13 June 1994

The Gamma Distribution Random Number Generator

Orin Dahl
Lawrence Berkeley Laboratory, Berkeley, CA 94720

I have checked the new gamma distribution random number generator description written by H.-J. Trost for the 1994 edition of RPP. (See the appendices for a description of the gamma distribution and of the algorithm for generating random numbers.)

For each of several values of k (with $\lambda = 1$), I generated and histogrammed 100000 random numbers and plotted the function over it. The function is absolute; it is not a fit. Since the gamma distribution is normalized so the integral from 0 to ∞ is 1, one can simply scale the function by multiplying it by the number of random numbers generated.

From the plots one can see that the agreement is excellent.

I have included the following:

1. The description of the gamma distribution and a random number generator for it from the 1994 edition of RPP.
2. Histograms of the random numbers and a plot of the function for $k = 0.1$, 0.3, 0.9, 1.0, 1.1, 3.0, 10.0, and 30.0.
3. The kumac and comis (fortran) files used in PAW to generate the plots. They are:
    1. mkplots.kumac – make the plots
    2. gamdis.kumac – make one plot
    3. mkgam.f – generate the random numbers and fill the histogram
    4. gamdis.f – calculate the function

1.3.7 The Gamma distribution (continuous)

If a process generating events as a function of x (*e.g.*, space or time) satisfies conditions (a)-(c) of the Poisson distribution, then the x distance from an arbitrary starting point (which may be some particular event) to the k^{th} event belongs to a *gamma* distribution:

$$f(x;\,\lambda,\,k) = \frac{x^{k-1}\lambda^k e^{-\lambda x}}{\Gamma(k)}\,,\quad 0 < x < \infty. \tag{1.37}$$

$\Gamma(k)$ is the gamma function, equal to $(k-1)!$ if k is an integer. The Poisson parameter μ is λ per unit x;

$$E(x) = k/\lambda;\ \ \mathrm{Var}(x) = k/\lambda^2. \tag{1.38}$$

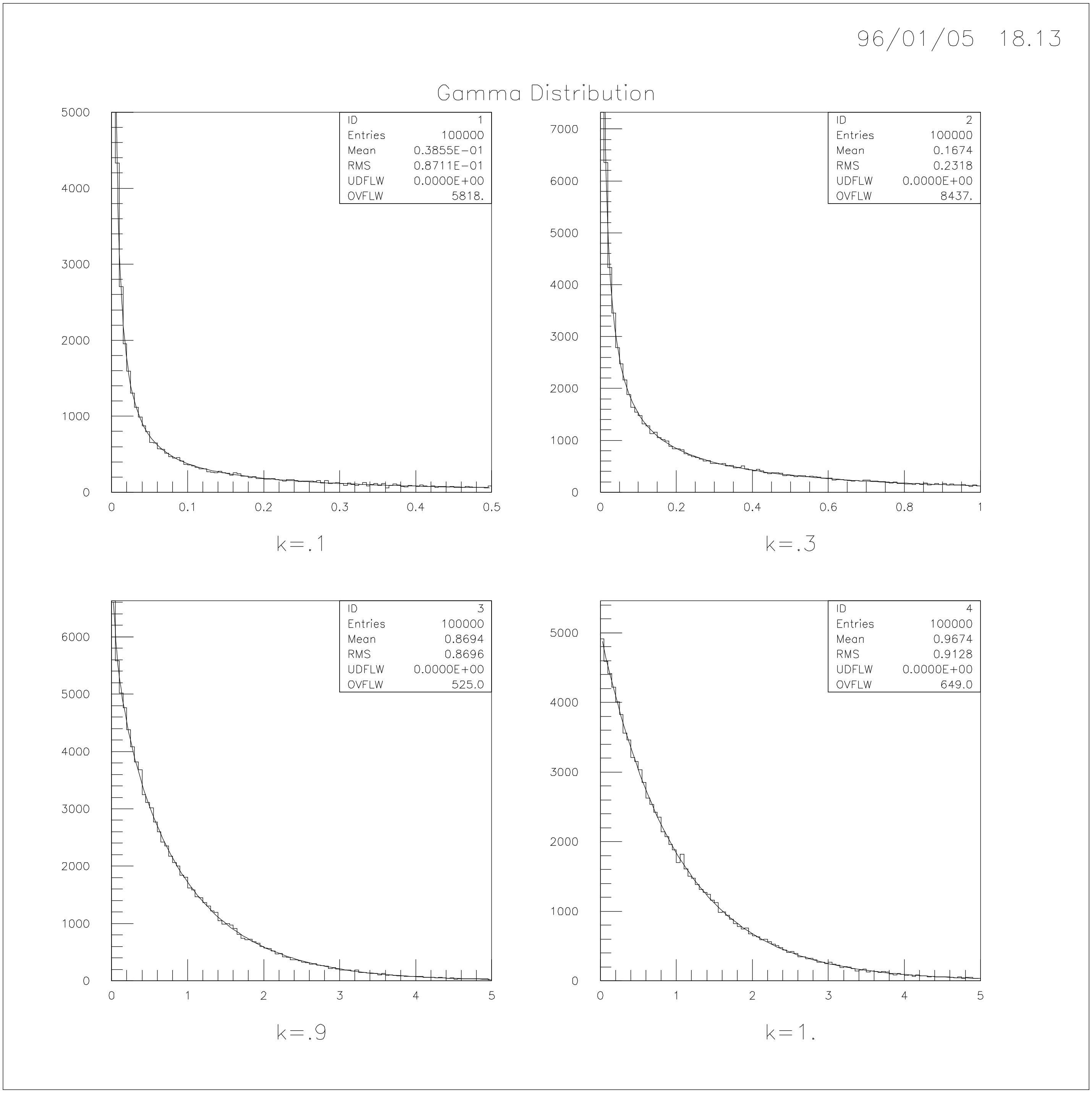
The special case $k = 1$ is called the *exponential* distribution. A sum of k' exponential random variables x_i is distributed as $f(\sum x_i; \lambda, k')$. Eq. 1.37 allows $k > 0$ to be nonintegral. If $\lambda = 1/2$ and $k = n/2$, the gamma and $\chi^2(n)$ distributions are identical.

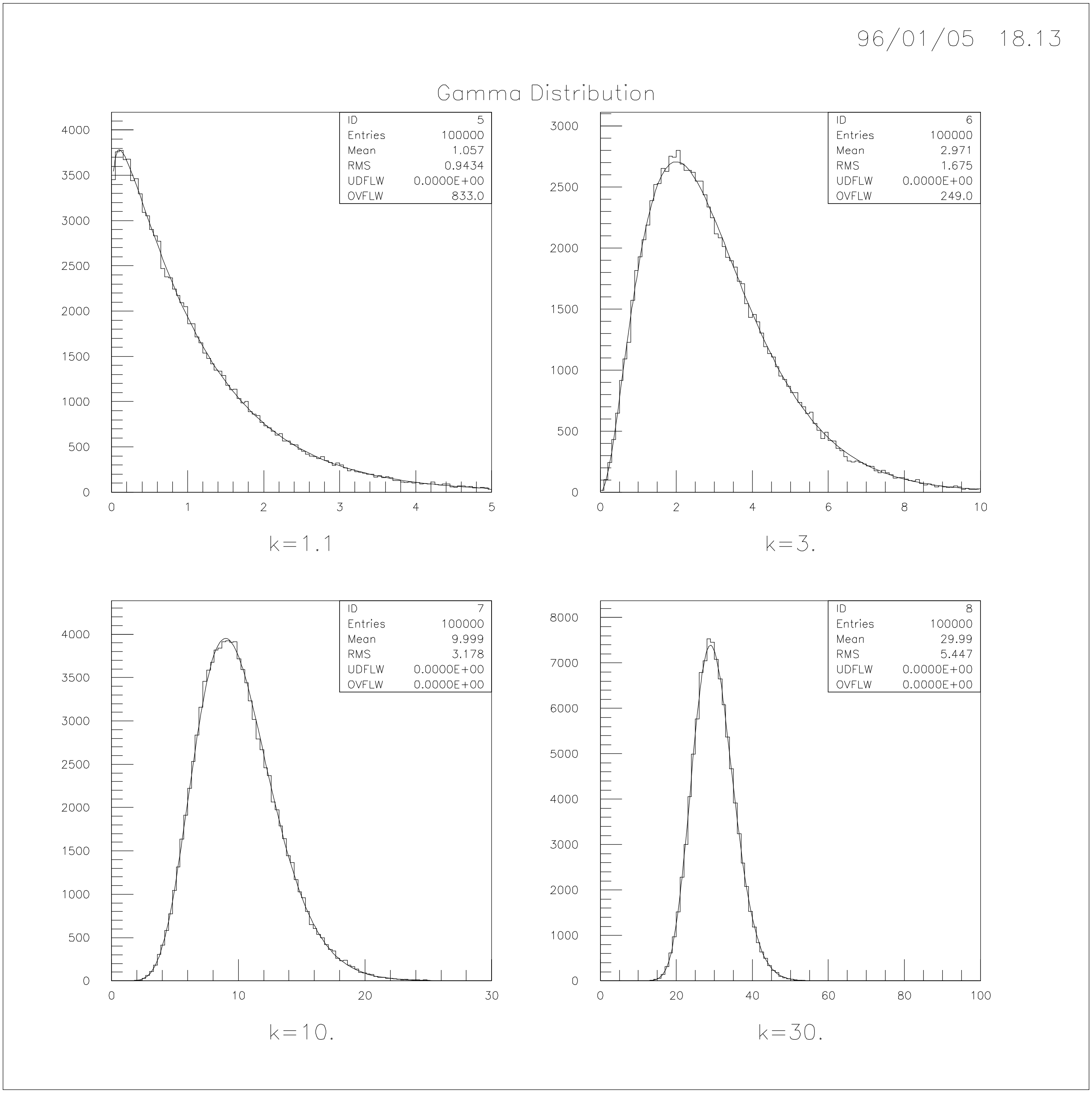
3.3.4 Algorithm for gamma distribution random number generator

All of the following algorithms are given for $\lambda = 1$. For $\lambda \neq 1$, divide the resulting random number x by λ. More elaborate and efficient algorithms can be found in Ref. 1.

- If $k = 1$ in Eq. 1.37 (the *exponential* distribution), accept $x = -(\ln u)$.
- If $0 < k < 1$, initialize with $v_1 = (e + k)/e$ (with $e = 2.71828\ldots$ being the natural log base). Generate u_1, u_2. Define $v_2 = v_1 u_1$. **Case 1:** $v_2 \leq 1$. Define $x = v_2^{1/k}$. If $u_2 \leq e^{-x}$, accept x and stop, else restart by generating new u_1, u_2. **Case 2:** $v_2 > 1$. Define $x = -\ln([v_1 - v_2]/k)$. If $u_2 \leq x^{k-1}$, accept x and stop, else restart by generating new u_1, u_2. Note that, for $k < 1$, the probability density has a pole at $x = 0$, so that return values of zero due to underflow must be accepted or otherwise dealt with.
- Otherwise, if $k > 1$, initialize with $c = 3k - 0.75$. Generate u_1 and compute $v_1 = u_1(1 - u_1)$ and $v_2 = (u_1 - 0.5)\sqrt{c/v_1}$. If $x = k + v_2 - 1 \leq 0$, go back and generate new u_1; otherwise generate u_2 and compute $v_3 = 64 v_1^3 u_2^2$. If $v_3 \leq 1 - 2v_2^2/x$ **or** if $\ln v_3 \leq 2\{[k-1]\ln[x/(k-1)] - v_2\}$, accept x and stop; otherwise go back and generate new u_1.

1. W.H. Press *et al.*, Numerical Recipes (Cambridge University Press, New York, 1986).


96/01/05 18.13
Gamma Distribution
ID 1
Entries 100000
Mean 0.3855E−01
RMS 0.8711E−01
UDFLW 0.0000E+00
OVFLW 5818.
k=.1
ID 2
Entries 100000
Mean 0.1674
RMS 0.2318
UDFLW 0.0000E+00
OVFLW 8437.
k=.3
ID 3
Entries 100000
Mean 0.8694
RMS 0.8696
UDFLW 0.0000E+00
OVFLW 525.0
k=.9
ID 4
Entries 100000
Mean 0.9674
RMS 0.9128
UDFLW 0.0000E+00
OVFLW 649.0
k=1.


96/01/05 18.13
Gamma Distribution
ID 5
Entries 100000
Mean 1.057
RMS 0.9434
UDFLW 0.0000E+00
OVFLW 833.0
k=1.1
ID 6
Entries 100000
Mean 2.971
RMS 1.675
UDFLW 0.0000E+00
OVFLW 249.0
k=3.
ID 7
Entries 100000
Mean 9.999
RMS 3.178
UDFLW 0.0000E+00
OVFLW 0.0000E+00
k=10.
ID 8
Entries 100000
Mean 29.99
RMS 5.447
UDFLW 0.0000E+00
OVFLW 0.0000E+00
k=30.

```
*** Begin of mkplots.kumac: Thu Jun  9 1994
*
*   Make the gamma distribution histograms and and functions
*   to check the gamma distribution generator in RPP 94.
*
exec gamdis#init
zone 2 2
*
exec gamdis 1.   .1  0.5 100000.
exec gamdis 2.   .3  1.  100000.
exec gamdis 3.   .9  5.  100000.
exec gamdis 4.  1.   5.  100000.
@ gamdis_1.ps
*
exec gamdis 5.  1.1  5.  100000.
exec gamdis 6.  3.  10.  100000.
exec gamdis 7. 10.  30.  100000.
exec gamdis 8. 30. 100.  100000.
@ gamdis_2.ps
*
*** End   of mkplots.kumac: Thu Jun  9 1994
```

```
*** Begin of gamdis.kumac: Tue Jun  7 1994
*
*   Histogram random numbers for the continuous gamma distribution and
*   plot the function for comparison.  This is to check the algorithm for
*   generating the random numbers for the continuous gamma distribution.
*
*   The gamma distribution (for lambda = 1) is:
*       f(x;k) = x**(k-1) * exp(-x) / gamma(k)
*
*   First initialize:
*       exec gamdis#init
*
*   Then make histograms and the function:
*       exec gamdis ID K Xmax [Count] [Nbins]
*
*
***   Histogram the random numbers and plot the function.
*
macro fit id k xmax count=10000. nbins=100.
*
*                Create and fill the histogram.
hist/create/1dhist [id] k=[k] [nbins] 0. [xmax]
fort/call mkgam.f([id],[k],[count])
*                Set up input for gamdis.f
h = [xmax] * [count] / [nbins]
vect/input parm [h] [k]
*                Limit the scale if k < 1
if [k] < 1.  then
    x = [xmax] / [nbins]
    call getgmds.f([x])
    ymax = 1.11 * (1. + [x]/2.) * (0.7 + 0.3*[k]) * vout(1)
    hist/set/max [id] [ymax]
endif
*                Plot the function
hist/fit [id] gamdis.f 'bq' 2 parm step pmin pmax err
return
```

```
*** gamdis.kumac:  (continued)
*
*
*
***   Initialize.
*
macro init
*
hist/create/title 'Gamma Distribution'
option nfil
vector/create vout(1)
*
vector/create parm(2)
vector/create step(2)
vector/create pmin(2)
vector/create pmax(2)
vector/create err(2)
vector/input step 0. 0.
*
return
*
*** End   of gamdis.kumac: Tue Jun  7 1994
```

```
*** Begin of mkgam.f
*
      SUBROUTINE RANGAM(K,X)
*
*           Generate a random number from the gamma distribution.
*           This algorithm is from the 1994 edition of RPP.
*
*           The gamma distribution (for lambda = 1) is:
*                 f(x;k) = x**(k-1) * exp(-x) / gamma(k)
*
*            Here K is input and X is output.
*
      IMPLICIT  NONE
*
      REAL K,  X
      REAL KK, XX, C,  U1, U2, V1, V2, V3, T, D
*
      REAL RNDM
*
      REAL E
      PARAMETER ( E = 2.718281828 )
*
*                 Fetch the input K
      KK = K
*                 Select type of K
      IF ( KK.GT.1. )  THEN
          C  = 3.*KK - 0.75
          XX = -1.
          DO WHILE ( XX.LE.0. .OR. V3.GT.T )
              D  = XX
              U1 = RNDM(D)
              V1 = U1 * (1.-U1)
              V2 = (U1-0.5) * SQRT( C/V1 )
              XX = KK + V2 - 1.
              IF ( XX.GT.0. )  THEN
                  U2 = RNDM(-D)
                  V3 = 64. * V1 * ( V1 * U2 )**2
                  T  = 1. - 2.*V2**2/XX
                  IF  ( V3.GT.T )  THEN
                      V3 = LOG(V3)
                      T  = 2. * ( (KK-1.) * LOG(XX/(KK-1.)) - V2 )
                  ENDIF
              ENDIF
          ENDDO
```

```
*** mkgam.f:  (continued)
*
*
      ELSE IF ( KK.EQ.1. )  THEN
          D  = KK
          U1 = RNDM(D)
          XX = - LOG(U1)
*
      ELSE IF ( KK.GT.0. )  THEN
          V1 = ( E + KK ) / E
          U2 = 1.
          T  = 0.
          DO WHILE ( U2 .GT. T )
              D  = T
              U1 = RNDM(D)
              U2 = RNDM(-D)
              V2 = V1 * U1
              IF ( V2.LE.1. )  THEN
                  XX = V2**(1./KK)
                  T  = EXP(-XX)
              ELSE
                  XX = -LOG( (V1-V2) / KK )
                  T  = XX**(KK-1.)
              ENDIF
          ENDDO
*
      ELSE
          XX = -1.
*
      ENDIF
*                 Store the result
      X = XX
*
      RETURN
      END
```

```
*** mkgam.f:  (continued)
*
      SUBROUTINE MKGAM(ID, K, COUNT)
*
*           Calculate and plot the 1994 PDG gamma distribution random
*           number generator.  This is for PAW.  June 7, 1994.
*
*           ID is histogram ID
*           K is the parameter k in the gamma distribution. (lambda = 1)
*           COUNT is the number of random numbers to generate and plot.
*
      IMPLICIT  NONE
*
      REAL    ID, K, COUNT
      REAL    X
      INTEGER IID, N, I
*
*
      IID = ID
      N = COUNT
      DO I = 1,N
          CALL RANGAM(K, X)
          CALL HFILL(IID, X, 0., 1.)
      END DO
      RETURN
      END
*
*** End   of mkgam.f
```

```
*** Begin of gamdis.f
*
      REAL FUNCTION GAMMA(X)
*           This is the function GAMMA in CERNLIB package C305
      REAL X
      DOUBLE PRECISION U,V,F,ZERO,ONE,THREE,FOUR,PI
      DOUBLE PRECISION C(0:15),H,ALFA,B0,B1,B2
      DATA ZERO /0.0D0/, ONE /1.0D0/, THREE /3.0D0/, FOUR /4.0D0/
      DATA PI    /3.14159265358979324D0/
      DATA C( 0) /3.65738772508338244D0/
      DATA C( 1) /1.95754345666126827D0/
      DATA C( 2) / .33829711382616039D0/
      DATA C( 3) / .04208951276557549D0/
      DATA C( 4) / .00428765048212909D0/
      DATA C( 5) / .00036521216929462D0/
      DATA C( 6) / .00002740064222642D0/
      DATA C( 7) / .00000181240233365D0/
      DATA C( 8) / .00000010965775866D0/
      DATA C( 9) / .00000000598718405D0/
      DATA C(10) / .00000000030769081D0/
      DATA C(11) / .00000000001431793D0/
      DATA C(12) / .00000000000065109D0/
      DATA C(13) / .00000000000002596D0/
      DATA C(14) / .00000000000000111D0/
      DATA C(15) / .00000000000000004D0/
```

```
*** gamdis.f:  (continued)
*
*      DOUBLE PRECISION D
*      REAL ROUND
*      ROUND(D)  =  SNGL(D+(D-DBLE(SNGL(D))))
      U=X
      V=U
      IF(X .LE. ZERO) THEN
       IF(X .EQ. INT(X)) THEN
        GAMMA=ZERO
        RETURN
       ELSE
        U=ONE-U
       END IF
      END IF
      F=ONE
      IF(U .LT. THREE) THEN
       DO 1 I = 1,INT(FOUR-U)
       F=F/U
    1  U=U+ONE
      ELSE
       DO 2 I = 1,INT(U-THREE)
       U=U-ONE
    2  F=F*U
      END IF
      U=U-THREE
      H=U+U-ONE
      ALFA=H+H
      B1=ZERO
      B2=ZERO
      DO 3 I = 15,0,-1
      B0=C(I)+ALFA*B1-B2
      B2=B1
    3 B1=B0
      U=F*(B0-H*B2)
      IF(V .LT. ZERO) U=PI/(SIN(PI*V)*U)
*      GAMMA=ROUND(U)
      GAMMA  =  SNGL(U+(U-DBLE(SNGL(U))))
      RETURN
      END
```

```
*** gamdis.f:  (continued)
*
      FUNCTION GAMDIS(X)
*
*                 Form the gamma distribution function (for lambda = 1).  It is:
*                       f(x;k) = x**(k-1) * exp(-x) / gamma(k)
*
*                 Calculate GAMDIS = height * f(x;k)
*                 PAR(1) = height
*                 PAR(2) = k
*
      IMPLICIT  NONE
*
      REAL  X, GAMDIS, K
      REAL  GAMMA
*
      REAL PI
      PARAMETER ( PI = 3.14159265359 )
*
      REAL PAR
      COMMON /PAWPAR/ PAR(2)
*
*                 First get k.
      K = PAR(2)
*                 Calculate the gamma distribution
*                      Large K.  Approximate the gamma function as
*                      Gamma(K) = K**K * Exp(-K) * Sqrt(2*Pi/K) * (1+1/(12*K))
      IF ( K .GT. 20. )  THEN
          GAMDIS = (X/K)**(K-1.) * EXP(K-X) / SQRT(2.*PI*K) * PAR(1)
          GAMDIS = GAMDIS / (1.+1./(12.*K))
*                      Small K and large X.  Set the distribution to zero.
      ELSE IF ( X .GT. 100. )  THEN
          GAMDIS = 0.
*                      Normal case.  Calculate the distribution exactly.
      ELSE
          GAMDIS = X**(K-1.) * EXP(-X) / GAMMA(K) * PAR(1)
*
      ENDIF
*
      RETURN
      END
```

```
*** gamdis.f:  (continued)
*
      SUBROUTINE GETGMDS(X)
*
*                 Get the gamma distribution function in vector vout.
*
*                 This subroutine name MUST be logically linked to gamdis.f
*                 with the UNIX command:
*                      ln -si gamdis.f getgmds.f
*
      IMPLICIT  NONE
*
      REAL PAR
      COMMON /PAWPAR/ PAR(2)
*
      REAL   GAMDIS,  X
      VECTOR PARM(2), VOUT(1)
*
*                      Load the parameter common /PAWPAR/
      PAR(1) = PARM(1)
      PAR(2) = PARM(2)
*                      Get the value of the gamma distribution.
      VOUT(1) = GAMDIS(X)
      RETURN
      END
*
*** End   of gamdis.f
```