Computing Confidence Intervals for Floods II

Robert Whitley (1) and T.V. Hromadka II (2)

(1) Professor of Mathematics, Department of Mathematics, University of California at Irvine, Irvine, CA, 92717, USA
(2) Research Associate, Department of Civil Engineering, Princeton University, Princeton, N.J., 08544, USA

I. ABSTRACT

A computer program for simulating the sampling distribution for the size of the T-year flood is described in [1]. An approximate formula due to Stedinger [2] allows a quick computation of the percentiles of a random variable from which the computation of an approximate sampling distribution for the T-year flood is easy. The program discussed below computes these approximate percentiles for 5% (5%) 95%.

II. KEY WORDS

T-year Flood, Confidence Interval, Sampling Distribution, Log Pearson III, Distribution

III. INTRODUCTION

Obtaining a confidence interval and an empirical sampling distribution for the T-year flood by means of a simulation has been described in [1]. Stedinger derived an approximate expression for these confidence intervals in [2] using a variance formula due to Kite [3]. In [4] it was shown, by means of comparison with simulations, that this formula is actually accurate for determining the entire sampling distribution of a random variable, which was there called the "non-central gamma distribution", from which the sampling distribution for the T-year flood is derived. In order to do this comparison, a program was developed which computes the percentiles 5%(5%)95% by means of Stedinger's formula. This program is discussed below.

IV. MATHEMATICAL DISCUSSION

Stedinger, in [2], uses Kite's formula for the variance of the sample estimate $\hat{y}_p$ of the p-th percentile of a Pearson III distribution with known skew coefficient and thereby derives an approximate confidence interval for the actual percentile y_p; see his equations (20) and (21). His success in using this formula to construct correct confidence intervals suggests that his formula might even be accurate for determining the q-th percentile Γ_q for the non-central gamma distribution, and this was shown to be so in [4].

The logarithm X of the yearly peak discharge is assumed [5] to have a Pearson type III distribution with density function

$$f(x)=(1/|a|\Gamma(b))[(x-c)/a]^{b-1}\exp-[(x-c)/a] \qquad (1)$$

where, in the case of positive a, the density is given by (1) for x>c and is zero for x<c, while in the case of negative a the density is given by (1) for x<c and is zero for x>c. Computing the mean μ, standard deviation σ, and skew γ from (1) shows that (see, for example [5])

$$\begin{aligned}\sigma^2&=a^2 b\\ \gamma^2&=4/b,\\ \mu&=c+ab\end{aligned} \qquad (2)$$

where a has the same sign as γ.

When restated in terms of the percentiles of the non-central gamma distribution, Stedinger's formula is:

$$\Gamma_q=K_p+\lambda(\zeta_q(p)-z_p) \qquad (3)$$

The index p is related to the T-year flood by p=1-1/T, and the constant z_p is the p-th percentile for a normal N(0,1) distribution. The constant K_p is given by

$$\begin{aligned}K_p&=(t_p-b)/\sqrt{b} \text{ for } a>0,\\ K_p&=(b-t_{1-p})/\sqrt{b} \text{ for } a<0.\end{aligned} \qquad (4)$$

in which t_p and t_{1-p} are the 100p-th and 100(1-p)-th percentiles of a gamma distribution with scale parameter 1 and shape parameter b, and which can be obtained by applying the Wilson-Hilferty transformation [6] to either z_p or z_{1-p}. The factor λ is a positive number given by Kite's variance formula

0266-9463/86/030152 - 08 $2.00

$$\lambda^2=[1+\gamma K_p+(.5)(1+.75\gamma^2)K_p^2]/[1+.5z_p^2] \qquad (5)$$

To use formula (3) we now need to find $\zeta_q(p)$, where $\zeta_q(p)\sqrt{m}$ is the q-th percentile for the non-central t-distribution with non-centrality parameter $\delta=z_p\sqrt{m}$, which was discussed briefly in [1]. This non-central t-distribution is well-known and its q-th percentile can be found by means of a double numerical integration.

V. PROGRAMMING DISCUSSION

To evaluate formula (3) the gamma percentile t_p or t_{1-p} must be found. The functions "NormalF", "InverseF", and "Wilson-Hilferty" are the same as in the program in [1], and the percentile is computed the same way it is computed in that program.

The density function for the non-central t-distribution with ν degrees of freedom and non-centrality parameter δ, is [7]:

$$C\exp(-\nu\delta^2/2(\nu+t^2))[\nu/\nu+t^2]^{(\nu+1)/2}Hh_\nu(-\delta t/\sqrt{(\nu+t^2)}) \qquad (6)$$

where the normalizing constant C, see the function "constant" in the program listing below, is choosen so as to have area one under the density curve.

The function Hh_ν which occurs in the density is defined by

$$Hh_\nu(y)=(1/\nu!)\int_0^\infty t^\nu\exp(-(t+y)^2/2)dt \qquad (7)$$

An integration by parts establishes a recursion formula for the function Hh_n in terms of Hh_{n-1} and Hh_{n-2}. In the program we begin with Hh_0, to which we apply Hasting's approximation for the error function [8, page 299], and Hh_1, which can be integrated in terms of Hh_0, and use this recursion to compute Hh_ν. See the functions "erfc" and "Hh" in the program listing.

Once Hh_ν is computed it is easy to evaluate the density which is done in the function "density". Then the distribution function F can be evaluated at any point x by a numerical integration. The limits for this integration are found in "limits" and the integration is done in functions "sumit" and "Simpson".

The program computes nineteen percentiles and so solves the equation

$$F(x)=p \qquad (8)$$

for nineteen values of p, where each evaluation of the density function F involves the integration discussed above. Consequently Newton's method was used because of its rapid convergence to the solution of (8). In order for Newton's method to converge at all, a good starting value is needed, which is why Newton's method was not used in the function "InverseF". To get these starting values, the procedure "probint" begins with the non-centrality value δ, which is a good guess for the 50-th percentile, and then proceeds down to the 5-th percentile and up to the 95-th percentile by using the previously computed percentile as a starting guess for the adjacent percentile.

The program requires the number of data points and the skew coefficient as input. The output is a table of the approximate percentiles 5%(5%)95% of the non-central gamma distribution. The percentiles given in the sample output below can be compared with the simulated percentiles given in the sample output in [1]. A more extensive comparison is given in [4].

The program "Stedinger", described above and listed below, was written in Turbo-Pascal, with 8087 support, on the IBM PC. A run like the sample below takes about four minutes with the 8087 chip. Recall that the simulation of these values takes an order of magnitude longer [1].

VI. REFERENCES

1. Whitley, R. and Hromadka II, T., Computing confidence limits for floods I, Microsoftware for Engineers.
2. Stedinger, J. R., Confidence intervals for design events, J. of Hydraulic Engineering 109(1983)13-27.
3. Kite, G.W., Confidence limits for design events, Water Res. Research 11(1975)48-53.
4. Whitley,R. and Hromadka II, T., Estimating 100 year flood confidence intervals, submitted to Advances in Water Research, 1986.
5. Hall, M. J., Urban Hydrology, Elsevier, London, 1984.
6. Wilson, E.B. and Hilferty, M. M., The distribution of chi-square, Proc. Nat. Acad. Sci., U.S.A. 17(1931)684-8.
7. Resnikoff, G.J. and Lieberman, G.J., Tables of the Non-Central t-Distribution, Standford Univ. Press, Calif., 1957.
8. Abramowitz, M. and Stegun, I., Handbook of Mathematical Functions, National Bureau of Standards, Washington,D.C., 1965.

VII. SAMPLE OUTPUT

```
      Stedinger's approximate percentiles
           for the 100.0 year flood

      number of data points =   30
      skew = 0.25

      Kp =  2.50907
      tp = 84.07257
      Zp =  2.32635
      lambda =   1.15251

      Percentile = Kp+lambda*(zeta-Zp)

 Percent    Percentile         Zeta
-----------------------------------------
   5         1.94816         1.83966
  10         2.06381         1.94001
  15         2.14608         2.01139
  20         2.21409         2.07040
  25         2.27440         2.12273
  30         2.33019         2.17114
  35         2.38332         2.21723
  40         2.43506         2.26213
  45         2.48640         2.30668
  50         2.53821         2.35163
  55         2.59133         2.39772
  60         2.64671         2.44577
  65         2.70549         2.49677
  70         2.76920         2.55206
  75         2.84008         2.61356
  80         2.92173         2.68440
  85         3.02071         2.77028
  90         3.15145         2.88372
  95         3.35911         3.06390
```

VIII. PROGRAM LISTING

```
Program stedinger;
{Ted Hromadka, Williamson and Schmid, 17782 Sky Park Blvd., Irvine, CA 92714
 Robert Whitley, Dept. of Math., Univ. of Ca., Irvine, CA 92717
 1986
}
 {This program computes percentiles for the "non-central gamma distribution",
  see R. Whitley and T. Hromadka II, Estimating 100 Year Flood Confidence
  Intervals, to appear, by using Stedinger's approximate formula, for which
  see Confidence Intervals for Design Events, Jery Stedinger, J. of Hydraulic
  Engineering 109(1981)13-27. Input is the number of data points and the skew
  coefficient. Output is the approximate percentile points for 5%(5%)95%.
 }

    var
      nodatapoints:integer;     {no of data points in sample}
      gamma:real;               {skew coefficient}
      b:real;                   {b=4/sqrt(gamma)}
      gammanotzero:boolean;     {true if abs(gamma)>.01, else gamma =0}
      gammapositive:boolean;    {true if gamma >.01}
      T:real;                   {the T year flood}
      tp:real;                  {F(tp)=1-1/T; F the normal d.f.
                                 for gamma = 0, else F the gamma
                                 d.f. with shape parameter b.}
      Kp:real;                  {the point estimate for the
                                 design value is mean+s.d.*Kp}
      Zp:real;                  {(1-1/T)-th percentile for N(0,1), corresponding
                                  to the T year flood}
      lambda:real;              {Stedinger's fudge factor to convert
                                 non-central t percentiles to
                                 Pearson III percentiles.}
      p:array[1..19] of real;   {p[j] is the 5*j-th percentile of the
                                 noncentral t-distribution with nocentrality
                                 parameter d=sqrt(nodatapoints)*Zp}
      zeta:real;                {zeta = p[j]/sqrt(nodatapoints)}
      temp:real;
      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) random variable.
 }
   const
     p = 0.2316419;
     b1 =0.319381530;
     b2 = -0.356563782;
     b3 = 1.781477937;
     b4 = -1.821255978;
     b5 = 1.330274429;

   var
     temp,temp2,t: real;

  begin {function NormalF}
    t:=1/(1+p*x);
    temp:=b4+b5*t;
    temp:=b3+temp*t;
    temp:=b2+temp*t;
    temp:=b1+temp*t;
    temp:=temp*t;
    temp2:=exp(-x*x/2)/sqrt(2*pi);
    NormalF:=1-temp2*temp;
  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;

     InverseF:=(x1+x2)/2;
  end;{function InverseF}

   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.I,page 400, and see Mathur, A Note on Wilson-Hilferty
           Transformation of Chi-squared, Bull. of the Calcutta Stat.
           Assoc.10(1961)103-105 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/(9*shape))+t*sqrt(1/(9*shape));
           temp:=exp(3*ln(temp));
           WilsonHilferty:=shape*temp;
        end;{function WilsonHilferty}

procedure probint(n:integer;d:real);
{probint computes the j*5 percentiles p[j], an array global to the
 program stedinger, j=1..19, for a noncentral, t-distribution with
 n degrees of freedom,in this application n=nodatapoints-1, and
 noncentrality parameter d = sqrt(n+1)*Zp in this application.
}

   var
   L:integer;                   {lower bound for the limit
                                 of integration}
   U:integer;                   {Upper bound for the limit
                                 of integration}
  c:real;                     {normalizing constant for the
                               density function}
  pt:real;                    {percentile: want F(x)=pt}
  initialguess:real;          {for Newton}
  temp:real;
  j:integer;

  function constant(n:integer):real;         {*****nested in probint}
  {computes the normalizing constant c for the noncentral t
  }
     var
      j,k:integer;
      c:real;
     begin
     k:=n div 2;
     if odd(n) then
       begin
       c:=exp(k*ln(2))*sqrt(n)/pi;
         for j:=1 to k do begin
          c:=c*j;
         end;{for}
       constant:=c;
       end{if then}
     else {even n}
       begin
       c:=sqrt(k/pi);
         for j:=1 to k do begin
          c:=c*(2*j-1);
         end;{for}
       constant:=c;
     end;{else}
     end;{function constant}

 function erfc(x:real):real;                        {*****nested in probint}
{See Abramowitz and Stegun, page 299, for Hasting's erf approximation.
 This is for x positive.
}
 var
    a:array[Ø..5] of real;
    t,temp:real;
    j:integer;
 begin
    a[Ø]:=Ø.3275911;
    t:=1/(1+a[Ø]*x);
    a[1]:=Ø.254829592;
    a[2]:=-Ø.284496736;
    a[3]:=1.421413741;
    a[4]:=-1.453152Ø27;
    a[5]:=1.Ø614Ø5429;
    temp:=a[5];
    for j:= 5 downto 2 do begin
       temp:=t*temp+a[j-1];
    end; {for}
    erfc:=t*temp*exp(-x*x)
end;{function erfc}

function Hh(n:integer;x:real):real;                 {*****nested in probint}
{See Abramowitz and Stegun, page 691 and page 3ØØ.
 Using Hasting's erf approximation for Hh(Ø,x)
 and then using recursion to get Hh(n,x) is done in both
 G.J. Resnikoff and G. J Lieberman, Table of the Non-central
 t-Distribution, Standford. U. Press, 1957, and M.O.Locks,
 M. J. Alexander, and B. J. Byars, New Tables of the Noncentral
 t-Distribution, Aerospace Research Laboratories Report ARL 63-19,
 Wright-Patterson Airforce Base, Ohio, 1973
 }
   var
     hØ,h1,h2:real;
      j:integer;
   begin
   if (x>Ø) then
     begin
```

```
      hØ:=sqrt(pi/2)*erfc(x/sqrt(2));{using Hasting's for hØ gives an error}
      end                            {on the order of 1Ø to the -7. This    }
    else                             {can be the size of Hh itself for      }
      begin                          {modest size x,e.g.n=2, x=5.           }
      hØ:=sqrt(pi/2)*(2-erfc(-x/sqrt(2)));
      end; {else}
      h1:=exp(-x*x/2)-x*hØ;
      for j:=2 to n do begin
        h2:=(1/j)*(hØ-x*h1);
        hØ:=h1;
        h1:=h2;
      end;{for}
      Hh:=h2;
    end;{function Hh}

function density(x,d,c:real;n:integer):real;      {*****nested in probint}
{This is the density function for the noncentral t-distribution.
 Abramowitz and Stegun have the wrong normalizing constant on
 page 949; see function constant for the correct value which is given
 in, e.g., Tables of the Noncentral t-distribution, G.J. Resnikoff and
 G. J. Lieberman, Standford U. Press, 1957. For efficiency, this
 normalizing constant is passed as c from the main calling program.
 The constant d is the noncentrality parameter delta, n is the number
 of degrees of freedom, and x is the point at which the density is
 being evaluated.
}
    var
      temp:real;
    begin
    temp:=c*exp(((n+1)/2)*ln(n/(n+sqr(x))));
    temp:=temp*exp(-(n*sqr(d))/(2*(n+sqr(x))));
    density:=temp*Hh(n,(-(d*x)/sqrt(n+sqr(x))));
    end;{function density}

procedure limits(var L,U:integer);                 {*****nested in probint}
{Limits establishes a lower value L and the upper value U so that
 the integral from L to U is approximately one.
}
    const
      small=1ØE-ØØ6;
    begin
      L:=round(d); {approximate expectation of noncentral t is d=delta}
      repeat
        L:=L-1;
      until (density(L,d,c,n)<small);
      U:=round(d);
      repeat
        U:=U+1;
      until (density(U,d,c,n)<small);
    end;{procedure limits}


    function f(x:real):real;
      begin
        f:=density(x,d,c,n);
      end; {function f}

    function sumit(c,h:real;m:integer):real;       {*****nexted in probint}
    {converts density(x,d,c,n) to f(x) for use by Simpson integration
    }
      var
        tpoint,tsum:real;
        j:integer;
      begin
        tsum:=Ø;
        tpoint:=c;
        for j:=Ø to m do begin
          tsum:=tsum+f(tpoint);
          tpoint:=tpoint+h;
        end;{for}
        sumit:=tsum;
      end;{function sumit}
```

```
  function simpson(a,b,small:real):real;        {*****nexted in probint}
  {Simpson's rule integration of f(x) from a to b. Stops when doubling
   the number of integration points changes the integral by less than
   the value "small".
  }
    var
      h,I1,I2,oddsum,evensum,sum:real;
      n:integer;
    begin
      n:=4;
      h:=(b-a)/(2*n);
      evensum:=sumit(a+2*h,2*h,n-2);
      oddsum:=sumit(a+h,2*h,n-1);
      sum:=oddsum+evensum;
      I2:=f(a)+2*evensum+4*oddsum+f(b);
      I2:=(h/3)*I2;
      repeat
        I1:=I2;
        n:=2*n;
        h:=h/2;
        evensum:=sum;
        oddsum:=sumit(a+h,2*h,n-1);
        sum:=oddsum+evensum;
        I2:=f(a)+2*evensum+4*oddsum+f(b);
        I2:=(h/3)*I2;
      until (abs(I2-I1)<small) or (n>5000);
      simpson:=I2;
      end;{function simpson}

  function Newton(initialguess, pt:real):real;   {*****nested in probint}
  {Uses Newton's method to iterate to a point z with
   probability(noncentralt<z)=pt.
  }
    var
      F:real;                  {F= dist.fctn for noncentral t at pt}
      x0:real;                 {last iterate}
      x1:real;                 {new iterate}
      x2:real;                 {newest iterate}
     begin
      x0:=initialguess;
      repeat
        if (x0<(L+U)/2) then F:=simpson(L,x0,10E-7)
        else F:=1-simpson(x0,U,10E-7);
        x1:=x0-((F-pt)/density(x0,d,c,n));
        x2:=x0;
        x0:=x1;
      until (abs(x1-x2)<10E-5);
      Newton:=x1;
     end;{function Newton}

begin {procedure probint}
  c:=constant(n);
  limits(L,U);
  writeln('Percentile now calculated:');
  initialguess:=d;
    for j:=10 downto 1 do begin
      pt:=0.05*j;
      p[j]:=Newton(initialguess,pt);
      initialguess:=p[j];
      write('    ',5*j);
    end;{for}
    initialguess:=d;
    for j:=11 to 19 do begin
      pt:=0.05*j;
      p[j]:=Newton(initialguess,pt);
      initialguess:=p[j];
      write('    ',5*j);
    end;{for}
end; {procedure probint}


begin {main program stedinger}
    writeln('input the number of data points');
    read(nodatapoints);
```

```
     writeln;
     writeln('input skew coefficient gamma');
     read(gamma);
     writeln;
       if abs(gamma)<0.01 then
          GammaNotZero:=false
        else
            begin
            if (gamma>0) then gammapositive:=true else gammapositive:=false;
            b:=4/sqr(gamma);
            GammaNotZero:=true;
        end; {of else}
     writeln('Computation will be for the T-year flood. Input T.');
      read(T);
      writeln;
      tp:=inverseF(1-1/T);    {F N(0,1) d.f.}
      Zp:=tp;
      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 gammanotzero then}


      lambda:=sqr(Kp)*(1+(3*sqr(gamma)/4))/2;
      lambda:=lambda+gamma*Kp+1;
      lambda:=lambda/(1+(sqr(Zp)/2));
      lambda:=sqrt(lambda);

      probint(nodatapoints-1,sqrt(nodatapoints)*Zp);  {this computes p[1..19]}


      writeln(lst,'      Stedinger's approximate percentiles');
      writeln(lst,'           for the ',T:5:1,' year flood');
      writeln(lst);
      writeln(lst,'      number of data points = ',nodatapoints:4);
      writeln(lst,'      skew = ',gamma:4:2);
      writeln(lst);
      writeln(lst,'      Kp = ',Kp:8:5);
      writeln(lst,'      tp = ',tp:8:5);
      writeln(lst,'      Zp = ',Zp:8:5);
      writeln(lst,'      lambda = ',lambda:8:5);
      writeln(lst);
      writeln(lst,'      Percentile = Kp+lambda*(zeta-Zp)');
      writeln(lst);
      writeln(lst,'Percent   Percentile       Zeta');
      writeln(lst,'_______________________________________________');
      write(lst,' ');
      for j:=1 to 19 do begin
         zeta:=p[j]/sqrt(nodatapoints);
         if gammanotzero then temp:=Kp+lambda*(zeta-Zp) else temp:=zeta;
         writeln(lst,j*5,'        ',temp:8:5,'        ',zeta:8:5);
         end;{for}
end.{program stedinger}
```