Algorithm AS 155

The Distribution of a Linear Combination of $\chi^2$ Random Variables

BY ROBERT B. DAVIES

Applied Maths Division, D.S.I.R., Wellington, New Zealand

Keywords: CHARACTERISTIC FUNCTION; CHI-SQUARED VARIABLE; LINEAR COMBINATION; NORMAL VARIABLE; NUMERICAL INVERSION; QUADRATIC FORM; RATIO OF QUADRATIC FORMS

LANGUAGE

Algol 60

DESCRIPTION AND PURPOSE

Let

$$Q = \sum_{j=1}^{r} \lambda_j X_j + \sigma X_0,$$

where $X_j$ are independent random variables, $X_j$ having a non-central $\chi^2$ distribution with $n_j$ degrees of freedom and non-centrality parameter $\delta_j^2$ for $j = 1, \ldots, r$ and $X_0$ having a standard normal distribution. Then the purpose of this algorithm is to calculate

$$\Pr(Q < c).$$

The algorithm is based on the method of Davis (1973) involving the numerical inversion of the characteristic function. It will yield results for most linear combinations that are likely to be encountered in practice but is more satisfactory if the sum (1) is not dominated by terms involving a total of less than four degrees of freedom. The accuracy is set by the user, a maximum error of 0.0001 being an appropriate value.

Any quadratic form in independent normal variables can be reduced to the form (1) and so this algorithm can be used to calculate the distribution of such a quadratic form. Since the $\lambda_j$ need not all be positive the quadratic form need not be positive definite. In particular, the algorithm can be used to find the distribution of the ratio of two quadratic forms.

METHOD

The basic formula is formula (9) in Davies (1973) with the integration error being bounded as in that paper. Not discussed is the truncation error

$$\sum_{k=K+1}^{\infty} \frac{\text{Im}[(\phi((k+1/2)\Delta) e^{-ik+1/2)\Delta})]\{(\pi(k+1/2))}{(\pi(k+1/2))},

where $\phi$ is the characteristic function of $Q$ given in Section 4 of Davies (1973) and $\Delta$ is the integration interval. If $|\phi(u)| \leq B(u)$ and $B(u)$ is a monotonically decreasing function of $u$ (for $u \geq U$) then (3) is bounded by

$$\sum_{k=K+1}^{\infty} B((k+1/2)\Delta)/{(\pi(k+1/2))} \leq \int_{u=U}^{\infty} B(u)/(\pi u) du,$

where $U = (K+1/2)\Delta$.

Writing

$$N(u) = \exp\left\{-2u^2 \sum_{j=1}^{r} \lambda_j^2 8u^2/(1 + 4u^2 \lambda_j^2)\right\}$$

Writing
three possible forms for $B(u)$ are

$$N(u) \exp \left( -U^2 \frac{\sigma^2}{2} \right) \prod_{1 \leq j \leq n} \left( 1 + 4U^2 \lambda_j^2 \right)^{-n_j/4} \prod_{1 \leq j \leq n} \left( 4U^2 \lambda_j^2 \right)^{-n_j/4},$$

where product (i) is over all values of $j$ with $|\lambda_j| \leq 1$ and product (ii) is over values of $j$ with $|\lambda_j| > 1$;

$$N(U) \exp \left( -u^2 \frac{\sigma^2}{2} \right) \prod_{1 \leq j \leq n} \left( 1 + 4U^2 \lambda_j^2 \right)^{-n_j/4}$$

and

$$N(U) \left( \prod_{1 \leq j \leq n} \left( 1 + 4U^2 \lambda_j^2 \right)^{n_j} \exp \left( 2U^2 \sigma^2 \right) - 1 \right)^{-1/4}$$

$$-1/2S(U) \exp \left( -U^2 \frac{\sigma^2}{2} \right) \prod_{1 \leq j \leq n} \left( 1 + 4U^2 \lambda_j^2 \right)^{-n_j/4} \prod_{1 \leq j \leq n} \left( 4U^2 \lambda_j^2 \right)^{-n_j/4}$$

provided

$$\prod_{1 \leq j \leq n} \left( 1 + 4U^2 \lambda_j^2 \right)^{n_j} \exp \left( 2U^2 \sigma^2 \right) \geq e$$

leading to bounds on the truncation error

$$\{2/(n\pi)\} N(U) \exp \left( -U^2 \frac{\sigma^2}{2} \right) \prod_{1 \leq j \leq n} \left( 1 + 4U^2 \lambda_j^2 \right)^{-n_j/4} \prod_{1 \leq j \leq n} \left( 4U^2 \lambda_j^2 \right)^{-n_j/4}$$

where $s = \sum_1^n n_j$,

$$\{1/(n\pi U^2 \sigma^2)\} N(U) \exp \left( -U^2 \frac{\sigma^2}{2} \right) \prod_{1 \leq j \leq n} \left( 1 + 4U^2 \lambda_j^2 \right)^{-n_j/4}$$

and

$$\{2/(\pi)\} N(U) \exp \left( -U^2 \frac{\sigma^2}{2} \right) \prod_{1 \leq j \leq n} \left( 1 + 4U^2 \lambda_j^2 \right)^{-n_j/4}$$

provided (5) is satisfied. The algorithm uses the minimum of (6), (7) and (8) as the truncation bound. Note that the bound (8) would need to be modified if the program was extended to allow non-integer values of $n_j$.

The truncation point, $U$, may sometimes be reduced by introducing a convergence factor. Suppose that the characteristic function $\phi(u)$ is multiplied by

$$\exp \left( -\tau^2 u^2 / 2 \right)$$

corresponding to the addition of another normal variable $\tau Z$ to the sum (1), $Z$ being standard normal. Then the error introduced

$$\Pr(Q + \tau Z < c) - \Pr(Q < c) = \int_{-\infty}^{\infty} e^{-iuc} \{ \exp \left( -\tau^2 u^2 / 2 \right) - 1 \} \phi(u)/(2\pi u) du.$$

Suppose that $c > 0$, a corresponding formula being available when $c < 0$. Then integrating along $u = v + iv$ for $-\infty < v < 0$ and $u = v - iv$ for $0 < v < \infty$ we obtain

$$\int_0^\infty \{ \Pr(Q + \tau Z < c) - \Pr(Q < c) \} \exp \left\{ \frac{\tau}{2} \sum_{1 \leq j \leq n} (1 - 4v\lambda_j) \lambda_j (1 - 4v\lambda_j + 8v^2 \lambda_j^2) \right\}$$

$$\times \prod_{1 \leq j \leq n} \left( 1 - 4v\lambda_j + 8v^2 \lambda_j^2 \right)^{-n_j/4} \exp \left( \frac{\tau^2}{2u^2} \right)$$

the product (i) and the sum (ii) involving only those values of $j$ for which $\lambda_j > 0$; those corresponding to large values of $\lambda_j$ being in the product (i) and the others in the sum (ii) with the
exact point at which the split is made being adjusted for the optimum bound. Evaluating the integral yields the bound

\[ (r^2/\pi) \sum_k \prod_i 2^{(n_i + \delta_i^2)/4}/(c - \sum_j \lambda_j (n_j + \delta_j^2))^2. \] (10)

For large values of \( c \) (10) will tend to be small and hence a useful factor will be able to be introduced. However, (10) can also be used in a different way. We express

\[ \text{pr}(Q < c) = \{\text{pr}(Q < c) - \text{pr}(Q + rZ < c)\} + \text{pr}(Q + rZ < c). \] (11)

The first term on the right-hand side of (11) can be integrated numerically with integration error, according to equation (7) of Davies (1973), being given by

\[ \sum_{n=1}^{\infty} (-1)^n \{\text{pr}(Q + rZ < c - 2\pi n/\Delta) - \text{pr}(Q < c - 2\pi n/\Delta) \]
\[ + \text{pr}(Q + rZ < c + 2\pi n/\Delta) - \text{pr}(Q < c + 2\pi n/\Delta)\}. \] (12)

In (9), after replacing \( v \) by \( v - iv \) and summing

\[ \sum_{n=1}^{\infty} (-1)^n \{\text{pr}(Q + rZ < c + 2\pi n/\Delta) - \text{pr}(Q < c + 2\pi n/\Delta)\} \]
we find the term \( \exp \{ -i(v - iv)c \} \) must be replaced by

\[ \exp \{ -i(v - iv)(c + 2\pi n/\Delta)\}/[1 - \exp(-w + iw)], \]
where \( w = 2\pi v/\Delta \). But \( 1/[1 - \exp(-w + iw)] \ll 1 \) and so (10) applied to \( c + 2\pi \Delta \) and its analogue for negative constant to \( c - 2\pi \Delta \) can be used to bound the integration error (12). The truncation error can be bounded as before. The second term in (11) may be evaluated by numerical integration or possibly further split up. This completes the description of the error bounds. The actual way they are used is best described by the algorithm itself.

The formula (9) of Davies (1973) used to compute (1) can be expressed as

\[ 1/2 - \sum_{k=0}^{K} \exp \left\{ -2\pi^2 \sum_{j=1}^{r} \lambda_j^2 \delta_j^2/(1 + 4u^2 \lambda_j^2) - u^2 \sigma^2/2 \right\} \prod_{j=1}^{r} (1 + 4u^2 \lambda_j^2)^{-n/j} \]
\[ \times \sin \left\{ \sum_{j=1}^{r} [n_j \arctan(2u\lambda_j)/(1 + \delta_j^2) - u \lambda_j]/(1 + 4u^2 \lambda_j^2) \right\} \} \{\pi(k + 1/2)\}. \] (13)

where we have written \( u \) for \( (k + 1/2)/\Delta \). For the auxiliary integration in (11) formula (13) must be multiplied by

\[ 1 - \exp(r^2 u^2/2). \]

It is possible that the sum (13) contains terms which are of large magnitude and fluctuating sign or that the argument of the sine function is large. In both cases significant round-off error could accumulate. For this reason (13) is also calculated with the sine term replaced by the sum of the absolute values of the summands of its argument. A fault indication is returned if this sum is excessively large. In practice this does not seem to be a problem.

**Structure**

**real procedure** `gf(lb, nc, n, r, sigma, c, lim, acc, trace, ifault)`

**Formal parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lb</code></td>
<td>Real array [1 : r] input : values of ( \lambda_i )</td>
</tr>
<tr>
<td><code>nc</code></td>
<td>Real array [1 : r] input : values of ( \delta_i^2 )</td>
</tr>
<tr>
<td><code>n</code></td>
<td>Integer array [1 : r] input : degrees of freedom of ( j )th term</td>
</tr>
<tr>
<td><code>r</code></td>
<td>Integer value : number of ( \chi^2 ) terms in sum</td>
</tr>
<tr>
<td><code>sigma</code></td>
<td>Real value : coefficient of normal variable</td>
</tr>
<tr>
<td>Var</td>
<td>Type</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td><code>c</code></td>
<td>Real</td>
</tr>
<tr>
<td><code>lim</code></td>
<td>Integer</td>
</tr>
<tr>
<td><code>acc</code></td>
<td>Real</td>
</tr>
</tbody>
</table>
| `trace`| Real array [1:7] | output: indicate performance of procedure:  

- `trace[1]`: absolute value sum  
- `trace[2]`: total number of integration terms  
- `trace[3]`: number of integrations  
- `trace[4]`: integration interval in main integration  
- `trace[5]`: truncation point in initial integration  
- `trace[7]`: number of cycles to locate integration parameters  

<table>
<thead>
<tr>
<th><code>ifault</code></th>
<th>Integer</th>
<th>output: fault indicator:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ifault = 0</code></td>
<td>no error</td>
<td></td>
</tr>
<tr>
<td><code>ifault = 1</code></td>
<td>requested accuracy could not be obtained</td>
<td></td>
</tr>
<tr>
<td><code>ifault = 2</code></td>
<td>round-off error possibly significant</td>
<td></td>
</tr>
<tr>
<td><code>ifault = 3</code></td>
<td>invalid parameters</td>
<td></td>
</tr>
<tr>
<td><code>ifault = 4</code></td>
<td>unable to locate integration parameters</td>
<td></td>
</tr>
</tbody>
</table>

Realistic values for "lim" range from 1000 if the procedure is to be called repeatedly up to 50000 if it is to be called only occasionally. Suitable values for "acc" range from 0001 to 000005 which should be adequate for most statistical purposes. Meaningful results are returned only if "ifault" is returned as 0 or possibly 2.

To simplify use with compilers that require labels to be declared the positions of such declarations have been noted with comments.

### Restriction

It is supposed that at least one $\chi^2$ term has non-zero degrees of freedom and non-zero $\lambda_j$ or that $\sigma$ is non-zero.

### Precision

As far as possible numerical techniques have been used to enable single precision to provide adequate accuracy with, for example, 32 bit word lengths. However if "ifault = 2" occurs, indicating that round-off error might be significant, or extremely small values of "acc" are being used, then procedure "integrate" and variables "int1", "int2", "ersm1", "ersm2" should be converted to double precision and a double precision version of procedure "ln1" included.

### Related Algorithm

An alternative algorithm, AS 106, which can be adapted to calculate the distribution of (1) provided that all the $\lambda_j$ are positive and $\sigma = 0$ has been published by Sheil and O’Muircheartaigh (1977). In general, their algorithm is very much faster than the one presented here if the total number of degrees of freedom is small with the ratio of the largest $\lambda_j$ to the smallest $\lambda_j$ not large. On the other hand, if the ratio of the largest $\lambda_j$ to the smallest $\lambda_j$ is very large or the total number of degrees of freedom large this algorithm has the advantage particularly if there are also large non-centrality parameters. Of course only this one is applicable if the $\lambda_j$ are of varying sign or $\sigma > 0$; in addition it is more robust against extreme parameter values such as large numbers of degrees of freedom, large non-centrality parameters or large ratios of the $\lambda_j$. 


### Table 1

<table>
<thead>
<tr>
<th>Degrees of freedom</th>
<th>Non-centrality parameter</th>
<th>$\chi^2$ probability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>9965</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1815</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>584</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>68</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>7.84</td>
<td>2268</td>
</tr>
<tr>
<td>3</td>
<td>11.56</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>12.96</td>
<td>16</td>
</tr>
</tbody>
</table>

### Table 2

<table>
<thead>
<tr>
<th>Degrees of freedom</th>
<th>F probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. Den.</td>
<td>0.01</td>
</tr>
<tr>
<td>1</td>
<td>6110</td>
</tr>
<tr>
<td>1</td>
<td>4315</td>
</tr>
<tr>
<td>1</td>
<td>4210</td>
</tr>
<tr>
<td>3</td>
<td>182</td>
</tr>
<tr>
<td>3</td>
<td>182</td>
</tr>
<tr>
<td>5</td>
<td>41</td>
</tr>
</tbody>
</table>

### Table 3

<table>
<thead>
<tr>
<th>Quadratic form</th>
<th>c</th>
<th>Probability</th>
<th>Number of terms</th>
<th>Times (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6, 1; 3, 1; 1, 1</td>
<td>6.1; 3.1; 1</td>
<td>0.0542</td>
<td>744</td>
<td>2532</td>
</tr>
<tr>
<td>6, 2; 3, 2; 1, 2</td>
<td>6.2; 3.2; 1</td>
<td>0.066</td>
<td>74</td>
<td>269</td>
</tr>
<tr>
<td>6, 6; 3, 4; 1, 2</td>
<td>6.6; 3.4; 1</td>
<td>0.027</td>
<td>18</td>
<td>103</td>
</tr>
<tr>
<td>7, 6; 3, 2, 2</td>
<td>7.6; 3.2</td>
<td>0.061</td>
<td>16</td>
<td>77</td>
</tr>
<tr>
<td>7, 1, 6; 3, 1, 2</td>
<td>7.1, 6; 3.1</td>
<td>0.0451</td>
<td>603</td>
<td>1554</td>
</tr>
<tr>
<td>7, 6; 3, 2; 7, 1, 6; 3, 1, 2</td>
<td>7.6; 3.2, 7.1, 6; 3.1</td>
<td>0.0437</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>7, 6; 3, 2</td>
<td>7.6; 3.2</td>
<td>0.0782</td>
<td>10</td>
<td>98</td>
</tr>
<tr>
<td>-7, 1, 6; -3, 1, 2</td>
<td>-7.1, 6; -3.1, 2</td>
<td>0.5221</td>
<td>8</td>
<td>92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quadratic form</th>
<th>c</th>
<th>Probability</th>
<th>Number of terms</th>
<th>Times (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6, 1; 3, 1; 1, 1</td>
<td>6.1; 3.1; 1</td>
<td>0.054</td>
<td>744</td>
<td>22</td>
</tr>
<tr>
<td>6, 2; 3, 2; 1, 2</td>
<td>6.2; 3.2; 1</td>
<td>0.066</td>
<td>74</td>
<td>19</td>
</tr>
<tr>
<td>6, 6; 3, 4; 1, 2</td>
<td>6.6; 3.4; 1</td>
<td>0.027</td>
<td>18</td>
<td>35</td>
</tr>
<tr>
<td>7, 6; 3, 2, 2</td>
<td>7.6; 3.2</td>
<td>0.061</td>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td>7, 1, 6; 3, 1, 2</td>
<td>7.1, 6; 3.1</td>
<td>0.0451</td>
<td>603</td>
<td>22</td>
</tr>
<tr>
<td>7, 6; 3, 2; 7, 1, 6; 3, 1, 2</td>
<td>7.6; 3.2, 7.1, 6; 3.1</td>
<td>0.0437</td>
<td>10</td>
<td>92</td>
</tr>
<tr>
<td>7, 6; 3, 2</td>
<td>7.6; 3.2</td>
<td>0.0782</td>
<td>10</td>
<td>98</td>
</tr>
<tr>
<td>-7, 1, 6; -3, 1, 2</td>
<td>-7.1, 6; -3.1, 2</td>
<td>0.5221</td>
<td>8</td>
<td>92</td>
</tr>
</tbody>
</table>
APPLIED STATISTICS

PERFORMANCE AND TIMING

The number of terms required for the integration is determined approximately by the total number of degrees of freedom and the sum of the non-centrality parameters of the dominant terms in the sum (1) and by the value \( c \), at which the distribution function is evaluated. Hence to give some idea of the performance of the algorithm we have found the number of terms required to calculate the distribution function of a \( \chi^2 \) random variable with various degrees of freedom and non-centrality parameters. In each case, three values of \( c \) have been used, corresponding to distribution function values of 0.01, 0.5 and 0.99. The accuracy has been set to 0.0001. The results are listed in Table 1. To indicate the performance for ratios of quadratic forms, we have also found the number of terms required to calculate various central \( F \) probabilities. In each case \( c = 0, \lambda_1 = 1, \) and \( \lambda_2 \) is set to give the distribution values 0.01, 0.5 and 0.99. Again "acc" is set to 0.0001. The results are listed in Table 2. Of course, the algorithm is not intended for calculating pure \( \chi^2 \) and \( F \) probabilities so the poor performance for \( \chi^2 \) with one degree of freedom or the \( F_{1,1} \) distribution is not very worrying. With "genuine" linear combinations other terms would usually be present in the sum to assist with convergence.

Finally we have tested the algorithm on some of the quadratic forms listed by Imhof (1961). In this case we have given in Table 3 the number of integration terms, the processor time required by this algorithm and the time required by the algorithm adapted from that of Sheil and O'Muircheartaigh. In the table we have specified the quadratic forms by giving, for each \( \chi^2 \) random variable, a set of 2 or 3 numbers being the values of the weight, \( \lambda \), the number of degrees of freedom and, when non-zero, the non-centrality parameter, \( \delta^2 \). The accuracy was again set to 0.0001. The computer used was the Burroughs 6700 belonging to Victoria University of Wellington.

REFERENCES


real procedure qflb(\(q\), \(n\), \(r\), \(\sigma\), \(\lambda\), \(a\), \(b\), \(c\), \(d\), \(e\), \(f\), \(g\), \(h\), \(i\), \(j\), \(k\), \(l\), \(m\), \(n\), \(o\), \(p\), \(q\), \(r\), \(s\), \(t\), \(u\), \(v\), \(w\), \(x\), \(y\), \(z\));


value \(r\), \(\sigma\), \(\lambda\), \(a\), \(b\), \(c\), \(d\), \(e\), \(f\), \(g\), \(h\), \(i\), \(j\), \(k\), \(l\), \(m\), \(n\), \(o\), \(p\), \(q\), \(r\), \(s\), \(t\), \(u\), \(v\), \(w\), \(x\), \(y\), \(z\);

real \(\sigma\), \(\lambda\), \(a\), \(b\), \(c\), \(d\), \(e\), \(f\), \(g\), \(h\), \(i\), \(j\), \(k\), \(l\), \(m\), \(n\), \(o\), \(p\), \(q\), \(r\), \(s\), \(t\), \(u\), \(v\), \(w\), \(x\), \(y\), \(z\);

color comment distribution function of a linear combination of non-central chi-squared random variables;

begin
real \(p\), \(q\), \(\sigma\), \(\lambda\), \(a\), \(b\), \(c\), \(d\), \(e\), \(f\), \(g\), \(h\), \(i\), \(j\), \(k\), \(l\), \(m\), \(n\), \(o\), \(p\), \(q\), \(r\), \(s\), \(t\), \(u\), \(v\), \(w\), \(x\), \(y\), \(z\);

integer \(count\);

procedure counter;

color comment label EXIT;

begin

\(count := count + 1;\)
\(if count > lim then\)
\(begin\)
\(\color{red}{color comment this error exit should almost never occur and could\}
\(\color{red}{be replaced by an error message and stop, on compilers that\}
\(\color{red}{do not handle goto exits from procedures;\}
\(\color{red}{ifault := 4; goto EXIT\}
\color{red}{end\}
\color{red}{and counter;\}
\color{red}{end\}
\color{red}{begin\}
\color{red}{comment this error exit should almost never occur and could\}
\color{red}{be replaced by an error message and stop, on compilers that\}
\color{red}{do not handle goto exits from procedures;\}
\color{red}{ifault := 4; goto EXIT\}
\color{red}{end\}
\color{red}{and counter;\}
\color{red}{end\}
\color{red}{begin\}
\color{red}{comment this error exit should almost never occur and could\}
\color{red}{be replaced by an error message and stop, on compilers that\}
\color{red}{do not handle goto exits from procedures;\}
\color{red}{ifault := 4; goto EXIT\}
\color{red}{end\}
\color{red}{and counter;\}
real procedure ln1(x, first); value x, first;
real x; Boolean first;

comment if first then ln(1 + x) else ln(1 + x) - x;

if abs(x) > 0.1 then
ln1 := if first then ln(1.0 + x) else ln(1.0 + x) - x
else
begin real s, s1, term, y, k;
y := x / (2.0 + x); term := 2.0 * y & 3;
k := 3.0; s := (if first then 2.0 else -x) * y;
y := y & 2;
for s1 := s + term / k while s1 != s do
begin
k := k + 2.0; term := term * y;
s := s1
end;
ln1 := s
end ln1;

procedure order;

comment find order of absolute values of lb;

begin integer j, k; real lj;

comment label L1;

for j := 1 step 1 until r do
begin
lj := abs(lb[j]);
for k := j - 1 step -1 until 1 do
if lj > abs(lb[th[k]]) then th[k + 1] := th[k] else goto L1;
k := 0;
L1:th[k + 1] := j
end;
notert := false
end order;

real procedure errbd(u, cx); value u; real u, cx;

comment find bound on tail probability using mgf. Cutoff point returned to cx;

begin real sum1, lj, ncj, x, y, const; integer j, nj;
counter; const := u * e^gamma;
sum1 := u * const; u := 2.0 * u;
for j := r step -1 until 1 do
begin
nj := n[j]; lj := lb[j];
ncj := nc[j]; x := u * x 1;
y := 1.0 - x; const := const + 1j * (ncj / y + nj) / y;
sum1 := sum1 + ncj * (x / y) & 2 + nj * (x & 2 / y + ln1(-x, false))
end j;
errbd := exp(-0.5 * sum1); cx := const
end errbd;

real procedure cttf(accx, upn); value accx; real accx, upn;

comment find cttf so that P(qf > cttf) < accx if upn > 0,
P(qf < cttf) < accx otherwise;

begin real u1, u2, u, rb, const, c1, c2;
u2 := upn; u1 := 0.0;
c1 := mean; rb := 2.0 * (if u2 > 0.0 then lmax else lmin);
for u := u2 / (1.0 + u2 * rb) while errbd(u, c2) > accx do
begin
u1 := u2; c1 := c2;
u2 := 2.0 * u2
end;
for u := (c1 - mean) / (c2 - mean) while u < 0.0 do
    begin
        u := (u1 + u2) / 2.0;
        if errb1(u / (1.0 + u * xb), const) > accx then
            begin
                u1 := u; cl := const
            end
        else
            begin
                u2 := u; c2 := const
            end
        end:
        eff := c5; upn := u2
    end eff;

real procedure truncation(u, tauq); value u, tauq; real u, tauq;

comment bound integration error due to truncation at u:

begin
    real sum1, sum2, prod1, prod2, prod3, l1, ncj, x, y, err1, err2;
    integer j, nj, s;
    counter: sum1 := prod2 := prod3 := 0, 0;
    s := 0; sum2 := (signq + tauq) * u & 2;
    prod1 := 2, 0 * sum2; u := 2, 0 * u;
    for j := 1 step 1 until r do
        begin
            l1 := lb[j]; ncj := nc[j];
            nj := n[j]; x := (u * l1) & 2;
            sum1 := sum1 + ncj * x / (1.0 + x);
            if x > 1.0 then
                begin
                    sum2 := prod2 + nj * ln(x);
                    prod3 := prod3 + nj * ln(x, true); s := s + nj
                    end
            else
                begin
                    prod1 := prod1 + nj * ln(x, true)
                    end
                end;
    sum1 := 0.5 * sum1; prod2 := prod1 + prod2;
    prod3 := prod1 + prod3; x := exp(-sum1 - 0.25 * prod2) / p1;
    y := exp(-sum1 - 0.25 * prod3) / p1;
    err1 := if s = 0 then 1.0 else x * 2.0 / s;
    err2 := if prod3 > 1.0 else 2.5 * y else 1.0;
    if err2 < err1 then err1 := err2;
    x := 0.5 * sum2; err2 := if x < y then 1.0 else y / x;
    truncation := if err1 < err2 then err1 else err2
    end truncation;

procedure findu(utx, accx); value accx; real utx, accx;

comment find u such that truncation(u) < accx
    and truncation(u / 1.2) > accx;

begin real u, ut;
    ut := utx; u := ut / 4.0;
    if truncation(u, 0) > accx then
        begin
            for u := ut while truncation(u, 0) > accx do
                ut := ut / 4.0
        end
    else
        begin
            ut := u;
            for u := u / 4.0 while truncation(u, 0) < accx do ut := u
        end:
        for u := ut / 2.0, ut / 1.4, ut / 1.2, ut / 1.1 do
            if truncation(u, 0) ≤ accx then ut := u;
    utx := ut
end findu;

procedure integrate(nterm, interv, tauq, main);
    value nterm, interv, tauq, main; integer nterm;
    real interv, tauq; boolean main;
begin real impl, u, sum1, sum2, sum3, x, y, z; integer k, j, n;j
impl := interw / pi;
for k := nterm step -1 until 0 do
begin
u := (k + 0.5) X interw; sum1 := -2.0 X u X c; sum2 := abs(sum1); sum3 := -0.5 X sigeq X u X 2;
for j := r step -1 until 1 do
begin
nj := n[j]; x := 2.0 X lb[j] X u;
y := x X 2; sum3 := sum3 - 0.25 X nj X ln1(y, true);
y := nof[j] X x / (1.0 + y); x := nj X arctan(x) + y;
sum1 := sum1 + x; sum2 := sum2 + abs(z);
sum3 := sum3 - 0.5 X x X y
end; x := impl X exp(sum3) / u;
if main then x := x X (1.0 - exp(-0.5 X tauaq X u X 2));
sum1 := sin(0.5 X sum1) X x; sum2 := 0.5 X sum2 X x;
if abs(sum1) < acc then
begin
int11 := int11 + sum1; ors1 := ors1 + sum2
and end
else
begin
int12 := int12 + sum1; ors2 := ors2 + sum2
and end
end integrate;
real procedure cfe(x); value x; real x;

comment coef of tauq in error when convergence factor of exp(-0.5 X tauq X u X 2) is used when df is evaluated at x;
begin real ax1, ax11, ax12, ax1, sum1, 1j; integer j, k, t;
comment label L;
counter;
if ntest then order;
ax1 := abs(x); ax11 := sign(x);
sum1 := 0.0;
for j := r step -1 until 1 do
begin
t := th[j];
if lb[t] X ax1 > 0.0 then
begin
1j := abs(lb[t]); ax11 := ax1 - 1j X (n[t] + nc[t]);
ax12 := 1j / 1x20;
if ax1 > ax12 then ax1 := ax11 else
begin
if ax1 > ax12 then ax1 := ax12;
sum1 := (ax1 - ax11) / 1j;
for k := j - 1 step -1 until 1 do
sum1 := sum1 + (n[th[k]] + nc[th[k]])
goto L
end
end
end j:
L:if sum1 > 100.0 then
begin
cfe := 1.0; fail := true
and else cfe := 2.0 X (sum1 / 4.0) / (pi X ax1 X 2)
and cfe;
comment ln2 = ln(2.0) / 8.0;
\[ \ln x = 0.05 \text{; } p_i = 3.14159265358979; \]
\begin{verbatim}
begin integer i, nj, nt, ntm;
real accl, alnx, utx, tauq, sd, intv, intvl, x, up, un, d1, d2, lj, ncj;

comment label L1, L2;

for j := 1 step 1 until 7 do trace[j] := 0.0;
ifault := count := 0; intfl := intl2 := orsml := orsml2 := 0.0;
qf := -1.0; acel := accl;
start := true; fail := false;

comment find mean, sd, max and min of 1b, check that parameter values are valid;
accl := sigeq := sigma A \sigma; lmax := lmin := mean := 0.0;
for j := 1 step 1 until r do
begin
nj := ncj[j]; lj := lb[j];
ncj := ncj[j];
if nj < 0 \lor ncj < 0.0 then
begin
ifault := 3; goto EXIT
end;
accl := accl + lj A \sigma \times \ln(1.0 + n(j + 1.0));
mean := mean + lj A \sigma \times (nj + ncj);
if lmax < lj then lmax := lj else
if lmin > lj then lmin := lj
end;
if sd = 0.0 then
begin
qf := if c > 0.0 then 1.0 else 0.0; goto EXIT
end;
if lmin = 0.0 \land lmax = 0.0 \land sigma = 0.0 then
begin
ifault := 3; goto EXIT
end;
accl := sqrt(sd); alnx := if lmax < -lmin then -lmin else lmax;
comment starting values for findu, ctfu,
utx := 16.0 / sd; up := 4.5 / sd;
un := -up;

comment truncation point with no convergence factor;
findu(utx, 0.5 A accl);

comment does convergence factor help?;
if c > 0.0 \land alnx > 0.07 A sd then
begin
tauq := 0.25 A accl / cfe(c);
if fail then fail := false else
if truncation(utx, tauq) < 0.2 A accl then
begin
sigeq := sigeq + tauq; findu(utx, 0.25 A accl);
end;
end;
trace[5] := utx; accl := 0.5 A accl;

comment find 'range' of distribution, quit if outside this;
L1-d1 := ctfu(accl, up) - c;
if d1 < 0.0 then
begin
qf := 1.0; goto EXIT
end;
d1 := c - ctfu(accl, un); if d1 < 0.0 then
begin
qf := 0.0; goto EXIT
end;
\end{verbatim}
comment find integration interval:
intv := 2.0 X pi / (if d1 > d2 then d1 else d2);

comment calculate number of terms required for main and
auxiliary integrations:
nt := utx / intv; ntm := 3.0 / sqrt(accl);
if nt > ntm X 1.5 then
begin

comment parameters for auxiliary integration:
intv1 := utx / ntm; x := 2.0 X pi / intv1;
if x <= abs(c) then goto L2;

comment calculate convergence factor:
tausq := 0.33 X accl / (1.1 X (cfe(c - x) + cfe(c + x)));
if fail then goto L2;
accl := 0.07 X accl;
if ntm > lim then
begin
fault := 1; goto EXIT
end;

comment auxiliary integration:
integrate(ntm, intv1, tau sq, false); lim := lim - ntm;

comment find truncation point with new convergence factor:
findx(utx, 0.25 X accl); accl := 0.75 X accl;
goto L1
end;

comment main integration:
if nt > lim then
begin
iffault := 1; goto EXIT
end;
integrate(nt, intv, n, true);
qf := 0.5 - int11 - int12; trace[1] := eranl := eranl + eranc;

comment test whether round-off error could be significant. Allow
for radix 8 or 10 machines.
x := eranl + acc / 10.0;
for j := 1, 2, 4, 8 do
if j X x = j X eran then ifault := 2
end:
EXIT:
trace[7] := count
end qf