7. Summary

We have introduced a formalism which allows us to explicate certain rather gross properties of language processing systems. As it is, the notation should be useful for designing the outlines of complex programming systems and their implementation, and it should be especially good for documentation. The formalism should also provide a mathematical basis which can be extended to handle more detailed properties of such systems. Some specific inadequacies where it could be extended follow.

1. It does not describe the amount of compilation or interpretation, unless it is coupled with precise definitions of the languages involved. For instance, in (7) we have no idea whether IL is close to machine language or to the source language. IL could be little more than assembly language, or just a trivial modification of the source language, or anything in between. Of course precise definitions of SL, IL, and ML would clear this up.

2. It does not permit the description of such processes as incremental compilation.

3. It does not permit the formal description of systems involving programs which consist of two or more pieces written in different languages, such as FSL.

Acknowledgment. We have benefitted from comments by J. Gray and J. Reynolds in preparing this paper.

Received January, 1970; revised June, 1970

REFERENCES

1. Bratman, H. An alternate form of the "UNCOL diagram." *Comm. ACM 4*, 3 (Mar. 1961), 142.
2. Burkhardt, W. H. Universal programming languages and processors: A brief survey and new concepts. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Pt. 1, Spartan Books, New York, pp. 1-21.
3. Evans, A. An Algol 60 compiler. Proc. ACM 18th Nat. Conf., 1963.
4. Feldman, Jerome A. A formal semantics for computer languages and its application in a compiler-compiler. *Comm. ACM 9*, 1 (Jan. 1966), 3-12.
5. McCarthy, J., et al. *Lisp 1.5 Programmers Manual.* MIT Press, Cambridge, Mass., 1968, pp. 76–77.
6. Newell, A. *IPL-V Manual.* Prentice-Hall, Englewood Cliffs, N.J., 1961.
7. Shaw, J. C. JOSS: A designer's view of an experimental online computing system. Rand Corp. P-2922, Santa Monica, Calif., Aug. 1964.
8. Sklansky, J., Finkelstein, M., and Russell, E. C. A formalism for program translation. *J. ACM 15*, 2 (Apr. 1968), 165–175.
9. Iturriaga, R., Standish, T., Krutar, R., Earley, J. Techniques and advantages of using the formal compiler writing system FSL to implement a Formula Algol compiler. Proc. AFIPS 1966 Spring Joint Comput. Conf. Vol. 28, Spartan Books, New York, 241-252.
10. Strong, J., et al. The problem of programming communication with changing machines: A proposed solution. *Comm. ACM 1*, 8 (Aug. 1958), 12–18; and *1*, 9 (Sept. 1958), 9–15.

ALGORITHM 395
STUDENT'S *t*-DISTRIBUTION [S14]

G. W. Hill (Recd. 17 Nov. 1969 and 23 Mar. 1970)
C.S.I.R.O., Division of Mathematical Statistics, Glen Osmond, South Australia

KEY WORDS AND PHRASES: Student's *t*-statistic, distribution function, approximation, asymptotic expansion
CR CATEGORIES: 5.12, 5.5

real procedure *student* (*t*, *n*, *normal*, *error*); **value** *t*, *n*; **real** *t*, *n*;
real procedure *normal*, *error*;

comment *student* evaluates the two-tail probability $P(t \mid n)$ that t is exceeded in magnitude for Student's [1] t-distribution with n degrees of freedom. The procedure provides results accurate to 11 decimal places and 8 significant digits for integer values of n, with approximate continuation of the function through noninteger values of n (over 6 decimal places for $n > 4.3$).

The procedure *normal* (χ) returns the area under the standard normal frequency curve to the left of χ, so that a negative argument yields the lower-tail area. The user-supplied procedure, *error*(n), should produce a diagnostic warning and may **go to** a label, terminate, or return a distinctive value (zero or -1.0) as a signal of error to the calling program.

Student's series expansion of the probability integral is supplemented by a faster asymptotic approximation for large values of n and by a more precise "tail" series expansion for large values of t.

The value of χ, defined as the normal deviate at the same probability level as t, may be approximated by an asymptotic normalizing expansion of Cornish-Fisher type [2].

$$\chi = z + (z^3+3z)/b - (4z^7+33z^5+240z^3+855z)/10b^2$$
$$+(64z^{11}+788z^9+9801z^7+89775z^5+543375z^3+1788885z)/210b^3 - \cdots$$

where $z = (a \times \ln(1+t^2/n))^{\frac{1}{2}}$, $a = n - \frac{1}{2}$ and $b = 48a^2$ [3]. This is well approximated by the first three terms with the third term's divisor replaced by

$$10b(b+0.8z^4+100).$$

The *student* probability is double the normal single-tail area, corresponding to the deviate χ.

The maximum error in the probability result for all values of t is displayed as a function of n in Figure 1, for this approximation, for the first few terms of the asymptotic expansion and for Fisher's [4] fifth-order approximation used in Algorithm 321 [5] for $n \geq 30$.

For small n and moderate t the result is calculated as $P(t \mid n) = 1 - A(t \mid n)$ using Student's cosine series for $A(t \mid n)$, rearranging formulas 26.7.3 and 26.7.4 of the NBS Handbook [6] in nested form

$$A(t|n \text{ odd}) = \frac{2}{\pi}\left[\arctan(y) + \frac{y}{b}\left\{1 + \frac{2}{3b}\left\{\cdots \frac{(n-5)}{(n-4)b} \cdot \left\{1 + \frac{(n-3)}{(n-2)b}\right\} \cdots\right\}\right\}\right]$$

$$A(t|n \text{ even}) = \frac{y}{\sqrt{(b)}}\left\{1 + \frac{1}{2b}\left\{\cdots \frac{(n-5)}{(n-4)b}\left\{1 + \frac{(n-3)}{(n-2)b}\right\} \cdots\right\}\right\},$$

where $y = \sqrt{(t^2/n)}$ and $b = 1 + t^2/n$. In the nested form, terms are treated in reverse order to the summation in Algorithm 321 and Algorithm 344 [7], reducing the number of operations required and reducing build up of roundoff error. Explicit decrementing of the "loop" parameter ensures that its final value remains defined on exit from the loop for use in an odd/even test.

Execution times for Fortran versions run on a CDC 3200 with programmed floating point are displayed in Figure 2, which indicates that nesting decreases the time for the cosine series method by about 30 percent and that it is appropriate to change over to the asymptotic method (using Algorithm 209 [8] for *normal*) when $n \geq 20$. Although this approximation would be accurate to more than 11 decimal places, the use of Algorithm 209 limits accuracy to about 9 decimals. This accuracy may be sufficient for many applications, in which case *student* may be abbreviated by deleting lines 15 and 27 through 35, removing the declaration and assignment of z from line 3, replacing line 5 by

if $n > entier(n) \vee n \geq 20$ **then**

and replacing line 25 by

student := **if** $a > 1.0$ **then** 0.0 **else** $1.0 - a$

The latter avoids spurious negative results due to roundoff error when a is near 1 for large values of t. The storage required for this abbreviated version was a little less than for Algorithm 344 and less than half that for Algorithm 321.

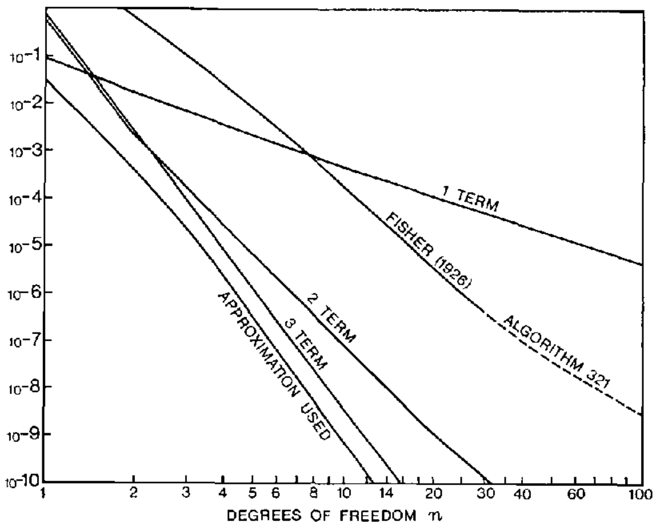

FIG. 1. Maximum error of approximations for "Student's" t-probability: 1, 2, and 3 term expansion, approximation with adjusted divisor, and Fisher's 5th order approximation

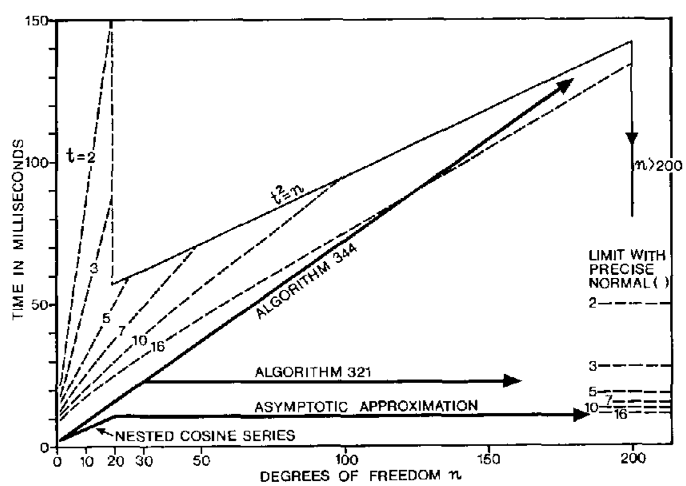

FIG. 2. Execution times (CDC3200 with programmed floating point). Broken lines: "tail" series for selected values of t (upper left); asymptotic method using precise *normal* (right)

Applications such as production of tables or function inversion to obtain extreme quantiles may require greater precision at extreme probability levels than these methods provide. For the cosine series and the asymptotic approximation using a high precision procedure for *normal*, such as Algorithm 304 [9], the relative error in the result increases in magnitude as the result decreases to extremely small values, as illustrated in Figure 3.

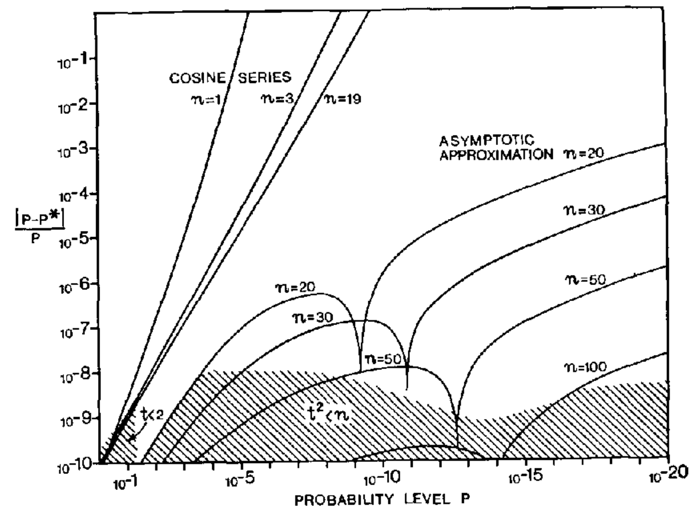

FIG. 3. Relative error, $|P - P^*|/P$, of approximation P^*; shaded region for restricted t values

For small P more precise results are obtained using a series expansion of $P(t \mid n)$ in terms of $w = 1/sqrt(1+t^2/n)$,

$$P(t|n) = C(n) \times w^n \left\{ \frac{1}{n} + \frac{1 \times w^2}{2(n+2)} + \frac{1 \times 3 \times w^4}{2 \times 4(n+4)} + \cdots \right\},$$

where $C(n) = \Gamma((n+1)/2)/(\sqrt{\pi} \times \Gamma(n/2))$. The series is summed till a negligible term occurs and then the factor $C(n) \times w^n$ is applied using the same repeated loop as the cosine series. Except for w near 1 when t is small, the truncation error is small, and accumulation of error in the repeated loop is moderate unless n is very large.

The cosine series method loses precision mainly in the subtraction $1 - A(t \mid n)$ as well as from the *sqrt* procedure and *arctan* when n is odd. In the worst case, $n = 19$, the error is kept below 3 decimals by changing to the tail series if $t > 2$, which ensures 8 significant digits in the result for the 36-bit (about 11 decimal) precision real variables for the processor used. As shown in Figure 3, change over from the asymptotic method to the tail series when $t^2 > n$ maintains about 8 significant digits in the result. For a machine of greater precision the use of more terms in the asymptotic series may be warranted, and the change over criteria would need adjustment to balance speeds and precision between the three methods.

Execution times for the tail series are shown as broken lines in Figure 2 for selected values of t: with bounds $t \geq 2$ for $n < 20$, $t^2 \leq n$ for $n \geq 20$ and with the limit $n < 200$ preventing excessive time for large t beyond a probability level near 10^{-40}. For the asymptotic method, using for *normal* a higher precision procedure based on Algorithm 304, the execution times for different values of the argument approach those shown at the right of Figure 2. Averaged over a range of arguments arising in practice, the provision for higher precision more than doubles the time required. In the case of Smirnov's [10] 6D tables of $S(t \mid n) = 1 - 0.5 \times P(t \mid n)$, retabulation to 10D, using the more precise procedure for *normal*, increased the time from about 7 minutes to 12 minutes, while introducing the tail series method to tabulate $P(t \mid n)$ over the same range to 8 significant digits increased the time further to about 16 minutes. Use of the asymptotic approximation enabled Smirnov's 6D tables of $\psi(t \mid 1000/\xi)$, which is an approximate continuation of $S(t \mid n)$ over noninteger values of $n = 1000/\xi$, to be extended to 10D for $\xi = 0(2)30$ in 5 minutes, and permits continuation to $\xi = 200$ with over 6D accuracy as indicated in Figure 1.

The preparation of diagrams by Murray C. Childs is gratefully acknowledged.

REFERENCES:

1. GOSSET, W. S. (Student). On the probable error of a mean. *Biometrika 6* (1908), 1.
2. HILL, G. W., AND DAVIS, A. W. Generalized asymptotic expansions of Cornish-Fisher type. *Ann. Math. Statist. 39*, 4(1968), 1264.
3. HILL, G. W. Progress results on asymptotic approximations for Student's t. Unpublished manuscript, Oct. 1969.
4. FISHER, R. A. Expansion of "Student's" integral in powers of n^{-1}. *Metron, 5* (1926), 109–112.
5. MORRIS, J. Algorithm 321, t-test. *Comm. ACM 11* (Feb. 1968), 115.
6. ABRAMOWITZ, M., AND STEGUN, I. A. (Eds.) Handbook of Mathematical Functions. Appl. Math. Ser. Vol. 55, Nat. Bur. Stand., US Govt. Printing Off., Washington, D.C., 1965, p. 948.
7. LEVINE, D. A. Algorithm 344, Student's t-distribution. *Comm. ACM 12* (Jan. 1969), 37.
8. IBBETSON, D. Algorithm 209, Gauss. *Comm. ACM 6* (Oct. 1963), 616.
9. HILL, I. D., AND JOYCE, S. A. Algorithm 304, Normal. *Comm. ACM 10* (June 1967), 374.
10. SMIRNOV, N. V. *Tables for the Distribution and Density Functions of t-Distribution.* Pergamon Press, New York, 1961;

```
if n < 1 then student := error(n) else
begin
  real a, b, y, z;  z := 1.0;
  t := t ↑ 2;  y := t/n;  b := 1.0 + y;
  if n > entier(n) ∨ n ≥ 20 ∧ t < n ∨ n > 200 then
  begin
    comment Asymptotic series for large or noninteger n;
    if y > ₁₀−6 then y := ln(b);
    a := n − 0.5;  b := 48.0 × a ↑ 2;  y := a × y;
    y := (((((−0.4×y−3.3)×y−24.0)×y−85.5)/
      (0.8×y↑2+100.0+b)+y+3.0)/b+1.0)×sqrt(y);
    student := 2.0 × normal(−y);
  end
  else
  if n < 20 ∧ t < 4.0 then
  begin
    comment Nested summation of "cosine" series;
    a := y := sqrt(y);  if n = 1 then a := 0.0;
loop:
    n := n − 2;  if n > 1 then
    begin a := (n−1)/(b×n) × a + y;  go to loop end;
    a := if n = 0 then a/sqrt(b)
      else (arctan(y)+a/b) × 0.63661977236;
    comment 2/π = 0.63661977236758134307553 51 ··· ;
    student := z − a
  end
  else
  begin
    comment "tail" series expansion for large t-values;
    integer j;  a := sqrt(b);  y := a × n;  j := 0;
    for j := j + 2 while a ≠ z do
    begin
      z := a;  y := y × (j−1)/(b×j);  a := a + y/(n+j)
    end;
    n := n + 2;  z := y := 0.0;  a := −a;  go to loop
  end
end
```

ALGORITHM 396
STUDENT'S t-QUANTILES [S14]

G. W. HILL (Recd. 6 Jan. 1970 and 18 May 1970)
C.S.I.R.O., Division of Mathematical Statistics, Glen Osmond, South Australia

KEY WORDS AND PHRASES: Student's t-statistic, quantile, asymptotic approximation
CR CATEGORIES: 5.12, 5.5

```
real procedure t quantile (P, n, normdev, error);
  value P, n; real P, n; real procedure normdev, error;
comment This algorithm evaluates the positive quantile at the
  (two-tail) probability level P, for Student's t-distribution with
  n degrees of freedom. The quantile function is an inverse of the
  two-tail
```

$$P(t|n) = 2\frac{\Gamma(\frac{1}{2}n+\frac{1}{2})}{\sqrt{(\pi n)}\Gamma(\frac{1}{2}n)}\int_t^\infty \frac{du}{(1+u^2/n)^{(\frac{1}{2}n+\frac{1}{2})}}$$

which is approximated in Algorithm 395 [1] by series whose inverses are used in this algorithm for t quantiles. Test calculations to 36-bit precision indicate that the result is correct to at least 6 significant digits, even for the analytic continuation through noninteger values of $n > 5$.

The procedure *normdev(p)* is assumed to return a negative normal deviate at the lower tail probability level p, e.g. -2.32 for $p = 0.01$. The user-supplied procedure for *error(n)* should give a diagnostic warning that the value of P or n is invalid and may **go to** a label, terminate, or return a distinctive value as an error signal to the calling program.

For $n = 1$ and $n = 2$ the exact result of integration is readily inverted to yield $t = cot(P\times\pi/2)$ and $t^2 = 2/(P(2-P))-2$, respectively. For larger n an asymptotic inverse expansion about normal deviates is applicable, while for smaller values of P a second series expansion is used to achieve sufficient precision. Both approximations have been adjusted to enhance precision for n as low as 3.

Both methods involve an expansion of the factor

$$d/n = \tfrac{1}{2}\sqrt{\pi}\Gamma(\tfrac{1}{2}n)/\Gamma(\tfrac{1}{2}n+\tfrac{1}{2})$$

in terms of $a = 1/(n-\frac{1}{2})$ and $b=48/a^2$

$$d/n = \sqrt{(a\pi/2)}(1-3/b+94.5/b^2-9058.5/b^3+\cdots)\ [2].$$

A three term approximation uses $b(b+c)$ instead of b^2 as a divisor, where the coefficients in

$$c = 96.36 - 16a - 98a^2 + 20700a^3/b,$$

have been fitted to ensure 8 significant digits in d for n as low as 3.

The inverse asymptotic expansion of Cornish-Fisher type relates a function $y(t) = \sqrt{[(n-\frac{1}{2})\ln(1+t^2/n)]}$ to the normal deviate χ at the corresponding probability level, $P/2$:

$$y=\chi - (\chi^3+3\chi)/b + (4\chi^7+63\chi^5+360\chi^3+945\chi)/10b^2$$
$$- (64\chi^{11}+1628\chi^9+19881\chi^7+145719\chi^5+694575\chi^3$$
$$+1902285\chi)/210b^3 + \cdots\ [2],$$

whence $t = \sqrt{[n\times(exp(a\times y^2)-1)]}$. For a three term approximation the third term's divisor is replaced by

$$10b \times (b+c-2\chi-7\chi^2-5\chi^3+0.05\times d\times\chi^4),$$

whose coefficients have been fitted to reduce the error for small n and for larger n and χ. For $n < 5$, c is increased by $0.3(n-4.5)(\chi+0.6)$ to further reduce error in an interval of P not well covered by the following approximation.

For small P, where t^2/n is large, the integrand may be expanded in terms of $w^2 = 1/(1+t^2/n)$ and integrated term by term to yield

$$P = \frac{nw^n}{d}\left\{\frac{1}{n} + \frac{w^2}{2(n+2)} + \frac{1 \times 3w^4}{2 \times 4(n+4)} + \cdots\right\},$$

which may be inverted to express t^2/n in terms of $y = (P \times d)^{2/n}$

$$\frac{t^2}{n} = \frac{1}{y} + \frac{n+1}{n+2}\left\{-1 + \frac{y}{2(n+4)} + \frac{n \times y^2}{3(n+2)(n+6)} + \frac{n(n+3)(2n^2+9n-2)y^3}{8(n+2)^2(n+4)^2(n+8)} + \cdots\right\}.$$

Since the ratio of successive terms is nearly $n \times y/(n+6)$ for small n, replacement of the term in y^3 by $y/[3(n+2)\{(n+6)/(n \times y)-1.0\}]$ provides an approximate allowance for subsequent terms in the series, which is empirically improved by replacing the -1.0 by $-0.822 - 0.089 \times d$.

As n and P increase, the errors for the asymptotic approximation decrease, whereas errors for the second series increase, so that for each value of n the error curves intersect at a value of P above which the asymptotic approximation is better and below which the second series should be used. By adjusting the two approximations the error level at these intersections has been balanced at about the seventh significant digit for $n \geq 3$ and $P > 10^{-24}$. The value of y at these points is about $a + 0.05$ and this fact provides a convenient criterion for selecting which approximation to use: the asymptotic series if y exceeds $a + 0.05$, otherwise the second series.

Although better approximations could be obtained by use of more terms in each series, greater precision can be achieved by using the result of this algorithm as a starting value for iterative inversion of $P(t \mid n)$, whose value and derivative can be computed with considerable precision using recurrence relations as in Algorithm 395.

A comparison of results from this algorithm against values obtained by inverting the function provided by Algorithm 395 indicates a precision of over 6 significant digits for $10^{-24} \leq P \leq 0.9$, $n \geq 1$. At the conventional tabulation points in $0.001 \leq P \leq 0.9$ results for $n = 1$, $n = 2$, and $n > 10$ checked to 8 significant digits.

Previously published tables [3, 4, 5] provide 3 or 4 decimal place check values, some of which are found to be slightly in error. Thus for $n = 2$, $P = 0.001$, t is given as 31.598 by Fisher and Yates and by Federighi, 31.5991 by Smirnov, and 31.5990546 by this procedure, while for $n = 1$, $P = 0.001$ the value 636.6096 given by Smirnov conflicts with Fisher and Yates, Federighi (636.619) and this procedure (636.61925). Other errors in the last few digits in Smirnov's table for low values of n and P include 10.2129 for $n = 3$, $P = 0.002$, which should be 10.2145, and 4.7812 for $n = 9$, $P = 0.001$, which should be 4.7809.

t quantile may be used to obtain percentiles at values of P and n not provided in existing tables or for extending their accuracy. Such tables are customarily used for assessing the significance of a sample value for t, but for automatic computation the probability level is more effectively determined as $P(t \mid n)$ using a direct procedure such as Algorithm 395.

Pseudorandom t-values may be generated for sampling applications by using uniformly distributed pseudorandom numbers for P, and in this case *normdev* may be a **real procedure** returning pseudorandom normal deviates which are independent of P.

REFERENCES:

1. Hill, G. W. Algorithm 395, Student's t-distribution *Comm. ACM 13* (Oct. 1970), 617–618.
2. Hill, G. W. Progress results on asymptotic approximations for Student's t. Unpublished manuscript, Oct. 1969.
3. Fisher, R. A., and Yates, F. *Statistical Tables for Biological Agricultural and Medical Research.* Oliver and Boyd, London, 1963.
4. Smirnov, N. V. *Tables for the Distribution and Density Functions of t-Distribution.* Pergamon Press, New York, 1961.
5. Federighi, E. T. Extended tables of the percentage points of Student's t-distribution. *J. Amer. Stat. Assoc. 54* (1959), 683–688;


```
if n < 1 ∨ P > 1.0 ∨ P ≤ 0.0 then t quantile := error(n)
else if n = 2 then t quantile := sqrt(2.0/(P×(2.0−P))−2.0)
else
begin
  real half pi;  half pi := 1.5707963268;
  if n = 1 then
  begin P := P × half pi;  t quantile := cos(P)/sin(P) end
  else
  begin
    real a, b, c, d, x, y;
    a := 1.0/(n−0.5);  b := 48.0/a ↑ 2;
    c := ((20700×a/b−98)×a−16) × a + 96.36;
    d := ((94.5/(b+c)−3.0)/b+1.0) × sqrt(a×half pi) × n;
    x := d × P;  y := x ↑ (2.0/n);
    if y > 0.05 + a then
    begin
      comment  Asymptotic inverse expansion about normal;
      x := normdev(P×0.5);  y := x ↑ 2;
      if n < 5 then c := c + 0.3 × (n−4.5) × (x+0.6);
      c := (((0.05×d×x−5.0)×x−7.0)×x−2.0) × x + b + c;
      y  :=  (((((0.4×y+6.3)×y+36.0)×y+94.5)/c−y−3.0)/b+
        1.0) × x;
      y := a × y ↑ 2;
      y := if y>0.002 then exp(y) − 1.0 else 0.5 × y ↑ 2 + y
    end
    else y := ((1.0/(((n+6.0)/(n×y)−0.089×d−0.822)×
      (n+2.0)×3.0)+0.5/(n+4.0))×y−1.0) ×
        (n+1.0)/(n+2.0) + 1.0/y;
    t quantile := sqrt(n×y)
  end
end Student's t-quantile
```

ALGORITHM 397
AN INTEGER PROGRAMMING PROBLEM [H]

S. K. Chang and A. Gill (Recd. 16 Feb. 1970 and 11 May 1970)

Electronics Research Laboratory and Department of Electrical Engineering and Computer Sciences, University of California,* Berkeley, CA 94720

* Research sponsored by the Air Force Office of Scientific Research Office of Aerospace Research, United States Air Force, AFOSR Grant AF-AFOSR-639-67 and the National Science Foundation, Grant GK2277.

KEY WORDS AND PHRASES: integer programming, change-making problem

CR CATEGORIES: 5.41


```
procedure MINDIST(C, M, SENSE, W, RESULT);
  value C, M;  integer C, M;  Boolean SENSE;
  integer array W, RESULT;
comment  This algorithm solves an integer programming prob-
  lem described in [1]. Given is a fixed weight vector w = (w1 ,
  w2 , ··· , wm), where the wi are nonnegative integers, where m
  is a positive integer, and where

                    1 = w1 < w2 < ··· < wm
```

For any nonnegative integer c (representing cost), an m-distribution of c relative to w is an m-tuple $(a_1, a_2, \cdots, a_m)$ such that the a_i are nonnegative integers, and such that $\sum_{i=1}^{m} a_i w_i = c$. The m-distribution $(a_1, a_2, \cdots, a_m)$ is minimal if, for any m-distribution $(b_1, b_2, \cdots, b_m)$ of c relative to w, we have $\sum_{i=1}^{m} a_i \leq \sum_{i=1}^{m} b_i$. The m-distribution $(a_1, a_2, \cdots, a_m)$ is standard if it is obtainable as follows:

$$c_m = c$$

$$c_i = c_{i+1} - a_{i+1} \times w_{i+1} \qquad (i=m-1, m-2, \cdots, 1)$$

$$a_i = c_i/w_i \qquad (i=m, m-1, \cdots, 1)$$

(where all divisions are integer divisions).

If *MINDIST*(*C*, *M*, *SENSE*, *W*, *RESULT*) is called with a nonnegative integer *C*, a positive integer *M*, and an array *W* = (*W*[1], *W*[2], $\cdots$, *W*[*M*]), then the resulting array
RESULT = (*RESULT*[1], *RESULT*[2], $\cdots$, *RESULT*[*M*])
is a minimal *M*-distribution of *C* relative to *W*. If, before calling *MINDIST*, *SENSE* is set to **true**, then *MINDIST* retains *SENSE* as **true** if and only if *RESULT* is also a standard *M*-distribution of *C* relative to *W*.

REFERENCE:

1. CHANG, S. K., AND GILL, A. Algorithmic solution of the change-making problem. *J. ACM 17* (Jan. 1970) 113–122;


```
begin
  integer I, J, R, Q, SUM, SUN;
  integer array A[1:M], B[1:M];
  if M = 1 then
  begin
    RESULT[1] := C;
EXIT1 :
    go to EXIT
  end
  Q := C/W[M];
  if (Q×W[M]) > C then Q := Q − 1;
  R := C − W[M] × Q;
  if M = 2 then
  begin
    RESULT[1] := R;  RESULT[2] := Q;
EXIT2 :
    go to EXIT
  end;
  J := 0;
LOOP:
  MINDIST (R+J×W[M], M−1, SENSE, W, B);
  if J ≠ 0 then go to NOT ZERO;
BETA:
  for I := 1 step 1 until M−1 do A[I] := B[I];
  A[M] := 0;
GAMMA:
  if J = Q then
  begin
    for I := 1 step 1 until M do RESULT[I] := A[I];
EXIT3:
    go to EXIT
  end;
  SUM := 0;
  for I := 1 step 1 until M do SUM := SUM + A[I];
  if (W[M]×SUM−R−J×W[M])/(W[M]−W[M−1]) ≤ 0 then
  begin
    for I := 1 step 1 until M − 1 do RESULT[I] := A[I];
    RESULT[M] := A[M] + Q − J;
EXIT4:
    go to EXIT
  end;
  J := J + 1;
  go to LOOP;
NOT ZERO:
  SUM := 0;  SUN := 0;
  for I := 1 step 1 until M do SUM := SUM + A[I];
  for I := 1 step 1 until M − 1 do SUN := SUN + B[I];
  if SUM ≤ SUN then
  begin A[M] := A[M] + 1; go to GAMMA end;
  SENSE := false;
  go to BETA;
EXIT:
end PROCEDURE MINDIST
```

ALGORITHM 398
TABLELESS DATE CONVERSION* [Z]

RICHARD A. STONE (Recd. 2 Jan. 1970 and 6 April 1970)
Western Electric Company, P.O. Box 900,
Princeton, NJ 08540

* Patent applied for.

KEY WORDS AND PHRASES: date, calendar
CR CATEGORIES: 5.9


```
procedure calendar(y, n, m, d);
  value y, n;  integer y, n, m, d, t;
comment  calendar is called with the year in y and the day of the
  year in n. The month number is returned in m, and the day of the
  month is returned in d. The first section of the procedure changes
  the dates so that February has 30 days. The second section uses
  the fact that 30.55 (m+2) − 91 passes through the number of
  days preceeding each month.
    Error detection: m will be in the range 1-12 if and only if n
  is in the correct range;
begin
  t := if (y ÷ 4)*4 = y then 1 else 0;
  comment  The following statement is unnecessary
    if it is known that 1900 < y < 2100;
  t := if (y÷400)*400 = y ∨ (y÷100)*100 ≠ y then t else 0;
  d := n + (if n > (59+t) then 2 − t else 0);
  m := ((d+91)*100) ÷ 3055;
  d := (d+91) − (m*3055) ÷ 100;
  m := m − 2
end calendar
```

ALGORITHM 399
SPANNING TREE [H]

JOUKO J. SEPPÄNEN (Recd. 6 Jan. 1970 and 8 May 1970)
Computing Center, Helsinki University of Technology,
Otaniemi, Finland

KEY WORDS AND PHRASES: graph, tree, spanning tree
CR CATEGORIES: 5.32


```
procedure spanning tree(v, e, I, J, p, T);
  value v, e;  integer v, e, p;  integer array I, J, T;
```

comment This procedure grows a spanning tree T for a given undirected loop-free graph $G = (N, E)$ of v vertices and e edges. If G is disconnected a spanning forest will be grown.

The edges $(I[k], J[k]) \in E$ for $k = 1, 2, \cdots, e$ are assumed to be stored in the arrays $I[1{:}e]$ and $J[1{:}e]$. At each stage of the algorithm one edge is considered whereby one of four possible conditions will arise. If neither of the vertices is included in a tree, this edge is taken as a new tree and its vertices numbered by an incremented component number c. If one vertex is in a tree, the edge will be grown to this tree. If the two vertices are in different trees, these will be grafted into a single tree by renumbering the vertices of the other component. Finally, if both vertices are in the same tree, the edge completes a fundamental cycle of the graph with respect to the spanning tree and consequently will not be considered further. At the end, the indices of the edges in the spanning tree are stored in the array $T[1{:}v-p]$ where p is the number of trees in the forest. The procedure can also be used to find a minimal spanning tree by sorting the edges into ascending order before calling the procedure.

The main loop in the procedure is executed e times. For cases where the ratio e/v is high it could be worthwhile to introduce an additional variable, say d, in the program, for keeping a count of the number of edges included in T. When d has attained the value of $v - 1$ the algorithm could terminate.

REFERENCES:

1. BERGE, C., AND GHOUILA-HOURI, A. *Programmes, Jeux et Reseaux de Transport*. Dunod, Paris, 1962, pp. 179–182.
2. BERGE, C., AND GHOUILA-HOURI, A. *Programming, Games and Transportation Networks*. Methuen, London, and Wiley, New York, 1965, pp. 177–180.
3. KRUSKAL, J. B., JR. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proc. Amer. Math. Soc. 7* (1956) 48–50.
4. OBRUCA, A. Algorithm 1. Mintree. *Computer Bull.* (Sept. 1964) 67.
5. KNUTH, D. E. *The Art of Computer Programming, Vol I Fundamental Algorithms*. Addison-Wesley, Reading, Mass., 1968. pp. 370–371;


```
begin
  integer i, j, k, c, n, r;
  integer array V[1:v];
  c := n := 0;
  for k := 1 step 1 until v do V[k] := 0;
  for k := 1 step 1 until e do
  begin
    i := I[k];  j := J[k];
    if V[i] = 0 then
    begin
      T[k-n] := k;
      if V[j] = 0 then V[i] := V[j] := c := c + 1
      else
      V[i] := V[j]
    end
    else if V[j] = 0 then
    begin
      T[k-n] := k;  V[j] := V[i]
    end
    else if V[i] ≠ V[j] then
    begin
      T[k-n] := k;  i := V[i];  j := V[j];
      for r := 1 step 1 until v do
        if V[r] = j then V[r] := i
    end graft
    else n := n + 1
  end edge;
  p := v - e + n
end spanning tree
```

ALGORITHM 400
MODIFIED HAVIE INTEGRATION [D1]

GEORGE C. WALLICK (Recd. 26 Jan. 1970 and 25 Apr. 1970)

Mobil Research and Development Corporation, Field Research Laboratory, P.O. Box 900, Dallas, TX 75221

KEY WORDS AND PHRASES: numerical integration, Havie integration, Romberg quadrature, modified Romberg-quadrature, trapezoid values, rectangle values

CR CATEGORIES: 5.16

DESCRIPTION:

The Havie integration method for the approximate evaluation of the definite integral

$$I = \int_A^B F(x)\,dx \tag{1}$$

as implemented in ACM Algorithm 257 [4] is based upon the parallel generation of the Romberg table of trapezoidal $T_j{}^k$ values [1] and the table of rectangular $R_j{}^k$ values also used by Krasun and Prager [3]. At each step in the development of the tables the difference $| T_j{}^k - R_j{}^k |$ is examined. If $| T_j{}^k - R_j{}^k | \le \epsilon$ the process is said to have converged and the algorithm returns a value of

$$T_j^{k+1} = \tfrac{1}{2}(T_j{}^k + R_j{}^k). \tag{2}$$

For some $F(X)$, e.g. $F(X) = e^{-X^2}$ and $F(X) = 2/(2+\sin 10\pi X)$, the $R_j{}^k$, $T_j{}^k$ pairs converge more rapidly than the Romberg sequence of $T_j{}^k$ values. (This is the same class of $F(X)$ for which a simple nonadaptive Simpsons Rule algorithm [5] is competitive with the Havie algorithm.) For other $F(X)$, the Havie algorithm is slightly less efficient than the Romberg algorithm.

Like Romberg quadrature, Havie integration requires the evaluation of the rectangular values

$$R_o{}^k = \frac{B-A}{2^k} \sum_{j=1}^{2^k} F\left[A + (j-\tfrac{1}{2})\,\frac{B-A}{2^k}\right]. \tag{3}$$

Rutishauser [6] recognized that this repeated addition of small terms to a large partial sum can lead to serious roundoff error. He suggested a procedure for the evaluation of the $R_o{}^k$ which significantly reduces this error. The method, used by Fairweather [2] in a modified Romberg algorithm, leads to a significant improvement in accuracy for large orders of extrapolation.

In the modified Havie integration algorithm HRVINT the $R_o{}^k$ are evaluated using a 3-level version of the Rutishauser procedure. The arguments X of the generating function $F(X)$ are evaluated as in eq. (3) rather than by accumulative addition as in Algorithm 257.

In the argument list for HRVINT, F is the name of the generating function FUNCTION $F(X)$ which returns a value of $F(X)$ corresponding to a specified value of X, A, and B represent the lower and upper limits of integration, and MAX is the maximum order of extrapolation to be permitted, MAX $\le$ 16. Values of MAX $>$ 16 are interpreted as MAX = 16; the value of MAX is not changed by the subprogram. Computation is terminated when

$$| T_j{}^k - R_j{}^k | \le \text{ACC} * | T_j{}^k |$$

or when the order of extrapolation MFIN = MAX. Here ACC is a measure of the desired relative accuracy, ACC $>$ 0. Upon exit HRVINT is the approximate value of the integral, FAC is a measure of the final relative accuracy achieved

$$\text{FAC} = | T_j{}^k - R_j{}^k | / | T_j{}^k |$$

and MFIN is the order of extrapolation.

Test case. HRVINT was tested in Fortran IV on a CDC 6400 computer using single-precision floating point arithmetic (14+ decimal digits). Corresponding integral values were also obtained using a Fortran version of the standard Havie Algorithm 257. The results of these tests are summarized in Table I.

For modest accuracy requirements, the two algorithms are seen to be equivalent. For both algorithms the maximum accuracy achievable is limited by truncation and roundoff error. Since the Rutishauser modification serves to reduce the magnitude of such errors, the modified Havie algorithm can, in many cases, return optimum integral values that are from 1 to 2 significant figures more accurate than those returned by Algorithm 257.

In the routine use of the algorithms it is possible to specify an accuracy requirement that cannot be satisfied. When this condition obtains, the algorithms are forced to proceed to the maximum permitted extrapolation order. With Algorithm 257 error accumulation accompanying such an overspecification can lead to a serious decline in evaluation accuracy. With the modified Havie algorithm HRVINT this loss is minimized and in most cases virtually eliminated.

TABLE I. A Comparison of the Havie and Modified Havie Algorithms

$$I = \int_A^B F(X)\,dX$$

(*m = Extrapolation Order, m ≤ 16; N.S.F. = Number of Significant Figures*)

F(X)	A	B	Correct value (digits 10–16)	*Numerical Evaluation*						
				Specified relative accuracy	*Havie*			*Modified Havie*		
					I (digits 10–14)	m	N.S.F.	I (digits 10–14)	m	N.S.F.
e^{-x^2}	0.0	5.0	45139 55	10^{-1}–10^{-2}	46726	3	10	46726	3	10
				10^{-3}–10^{-10}	45039	4	11	45039	4	11
				10^{-11}	45110	5	12	45111	5	12
				10^{-12}	45128	6	12	45131	6	12
				10^{-13}	45134	6	12	45137	6	13
				10^{-14}	39757	16	9	45137	7	13
				10^{-15}	39757	16	9	45136	10	13
$\ln x$	1.0	10.0	29940 46	10^{-9}	29845	8	11	29846	8	11
				10^{-10}	29937	8	13	29939	8	13
				10^{-11}–10^{-12}	29937	9	13	29940	9	14
				10^{-13}	29937	9	13	29940	10	14
				10^{-14}	29556	16	11	29940	10	14
$(1 + x)^{-1}$	0.0	1.0	55994 53	10^{-9}	56353	6	11	56354	6	11
				10^{-10}	55996	6	13	55997	6	13
				10^{-11}	55990	6	13	55991	6	13
				10^{-12}	55988	7	12	55991	7	13
				10^{-13}	55987	8	12	55991	7	13
				10^{-14}–10^{-15}	53242	16	10	55991	9	13
$(1 + x^4)^{-1}$	0.0	1.0	33991 10	10^{-6}–10^{-7}	35633	5	10	35634	5	10
				10^{-8}–10^{-10}	33993	6	13	33995	6	13
				10^{-11}–10^{-12}	33984	7	12	33989	7	13
				10^{-13}	30854	16	10	33987	7	13
				10^{-14}–10^{-15}	30854	16	10	33988	9	13
x^{-3}	0.01	1.1	68595 04	10^{-8}	71022	13	10	71529	13	10
				10^{-9}	68136	13	11	68647	13	11
				10^{-10}	68076	13	10	68589	13	12
				10^{-11}	64508	16	10	68590	14	12
				10^{-12}–10^{-13}	64508	16	10	68589	14	12
				10^{-14}–10^{-15}	64508	16	10	68584	16	12
x^{-4}	0.01	1.1	89506 64	10^{-8}	89368	13	11	89694	13	11
				10^{-9}	89199	13	11	89526	13	12
				10^{-10}	88857	14	10	89503	14	13
				10^{-11}–10^{-12}	86878	16	10	89502	14	13
				10^{-13}	86878	16	10	89502	15	13
				10^{-14}–10^{-15}	86878	16	10	89499	16	12
x^{-5}	0.01	1.1	29246 64	10^{-8}	29556	13	11	29767	13	10
				10^{-9}–10^{-10}	28828	14	11	29247	13	14
				10^{-11}	27557	16	10	29245	14	13
				10^{-12}–10^{-13}	27557	16	10	29244	15	13
				10^{-14}	27557	16	10	29244	16	13
				10^{-15}	27557	16	10	29242	16	13

Acknowledgment. The author wishes to thank Mobil Research and Development Corporation for permission to publish this information.

References:

1. Bauer, F. L. Algorithm 60, Romberg integration. *Comm. ACM 4* (June 1961), 255.
2. Fairweather, G. Algorithm 351, Modified Romberg quadrature. *Comm. ACM 12* (June 1969), 324–325.
3. Krasun, A. M., and Prager, W. Remark on Romberg quadrature. *Comm. ACM 8* (Apr. 1965), 236–237.
4. Kubik, R. N. Algorithm 257, Havie integrator. *Comm. ACM 8* (June 1965), 381.
5. Perlis, A. J., and Samelson, K. Preliminary report—international algebraic language. *Comm. ACM 1* (Dec. 1958), 8–22.
6. Rutishauser, H. Description of Algol 60. In *Handbook for Automatic Computation, Vol. 1.* Springer-Verlag, New York, 1967, Part a, pp. 105–106.

Algorithm:

```
      FUNCTION HRVINT(F,A,B,MAX,ACC,FAC,MFIN)
C HAVIE INTEGRATION WITH AN EXPANDED RUTISHAUSER-
C TYPE SUMMATION PROCEDURE
      DIMENSION T(17),U(17),TPREV(17),UPREV(17)
C TEST FOR MAX GREATER THAN 16
      MUX=MAX
      IF(MAX-16)10,10,5
    5 MUX=16
C INITIALIZATION
   10 ENPT=0.5*(F(A)+F(B))
      SUMT=0.0
      MFIN=1
      N=1
      H=B-A
      SH=H
C BEGIN REPETITIVE LOOP FROM ORDER 1 TO ORDER MAX
   15 T(1)=H*(ENPT+SUMT)
      SUM=0.
      NN=N+N
      EN=NN
      EM=SH/EN
C BEGIN RUTISHAUSER EVALUATION OF RECTANGULAR SUMS
C INITIALIZATION
      IF(NN-16)20,20,25
   20 NZ=NN
      GO TO 30
   25 NZ=16
      IF(NN-256)30,30,35
   30 NA=NN
      GO TO 40
   35 NA=256
      IF(NN-4096)40,40,45
   40 NB=NN
      GO TO 50
   45 NB=4096
C DEVELOPMENT OF RECTANGULAR SUMS
   50 DO 70 KC=1,NN,4096
         SUMB=0.
         KK=KC+NB-1
         DO 65 KB=KC,KK,256
            SUMA=0.
            KKK=KB+NA-1
            DO 60 KA=KB,KKK,16
               SUMZ=0.
               KFR=KA+NZ-1
               DO 55 KZ=KA,KFR,2
                  ZKZ=KZ
   55          SUMZ=SUMZ+F(A+ZKZ*EM)
   60       SUMA=SUMZ+SUMA
   65    SUMB=SUMA+SUMB
   70 SUM=SUMB+SUM
C END OF RUTISHAUSER PROCEDURE
      U(1)=H*SUM
      K=1
C BEGIN EXTRAPOLATION LOOP
   75 FAC=ABS(T(K)-U(K))
      IF(T(K))80,85,80
C TEST FOR RELATIVE ACCURACY
   80 IF(FAC-ABS(ACC*T(K)))90,90,100
C TEST FOR ABSOLUTE ACCURACY WHEN T(K)=0
   85 IF(FAC-ABS(ACC))95,95,100
   90 FAC=FAC/ABS(T(K))
C INTEGRAL EVALUATION BEFORE EXIT
   95 HRVINT=0.5*(T(K)+U(K))
      RETURN
  100 IF(K-MFIN)105,115,115
  105 AK=K+K
      D=2.**AK
      DMA=D-1.0
```

```
C BEGIN EXTRAPOLATION
         T(K+1)=(D*T(K)-TPREV(K))/DMA
         TPREV(K)=T(K)
         U(K+1)=(D*U(K)-UPREV(K))/DMA
         UPREV(K)=U(K)
C END EXTRAPOLATION
         K=K+1
         IF(K-MUX)75,110,110
C END EXTRAPOLATION LOOP
  110    FAC=ABS(T(K)-U(K))
         IF(T(K))90,95,90
C ORDER IS INCREASED BY ONE
  115    H=0.5*H
         SUMT=SUMT+SUM
         TPREV(K)=T(K)
         UPREV(K)=U(K)
         MFIN=MFIN+1
         N=NN
         GO TO 15
C RETURN FOR NEXT ORDER EXTRAPOLATION
         END
```

REMARK ON ALGORITHM 304[S15]
NORMAL CURVE INTEGRAL [I. D. Hill and S. A. Joyce, *Comm. ACM 10*(June 1967), 374]

Bo Holmgren (Recd. 30 Apr. 1970)
Dept. KDO, ASEA, S-721 83 Västerås, Sweden

KEY WORDS AND PHRASES: normal curve integral, probability, special functions
CR CATEGORIES: 5.12, 5.5

Algorithm 304 with the remark of Adams was translated into Fortran IV and run on a GE-625 computer. The GE-625 has a 28-bit mantissa and allows exponents up to 10^{38}. With *upper* = **false** and $x < -2.32$, the routine ran into overflow at several values of x. To avoid this the following lines

if $q2 > 10^{30}$ **then**
begin
$p1 := p1 \times {}_{10}{-30}$; $p2 := p2 \times {}_{10}{-30}$;
$q1 := q1 \times {}_{10}{-30}$; $q2 := q2 \times {}_{10}{-30}$
end;

were inserted after the line

$s := m$; $m := t$;

REMARK ON ALGORITHM 347 [M1]
AN EFFICIENT ALGORITHM FOR SORTING WITH MINIMAL STORAGE [Richard C. Singleton, *Comm. ACM 12* (Mar. 1969), 185]

Richard Peto (Recd. 18 Feb. 1970)
Medical Research Council, 115 Gower Street, London W. C. 1

KEY WORDS AND PHRASES: sorting, ranking, minimal storage sorting, digital computer sorting
CR CATEGORIES: 5.31

If the values of ij, instead of always being $(i+j) \div 2$, are at varying positions between i and j, then there is less likelihood of peculiar initial structure causing failure of the algorithm to perform rapidly. The position of ij can be made to vary by replacing the statements

$m := 0$; $ii := i$; **go to** $L4$; $L1$: $ij := (i+j) \div 2$;

by

real r; $r := 0.375$; $m := 0$; $ii := i$; **go to** $L4$;
$L1$: $r :=$ **if** $r > 0.58984375$ **then** $r - 0.21875$ **else** $r + 0.0390625$;
$ij := i + (j-i) \times r$;

comment These four decimal constants, which are respectively 48/128, 75.5/128, 28/128, and 5/128, are rather arbitrary. On most compilers their binary representations will be exact, and the use of them in the statement $L1$ causes r to vary cyclically over the 33 values $48/128 \cdots 80/128$. Therefore ij takes a variable position somewhere within the middle quarter of the segment to be sorted. Wider variation of ij would be undesirable in the special case of a partially presorted array;

In sorting an array of N elements which are initially in random order this will waste (on ICL Atlas) less than $N/10^5$ seconds, but if the array is, for example, composed initially of two equal presorted halves, then the use of the original rather than the modified version would more than double the sorting time required if $N > 10^4$.

As the author points out, the published version could fail if used to sort arrays of 1024 or more elements because the upper bounds of IU and IL might be inadequate. For a standard procedure the declaration IL, IU [0:8] should be replaced by the declaration IL, IU [0:20]. This permits the sorting of arrays of up to 4 million elements, which is, with present core store sizes, sufficient.

The statement $tt := a[L]$ which precedes $L3$: will be executed less frequently if it is transferred into the next conditional statement, which then reads

if $k \le L$ **then begin** $tt := a[L]$; $a[L] := a[k]$; $a[k] := tt$; **go to** $L2$ **end**

REMARK ON ALGORITHM 368 [D5]
NUMERICAL INVERSION OF LAPLACE TRANSFORMS [Harald Stehfest, *Comm. ACM 13* (Jan. 1970),47]

Harald Stehfest (Recd. 6 May 1970)
Institut f. angew. Physik, J. W. Goethe-Universität
6000 Frankfurt a.M., W. Germany

KEY WORDS AND PHRASES: Laplace transform inversion, integral transformations, integral equations
CR CATEGORIES: 5.15, 5.18

Some errors have crept into the comment of the procedure after proof-reading:

The formula following "and thus" should read

$$\sum_{i=1}^{K} x_i(K)\bar{F}_{N/2+1-i} = F\left(\frac{\ln 2}{\alpha}\right) + (-1)^{K+1}\alpha_K \frac{(N/2-K)!}{(N/2)!} + o\left(\frac{(N/2-K)!}{(N/2)!}\right).$$

The formula following "with" should read

$$V_i = (-1)^{N/2+i} \sum_{k=\left[\frac{i+1}{2}\right]}^{Min(i,N/2)} \frac{k^{N/2}(2k)!}{(N/2-k)!k!(k-1)!(i-k)!(2k-i)!}.$$

The policy concerning the contributions of algorithms to *Communications of the ACM* has been revised and was published in the August 1970 issue, page 513. Copies of "Algorithm Policy / Revised August 1970" will be mailed upon request. Sept 1970 p. 573