HONAM MATHEMATICAL JOURNAL
Volume 5, Number 1, September, 1983

On Computer Method for Sampling from Binomial Distribution

By Byung-Ki Kim

1. Introduction

Several procedures exist for sampling from the binomial distribution $B(n, p)$

$$Pr(X=x)=\binom{n}{x}p^x(1-p)^{n-x} \qquad x=0, 1, \ldots\ldots\ldots, n$$
$$=0 \qquad \text{otherwise} \qquad (1.1)$$

on a computer. The most common procedure uses Bernoulli trials with a cost proportional to n and inverse transform approach with a cost proportional to np. Ahrens and Dieter (1974) and Relles (1972) suggested sampling method using Beta distribution which induce a cost proportional to $\ln n$. Fishman (1979) presents an alternative binomial sampling method that combines the Poisson sampling procedure in Fishman (1976) with acceptance-rejection method to induce a cost proportional to $\{n\cdot\min[p/(1-p), (1-p)/p]\}^{\frac{1}{2}}$

In this paper we introduce an alternative binomial sampling procedure using modification of the inverse transform approach that begins the search at the mode. For large n the sampling cost is proportional to $(np)^{\frac{1}{2}}$. In section 2, we described the four different approaches to binomial sampling found in the literature and then presents the new proposal. In section 3, we compared computer CPU times of the new proposal with the inverse transform procedure and the Ahrens and Dieter (1974) and Fishman (1979) procedure for selected n and p. The results favor the Poisson approach and the newly suggested modification of the inverse transform approach.

2. Procedures

Let $q_k=\sum_{x=0}^{k} P_x=\sum_{x=0}^{k}\binom{n}{x}p^x(1-p)^{n-x}$ $k=0, 1, \ldots\ldots, n$

and $\{U_i\}$ be a sequence of independent identically distributed random variables from $U(0.1)$.

2.1 Bernoulli method.

$$X=\sum_{j=0}^{n} I_{(0,p)}(U_j) \tag{2.1}$$

defines a random variable from binomial distribution $B(n, p)$ where I is the indicator function. Mean execution CPU time is proportional to n.

Algorithm $BU(n, p)$

1. $X=0, M=0$
2. Generate U
3. If $U \leqslant p$ then $X=X+1$
4. $M=M+1$
5. If $M<n$ then go to 2
6. Return with X

2.2 Inverse transform method

Since $P_r(q_{x-1}<U \leqslant q_x)=P_r(X=x)=P_x,\ x=0, 1, \ldots\ldots, n\ ;\ q_{-1}=0$

$X=\min(x: U \leqslant q_x)$

is from $B(n, p)$. Mean CPU time is proportional to np. If X is from $B(n, p)$ then $Y=n-X$ is from $B(n, 1-p)$. Therefore we substitutes $\min(p, 1-p)$ for p and takes X as the sample if $p \leqslant 0.5$ and $n-X$ if $p>0.5$. Then mean CPU time is proportional to $n \cdot min(p, 1-p)$.

Algorithm $BI(n, p)$

1. $q=p$
2. If $p>0.5$ then $q=1-p$
3. $s=1-q,\ A=1,\ B=q/s,\ C=(n+1)B,\ D=A,\ X=0$
4. Generate U from $U(0, 1)$
5. $V=U/s$
6. If $V \leqslant A$ then go to 11
7. $X=X+1$
8. $D=D(C/X-B)$
9. $A=A+D$
10. If $X<n$ then go to 6
11. If $p>0.5$ then return with $n-X$
12. Return with x

2.3 Beta method

Relles(1972) suggests this method and Ahrens and Dieter(1974) offer several useful refinements. The method avoids the actual generation $U_1, U_2, \ldots U_n$ in (2.1) and counts efficiently the number of uniform(0, 1) random variables $\leqslant p$ in a sample of size n. Beta method base on the facts that the jth order statistic of n independent uniform deviates has the Beta distribution $\mathrm{Be}(j, n-j+1)$. The method proceed as follows:

step 1 : If n is smaller than 15, use the Bernoulli method and quit.

step 2 : If n is odd, generate sample Y from $Be[(n+1)/2, (n+1)/2]$. If $Y \leqslant p$ then $X \leqslant (n+1)/2$, otherwise $X > (n+1)/2$: but whatever the value of Y, it remains to count the number between Y and p of the remaining $(n+1)/2$. That number is easily shown to have a binomial distribution $B[(n+1)/2, p^*]$,

where $p^* = \begin{cases} (p-Y)/(1-Y) & \text{if } Y \leqslant p \\ p/Y & \text{if } Y > p \end{cases}$

Return step 1.

step 3 : If n is even, reduce it by 1 in observing whether a single uniform(0, 1) random variable is $\leqslant p$. Now apply step 2 with $n-1$.

Mean CPU time is proportional to In n for large n, apart from the cost of Beta sampling.

Algorithm $BB(n, p)$

1. $X=0$, $m=n$, $q=p$, $y=0$, $h=1$
2. If $m>15$ then go to 6
3. Sample j from $B(m, q)$ using BU
4. $X=X+j$
5. Return with X
6. If m is odd then go to 10
7. $m=m-1$
8. Generate U
9. If $U \leqslant q$ then $X=X+1$
10. $a=(m+1)/2$
11. Sample s from $Be(a, a)$

12. $g=hs$, $z=y+g$

13. If $z\leqslant p$ then go to 16

14. $h=g$, $q=(p-y)/h$

15. Go to 17

16. $y=z$, $h=h-q$, $q=(p-z)/h$, $X=X+a$

17. $m=a-1$

18. Go to 2

Relles (1972, p. 162) used an approximation to generate s from $Be(j, j)$. Ahrens and Dieter (1974, p. 242) used algorithm $BS^{(1)}$ for sampling from $Be(j, j)$ whose CPU time becomes constants as j increases.

2.4 Poisson method

Fishman (1979) suggests this method which relies on the acceptance-rejection technique for sampling from a distribution. Suppose one write equivalently as

$$P_x=C_{n,x}(\mu, p)\,h_x(\mu)$$

where

$$h_x(\mu)=e^{-\mu}\mu^x/x! \quad x=0, 1, 2, \ldots.$$

and

$$C_{n,x}(\mu, p)=\begin{cases} n!\,e^{\mu}(1-p)^n\,[\mu(1-p)/p]^{-x}/(n-x)! & 0\leqslant x\leqslant n \\ 0 & \text{otherwise} \end{cases}$$

Here $h_x(\mu)$ is a probability Poisson distribution $p(\mu)$. If $C_n(\mu, p)$ is independent of X and is an upperbound on $C_{n,x}(\mu, p)$. One can sample from $B(n, p)$ as follows:

1. Sample X from Poisson distribution $p(\mu)$

2. If $X>n$ then try again

3. Sample U from $U(0, 1)$

4. If $U\leqslant C_{n,x}(\mu, p)/C_n(\mu, p)$ then X is from $B(n, p)$

5. Otherwise go to 1

Fishman (1979, p. 420) describe algorithm BP based on the Poisson method that uses algorithm PIF (Fishman 1976, pp. 147-146) for sampling from Poisson distribution $p(\mu)$ whose mean CPU time is proportional to $\mu^{\frac{1}{2}}$. Algorithm BP describes a procedure for implementing the binomial sampling plan. A user provided table of f_i facilitates the comparison needed for acceptance by replacing the ratio $C_{n,x}(\mu, p)/C_n(\mu, p)$ and the uniform test deviate U by a difference of the sums of logalithms and an exponential deviate, respectively.

Algorithm $BP(n, p)$ $n\leqslant r$

table: $f_1=0$, $f_{i+1}=\sum_{j=1}^{i} \ln j$ *for* $i=1, 2, \ldots\ldots, r$

1. $q=p$

2. *If* $p>0.5$ then $q=1-p$

3. $\mu=n-\langle n(1-q)\rangle$*

4. If $n(1-q)-\langle n(1-q)\rangle > q$ then $\mu=q(\langle n(1-q)\rangle+1)/(1-q)$

5. $\theta=(1/q-1)\mu$, $\phi=\ln\ \theta$, $m=\langle\theta\rangle$

6. Sample X from $p(\mu)$

7. If $X>n$ then go to 6

8. Sample V from exponential distribution on (0, 1)

9. $Y=n-X$

10. If $V<(m-Y)\phi-f_{m+1}+f_{y+1}$ then go to 6

11. If $p>0.5$ then $X=n-X$

12. Return with X

*$\langle\theta\rangle$ denotes the largest integer in θ.

2.5 Modification of the inverse transform method

This method uses the inverse transform approach. The representation

$$X=\begin{cases}\min(k+j\ :\ U\leqslant q_{k+j}) & \text{if } U>q_k,\ j=1, \ldots\ldots, n-k\\ \max(k-i+1\ :\ U>q_{k-i}) & \text{if } U\leqslant q_k,\ i=1, \ldots\ldots, k+1;\ q_{-1}=0\end{cases} \tag{2.2}$$

defines an random variable from $B(n, p)$ where $k=[np+0.5]$. (2.2) suggests that the search begin at the modal value $x=[np+0.5]$ instead of at the origin $X=0$ such as the inverse transform method. Mean execution CPU time is proportional to $\{n\cdot\min(p, 1-p)\}^{\frac{1}{2}}$, apart from the cost of searching q_k.

Algorithm $BK(n, p)$

$k=[np+0.5]$, q_k, p_k

1. $q=p$

2. If $p>0.5$ then $q=1-p$

3. Generate U

4. If $U\leqslant p_k$ then go to 10

5. $p_{k+1}=[(n-k)q/(k+1)(1-p)]p_k$, $q_{k+1}=p_{k+1}+q_k$

6. If $U\leqslant q_{k+1}$ then $X=k+1$, go to 19

7. $k=k+1$

8. If $k+1<n$ then go to 5

9. $X=k+1$, go to 19
10. If $k=0$ then $X=0$, go to 19
11. $q_{k-1}=q_k-p_k$
12. If $U>q_{k-1}$ then $X=k$, go to 19
13. $p_{k-1}=k(1-q)p_k/(n-k+1)q$
14. $q_{k-2}=q_{k-1}-p_{k-1}$
15. If $U>q_{k-2}$ then $X=k-1$, go to 19
16. $k=k-1$
17. If $k>0$ then go to 13
18. $X=0$
19. If $p>0.5$ then $X=n-X$
20. Return with X

We can obtain the value $q_k=\sum_{x=0}^{k}\binom{n}{x}p^x(1-p)^{n-x}$ as follows:

Algorithm $CU(n, p)$

1. $k=[np+0.5]$
2. $q_k=0$, $E=0$
3. $D=(1-p)^n$
4. $i=0$
5. $i=i+1$
6. $D=[Dp/(1-p)](n-i+1)/i$
7. $q_k=q_k+D$
8. If $i<k$ then go to 5
9. Return with q_k

3. Comparison of Algorithms

Table 1 presents sample mean execution times for four alternative FORTRAN program for sampling from the binomial distribution using HP3000-II.

The uniformly distributed variables $U_i\in(0, 1)$ were generated by means of the multiplicative congruential method as follows:

1. $y_o=32767$
2. $y_{i+1}=46341*y_i \pmod{2^{15}}$ $\quad i=1, 2, \ldots\ldots$
3. $U_{i+1}=y_{i+1}/2^{15}$

Each time is the average over 1000 calls.

To obtain the value of q_k, algorithm *BK* introduced in this paper calls the subprogram *CU* once per each 20 times.

Table 1. Sample mean CPU times for *BI*, *BP*, *BB* and *BK* On a HP3000-II

10^{-5}sec.

n	10				20				50				100				150				200			
p	*BI*	*BP*	*BB*	*BK*	*BI*	*BP*	*BB*	*BK*	*BI*	*BP*	*BB*	*BK*	*BI*	*BP*	*BB*	*BK*	*BI*	*BP*	*BB*	*BK*	*BI*	*BP*	*BB*	*BK*
0.01	54	141	123	38	58	151	238	41	68	153	361	55	76	151	385	67	88	162	417	76	91	161	415	83
0.06	61	156	132	63	71	129	241	70	99	196	390	91	141	243	493	116	185	222	492	133	222	235	564	150
0.11	66	127	128	67	83	138	243	83	132	149	385	111	207	227	450	141	281	232	490	168	352	237	527	187
0.21	79	131	123	78	109	142	229	96	197	149	351	133	335	165	429	175	478	211	465	206	609	256	495	235
0.31	94	128	112	83	136	134	200	101	260	156	331	147	466	170	390	193	668	177	412	229	867	194	456	262
0.41	106	151	123	87	162	145	201	109	327	170	324	155	595	185	378	204	864	192	399	244	1125	210	442	277
0.51	121	158	119	95	187	170	217	117	382	174	307	163	704	201	365	214	1025	222	403	257	1336	237	436	291
0.61	107	229	120	94	161	216	211	116	315	220	304	162	569	254	386	208	831	278	431	249	1078	270	431	283
0.71	94	209	127	91	135	249	225	110	254	226	344	151	446	233	398	196	636	233	411	236	819	242	438	263
0.81	81	204	140	81	109	225	244	99	189	217	361	136	313	221	414	173	442	238	466	207	563	229	461	234
0.91	68	177	138	71	83	191	253	84	123	223	372	110	184	200	419	138	247	203	448	159	304	204	467	181
0.96	62	165	144	56	69	174	254	55	91	187	376	87	119	209	423	107	151	224	506	121	174	183	454	135

4. Conclusion

Table 1 indicates which algorithm yields the smallest sample mean execution time for each (n, p) combination. The benefit of the modification of the inverse transform method in *BK* is efficient for very small and very large p, and the Poisson method in *BP* for $n \geqslant 100$ and $0.3 \leqslant p \leqslant 0.7$. Even though the execution time for *BP* is small for given $n \geqslant 100$ and $0.3 \leqslant p \leqslant 0.7$, we know that *BP* technique is including Poisson and exponential deviates. But if we want to choose more than 20 binomial deviates, for given n and p, then the *BK* procedure just introduced will be more elegant, more accurate and easier to write program comparing to *BP* procedure.

5. APPENDIX

FORTRAN SUBPROGRAM FOR BINOMIAL SAMPLING.

```
      SUBROUTINE BIK(N,P,KA,K,QK,PK,X)
      DOUBLE PRECISION P,Q,QK,QKP,PKP
      INTEGER*4 KA,X
      Q=P
      KM=K
      PKP=PK
      QKP=QK
      IF(P.GT.0.5) Q=1-P
      CALL RND(KA,U)
      IF(U.LE.QK) GO TO 6
    3 PKP=(((N-KM)*Q)/((KM+1)*(1-Q)))*PKP
      QKP=QKP+PKP
      IF(U.LE.QKP) GO TO 5
      KM=KM+1
      IF(KM+1.LT.N) GO TO 3
    5 X=KM+1
      GO TO 15
    6 IF(KM.NE.0) GO TO 9
      X=0
      GO TO 15
    9 QKP=QK-PKP
      IF(U.LE.QKP) GO TO 10
      X=KM
      GO TO 15
   10 PKP=((KM*(1-Q))/((N-KM+1)*Q))*PKP
      QKP=QKP-PKP
```

```
      IF(U.LE.QKP) GO TO 12
      X=KM-1
      GO TO 15
   12 KM=KM-1
      IF(KM.GT.0) GO TO 10
      X=0
   15 IF(P.GT.0.5) X=N-X
      RETURN
      END

      SUBROUTINE CU(N,P,K,QK,QKM)
      DOUBLE PRECISION D,P,QK,R,E
      E=N
      D=(1-P)**E
      QK=D
      IF(K.LT.1) GO TO 20
      DO 10 I=1,K
      R=I
      D=((D*P)/(1-P))*(N-(R-1))/R
   10 QK=QK+D
      QKM=QK-D
      RETURN
   20 QKM=0.
      RETURN
      END

      SUBROUTINE RND(KA,U)
      INTEGER*4 KA,KB
      KA=KA*46341
      KB=32760
      KA=JMOD(KA,KB)
      U=FLOATJ(KA)/FLOATJ(KB)
      RETURN
      END
```

References

1. Ahrens J. H., Dieter U. : *Computer methods for Sampling Gamma, Beta, Poisson and Binomial Distribution.* Computing 12 (1974), 223-246.
2. Ahrens J. H., Dieter U. : *Computer methods for Sampling from the Exponential and Normal Distribution.* Comm. ACM, Vol. 15, No. 10 (1972), 873-882.
3. Ahrens J. H., Dieter U., Grube A. : *Pseudo-random numbers, A New Proposal for the Choice of Multiplicators.* Computing 6 (1970), 121-138.
4. Dieter U., Ahrens J. H. : *A Combinatorial Method for the Generation of Normally Distributed Random Numbers.* Computing 11 (1973), 137-146.
5. Fishman G. S. : *Sampling from the Gamma Distribution on a computer. Comm.* ACM, Vol. 19, No. 7 (1976), 407-409.
6. Fishman G. S. : *Sampling from the Binomial Distribution on a Computer.* JASA, Vol. 174, No. 336 (1979), 418-423.
7. Fishman G. S. : *Sampling from the Poisson Distribution on a Computer.* Computing 17 (1976), 147-156.
8. Forsythe G. E. : *Von Neumann's Comparison method for Random Sampling from the normal and Other Distributions.* Math. Comput. Vol. 126, No. 120 (1972), 817-826.
9. Hull T. E., Dobel A. R. : *Random Number Generators.* Siam Review, Vol. 4, No. 3 (1962), 230-251.
10. Kendall M. G., Stuart A. : *The Advanced Theory of Statistics.* Vol. 1. (1969), London, Charles Griffin & Co., Ltd.
11. Relles D. A. : *A Simple Algorithm for Generating Binomial Random Variables When N is Large.* JASA, Vol. 67, No. 339 (1972), 612-613.

Department of Computer Science & Statistics,
College of Natural Science, Chonnam National University.