```
      1 A = ALPHA
        PP = P
        QQ = Q
        INDEX = .FALSE.
C
C          CALCULATE THE INITIAL APPROXIMATION.
C
      2 R = SQRT(-ALOG(A * A))
        Y = R - (2.30753 + 0.27061 * R)/(1.0 + (0.99229 + 0.04481 * R)* R)
        R = 2.0 * QQ
        T = 1.0 / (9.0 * QQ)
        T = R * (1.0 - T + Y * SQRT(T)) ** 3
        IF (T.LE.0.0) GOTO 3
        T = (4.0 * PP + R - 2.0) / T
        IF (T.LE.1.0) GOTO 4
        XINBTA = 1.0 - 2.0 / (T + 1.0)
        GOTO 5
      3 XINBTA = 1.0 - EXP(ALOG((1.0 - A) * QQ * BETA) / QQ)
        GOTO 5
      4 XINBTA = EXP(ALOG(A * PP * BETA) / PP)
C
C          SOLVE FOR X BY THE NEWTON-RAPHSON METHOD,
C          USING THE FUNCTION BETAIN.
C
      5 R = 1.0 - PP
        T = 1.0 - QQ
      6 Y = BETAIN(XINBTA, PP, QQ, BETA, IFAULT)
        IF (IFAULT.EQ.0) GOTO 7
        IFAULT = 3
        RETURN
      7 Y = (Y - A) * BETA * EXP(R * ALOG(XINBTA) + T * ALOG(1.0-XINBTA))
        XINBTA = XINBTA - Y
        IF (ABS(Y).GT.ACU) GOTO 6
        IF (INDEX) XINBTA = 1.0 - XINBTA
        RETURN
        END
```

Algorithm AS 65

Interpreting Structure Formulae

By C. E. Rogers

Rothamsted Experimental Station, Harpenden, Herts

Keywords: STRUCTURE FORMULAE; FACTORIAL MODELS FOR ANOVA; 1—1 PRECEDENCE GRAMMAR

Language

ISO Fortran

Purpose

The structure formulae described by Wilkinson and Rogers (1973) provide a convenient, compact method of specifying simple factorial models for analysis of variance or constant-fitting algorithms.

Structure formulae resemble ordinary algebraic expressions with factors as the basic operands and the following operators:

Symbol	*Operator*	*Precedence*
.	compounding	1 (highest)
/	nesting	2
*	crossing	3
+	adding	4
−, −*, −/	deletion	4

Brackets are used to alter the natural precedence in formulae. Any structure formula may be expanded into an equivalent elementary form consisting of a sum of model terms. The elementary form, which is unique except for the order in which the model terms and their component factors are written, provides a representation of the model which may be used to control the analysis.

An elementary form may be represented by a list of positive binary integers by assigning to each factor in the model a different power of 2 and representing each model term by the sum of the integers corresponding to its component factors, e.g. if factors A, B, C are represented by 1, 2 and 4, i.e. 1, 10 and 100 in binary:

Structure formula	*Elementary form binary representation*
$A*B*C$	$A+B+C+A.B+A.C+B.C+A.B.C$ 1 10 100 11 101 110 111
$A*B/C$	$A+B+A.B+B.C+A.B.C$ 1 10 11 110 111
$(A*B)/C$	$A+B+A.B+A.B.C$ 1 10 11 111

The subroutine *SFINT* transforms any structure formula into the corresponding list of binary integers.

Description

The structure formula is encoded as a list of integers for input to *SFINT*. The integer coding of a factor determines its representation as a binary integer. Options specify the ordering of the output binary integers and the maximum number of 1-digits in any integer (i.e. high-order interactions may be deleted from the model).

The syntax of structure formulae may be defined formally so that it belongs to the 1–1 precedence class (Floyd, 1963; Wirth and Weber, 1966). Following Claringbold (1969) the production rules may be written:

Production	*Left part*	*Right part*	*Semantics*
1	$K(1)$	$K(0) \equiv$ Factor	Load
2	$K(i)$ $i = 2, 3, \ldots, 2n+1$	$K(i-1)$	Copy
3	$K(1)$	$(\ K(2n+1)\ )$	Copy
4	$K(i)$ $i = 2, 4, \ldots, 2n$	$K(i)\, O(i)\, K(i-1)$	Operate
5	$K(2n+2)$ $\equiv$ Expression	Limit $K(2n+1)$ Limit	Edit

where n is the number of precedence classes, i.e. 4. Note that Claringbold's Operand

and Head symbols correspond to the K-symbols with odd and even indices respectively and his operator symbol $*(i)$ corresponds to $O(2i)$.

The auxiliary precedence function $IPREC$:

Symbol = $O(2i)$	$K(i)$	(	)	Limit
$IPREC = 2i$	i	$2n+2$	$2n+1$	$2n+3$

allows the $3(n+1) \times 3(n+1)$ table of precedence relations to be written as a 5×5 table with two extra rules to be used when the cell of the table contains 1 or 2:

		Right symbol			
Left symbol	K	)	(	0	Limit
K	.	2	.	2	>
)	.	>	.	>	>
(	1	.	<	.	.
0	1	.	<	.	.
Limit	<	.	<	.	.

If $l = IPREC$ (left symbol), $r = IPREC$ (right symbol)

Rule		*Relation*
1	$l-1 < r$	.
	$l-1 = r$	=
	$l-1 > r$	<
2	$l < r$	>
	$l = r$	=
	$l > r$	.

A simple parsing algorithm is used which scans the input string in *ICODE* from left to right and maintains a symbol stack. Possible right parts are detected on top of the symbol stack as strings of symbols related by the = (concatenation) relation and delimited by < (opening) and > (closing) relations. Syntactical errors are revealed by the relation . or by the failure to match a purported right part with the above list of productions.

The function $IPREC$ is also used to short-circuit a chain of productions by allowing "promotion" of the left part K-symbol until it concatenates (i.e. has the = relation) with a neighbouring symbol on the stack.

When a production is identified the syntactical action is to replace the right part symbol(s) by the (promoted) left part K-symbol. The corresponding semantic action is to replace the value(s) of the right part K-symbol(s) (which are held on a separate values stack) according to the semantic rules.

A K-symbol has the value (SUM, FAC) where SUM = list of binary integers and FAC = special binary integer, used only in the / operation. Two dyadic operations on lists are needed (+ and .) together with the idea of marginality:

If $SUM1$ and $SUM2$ are lists of binary integers then

$SUM1 + SUM2$ = list formed by concatenating lists $SUM1$ and $SUM2$

$SUM1$. $SUM2$ = list of all pairwise logical sums of elements of $SUM1$ and $SUM2$.

Integer $I1$ is a margin of $I2$ if $I2$ has a 1 digit in every position where $I1$ has a 1 digit. $I1$ is a strict margin of $I2$ if $I1$ is a margin of $I2$ but not equal to $I2$.

If L and R are the left-most and right-most K-symbols in the right part then the semantic rules may be written:

Rule	*Left part*		
	FAC	*SUM*	
Load	$2^{(i-1)}$	$2^{(i-1)}$	where $-i$ = code for factor in *ICODE* (the input coded formula)
Copy	*FAC*(*L*)	*SUM*(*L*)	
Edit	*FAC*(*L*)	*SUM*(*L*)	with duplicate integers deleted and the rest reordered as specified by *IORDER*
Operate			
.	*FAC*(*L*).*FAC*(*R*)	*SUM*(*L*).*SUM*(*R*)	
/	*FAC*(*L*).*FAC*(*R*)	*SUM*(*L*)+(*FAC*(*L*).*SUM*(*R*))	
*	*FAC*(*L*).*FAC*(*R*)	*SUM*(*L*)+*SUM*(*R*)+(*SUM*(*L*).*SUM*(*R*))	
+	*FAC*(*L*).*FAC*(*R*)	*SUM*(*L*)+*SUM*(*R*)	
−	*FAC*(*L*)	*SUM*(*L*)	deleting integers also in *SUM*(*R*)
−/	*FAC*(*L*)	*SUM*(*L*)	deleting integers that have a strict margin in *SUM*(*R*)
−*	*FAC*(*L*)	*SUM*(*L*)	deleting integers that have a margin in *SUM*(*R*)

To facilitate amendments and adaptations *SFINT* is written so that the sections dealing with syntax and the semantics are kept as independent as possible.

Structure

SUBROUTINE SFINT (*ICODE*, *LCODE*, *IORDER*, *LIMIT*, *MAXLBP*, *LISTBP*, *LENBP*, *IFAULT*)

Formal parameters

ICODE	Integer array (*LCODE*)	input: the coded formula
LCODE	Integer	input: the number of symbols in the formula
IORDER	Integer	input: the output integers in *LISTBP* are arranged in increasing numerical order if *IORDER* $\geqslant 0$ otherwise they are first arranged into groups by increasing number of 1-digits and then arranged in increasing numerical order within each group
LIMIT	Integer	input: if *LIMIT* > 0 then high-order terms comprising more than *LIMIT* factors are deleted from the model
MAXLBP	Integer	input: the dimension of *LISTBP*
LISTBP	Integer array (*MAXLBP*)	output: the list of binary integers
LENBP	Integer	output: if *IFAULT* = 0 then *LENBP* = number of integers in *LISTBP* else *LENBP* = number of symbols read from *ICODE* before a fault occurred.
IFAULT	Integer	output: see *Failure indications* below.

Input code for ICODE

Symbol	factor	.	/	*	+	−	−/	−*	(	)
Code	$-i\,(i>0)$	1	2	3	4	5	6	7	8	9

A different integer $-i$ is assigned to each factor in the model. The integer representation of the factor is 2^{i-1}. The factor coding need not use consecutive powers of 2 nor start at 2^0. The pre-scan of the formula to create the coded form in *ICODE* may be easily incorporated in *SFINT*.

Failure indications

If *IFAULT* = 0 no failure occurred
If *IFAULT* = 1 then *LCODE* < 0
If *IFAULT* = 2 then not enough space in *LISTBP*
If *IFAULT* = 3 then factor code $i >$ *NBI* (cf. restrictions)
If *IFAULT* = 4 then integer in *ICODE* not in above list
If *IFAULT* = 5 then syntax error detected in the formula

NOTE: Only syntactic errors are detected by *SFINT*, e.g. the formulae

$$A*A*B \quad \text{and} \quad A*B-C$$

are legal although they contain redundant terms (both are equivalent to the simpler formula $A*B$). There are no checks that a formula represents a statistically sensible model.

Auxiliary Algorithms

The subroutine *BISORT*(*LISTBP, II, JJ, IORDER*) sorts the elements *LISTBP* $(II+1, \ldots, JJ)$ of the array *LISTBP* according to the switch *IORDER* (which has same specification as *IORDER* in *SFINT*).

Two functions to allow bit-manipulation are needed. All variables and functions are integers:

ICNT(*I*) = sum of binary digits of $I \quad (I \geqslant 0)$
IOR(*I, J*) = logical sum of the binary integers I and $J \quad (I \geqslant 0, J \geqslant 0)$.

Restrictions

To ensure that all generated binary integers are legal Fortran integers a factor coded as $-i$ must have $i \leqslant NBI$ where *NBI*, a local constant in *SFINT*, has been chosen as large as possible consistent with $2^{NBI}-1$ being a legal positive integer.

Work-space

The array *LISTBP* is used as work-space during expansion of a formula. The amount of storage required depends on the formula being expanded. An upper bound is given by

$$6+3\times LCODE+2^N,$$

where N = number of occurrences of factors in the formula, i.e. number of negative integers in *ICODE*.

REFERENCES

CLARINGBOLD, P. J. (1969). An approach to conversational statistics. In *Statistical Computation* (R. C. Milton and J. A. Nelder, eds), pp. 267–283. New York: Academic Press.

FLOYD, R. W. (1963). Syntactic analysis and operator precedence. *J. Ass. Comput. Mach.*, **10**, 316–333.

SHELL, D. E. (1959). A high-speed sorting procedure. *Commun. Ass. Comput. Mach.*, **2**, 30–32.

WILKINSON, G. N. (1970). A general recursive procedure for analysis of variance. *Biometrika*, **57**, 19–46.

WILKINSON, G. N. and ROGERS, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Appl. Statist.*, **22**, 392–399.

WIRTH, N. and WEBER, H. (1966). EULER—a generalization of ALGOL and its formal definition. *Commun. Ass. Comput. Mach.*, **9**, 13–25, 89–99.

```
      SUBROUTINE SFINT(ICODE, LCODE, IORDER, LIMIT, MAXLBP, LISTBP,
     *  LENBP, IFAULT)
C
C         ALGORITHM AS 65  APPL.STATIST. (1973), VOL.22, NO.3
C
C         EXPANDS A STRUCTURE FORMULA INTO A LIST OF BINARY INTEGERS
C
C     LITERAL =     .  /  *  +  - -/ **   (  )    FACTOR
C     SYMBOL  =    02 04 06 08 08 08 08   (  )    KO K1 ... K10   L
C     CODE    =     1  2  3  4  5  6  7   8  9    10 11 ...  20  21
C     IENTRY  =     4  4  4  4  4  4  4   3  2     1  1 ...   1   5
C     IPREC   =     2  4  6  8  8  8  8  10  9     0  1 ...  10  11
C
      DIMENSION ICODE(LCODE), LISTBP(MAXLBP)
      DIMENSION IPRTAB(5, 5), IENTRY(21), IPREC(21)
C
C*****    SET NBI SO THAT 2**NBI-1 IS LARGE LEGAL POSITIVE INTEGER
C
      DATA NBI /31/
C
C         INITIALISE COMPACT PRECEDENCE TABLE + AUXILIARY QUANTITIES
C
      DATA KILLEG,KOPEN,KCON,KCLOSE/0,1,2,3/
      DATA IPRTAB(1,1),IPRTAB(2,1),IPRTAB(3,1),IPRTAB(4,1),IPRTAB(5,1),
     2     IPRTAB(1,2),IPRTAB(2,2),IPRTAB(3,2),IPRTAB(4,2),IPRTAB(5,2),
     3     IPRTAB(1,3),IPRTAB(2,3),IPRTAB(3,3),IPRTAB(4,3),IPRTAB(5,3),
     4     IPRTAB(1,4),IPRTAB(2,4),IPRTAB(3,4),IPRTAB(4,4),IPRTAB(5,4),
     5     IPRTAB(1,5),IPRTAB(2,5),IPRTAB(3,5),IPRTAB(4,5),IPRTAB(5,5)
     6     /2*0,-1,-1,1,'  -3,3,3*0,   2*0,3*1,   -3,3,3*0,   2*3,3*0/
      DATA IENTRY( 1),IENTRY( 2),IENTRY( 3),IENTRY( 4),IENTRY( 5),
     2     IENTRY( 6),IENTRY( 7),IENTRY( 8),IENTRY( 9),IENTRY(10),
     3     IENTRY(11),IENTRY(12),IENTRY(13),IENTRY(14),IENTRY(15),
     4     IENTRY(16),IENTRY(17),IENTRY(18),IENTRY(19),IENTRY(20),
     5     IENTRY(21) /7 * 4, 3, 2, 11 * 1, 5/
      DATA IPREC( 1),IPREC( 2),IPREC( 3),IPREC( 4),IPREC( 5),IPREC( 6),
     2     IPREC( 7),IPREC( 8),IPREC( 9),IPREC(10),IPREC(11),IPREC(12),
     3     IPREC(13),IPREC(14),IPREC(15),IPREC(16),IPREC(17),IPREC(18),
     4     IPREC(19),IPREC(20),IPREC(21)
     5     /2, 4, 6, 4*8, 10, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11/
C
C---------------- INITIALISATION ----------------------------------
C         IORD=HIGHEST ORDER OF RETAINED TERMS
C         JC=POINTER TO NEXT INPUT SYMBOL IN ICODE
C         JB=POINTER TO TOP OF SUM STACK LISTBP(1...)
C         JP=POINTER TO TOP OF SYMBOL STACK LISTBP(...MAXLBP) (TOP-DOWN)
C         CONTAINING 3-INTEGER CELLS (SYMBOL CODE,POINTER TO SUM,FAC)
```

```
      IFAULT = 0
      LENBP = 0
      IORD = LIMIT
      IF (IORD.LE.0) IORD = NBI
      JC = 0
      IF (LCODE) 1001, 999, 10
   10 JB = 0
      JP = MAXLBP + 1
C
C------------------ SYNTAX ANALYSER ----------------------------------
C-------- STACK LIMITSYMBOL L AND/OR NEXT ICODE SYMBOL
C         STACK =INPUT CODE FOR FACTOR
C
  100 I = 21
      IF (JC.LT.1.OR.JC.GE.LCODE) GOTO 115
  110 JC = JC + 1
      I = ICODE(JC)
      IF (I.EQ.0.OR.I.GT.9) GOTO 1004
  115 JP = JP - 3
      IF (JB.GE.JP) GOTO 1002
      LISTBP(JP + 2) = -I
      IF (I.LT.0) I = 10
      LISTBP(JP) = I
      IF (JC.EQ.0) GOTO 110
C
C-------- SEARCH FOR PRODUCTION
C         IF CLOSING RELATION BETWEEN TOP TWO STACKED SYMBOLS SCAN FOR
C         MATCHING OPENING RELATION TO DELIMIT POSSIBLE RIGHTPART
C
  120 IP = JP
      IBS = 0
  130 IP = IP + 3
C
C         IREL=RELATION BETWEEN LEFTSYMBOL AT IP AND RIGHTSYMBOL AT IP-3
C
      ILS = LISTBP(IP)
      IRS = LISTBP(IP - 3)
      I = IENTRY(ILS)
      J = IENTRY(IRS)
      IREL = IPRTAB(I, J)
      IF (IREL.GE.0) GOTO 140
      IREL = -IREL
      M = IPREC(ILS) - IPREC(IRS)
      IF (IREL.EQ.KOPEN) M = 1 - M
      IF (M.EQ.0) IREL = KCON
      IF (M.GT.0) IREL = KILLEG
C
C         BRANCH ON IBS=0/1 FOR TERMINAL/EARLIER RELATION
C
  140 IF (IBS.EQ.1) GOTO 170
C
C         TERMINAL RELATION
C
      IF (IREL.EQ.KILLEG) GOTO 1005
      IBS = 1
      IF (IREL - KCLOSE) 100, 130, 100
C
C         EARLIER RELATION
C
  170 IF (IREL.NE.KOPEN) GOTO 130
C
C-------- IDENTIFY PRODUCTION
C         LEFTPART  RIGHTPART              SEMANTIC ACTION   IPRON
C         K1        KO                     LOAD              1
C         K1        ( K9 )                 COPY              2
C         KI        KI OI KI-1 I=2,4,6,8   DYADIC OPERATION  3
C         KI        KI-1       I=1,2...9         LAST TWO PRODUCTIONS
C         K10       L K9 L                       ARE SHORTCIRCUITED
```

```
C          IDENTIFY PRODUCTION, IF ANY, WITH RIGHTPART MATCHING THE NS
C          STACKED SYMBOLS BETWEEN THE OPENING AND CLOSING RELATIONS
C          COPY TERMINAL + RIGHTPART CELLS
C
      NS = (IP - JP - 3) /3
      IF (NS.NE.1.AND.NS.NE.3) GOTO 1005
      JTSYM = LISTBP(JP)
      JTSUM = LISTBP(JP + 1)
      JTFAC = LISTBP(JP + 2)
      JRSYM = LISTBP(JP + 3)
      JRSUM = LISTBP(JP + 4)
      JRFAC = LISTBP(JP + 5)
      IPRONO = 4
      IF (NS.EQ.3) GOTO 180
      IF (JRSYM.EQ.10) IPRONO = 1
      GOTO 190
  180 JMSYM = LISTBP(JP + 6)
      JMSUM = LISTBP(JP + 7)
      JMFAC = LISTBP(JP + 8)
      JLSYM = LISTBP(JP + 9)
      JLSUM = LISTBP(JP + 10)
      JLFAC = LISTBP(JP + 11)
      IF (JMSYM.EQ.19.AND.JLSYM.EQ.8.AND.JRSYM.EQ.9) IPRONO = 2
      I = IPREC(JMSYM)
      IF (IENTRY(JMSYM).EQ.4.AND.JLSYM.EQ.10 + I.AND.JRSYM.EQ.9 + I)
     *  IPRONO = 3
C
C          BRANCH TO SEMANTICS SECTION
C
  190 GOTO (500, 600, 700, 1005), IPRONO
C
C-------- UPDATE STACK
C          SET JRSYM=K-SYMBOL IN LEFTPART. ANTICIPATE AND SHORTCIRCUIT
C          CHAIN OF 1-SYMBOL PRODUCTIONS BY PROMOTING JRSYM UNTIL IT
C          CONCATENATES WITH A STACKED NEIGHBOURING SYMBOL
C
  200 I = LISTBP(IP)
      J = LISTBP(JP)
      JRSYM = 10 + MINO(IPREC(I) - 1, IPREC(J))
C
C          IF LEFTPART=K10 (FORMULA) SET LENBP AND RETURN
C          ELSE STACK TERMINAL + LEFTPART CELLS
C
      IF (JRSYM.LT.20) GOTO 210
      LENBP = JB
      GOTO 999
  210 JP = IP - 6
      LISTBP(JP) = JTSYM
      LISTBP(JP + 1) = JTSUM
      LISTBP(JP + 2) = JTFAC
      LISTBP(JP + 3) = JRSYM
      LISTBP(JP + 4) = JRSUM
      LISTBP(JP + 5) = JRFAC
      GOTO 120
C
C----------------- SEMANTICS ----------------------------------------
C-------- LOAD
C
  500 IF (JRFAC.GT.NBI) GOTO 1003
      JRFAC = 2 ** (JRFAC - 1)
      JRSUM = JB
      JB = JB + 1
      IF (JB.GE.JP) GOTO 1002
      LISTBP(JB) = JRFAC
      GOTO 200
C
C-------- COPY
```

```
  600 JRSYM = JMSYM
      JRSUM = JMSUM
      JRFAC = JMFAC
      GOTO 200
C
C-------- DYADIC OPERATION
C
  700 JOP = JMSYM
      IJB = JB
C
C           .    /    *    +    -    -/   -*
      GOTO (730, 710, 730, 775, 720, 720, 720), JOP
  710 JB = JRSUM
      IBP = JLFAC
      I = JRSUM
      GOTO 740
  720 JB = JLSUM
C
C         OUTER LOOP OVER LEFT OPERAND
C
  730 I = JLSUM
  735 I = I + 1
      IF (I.GT.JRSUM) GOTO 770
      IBP = LISTBP(I)
C
C         INNER LOOP OVER RIGHT OPERAND
C
  740 J = JRSUM
  745 J = J + 1
      IF (J.GT.IJB) GOTO 760
      JBP = LISTBP(J)
      IORJ = IOR(IBP, JBP)
      IF (JOP.GT.4) GOTO 750
C
C         . / AND * ONLY
C
      IF (ICNT(IORJ).GT.IORD) GOTO 745
      JB = JB + 1
      IF (JB.GE.JP) GOTO 1002
      LISTBP(JB) = IORJ
      GOTO 745
C
C         -  -/ AND -* ONLY
C
  750 IF (IORJ.NE.IBP) GOTO 745
      K = 5
      IF (IBP.EQ.JBP) K = 6
      IF (JOP - K) 735, 745, 735
C
C         END OF INNER LOOP
C         COPY INTEGER NOT DELETED IN *  -/ OR -* OPERATION
C
  760 IF (JOP.LT.5) GOTO 735
      JB = JB + 1
      LISTBP(JB) = IBP
      GOTO 735
C
C         END OF OUTER LOOP
C         FOR .  /  * AND + REORDER SUM AND COPY TO JLSUM OMITTING REPEATS
C
  770 IF (JOP.GT.4) GOTO 790
  775 IF (JOP.GT.1) IJB = JLSUM
      CALL BISORT(LISTBP, IJB, JB, IORDER)
      J = JB
      JB = JLSUM
      IBP = -1
  780 IJB = IJB + 1
```

```
      IF (IJB.GT.J) GOTO 790
      IF (LISTBP(IJB).EQ.IBP) GOTO 780
      IBP = LISTBP(IJB)
      JB = JB + 1
      LISTBP(JB) = IBP
      GOTO 780
C
C         SET JRSUM, JRFAC
C
  790 JRSUM = JLSUM
      JRFAC = IOR(JLFAC, JRFAC)
      IF (JOP.GT.4) JRFAC = JLFAC
      GOTO 200
C
C----------------- RETURN AND ERROR SETTING -------------------------
C
 1005 IFAULT = IFAULT + 1
 1004 IFAULT = IFAULT + 1
 1003 IFAULT = IFAULT + 1
 1002 IFAULT = IFAULT + 1
 1001 IFAULT = IFAULT + 1
      LENBP = JC
  999 RETURN
      END



      SUBROUTINE BISORT(LISTBP, II, JJ, IORDER)
C
C        ALGORITHM AS 65.1  APPL.STATIST. (1973), VOL.22, NO.3
C
C        SORTS LISTBP(II+1)...LISTBP(JJ) WITH GROUPING IF IORDER.LT.0
C        BASED ON THE PROCEDURE OF SHELL(1959)
C
      DIMENSION LISTBP(JJ)
      N = JJ - II
      M = N
  810 M = M / 2
      IF (M.LT.1) RETURN
      K = N - M
      DO 850 J = 1, K
      I = II + J
  820 IM = I + M
      L1 = LISTBP(I)
      L2 = LISTBP(IM)
      IF (IORDER.GE.0) GOTO 830
      IF (ICNT(L1) - ICNT(L2)) 850, 830, 840
  830 IF (L1.LE.L2) GOTO 850
  840 LISTBP(I) = L2
      LISTBP(IM) = L1
      I = I - M
      IF (I.GT.II) GOTO 820
  850 CONTINUE
      GOTO 810
      END



      FUNCTION IOR(I1, I2)
C
C        ALGORITHM AS 65.2  APPL.STATIST. (1973), VOL.22, NO.3
C
C        RESULT IS LOGICAL SUM OF NON-NEGATIVE BINARY INTEGERS I1 AND
C        I2. THIS FUNCTION CAN BE CODED MUCH MORE EFFICIENTLY IN
C        ASSEMBLY CODE.
```

```
      J1 = I1
      J2 = I2
      IOR = 0
      M = 1
   10 L1 = J1 / 2
      L2 = J2 / 2
      IF(J1.NE.L1 * 2.OR.J2.NE.L2 * 2) IOR = IOR + M
      IF(L1.EQ.0.AND.L2.EQ.0) RETURN
      J1 = L1
      J2 = L2
      M = M + M
      GOTO 10
      END



      FUNCTION ICNT(I)
C
C        ALGORITHM AS 65.3  APPL.STATIST. (1973), VOL.22, NO.3
C
C        RESULT IS SUM OF DIGITS IN NON-NEGATIVE BINARY INTEGER I. THIS
C        FUNCTION CAN BE CODED MUCH MORE EFFICIENTLY IN ASSEMBLY CODE.
C
      J = I
      ICNT = 0
   10 L = J / 2
      IF(J.NE.L * 2) ICNT = ICNT + 1
      IF(L.EQ.0) RETURN
      J = L
      GOTO 10
      END
```

Algorithm AS 66

The Normal Integral

By I. D. Hill

Medical Research Council, London

Keywords: NORMAL CURVE; TAIL AREA

Language

ISO Fortran

Description and Purpose

Calculates the upper, or lower, tail area of the standardized normal curve corresponding to any given argument.

Numerical Method

The method is that of Adams (1969), but incorporated in a different surrounding structure.

Structure

FUNCTION ALNORM(*X*, *UPPER*)