CR-171 643

AgRISTARS

SR-X1-04033
NAS9-15981

E83-10309

A Joint Program for Agriculture and Resources Inventory Surveys Through Aerospace Remote Sensing

Supporting Research

January 1981

ASSEMBLY LANGUAGE CODING FOR CLASSY

(E83-10309) ASSEMBLY LANGUAGE CODING FOR CLASSY (Elogic, Inc.) 36 p HC A03/MF A01 CSCL 02C

N83-27299

Unclas
G3/43 00309

M. E. Rassbach

Elogic, Inc.
4242 S.W. Freeway, Suite 304
Houston, Texas 77027

Lyndon B. Johnson Space Center
Houston, Texas 77058

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
SR-X1-04033		

4. Title and Subtitle	5. Report Date
Assembly Language Coding for CLASSY	
	6. Performing Organization Code
7. Author(s) M. E. Rassbach	8. Performing Organization Report No. NAS811
	10. Work Unit No.
9. Performing Organization Name and Address Elogic, Inc. 4242 S.W. Freeway, Suite 304 Houston, TX 77027	11. Contract or Grant No. NAS9-15981
	13. Type of Report and Period Covered Technical Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Lyndon B. Johnson Space Center Houston, TX 77058	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A set of assembly language improvements to the CLASSY clustering algorithm have been developed and tested. These are descrived in detail and analyzed.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
CLASSY algorithm	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	35	

*For sale by the National Technical Information Service, Springfield, Virginia 22161

SR-X1-04033
NAS9-15981

ASSEMBLY LANGUAGE CODING FOR CLASSY

BY

M. E. RASSBACH

This report describes Classification activities of the Supporting Research project of the AgRISTARS program.

Elogic, inc.
4242 S.W. Freeway, Suite 304
Houston, Texas, 77027

January 20, 1981

CONTENTS

Assembly language coding for CLASSY

Elogic has developed several assembly language routines to speed up the CLASSY algorithm.

These fall into two catagories

1) matrix arithmetic

2) improvement in the exponential routine

<u>Purpose of assembly-coded Matrix Arithmetic and Exponential Routines</u>.

Currently available compilers produce a great number of extra instructions, particularly "overhead" instructions such as Load, Store, and index arithmetic. These instructions can be eliminated by careful hand coding of the routines. Thus it is possible to reduce the machine-processing time for a program by coding the most frequently used routines in assembly language.

CLASSY is particularly suited for this form of optimization, since most of its execution time is spent in a few vector and matrix processing routines. In addition, as was noted while designing the vector processing routines, application of the distributive law will allow putting one multiplication outside of the inner loop in several instances. (It was later realized that this change could be made in the Fortran version, as well.) Thus by using new, assembly language versions of the matrix arithmetic routines, CLASSY could be greatly accelerated. Some results are given in Table 1.

Matrix Arithmetic

The matrix arithmetic routines correspond exactly (bit-by-bit) to Fortran versions ("new Fortran") of the old matrix routines. The new Fortran versions differ from the original Fortran versions ("old Fortran") by the application of the distributive law and precalculation of some quantities to reduce the number of operations inside the inner loop. Rounding error differences can make the new version differ from the old version, which may become substantial due to the characteristics of the CLASSY algorithm (See "Path Differences in CLASSY" below). However, since these differences are only due to floating-point rounding error, which is random in both the new and old versions, the results are equivalent. The new version may in fact give slightly higher precision, due to the method of summation used. Table 2 gives the routine names for the new and old Fortran versions, as well as the assembly version. The names of EXEC'S used to switch from one version to another are also given. The names given are the names of the source language files; the object programs must have the original name.

Version	Routines (Vector, XP)	Iterations, Channels		Time (Improvement)	Time per Cluster per Iteration (Improvement)
Original	Old Fort., Fort. XP	10	4	369	4.10
Old Fort.	Old Fort., ALC XP	10	4	518 (0.71)	3.99 (1.03)
New Fort.	New Fort., ALC XP	10	4	301 (1.23)	3.34 (1.23)
ALC	ALC, ALC XP	10	4	240 (1.54)	2.67 (1.54)
Original	Old Fort., Fort XP	7	8	1590	15.14
Old Fort.	Old Fort., ALC XP	7	8	1164 (1.37)	18.47 (0.82)
New Fort.	New Fort., ALC XP	7	8	927 (1.72)	9.46 (1.60)
ALC	ALC, ALC XP	7	8	641 (2.48)	6.54 (2.31)

Table 1: Run time comparison.

This table shows a comparison of the run times of CLASSY using different combinations of routines. The run was 10 iterations on 4 channels of the simulated data, without any map generation. Initialization and randomization time has been subtracted off. Improvement is the speed ratio relative to the corresponding original version. (ALC means Assembly Language Coding, Fort. means Fortram.)

All runs were done on a simulated data set using a 4900 point data set. The scatter in the results is due to the effects of path differences (see heading on these).

Table 2: Routine names for matrix arithmetic.

Routine	Old Fortran	New Fortran	Assembly
DOTSQ	DOTSQB	DOTSQ	DOTSQA
CORECT	CORECTB	CORECT	CORECTA
MPVS	MPVSB	MPVS	MPVSA
VPV	VPV	VPV	VPVA
VMTV	VMTV	VMTV	VMTVA
EXEC to compile	VERSOLD	VERSNEW	VERSALC

New Exponential routine (XP)

In order to speed up executions, especially when the number of channels is few, Elogic has written a special exponential routine, called XP, specialized to the needs of the CLASSY algorithm. In CLASSY, the exponential is calculated as often as is the primary quadratic form, and the standard exponential routine requires several multiplies and a divide. The new exponential used for CLASSY reduces these times by table look up and by allowing a small increase in the RMS error. The new routine has an RMS precision of about 1 part in 24,000, which is easily adequate for the needs of CLASSY. In addition, the new routine uses linear interpolation and requires only one multiply.

The new exponential routine calculates XP (x) = exp (-x/2) by subdividing the interval between 1 and 16 into 240 equal parts. A table contains the coefficients of the least-squares fit of a linear form (First - order (polynomial) to the desired exponential in this region. These coefficients give a result accurate to about one part in 24,000 RMS in the result. Arguments outside the range of 1 to 16 are handled by directly obtaining the exponential of the remaining part and multiplying it times the part calculated above. Note that because of limitations on the exponent range of the result to the routine, arguments are limited to lie between 2 log $10^{\pm 76}$. This routine will not raise an exponent underflow condition if the result is too small; rather, it returns a result of zero. This saves additional time since STATIS will not be required to check the argument range before calculating the exponential.

The table used XP is currently calculated at execution time by the routine XPREP which must be called by CLINIT during initialization. (A version of CLINIT making this call must be used whenever the XP routine is to be used.) XP could be modified to contain these constants directly. A modified version must be also used of STATIS, which calls the XP routine.

The new exponential routine $XP(x) = \exp(-x/2)$ is thus a fast assembly language routine designed to calculate the oft-repeated exponential function quickly and with an accuracy easily sufficient for use in CLASSY.

The XP routine causes slight variations in the calculated probabilities in CLASSY, which can cause the algorithm to take a different path (See the note "Path Differences in CLASSY" below.)

XP routine

The exponential $e^{-x/2}$ is calculated as follows: x is divided into three parts, I, n, and Σ.

$$x = 16I + \frac{n}{16} + \Sigma$$

$$-22 \leq I \leq 21 \quad ; \quad 0 \leq n < 256 \quad ; \quad 0 \leq \Sigma < \frac{1}{16}$$

Then

$$e^{-x/2} \approx \text{XPBIG(I)*XPAR(n)*(XPD} + \Sigma)$$

$$\approx 0 \quad I < -22 \text{ or } I > 21$$

This approximation has a relative RMS error of about 1/27000, which is well adequate for the CLASSY statistical system. The variable OUTCNT counts the number of overflows made.

XPP, XPAR, and XPBIG are generated by the routine XPREP.

Equations for constants used in exponential routine:

$$\text{XPAR}(n) = d_1 e^{-2\alpha n-\alpha} \qquad n = 0(1)256$$

$$\text{XPD} = d_0/d_1 - 2\alpha$$

$$d_0 = \sinh \alpha/\alpha \qquad \approx 1 + \frac{\alpha^2}{6} + \cdots$$

$$d_1 = \frac{-3}{2\alpha^3} (\sinh \alpha - \alpha \cosh \alpha) \approx -\frac{1}{2} - \frac{\alpha^2}{20} - \cdots$$

$$\alpha = \frac{\beta}{4} = \frac{1}{64} \text{ for intervals of } \frac{1}{16} \text{ in original argument (of } e^{-x/2})$$

These coefficients are the result of a least-squares fit of $(d_0 + d_1 \Sigma)$ to $e^{-\Sigma/2}$ for $\frac{-\beta}{2} < \Sigma < \frac{\beta}{2}$

$$\text{XPBIG}(I) = e^{-8I} \qquad -23 \leq I \leq 22$$

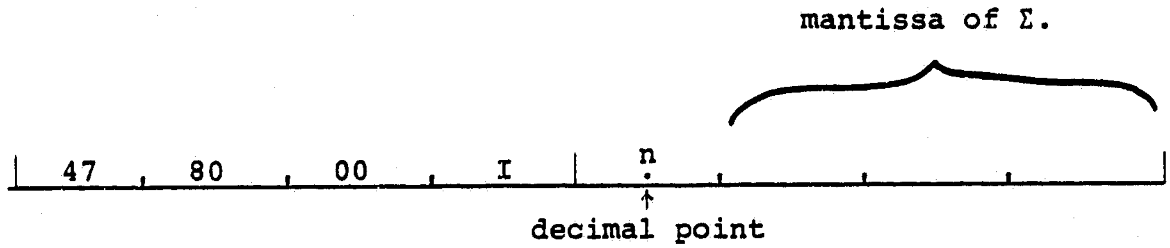
These are arranged in common block XPCOM (which is part of the exponential routine) as follows:

Quantity	Type	Bytes
OUTCNT	I * 4	4
XPXTRA	I * 4	4
XPD	R * 8	8
XPAR	R * 4	1024
XPBIG	R * 4	354

XP procedure:

The argument mantissa is shifted into the lower half of a double word (CARVE), by adding a special constant. If the argument x is $0 \leq x < 16$, the upper word is fixed, and is checked. The upper byte (n) of the mantissa is used as an index to XPAR, and is then replaced with a standard sign-exponent byte, to yield ξ. (The non-normalized character of ξ does not matter because it is immediately added to XPD, which has a larger exponent. This also eliminates the random 4 low-order bits of ξ.) The calculation is then direct. For x outside (0,16), the procedure is the same, except for a range check and an additional factor of XPBIG, derived from an index calculated from the upper half of the double word. If x is out of range (± 352, corresponding about to $10^{\pm 76}$), the result is set to 0, and overflows are counted.

CARVE structure:



Path Differences in CLASSY

The CLASSY algorithm may take widely different paths to the maximum likelihood solution, depending on very small effects. In some cases, whenever the clusters have not been fully resolved statistically due either to too few iterations or too little data, the two paths may end at slightly different points. Variations causing the algorithm to go on different paths include rounding error (for example use the associative and/or distributive laws) use of new standard function routines (such as the exponential routine), or other minor changes.

The reason CLASSY is so susceptible to small changes in the current version is the use of Monte-Carlo methods in processing the points with respect to classes which give them low probability. Small fluctuations in the calculations may accumulate until the threshold for the Monte-Carlo process is exceeded in one version and not in another. Since the random number in generator is sequential, this offsets the usage of all the succeeding random numbers, giving completely different Monte-Carlo choice for each point.

This, in turn, leads to moderately large fluctuations in the numerical values used, which ultimately changes the point at which the SPLIT or other threshold is passed. Since one cluster being SPLIT and separated can make a major difference in the processing of other clusters, the path taken by CLASSY rapidly becomes quite different.

In summary, minor changes in CLASSY can make large differences in the path the algorithm takes. This is mediated by the amplification caused by various thresholds, in particular those used in the Monte-Carlo subsystem. All the paths should, however, converge to the same point, except when insufficient iterations or data are used.

Appendix 1: Program listings for vector routines

This appendix contains listings of all the routines listed in Table 2. They are in the order: old Fortran, new Fortran, and assembly language. Preceding the routine listings is a listing of the EXEC routines for converting one set to the other.

```
      SUBROUTINE CORECT (REL,PV,P,S)
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,
     2   INDXVL,WFAC,NPTSO,PQRATH,SPMVTH,DWFAC,GRACTM,AMOFAC,
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,WSIM,WDELSM,
     4   BETTER,MODE,CORLEN,SPCOR
      REAL REL(30), PV(30), S(30)
      DO 10 I = 1,MQ
        REL(I) = PV(I) - S(I) / P
       II = I
C     WRITE (6,9999) II,REL(I),PV(I),S(I),P
9999  FORMAT ('CORECT I,REL,PV,S,P',I4,4(E10.4,2X))
10     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CORECT (REL,PV,P,S)
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,
     2   INDXVL,WFAC,NPTSO,PQRATH,SPMVTH,DWFAC,GRACTM,AMOFAC,
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,WSIM,WDELSM,
     4   BETTER,MODE,CORLEN,SPCOR
      REAL REL(30), PV(30), S(30)
C
      POV=1./P
      DO 10 I = 1,MQ
        REL(I) = PV(I) - S(I)*POV
C     WRITE (6,9999) I,REL(I),PV(I),S(I),P
9999  FORMAT ('CORECT I,REL,PV,S,P',I4,4(E10.4,2X))
10     CONTINUE
      RETURN
      END
```

CORECTA

```
*   THIS ROUTINE WAS WRITTEN BY ELOGIC,INC., NOV,1979, TO REPLACE THE
*   CORRESPONDING FORTRAN ROUTINE IN THE CLASSY SYSTEM. FOR ADDITIONAL
*   INFORMATION CONTACT ELOGIC,INC.
*
* SUBROUTINE  CORECT(REL,PV,P,S): REL(I)=PV(I)-S(I)/P
CORECT   CSECT
         USING CORECT,15
         STM   2,9,SAVE
         LM    1,4,0(1)            ARGS ADDR
         LA    8,4(0,0)            INCREMENT IS 1
         EXTRN MISC
         L     7,=A(MISC)
         USING MISCD,7
         L     9,MQ                COMPARAND IS MQ
         SLA   9,2                 MULTIPLY BY 4
         SR    9,8                 SUBTRACT 4
*
         LE    6,=E'1.0'
         DE    6,0(3)              GET 1/P
*
         SR    5,5                 DO LOOP INDEX I
IC       LE    2,0(4,5)            S(I)
         MER   2,6                   MULTIPLY BY 1/P
         LE    0,0(2,5)            PV(I)
         SER   0,2                   SUBTRACT REG2
         STE   0,0(1,5)            STORE REL(I)
         BXLE  5,8,IC
*
         LM    2,9,SAVE
         BR    14
SAVE     DS    9F
* COMMON  /MISC/
MISCD    DSECT
MQ       DS    F
         END
```

```
      FUNCTION DOTSQ(V,AMET)
C       CALCULATES THE INNER PRODUCT V.V RELATIVE TO THE METRIC AMET
C
      DIMENSION    LIST(38),  XTMP( 8),  YTMP( 7),  WADJ(24),     W(26),
     1 INDEX(37), LSUBS(40),LSUPER(39), IDADJ(23), NSYMB(36),    OW(25),
     2  PCUM(33),PRIRCM(32),   CIN(22),  CTOT(21),  PROP(29), SPFAC(15),
     3 OPROP(28), VOLIN(18), VOLRT(17),  DCON(16), PQRAT(14),  ODEN(19),
     4  DISS(35), PPASS(34),   PST(30),  OCIN(20), PCOND(31),OPRIOR(27),
     5  PAVE(13),  PILE(12)
      DIMENSION VRIN(475),GEN(999),GREF(999),ALINK(99)
      EQUIVALENCE (LINK(41),ALINK(41),GREF(8), GEN(7),VRIN(7))
      EQUIVALENCE (  LINK(41), LSUBS(40),LSUPER(39),  LIST(38),
     1  INDEX(37),  NSYMB(36),  DISS(35), PPASS(34),  PCUM(33),
     2  PRIRCM(32), PCOND(31),    PST(30),  PROP(29), OPROP(28),
     3  OPRIOR(27),      W(26),     OW(25),  WADJ(24), IDADJ(23),
     4     CIN(22),  CTOT(21),  OCIN(20),  ODEN(19), VOLIN(18),
     5   VOLRT(17),  DCON(16), SPFAC(15), PQRAT(14),  PAVE(13),
     6    PILE(12),  XTMP( 8),  YTMP( 7))
      COMMON/CLUS/ JUNK(12),NARL,NTOP,NTBSZM,NWANT,LINK(14000)
      DIMENSION MXAR(31),LR(3),LV(3)
      EQUIVALENCE (LR(1),LVRIN),(LR(2),LKURT),
     1   (LR(3),LOVAR),(LV(1),LSUM),(LV(2),LSKEW),(LV(3),LOSUM)
C
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,
     2   INDXVL,WFAC,NPTSO,PQRATH,SPMVTH,DWFAC,GRACTM,AMOFAC,
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,WSIM,WDELSM,
     4   BETTER, MODE
C      ( END OF STANDARD BLOCK. )
C
      DIMENSION NTB(32)
C
      COMMON /STPAR/WAIT,CONLV,SKBND,SKCHI,TRBND,TRCHI,URKBND,URKCHI,
     1    PACCEL(2),MACCEL(2),VACCEL(2)
      REAL V(30),AMET(475)
       REAL*8 DDOTSQ,DGDOT
      DDOTSQ=0.
      DGDOT=V(1)*V(1)*AMET(1)
      DO 10 I=2,MQ
      MX=MXAR(I)
    7 DO 8 J=2,I
    8 DDOTSQ=DDOTSQ+V(I)*V(J-1)*AMET(MX+J-1)
   10 DGDOT=DGDOT+V(I)*V(I)*AMET(MX+I)
C       THE DIAGONALS ARE HANDLED SEPARATELY BECAUSE EACH OFF-
C       DIAGONAL APPEARS TWICE, AND SO MUST BE DOUBLED.
      DDOTSQ=DDOTSQ+DDOTSQ+DGDOT
       DOTSQ = DDOTSQ
      RETURN
      END
```

```
      FUNCTION DOTSQ(V,AMET)
C       CALCULATES THE INNER PRODUCT V.V RELATIVE TO THE METRIC AMET
C
      DIMENSION    LIST(38),  XTMP( 8),  YTMP( 7),  WADJ(24),     W(26),
     1 INDEX(37), LSUBS(40),LSUPER(39), IDADJ(23), NSYMB(36),    OW(25),
     2  PCUM(33),PRIRCM(32),    CIN(22),  CTOT(21),  PROP(29), SPFAC(15),
     3 OPROP(28), VOLIN(18), VOLRT(17),  DCON(16), PQRAT(14),  ODEN(19),
     4  DISS(35), PPASS(34),   PST(30),  OCIN(20), PCOND(31),OPRIOR(27),
     5  PAVE(13),  PILE(12)
      DIMENSION VRIN(475),GEN(999),GREF(999),ALINK(99)
      EQUIVALENCE (LINK(41),ALINK(41),GREF(8), GEN(7),VRIN(7))
      EQUIVALENCE (  LINK(41), LSUBS(40),LSUPER(39),  LIST(38),
     1  INDEX(37),  NSYMB(36),  DISS(35), PPASS(34),  PCUM(33),
     2  PRIRCM(32), PCOND(31),   PST(30),  PROP(29), OPROP(28),
     3  OPRIOR(27),      W(26),    OW(25),  WADJ(24), IDADJ(23),
     4     CIN(22),  CTOT(21),  OCIN(20),  ODEN(19), VOLIN(18),
     5   VOLRT(17),  DCON(16), SPFAC(15), PQRAT(14),  PAVE(13),
     6    PILE(12),  XTMP( 8),  YTMP( 7))
      COMMON/CLUS/ JUNK(12),NARL,NTOP,NTBSZM,NWANT,LINK(14000)
      DIMENSION MXAR(31),LR(3),LV(3)
      EQUIVALENCE (LR(1),LVRIN),(LR(2),LKURT),
     1   (LR(3),LOVAR),(LV(1),LSUM),(LV(2),LSKEW),(LV(3),LOSUM)
C
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,
     2   INDXVL,WFAC,NPTSQ,PQRATH,SPMVTH,DWFAC,GRACTM,AMOFAC,
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,WSIM,WDELSM,
     4   BETTER, MODE
C     ( END OF STANDARD BLOCK. )
C
      DIMENSION NTB(32)
C
      COMMON /STPAR/WAIT,CONLV,SKBND,SKCHI,TRBND,TRCHI,URKBND,URKCHI,
     1    PACCEL(2),MACCEL(2),VACCEL(2)
      REAL V(30),AMET(475)
      DOTSQ=V(1)*V(1)*AMET(1)
      DO 10 I=2,MQ
      SQ=0.
      MX=MXAR(I)
    7 DO 8 J=2,I
    8 SQ=SQ+V(J-1)*AMET(MX+J-1)
   10 DOTSQ=DOTSQ+V(I)*((V(I)*AMET(MX+I)+SQ)+SQ)
C       THE DIAGONALS ARE HANDLED SEPARATELY BECAUSE EACH OFF-
C       DIAGONAL APPEARS TWICE, AND SO MUST BE DOUBLED.
      RETURN
      END
```

DOTSQA

```
*    THIS ROUTINE WAS WRITTEN BY ELOGIC,INC., NOV,1979, TO REPLACE THE
*    CORRESPONDING FORTRAN ROUTINE IN THE CLASSY SYSTEM. FOR ADDITIONAL
*    INFORMATION CONTACT ELOGIC,INC.
*
*FUNCTION DOTSQ(V,AMET): CALCULATE THE INNER PRODUCT
         GBLC  &P                  ADD - SUBTRACT PRECISION
&P       SETC  'ER'                ER SINGLE PRECISION; DR- DOUBLE PREC
DOTSQ    CSECT
         USING DOTSQ,15
         STM   2,12,SAVE
         LM    1,2,0(1)           ARGS ADDR
         LA    4,4(0,0)           I INCREMENT IS 1
         EXTRN MISC
         L     11,=A(MISC)
         USING MISCD,11
         L     5,MQ               I COMPARAND IS MQ
         SLA   5,2                MULTIPLY BY 4
*
         LR    6,4                J INCREMENT IS 1
         LA    8,8(0,0)
         SR    1,8                INDEX OF V STARTS FROM 1
         SR    2,8                INDEX OF AMET STARTS FROM 1
*
         LE    0,4(1,4)           V(1)
         MER   0,0                V(1)*V(1)
         ME    0,4(2,4)             *AMET(1)
         S&P   2,2                0.
*
         LR    7,8                DO LOOP INDEX I STARTS FROM 2
I1       L&P   4,2                SQ=0.
         L     12,MXAR-4(7)       MX=MXAR(I)
         SLA   12,2               MULTIPLY BY 4
         AR    12,2               ADD ADDR AMET
*
         LR    9,8                DO LOOP INDEX J STARTS FROM 2
I2       LE    6,0(1,9)           V(J-1)
         ME    6,0(12,9)          V(J-1)*AMET(MX+J-1)
         A&P   4,6                SQ
         BXLE  9,6,I2
```

```
*
         LE    6,4(1,7)           V(I)
         ME    6,4(12,7)          V(I)*AMET(MX+I)
         A&P   6,4                  +SQ
         A&P   6,4                    +SQ
         ME    6,4(1,7)                 *V(I)
         A&P   0,6                DOTSQ
         BXLE  7,4,I1
*
         LM    2,12,SAVE
         BR    14
*
SAVE     DS    12F
*
*
* COMMON  /MISC/
MISCD    DSECT
MQ       DS    F
MM       DS    F
LR       DS    3F
LV       DS    3F
NINCLS   DS    F
MXAR     DS    31F
         END
```

```
      SUBROUTINE MPVS(AM,C,V)
C        SETS AM=AM+V*V*C     (TENSOR PRODUCT)
C
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,
     2   INDXVL,WFAC,NPTSO,PQRATH,SPMVTH,DWFAC,GRACTM,AMOFAC,
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,WSIM,WDELSM,
     4   BETTER,MODE,CORLEN,SPCOR
C
      REAL AM(475),V(30)
      LOC=0
      DO 10 I=1,MQ
      DO 10 J=1,I
      LOC=LOC+1
   10 AM(LOC)=AM(LOC)+V(I)*V(J)*C
      RETURN
      END
```

```
      SUBROUTINE MPVS(AM,C,V)
C        SETS AM=AM+V*V*C     (TENSOR PRODUCT)
C
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,
     2   INDXVL,WFAC,NPTSO,PORATH,SPMVTH,DWFAC,GRACTH,AMOFAC,
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,WSIM,WDELSM,
     4   BETTER,MODE
C
      REAL AM(475),V(30)
      LOC=0
      DO 10 I=1,MQ
      CV=C*V(I)
      DO 10 J=1,I
      LOC=LOC+1
   10 AM(LOC)=AM(LOC)+V(J)*CV
      RETURN
      END
```

MPVSA

```
*   THIS ROUTINE WAS WRITTEN BY ELOGIC,INC., NOV,1979, TO REPLACE THE
*   CORRESPONDING FORTRAN ROUTINE IN THE CLASSY SYSTEM. FOR ADDITIONAL
*   INFORMATION CONTACT ELOGIC,INC.
*
* SUBROUTINE  MPVS(AM,C,V): SETS  AM=AM+V*V*C
MPVS     CSECT
         USING MPVS,15
         STM   2,12,SAVE
         LM    1,3,0(1)             ARGS ADDR
         LA    4,4(0,0)             I INCREMENT IS 1
         EXTRN MISC
         L     12,=A(MISC)
         USING MISCD,12
         L     5,MQ                 COMPARAND IS MQ
         SLA   5,2                  MULTIPLY BY 4
         SR    5,4                  SUBTRACT 4
*
         LR    8,4                  J INCREMENT IS 1
         SR    7,7                  LOC
*
         SR    9,9                  DO LOOP INDEX I
         LE    2,0(2)               C
*
I1       LER   6,2                  C
         ME    6,0(3,9)             C*V(I)
*
         SR    11,11                DO LOOP INDEX J
I2       LER   0,6                  C*V(I)
         ME    0,0(3,11)            C*V(I)*V(J)
         AE    0,0(1,7)             ADD AM(LOC)
         STE   0,0(1,7)             STORE AM(LOC)
         AR    7,4                  ADD 1 TO LOC
         BXLE  11,8,I2
         BXLE  9,4,I1
*
         LM    2,12,SAVE
         BR    14
SAVE     DS    12F
* COMMON  /MISC/
MISCD    DSECT
MQ       DS    F
         END
```

ORIGINAL PAGE IS
OF POOR QUALITY


```
      SUBROUTINE VMTV(VA,AMET,VB)
C       SETS VA=AMET*VB
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,
     2   INDXVL,WFAC,NPTSO,PQRATH,SPMVTH,DWFAC,GRACTM,AMOFAC,
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,USIN,UDELSM,
     4   BETTER,MODE,CORLEN,SPCOR
      REAL VA(30),VB(30),AMET(475)
      LOCA=0
      DO 20 I=1,MQ
      SUM=0.
      DO 10 J=1,I
      LOCA=LOCA+1
   10 SUM=SUM+ AMET(LOCA)*VB(J)
      IF(I.EQ.MQ) GO TO 20
      JS=I+1
      LOCB=LOCA+I
      DO 11 J=JS,MQ
      SUM=SUM+AMET(LOCB)*VB(J)
   11 LOCB=LOCB+J
   20 VA(I)=SUM
      RETURN
      END
```

VMTVA

```
*   THIS ROUTINE WAS WRITTEN BY ELOGIC,INC., NOV,1979, TO REPLACE
*   CORRESPONDING FORTRAN ROUTINE IN THE CLASSY SYSTEM. FOR ADDITIONAL
*   INFORMATION CONTACT ELOGIC,INC.
*
* SUBROUTINE  VMTV(VA,AMET,VB): SETS  VA=AMET*VB
VMTV     CSECT
         USING VMTV,15
         STM   2,12,SAVE
         LM    1,3,0(1)           ARGS ADDR
         LA    4,4(0,0)           I INCREMENT IS 1
         EXTRN MISC
         L     7,=A(MISC)
         USING MISCD,7
         L     5,MQ               COMPARAND IS MQ
         SLA   5,2                MULTIPLY BY 4
         DROP  7
*
         LR    8,4                J INCREMENT IS 1
         SR    1,4                INDEX OF VA STARTS FROM 1
         SR    2,4                INDEX OF AMET STARTS FROM 1
         SR    3,4                INDEX OF VB STARTS FROM 1
         SR    7,7                INITIALIZE LOCA
*
         LR    9,4                DO LOOP INDEX I STARTS FROM 1
I1       LE    6,=E'0.0'          INITIALIZE SUM
*
         LR    11,4               DO LOOP INDEX J STARTS FROM 1
I2       AR    7,4                ADD 1 TO LOCA
         LE    0,0(2,7)           AMET(LOCA)
         ME    0,0(3,11)            MULTIPLY BY VB(J)
         AER   6,0                    ADD TO SUM
         BXLE  11,8,I2
*
         CR    9,5                COMPARE INDEX I WITH ITS COMPARAND
         BZ    A12
         LR    11,9               I
         AR    11,4               JS=I+1
         LR    12,7               LOCA
         AR    12,9               LOCA +I
I3       LE    0,0(2,12)          AMET(LOCB)
         ME    0,0(3,11)            MULTIPLY BY VB(J)
         AER   6,0                    ADD TO SUM
         AR    12,11              ADD J TO LOCB
         BXLE  11,4,I3
*
A12      STE   6,0(1,9)           VA(I)
         BXLE  9,4,I1
*
         LM    2,12,SAVE
         BR    14
SAVE     DS    12F
* COMMON  /MISC/
MISCD    DSECT
MQ       DS    F
         END
```

```
      SUBROUTINE VPV(VA,FAC,VB)
C       SETS VA=VA+FAC*VB
C
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,
     2   INDXVL,WFAC,NPTSO,PQRATH,SPMVTH,DWFAC,GRACTM,AMOFAC,
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,WSIM,WDELSM,
     4   BETTER,MODE,CORLEN,SPCOR
C
      REAL VA(30),VB(30)
      DO 10 I=1,MQ
   10 VA(I)=VA(I)+VB(I)*FAC
      RETURN
      END
```

```
*   THIS ROUTINE WAS WRITTEN BY ELOGIC,INC., NOV,1979, TO REPLACE THE
*   CORRESPONDING FORTRAN ROUTINE IN THE CLASSY SYSTEM. FOR ADDITIONAL
*   INFORMATION CONTACT ELOGIC,INC.
*
* SUBROUTINE  VPV(VA,FAC,VB):  VA(I)=VA(I)+FAC*VB(I)
VPV      CSECT
         USING VPV,15
         STM   2,7,SAVE
         LM    1,3,0(1)            ARGS ADDR
         LE    2,0(2)              ARG2
         LA    4,4(0,0)            INCREMENT IS 1
         EXTRN MISC
         L     7,=A(MISC)
         USING MISCD,7
         L     5,MQ                COMPARAND IS MQ
         SLA   5,2                 MULTIPLY BY 4
         SR    5,4                 SUBTRACT 4
*
         SR    2,2                 DO LOOP INDEX I STARTS FROM 1
IC       LER   0,2                 FAC
         ME    0,0(3,2)            FAC*VB(I)
         AE    0,0(1,2)            VA(I)
         STE   0,0(1,2)            STORE VA(I)
         BXLE  2,4,IC              BRANCH LOCATION
*
         LM    2,7,SAVE
         BR    14
SAVE     DS    7F
* COMMON  /MISC/
MISCD    DSECT
MQ       DS    F
         END
```

Appendix 2: Program listings for the fast exponential system.

This appendix contains the routines used by the fast exponential system. They include the routine XP, the table generator XPREP, the modified version of CLINIT, showing where XPREP is called; and the modified version of STATIS, showing the calls to XP. (The version of STATIS displayed also contains modifications for the additional statistics collection, which will be the subject of an additional report.) The modification to CLINIT consists only of the one line calling XPREP, which may be executed even if XP is not used. Elogic recommends that the version containing the call to XPREP be used at all times.

```
*   THIS ROUTINE WAS WRITTEN BY ELOGIC,INC., NOV,1979, TO REPLACE THE
*   CORRESPONDING FORTRAN ROUTINE IN THE CLASSY SYSTEM. FOR ADDITIONAL
*   INFORMATION CONTACT ELOGIC,INC.
*
* EXPONENTIAL ROUTINE, EXP(-X/2)
XP       CSECT
         USING    *,15
         L        1,0(1)             ARG ADDR
         LE       0,0(1)             ARG
         AD       0,SHIFWRD          EXTRACT MANTISSA
         STD      0,CARVE
         L        1,CARVE            GET UPPERWORD
         S        1,UPCARVE          CHK ARG BETWEEN
         BNZ      NSEXP                0 AND 16
XPSEQ    IC       1,CARVE+4          GET UPPER BYTE
         MVI      CARVE+4,X'3F'      PUT IN EXPONENT
         LE       0,CARVE+4          LOAD XI
         AE       0,XPD              CALC EXPRESSION
         ALR      1,1                INDEX
         ALR      1,1
         ME       0,XPAR(1)          EXPONENTIAL TABLE
         BR       14                 (BYE)
*
NSEXP    C        1,MINCARV          CHK UNDERFLOW
         BM       ZEROIT
         C        1,MAXCARV          CHK OVERFLOW
         BP       OUTRANGE
         SR       1,1
         IC       1,CARVE+4          (DUP XPSEQ)
         MVI      CARVE+4,X'3F'
         LE       0,CARVE+4
         AE       0,XPD
         ALR      1,1
         ALR      1,1
         ME       0,XPAR(1)
         L        1,CARVE            GET BIG PART
         ALR      1,1                INDEX -USES SPEC
         ALR      1,1                  PROPERTIES OF SHIFWRD
         ME       0,XPBIG(1)         BIG EXPONENTIAL TABLE
         BR       14                 (BYE)
*
ZEROIT   LE       0,=F'0'            UNDERFLOW -
         BR       14                 RETURN 0.
*
OUTRANGE L        1,OUTCNT
         A        1,=F'1'
         ST       1,OUTCNT
         B        ZEROIT
```

```
*
*
CARVE    DS    D
SHIFWRD  DC    A(X'47800000')      SHIFTS ARGUMENT TO GET
         DC    F'0'                  MANTISSA
UPCARVE  EQU   SHIFWRD             ELIMS SHIFWRD
MINCARV  DC    F'-21'
MAXCARV  DC    F'21'
*
XPCOM    DS    0D
         ENTRY XPCOM
OUTCNT   DC    F'0'
XPXTRA   DC    F'0'
XPD      DS    D
XPAR     DS    256F
         DS    21F                  XPBIG NEG ARG
XPBIG    DS    22F
         END
```

```
      SUBROUTINE XPREP
      COMMON /XPCOM/ OUTCNT,XPXTRA,XPD,XPARO(256),XPBIGO(43)
      REAL*8 XPD,ALPHA,C2,D1
C
      BETA=1./16.
      ALPHA=BETA/4.
      C2=2.*ALPHA
      D1=3.*(DSINH(ALPHA)-ALPHA*DCOSH(ALPHA))/(C2*ALPHA*ALPHA)
      XPD=DSINH(ALPHA)/(ALPHA*D1) - C2
      DO  10  N=1,256
   10 XPARO(N)=D1*DEXP(-C2*(N-1)-ALPHA)
      DO  20  I=1,43
   20 XPBIGO(I)=DEXP(-8.D0*(I-22))
      RETURN
      END
```

```
      SUBROUTINE CLINIT(KROT)                                           CLI00010
C   THIS ROUTINE CONTAINS THE VARIOUS STATEMENTS NECESSARY TO           CLI00020
C      INITIALIZE THE CLUSTERING ALGORITHM.                             CLI00030
C                                                                       CLI00040
      DIMENSION    LIST(38),  XTMP( 8),  YTMP( 7),  WADJ(24),     W(26),CLI00050
     1 INDEX(37), LSUBS(40),LSUPER(39), IDADJ(23), NSYMB(36),    OW(25),CLI00060
     2  PCUM(33),PRIRCM(32),     CIN(22),  CTOT(21),  PROP(29), SPFAC(15),CLI00070
     3 OPROP(28), VOLIN(18), VOLRT(17),  DCON(16), PQRAT(14),  ODEN(19),CLI00080
     4  DISS(35), PPASS(34),     PST(30),  OCIN(20), PCOND(31),OPRIOR(27),CLI00090
     5  PAVE(13),  PILE(12)                                             CLI00100
      DIMENSION VRIN(475),GEN(999),GREF(999),ALINK(99)                  CLI00110
      EQUIVALENCE (LINK(41),ALINK(41),GREF(8), GEN(7),VRIN(7))          CLI00120
      EQUIVALENCE (  LINK(41), LSUBS(40),LSUPER(39),  LIST(38),         CLI00130
     1  INDEX(37),  NSYMB(36),  DISS(35), PPASS(34),  PCUM(33),         CLI00140
     2  PRIRCM(32), PCOND(31),    PST(30),  PROP(29), OPROP(28),        CLI00150
     3  OPRIOR(27),      W(26),     OW(25),  WADJ(24), IDADJ(23),       CLI00160
     4     CIN(22),  CTOT(21),  OCIN(20),  ODEN(19), VOLIN(18),         CLI00170
     5   VOLRT(17),  DCON(16), SPFAC(15), PQRAT(14),  PAVE(13),         CLI00180
     6    PILE(12),  XTMP( 8),  YTMP( 7))                               CLI00190
      COMMON/CLUS/ JUNK(12),NARL,NTOP,NTBSZM,NWANT,LINK(14000)          CLI00200
      DIMENSION MXAR(31),LR(3),LV(3)                                    CLI00210
      EQUIVALENCE (LR(1),LVRIN),(LR(2),LKURT),                          CLI00220
     1    (LR(3),LOVAR),(LV(1),LSUM),(LV(2),LSKEW),(LV(3),LOSUM)        CLI00230
C                                                                       CLI00240
      COMMON /MISC/ MQ,MM,LR,LV,NINCLS,MXAR,WTINIT,KROOT,EPS,DELT,      CLI00250
     1   AMQ,ODCON,XOVFLO,XUNFLO,WADJIN,ELIMTH,SEPTH,VFAC,AMM,SBLTH,    CLI00260
     2   INDXVL,WFAC,NPTSO,PQRATH,SPMVTH,DWFAC,GRACTM,AMOFAC,           CLI00270
     3    AMOMIN,AMOMAX,AMORAT,VOLLIM,BIAS,PJOIN,VRJOIN,WSIM,WDELSM,    CLI00280
     4   BETTER, MODE                                                   CLI00290
C     ( END OF STANDARD BLOCK. )                                        CLI00300
C                                                                       CLI00310
      DOUBLE PRECISION XTEMP,YTEMP,ZTEMP,DURK,DURKD                     CLI00320
C                                                                       CLI00330
      COMMON /STPAR/WAIT,CONLV,SKBND,SKCHI,TRBND,TRCHI,URKBND,URKCHI,   CLI00340
     1    PACCEL(2),MACCEL(2),VACCEL(2)                                 CLI00350
C                                                                       CLI00360
      COMMON/CLUSTR/ IBEGIN,TOTWRD,CLSNAM,IPT,NOFLD, SYM(61) ,          CLI00370
     1               LNCAT, PRNT(4) , KLBC , PRTME, PROUT, TOTPIX,      CLI00380
     2  SCRAM1,BUFPIX,BUFTOT,NBUFSD,NDUMP,LBUFD                         CLI00390
     3, MAXBF, AREA, NWDS, NWDRS, NPTS, LBUF, IQ1,NOCYCL, NCL           CLI00400
C                                                                       CLI00410
      INTEGER TOTWRD,SYM,PRNT,PRTME,PROUT,TOTPIX,SCRAM1,BUFPIX,BUFTOT   CLI00420
     1 ,CLSNAM                                                          CLI00430
C                                                                       CLI00440
      COMMON /MXLL/ MXLLWT, MXLLFN, RELPRP(200)                         CLI00450
C                                                                       CLI00460
      COMMON /INITL/WTNEW,DEVINI,CHANIN                                 CLI00470
      CHIVAL(DF)=DF*(1.-.222/DF+CONLV*SQRT(.222/DF))**3                 CLI00480
```

```
C  INITIALIZS ADDRESS
C
      CALL DATFIX
C
      AMQ=MQ
      MQS=MQ*MQ
C
C  DEFINE VALUE OF SEPTH IN TERMS OF CHI SQUARE VALUE
      DFT = AMQ + 1
      SEPTH = (CHIVAL(DFT))/2
C
C  WE FIRST SET UP VARIOUS INDEX ARRAYS FOR A PARTICULAR
C        NUMBER OF CHANNELS MQ.
C       SET UP THE TRIANGULAR POSITION ARRAY MXAR.
      MM=0
      DO 10 I=1,31
      MXAR(I)=MM
   10 MM=MM+I
      MM=MXAR(MQ+1)
      AMM=MM
C  SET UP TABLES FOR THE XP FUNCTION
      CALL XPREP
C NOW WE SET UP THE ORIGIN VECTORS, LR AND LV, OF THE VARIOUS ARRAYS
C     AND VECTORS IN A CLUSTER NODE.
      NINCLS=1
C ***** THIS CONSTANT MUST BE SET TO THE NUMBER OF ARRAYS *****
      DO 21 I=1,3
      LR(I)=NINCLS
   21 NINCLS=NINCLS+MM
      DO 22 I=1,3
      LV(I)=NINCLS
   22 NINCLS=NINCLS+MQ
      NSCALS = 35
      NINCLS=NINCLS+NSCALS-1
C  WE MUST ALSO SET UP SOME THRESHOLDS FOR USE BY THE STATISTICAL
C       SYSTEM.
      SKCHI=(AMQ+2.)*(AMQ+4.)*CHIVAL(AMQ)
      URKCHI=AMQ*(AMQ+4.)*(AMQ+6.)/(AMQ-.999)*CHIVAL(AMM-1.)
      TRCHI=CONLV*CONLV*(AMQ*(AMQ+2.)*(AMQ+3.)*8.)
C  WE NOW CREATE THE HEAD NODE OF THE CLUSTER TREE.  THIS IS NOT
C     AN ACTUAL CLUSTER, AND DOES NOT HAVE STORAGE FOR ANY
C     OF THE STATISTICAL ARRAYS.
      NPTSO=0
      KROT=MORSTR(NSCALS)
C   MAKE FIRST NODE START AT AN ODD NUMBER
       IF (MOD(NTOP,2) .NE. 1) NTOP = NTOP + 1
      LINK(KROT)=-262139
      LSUPER(KROT)=-262142
```

```
      IDADJ(KROT)=999999
      INDEX(KROT)=-0
      SPFAC(KROT)=99999.
      W(KROT)=WTINIT
      OW(KROT)=W(KROT)
      PQRAT(KROT)=0.
      PROP(KROT)=1.
      OPROP(KROT)=1.
      CIN(KROT)=W(KROT)
      OCIN(KROT)=CIN(KROT)
      CTOT(KROT)=0.
      ODEN(KROT)=W(KROT)
      PRIRCH(KROT)=1.
C  NEXT THE INITIAL NODE IS SET UP, TOGETHER WITH SOME CONTROL THRESHOLD
      ODCON =60.
      PROPI = 1.
      INDXVL = 0
   57 KFIR = NEWCLS(KROT, PROPI, WTINIT)
      LSUBS(KROT) =KFIR
      LINK(KFIR)=0
      DEV2WT =DEVINI*DEVINI*WTINIT
      LA=MORSTR(MQS)
      DO 52 I=1,MQ
      GREF(KFIR+LSUM+I)=WTINIT*CHANIN
      DO 52 J=1,MQ
      IJ=J-1+(I-1)*MQ+LA
      ALINK(IJ) =0.
      IF(I.EQ.J) ALINK(IJ)=DEV2WT
   52 CONTINUE
C
      CALL ZEROCL(KFIR,ALINK(LA),GREF(KFIR+LSUM+1),GREF(KFIR+LSKEW+1),
     1 GREF(KFIR+LKURT+1), GREF(KFIR+LOSUM+1),GREF(KFIR+LOVAR+1))
      CALL FREE(LA, MQS)
C  SET SWITCHES FOR MAX LIKLIHOOD LABELING
      MXLLWT = 0
      WADJ(KFIR) =WADJIN
      MXLLFN = 23
      PRINT 273,MQ,CONLV,TRCHI,SKCHI,URKCHI
  273 FORMAT ('1 CONFIDENCE LEVELS',I4,' CHANNELS',F8.4,'  CHISQUARES',
     1 3F12.2)
C
C
C    INITIALIZE ADDRESS :
C
      CALL STATAD(GEN(LSUM),GEN(LSKEW),GEN(LKURT),
     1 GEN(LOSUM),GEN(LOVAR))
      RETURN
      END
```