NASA CR-
147825

(NASA-CR-147825) A QUASI-NEWTON APPROACH TO OPTIMIZATION PROBLEMS WITH PROBABILITY DENSITY CONSTRAINTS (Rice Univ.) 41 p HC $4.00 CSCL 12A

N76-27943

Unclas

G3/65 44567

ICSA

INSTITUTE FOR COMPUTER SERVICES AND APPLICATIONS

RICE UNIVERSITY

275-025-029

A Quasi-Newton Approach to Optimization Problems with Probability Density Constraints

by

R.A. Tapia and D.L. Van Rooy
Rice University

ABSTRACT:

A quasi-Newton method is presented for minimizing a nonlinear functional while constraining the variables to be nonnegative and sum to one. The nonnegativity constraints are eliminated by working with the squares of the variables and the resulting problem is solved using Tapia's general theory of quasi-Newton methods for constrained optimization. A user's guide for a program implementing this algorithm is provided.

Institute for Computer Services and Applications
Rice University
Houston, TX 77001
June, 1976

This work was sponsored in part by NASA contract NAS 9-12776

275-025-029

A Quasi-Newton Approach to Optimization Problems with Probability Density Constraints

by

R.A. Tapia and D.L. Van Rooy
Rice University

ABSTRACT:

A quasi-Newton method is presented for minimizing a nonlinear functional while constraining the variables to be nonnegative and sum to one. The nonnegativity constraints are eliminated by working with the squares of the variables and the resulting problem is solved using Tapia's general theory of quasi-Newton methods for constrained optimization. A user's guide for a program implementing this algorithm is provided.

Institute for Computer Services and Applications
Rice University
Houston, TX 77001
June, 1976

This work was sponsored in part by NASA contract NAS 9-12776

I. INTRODUCTION

Consider the optimization problem

$$\text{(1)}\quad \begin{aligned} &\min f(x_1, \dots, x_n)\ ;\ \text{subject to}\quad 1 - \sum_{i=1}^{n} x_i = 0, \\ &x_i \geq 0,\quad i=1,\dots,n \end{aligned}$$

Problems of the form (1) occur frequently in statistical applications. In [illegible]se applications, the variables x_i usually represent a discrete probability density function and the functional f represents an optimality criterion, e.g., the negative likelihood or negative log likelihood.

Let us make the change of variables

$$\text{(2)}\quad x_i = \tfrac{1}{2} y_i^2 \quad , \quad i=1,\dots,n$$

and consider the optimization problem

$$\text{(3)}\quad \min \hat{f}(y_1, \dots, y_n)\ ;\ \text{subject to}\quad 1 - \sum_{i=1}^{n} \tfrac{1}{2} y_i^2 = 0$$

where

$$\text{(4)}\quad \hat{f}(y_1, \dots, y_n) \equiv f(\tfrac{1}{2} y_1^2, \dots, \tfrac{1}{2} y_n^2)$$

Proposition 1.1

If $y = (y_1, \dots, y_n)^T$ solves problem (3), then $x = (\frac{1}{2} y_1^2, \dots, \frac{1}{2} y_n^2)^T$ solves problem (1). Conversely, if $x = (x_1, \dots, x_n)^T$ solves problem (1), then $y = (\sqrt{2x_1}, \dots, \sqrt{2x_n})^T$ solves problem (3).

Proof

The proof is not difficult.

Problem (3), in contrast to problem (1), does not involve inequality constraints. The price one pays for this simplification is that problem (3) will have nore critical points than problem (1) and the equality constraint is quadratic instead of linear. We maintain, that according to Proposition 1.1, if one starts sufficiently near the solution, the extraneous critical points will not affect the algorithm. Moreover, when f is nonlinear, the quadratic constraints should not significantly increase the degree of complexity of the problem.

Tapia's approach to quasi-Newton methods for constrained optimization ([1] and [2]) depends heavily on the invertibility of the Hessian of the Lagrangian at the solution. Hence, we are certainly interested in determing the relationship between the invertibility of the Hessian of the Lagrangian for problem (1) and the invertibility of the Hessian of the Lagrangian for problem (3). Toward this end let

$$(5)\quad L(x,\mu,\lambda) = f(x) - \sum_{i=1}^{n} \mu_i x_i + \lambda(1 - \sum x_i)$$

and

$$(6)\quad \hat{L}(y,\lambda) = \hat{f}(y) + \lambda(1 - \sum_{i=1}^{n} \tfrac{1}{2} y_i^2)$$

we use the notation

$$\Lambda = (\lambda, \ldots, \lambda)^T,$$

$$Y = \operatorname{diag}(y_1, \ldots, y_n)$$

and

$$X = \operatorname{diag}(x_1, \ldots, x_n)$$

Straightforward calculations show that

$$(7)\quad \nabla_y \hat{L}(y, \lambda) = Y\left(\nabla f(x) - \Lambda\right) ,$$

$$(8)\quad \nabla_y^2 L(y, \lambda) = Y \nabla^2 f(x) Y + \operatorname{diag}\left(\nabla f(x) - \Lambda\right) ,$$

$$(9)\quad \nabla_x L(x, \mu, \lambda) = \nabla f(x) - \Lambda - \mu ,$$

and

$$(10)\quad \nabla_x^2 L(x, \mu, \lambda) = \nabla^2 f(x) .$$

The first order necessary conditions for problem (1) are

$$(11)\quad \begin{aligned} \nabla f(x) - \Lambda - \mu &= 0 \\ 1 - \sum_{i=1}^{n} x_i &= 0 \\ \mu_i &\ge 0 \\ x_i &\ge 0 \\ \mu_i x_i &= 0 , \quad i=1,\ldots,n ; \end{aligned}$$

while the first order necessary conditions for problem (3) are

$$(12)\quad \begin{aligned} Y\left(\nabla f(x) - \Lambda\right) &= 0 \\ 1 - \sum_{i=1}^{n} \tfrac{1}{2} y_i^2 &= 0 \end{aligned}$$

Recall that <u>strict complementarity</u> means that in (11) we do not have $\mu_i = x_i = 0$ for any i .

Proposition 1.2

Let (x, u, λ) be a solution of problem (1) and (y, λ) the corresponding solution of problem (3). Then

(i) we have strict complementarity and $\nabla^2 f(x)$ is positive definite $\Longleftrightarrow \nabla_y^2 \hat{L}(y, \lambda)$ is positive definite

(ii) we have strict complementarity and $\nabla^2 f(x)$ is invertible $\Longleftrightarrow \nabla_y^2 L(y, \lambda)$ is invertible.

Proof

Again the proof is not difficult. One merely works with (8), (10), (11) and (12).

Proposition 1.2 is very satisfying and warns us tha in the algorithm we must guard against the loss of strict complementarity. This will be accomplished by working with the variable

$$(13) \qquad x_i = \frac{1}{E_i} y_i^{E_i}$$

where $E_i = 2$ if $|x_i| + |\mu_i| \neq 0$ and $E_i = 1$ if $|x_i| + |\mu_i| = 0$. Observe that, when $E_i = 1$ in (13), we have effectively ignored the nonnegativity constraint on x_i.

II. THE QUASI-NEWTON ALGORITHM

The algorithm we are about to present is a special case of the approach given by Tapia in [2] applied to problem (3). The reader is referred to that paper for a better understanding of the theory and algorithm. In describing an iterative method, it is convenient to denote the present iterate by x and the subsequent iterate by $\bar{x}$ and similarly for other variables. Also, we let I denote the identity matrix , $\cdot\rangle$ denotes the Euclidean inner product, $e_1 = (1, 0, \ldots, {}^T, \ldots, e_n = (0, \ldots, 1)^T$, (a_{ij}) denotes the matrix whose (i, j)-th element is a_{ij} and b_i or $(b)_i$ denotes the i-th element of the vector b.

Consider the constrained optimization problem

$$(14) \qquad \min \ f(x), \quad \text{subject to} \quad g(x) = 0 \ ;$$

where $f, g : R^n \to R^1$. We first describe the quasi-Newton method BFGS given in [2] for problem (14). Let

$$(15) \qquad L(x, \lambda) = f(x) + \lambda \ g(x)$$

Quasi-Newton BFGS

Step 1: (Initialization) Determine x and B.

Step 2: (Update x and λ) Let

$$\bar{\lambda} = \frac{g(x) - \langle \nabla g(x), B^{-1} \nabla f(x) \rangle}{\langle \nabla g(x), B^{-1} \nabla g(x) \rangle}$$

$$\bar{x} = x - B^{-1} \nabla_x L(x, \bar{\lambda}) .$$

Step 3: (Update approximate Hessian)

$$\bar{B} = \left(B + \frac{y y^T}{\langle y, s \rangle} - \frac{B s s^T B}{\langle s, Bs \rangle} \right)$$

where

$$s = \bar{x} - x$$

and

$$y = \nabla_x L(\bar{x}, \bar{\lambda}) - \nabla_x L(x, \bar{\lambda}) .$$

Step 4: GO TO STEP 2 .

In implementing this algorithm, one would have to make provisions for output and a stopping criterion. In [2], Tapia has demonstrated that the above algorithm is locally superlinearly convergent.

Remark 1:

If the initial B depended on λ, we would make an initial approximation of λ according to the projection formula (see [2])

$$(16) \qquad \lambda = - \frac{\langle \nabla g(x), \nabla f(x) \rangle}{\langle \nabla g(x), \nabla g(x) \rangle}$$

Remark 2:

In updating B in Step 3, it is advantageous to store B factored according to a Cholesky decomposition and then merely update these factors. Moreover, B can be slightly modified so that the resultant matrix is always positive definite. For a detailed description of these modifications, the reader is referred to Gill et al [3], Van Rooy et al [4], Fletcher and Powell [5] and Van Rooy [6]. The quantities $B^{-1} \nabla f(x)$ and $B^{-1} \nabla g(x)$ in Step 2 are then obtained in the standard back-substitution manner.

We now apply the above algorithm for problem (3) with the additional feature of obtaining a discrete Newton approximation to the Hessian every M iterations (where M is an input parameter).

Define the following functions

$$(17) \qquad x_i = \frac{y_i^{E_i}}{E_i}$$

$$(18) \qquad \nabla g_i = - \left[2 - E_i + (E_i - 1)\, y_i \right], \quad i=1,\ldots,n$$

and

$$(19) \qquad g = 1 - \sum_{i=1}^{n} x_i$$

The quantity ∇g is merely the gradient of g with respect to y and (13) and (17) are the same.

Quasi-Newton Algorithm for Problem (3)

Step 1: (Initialization)

Read	CRC	(constraint regection criterion)
	h	(discretization step)
	ε	(stopping criterion)
	DI	(descent indicator)
	IØUTPUT	(output indicator)
	M	(discrete Hessian indicator)
	MAX	(maximum number of iterations)

Let

$$E_i = 2$$

$$y_i = \sqrt{\frac{2}{n}}$$

$$\lambda = \langle \nabla f(x) , x \rangle$$

$$B = I$$

Step 2: Output according to IØUTPUT, check stopping criterion

IF $\left(\| \nabla_y L(y, \lambda) \| + g^2 < \epsilon\right.$ or

ITERATION > MAX $\left.\right)$ stop

Step 3: Calculate discrete Hessian according to M

$$B = \left(\frac{\nabla_y \hat{L}(y + he_j, \lambda)_i - \nabla_y \hat{L}(y, \lambda)_i}{h} \right)$$

Calculate modified Cholesky decomposition of B .

Step 4: Calculate multiplier correction

$$\Delta\lambda = \frac{g - \langle \nabla g , B^{-1} \nabla_y \hat{L}(y, \lambda) \rangle}{\langle \nabla g , B^{-1} \nabla g \rangle}$$

Step 5: Calculate y-variable correction

$$\Delta y = - B^{-1} \nabla_y \hat{L}(y, \lambda) - \Delta\lambda B^{-1} \nabla g$$

Step 6: Update y according to descent indicator

$\bar{\lambda} = \lambda + \alpha \Delta \lambda$

$\bar{y} = y + \alpha \Delta y$

where $\alpha = 1$ if descent is not desired, otherwise α is chosen according to the descent principle on $\| \nabla_y \hat{L} (y, - \lambda) \|^2 + g^2$ (see [2]).

Step 7: Update approximate Hessian according to BFGS and modified Cholesky decomposition

$$\overline{B} = B + \frac{Dy\ Dy^T}{\langle Dy, Ds \rangle} - \frac{B\ Ds\ Ds^T B}{\langle Ds, B\ Ds \rangle}$$

where $Ds = \overline{x} - x$ and

$$Dy = \nabla_y \hat{L}(\overline{y}, \overline{\lambda}) - \nabla_y \hat{L}(y, \overline{\lambda})$$

Step 8: Update constraint rejection indicators

$$T_0 = a_1 \left[\| \nabla_y \hat{L}(\overline{y}, \overline{\lambda}) \|^2 + \left(1 - \sum_{i=1}^{n} x_i\right)^2 \right]^{a_2},$$

$$T_1 = \min\left(\frac{1}{4n}, T_0\right),$$

$$T_2 = \min\left(CRC, T_0\right)$$

If $\left(|\overline{x}_i| < T_1 \text{ and } \left| \frac{\partial f}{\partial x_i} - \lambda \right| < T_2 \right)$, let $E_i = 1$

If E_i has changed, reset $\overline{y}_i$ $\quad i=1,\ldots,n$

Step 9: GO TO STEP 2 .

III. USER'S GUIDE

The above algorithm has been implemented in a set of FORTRAN subroutines whose entry point is the subroutine CONOPT. In this section, we shall describe how to use this program, provide a description of the workings of the program, and illustrate its use with examples. A listing of the program is given in the appendix. This program runs on an IBM 370/155 using the FORTRAN G, G1, or H compilers. All computations are done in double precision. Unit 6 is used for output, if requested.

To utilize this program, the user supplies several parameters and two subprograms which calculate the function value and its gradient at a specified point. The program will then attempt to minimize the function subject to the constraints and return the argument at the minimum.

The calling sequence of the program is CALL CONOPT (N, DELF, FUNCT, X, WKA, MITER, STC, CRC, IDN, DH, IØUT, DSNT, A, RSTRT, USELAM, XLAM).

These parameters are described below:

N - integer variable indicating the dimension of the problem.

DELF - a user supplied subroutine which calculates the gradient of the function. The calling sequence is CALL DELF (DF, X, N), where X is the N dimensional argument and DELF returns the gradient in the vector DF. Note that DF and X are both double precision variables.

FUNCT - a user supplied double precision function subprogram which calculates the function value at a specified point. The calling sequence is F = FUNCT (X, N) where X is the (double precision) argument of length N . Though the function value is not required by the algorithm, CONOPT will output it in the convergence test step if the user requests output. (It is called only when output is requested.) If, for any reason, the user is not interested in this value, he may just supply a function subprogram that sets the function value to zero and returns. [N.B. The two programs DELF and FUNCT must be declared external in the calling program. The two programs may have other entry points, with different arguments, referred to by the user's calling program prior to his calling CONOPT in order to make other variables or their addresses available to either or these subprograms.]

X - on output, this will contain the N-dimensional double precision argument last calculated by the program (at the minimum if the program converged). On input, the program will set each element of X to the value 1/N , unless the user has specified restart (see description of the variable RSTRT below), in which case the supplied values of X will be used as the starting point.

WKA - a double precision working storage array of length $\geq \frac{1}{2} N^2 + (23N + 7) / 2$. If a restart has been specified, the first N (2-byte integer) locations should contain the desired constraint rejection indicators. However, if these variables do not contain either 1's or 2's , the program will set them to 2. The final constraint rejection indicators will be in these locations.

MITER - an integer variable indicating the maximum number of iterations to perform. If the user sets this to 0, the program will reset it to max (30, 2 * N).

STC - a real variable specifying the stopping criterion on the norm squared of the gradient of the Lagrangian plus the constraint, i.e. $\| \nabla L \|^2 + \left(1 - \sum_{i=1}^{n} x_i\right)^2$. If 0 is specified, the program will set STC = 10^{-8}.

CRC - a real variable specifyi the constraint rejection criterion. See the algorithm description section for a precise definition of its use. If 0, is specified, the program will set CRC = 10^{-2} .

IDN - an integer variable specifying the frequency of calculation of the discrete approximation to the Hessian, i.e., this finite difference Hessian is calculated every IDN iterations. If 0 is specified, this approximation is never used.

If a negative value is specified, the program sets IDN = min (10, N).

DH - a real variable specifying the step size to be used in calculating the finite difference Hessian (see the algorithm description for a detailed definition). If 0 is specified, the program will set DH = 10^{-2} .

IØUT - an integer variable specifying the frequency of output; i.e., the program will output various computed quantities every IØUT iterations. If 0 is specified, no output will be produced. If a negative value is specified, the program sets IØUT = 1 .

DSNT - a one-byte logical variable indicating whether the program should implement descent in its search at each iteration.

A - a two-word real vector containing respectively the multiplier and exponent for DNORM $\left(= \| \nabla L \|^2 + (1 - \sum_{i=1}^{n} x_i)^2 \right)$ for use in the constraint rejection test; i.e., the appropriate quantity is A(1) * DNORM * * A(2) . The defaults for A(1) and A(2) are 1. and .2 respectively. If the user wishes to set A(2) = 0., he should set A(1) to the negative of what he wants for A(1) and then a value of zero will be used for A(2) .

RSTRT - a one-byte logical variable indicating whether the user is supplying an initial value for X (as in a restart case). If RSTRT = .TRUE. , the first 2N bytes of WKA should contain the vector ICR (see description of WKS and the algorithm description— the E_i's).

USELAM - a one-byte logical variable used if RESTR = .TRUE. to indicate if the user is supplying a value for XLAM . If RESTR = .TRUE. and USELAM = .FALSE. , the value $< \nabla f , x >$ is used for XLAM .

XLAM - the Langrange multiplier for the equality constraint. Used on input only if USELAM = RSTRT = .TRUE.

The default values for the above parameters have been chosen so that many problems will converge; thus, the only input parameters with which the user need concern himself in many cases is N, IØUT, DSNT and RSTRT (usually = .FALSE.)

The subprograms used and a brief description of their function is given below:

CONOPT - main driver subroutine

DIMS - computes base addresses for various working arrays used throughout the calculations.

INPT - sets default values for parameters and outputs the parameters used.

INIT initializes y, x, ICR, H and λ . Also fetches values of ∇f and ∇L .

CHKSTP - checks the stopping criterion and outputs quantities if requested.

DHESS - calculates a finite difference approximation to the Hessian and its modified Cholesky decomposition, if requested. The Hessian is forced to be positive definite.

DELAMY - computes the update terms $\nabla \lambda$ and ∇y.

NEWY - updates y and λ based optionally on descent of $\| \nabla L \|^2 + \left(1 - \sum_{i=1}^{n} x_i \right)^2$.

UPDTH - updates the approximate Hessian using the Broyden-Fletcher-Goldfarb-Shannon (BFGS) update technique. (Really the MCD of it is updated.) Here again the Hessian is forced to be positive definite.

CONSTR - updates the constraint rejection indicators and, if necessary, y and ∇L.

GRADL - computes ∇L given ∇f, λ, y, and ICR.

SUM - a function subprogram that performs various utility functions $\left(\sum_{i=1}^{n} x_i ,\ \sum_{i=1}^{n} x_i^2 ,\ \sum_{i=1}^{n} x_i \, y_i \right)$.

MCHLSK - computes the modified Cholesky decomposition (MCD) of a matrix. If the matrix is not positive definite, the MCD is modified to represent a matrix that is (see e.g. ref 4).

PDSOLV - solves $Ax = y$ where A contains the MCD of a matrix.

COMPT - computes a rank one update of an MCD using t' e composite-t method (see refs. 5 and 6).

LDMUL - multiply a vector by a matrix whose MCD is supplied.

Examples:

The following quadratic programming (i.e., $\min \| Ax - b \|^2$ s. t. $x_i \geq 0$ and $\sum x_i = 1$) examples were run using all the default options:

1. $$A = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$x = (.11112108,\ .44443959,\ .44443959)^T$$

congerged in 5 iterations. Exact answer = $(1/9,\ 4/9,\ 4/9)^T$

The robustness of the algorithm is exemplified by the following two examples for which the algorithm, theoretically, should not converge since the Hessian is singular at the solution.

2. $$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 2 & 1 & 2 \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$$

$$x = (.50000019,\ 0.,\ .50000004)^T$$

converged in 14 iterations. Exact answer = $(\frac{1}{2},\ 0,\ \frac{1}{2})^T$.

The constraint for x_2 is not active in this case.

3.
$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$x = (0, 0, 1)^T$$

congerged in 11 iterations. The constraints for x_1 and x_2 are not active in this case.

REFERENCES

[1] Tapia, R. A., "A Stable Approach to Newton's Method for General Mathematical Programming Problems in R^n," Journal of Optimization Theory and Applications, 14 (1974), pp. 453-476.

[2] Tapia, R. A., "Diagonalized Multiplier Methods and Quasi-Newton Methods for Constrained Optimization," Journal of Optimization Theory and Applications.

[3] Gill, P. E., Golub, G., Murray, W., and Saunders, M. A., "Methods for Modifying Matrix Factorizations," Math. Comp., 28 (1974), pp. 505-536.

[4] Van Rooy, D. L., Lynn, M. S., and Snyder, C. H., "The Use of the Modified Cholesky Decomposition in Divergence and Classification Calculations," Conf. Proc. Machine Processing of Remotely Sensed Data, LARS, Purdue, Oct. 16-18, 1973, pp. 3B-35 - 3B-47.

[5] Fletcher, R. and Powell, M. J. O., "On the Modification of LDL^T Factorizations," Math. Comp., 28 (1974), pp. 1067-1087.

[6] Van Rooy, D. L., "On the Computation and Updating of the Modified Cholesky Decomposition of a Covariance Matrix," ICSA Technical Report #275-025-024, Rice University, Houston, Texas, May, 1976

APPENDIX

Program Listings of CONOPT

```
       COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                          SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002            SUBROUTINE CONOPT(N,DELF,FUNCT,X,WKA,TMITER,TSTC,TCRC,TIDN,TDH,
                  1 TIOUT,TDSNT,A,RSTRT,USELAM,XLAM)
ISN 0003            IMPLICIT REAL*8 (A-H,O-Z)
              C
              C     THIS ROUTINE MINIMIZES F(X) S.T. X(I).GE.0 & SUM(X(I))=1.
              C     FOR A DETAILED DESCRIPTION OF THE PROFRAM SEE ICSA TECHNICAL
              C     REPORT NO. 275-025-029, RICE UNIV.,HOUSTON,TEX, JUNE, 1976.
              C
              C     FUNCT IS A FUNCTION WHICH RETURNS F(X), ITS ARGUMENTS ARE X,N
              C     DELF IS A SUBROUTINE WHICH COMPUTES THE GRADIENT OF F(X).   ITS
              C     ARGUMENTS ARE OF (THE GRAD), X, & N
              C     DELF & FUNCT MUST BE DECLARED EXTERNAL IN THE MAIN PROGRAM.
              C     WKA IS A WORK AREA OF LENGTH .GE. .5*N**2+(23*N+7)/2
              C     X ON RETURN IS THE ARGUMENT AT THE MINIMUM
              C     N IS THE DIMENSION OF X
              C     A(1) & A(2) ARE USED IN THE CONSTRAINT REJECTION TEST. AS
              C     A(1)*DNORM**A(2)  DEFAULTS ARE A(1)=1. & A(2)=.2 . IF THE USER
              C     WISHES A(2) TO =0, HE SHOULD SET A(1) TO THE NEGATIVE OF WHAT HE
              C     WANTS FOR A(1).
              C     RSTRT - A LOGICAL VARIABLE INDICATING WHETHER THE USER IS
              C     RESTARTING.  IF SO, THE X'S MUST BE SUPPLIED & THE CONSTRAINT
              C     REJECTION INDICATORS AS INTEGER*2 VARIABLES SHOULD BE IN THE FIRST
              C     N LOCATIONS OF WKA
              C     USELAM - A LOGICAL VARIABLE INDICATING WHETHER OR NOT THE USER
              C     IS SUPPLYING A VALUE FOR XLAM IN THE CASE OF A RESTART.
              C     XLAM - THE LAGRANGE MULTIPLIER FOR THE EQUALITY CONSTRAINT.
              C
ISN 0004            REAL*8 X(N),WKA(1)
ISN 0005            REAL*4 TSTC,TCRC,TDH
ISN 0006            REAL*4 STC,CRC,DH
ISN 0007            REAL*4 A(2)
ISN 0008            INTEGER*4 TMITER,TIDN,TIOUT
ISN 0009            LOGICAL*1 DSNT,       TDSNT,RSTR,USELAM
ISN 0010            COMMON /INPUT/ MITER,STC,CRC,IDN,DH,IOUT,DSNT
ISN 0011            EXTERNAL DELF,FUNCT
ISN 0012            COMMON /ARANGE/ IICR,IY,IH,IDF,IGL,     IGL2,IDY,IWK,IGY,IYSAV,
                  1 IHGL,IHGY,IDLY,IDS
              C
              C     COMPUTE THE STARTING ADDRESSES OF VARIOUS WORKING ARRRYS.
              C
ISN 0013            CALL DIMS(N)
              C
              C     SET DEFAULT VALUES & OUTPUT (STEP 1)
              C
ISN 0014            CALL INPT(TMITER,TSTC,TCRC,TIDN, TDH,TIOUT,TDSNT,N,A,RSTRT)
              C
              C      INITIALIZE VARIABLES
              C
ISN 0015            CALL INIT(X,WKA(IDF),WKA(IY),WKA(IH),WKA(IGL),WKA(IICR),XLAM,
                  1 NITER,N,DELF,RSTRT,USELAM)
              C
              C     STEP 2 - CHECK STOPPING CRITERION & OUTPUT IF REQUESTED
              C
ISN 0016        100 CALL CHKSTP (X,N,G,DNORM,NITER,XLAM,WKA(IGL),
                  1          FUNCT,     &200)
              C
```

```
              C      STEP 3 - CALCULATE DISCRETE HESSIAN (OPTIONAL)
              C
ISN 0017             CALL DHESS (    NITER,WKA(IH),          WKA(IWK),WKA(IGL),
                    1 WKA(IGL2),  DELF,  XLAM,WKA(IICR),WKA(IY),N,X)
              C
              C      STEPS 4 & 5 - CALCULATE MULTIPLIER & Y-VARIABLE CORRECTIONS
              C
ISN 0018             CALL DELAMY (WKA(IGY),WKA(IH),WKA(IGL),N,WKA(IHGL),WKA(IHGY),
                    1 DLAM,WKA(IDLY),G,WKA(IY),WKA(IICR))
              C
              C      STEP 6 - CALCULATE NEW Y BASED ON (OPTIONAL ) D%SCENT
              C
ISN 0019             CALL NEWY (DNORM,XLAM,WKA(IY),N,DLAM,WKA(IDLY),G,NITER,WKA(IICR),
                    1 WKA(IYSAV),X,WKA(IDF),DELF,WKA(IDY),WKA(IGL))
              C
              C      STEP 7 - UPDATE APPROXIMATE INVERSE HESSIAN
              C
ISN 0020             CALL UPDTH(NITER,WKA(IYSAV),N,          WKA(IGL),XLAM,WKA(IICR),
                    1 WKA(IY),WKA(IH),          WKA(IGL2),WKA(IDY),WKA(IDS),WKA(IWK))
              C
              C      STEP 8 - UPDATE CONSTRAINT REJECTION INDICATORS
              C
ISN 0021             CALL CONSTR(WKA(IDF),X,N,WKA(IICR),XLAM,DELF,WKA(IGL),WKA(IY),
                    1           WKA(IYSAV),DNORM,A)
ISN 0022             GO TO 100
ISN 0023         200 RETURN
ISN 0024             END
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                           SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002            SUBROUTINE DIMS(N)
              C
              C     COMPUTE STARTING POINTS OF WORKING ARRAYS USED.  THE ORDER IS ICR,
              C      Y,H, DF,GL,GL2,DY, & WK
              C
ISN 0003            INTEGER*4 IDM(8)
ISN 0004            COMMON /ARANGE/ IDM,IGY,IYSAV,IHGL,IHGY,IDLY,IDS
ISN 0005            IDM(1)=1
ISN 0006            IDM(2)=(N+1)/2
ISN 0007            IDM(3)=N
ISN 0008            IDM(4)=N*(N+1)/2
ISN 0009            IDM(5)=N
ISN 0010            IDM(6)=N
ISN 0011            IDM(7)=N
ISN 0012            IDM(8)=N
ISN 0013            DO 10 I=2,8
ISN 0014         10 IDM(I)=IDM(I)+IDM(I-1)
ISN 0015            IGY=IDM(8)
ISN 0016            IYSAV=IGY+N
ISN 0017            IHGL=IYSAV+N
ISN 0018            IHGY=IHGL+N
ISN 0019            IDLY=IHGY+N
ISN 0020            IDS=IDLY+N
ISN 0021            RETURN
ISN 0022            END
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                           SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
 ISN 0002            SUBROUTINE INPT (TMITER,TSTC,TCRC,TIDN,TDH,TIOUT,TDSNT,N,A,RSTRT)
               C
               C     SET DEFAULT VALUES FOR PARAMETERS USED.
               C
 ISN 0003            LOGICAL*1 DSNT,TDSNT,RSTRT
 ISN 0004            INTEGER*4 TMITER,TIDN,TIOUT
 ISN 0005            REAL*4 A(2)
 ISN 0006            REAL*4 TSTC,TCRC,TDH
 ISN 0007            REAL*4 STC,CRC,DH
 ISN 0008            COMMON /INPUT/ MITER,STC,CRC,IDN,DH,IOUT,DSNT
 ISN 0009            NAMELIST /IN/ MITER,STC,CRC,IDN,DH,IOUT,DSNT
 ISN 0010            EPS=1.E-20
 ISN 0011            MITER=MAX0(30,2*N)
 ISN 0012            IF (TMITER.GT.0) MITER=TMITER
 ISN 0014            STC=1.E-8
 ISN 0015            IF (TSTC.GT.EPS) STC=TSTC
 ISN 0017            CRC=.01
 ISN 0018            IF (TCRC.GT.EPS) CRC=TCRC
 ISN 0020            IDN=MINO(10,N)
 ISN 0021            IF (TIDN.GE.0) IDN=TIDN
 ISN 0023            DH=.01
 ISN 0024            IF (TDH.GT.EPS) DH=TDH
 ISN 0026            IOUT=1
 ISN 0027            IF (TIOUT.GE.0) IOUT=TIOUT
 ISN 0029            DSNT=TDSNT
 ISN 0030            IF (A(2) .LE.EPS) A(2)=.2
 ISN 0032            IF (A(1).LT.0.) A(2)=0.
 ISN 0034            A(1)=ABS(A(1))
 ISN 0035            IF (A(1).LE.EPS) A(1)=1.
 ISN 0037            WRITE (6,11)
 ISN 0038         11 FORMAT ('1 PARAMETERS USED')
 ISN 0039            WRITE (6,IN)
 ISN 0040            WRITE (6,12) A
 ISN 0041         12 FORMAT (' A(1)=',G16.8,' A(2)=',G16.8)
 ISN 0042            WRITE (6,13) RSTRT
 ISN 0043         13 FORMAT (' RSTRT=',L8)
 ISN 0044            RETURN
 ISN 0045            END
```

```
      COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                         SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002         SUBROUTINE INIT(X,DF,Y,H,GL,ICR,XLAM,NITER,N,DELF,RSTRT,USELAM)
ISN 0003         IMPLICIT REAL*8 (A-H,O-Z)
           C
           C     THIS ROUTINE INITIALIZES SEVERAL VARIABLES USED IN THE OPTIMIZATION
           C
ISN 0004         LOGICAL*1 USELAM
ISN 0005         LOGICAL*1 RSTRT
ISN 0006         INTEGER*2 ICR(N)
ISN 0007         REAL*8 X(N),DF(N),Y(N),GL(N),H(1)
ISN 0008         IF (RSTRT) GO TO 5
ISN 0010         SN=SQRT(2./N)
ISN 0011         DO 10 I=1,N
ISN 0012         ICR(I)=2
ISN 0013         Y(I)=SN
ISN 0014      10 X(I)=1./N
ISN 0015         GO TO 15
ISN 0016       5 DO 40 I=1,N
ISN 0017         Y(I)=X(I)
ISN 0018         IF (ICR(I).EQ.1) GO TO 40
ISN 0020         ICR(I)=2
ISN 0021         Y(I)=DSQRT(2.*X(I))
ISN 0022      40 CONTINUE
ISN 0023      15 CONTINUE
           C
           C     DF IS THE GRADIENT OF F
           C
ISN 0024         CALL DELF(DF,X,N)
           C
           C     SET H TO THE IDENTITY MATRIX
           C
ISN 0025         N2=N*(N+1)/2
ISN 0026         DO 20 K=1,N2
ISN 0027      20 H(K)=0.
ISN 0028         I=0
ISN 0029         DO 30 K=1,N
ISN 0030         I=I+K
ISN 0031      30 H(I)=1.
ISN 0032         IF (.NOT.USELAM) XLAM=DOT(DF,X,N)
ISN 0034         NITER=0
           C
           C     GL IS THE GRADIENT OF THE LAGRANGIAN
           C
ISN 0035         CALL GRADL(GL,DF,XLAM,ICR,Y,N)
ISN 0036         RETURN
ISN 0037         END
```

```
          COMPILER OPTIONS - NAME=   MAIN.OPT=00.LINECNT=60.SIZE=0000K.
                             SOURCE.EBCDIC.NOLIST.NODECK.LOAD.MAP.NOEDIT.ID.XREF
  ISN 0002           SUBROUTINE CHKSTP(X,N,G.DNORM.NITER.XLAM.GL.         FUNCT.
                    1*)
  ISN 0003           IMPLICIT REAL*8 (A-H,O-Z)
  ISN 0004           REAL*8 SUM.SUM2
  ISN 0005           REAL*8 X(N).GL(N)
  ISN 0006           REAL*4 STC.CRC.DH
  ISN 0007           COMMON /INPUT/ MITER.STC.CRC.IDN.DH.IOUT
              C
              C      EVALUATE THE NORM OF GRAD(L) + THE NORM OF THE EQUALITY CONSTRAINT. G.
              C
  ISN 0008           G=1.D0-SUM(X.N)
  ISN 0009           DNORM=G*G+SUM2(GL.N)
  ISN 0010           IF (DNORM.GE.STC.AND.NITER.LT.MITER) GO TO 15
  ISN 0012           IF (IOUT.EQ.0) RETURN 1
  ISN 0014           F=FUNCT(X.N)
  ISN 0015           WRITE (6.11) NITER.DNORM.G.XLAM.F.(X(I).I=1.N)
  ISN 0016        11 FORMAT (//' NITER.DNORM.G.LAM.F.& X'.I10.4G16.8.(/8G16.8))
  ISN 0017           RETUR' 1
              C
              C      INTERMEDIATE OUTPUT IF DESIRED
              C
  ISN 0018        15 IF (IOUT.EQ.0) GO TO 25
  ISN 0020           IF (NITER/IOUT*IOUT.NE.NITER) GO TO 25
  ISN 0022           F=FUNCT(X.N)
  ISN 0023           WRITE (6.11) NITER.DNORM.G.XLAM.F.(X(I).I=1.N)
  ISN 0024        25 NITER=NITER+1
  ISN 0025           RETURN
  ISN 0026           END
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                           SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
 ISN 0002             SUBROUTINE DHESS(    NITER,H,    WK,GL,GL2,DELF, XLAM,ICR,Y,N,X)
 ISN 0003             IMPLICIT REAL*8 (A-H,O-Z)
               C
               C      CALCULATE DISCRETE APPROX. TO THE HESSIAN IF DESIRED
               C
 ISN 0004             REAL*8 WK(1),H(1),          GL(N),GL2(N),      Y(N),X(N)
 ISN 0005             INTEGER*2 ICR(N)
 ISN 0006             REAL*4 STC,CRC,DH
 ISN 0007             COMMON /INPUT/ MITER,STC,CRC,IDN,DH
 ISN 0008             IF (IDN.EQ.0) RETURN
 ISN 0010             IF ((NITER-1)/IDN*IDN.NE.NITER-1) RETURN
               C
               C      COMPUTE THE HESSIAN
               C
 ISN 0012             K=0
 ISN 0013             DO 10 J=1,N
 ISN 0014             YT=Y(J)
 ISN 0015             Y(J)=Y(J)+DH
 ISN 0016             XT=X(J)
 ISN 0017             X(J)=Y(J)**ICR(J)/ICR(J)
 ISN 0018             CALL DELF(WK,X,N)
 ISN 0019             X(J)=XT
 ISN 0020             CALL GRADL(GL2,WK,XLAM,ICR,Y,N)
 ISN 0021             Y(J)=YT
 ISN 0022             DO 10 I=1,J
 ISN 0023             K=K+1
 ISN 0024             H (K)   =(GL2(I)-GL(I))/DH
 ISN 0025          10 CONTINUE
               C
               C      COMPUTE THE MODIFIED CHOLESKY DECOMP. OF H & STORE IN H
               C
 ISN 0026             CALL MCHLSK(H,N,WK)
 ISN 0027             RETURN
 ISN 0028             END
```

```
          COMPILER OPTIONS - NAME=   MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                             SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002              SUBROUTINE DELAMY(GY,H,GL,N,HGL,HGY,DLAM,DLY,G,Y,ICR)
ISN 0003              IMPLICIT REAL*8 (A-H,O-Z)
             C
             C        COMPUTE DELTA-LAMBDA & DELTA-Y
             C
ISN 0004              REAL*8 GY(N),H(1),  GL(N),HGL(N),HGY(N),DLY(N),Y(N)
ISN 0005              INTEGER*2 ICR(N)
ISN 0006              DO 10 I=1,N
ISN 0007              GY(I)=-Y(I)
ISN 0008           10 IF (ICR(I).NE.2) GY(I)=-1.
ISN 0010              CALL PDSOLV(H,HGL,GL,N)
ISN 0011              CALL PDSOLV(H,HGY,GY,N)
ISN 0012              DLAM=(G-DOT(GY,HGL,N))/DOT(GY,HGY,N)
ISN 0013              DO 20 I=1,N
ISN 0014           20 DLY(I)=-HGL(I)-DLAM*HGY(I)
ISN 0015              RETURN
ISN 0016              END
```

```
         COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                            SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
  ISN 0002            SUBROUTINE NEWY (DNORM,XLAM,Y,N,DLAM,DELY,G,NITER,ICR,YSAV,X,
                     1 DF,DELF,DFSAV,GL)
  ISN 0003            IMPLICIT REAL*8 (A-H,O-Z)
                C
                C     COMPUTE A NEW VALUE OF Y BASED (OPTIONALLY) ON THE DESCENT OF THE
                C     NORM OF GRAD(L) + NORM OF G.
                C
  ISN 0004            REAL*8 SUM2
  ISN 0005            REAL*8 SUM
  ISN 0006            LOGICAL*1 DSNT,DSNT2
  ISN 0007            REAL*8 GL(N),DFSAV(N)
  ISN 0008            REAL*8 Y(N),DELY(N),YSAV(N),X(N),DF(N)
  ISN 0009            REAL*4 STC,CRC,DH
  ISN 0010            INTEGER*2 ICR(N)
  ISN 0011            COMMON /INPUT/ NITER,STC,CRC,IDN,DH,IOUT,DSNT
  ISN 0012            DSNT2=DSNT
  ISN 0013            ALP=1.
  ISN 0014            DNSAVE=DNORM
  ISN 0015            XLSAV=XLAM
  ISN 0016            DO 10 I=1,N
  ISN 0017            DFSAV(I)=DF(I)
  ISN 0018         10 YSAV(I)=Y(I)
  ISN 0019         15 DO 20 I=1,10
  ISN 0020            XLAM=XLSAV+ALP*DLAM
  ISN 0021            DO 30 J=1,N
  ISN 0022            Y(J)=YSAV(J)+ALP*DELY(J)
  ISN 0023            IF (ICR(J).EQ.1.AND.Y(J).LT.0.) Y(J)=0.
  ISN 0025            X(J)=Y(J)**ICR(J)/ICR(J)
  ISN 0026         30 CONTINUE
  ISN 0027            CALL DELF(DF,X,N)
                C
                C     DESCEND IF DESIRED
                C
  ISN 0028            G=1.D0-SUM(X,N)
  ISN 0029            CALL GRADL(GL,DF,XLAM,ICR,Y,N)
  ISN 0030            IF (.NOT.DSNT2) RETURN
  ISN 0032            DNORM=G*G+SUM2(GL,N)
  ISN 0033            IF (DNORM.LT.DNSAVE) RETURN
  ISN 0035         20 ALP=ALP/2.
  ISN 0036            IF (ALP.LT.0.) GO TO 25
                C
                C     TRY OTHER DIRECTION
                C
  ISN 0038            ALP=-1.
  ISN 0039            GO TO 15
  ISN 0040         25 WRITE (6,11) NITER
  ISN 0041         11 FORMAT (' *ON ITER',I5,' DESCENT COULD NOT BE IMPLEMENTED, STEP LE
                     1NGTH SET TO 1')
  ISN 0042            ALP=1.
  ISN 0043            DSNT2=.FALSE.
  ISN 0044            GO TO15
  ISN 0045            RETURN
  ISN 0046            END
```

```
        COMPILER OPTIONS - NAME=   MAIN.OPT=00.LINECNT=60.SIZE=0000K.
                           SOURCE.EBCDIC.NOLIST.NODECK.LOAD.MAP.NOEDIT.ID.XREF
 ISN 0002            SUBROUTINE UPDTH(NITER,YSAV.N.      GL.XLAM.ICR.Y.H.    GL2.DY.DS.T)
 ISN 0003            IMPLICIT REAL*8 (A-H.O-Z)
              C
              C      UPDATE APPROX. INVERSE HESSIAN
              C
 ISN 0004            REAL*8 H(1).            YSAV(N).              GL(N).Y(N)
 ISN 0005            REAL*8 GL2(N).DY(N).DS(N)
 ISN 0006            REAL*8 T(1)
 ISN 0007            REAL*4 STC.CRC
 ISN 0008            INTEGER*2 ICR(N)
 ISN 0009            COMMON /INPUT/ MITER.STC.CRC.IDN
 ISN 0010            IF (IDN.EQ.0) GO TO 15
 ISN 0012            IF (NITER/IDN*IDN.EQ.NITER) RETURN
              C
              C      CALCULATE INTERMEDIATE QUANTITIES
              C
 ISN 0014         15 CONTINUE
 ISN 0015            E1=1.E-16
 ISN 0016            E2=.1
 ISN 0017            CALL GRADL(GL2.DY.XLAM.ICR.YSAV.N)
 ISN 0018            DO 10 I=1.N
 ISN 0019            DY(I)=GL(I)-GL2(I)
 ISN 0020            DS(I)=Y(I)-YSAV(I)
 ISN 0021         10 CONTINUE
 ISN 0022            D=DOT(DY.DS.N)
 ISN 0023            IF (DABS(D) .GT.E1) GO TO 25
 ISN 0025            WRITE (6.12)
 ISN 0026         12 FORMAT (' D NEAR 0 IN UPDTH')
 ISN 0027            RETURN
 ISN 0028         25 CONTINUE
              C
              C      COMPUTE THE PRODUCT H*DS & STORE IN GL2
              C
 ISN 0029            CALL LDMUL(H.DS.GL2. T.N)
              C
              C      UPDATE H
              C
 ISN 0030            TX=-DOT(DS.GL2.N)
 ISN 0031            T(1)=D
 ISN 0032            CALL COMPT(H.T.DY.N.T(N+2).T(2*N+2))
 ISN 0033            T(1)=TX
 ISN 0034            CALL COMPT(H.T.GL2.N.T(N+2).T(2*N+2))
              C
              C      INSURE THAT H IS POS. DEFN.
              C
 ISN 0035            II=0
 ISN 0036            DO 20 I=1.N
 ISN 0037            II=II+I
 ISN 0038            IF (H(II).LE.E1) H(II)=E2
 ISN 0040         20 CONTINUE
 ISN 0041            RETURN
 ISN 0042            END
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                           SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002          SUBROUTINE CONSTR(DF,X,N,ICR,XLAM,DELF,GL,Y,      YSAV,DNORM,A)
ISN 0003          IMPLICIT REAL*8 (A-H,O-Z)
            C
            C     CHECK FOR REDUNDANT CONSTRAINTS & RESET CONSTRAINT INDICATORS
            C
ISN 0004          INTEGER*2 ICR(N)
ISN 0005          REAL*8 YSAV(N)
ISN 0006          REAL*8 DF(N),X(N)
ISN 0007          REAL*8 GL(N),Y(N)
ISN 0008          REAL*4 STC,CRC,DH
ISN 0009          REAL*4 A(2)
ISN 0010          LOGICAL*1 IND
ISN 0011          COMMON /INPUT/ NITER,STC,CRC,IDN,DH,IOUT
ISN 0012          XML=1.
ISN 0013          SN=SQRT(FLOAT(N))
ISN 0014          EPS=.25
ISN 0015          IND=.FALSE.
            C
            C     TESTING & RESETTING LOOP
            C
ISN 0016          DO 10 I=1,N
ISN 0017          T=DF(I)-XLAM
ISN 0018          TS=A(1)*DNORM**A(2)
ISN 0019          CR=EPS/N
ISN 0020          CR=DMIN1(CR,TS)
ISN 0021          IF (DABS(X(I)).GE.CR) GO TO 12
ISN 0023          CR=DMIN1(CRC/SN,TS)
ISN 0024          IF (DABS(T).GE.CR) GO TO 12
ISN 0026          IF (ICR(I).NE.1) IND=.TRUE.
ISN 0028          ICR(I)=1
ISN 0029          GO TO 10
ISN 0030       12 CONTINUE
ISN 0031          IF (ICR(I).NE.2) IND=.TRUE.
ISN 0033          ICR(I)=2
ISN 0034       10 CONTINUE
ISN 0035       15 CONTINUE
ISN 0036          IF (IOUT.EQ.0) GO TO 25
ISN 0038          IF (IOUT*(NITER/IOUT).EQ.NITER) WRITE (6,21) (ICR(I),I=1,N)
ISN 0040       21 FORMAT (' ICR',20I5)
ISN 0041       25 CONTINUE
ISN 0042          IF (.NOT.IND) RETURN
ISN 0044          DO 30 I=1,N
ISN 0045          Y(I)=X(I)
ISN 0046          IF (ICR(I).NE.2) GO TO 30
ISN 0048          Y(I)=0.
ISN 0049          IF (X(I).GT.0.) Y(I)=DSQRT(2.*X(I))
ISN 0051       30 CONTINUE
ISN 0052          CALL GRADL(GL,DF,XLAM,ICR,Y,N)
ISN 0053          RETURN
ISN 0054          END
```

```
          COMPILER OPTIONS - NAME=  MAIN.OPT=00.LINECNT=60.SIZE=0000K.
                             SOURCE.EBCDIC.NOLIST.NODECK.LOAD.MAP.NOEDIT.ID.XREF
  ISN 0002              SUBROUTINE GRADL(GL.DF.XLAM.ICR.Y.N)
  ISN 0003              IMPLICIT REAL*8 (A-H,O-Z)
                  C
                  C     COMPUTE THE GRADIENT OF THE LAGRANGIAN
                  C
  ISN 0004              REAL*8 GL(N).DF(N),Y(N)
  ISN 0005              INTEGER*2 ICR(N)
  ISN 0006              DO 10 I=1.N
  ISN 0007              GY=-Y(I)
  ISN 0008              IF (ICR(I).NE.2) GY=-1.
  ISN 0010           10 GL(I)=-(DF(I)-XLAM)*GY
  ISN 0011              RETURN
  ISN 0012              END
```

```
          COMPILER OPTIONS - NAME=   MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                             SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
  ISN 0002               REAL FUNCTION SUM*8 (X,N)
  ISN 0003               IMPLICIT REAL*8 (A-H,O-Z)
                C
                C        PERFORM UTILITY FUNCTIONS
                C
  ISN 0004               REAL*8 X(N),Y(N)
  ISN 0005               REAL*8 S,SUM2
  ISN 0006               S=0.D0
  ISN 0007               DO 20 I=1,N
  ISN 0008            20 S=S+X(I)
  ISN 0009               SUM=S
  ISN 0010               RETURN
  ISN 0011               ENTRY SUM2(X,N)
  ISN 0012               S=0.D0
  ISN 0013               DO 30 I=1,N
  ISN 0014            30 S=S+X(I)*X(I)
  ISN 0015               SUM2=S
  ISN 0016               RETURN
  ISN 0017               ENTRY DOT(X,Y,N)
  ISN 0018               S=0.D0
  ISN 0019               DO 40 I=1,N
  ISN 0020            40 S=S+X(I)*Y(I)
  ISN 0021               DOT=S
  ISN 0022               RETURN
  ISN 0023               END
```

```
          COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                                SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
  ISN 0002            SUBROUTINE MCHLSK(KK,NV,DUM)
  ISN 0003            IMPLICIT REAL*8 (A-H,O-Z)
                C                                                                       DVR04770
                C   *********************************************************************DVR04780
                C         THIS ROUTINE COMPUTES THE MODIFIED CHOLESKY DECOMPOSITION
                C      (MCD) OF A SYMMETRIC MATRIX STORED IN SYM. STORAGE MODE.  THE
                C      DECOMPOSITION OVERLAYS THE ELEMENTS OF THE MATRIX.  IF THE MATRIX
                C      IS NOT POSITIVE DEFINITE, THE MCD REPRESENTS A MODIFIED MATRIX
                C      WHICH IS.  THIS IS DONE BY SETTING THE APPROPRIATE ELEMENTS OF
                C      D TO 1.
                C      SEE E.G. 'THE USE OF THE MODIFIED CHOLESKY DECOMPOSITION IN
                C      DIVERGENCE AND CLASSIFICATION CALCULATIONS',BY D.L. VAN ROOY,
                C      M.S. LYNN, & C.H. SNUDER, ICSA TECH. RPT. 275-025-008, RICE
                C      UNIV., HOUSTON,TX, MAY,1973.
                C   *********************************************************************DVR04830
                C                                                                       DVR04840
                C         KK - THE COVARIANCE MATRIX STORED IN SYMMETRIC STORAGE MODE.  DVR04850
                C         KK=L D L*                                                     DVR04820
                C            WHERRE L IS UNIT LOWER TRIANGULAR & D DIAGONAL WITH POS.
                C            ENTRIES.
                C         NV - THE NUMBER OF CHANNELS USED                              DVR04860
                C         DUM - A DOUBLE PRECISION WORK AREA OF SIZE NV-1
                C                                                                       DVR04890
  ISN 0004            REAL*8 KK(1)
  ISN 0005            REAL*8 DUM(1)
  ISN 0006            REAL*8 R,R1,T1,TF
  ISN 0007            LOGICAL*1 JE1                                                     DVR04910
  ISN 0008            JE1=.TRUE.                                                        DVR04920
  ISN 0009            EPS=.1
  ISN 0010            J1=0                                                              DVR04930
  ISN 0011            JD=0                                                              DVR04940
                C                                                                       DVR04960
                C         LOOP OVER ALL DIMENSIONS
                C                                                                       DVR04980
  ISN 0012            DO 10 J=1,NV                                                      DVR04990
  ISN 0013            KL=J-1                                                            DVR05000
  ISN 0014            L=J+1                                                             DVR05010
  ISN 0015            JD=J1                                                             DVR05020
  ISN 0016            J1=J1+J                                                           DVR05030
  ISN 0017            TF=0.D0
  ISN 0018            IF (JE1) GO TO 12                                                 DVR05050
  ISN 0020            K1=0                                                              DVR05060
                C                                                                       DVR05070
                C         COMPUTE THE DIAGONAL ELEMENTS OF D AND STORE IN KK            DVR05080
                C         TEMPORARILY STORE THE PEODUCT KK(I,I)*KK(J,I) IN DUM(I)       DVR05090
                C                                                                       DVR05100
  ISN 0021            DO 15 I=1,KL                                                      DVR05110
  ISN 0022            R=KK(JD+I)                                                        DVR05120
  ISN 0023            K1=K1+I                                                           DVR05130
  ISN 0024            R1=KK(K1)*R                                                       DVR05140
  ISN 0025            TF=TF-R1*R                                                        DVR05150
  ISN 0026            DUM(I)=R1                                                         DVR05160
  ISN 0027         15 CONTINUE                                                          DVR05170
  ISN 0028         12 CONTINUE                                                          DVR05190
  ISN 0029            TF=TF+KK(J1)
                C
```

```
            C      IF K IS NOT POS. DEF. , MAKE IT SO
            C
ISN 0030           IF (TF.LT.0.) TF=EPS
ISN 0032           KK(J1      )=TF                                            DVR05180
ISN 0033           IF (L.GT.NV) GO TO 10                                      DVR05210
ISN 0035           IRD=J1-L+1                                                 DVR05220
            C                                                                 DVR05230
            C         COMPUTE THE R,J-TH ELEMENT OF L USING T1                DVR05240
            C                                                                 DVR05250
ISN 0036           DO 20 IR=L,NV                                              DVR05260
ISN 0037           IRD=IRD+IR-1                                               DVR05270
ISN 0038           T1=0.D0
ISN 0039           IF (JE1) GO TO 16                                          DVR05290
ISN 0041           DO 25 I=1,KL                                               DVR05300
ISN 0042           T1=T1-DUM(I)*KK(IRD+I)                                     DVR05310
ISN 0043        25 CONTINUE                                                   DVR05320
ISN 0044        16 KK(IRD+J)=(T1+KK(IRD+J))/TF
ISN 0045        20 CONTINUE                                                   DVR05340
ISN 0046           JE1=.FALSE.                                                DVR05350
ISN 0047        10 CONTINUE                                                   DVR05360
ISN 0048           RETURN
ISN 0049           END                                                        DVR05460
```

```
         COMPILER OPTIONS - NAME=   MAIN.OPT=00,LINECNT=60,SIZE=0000K,
                                SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
  ISN 0002               SUBROUTINE PDSOLV(A,X,Y,N)
  ISN 0003               IMPLICIT REAL*8 (A-H,O-Z)
  ISN 0004               REAL*8 A(1),X(N),Y(N)
  ISN 0005               REAL*8 S
                   C
                   C     SOLVE A*X=Y WHERE A IS A SYMMETRICALLY STORED MATRIX CONTAINING
                   C     THE MODIFIED CHOLESKY DECOMPOSITION OF  ANOTHER MATRIX.
                   C
  ISN 0006               ISB(J,I)=J*(J-1)/2+I
                   C
                   C     SOLVE L*Z=Y & PUT RESULT IN X
                   C
  ISN 0007               K=0
  ISN 0008               DO 10 I=1,N
  ISN 0009               S=Y(I)
  ISN 0010               IF (I.EQ.1) GO TO 15
  ISN 0012               I1=I-1
  ISN 0013               DO 20 J=1,I1
  ISN 0014               K=K+1
  ISN 0015            20 S=S-A(K)*X(J)
  ISN 0016            15 K=K+1
  ISN 0017            10 X(I)=S
                   C
                   C     SOLVE D*B=Z FOLLOWDD BY L-TRANSPOSE * X=B & OVERWRITE RESULT ON X
                   C
  ISN 0018               LL=N*(N+1)/2
  ISN 0019               X(N)=X(N)/A(LL)
  ISN 0020               IF (N.LE.1) RETURN
  ISN 0022               DO 30 II=2,N
  ISN 0023               I=N-II+1
  ISN 0024               LL=LL-I-1
  ISN 0025               X(I)=X(I)/A(LL)
  ISN 0026               I1=I+1
  ISN 0027               S=X(I)
  ISN 0028               DO 40 J=I1,N
  ISN 0029               L=ISB(J,I)
  ISN 0030            40 S=S-A(L)*X(J)
  ISN 0031            30 X(I)=S
  ISN 0032               RETURN
  ISN 0033               END
```

```
          COMPILER OPTIONS - NAME=   MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                                SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
  ISN 0002              SUBROUTINE COMPT (LD,T,Z,N,  V,TMP)
  ISN 0003              IMPLICIT REAL*8 (A-H,O-Z)
                  C
                  C     THIS ROUTINE IS AN IMPLEMENTATION OF THE COMPOSITE - T ALGORITHM
                  C     TO PRRFORM A RANK 1 UPDATE OF THE MCD STORED IN ARRAY LD(I.E.
                  C     K=L*D*L-TRANS & WE WISH TO COMPUTE L' &D' S.T.K'=K+Z*Z-TRANS/T(1)
                  C     & K'=L'*D'*L'-TRANS).
                  C     THE COMPOSITE-T ALGORITHM WAS DEVELOPED BY FLETCHER & POWELL
                  C     SEE 'ON THE COMPUTATION & UPDATING OF THE MODIFIED CHOLESKY
                  C     DECOMPOSITION OF A COVARIANCE MATRIX,' BY D.L. VAN ROOY, ICSA
                  C     TECH.RPT. 275-025-024,RICE UNIV., HOUSTON, TX. MAY, 1976.
                  C
  ISN 0004              REAL*8 LD(1),Z(N)
  ISN 0005              REAL*8 T(1),       V(N)
  ISN 0006              REAL*8 TMP(N)
  ISN 0007              REAL*8 S
  ISN 0008              LOGICAL*1 TPOS,RNDERR,LALP
                  C
                  C     LD - ARRAY CONTAINING L & D STORED IN SYM.STORAGE MODE
                  C     T - AN N+1 VECTOR WHOSE FIRST ELEMENT IS AS ABOVE
                  C     Z - VECTOR OF THE UPDAEE AS ABOVE
                  C     N - THE DIMENSION
                  C     V - WORKING STORAGE OF LENGTH .GE. N
                  C     TMP - DOUBLE PRECISION WORKING STORAGE OF LENGTH .GE. N
                  C
  ISN 0009              TPOS=T(1).GT.0.
  ISN 0010              IF (TPOS) GO TO 35
                  C
                  C     A POINT IS TO BE DELETED
                  C
  ISN 0012              EPS=5.97E-8
                  C
                  C     SOLVE L*V=Z FOR V
                  C
  ISN 0013              K=1
  ISN 0014              V(1)=Z(1)
  ISN 0015              DO 10 I=2,N
  ISN 0016              IJ=I-1
  ISN 0017              S=0.D0
  ISN 0018              DO 15 J=1,IJ
  ISN 0019              K=K+1
  ISN 0020           15 S=S+LD(K)*V(J)
  ISN 0021              K=K+1
  ISN 0022              V(I)=Z(I)-S
  ISN 0023           10 CONTINUE
                  C
                  C     COMPUTE THE T(I'S)
                  C
  ISN 0024              K=0
  ISN 0025              RNDERR=.FALSE.
  ISN 0026              DO 20 I=1,N
  ISN 0027              K=K+I
  ISN 0028              TMP(I)=V(I)*V(I)/LD(K)
  ISN 0029              T(I+1)=T(I)+TMP(I)
  ISN 0030              IF (T(I+1).GE.0.) RNDERR=.TRUE.
  ISN 0032           20 CONTINUE
```

```
ISN 0033          IF (.NOT.RNDERR) GO TO 35
            C
            C     ROUNDING ERROR HAS MADE A T(I+1).GE.0. SO CORRECT FOR THIS
            C
ISN 0035          T(N+1)=EPS*T(1)
ISN 0036          DO 30 J=1,N
ISN 0037          I=N-J+1
ISN 0038          T(I)=T(I+1)-TMP(I)
ISN 0039       30 CONTINUE
ISN 0040       35 CONTINUE
ISN 0041          IJ=0
ISN 0042          DO 40 I=1,N
ISN 0043          I1=I+1
ISN 0044          IJ=IJ+I
ISN 0045          V(I)=Z(I)
ISN 0046          DI=LD(IJ)
ISN 0047          IF (DI.GT.0.) GO TO 44
            C
            C     D(I) =0. SO RANK OF D WILL EITHER INCREASE OR REMAIN UNCHANGED.
            C
ISN 0049          IF (DABS(V(I)).GT.1.E-30) GO TO 42
            C
            C     RANK OF D WILL REMAIN UNCHANGED
            C
ISN 0051          T(I+1)=T(I)
ISN 0052          GO TO 40
            C
            C     RANK OF D WILL INCREASE BY 1
            C
ISN 0053       42 LD(IJ)=V(I)*V(I)/T(I)
ISN 0054          IF (I.EQ.N) RETURN
ISN 0056          K=IJ
ISN 0057          DO 45 J=I1,N
ISN 0058          K=K+J-1
ISN 0059          LD(K)=Z(J)/V(I)
ISN 0060       45 CONTINUE
ISN 0061          RETURN
ISN 0062       44 CONTINUE
            C
            C     UPDATE D
            C
ISN 0063          IF (TPOS) T(I+1)=T(I)+V(I)*V(I)/DI
ISN 0065          ALP=T(I+1)/T(I)
ISN 0066          LD(IJ)=DI*ALP
ISN 0067          IF (I.EQ.N) RETURN
            C
            C     UPDATE L & MODIFY Z ACCORDINGLY
            C
ISN 0069          BETA=(V(I)/DI)/T(I+1)
ISN 0070          LALP=.FALSE.
ISN 0071          IF (ALP.LE.4.) GO TO 52
            C
            C     THIS METHOD USED TO INSURE STABILITY IF ALPHA GT. 4
ISN 0073          LALP=.TRUE.
ISN 0074          GAM=T(I)/T(I+1)
ISN 0075          K=IJ
ISN 0076          DO 50 J=I1,N
ISN 0077          K=K+J-1
```

```
      XX=GAM*LD(K)+BETA*Z(J)
      Z(J)=Z(J)-V(I)*LD(K)
      LD(K)=XX
   50 CONTINUE
      GO TO 40
   52 K=IJ
      DO 60  J=I1,N
      K=K+J-1
      Z(J)=Z(J)-V(I)*LD(K)
      LD(K)=LD(K)+BETA*Z(J)
   60 CONTINUE
   40 CONTINUE
      RETURN
      END
```

```
         COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=60,SIZE=0000K,
                                   SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
  ISN 0002              SUBROUTINE LDMUL ( H,S,R,T,N)
  ISN 0003              IMPLICIT REAL*8 (A-H,O-Z)
                  C
                  C     MULTIPLY THE MATRIX WHOSE MCD IS STORED IN  H (IN SYM. STORAGE
                  C     MODE) BY S & STORE THE RESULT IN R
                  C     T IS WORKING STORAGE OF LENGTH .GE.N-1
                  C
  ISN 0004              REAL*8  H(1),S(N),R(N),T(N)
  ISN 0005              REAL*8 SK
  ISN 0006              II=1
  ISN 0007              DO 10 I=1,N
                  C
                  C     MULTIPLY BY L-TRANSPOSE
                  C
  ISN 0008              IS=II
  ISN 0009              SK= S(I)
  ISN 0010              IF (I.GE.N) GO TO 25
  ISN 0012              IP1=I+1
  ISN 0013              DO 20 J=IP1,N
  ISN 0014              IS=IS+J-1
  ISN 0015           20 SK=SK+S(J)*H(IS)
                  C
                  C     MULTIPLY BY D
                  C
  ISN 0016           25 SK=SK*H(II)
  ISN 0017              T(I)=SK
                  C
                  C     MULTIPLY BY L
                  C
  ISN 0018              IF (I.LE.1) GO TO 35
  ISN 0020              IM1=I-1
  ISN 0021              IJ=II-I
  ISN 0022              DO 30 J=1,IM1
  ISN 0023              IJ=IJ+1
  ISN 0024           30 SK=SK+T(J)*H(IJ)
  ISN 0025           35 R(I)=SK
  ISN 0026              II=II+I+1
  ISN 0027           10 CONTINUE
  ISN 0028              RETURN
  ISN 0029              END
```