STEEL, T. B., JR. (Ed.) *Formal Language Description Languages for Computer Programming*. North-Holland Pub. Co., Amsterdam, 1966, 330 pp.

——. A formalization of semantics for programming language description. In *Formal Language Description Languages for Computer Programming*, North-Holland Pub. Co., Amsterdam, 1966, pp. 25–36.

STRACHEY, C. A general purpose macrogenerator. *Comput. J. 8* 3 (Oct. 1965), 225–241.

——. Towards a formal semantics. In *Formal Language Description Languages for Computer Programming*, North-Holland Pub. Co., Amsterdam, 1966, pp. 198–220.

Survey of programming languages and processors *Comm. ACM 6*, 3 (Mar. 1963), 99–93.

TOBEY, R. G., BOBROW, R. J., AND ZILLES, S. N. Automatic simplification in FORMAC. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Pt. 1, Spartan Books, New York, pp. 37–57.

TURSKI, W. Some results of research on automatic programming in Eastern Europe. In *Advances in Computers, Vol. 5*, Academic Press, New York, 1964.

VAN KATWUK, A. A grammar of Dutch number names. *Foundations of Language 1*, 1 (Jan. 1965), 51–58.

VAN WIJNGAARDEN, A. Recursive definition of syntax and semantics. In *Formal Language Description Languages for Computer Programming*, North-Holland Pub. Co., Amsterdam, 1966, pp. 13–24.

VORONIN, V. A. An operational notation for the algorithm of mechanical translation. *Tr. In-Ta Tochnoi Mekhan. I Vychisl. Tekhn. An SSSR 2* (1961), pp. 70–84. (in Russian)

WALK, K. Entropy and testability of context-free languages. In *Formal Language Description Languages for Computer Programming*, North-Holland Pub. Co., Amsterdam, 1966, pp. 105–123.

WARSHALL, Stephen. A syntax directed generator. Proc. Eastern Joint Comput. Conf., Vol. 20, Dec. 1961, Spartan Books, New York, pp. 295–305.

WARSHALL, S., AND SHAPIRO, R. M. A general-purpose table-driven compiler. Proc. AFIPS 1964 Spring Joint Comput. Conf., Vol. 25, Spartan Books, New York, pp. 59–65.

WEGNER, P. An introduction to stack compilation techniques. In *Introduction to System Programming*, Wegner, Peter (Ed.), Academic Press, New York, 1964, pp. 101–121.

WEISS, ERIC. *The PL/I Converter*. McGraw-Hill, New York, 1966, 113 pp.

WEIZENBAUM, J. ELIZA—a computer program for the study of natural language communication between man and machine. *Comm. ACM 9*, 1 (Jan. 1966), 36–45.

WILKES, M. V. An experiment with a self-compiling compiler for a simple list-processing language. *Ann. Rev. Automat. Programming 4* (1964), 1–48.

——. Constraint-type statements in programming languages. *Comm. ACM 7*, (Oct. 1964), 587–588.

WIRTH, N., AND WEBER, H. EULER: a generalization of ALGOL, and its formal definition, Pt. I. *Comm. ACM 9*, 1 (Jan. 1966), 11–23; Pt. II, *Comm. ACM 9*, 2 (Feb. 1966), 89–99.

WYNN, P. A comparison technique for the numerical transformation of slowly convergent series based on the use of rational functions. *Numer. Math. 4*, 1 (Feb. 1962), 8–14.

YNGVE, V. H. COMIT. *Comm. ACM 6*, 3 (Mar. 1963), 83–84.

YUSHCHENKO, K. L. Levels and styles of address language and the problem of programming automation. Foreign Develop. Mach. Translat. Inf. Proc., JPRS 16446, Off. Tech. Serv., Washington, D. C., (Nov. 1962), pp. 11–14.

——, AND KOSTIUCHENKO, O. I. An algorithm for the translation of formulas written with the aid of parenthesis to the form of Lukashevich without parenthesis. *Zb. Prats. Z Obchisl. Mat. 1 Tekhn.* (1961), 84–89. (in Ukranian)

Algorithms

J. G. HERRIOT, Editor

ALGORITHM 344
STUDENT'S *t*-DISTRIBUTION [S14]

DAVID A. LEVINE (Recd. 26 Mar. 1968 and 2 Aug. 1968)
State University of New York at Stony Brook, Stony Brook, NY 11790

KEY WORDS AND PHRASES: Student's *t*-Distribution, *t*-test, small-sample statistics, distribution function
CR CATEGORIES: 5.12, 5.5

Comment *t*-Test evaluates in single-precision the value of Student's [2] *t*-distribution for argument T and degrees of freedom DF. The two-tailed Student's *t*-distribution, A, is obtained as the indefinite integral:

$$A(T, DF) = C \int_T^\infty \left(1 + \frac{x^2}{DF}\right)^{-\frac{DF+1}{2}} dx$$

where C is chosen so that $A\ (0, DF) = 1$.

The integration of A can be accomplished exactly by integrating by parts successively, obtaining:
for DF an odd integer,

$$A(T, DF) = 1 - \frac{2}{\pi}\left\{\arctan a + ab\left[1 + b\left(\frac{2}{3}\right) + b^2\left(\frac{2}{3}\cdot\frac{4}{5}\right) + \cdots + b^{\frac{DF-3}{2}}\left(\frac{2}{3}\cdot\frac{4}{5}\cdots\frac{DF-3}{DF-2}\right)\right]\right\},$$

and for DF an even integer,

$$A(T, DF) = 1 - a\sqrt{b}\left[1 + b\cdot\left(\frac{1}{2}\right) + b^2\left(\frac{1}{2}\cdot\frac{3}{4}\right) + \cdots + b^{\frac{DF-2}{2}}\left(\frac{1}{2}\cdot\frac{3}{4}\cdot\cdots\cdot\frac{DF-3}{DF-2}\right)\right],$$

where $a = \dfrac{T}{\sqrt{DF}}$, $b = (1 + a^2)^{-1}$.

A FORTRAN program evaluating these series is given below, giving at least six correct significant figures after the decimal—more than enough accuracy for most statistical applications. The *t*-Test is usually applied in small-sample statistics [1] where $DF \leq 30$. The algorithm presented here is faster and simpler, with accuracy equal to previous algorithms for $DF \leq 30$. In the range $30 \leq DF \leq 100$, this algorithm is competitive in speed and accuracy with previous algorithms. For the range $DF > 100$, small-sample assumptions may be altered by replacing the integrand of the distribution by a Gaussian (normal) curve; hence much greater speed is obtained in this range by employing, for example, Algorithm 209 [3]. Instructive comments and bibliography are obtainable from Algorithm 321 [4], where an algorithm competitive for the range $30 \leq DF \leq 100$ is presented and the use of Algorithm 209 is discussed.

Thanks to the referee for many helpful suggestions, which have been incorporated, and to Joan Warner, who has aided in the programming and testing of this algorithm.

REFERENCES:

1. ALDER, H. L., AND ROESSLER, E. B. Introduction to probability and statistics, 3rd ed. W. H. Freeman and Co., San Francisco, 1964, p. 125
2. GOSSET, W. S. (Student). The probable error of a mean. BIOMETRIKA *6* (1908), 1.
3. IBBETSON, D. Algorithm 209, Gauss. *Comm. ACM, 6* (Oct. 1963), 616.
4. MORRIS, J. Algorithm 321, *t*-test probabilities. *Comm. ACM 11* (Feb. 1968), 115.

```
      SUBROUTINE TTEST
C     ****************
     *   (T,DF,ANS,KERR)
C
      REAL      ANS,D1,D2,F1,F2,T,T1,T2
C
      INTEGER   DF,I,KERR,N
C
      DATA      D1/.63661977/
C
C     0.63661977236758134...= 2/ PI
C
      KERR = 0
C
      IF(DF.GT.0) GO TO 1
C
C     ERROR RETURN IF DF NOT POSITIVE
C
      KERR = 1
      ANS  = 0.
      RETURN
C
C     BEGIN COMPUTATION OF SERIES
C
    1 T    = ABS(T)
      T1   = T/SQRT(FLOAT(DF))
      T2   = 1./(1.+T1*T1)
C
      IF((DF/2)*2.EQ.DF) GO TO 5
C
C     DF IS AN ODD INTEGER
C
      ANS = 1.-D1*ATAN(T1)
C
      IF(DF.EQ.1) GO TO 4
C
      D2   = D1*T1*T2
      ANS = ANS-D2
C
      IF(DF.EQ.3) GO TO 4
C
      F1   = 0.
    2 N    = (DF-2)/2
      DO 3 I=1,N
      F2   = 2.*FLOAT(I)-F1
      D2   = D2*T2*F2/(F2+1.)
    3 ANS = ANS-D2
C
C     COMMON RETURN AFTER COMPUTATION
C
    4 IF(ANS.LT.0.) ANS = 0.
      RETURN
C
C     DF IS AN EVEN INTEGER
C
    5 D2   = T1*SQRT(T2)
      ANS = 1.-D2
C
      IF(DF.EQ.2) GO TO 4
C
      F1   = 1.
      GO TO 2
      END
```

CERTIFICATION OF ALGORITHM 165 [S21] COMPLETE ELLIPTIC INTEGRALS [Henry C. Thacher Jr., *Comm. ACM 6* (Apr. 1963), 163]

I. FARKAS (Recd. 1 Aug. 1968)
Dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada

KEY WORDS AND PHRASES: special functions, complete elliptic integral of the first kind, complete elliptic integral of the second kind
CR CATEGORIES: 5.12

One misprint and one semantic error were found in Algorithm 165:

1. The procedure heading

   procedure *KANDE* (*ml,K,E,tol,alarm*);

should read

   procedure *KANDE* (ml,*K,E,tol,alarm*);

2. The second statement in the procedure body

$$a := fact := 1;$$

should read

$$fact := 1; a := 1;$$

because *fact* and *a* are of different types.

Algorithm 165 was translated into FORTRAN IV on an IBM 7094-II, whose single-precision mantissa has 27 significant bits (about 8 significant decimal digits). Because our *SQRT* program has a relative accuracy of $.75_{10} - 8$, *tol* was chosen $3_{10} - 8$. *K* and *E* were generated for ml = (.01(.01)1.0) (to 27 bits) and the results obtained were compared with tables in [1]. For ml = .01 *E* differed by two units in the last place; for all other values of ml, the maximum absolute error was one unit in the last place. The time taken to activate *KANDE* for the above 100 values of ml was 0.1 sec.

REFERENCE:

1. ABRAMOWITZ, M., AND STEGUN, I. A. (Eds.) *Handbook of Mathematical Functions*. NBS Appl. Math. Ser. 55, US Govt. Printing Off., Washington, D. C., 1964.

REMARK ON ALGORITHM 314 [C5] FINDING A SOLUTION OF *N* FUNCTIONAL EQUATIONS IN *N* UNKNOWNS [D. B. Dulley and M. L. V. Pitteway, *Comm. ACM 10* (Nov. 1967), 726].

JAMES VANDERGRAFT AND CHARLES MESZTENYI (Recd. 12 Aug. 1968)
Computer Science Center, University of Maryland, College Park, MD 20742

KEY WORDS AND PHRASES: functional equations, interpolation, nonlinear equations, secant method
CR CATEGORIES: 5.13, 5.15

The algorithm, as published, requires four iterations to find the solution to a pair of linear equations. The difficulty seems to lie in the last statement of the first column. If this is replaced by

$$delf\ [j,i] := f[j] - prevf[j];$$

then the algorithm works well. In fact, however, it is now simply an *n*-dimensional secant method, which can be described by the iteration

$$x^{k+1} = x^k - \delta x_k (\delta F_k)^{-1} F(x^k), \quad k = 0,1,2,\ldots,$$

where δF_k and δx_k are matrices whose *i*th columns are $f(x^{k-i}) - f(x^k)$ and $x^{k-i} - x^k$, respectively. The iteration is started by setting

$$x^{-i} = x^0 + h\, e_i$$

where x^0 is a given vector, h is a small positive constant, and e_i is the ith unit coordinate vector.

It should be observed, also, that the algorithm will not break down if δx_k becomes singular. However, if this should happen it means that $x^k, x^{k-1}, \ldots, x^{k-n}$ lie in a proper subspace S of E^n, Euclidean n-space, and all successive iterates will also lie in S. Hence the algorithm may converge to a point in S which is not a solution to $f(x) = 0$. To prevent this, the norm of $f(x)$ should be checked before leaving the procedure.

Algorithms Policy • Revised August, 1966

(Includes Fortran)

A contribution to the Algorithms Department should be in the form of an algorithm, a certification, or a remark. Contributions should be sent in duplicate to the editor, typewritten double spaced. Authors should carefully follow the style of this department with especial attention to indentation and completeness of references.

An algorithm must normally be written in the ALGOL 60 Reference Language [*Comm. ACM 6* (Jan. 1963), 1-17] or in ASA Standard FORTRAN or Basic FORTRAN [*Comm.* ACM 7 (Oct. 1964), 590-625]. Consideration will be given to algorithms written in other languages provided the language has been fully documented in the open literature and provided the author presents convincing arguments that his algorithm is best described in the chosen language and cannot be adequately described in either ALGOL 60 or FORTRAN.

An algorithm written in ALGOL 60 normally consists of a commented procedure declaration. It should be typewritten double spaced in capital and lower-case letters. Material to appear in **boldface** type should be underlined in black. Blue underlining may be used to indicate italic type, but this is usually best left to the Editor. An algorithm written in FORTRAN normally consists of a commented subprogram. It should be typewritten double spaced in the form normally used for FORTRAN or it should be in the form of a listing of a FORTRAN card deck together with a copy of the card deck. Each algorithm must be accompanied by a complete driver program in its language which generates test data, calls the procedure, and produces test answers. Moreover, selected previously obtained test answers should be given in comments in either the driver program or the algorithm. The driver program may be published with the algorithm if it would be of major assistance to a user.

For ALGOL 60 programs, input and output should be achieved by procedure statements, using any of the following eleven procedures (whose body is not specified in ALGOL) [See "Report on Input-Output Procedures for ALGOL 60," *Comm. ACM 7* (Oct. 1964), 628-629]:

insymbol	*inreal*	*outarray*	*ininteger*
outsymbol	*outreal*	*outboolean*	*outinteger*
length	*inarray*	*outstring*	

If only one channel is used by the program for output, it should be designated by 1 and similarly a single input channel should be designated by 2. Examples:

outstring (1, 'x='); *outreal* (1,x);
for i := 1 **step** 1 **until** n **do** *outreal* (1,A[i]);
ininteger (2, *digit* [17]):

For FORTRAN programs, input and output should be achieved as described in the ASA preliminary report on FORTRAN and Basic FORTRAN.

It is intended that each published algorithm be well organized, clearly commented, syntactically correct, and a substantial contribution to the literature of Algorithms. It is necessary but not sufficient that a published algorithm operate on some machine and give correct answers. It must also communicate a method to the reader in a clear and unambiguous manner. All contributions will be refereed both by human beings and by an appropriate compiler. Authors should pay considerable attention to the correctness of their programs, since referees cannot be expected to debug them.

Certifications and remarks should add new information to that already published. Readers are especially encouraged to test and certify previously uncertified algorithms. Rewritten versions of previously published algorithms will be refereed as new contributions and should not be imbedded in certifications or remarks.

Galley proofs will be sent to authors; obviously rapid and careful proofreading is of paramount importance.

Although each algorithm has been tested by its author, no liability is assumed by the contributor, the editor, or the Association for Computing Machinery in connection therewith.

The reproduction of algorithms appearing in this department is explicitly permitted without any charge. When reproduction is for publication purposes, reference must be made to the algorithm author and to the *Communications* issue bearing the algorithm.—J.G.Herriot

CERTIFICATION OF ALGORITHM 322 [S14] F-DISTRIBUTION [Egon Dorrer, *Comm. ACM 11* (Feb. 1968), 116]

J. B. F. Field (Recd. 15 Aug. 1968)
Commonwealth Scientific and Industrial Research Organisation, Adelaide, South Australia

KEY WORDS AND PHRASES: Fisher's F-distribution, Student's t-distribution
CR CATEGORIES: 5.5

Algorithm 322 was coded into Fortran and run on a CDC 3200, and its accuracy for moderate probability levels was tested using (a) 5-figure critical values of the F-distribution at the .95 and .99 levels, taken from [1], and (b) 6-figure probability values of the t-distribution, taken from [2]. In both cases, limitations in the results appeared to be due to limitations in the tables, rather than in the algorithm.

232 values of the F-distribution were tested, for $m = 1$ and 12 using all tabulated values of n, and for $n = 10$ and 21 using all tabulated values of m. All the results agreed with the tabulated probability level to 4 significant figures, 89% to 5 figures, and over half the results agreed to 6 or more figures.

300 values of the t-distribution were tested, for $n = 1(1)30$ and $t = .5(.5)5$. All the results agreed with the tabulated probability to 5 significant figures, and 90% to the full 6 figures given in the tables.

To test extreme probability levels, another 100 values of the F-distribution were used: for $m = n = 2$, 10, 50, 75, 100, 120, 150, 200, 300, and 400 for each of the values $x = 10^i$, where $i = 5(1)5$. It was found that for probabilities which are extremely close to 0 or 1, the algorithm may produce probabilities which are slightly less than zero, or slightly greater than 1. It is recommended that a "guard" be inserted in the program to set these values equal to 0 or 1. For example, this could be done by inserting before *Fisher*:=p the additional statement

$$p := \textbf{if } p > 1 \textbf{ then } 1 \textbf{ else if } p < 0 \textbf{ then } 0 \textbf{ else } p;$$

The time taken by the algorithm was directly proportional to the sum of the degrees of freedom. The constant of proportionality depended mainly on whether m was even or odd (the time taken for m even being .81 of the time taken for m odd, using a CDC 3200 with programmed floating point). To a much lesser extent, it was influenced by whether n was even or odd (the time taken for n even being .99 of that for n odd).

REFERENCES

1. Owen, D. B. *Handbook of Statistical Tables.* Addison-Wesley, Reading, Mass., 1962.
2. Smirnov, N. V. *Tables for the Distribution and Density Functions of t-distribution.* Pergamon Press, Oxford, 1961.

REMARK ON ALGORITHM 337 [C1] CALCULATION OF A POLYNOMIAL AND ITS DERIVATIVE VALUES BY HORNER SCHEME [W. Pankiewicz, *Comm. ACM 11* (Sept. 1968), 633]

Oliver K. Smith (Recd. 27 Sept. 1968)
Applied Mathematics Dept., Systems Group of TRW, Inc., 1 Space Park, Redondo Beach, CA 90278

KEY WORDS AND PHRASES: function evaluation, polynomial evaluation, Algol procedure, Horner's scheme
CR CATEGORIES: 4.22, 5.12

The definition of the given polynomial is incorrect in the comment. In both the third line and the eighth line of the comment, $a[j]$ should be replaced by $a[n - j]$. Also the first word "and" of the fourth line of the comment should be changed to "at".