Algorithm AS 155

The Distribution of a Linear Combination of χ^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

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

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

$$\operatorname{pr}(Q<c). \tag{2}$$

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 λ_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

$$\sum_{k=K+1}^{\infty} \operatorname{Im}[\phi\{(k+1/2)\Delta\}\, e^{-i(k+1/2)\Delta c}]/\{\pi(k+1/2)\}, \tag{3}$$

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

$$\sum_{k=K+1}^{\infty} B\{(k+1/2)\Delta\}/\{\pi(k+1/2)\} \leqslant \int_{u=U}^{\infty} B(u)/(\pi u)\, du, \tag{4}$$

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

Writing

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

three possible forms for $B(u)$ are

$$N(u)\exp(-U^2\sigma^2/2)\prod_{(i)}(1+4U^2\lambda_j^2)^{-n_j/4}\prod_{(ii)}(4u^2\lambda_j^2)^{-n_j/4},$$

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

$$N(U)\exp(-u^2\sigma^2/2)\prod_1^r(1+4U^2\lambda_j^2)^{-n_j/4}$$

and

$$N(U)\left\{\prod_1^r(1+4U^2\lambda_j^2)^{n_j}\exp(2U^2\sigma^2)-1\right\}^{-1/4}$$

$$(U/u)^{1/2}\leqslant 1{\cdot}25N(U)\exp(-U^2\sigma^2/2)\prod_1^r(1+4U^2\lambda_j^2)^{-n_j/4}(U/u)^{1/2}$$

provided

$$\prod_1^r(1+4U^2\lambda_j^2)^{n_j}\exp(2U^2\sigma^2)\geqslant e \tag{5}$$

leading to bounds on the truncation error

$$\{2/(\pi s)\}\,N(U)\exp(-U^2\sigma^2/2)\prod_{(i)}(1+4U^2\lambda_j^2)^{-n_j/4}\prod_{(ii)}(4U^2\lambda_j^2)^{-n_j/4} \tag{6}$$

where $s=\sum_{(ii)}n_j$;

$$\{1/(\pi U^2\sigma^2)\}\,N(U)\exp(-U^2\sigma^2/2)\prod_1^r(1+4U^2\lambda_j^2)^{-n_j/4} \tag{7}$$

and

$$(2{\cdot}5/\pi)\,N(U)\exp(-U^2\sigma^2/2)\prod_1^r(1+4U^2\lambda_j^2)^{-n_j/4} \tag{8}$$

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

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

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

$$\mathrm{pr}(Q+\tau Z<c)-pr(Q<c)=\int_{-\infty}^{\infty}e^{-iuc}\{\exp(-\tau^2u^2/2)-1\}\,\phi(u)/(2\pi iu)\,du. \tag{9}$$

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

$$|\mathrm{pr}(Q+\tau Z<c)-\mathrm{pr}(Q<c)|\leqslant(\tau^2/\pi)\int_0^\infty\exp\left\{v\sum_1^r(1-4v\lambda_j)\lambda_j\delta_j^2/(1-4v\lambda_j+8v^2\lambda_j^2)\right\}$$

$$\times\prod_1^r(1-4v\lambda_j+8v^2\lambda_j^2)^{-n_j/4}\,v\,e^{-vc}\,dv\leqslant(\tau^2/\pi)\int_0^\infty\prod_{(i)}2^{(n_j+\delta_j^2)/4}\exp\{(v\sum_{(ii)}\lambda_j(n_j+\delta_j^2)\}\,v\,e^{-vc}\,dv$$

the product (i) and the sum (ii) involving only those values of j for which $\lambda_j>0$; those corresponding to large values of λ_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

$$(\tau^2/\pi)\Sigma \prod_{(i)} 2^{(n_j+\delta_j{}^2)/4}/\{c-\sum_{(ii)}\lambda_j(n_j+\delta_j^2)\}^2. \tag{10}$$

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

$$\mathrm{pr}(Q<c) = \{\mathrm{pr}(Q<c)-\mathrm{pr}(Q+\tau Z<c)\}+\mathrm{pr}(Q+\tau Z<c). \tag{11}$$

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

$$\sum_{n=1}^{\infty}(-1)^n\{\mathrm{pr}(Q+\tau Z<c-2\pi n/\Delta)-\mathrm{pr}(Q<c-2\pi n/\Delta) \\ +\mathrm{pr}(Q+\tau Z<c+2\pi n/\Delta)-\mathrm{pr}(Q<c+2\pi n/\Delta)\}. \tag{12}$$

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

$$\sum_{n=1}^{\infty}(-1)^n\{\mathrm{pr}(Q+\tau Z<c+2\pi n/\Delta)-\mathrm{pr}(Q<c+2\pi n/\Delta)\}$$

we find the term $\exp\{-i(v-iv)c\}$ must be replaced by

$$\exp\{-i(v-iv)(c+2\pi/\Delta)\}/\{1-\exp(-w+iw)\},$$

where $w = 2\pi v/\Delta$. But $|1/\{1-\exp(-w+iw)\}|\leqslant 1{\cdot}1$ 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

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

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

$$1-\exp(\tau^2u^2/2).$$

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 *qf*(*lb*, *nc*, *n*, *r*, *sigma*, *c*, *lim*, *acc*, *trace*, *ifault*)

Formal parameters

lb	Real array [1:*r*]	input : values of λ_j
nc	Real array [1:*r*]	input : values of δ_j^2
n	Integer array [1:*r*]	input : degrees of freedom of jth term
r	Integer	value : number of χ^2 terms in sum
sigma	Real	value : coefficient of normal variable

c	Real	value :	point at which distribution function is to be evaluated
lim	Integer	value :	maximum number of integration terms
acc	Real	value :	error bound
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*[6] standard deviation of convergence factor term *trace*[7] number of cycles to locate integration parameters
ifault	Integer	output :	fault indicator: ifault = 0 no error ifault = 1 requested accuracy could not be obtained ifault = 2 round-off error possibly significant ifault = 3 invalid parameters ifault = 4 unable to locate integration parameters

Realistic values for "*lim*" range from 1000 if the procedure is to be called repeatedly up to 50 000 if it is to be called only occasionally. Suitable values for "*acc*" range from 0·001 to 0·00005 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 χ^2 term has non-zero degrees of freedom and non-zero λ_j or that σ 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 "*intl*1", "*intl*2", "*ersm*1", "*ersm*2" should be converted to double precision and a double precision version of procedure "*ln*1" included.

Related Algorithm

An alternative algorithm, AS 106, which can be adapted to calculate the distribution of (1) provided that all the λ_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 λ_j to the smallest λ_j being not large. On the other hand, if the ratio of the largest λ_j to the smallest λ_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 λ_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 λ_j.

TABLE 1
Number of integration terms to calculate χ^2 probabilities

Degrees of freedom	Non-centrality parameter	χ^2 probability 0·01	0·5	0·99
1	0	9965	1327	182
2	0	1815	680	128
3	0	584	436	95
5	0	68	60	40
10	0	15	13	9
100	0	7	6	6
1	7·84	2268	494	81
3	11·56	35	28	19
5	12·96	16	13	9

TABLE 2
Number of integration terms to calculate F probabilities

Degrees of freedom		F probability		
Num.	Den.	0·01	0·5	0·99
1	1	6110	1784	6110
1	3	4315	401	254
1	5	4210	167	47
3	3	182	31	182
3	5	182	23	41
5	5	41	12	41

TABLE 3
Performance of algorithm

Quadratic form	c	Probability	Number of terms	Times (milliseconds) AS 155	AS 106
6, 1; 3, 1; 1, 1	1	0·0542	744	2532	22
	7	0·4936	625	2242	38
	20	0·8760	346	1174	65
6, 2; 3, 2; 1, 2	2	0·0064	74	269	19
	20	0·6002	66	255	66
	60	0·9838	50	203	176
6, 6; 3, 4; 1, 2	10	0·0027	18	103	35
	50	0·5648	15	96	168
	120	0·9912	10	82	525
7, 6, 6; 3, 2, 2	20	0·0061	16	77	23
	100	0·5913	13	70	88
	200	0·9779	10	63	156
7, 1, 6; 3, 1, 2	10	0·0451	603	1554	22
	60	0·5924	340	815	61
	150	0·9777	87	260	113
7, 6, 6; 3, 2, 2; 7, 1, 6; 3, 1, 2	70	0·0437	10	100	92
	160	0·5848	9	95	198
	260	0·9538	7	88	350
7, 6, 6; 3, 2, 2; −7, 1, 6; −3, 1, 2	−40	0·0782	10	98	—
	40	0·5221	8	92	—
	140	0·9604	10	96	—

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 χ^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 λ_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 χ^2 and F probabilities so the poor performance for χ^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 χ^2 random variable, a set of 2 or 3 numbers being the values of the weight, λ, the number of degrees of freedom and, when non-zero, the non-centrality parameter, δ^2. The accuracy was again set to 0·0001. The computer used was the Burroughs 6700 belonging to Victoria University of Wellington.

References

Davies, R. B. (1973). Numerical inversion of a characteristic function. *Biometrika*, **60**, 415–417.

Imhof, J. P. (1961). Computing the distribution of quadratic forms in normal variables. *Biometrika*, **48**, 419–426.

Sheil, J. and O'Muircheartaigh, I. (1977). Algorithm AS 106. The distribution of non-negative quadratic forms in normal variables. *Appl. Statist.*, **26**, 92–98.

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

comment  Algorithm AS 155  Appl. Statist. (1980) Vol. 29, No. 3;

value r, sigma, c, lim, acc; integer  r, lim, ifault;
real sigma, c, acc; real array lb, nc, trace; integer array n;

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

  begin
  real pi, ln28, sigsq, intl1, intl2, ersm1, ersm2, lmax, lmin, mean;
  integer count; Boolean ndtsrt, fail; integer array th[1 : r];

  comment label EXIT;

  procedure counter;

  comment count number of calls to errbd, truncation, cfe;

    begin
    count := count + 1;
    if count > lim then
      begin

      comment this error exit should almost never occur and could
      be replaced by an error message and stop, on compilers that
      do not handle goto exits from procedures;

      ifault := 4; goto EXIT
      end
    end 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;
  ndtsrt := 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 × sigsq;
  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 × lj;
    y := 1.0 - x; const := const + lj × (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 ctff(accx, upn); value accx; real accx, upn;

comment find ctff so that P(qf > ctff) < accx if upn > 0,
 P(qf < ctff) < 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 < 0.9 do
    begin
    u := (u1 + u2) / 2.0;
    if errbd(u / (1.0 + u X rb), const) > accx then
      begin
      u1 := u; c1 := const
      end
    else
      begin
      u2 := u; c2 := const
      end
    end;
  ctff := c2; upn := u2
  end ctff;

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

comment bound integration error due to truncation at u;

  begin
  real sum1, sum2, prod1, prod2, prod3, lj, ncj, x, y, err1, err2;
  integer j, nj, s;
  counter; sum1 := prod2 := prod3 := 0.0;
  s := 0; sum2 := (sigsq + tausq) X u ↑ 2;
  prod1 := 2.0 X sum2; u := 2.0 X u;
  for j := 1 step 1 until r do
    begin
    lj := lb[j]; ncj := nc[j];
    nj := n[j]; x := (u X lj) ↑ 2;
    sum1 := sum1 + ncj X x / (1.0 + x);
    if x > 1.0 then
      begin
      prod2 := prod2 + nj X ln(x);
      prod3 := prod3 + nj X ln1(x, true); s := s + nj
      end
    else prod1 := prod1 + nj X ln1(x, true)
    end j;
  sum1 := 0.5 X sum1; prod2 := prod1 + prod2;
  prod3 := prod1 + prod3; x := exp(-sum1 - 0.25 X prod2) / pi;
  y := exp(-sum1 - 0.25 X prod3) / pi;
  err1 := if s = 0 then 1.0 else x X 2.0 / s;
  err2 := if prod3 > 1.0 then 2.5 X y else 1.0;
  if err2 < err1 then err1 := err2;
  x := 0.5 X 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 X 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, tausq, main);
value nterm, interv, tausq, main; integer nterm;
real interv, tausq; Boolean main;
```

```
comment carry out integration with nterm terms, at stepsize interv. If
not main then multiply integrand by 1.0 - exp(-0.5 × tausq × u ↑ 2);

  begin real inpi, u, sum1, sum2, sum3, x, y, z; integer k, j, nj;
  inpi := interv / pi;
  for k := nterm step -1 until 0 do
    begin
    u := (k + 0.5) × interv; sum1 := -2.0 × u × c;
    sum2 := abs(sum1); sum3 := -0.5 × sigsq × u ↑ 2;
    for j := r step -1 until 1 do
      begin
      nj := n[j]; x := 2.0 × lb[j] × u;
      y := x ↑ 2; sum3 := sum3 - 0.25 × nj × ln1(y, true);
      y := nc[j] × x / (1.0 + y); z := nj × arctan(x) + y;
      sum1 := sum1 + z; sum2 := sum2 + abs(z);
      sum3 := sum3 - 0.5 × x × y
      end j;
    x := inpi × exp(sum3) / u;
    if ¬ main then x := x × (1.0 - exp(-0.5 × tausq × u ↑ 2));
    sum1 := sin(0.5 × sum1) × x; sum2 := 0.5 × sum2 × x;
    if abs(sum1) < acc then
      begin
      intl1 := intl1 + sum1; ersm1 := ersm1 + sum2
      end
    else
      begin
      intl2 := intl2 + sum1; ersm2 := ersm2 + sum2
      end
    end k
  end integrate;

real procedure cfe(x); value x; real x;

comment coef of tausq in error when convergence factor of
exp(-0.5 × tausq × u ↑ 2) is used when df is evaluated at x;

  begin real axl, axl1, axl2, sxl, sum1, lj; integer j, k, t;

  comment label L;

  counter;
  if ndtsrt then order;
  axl := abs(x); sxl := sign(x);
  sum1 := 0.0;
  for j := r step -1 until 1 do
    begin
    t := th[j];
    if lb[t] × sxl > 0.0 then
      begin
      lj := abs(lb[t]); axl1 := axl - lj × (n[t] + nc[t]);
      axl2 := lj / ln28;
      if axl1 > axl2 then axl := axl1 else
        begin
        if axl > axl2 then axl := axl2;
        sum1 := (axl - axl1) / lj;
        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
    end
  else cfe := 2.0 ↑ (sum1 / 4.0) / (pi × axl ↑ 2)
  end cfe;

comment ln28 = ln(2.0) / 8.0;
```

```
ln28 := 0.0866; pi := 3.14159265358979;
begin integer j, nj, nt, ntm;
real acc1, almx, utx, tausq, sd, intv, intv1, 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; intl1 := intl2 := ersm1 := ersm2 := 0.0;
qf := -1.0; acc1 := acc;
ndtsrt := true; fail := false;

comment find mean, sd, max and min of lb, check that parameter
values are valid;

sd := sigsq := sigma ↑ 2; lmax := lmin := mean := 0.0;
for j := 1 step 1 until r do
  begin
  nj := n[j]; lj := lb[j];
  ncj := nc[j];
  if nj < 0 ∨ ncj < 0.0 then
    begin
    ifault := 3; goto EXIT
    end;
  sd := sd + lj ↑ 2 × (2 × nj + 4.0 × ncj);
  mean := mean + lj × (nj + ncj);
  if lmax < lj then lmax := lj else
  if lmin > lj then lmin := lj
  end j;
if sd = 0.0 then
  begin
  qf := if c > 0.0 then 1.0 else 0.0; goto EXIT
  end;
if lmin = 0.0 ∧ lmax = 0.0 ∧ sigma = 0.0 then
  begin
  ifault := 3; goto EXIT
  end;
sd := sqrt(sd); almx := if lmax < -lmin then -lmin else lmax;

comment starting values for findu, ctff;

utx := 16.0 / sd; up := 4.5 / sd;
un := -up;

comment truncation point with no convergence factor;

findu(utx, 0.5 × acc1);

comment does convergence factor help?;

if c ≠ 0.0 ∧ almx > 0.07 × sd then
  begin
  tausq := 0.25 × acc1 / cfe(c);
  if fail then fail := false else
  if truncation(utx, tausq) < 0.2 × acc1 then
    begin
    sigsq := sigsq + tausq; findu(utx, 0.25 × acc1);
    trace[6] := sqrt(tausq)
    end
  end;
trace[5] := utx; acc1 := 0.5 × acc1;

comment find 'range' of distribution, quit if outside this;

L1:d1 := ctff(acc1, up) - c;
  if d1 < 0.0 then
    begin
    qf := 1.0; goto EXIT
    end;
  d2 := c - ctff(acc1, un);
  if d2 < 0.0 then
    begin
    qf := 0.0; goto EXIT
    end;
```

```
    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(acc1);
    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 acc1 / (1.1 X (cfe(c - x) + cfe(c + x)));
      if fail then goto L2;
      acc1 := 0.67 X acc1;
      if ntm > lim then
        begin
        ifault := 1; goto EXIT
        end;

      comment auxiliary integration;

      integrate(ntm, intv1, tausq, false); lim := lim - ntm;
      sigsq := sigsq + tausq; trace[3] := trace[3] + 1;
      trace[2] := trace[2] + ntm + 1;

      comment find truncation point with new convergence factor;

      findu(utx, 0.25 X acc1); acc1 := 0.75 X acc1;
      goto L1
      end;

    comment main integration;

 L2:trace[4] := intv;
    if nt > lim then
      begin
      ifault := 1; goto EXIT
      end;
    integrate(nt, intv, 0, true);
    trace[3] := trace[3] + 1; trace[2] := trace[2] + nt + 1;
    qf := 0.5 - intl1 - intl2; trace[1] := ersm1 := ersm1 + ersm2;

    comment test whether round-off error could be significant. Allow
    for radix 8 or 16 machines;

    x := ersm1 + acc / 10.0;
    for j := 1, 2, 4, 8 do
    if j X x = j X ersm1 then ifault := 2
    end;
EXIT:
  trace[7] := count
  end qf
```