SQUEEZE METHODS FOR
GENERATING GAMMA VARIATES.

Bruce W. Schmeiser

Department of Operations Research
and Engineering Management
Southern Methodist University
Dallas, Texas 75275

July 1978
Revised August 1978

This work supported by the Office of Naval
Research under Contract N00014-77-C-0425

78 10 16 001
410 597
ABSTRACT

Two algorithms are given for generating gamma distributed random variables. The algorithms, which are valid when the shape parameter is greater than one, use a uniform majorizing function for the body of the distribution and exponential majorizing functions for the tails. The algorithms are self-contained, requiring only U(0,1) variates. Comparisons are made to three competitive algorithms in terms of marginal generation times, initialization time, and memory requirements. Both algorithms are faster than existing methods, for all values of the shape parameter.

KEY WORDS

Gamma Distribution
Simulation
Random Number Generation
Rejection Methods
Monte Carlo
Distribution Sampling
1. INTRODUCTION

During the last few years many algorithms have been developed for generation of gamma random variables having density function

\[ f_\gamma(x) = x^{\alpha-1} \exp(-x)/\Gamma(\alpha) \quad 0 \leq x < \infty, \; 1 < \alpha < \infty. \]

To the author's knowledge, these include Ahrens and Dieter (1974), Atkinson and Pearce (1976), Fishman (1973), Fishman (1976), Jühnk (1974), Greenwood (1974), Marsaglia (1977), McGrath and Irving (1973), Tadikamalla (1978a, 1978b), C.S. Wallace (1976), N.D. Wallace (1974), and Whittaker (1974). Most have been implementations of the general acceptance/rejection algorithm, with many using the modification referred to as the "squeeze" technique by Marsaglia (1977). The algorithms developed in this paper use the squeeze technique, which in the general case proceeds as follows:

Let \( f(x) \) be the density function from which random variates are desired and let \( t(x) \) and \( b(x) \) be majorizing and minorizing functions of \( f(x) \), respectively (\( t(x) \geq f(x) \) for all \( x \) and \( b(x) \leq f(x) \) for all \( x \)). Then

1. Generate \( x \) having density \( r(x) = t(x)/\int_0^\infty t(y)dy \).
2. Generate \( v \sim U(0,1) \).
3. If \( v \leq b(x)/t(x) \), deliver \( x \).
4. If \( v \leq f(x)/t(x) \), deliver \( x \). Otherwise go to step 1.

If \( t(x) \) fits \( f(x) \) well, if \( r(x) \) yields variates quickly, and if \( b(x) \) both fits \( f(x) \) well and is quick to evaluate, the squeeze technique yields variates quickly even when \( f(x) \) is time consuming to evaluate.
2. The Algorithms

Similar to the beta algorithms of Schmeiser and Shalaby (1977), the points of inflection and the mode are central to this algorithm.

Define

\[ x_1 = x_2(1 - 1/(x_3 - x_2)) \]

\[ x_2 = \text{Max}(0, x_3 - x_3^{1/2}) \]

\[ x_3 = \alpha - 1 \]

\[ x_4 = x_3 + x_3^{1/2} \]

\[ x_5 = x_4(1 + 1/(x_4 - x_3)) \]

Here \( x_3 \) is the mode, \( x_2 \) and \( x_4 \) are the points of inflection of \( f(x) \) and \( x_1 \) and \( x_5 \) are the points at which the tangent of \( f(x) \) at \( x_2 \) and \( x_4 \) cross the X axis. If \( \alpha < 2 \), there is no left point of inflection and \( x_1 = x_2 = 0 \). These points are illustrated in Figure A.

The simpler, and slower, of the two algorithms is described first. The algorithm uses a uniform majorizing function for the body of the distribution and an exponential majorizing function for the tails. It is denoted G2PE since it is a gamma (G) generator which requires evaluating the density function at two points (2P) and uses an exponential (E) majorizing function for the tails.

For simplicity \( f_\gamma(x) \) is rescaled to

\[ f(x) = \exp[x_3 \ln(x/x_3) + x_3 - x] \]

to avoid evaluating the gamma function and to yield \( f(x_3) = 1 \).
Figure A. Algorithm G2PE for $\alpha = 5$. 
The majorizing function is

\[ t(x) = f(x_2) \exp(-\lambda_L(x - x_2)) \quad 0 < x \leq x_2 \]
\[ = 1 \quad x_2 < x \leq x_4 \]
\[ = f(x_4) \exp(-\lambda_R(x - x_4)) \quad x_4 < x < \infty \]

where \( \lambda_L = 1 - (x_3/x_2) \) and \( \lambda_R = 1 - (x_3/x_4) \) to make \( t(x) \) tangent to \( f(x) \) at \( x_2 \) and \( x_4 \). Several previous algorithms have used exponential tails, although not as implemented here. Schmeiser (1978) discusses the use of exponential majorizing functions for distribution tails in detail.

The minorizing function is

\[ b(x) = f(x_2)(x - x_1)/(x_2 - x_1) \quad 0 < x \leq x_2 \]
\[ = f(x_2) + (1 - f(x_2))(x - x_2)/(x_3 - x_2) \quad x_2 < x \leq x_3 \]
\[ = f(x_4) + (1 - f(x_4))(x_4 - x)/(x_4 - x_3) \quad x_3 < x \leq x_4 \]
\[ = f(x_4)(x_5 - x)/(x_5 - x_4) \quad x_4 < x \leq 1 \]

Both functions are shown in Figure A.

Based on these functions and the squeeze technique discussed in Section 1, algorithm G2PE can be implemented as follows.

**Algorithm G2PE**

**Initialization**

1. Set \( x_3 = a - 1, D = x_3^{1/2}, \lambda_L = 1, x_1 = x_2 = f_2 = 0 \).

   If \( D \geq x_3 \) go to step 2. Otherwise set

   \( x_2 = x_3 - D, \lambda_L = 1 - x_3/x_2, x_1 = x_2 + 1/\lambda_L \),

   and \( f_2 = f(x_2) \).
2. Set \( x_4 = x_3 + D, \lambda_R = 1 - x_3/x_4, x_5 = x_4 + 1/\lambda_R. \)
\[ f_4 = f(x_4), p_1 = x_4 - x_2, p_2 = p_1 - f_2/\lambda_L, \]
\[ p_3 = p_2 + f_4/\lambda_R. \]

---

**Generation**

3. Sample \( u, v \sim U(0,1) \) and set \( u = u p_3. \)
   
   If \( u > p_1 \), go to step 4. Otherwise set \( x = x_2 + u. \)
   
   If \( x > x_3 \) and \( v \leq f_4 + (x_4 - x)(1 - f_4)/(x_4 - x_3) \),
   deliver \( x. \) If \( x < x_3 \) and \( v \leq f_2 + (x - x_2)(1 - f_2)/\)
   
   \((x_3 - x_2), \) deliver \( x. \) Otherwise go to step 6.

4. If \( u > p_2 \), go to step 5. Otherwise set \( u = (u - p_1)/(p_2 - p_1), \)
   \( x = x_2 - \ln(u)/\lambda_L. \) If \( x < 0, \) go to step 3. Otherwise set
   \( v = v f_2 u. \) If \( v \leq f_2(x - x_1)/(x_2 - x_1), \) deliver \( x. \) Otherwise
   go to step 6.

5. Set \( u = (u - p_2)/(p_3 - p_2), x = x_4 - \ln(u)/\lambda_R, v = v f_4 u. \) If
   \( v \leq f_4(x_5 - x)/(x_5 - x_4), \) deliver \( x. \)

6. If \( \ln v \leq x_3 \ln(x/x_3) + x_3 - x, \) deliver \( x. \) Otherwise go to step 3.

---

The second algorithm, denoted G4PE for reasons analogous to G2PE, is
illustrated in Figure B. The majorizing function is uniform over the body
of the distribution, triangular over the shoulders, and exponential in the
tails. The resulting area under the majorizing function is partitioned into
the ten regions shown. Four regions have zero probability of rejection. Of
the remaining six regions, two require uniform variates, two require tri-
angular variates and two require exponential variates.

The algorithm may be implemented as follows:
Figure B. Algorithm G4PE for $\alpha = 5$. 
Algorithm G4PE

Initialization
1. Set $x_3 = \alpha - 1$, $D = x_3^{1/2}$, $x_1 = x_2 = f_1 = f_2 = 0$.
   If $D \geq x_3$, go to step 2. Otherwise set $x_2 = x_3 - D$, $x_1 = x_2(1 - 1/D)$, $\lambda_L = 1 - x_3/x_1$, $f_1 = f(x_1)$, and $f_2 = f(x_2)$.

2. Set $x_4 = x_3 + D$, $x_5 = x_4(1 + 1/D)$, $\lambda_R = 1 - x_3/x_5$, $f_4 = f(x_4)$, and $f_5 = f(x_5)$. Set $p_1 = f_2(x_3 - x_2)$, $p_2 = f_4(x_4 - x_3) + p_1$, $p_3 = f_1(x_2 - x_1) + p_2$, $p_4 = f_5(x_5 - x_4) + p_3$, $p_5 = (1 - f_2)(x_3 - x_2) + p_4$, $p_6 = (1 - f_4)(x_4 - x_3) + p_5$, $p_7 = (f_2 - f_1)(x_2 - x_1)/2 + p_6$, $p_8 = (f_4 - f_5)(x_5 - x_4)/2 + p_7$, $p_9 = -f_1/\lambda_L + p_8$, $p_{10} = f_5/\lambda_R + p_9$.

Generation
3. Sample $u \sim U(0,1)$ and set $u = u_{10}$. If $u > p_4$, go to step 7.
   If $u > p_1$, go to step 4. Otherwise deliver $x = x_2 + u/f_2$.

4. If $u > p_2$, go to step 5. Otherwise deliver $x = x_3 + (u - p_1)/f_4$.

5. If $u > p_3$, go to step 6. Otherwise deliver $x = x_1 + (u - p_2)/f_1$.

6. Deliver $x = x_4 + (u - p_3)/f_5$.

7. Sample $w \sim U(0,1)$. If $u > p_5$, go to step 8. Otherwise set $x = x_2 + (x_3 - x_2)w$. If $(u - p_4)/(p_5 - p_4) \leq w$, deliver $x$.
   Otherwise set $v = f_2 + (u - p_4)/(x_3 - x_2)$ and go to step 13.

8. If $u > p_6$, go to step 9. Otherwise set $x = x_3 + (x_4 - x_3)w$.
   If $(p_6 - u)/(p_6 - p_5) \geq w$, deliver $x$. Otherwise set $v = f_4 + (u - p_5)/(x_4 - x_3)$ and go to step 13.
9. If $u > p_8$, go to step 11. Otherwise sample $w_2 \sim U(0,1)$.
   If $w_2 > w$, set $w = w_2$. If $u > p_7$, go to step 10. Otherwise
   set $x = x_1 + (x_2 - x_1)w$, $v = f_1 + 2w(u - p_6)/(x_2 - x_1)$.
   If $v \leq f_2 w$, deliver $x$. Otherwise go to step 13.

10. Set $x = x_5 - w(x_5 - x_4)$, $v = f_5 + 2w(u - p_7)/(x_5 - x_4)$ and go to step 13.

11. If $u > p_9$, go to step 12. Otherwise set $u = (p_9 - u)/(p_9 - p_8)$,
    $x = x_1 - (\ln u)/\lambda_L$. If $x \leq 0$, go to step 3. If $w < (\lambda_L(x_1 - x) + 1)/u$,
    deliver $x$. Otherwise set $v = w f_1 u$ and go to step 13.

12. Set $u = (p_{10} - u)/(p_{10} - p_9)$, $x = x_5 - (\ln u)/\lambda_R$. If $w < (\lambda_R(x_5 - x) + 1)/u$, deliver $x$. Otherwise set $v = w f_5 u$.

13. If $\ln v > f(x)$, go to step 3. Otherwise deliver $x$. 
3. COMPUTATIONAL RESULTS

Based on the findings of Cheng (1976), Marsaglia (1977) and Tadikamalla (1978b), it appears that the three fastest existing algorithms are to be found in these three papers. Using the names used in the above papers, Cheng's algorithm is denoted by GB, Marsaglia's algorithm is denoted RGAMA, and Tadikamalla's algorithm is denoted GAMMA.

The table compares G2PE, G4PE, GB, RGAMA and GAMMA in terms of generation time per variate, initialization time, and memory requirements. The times are based on the generation of 10,000 variates and are accurate to within .02 milliseconds. The algorithms were coded in FORTRAN on the CDC Cyber 72 at Southern Methodist University. The uniform variates were generated by the relatively fast RANF internal to the FTN compiler.

Insert Table About Here

It is clear that all five algorithms are robust to changes in α, with all algorithms but GAMMA being slightly faster for larger α. In this implementation, the algorithms can be ranked in order of increasing marginal times as G4PE, G2PE, RGAMA, GB and GAMMA, except that GAMMA is faster than GB for α < 1.5. G2PE is about 15% faster, and G4PE is 30–40% faster, than the previous fastest algorithm RGAMA.
Marginal Generation Times (in Milliseconds) and Memory Requirements

<table>
<thead>
<tr>
<th>Method</th>
<th>α</th>
<th>RGAMA</th>
<th>GAMMA</th>
<th>GB</th>
<th>G2PE</th>
<th>G4PE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0001</td>
<td>.53</td>
<td>.56</td>
<td>.70</td>
<td>.46</td>
<td>.39</td>
</tr>
<tr>
<td></td>
<td>1.2</td>
<td>.53</td>
<td>.61</td>
<td>.67</td>
<td>.45</td>
<td>.33</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>.52</td>
<td>.64</td>
<td>.63</td>
<td>.41</td>
<td>.33</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>.52</td>
<td>.70</td>
<td>.62</td>
<td>.42</td>
<td>.33</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>.49</td>
<td>.71</td>
<td>.60</td>
<td>.40</td>
<td>.29</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>.49</td>
<td>.75</td>
<td>.56</td>
<td>.37</td>
<td>.28</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>.47</td>
<td>.76</td>
<td>.57</td>
<td>.39</td>
<td>.28</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>.46</td>
<td>.78</td>
<td>.54</td>
<td>.40</td>
<td>.28</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>.47</td>
<td>.76</td>
<td>.55</td>
<td>.38</td>
<td>.26</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>.45</td>
<td>.72</td>
<td>.53</td>
<td>.38</td>
<td>.26</td>
</tr>
<tr>
<td>Set-up Time</td>
<td>.24</td>
<td>.34</td>
<td>.18</td>
<td>.53-.83</td>
<td>1.01-1.57</td>
<td></td>
</tr>
<tr>
<td>Memory Requirements</td>
<td>538</td>
<td>316</td>
<td>290</td>
<td>405</td>
<td>566</td>
<td></td>
</tr>
</tbody>
</table>

a As implemented using the K normal generator. For the implementation using the polar method, add approximately .09 milliseconds for all α and decrease the memory requirements by 224.

b Depends upon the value of α. The lower set-up time applies when α < 2 and the higher value corresponds to α > 2.

c Memory requirements include necessary routines such as ALOG, EXP and SQRT.
Each of the algorithms requires a one time initialization. In order of increasing set-up time the algorithms are GB, RGAMA, GAMMA, G2PE and G4PE. Since the algorithms with lower marginal times tend to have higher set-up times, a tradeoff is made which depends upon $M$, the required number of variates for a fixed $\alpha$. For $\alpha \leq 2$, RGAMA is fastest for $M \leq 3$, G2PE is fastest for $M = 4$ or 5, and G4PE is fastest for $M \geq 6$. For $\alpha > 2$, RGAMA is fastest for $M < 7$ and G4PE is fastest for $M > 7$. For no combination of $\alpha$ and $M$ is either GAMMA or GB the fastest algorithm.

Memory requirements are also shown in the table. In order of increasing memory requirements the algorithms are GB, GAMMA, G2PE, RGAMA, and G4PE which includes necessary routines such as ALOG, EXP and SQRT. However RGAMA requires a normal variate generator, which as implemented here is algorithm KR given by Kinderman and Ramage (1976) with memory requirements of 289. While a normal variate algorithm requiring less memory could be used, marginal generation times would increase for RGAMA. For example, using Marsaglia’s polar method, total memory requirements for RGAMA were only 314, but marginal generation times increased approximately .09 millisecond for all values of $\alpha$. Of course different normal generators will result in various tradeoffs between speed and memory.

Ease of implementation may be crudely measured in lines of code and additional algorithms needed. In ascending order of lines of code the algorithms are GB, RGAMA, GAMMA, G2PE and G4PE. The only additional algorithm needed is the normal generator used by RGAMA.
REFERENCES


APPENDIX

PROGRAM MAIN (INPUT, OUTPUT, TAPE5=INPUT, TAPE6=OUTPUT)
C
BRUCE SCHARF JUNE 12, 1978 SCOTSFORD METHODIST UNIVERSITY
C
TO COMPARE GAMMA VARIATE GENERATION ROUTINES
C
A = SHAPE PARAMETER (OF THE GAMMA DISTRIBUTION)
C
DIMENSION NAME(5), ES(10)
C
DATA 45/1.0001, 1.0002, 1.0003, 1.0004, 1.0005, 1.0006, 1.0007, 1.0008, 1.0009, 1.0010
C
BETA = 1.
C
N = 10000
C
DO 10 I = 1, 10
   ALPHA = AS(1)
C
DO 20 J = 1, 5
   SUM = 0.
   SUM2 = 0.
   TIME = SECOND(X)
C
DO 30 K = 1, N
   GO TO (1, 2, 3, 4, 5, J)
   1 CALL GAMMA(ALPHA, BETA, ISFED, X)
      GO TO 15
   2 CALL GAMMA(ALPHA, BETA, ISFED, X)
      GO TO 15
   3 CALL GAMMA(ALPHA, BETA, ISFED, X)
      GO TO 15
   4 CALL GAMMA(ALPHA, BETA, ISFED, X)
      GO TO 15
   5 CALL GAMMA(ALPHA, BETA, ISFED, X)
15 SUM = SUM + X
30 SUM2 = SUM2 + X(X)
C
TIME = (SECOND(X) - TIME) * 1000 / N
SUM = SUM / N
SUM2 = SUM2 / N - SUM * SUM
C
20 WRITE (1, 10) NAME(J), ALPHA, SUM, SUM2, TIME, N
101 FORMAT (F0.0, 016.4, 016.4, 16)
C
CONTINUE
STOP
END
SUBROUTINE RGGM (A, BETA, IGAMMA, RGAMMA)

BRUCE SCHMEISER
JULY 12, 1978
SOUTHERN METHODIST UNIVERSITY

TO GENERATE STANDARD GAMMA VARIATES USING McGILWAIN'S SCF M-E METHOD

REFERENCE: His article in Comp. and Math. With Applications

VOL 3, PP. 321-325, 1977

A = SHAPE PARAMETER

X = GENERATED VARIATE

A MUST BE GREATER THAN 1/3 AND HE RECOMMENDS A .GT. 1

DATA T/L.
IF (B .EQ. A) GO TO 1
B = A
S = .3333333/SCRT(A)
Z0 = 1. -1.732051*5
CC = A * Z0**2 * S*(1.732051)**2
CS = 1. - S*5

REJECTION PROCEDURE BEGINS HERE

1 CALL NORM(IGAMMA, Y)
Z = S*X + CS
IF (Z .LE. C.) GO TO 1
RGAMMA = A**7**2
L = -ALGGRANF(IGAMMA)
CD = E + .5*Y**2 - RGAMMA + CC
T = 1. - 20/Z
IF (CD + CL*T*(1. + T* (.5 + .3333333*T)) .GT. C.) GO TO 2
IF (CD + CL*ALG(7/70) .LT. 0.) GO TO 1
2 RGAMMA = RGAMMA/BETA
RETURN
END

SUBROUTINE NORM(IGAMMA, X)

GENERATION OF ONE NORM(0,1) VARIATE USING THE ALGORITHM GIVEN BY KINDELMAN AND RAMAGE
IN THE JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION 12/76

CODED BY PETER PONNER AND MODIFIED BY BRUCE SCHMEISER
MARCH 1977 AND JUNE 1977 RESPECTIVELY

DATA TAIL/2.216035867766471/
LU=RANF(IGAMMA)
IF(UL.GT.*8407640298758) GO TO 2

RETURN TRAPEZIUM VARIATE 88 PERCENT OF THE TIME

Y=RANF(IGAMMA)
X=TAIL*(1.13113163*494180*LU+Y-1.0)
RETURN
2 IF(U.LT.0.723104873608) GO TO 4
C TAIL COMPUTATION
3 V=RANF(ISEED)
   W=RANF(ISEED)
   T=TAIL*TAIL/2.0
   IF(V.W.GT.T.DT) GO TO 3
   X=SQRT(2.0*T)
   IF(U.GE..98665477086949) X=-X
   RETURN
4 IF(U.LT.956.720824790463) GO TO 6
C FIRST NEARLY LINEAR DENSITY
5 V=RANF(ISEED)
   W=RANF(ISEED)
   Z=V-W
   LET V=MAX(V,W) AND LET W=MIN(V,W)
   IF(V.GT.W) GO TO 100
   TEMP=V
   V=W
   W=TEMP
100 T=TAIL-.63083406192196C4
   IF(V.LE..755915316646C1) GO TO 9
   DIFF=EXP(-(T*.4559/2.50662927463190-.180029106463)*
         (2.2160267164471-ABS(T))
   IF(ABS(T).LT.0.034240503750111.LE.DIFF) GO TO 6
   GO TO 5
6 IF(U.LT.9112780287903) GO TO 8
C SECOND NEARLY LINEAR DENSITY
7 V=RANF(ISEED)
   W=RANF(ISEED)
   Z=V-W
   LET V=MAX(V,W) AND LET W=MIN(V,W)
   IF(V.GT.W) GO TO 101
   TEMP=V
   V=W
   W=TEMP
101 T=4.79727404222441+1.10547386102207054W
   IF(V.LE..872954766471970) GO TO 9
   DIFF=EXP(-(T*.4559/2.50662927463190-.180029106463)*
         (2.2160267164471-ABS(T))
   IF(ABS(T).LT.0.42064063731285.LE.DIFF) GO TO 6
   GO TO 7
C THIRD NEARLY LINEAR DENSITY
8 V=RANF(ISEED)
   W=RANF(ISEED)
C
LET V = MAX(V,W) AND LET W = MIN(V,W)
IF (V .GT. W) GO TO 102
TEMP = V
W = W
102 T = 4.7972740422441 - 5.595507136015940*W
IF (V .LE. 3.05577924423817) GO TO 9
DIFF = EXP(-T**.5)/(2.5662827463100 - 1.862515168563*
* (2.21603567166471-ABS(T)))
IF (ABS(Z) .GT. 3.777777777777777777E-08) GO TO 1
GO TO 8
9 X = T
IF (Z.GE.1.0) Y = -X
RETURN
END

SUBROUTINE GB (ALPHA,BETA,ISEED,X)
HURST SCHMEISER, AUGUST 4, 1978 SOUTHERN METHODIST UNIVERSITY
C GAMMA VARIATE GENERATOR. REFERENCE CHENG, APPLIED STATISTICS
C (1977) 26-1, 71-75.
C ALPHA = SHAPE PARAMETER
C BETA = SCALE PARAMETER
C ISEED = RANDOM NUMBER SEED
C X = GENERATED GAMMA VARIATE
C
DATA ASAVE /1.0/
IF (ALPHA .EQ. ASAVE) GC TO 100
C
C********INITIALIZATION
C
ASAVE = ALPHA
A = 1.0 / SQRT(ALPHA-ALPHA + 1.0)
B = ALPHA - 1.3625
C = ALPHA + 1.25
C
C********GENERATION OF ONE GAMMA VARIATE X
C
100 U1 = RAND(ISEED)
U2 = RAND(ISEED)
V = A * ALOG(U1)/(1.0 - L1)
X = ALPHA * EXP(V)
Z = U1*U1*U2
H = B + COS (-X)
IF (R + 2.5940774 - 4.592*GE. 0.0) GO TO 200
IF (R .LT. ALOG(12)) GC TO 100
200 X = BETA*X
RETURN
END
SUBROUTINE G2PS(ALPHA, BETA, ISFED, Y)

C****PRCE SCHMEISER ULY 12,1978 SOUTHER MECHEST UNIVERSITY
C TO GENERATE A STANDARD GAMMA VARIATE USING EXPONENTIAL TAIL
C REJECTION AND RECTANGULAR REJECTION FOR THE BODY OF THE
C DISTRIBUTION.
C A = THE SHAPE PARAMETER (A > GT. 1)
C X = THE GENERATED VALUE
C
DATA ASAVE/-1./,YLL/1.1/
IF (ALPHA .GE. ASAVE) GO TO 100

C**** SET-UP RECTANG HERE
C
ASAVE = ALPHA
X1 = 0.
X2 = 0.
F2 = 0.
X3 = ALPHA - 1
C = SCRT(X3)
IF (D .GE. Y3) GO TO 10
X2 = X3/D
XLL = 1. - X3/X2
V1 = X2 + 1/XLL
F2 = EXP(-X3*ALNC(Y3/Y2) + X3 - X2)
10 X4 = X3 + D
XLR = 1. - Y3/X4
X5 = Y4 + 1/XLP
F4 = EXP(-X3*ALNC(Y4/Y3) + X3 - X4)
P1 = X4-X2
P2 = P1 - F2/XLL
F3 = P2 + F4/YLP

C****VARIATE GENERATION PROCEDURE BEGINS HERE
C
100 U = RAND(ISFEC) * P3
V = RAND(ISFED)

C C RECTANGULAR REJECTION
C
IF (U .GT. P1) GO TO 200
X = Y2 + U
IF (X .LT. X3) GO TO 110
IF (V .LT. F4 + (Y4-V)*(1-F4)/(Y4-Y3)) GO TO 500
GO TO 400
110 IF (V .LT. F2 + (X-Y2)*(1-F2)/(X3-Y2)) GO TO 500
GO TO 400

C C LEFT TAIL GENERATION
C
200 IF (L .GT. P2) GO TO 300
L = (U-F1)/(P2-P1)
X = Y2 - ALNC(L*U)/XLL
IF (X .LT. 0.) GO TO 100
V = V * F2 * U
IF (V .LT. F2 * (X-Y1)/(X2-X1)) GO TO 500
GO TO 400

18
RIGHT TAIL GENERATION

3CG L = (1-E2) / (D3-D2)
Y = X4 - ALOGC(U) / YLP
V = V * F4 * !
IF (V .LT. F4*(Y5-Y)/(X5-Y4)) GO TO 500

FINAL REJECTION TO Y

400 IF(ALOGC(V) .GT. X3*ALOGC(Y/X3) + X3 - X) GO TO 100
500 X = X*ALPHA
RETURN
END

SUBROUTINE GAMMA (ALPHA, BETA, ISEED, Y)

BRUCE SCHMIDT, SOUTHERN METHODIST UNIVERSITY, MAY 24, 1970
TO GENERATE GAMMA VARIATES WITH MEAN ALPHA*BETA
AND VARIANCE ALPHA*BETA^2 USING TADIKAMALLA'S ALGORITHM
DESCRIPTED IN GENERATION OF GAMMA VARIATES -- III.
VALID ONLY FOR ALPHAS .GT. 1

DATA ASAVE/-1.1/
IF (ALPHA .LE. ASAVE) GO TO 100

SET UP CONSTANTS

ASAVE = ALPHA
A = ALPHA - 1
B = .5 + .5*SQRT(4.*ALPHA-3)
C = A * (1. + P) / p
D = (B-A) / (A+B)
E = EXP(-A/B) / ?.

GENERATION OF ONE VARIATE

100 U = E + RANF(ISEED) * (1.-E)
IF (U .GT. .5) GO TO 200
Y = A + B*ALOG(1+U)
IF (Y .LT. 0.) GO TO 100
Y = A - X
GO TO 300
200 X = A - B*ALOG(2.*1-U)
Y = Y - A
300 L = RANF(ISEED)
IF (ALOGC(U) .GT. (A*ALOG(D*X) - X + (Y/B) + C)) GO TO 100
X = Y*ALPHA
RETURN
END
SUBROUTINE GADE (ALPHA, BETA, ISEED, X)

BRUCE SCHMEISER JULY 1979 SOUTHERN METHODIST UNIVERSITY

GENERATION OF ONE FRENET-RANDON VARIATE USING
THE FOUR POINT METHOD WITH EXPONENTIAL TAILS
FROM THE GAMMA DENSITY FUNCTION

ALPHA = SHAPE PARAMETER
BETA = SCALE PARAMETER
ISEED = RANDOM NUMBER SEED
X = GENERATED GAMMA VARIATE
REFERENCE H. SCHMEISER SCHEME FOR GAMMA VARIATE

GENERATION OF TECH REPORT 78009 JULY 1978
DATA ASAVE /-1.1/XLL/-1.1/
IF (ALPHA *EC. ASAVE) GO TO 100

C*******INITIALIZATION

ASAVE = ALPHA

1. X1 = X2 = F1 = F2 = 0.
X3 = ALPHA - 1.
0 = SQRT(X3)
IF (DGE. X3) GO TO 10
X2 = X3 - D
X1 = X2*(1./-1./D)
ALL = 1.-X3/X1
F1 = EXP (A3*ALNG(X1/X3) + X3 - X1)
F2 = EXP (A3*ALNG(X2/X3) + X3 - X2)

2. 10 X4 = X3 + U
X5 = X4*(1.1+1./D)
XLR = 1. - X3/X5
F4 = FXW (A3*ALNG(X4/X3) + X3 - X4)
F5 = EXP (A3*ALNG(X5/X3) + X3 - X5)

C CALCULATE PROBABILITY FOR EACH OF THE TEN REGIONS

P1 = F2*(X3-X2)
P2 = F4*(X4-X3) + P1
P3 = F4*(X2-X1) + P2
P4 = F5*(X5-X4) + P3
P5 = (1.-F2)*(X3-X2) + P4
P6 = (1.-F4)*(X4-X3) + P5
P7 = (F2-F1)*(X2-X1) + P6
P8 = (F4-F5)*(X4-X3) + P7
P9 = -F1/XLL + P8
P10 = F5/XLR + P5

C********GENERATE ONE GAMMA VARIATE X

3. 100 U = RANF(ISeED) * P10

C THE FIVN REGIONS WITH ZERO PROBABILITY OF REJECTION

IF (U > 0.1. P4) GO TO 50A
IF (U > 0.1. P1) GO TO 20A
X = X2 + U/F2
GO TO 1400

4. 200 IF (U > 0.1. P2) GO TO 30A
X = X3 + (U-P1)/F4
5. GO TO 1400
300 IF (U GT. P3) GO TO 400
\[ X = X1 + (U-P2) / F1 \]
GO TO 1400
6. 400 \[ X = X4 + (U-P3) / F5 \]
GO TO 1400

7. THE TWO REGIONS USING RECTANGULAR REJECTION
500 \[ W = RANF(ISEED) \]
IF (U LE. P5) GO TO 600
\[ X = X2 + (X3-X2) * W \]
IF \( \frac{(U-P7)}{(F5-P4)} \) LE. W \( \) GO TO 1400
\[ V = F2 + (U-P4) / (X3-X2) \]
GO TO 1300
8. 600 IF (U GT. P6) GO TO 700
\[ X = X3 + (X4-X3) * W \]
IF \( \frac{(F6-P)}{(F6-P2)} \) GE. W \( \) GO TO 1400
\[ V = F4 + (U-F5) / (X4-X3) \]
GO TO 1400

9. THE TWO TRIANGULAR REGIONS
700 IF (U GT. P8) GO TO 900
\[ W2 = RANF(ISEED) \]
IF (W2 GT. W) \( \) \( W = W2 \)
IF (U GT. P7) GO TO 800
\[ X = X1 + (X2-X1) * W \]
\[ V = F1 + 2*W2*(U-P6) / (X2-X1) \]
IF \( \frac{V}{LE. F2W2} \) GO TO 1400
GO TO 1300
10. 800 \[ X = X5 - W3*(X5-X4) \]
\[ V = F5 + 2*W3*(U-P7) / (X5-X4) \]
GO TO 1300

11. THE TWO EXPONENTIAL REGIONS
900 IF (U GT. P9) GO TO 1000
\[ U = (P9-U) / (F9-P8) \]
\[ X = X1 - ALOG(U) / XLL \]
IF (X LE. 0) \( \) GO TO 100
\[ IF \( \frac{W}{LE. (XLL*a(X1-X)+1) / U} \) \( \) GO TO 1400
\[ V = W*F1+U \]
GO TO 1400
12. 1000 \[ U = (P10-U) / (P10-P9) \]
\[ X = X5 - ALOG(U) / XLH \]
IF \( \frac{W}{LE. (XLH*a(X5-X)+1) / U} \) \( \) GO TO 1400
\[ V = W*F5+U \]

13. PERFORM THE ALGEBRAIC REJECTION
1300 \[ X = ALOG(V) * GT. (X3*ALG(X/X3) - X) \]
GO TO 100
RETURN
END
Two algorithms are given for generating gamma distributed random variables. The algorithms, which are valid when the shape parameter is greater than one, use a uniform majorizing function for the body of the distribution and exponential majorizing functions for the tails. The algorithms are self-contained, requiring only U(0,1) variates. Comparisons are made to three competitive algorithms in terms of marginal generation times, initialization time, and memory requirements. Both algorithms are faster than existing methods, for all values of the shape parameter.