I. ABSTRACT

The estimation of the size of the T-year flood is a basic problem in hydrology. One important source of uncertainty in this estimate is that caused by the uncertain estimation of the parameters in the log Pearson III distribution which describes the occurrence of the maximum annual discharge. We describe here a program which, by means of a simulation, allows the computation of confidence intervals for the magnitude of the T-year flood as well as for determining the distribution of the estimates for this flood.

II. KEY WORDS

T-year flood, Confidence Interval, Log Pearson III Distribution, Simulation

III. INTRODUCTION

It has been known for a long time, and is given in Bulletins 17A and 17B of the Water Resources Council [1,2], that in the case of a flood distribution whose logarithm is normally distributed, confidence intervals for the T-year flood can be obtained by the use of the central student's t-distribution.

In the more general case, following the guidelines of Bulletins 17A and 17B, is when the logarithm of the flood distribution has a Pearson III distribution with a non-zero skew parameter. This case of non-zero skew is more complicated than the case of a lognormally distributed flood which is the case of zero skew [3,4,5,6,7,8,9]. In an important paper [9] Steedinger shows that the confidence intervals, for quantiles, which are given in the U.S. Water Resources Council guidelines [1,2] are not satisfactory. He uses a formula due to Kite [10] and derives an expression for confidence intervals for the quantiles which he shows is satisfactory in several simulations.

In [6] Harishtin simulated a random variable which we will call the "non-central gamma distribution"; note that this statistics name is also used for a different random variable. He uses this simulation to assess the accuracy of an approximate formula for confidence limits for flood-fraction curves given in Bulletin 17A. This is one of the formulas which Steedinger in [9] found so be unsatisfactory. Harishtin's simulation involved 1 values of 10, 50, 100, and 500 years, sample sizes 10, 20, and 40, skew coefficients of -1.0 and 1.0, levels of significance 0.05, 0.01, 0.001, and 0.0001, and were each determined by means of 1000 point simulations.

To obtain more accuracy and to cover a more extensive range of T, sample sizes, skew, and levels of significance than [6], a program was developed to simulate a set of values of the non-central gamma random variable. For the purpose of comparing choices of probabilities and the size of the associated one-sided confidence intervals it is useful to have a table of the percentiles of the non-central gamma distribution; our simulation gives percentiles for the range 5% (3%) 95%. For applications of this see [10]. From this table an empirical distribution function for the value $Z_T$ of the T-year flood is easily found and this distribution is essential for other hydrological calculations; for an example see [11].

IV. MATHEMATICAL DISCUSSION

A. Zero skew. The case of zero skew is a model for the case of non-zero skew. So we first consider the case where $X$, the logarithm of the maximum annual discharge, has a normal distribution. For the T-year flood, take $p=1/T$, and let $Y_p$ be the p-th quantile of $X$; i.e.,

$$P(Y_p \leq y) = p$$  

It is $Y_p$ that we want to estimate. If the mean $\mu$ and standard deviation $\sigma$ of $X$ were known, then since $\log(Y_p)/\sigma$ has a N(0,1) distribution, i.e., a normal distribution with mean 0 and standard deviation 1,

$$Y_p = \mu + \sigma Z_p$$

where $Z_p$ is the p-th quantile for an N(0,1) distribution. Of course $\mu$ and $\sigma$ are not actually known; we only have estimates for them, $\hat{\mu}$ and $\hat{\sigma}$, based on m data points. It is natural to use the estimate

$$\hat{Y}_p = \hat{\mu} + \hat{\sigma} Z_p,$$

but it is important to note that the estimator $\hat{Y}_p$ is itself a random variable depending on $\hat{\mu}$ and $\hat{\sigma}$.  

0266-9463/86/050138 - 13 $2.00
© COMPUTATIONAL MECHANICS PUBLICATIONS 1986

Consider
\[ (y_p, \beta) = (\alpha, \beta) \gamma_p/\beta(\gamma_p/\beta). \]  
Equation (14) can be written, see (12), as
\[ (1/\gamma_p)(1/\beta) \sim \text{Gamma}(m/\gamma_p, \beta). \]  

The random variable in brackets in (5) has a non-central \( t \)-distribution, with non-centrality parameter \( \beta \gamma_p/\beta \). The special case \( \gamma_p = 0 \) is the student's \( t \)-distribution.

Since the distribution of (4) can be written in terms of the known non-central \( t \)-distribution, confidence intervals for \( y_p \) can be determined, because of the way in which the flood design value is used, the confidence interval of greatest interest is a one-sided interval which gives an upper bound, but of course other confidence intervals can be obtained by using the appropriate percentiles of the non-central \( t \)-distribution. And, as was mentioned above, the empirical distribution function is also needed for other problems. The choice of the value \( \gamma \), for the 1-year flood, and the value \( m \) of data points determine the non-centrality parameter \( \beta \). The other quantity which must be specified is a probability \( \gamma \), for the one-sided confidence interval. If \( \gamma_p \) is the \( p \)-th quantile of the non-central \( t \)-distribution, then
\[ F(y_p, \beta) = \text{Beta}(m/\gamma_p, \beta) \]  
and we have found the one-sided 100(1-\( \gamma \)) percent confidence interval for \( y_p \). The choice of the set of values \( \gamma = .01, .05, .09 \) gives an empirical distribution function.

B. Non-zero skew. In this case, the logarithm \( x \) of the yearly peak discharge is assumed to have a Pearson type 3 distribution with density function
\[ f(x) = 1/(a \Gamma(b/c)) x^{b} e^{-c-x} \]  
where, in the case of positive \( a \), the density is given by (7) for \( x > 0 \) and is zero for \( x < 0 \), while in the case of negative \( a \) the density is given by (7) for \( x < 0 \) and is zero for \( x > 0 \). Computing the mean \( \mu \), standard deviation \( \sigma \), and skew \( \gamma \) from (7) shows that (17) has the same sign as \( \gamma \). In following the guidelines of bulletin 1/4 and 1/6, the skew coefficient \( \gamma \) is estimated either from a map of regional skew or from a large pool of data from that region. Consequently the error in estimating \( \gamma \) is of an entirely different kind than that which arises in estimating \( \mu \) and \( \sigma \) by means of mean and standard deviation \( \mu \) and \( \sigma \) of a population for the specific area for which the 1-year flood is being estimated, what is usually done to simplify this complicated situation, and what was done for the simulation, is to suppose that \( \gamma \) is given exactly. This focuses attention on that part of the variability in the estimate of the 1-year flood which arises from the uncertainty in the estimation of \( \mu \) and \( \sigma \), and ignores the variability which comes from the uncertainty in the estimate of \( \gamma \).

The form of the density (7) shows that the random variable
\[ z = (x - c)/a \]  
has the one parameter density
\[ g(z; \beta) = \beta^{-1} e^{-z \beta}. \]  
for \( z > 0 \) and 0 for \( z < 0 \), i.e. \( z \) has a gamma distribution with shape parameter \( \beta \) and scale parameter 1.

In (10) it is shown that if \( t_{\gamma_p} \) is the 100(1-\( \gamma \)) percentile for the gamma distribution \( \gamma \), \( \beta \) and \( \gamma \) are the usual estimators for the mean of \( z \) and the variance of \( z \) using \( m \) sample points, then
\[ (y_p - \beta)/\beta = t_{\gamma_p}/\beta \]  
for \( \gamma = 0 \) and
\[ (y_p - \beta)/\beta = (t_{\gamma_p} - 1)/\beta \]  
for \( \gamma > 0 \).

So confidence intervals for \( y_p \) can be obtained, for \( z \) having the gamma distribution (10), we can find the distribution of
\[ \{(t_{\gamma_p} - 1)/\beta \}. \]

It is this distribution which has been called the noncentral gamma distribution (10).

V. PROGRAMMING DISCUSSION

In order to simulate the distribution of the random variable or equations (11) we first must find the percentile \( t_{\gamma_p} \) or \( t_{\gamma_p} \) of the gamma distribution \( \gamma \). In our program this is obtained by first finding the corresponding percentile of the normal \( N(0,1) \) distribution, and to do this Hastings's method of approximation (14, page 40) for the normal distribution function \( F \) is used to find \( F(z) \) and interval halving is used to find the value of \( z \) with \( F(z)/p \) or the \( z \) with \( F(z)/p \) for the function "NormCDF" and the function "InversCDF" in the program listing. Then the Wilson-Hilferty transformation (15,16) is applied to this percentile for the normal to get the percentile for the gamma distribution; see the function "Wilson-Hilferty" in the program.

Next we must generate a sequence of random values whose distribution is that of the given gamma distribution \( \gamma \). To do this, we used the ingenious Box-Muller method (17,18) for generating normally distributed random variables and transform the Wilson-Hilferty transformation of those variables to get the desired sequence; see the procedure "RandomNormal" in the program.

Now for a given \( m \), \( m \) independent gamma distributed random variables can be generated; the sample mean and standard deviation calculated, and, together with the gamma percentile which we have discussed, simulating above, the value of the random variable given by equations (11) can be calculated. This is done in procedure "means" and function "statistics". The purpose of the function "statistics" is to separate out the three cases of skew equal zero, the case of a normal distribution, skew greater than zero, the case of the first of the two equations (11), and skew less than zero, the case of the second of the two equations (11).

In order to use a sequence of numbers calculated as discussed above to get an approximate probability distribution for the random variable \( z \) we are simulating, a histogram must be constructed for this sequence. To this end, a small sequence of size 100 is calculated in order to be able to scale the numbers so that they will almost all lie in \([0,1]\) and also so that they will be spread out enough in \([0,1]\) to obtain good accuracy for the distribution function; see procedure "scale". Once this scaling is...
Confidence Intervals for Floods I: R. Whitley and T. Hromadka

found, the interval (t-σ) is divided up with a step size of .01. Any in the procedure "tally" a count is kept of the number of time a random variable falls in this subinterval or length .01.

The procedure "percentiles" uses the results of "tally" and linear interpolation to give the percentiles .05, .01, .05. The user inputs the number of m, or data points to be used in the sample mean and sample standard deviation, the skew coefficient, and the value T for the 1-year flood. The number of blocks to be used in the simulation, each of size 1000, can also be chosen by the user. For example, for m = 30 data points and a choice of 10 blocks, the program will 10,000 times sample 30 points thereby sending 10,000 values of the statistic, given by the function "statistic", in the procedure "tally" and in the process of doing this it will generate 80,000 gamma distributed random variables.

For output, the user first sees the percentiles for the distribution, see the procedure "percentiles" and the sample output below.

Concerning the accuracy of a simulation, we quote (12), page 85: "Appreciably, however, the testing of generators of random vectors would best be carried out by constructing a simulation problem to which the solution is known, and which requires essentially the same characteristics as the problem which is ultimately to be solved." This advice is implemented in the procedure "percentiles." Although the main purpose of the procedure "tally" is to tally up the histogram for the statistic of interest, it also keeps a tally for the the sample mean; the distribution of the sample mean of a gamma distribution is known exactly and as one check on the accuracy of the simulation the difference between the sample mean percentile and the simulated sample mean percentile is printed out along with the percentiles of the statistic. Another check on the simulation accuracy is obtained by running the case of zero skew. The resulting distribution is the noncentral t-distribution and the simulation percentiles can be checked against published tables (16) or by generating the noncentral t-distribution itself for which see, for example, (20).

The user can also choose to see the entire histogram from which the percentiles have been interpolated, see the procedure "histogram" and the user can choose to input estimates for the mean and standard deviation at a specific site so as to see the transformed simulation percentiles which give the percentiles in terms of the discharge values see the procedure "floodpercentiles.

The program "confidence", described above and listed below, was written in Turbo-Pascal with IBM-PC support, on the IBM-PC. With the Intel 8087 numerical data processor chip, a simulation like the example given below, which involves the generation of several hundred thousand gamma distributed random numbers, takes roughly a half-hour. The program would be an order of magnitude slower without the 8087.

VI. REFERENCES

PERCENTILES

(linearly interpolated from histogram)

$m = $sample mean, $s$ sample s.d.

with 30 points in the sample from a gamma random variable with

\[
gamma = 2.5000
\]

\[
b = 64.0000
\]

$K_p = 2.5000$  
$T_p = 84.07267$  
This is the $K_p$ and $T_p$ for the 100.0 year flood.

\[
Y = \{ \frac{(T_p - m)}{s} \}
\]

error in mean $= a - true$ sample mean

Number of points in the simulation: 100 blocks of size 1000

<table>
<thead>
<tr>
<th>Percentile</th>
<th>$Y$</th>
<th>$Y-K_p$</th>
<th>error in mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.00</td>
<td>1.950</td>
<td>-0.559</td>
<td>-0.004</td>
</tr>
<tr>
<td>10.00</td>
<td>2.071</td>
<td>-0.436</td>
<td>0.010</td>
</tr>
<tr>
<td>15.00</td>
<td>2.154</td>
<td>-0.365</td>
<td>0.012</td>
</tr>
<tr>
<td>20.00</td>
<td>2.225</td>
<td>-0.284</td>
<td>0.013</td>
</tr>
<tr>
<td>25.00</td>
<td>2.286</td>
<td>-0.224</td>
<td>0.025</td>
</tr>
<tr>
<td>30.00</td>
<td>2.336</td>
<td>-0.171</td>
<td>0.011</td>
</tr>
<tr>
<td>35.00</td>
<td>2.390</td>
<td>-0.119</td>
<td>0.012</td>
</tr>
<tr>
<td>40.00</td>
<td>2.440</td>
<td>-0.069</td>
<td>0.017</td>
</tr>
<tr>
<td>45.00</td>
<td>2.490</td>
<td>-0.019</td>
<td>0.007</td>
</tr>
<tr>
<td>50.00</td>
<td>2.541</td>
<td>0.032</td>
<td>0.017</td>
</tr>
<tr>
<td>55.00</td>
<td>2.599</td>
<td>0.090</td>
<td>0.016</td>
</tr>
<tr>
<td>60.00</td>
<td>2.650</td>
<td>0.141</td>
<td>0.008</td>
</tr>
<tr>
<td>65.00</td>
<td>2.706</td>
<td>0.197</td>
<td>0.009</td>
</tr>
<tr>
<td>70.00</td>
<td>2.774</td>
<td>0.255</td>
<td>0.008</td>
</tr>
<tr>
<td>75.00</td>
<td>2.844</td>
<td>0.335</td>
<td>0.014</td>
</tr>
<tr>
<td>80.00</td>
<td>2.923</td>
<td>0.414</td>
<td>0.019</td>
</tr>
<tr>
<td>85.00</td>
<td>3.016</td>
<td>0.507</td>
<td>0.011</td>
</tr>
<tr>
<td>90.00</td>
<td>3.147</td>
<td>0.638</td>
<td>-0.009</td>
</tr>
<tr>
<td>95.00</td>
<td>3.349</td>
<td>0.840</td>
<td>-0.032</td>
</tr>
</tbody>
</table>

FLOOD PERCENTILES

10 to the power m + z*Yp

Yp the percentile of Y

given in the percentile table

$m = 3.400000$

$s = 0.200000$

skew = 0.2500

Number of data points = 30

For the T year flood, T = 100.0

<table>
<thead>
<tr>
<th>Percentile</th>
<th>Flood</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>6186</td>
</tr>
<tr>
<td>0.10</td>
<td>6315</td>
</tr>
<tr>
<td>0.15</td>
<td>6774</td>
</tr>
<tr>
<td>0.20</td>
<td>6998</td>
</tr>
<tr>
<td>0.25</td>
<td>7193</td>
</tr>
<tr>
<td>0.30</td>
<td>7372</td>
</tr>
<tr>
<td>0.35</td>
<td>7550</td>
</tr>
<tr>
<td>0.40</td>
<td>7728</td>
</tr>
<tr>
<td>0.45</td>
<td>7906</td>
</tr>
<tr>
<td>0.50</td>
<td>8085</td>
</tr>
<tr>
<td>0.55</td>
<td>8313</td>
</tr>
<tr>
<td>0.60</td>
<td>8512</td>
</tr>
<tr>
<td>0.65</td>
<td>8735</td>
</tr>
<tr>
<td>0.70</td>
<td>8962</td>
</tr>
<tr>
<td>0.75</td>
<td>9029</td>
</tr>
<tr>
<td>0.80</td>
<td>9252</td>
</tr>
<tr>
<td>0.85</td>
<td>9473</td>
</tr>
<tr>
<td>0.90</td>
<td>10701</td>
</tr>
<tr>
<td>0.95</td>
<td>11745</td>
</tr>
</tbody>
</table>
VIII. PROGRAM LISTING

program confidence;
(Ted Hromadka, Williamson and Schmid, 17/82 Sky Park Blvd. Irvine, CA 92714
Robert Whitley, Dept. of Math., Univ. of Ca., Irvine, CA, 92717
1986)

(This program computes confidence percentiles, 5% (5%) 95%, for sampling
from a Pearson type III distribution with a given skew parameter.
Input consists of this skew parameter, T for the T-year flood, and the
number of datapoints used in the sample mean and sample standard deviation.
The user can choose the number of blocks of size 100% in the simulation
to be between 1 and 30 and can print out the total histogram of the
simulation if desired. Given estimates for the mean and the standard
deviation of the log (base 10) of specific data, percentiles for this
data can be printed.)

const
block = 1000;
maxdata = 50;
b = 0.01; (step size in the tally)
L = 2; (simulation tallied in [-L,L])
maxj = 399; (-(L/L)-1)
maxjplusone = 400;
scaledint = 100; (sample size used to get scalings)
leastint = -32767; (almost) smallest integer in turbo pascal
 greatestint = 32767; (largest integer in turbo pascal)
type
maxvector = array[1..maxdata] of real;
tvector = array[0..maxj] of real;
factor = array[0..18] of real;
var
NoBlocks:integer; (# of blocks used in simulation)
NoDataPoints:integer; (# of points used in the sample estimators for
the distribution parameters)
dvector;maxvector; (vector of nodatapoints r.v. simulating a
sequence of datapoints)
tally1;tvector; (histogram tallies of the mean as a check)
tally2;tvector; (histogram tallies of the statistic Y of
interest; see procedure "statistic")
pfactor;factor; (pfactor[j] is the 0.05×j×0.05) percentile
of Y for j=0(1)18
gamma;real;
h;real; (skew coefficient)
sqr(gamma);4/b, gamma is zero and b infinity
 corresponds to the case of a normal distribution)
tp;real; (F(tp)=1-1/T, F N(0,1) d.f, if gamma is zero, else
F is the gamma d.f. with parameter b and
F(tp)=1-p for gamma positive while F(tp)=p
for gamma negative.)
Kp;real; (point estimate for the p-th percentile of a
Pearson III is mean + s.d. × Kp)
T;real; (the T-year flood)
p;real; (p=1/(1/T))
scale1;real; (mean-mid1)/scale1 scales the mean to [-2,2])
mid1;real;
scale2;real; ((Y-mid2)/scale2 scales the statistic Y
to the range [-2,2])
mid2;real;
GammaNotZero:boolean; (false in the non-normal case of finite b)
GammaPositive:boolean; (true if gamma>0)
ans1;char; (no if do not print histogram)
ans2;char; (no if don't want percentiles for specific data)
x,y;real;k,j;integer;

function NormalF(x:real):real;
(see Abramowitz and Stegun, Handbook of Mathematical Functions, page 932.
NormalF is the distribution function for a normal N(0,1)
const
    p = 0.2318419;
    b1 = 0.3988504779;
    b2 = -0.2663619713;
    b3 = 0.9952010865;
    b4 = -0.7722847844;
    b5 = 0.2733476219;

    var
    temp, temp2.t: real;
    begin (function NormalF)
        t := 1/(1+|p|);
        temp := b4*b5*t;
        temp := b3*temp*t;
        temp := b2*temp*t;
        temp := b1*temp*t;
        temp := temp*t;
        temp2 := exp(-t*x/2)/(sqrt(2*pi));
    end; (function NormalF)

    function InverseF(p:real):real;
    {finds x such that F(x)=p by interval halving}
    const
        small = 10E-8;
    var
        x1, x2, x3: real;
        F1, F2, F3: real;
    begin (function InverseF)
        x1 := 0;
        F1 := NormalF(x1);
        x2 := 1;
        F2 := NormalF(x2);
        while p > F2 do begin
            x1 := x2;
            F1 := F2;
            x2 := x2 + 1;
            F2 := NormalF(x2);
        end; (of while)
        repeat
            x3 := (x1 + x2)/2;
            F3 := NormalF(x3);
            if F3 > p then begin
                x2 := x3;
                F2 := F3;
            end (of if)
            else begin
                x1 := x3;
                F1 := F3;
            end;
        until F2 - F1 < small;
    end; (function InverseF)

    procedure RandomNormal(var rn1, rn2: real);
    {a call to RandomNormal(x,y) returns two independent 
     normally distributed r.v. as x and y. See Newman and Odell, 
     The Generation of Random Variates, page 19, or Abramowitz 
     and Stegun, Handbook of Mathematical Functions, page 953 
     }
    var
        t1, t2, t3: real;
    begin
        t1 := random;
        t2 := random;
        t3 := sqrt(-2*ln(t1));
        rn1 := t3*sin(2*pi*t2);
        rn2 := t3*cos(2*pi*t2);
    end; (procedure randomnormal)
function WilsonHilferty(t,shape:real):real;
(A call to WilsonHilferty(t,shape), with t normally distributed, returns a variable which has (approximately) a gamma distribution with a (shape) parameter "shape". See Kendall and Stuart, The Advanced Theory of Statistics, Vol.1, page 400, and see Mathur, A Note on Wilson-Hilferty Transformation of Chi-squared, Bull. of the Calcutta Stat. Assoc.18(1968)100-106 for a discussion of the error in this approximation. It is very good as long as the number of d.f., which is 2b when you convert gamma, with shape parameter b, to chi-squared, is >14.)

var
  temp:real;
begin
  temp:=1-(1/(shape))+sqrt(1/(shape));
  temp:=temp**(shape);
  WilsonHilferty:=temp;
end;(function WilsonHilferty)

procedure RandomGamma(var rnl,rn2:real);
(A call to randomgamma(x,y) returns two independent gamma distributed r.v. as x and y. The approximation used is discussed in the function WilsonHilferty. Note that if $X$ has a Pearson II distribution with parameters a,b, and c, then $(X-c)/a$ has a gamma distribution with parameter b. This procedure assumes a global shape parameter $b$ for the gamma distribution.

var
  t1,t2:real;
begin
  RandomNormal(t1,t2);
  rnl:=WilsonHilferty(t1,b);
  rn2:=WilsonHilferty(t2,b);
end;(procedure RandomGamma)

procedure datavector;
(Procedure is to generate a vector of dimension nodatapoints whose components are independent with the same gamma distribution with skew parameter gamma; in the case of a zero skew parameter, they are normally distributed.)

var
  j:integer;
  a,b:real;
begin
  j:=1;
  u:=trunc(nodatapoints/2)+0.5;
  repeat
    if gammarnotzero then randomgamma(a,b) else randomnormal(a,b);
    dvector[2*j-1]:=a;
    dvector[2*j]:=b;
    j:=j+1;
    until a=0;
  if odd(nodatapoints) then begin
    if gammarnotzero then randomgamma(a,b) else randomnormal(a,b);
    dvector[nodatapoints+1]:=a;
  end;
end;(procedure datavector)

procedure MeanStDev(var mean,sd:real);
(Returns the mean and standard deviation of the data in the data vector.)

var
  j:integer;
begin
  mean:=0;
  sd:=0;
  for j:=1 to nodatapoints do begin
    mean:=mean + dvector(j);
    sd:=sd + sqr(dvector(j));
  end;
  mean:=mean/nodatapoints;
  sd:=sqrt(sd/nodatapoints);
end;(procedure MeanStDev)
end; (of for)
mean: = mean/numdatapoints;
sd: = sqrt(mean); (unbiased)
end. (of procedure mands)

function statistic(x,y: real): real;
\{ Given mean \( x \) and standard deviation \( y \), this computes the statistic which is being simulated. \}
begin
  if gamma > 0 then
    begin
      if gama > 0 then statistic: = ((x-m)/y)
        else statistic: = -((x-m)/y);
        end
  else statistic: = ((x-m)/y); (then gamma is zero)
end; (function statistic)

procedure scale(var scale1, scale2, mid1, mid2: real);
\{ procedure scale samples the mean and statistic scale test times and keeps track of the max and min and sets the scale factors so that \( (x-mid1)/scale1 \) is exactly scaled to \([-1,1]\) for the mean and \( (x-mid2)/scale2 \) is exactly scaled to \([-1,1]\) for the statistic. For the definition of the statistic see the function statistic. The procedure tally then counts the occurrences of the scaled numbers in \([-2,2]\) with an interval size of \( h \). \}
begin
  var
    j: integer;
    temp, y: real;
    min1, max1, min2, max2: real;
  begin
    datavector;
    mands(x,y);
    min1: = x;
    max1: = x;
    min2: = statistic(x,y);
    max2: = statistic(x,y);
    for j: = 1 to scale test do begin
      datavector;
      mands(x,y);
      if min1 > x then min1: = x;
      if max1 < x then max1: = x;
      temp: = statistic(x,y);
      if min2 > temp then min2: = temp;
      if max2 < temp then max2: = temp;
    end; \( (j \) for)
    mid1: = (max1 + min1)/2;
    scale1: = (max1 - min1)/2;
    mid2: = (max2 + min2)/2;
    scale2: = (max2 - min2)/2;
  end; \{procedure scale\}

procedure tally(var mean, sd: real);
\{ Counts the number of times the simulated statistic falls in each interval \([-h,+h)\) and does the same for the sample mean. The distribution of the sample mean is known, which gives a check on the accuracy of the simulation. If more than 32 blocks of 1000 are desired, the sum variable must be changed from integer to real to avoid overflow. \}
begin
  j1, j2: integer;
  c1, c2: real;
  begin
    c1: = mean;
    c2: = (c1-mid1)/scale1;
    if c1 > max1 then tally1[j2] := tally1[j2] + 1
    else if c1 < max1 then tally1[j2] := tally1[j2] + 1
    else (the first two ifs are to avoid integer overflow)
    begin
      j1: = round((c1+1)/h);
      case j1 of
        lolestint...-1: tally1[0] := tally1[0] + 1;
        0..maxj: tally1[j1] := tally1[j1] + 1;
      end; \( (j1 \) of)
maxint++; maxint++; tally1[maxj] = tally1[maxj] + 1;
end (case)
end (third else)
end 

procedure histogram;
(If requested, prints out the histogram outcome of tally)

var s2: real;
j: integer;

begin
writeln(lst, 'HISTOGRAM');
writeln(lst);
if GammaNotZero then begin
if gammapositive then writeln(lst, 'of the simulation of Y=\{(tp-m)/s\}');
else writeln(lst, 'of the simulation of Y=\{(tp-m)/s\}');
with gamma = 0.7, gamma=2.9;
end of if
else
write(lst, 'of the simulation of Y=\{(t-m)/m\}');

writeln(lst, m=sample mean, s sample s.d.');
writeln(lst, with 'NoDataPoints', 'points in the sample');

writeln(lst, Number of points in the simulation');
writeln(lst, 'Number of blocks, blocks of size ', block);
writeln(lst, For each point p listed, the tally is the ');
writeln(lst, number of points in the interval [p-K/2,p+K/2]');
writeln(lst, no listing if tally is zero ');
writeln(lst, Sum is a running subtotal ');
writeln(lst, Scale factors ');
writeln(lst, scale1 = ', scale1);
writeln(lst, mid1 = ', mid1);
writeln(lst, scale2 = ', scale2);
writeln(lst, mid2 = ', mid2);
writeln(lst);
writeln(lst, p Y sum ');
writeln(lst, _________________________________');
s2 := 0;
for j := 0 to maxj do begin
if (tally2[j])<>0 then begin
write(lst, (j*H-L)*scale2+mid2):10:5);
write(lst, 'tally2[j]:5:0);
s2 := s2+tally2[j];
write(lst, ', s2:5:0);
end (of if)
end (of j for)
writeln(lst);
writeln(lst);
end (of procedure histogram)
function truemean(normalp:real):real;
{This function returns the true percentage point of the sample mean
  corresponding to the percentage point normalp of N(0,1).}
var
temp:real;
begin
  temp:=normalp;
  if GammaNotZero then (*non-normal distribution*)
  begin
    temp:=WilsonHilferty(temp,b*nodapoints);
    truemean:=temp/nodapoints;
  end (of if)
else (*normal distribution, b infinite*)
begin
  truemean:=temp/sqrt(NoDataPoints);
end (of else)
end; 
{function truemean}

procedure percentiles;
{Prints the percentiles 5%5%95% of the simulated statistic which
are found by linear interpolation in the histogram of procedure tally.}
var
  i, j, k: integer;
  x, tmean, error, s, NoSimPoints, pN, interpolation: real;
begin
  writeln(list); writeln(list);
  writeln(list, ' PERCENTILES');
  writeln(list);
  writeln(list, ' (linearly interpolated from histogram)');
  writeln(list, ' sample mean, s sample s.d.');
  writeln(list, ' with ', NoDataPoints, ' points in the sample');
  if GammaNotZero then
  begin
    writeln(list, ' from a gamma random variable with ');
    writeln(list, ' gamma = ', gamma:6:4);
    writeln(list, ' b = ', b:8:4);
    writeln(list, ' Kp = ', Kp:6:5);
    writeln(list, ' tp = ', tp:6:5);
    writeln(list, ' This is the Kp and tp for the ', T:5:1,
      ' year flood.');
    writeln(list);
    if gammapositive then
      writeln(list, ' Y=((tp-s)/s)');
    else
      writeln(list, ' Y=((tp-m)/s)');
  end (of if)
else
begin
  writeln(list, ' from a normal N(0,1) random variable with ');
  writeln(list, ' t = ', tp:8:5);
  writeln(list, ' This is the t for the ', T:5:1, ' year flood.');
  writeln(list, ' Y=(t-m)/s');
end (of else)
write(list);
  error in mean = m - true sample mean';
writeln(list);
write(list, ' Number of points in the simulation:');
writeln(list, ' NoBlocks, blocks of size ', block);
write(list);
write(list);
write(list, ' Percentile Y');
if gammazero then writeln(list, ' Y=Kp') else writeln(list, ' Y=t');
write(list, ' error in mean ');
writeln(list);
write(list, ' NoSimPoints, block*NoBlocks;');
write(list, ');');
for j:=0 to 18 do begin
  percent:=(5+j*5);
  write(list, percent:4:2);
  pN:=(0.05+(0.05*j))*NoSimPoints;
  if tally[0]>pN then interpolation:=(L*scale2+mid2)
else begin
  k := 0;
  s := tally2(0);
  repeat
    k := k + 1;
    s := s + tally2(k);
  until s > pN;
  j2 := k;
  if k = maxj then interpolation := l * scale2 + mid2
  else begin
    repeat
      k := k + 1;
      until tally2(k) > 0;
      j1 := k;
    end; (else with repeat)
  end; (else executed if tally2(0) < pN)
  interpolation := ((pN - (s-tally2(j2)))/tally2[j2])*(j2-j1)*b+(j1*b-L)+b/2;
  if tally1(0) < pN then interpolation := ((-l)*scale1+mid1)
  else begin
    k := 0;
    s := tally1(0);
    repeat
      k := k + 1;
      s := s + tally1(k);
    until s > pN;
    j2 := k;
    if j2 = maxj then interpolation := L*scale1+mid1
    else begin
      repeat
        k := k + 1;
        until tally1[k] > 0;
        j1 := k;
      end; (else with repeat)
    end; (else executed if tally1[0] < pN)
    interpolation := ((pN-(s-tally1[j2])))/tally1[j2]*(j2-j1)*b+(j1*b-L)+b/2;
  end; (else executed if tally1[0] < pN)

  case j of
    (the argument of true mean is the 0.05 + j*(0.05) percentage point for a N(0,1) random variable)
    0: tmean := true mean(-1.64485); (.05)
    1: tmean := true mean(-1.28155); (.10)
    2: tmean := true mean(-0.94162); (.15)
    3: tmean := true mean(-0.67440); (.20)
    4: tmean := true mean(-0.38532); (.25)
    5: tmean := true mean(-0.10860); (.30)
    6: tmean := true mean(0.12602); (.35)
    7: tmean := true mean(0.25336); (.40)
    8: tmean := true mean(0.38532); (.45)
    9: tmean := true mean(0.50000); (.50)
  end; (case)

  error := interpolation - tmean;
  writeln(list,' ',error:7:3);
end; (if for)
write(list);
write(list);
write(list);
write(list);
end; (of procedure percentiles)
procedure floodpercentiles;
  (Uses the global variable pfactor[0..18] and prints a table
  of 10 raised to the power (inputmean + inputs*pfactor[j]).)
  var
     j:integer;
     inputm, inputs:real;

begin
  writeln;
  writeln('Input estimates for mean and s.d.');
  writeln('mean = ?');
  read(inputs);
  writeln;
  writeln('standard deviation = ?');
  read(inputs);
  writeln;
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);

  FLOOD PERCENTILES;
  writeln(lst, '10 to the power m=s*Yp');
  writeln(lst, 'Yp the percentile of Y');
  writeln(lst, 'm = ', inputs, 8:5);
  writeln(lst, 's = ', inputs, 8:5);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst, 'Flood');
  writeln(lst, 'Percentile

  for j:=0 to 18 do begin
    writeln(lst, 0.05+j*(0.05):4:2, '
    writeln(lst, exp(ln(10)*(inputs*pfactor[j]*inputs)):0);
  end;  (j for)
end;  (procedure floodpercentiles)

begin (program confidence)
  writeln('input the number of data vectors');
  read(nodatapoints);
  writeln;
  writeln('input skew coefficient gamma');
  read(gamma);
  writeln;
  if abs(gamma)<0.05 then
    GammaNotZero:=false
  else
    GammaNotZero:=true;
    begin
      if (gamma>0) then gammapositive:=true else gammapositive:=false;
      b:=4/sqrt(gamma);
      GammaNotZero:=true;
    end;  (of else)
  writeln('Computation will be for the T-year flood. Input T.');
  read(T);
  writeln;
  tp:=inverseF(p);  (F N(0,1) d.f.)
  if GammaNotZero then
    begin
      if gammapositive then tp:=WilsonHilferty(tp,b)
      else tp:=WilsonHilferty(-tp,b);
      if gammapositive then Kp:=(tp-b)/sqrt(b)
      else Kp:=(tp-b)/sqrt(b);
    end;  (of gammapositive)
  scale:=scale1,scale2,mid1,mid2;
  writeln('How many blocks (1-30) do you want to do?');
  read(Nblocks);
end.
writein;
  for k:=0 to maxj do begin
    tally[k] := 0;
    tally2[k] := 0;
  end; //of for
randomize;
for k:=1 to NoBlocks do begin
  writeln(' block #',k:2);
  for j:=1 to Block do begin
    datas(x,y);
    tally(x,y);
  end; //of j for
end; //of k for
percentiles;
writeln;
writeln('Do you want to print the histogram(y/n)?');
read(ans1);
if (ans1 = 'y') then histogram;
writeln;
writeln('Do you want percentiles for specific data(y/n)?');
read(ans2);
if (ans2 = 'y') then floodpercentiles;
end. //of confidence