution $\hat{P}$. For the original interval algorithm, let $\hat{I}(\boldsymbol{x})$ and $\hat{J}(\boldsymbol{y})$ be the intervals corresponding to the input sequence $\boldsymbol{x} \in \mathcal{X}^*$ and output sequence $\boldsymbol{y} \in \mathcal{Y}^n$, respectively. Since the scaling and translation in step.6 of the algorithm II does not affect the inclusion relationship between intervals, we have

$$I(\boldsymbol{x}) \subset J(\boldsymbol{y}) \Longleftrightarrow \hat{I}(\boldsymbol{x}) \subset \hat{J}(\boldsymbol{y}).$$

This implies that any input sequence $\boldsymbol{x}$ which generates $\boldsymbol{y} \in \mathcal{Y}^n$ by the algorithm II also generates the same $\boldsymbol{y}$ by the original interval algorithm, and vice versa. Therefore, according to [3, (4.15)], for any $\boldsymbol{y} \in \mathcal{Y}^n$, we have

$$2^{-n} = \|\hat{J}(\boldsymbol{y})\| = \sum_{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y})} \|\hat{I}(\boldsymbol{x})\| = \sum_{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y})} \hat{P}(\boldsymbol{x}),$$

which completes the proof. Q.E.D.

Proof of Theorem 3: In what follows, fix t as

$$t = \lceil -n/\log_2(p_{max} + 2^{-w+2}) \rceil. \tag{25}$$

Then, according to (20), for any $\boldsymbol{x} \in \mathcal{X}^t$, we have

$$\hat{P}(\boldsymbol{x}) \le (p_{max} + 2^{-w+2})^t \le 2^{-n}. \tag{26}$$

Let θ be the approximation error of the algorithm II. Since

$$\hat{Q}^n(\boldsymbol{y}) = \sum_{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y})} P(\boldsymbol{x}). \tag{27}$$

for any $\boldsymbol{y} \in \mathcal{Y}^n$, combining (27), Lemma 3 and Lemma 6, we have

$$\begin{aligned}
\theta &= \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \left| \hat{Q}^n(\boldsymbol{y}) - 2^{-n} \right| \\
&= \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \left| \sum_{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y})} P(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y})} \hat{P}(\boldsymbol{x}) \right| \\
&\le \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \left| \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ \iota(\boldsymbol{x}) \le t}} (\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})) \right| \\
&\quad + \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \left| \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ \iota(\boldsymbol{x}) > t}} (\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})) \right| \\
&\le \sum_{\boldsymbol{x} \in \mathcal{X}^t} \left| \hat{P}(\boldsymbol{x}) - P(\boldsymbol{x}) \right| \\
&\quad + \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \left| \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ \iota(\boldsymbol{x}) > t}} (\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})) \right|.
\end{aligned} \tag{28}$$

Next, we bound the second summation in the right hand side of (28). For any integer $K > 0$, define $S(K)$ as

$$S(K) \stackrel{\triangle}{=} \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \left| \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ t+1 \le \iota(\boldsymbol{x}) \le K}} (\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})) \right|.$$

If $x_1 \cdots x_K$ is a leaf, $x_1 \cdots x_{K-1}$ must be a node. Hence,

$$\begin{aligned}
S(K) &\le \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \left\{ \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ t+1 \le \iota(\boldsymbol{x}) \le K-1}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \right. \\
&\qquad \left. + \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ \iota(\boldsymbol{x}) = K}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \right\} \\
&\le \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ t+1 \le \iota(\boldsymbol{x}) \le K-1}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + \sum_{\substack{\boldsymbol{x} \in \hat{\mathcal{T}}: \\ \iota(\boldsymbol{x}) = K-1}} \sum_{a \in \mathcal{X}} |\hat{P}(\boldsymbol{x}a) - P(\boldsymbol{x}a)|.
\end{aligned}$$

For the generating tree T corresponding to the algorithm II, let a_i denote the number of nodes at level i of the tree T. Then, according to Lemma 5, we have

$$\begin{aligned}
S(K) &\le \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ t+1 \le \iota(\boldsymbol{x}) \le K-1}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + \sum_{\substack{\boldsymbol{x} \in \hat{\mathcal{T}}: \\ \iota(\boldsymbol{x}) = K-1}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + a_{K-1} M 2^{-w+2} (p_{max})^{K-1} \\
&= \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ t+1 \le \iota(\boldsymbol{x}) \le K-2}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + \sum_{\substack{\boldsymbol{x} \in \hat{\mathcal{T}}: \\ \iota(\boldsymbol{x}) = K-2}} \sum_{a \in \mathcal{X}} |\hat{P}(\boldsymbol{x}a) - P(\boldsymbol{x}a)| \\
&\quad + a_{K-1} M 2^{-w+2} (p_{max})^{K-1} \\
&\le \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \sum_{\substack{\boldsymbol{x} \in \mathcal{T}(\boldsymbol{y}): \\ t+1 \le \iota(\boldsymbol{x}) \le K-2}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + \sum_{\substack{\boldsymbol{x} \in \hat{\mathcal{T}}: \\ \iota(\boldsymbol{x}) = K-2}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + (a_{K-2}(p_{max})^{K-2} + a_{K-1}(p_{max})^{K-1}) M 2^{-w+2}.
\end{aligned}$$

By repeating the last deviation $K - t + 2$ times, we obtain

$$\begin{aligned}
S(K) &\le \sum_{\substack{\boldsymbol{x} \in \hat{\mathcal{T}}: \\ \iota(\boldsymbol{x}) = t}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + M 2^{-w+2} \sum_{i=t}^{K-1} a_i (p_{max})^i \\
&< \sum_{\substack{\boldsymbol{x} \in \hat{\mathcal{T}}: \\ \iota(\boldsymbol{x}) = t}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + \frac{(p_{max})^t M 2^{n-w+2}}{1 - p_{max}} \\
&< \sum_{\substack{\boldsymbol{x} \in \hat{\mathcal{T}}: \\ \iota(\boldsymbol{x}) = t}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| + \frac{M 2^{-w+2}}{1 - p_{max}},
\end{aligned}$$

where the second inequality comes from the relation $a_i \le 2^n - 1$ $(i = 1, 2, \cdots)$, and the last inequality comes from (25). Substituting this into (28) and using Lemma 4, we have

$$\begin{aligned}
\theta &\le \sum_{\boldsymbol{x} \in \mathcal{X}^t} \left| \hat{P}(\boldsymbol{x}) - P(\boldsymbol{x}) \right| + \lim_{K \to \infty} S(K) \\
&\le \sum_{\boldsymbol{x} \in \mathcal{X}^t} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| \\
&\quad + \sum_{\substack{\boldsymbol{x} \in \hat{\mathcal{T}}: \\ \iota(\boldsymbol{x}) = t}} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| + \frac{M 2^{-w+2}}{1 - p_{max}} \\
&\le 2 \sum_{\boldsymbol{x} \in \mathcal{X}^t} |\hat{P}(\boldsymbol{x}) - P(\boldsymbol{x})| + \frac{M 2^{-w+2}}{1 - p_{max}}
\end{aligned}$$

$$\leq 2tM2^{-w+2} + \frac{M2^{-w+2}}{1-p_{max}}. \quad (29)$$

Again substituting (25) into (29), we finally obtain

$$\theta \leq \left(\frac{1}{1-p_{max}} + 2 \left\lceil \frac{-n}{\log_2(p_{max}+2^{-w+2})} \right\rceil \right) \times M2^{-w+2}, \quad (30)$$

which completes the proof of Theorem 3. Q.E.D.

The next theorem shows bounds on the expected number $E(L)$ of input length per one output symbol for the algorithm II.

Theorem 4: Let L be the required length of input sequence to generate a sequence of length n by using the algorithm II. Then, if $M2^{-w+2} \leq 1/2$, the expected value of L can be bounded as

$$\frac{n}{H(P)+\hat{\delta}} \leq E(L) \leq \frac{n}{H(P)-\hat{\delta}} + \frac{w+\log_2(M-1)}{H(P)-\hat{\delta}} + \frac{1}{(1-p_{max}-2^{-w+2})(H(P)-\hat{\delta})} \quad (31)$$

where $\hat{\delta} \stackrel{\triangle}{=} (w-2)M2^{-w+2}$.

Before we prove the theorem, we need some definitions. Let $P(a|s)$ $(a \in \mathcal{X}, s \in \mathcal{S})$ denote the probability distribution of the finite-state source. Then, we define $H_{max}(P)$ and $H_{min}(P)$ as

$$H_{max}(P) \stackrel{\triangle}{=} \max_{s \in \mathcal{S}} H(P(\cdot|s)),$$
$$H_{min}(P) \stackrel{\triangle}{=} \min_{s \in \mathcal{S}} H(P(\cdot|s)).$$

Further, we define

$$\hat{p}_{max} \stackrel{\triangle}{=} \max_{a \in \mathcal{X}, s \in \mathcal{S}} P(a|s). \quad (32)$$

The next lemma shows bounds on the expected length of the input sequence for the original interval algorithm.

Lemma 7: Consider to generate a binary i.i.d. sequence of length n with uniform distribution from a finite-state source over $\mathcal{X}$ with a set of state $\mathcal{S}$ by using the original interval algorithm. Then, the expected length $E(\hat{L})$ of the input sequence is bounded as

$$\frac{n}{H_{max}(P)} \leq E(\hat{L}) \leq \frac{n}{H_{min}(P)} + \frac{\log_2(2|\mathcal{S}|(M-1))}{H_{min}(P)} + \frac{1}{(1-\hat{p}_{max})H_{min}(P)}. \quad (33)$$

This lemma is a simple extension of [3, Theorem 4], and can be proven in a similar manner. So, we omit the proof.

Proof of Theorem 4: The approximated input distribution $\hat{P}$ defined by (14) can be regarded as a distribution of a finite state source with its state determined by the width of total intervals, i.e. with a set of states $\mathcal{S} = \{2^{w-1}, 2^{w-1}+1, \cdots, 2^w - 1\}$.

Since the number of states in $\mathcal{S}$ is 2^{w-1}, by using Lemma 7, we immediately have

$$\frac{n}{H_{max}(\hat{P})} \leq E(L) \quad (34)$$

$$\leq \frac{n}{H_{min}(\hat{P})} + \frac{w+\log_2(M-1)}{H_{min}(\hat{P})} + \frac{1}{(1-\hat{p}_{max})H_{min}(\hat{P})}. \quad (35)$$

On the other hand, from Lemma 4 we have

$$\sum_{a \in \mathcal{X}} |\hat{P}(a|\boldsymbol{x}) - P(a)| \leq M2^{-w+2} \quad (36)$$

for any sequence $\boldsymbol{x} \in \mathcal{X}^*$. Since the sequence $\boldsymbol{x}$ can determine the state, combining this with Lemma 1, if $M2^{-w+2} \leq 1/2$, we have

$$|H_{min}(\hat{P}) - H(P)| \leq (w-2)M2^{-w+2} = \hat{\delta}. \quad (37)$$

This implies that

$$H_{min}(\hat{P}) \geq H(P) - \hat{\delta} \quad (38)$$

In a similar manner, we also have

$$H_{max}(\hat{P}) \leq H(P) + \hat{\delta} \quad (39)$$

According to (20), $\hat{p}_{max} \leq p_{max} + 1/2^{w-2}$. Combining (35), (38), (39) and this, we obtain (31). Q.E.D.

For the algorithm I, according to Theorem 2, the bounds on the expected length of input sequence $E(L)$ essentially depends on the approximation error θ. Hence, the smaller approximation error results in the tighter bounds on $E(L)$. In contrast, for the algorithm II, $E(L)$ depends only on the word length w. Further, for fixed n, by increasing w infinitely, (31) tends to

$$\frac{n}{H(P)} \leq E(L) \leq \infty,$$

and we cannot bound $E(L)$ from above. There also exists an optimum value of w which minimizes the upper bound of (31). On the other hand, according to Lemma 7 (with $|\mathcal{S}| = 1$, $H_{min}(P) = H_{max}(P) = H(P)$), for the original interval algorithm, the expected length of input sequence $E(\hat{L})$ can be bounded by

$$\frac{n}{H(P)} \leq E(\hat{L}) \leq \frac{n}{H(P)} + \frac{\log(2M)}{H(P)}$$

$$+ \frac{1}{(1-p_{max})H(P)}. \quad (40)$$

Hence, it is still an open problem to obtain tighter bounds or efficient algorithms such that the bounds on $E(L)$ approaches (40), as $w \to \infty$.

IV. Conclusion

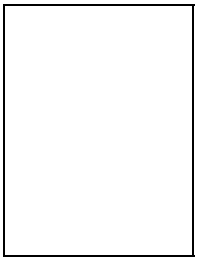
In this paper, we have proposed two algorithms for the random number generation using the integer arithmetic of limited precision. We have analyzed the performance of the algorithms for finite output length, and clarified the bounds on the approximation error and the bounds on the expected length of input sequence per one output symbol. We are now exploring the practical performance of the algorithm II, and some results were reported in [7].

Acknowledgment

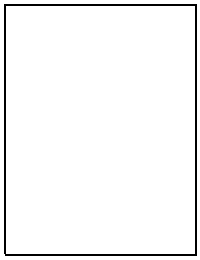
The authors are grateful to anonymous referees for their helpful comments.

References

[1] T. S. Han, *Information-Spectrum Methods in Information Theory*, Springer, 2002.
[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: John Wiley & Sons, 1991.
[3] T. S. Han and M. Hoshi, "Interval algorithm for random number generation," *IEEE Trans. Inform. Theory*, vol.43, no.2, pp.599-611, March 1997.
[4] C. B. Jones, "An efficient coding system for long source sequences," *IEEE Trans. Inform. Theory*, vol.IT-27, no.3, pp.280-291, May 1981.
[5] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless System*, Academic Press, 1981.
[6] T. S. Han, *Information-Spectrum Methods in Information Theory*, Springer 2002.
[7] Y. Kanasugi, R. Matsumoto, and T. Uyematsu, "Verification of approximate error probability of random number generation algorithm implemented by using arithmetic of limited precision," Proc. of SITA '02, vol.1, pp.51-54, Dec. 2002.



Tomohiko Uyematsu received the B. E., M. E. and D. E. degrees from Tokyo Institute of Technology in 1982, 1984 and 1988, respectively. From 1984 to 1992, He was with the Department of Electrical and Electronic Engineering of Tokyo Institute of Technology, first as research associate, next as lecturer, and lastly as associate professor. From 1992 to 1997, he was with School of Information Science of Japan Advanced Institute of Science and Technology as associate professor. Since 1997, he returned to Tokyo Institute of Technology as associate professor, and currently he is with the Department of Communications and Integrated Systems as professor. In 1992 and 1996, he was a visiting researcher at the Centre National de la Recherche Scientifique, France and Delft University of Technology, Netherlands, respectively. He received Shinohara Memorial Young Engineer Award in 1989, and the Best Paper Award in 1993, 1996, and 2002 both from IEICE. His current research interests are in the areas of information theory, especially Shannon theory and multi-terminal information theory. Dr. Uyematsu is a member of IEEE.



Yuan Li was born in Shanghai, China in 1977. She graduated from the Japanese language center affiliated to Kawai Jyuku in 1996. She received her B. E. in International Development of Eng. and M. E. in the Department of Communications and Integrated Systems from Tokyo Institute of Technology in 2000 and 2002 respectively.