ALGORITHM 587

Two Algorithms for the Linearly Constrained Least Squares Problem

RICHARD J. HANSON and KAREN H. HASKELL
Sandia National Laboratories

Categories and Subject Descriptors. G.1.2. [**Numerical Analysis**]: Approximation—*least squares approximation*; G.1.6. [**Numerical Analysis**]: Optimization—*constrained optimization*; *least squares methods*, G.m [**Mathematics of Computing**]. Miscellaneous—*FORTRAN*

General Terms. Algorithms

Additional Key Words and Phrases: linear least squares solution, equality constraints, inequality constraints, nonnegativity constraints, inconsistent constraints, covariance matrix

---

1. INTRODUCTION

This paper discusses subroutines for computing numerical solutions of the following two linearly constrained linear least squares problems.

Problem NNLSE

$$\begin{aligned} E\mathbf{x} &= \mathbf{f} && \text{(equations to be exactly satisfied)} \\ A\mathbf{x} &\cong \mathbf{b} && \text{(equations to be approximately satisfied, least squares sense)} \\ x_i &\geq 0, && i = l+1, \ldots, n, \quad 0 \leq l \leq n \end{aligned} \tag{1}$$

Problem LSEI

$$\begin{aligned} E\mathbf{x} &= \mathbf{f} && \text{(equations to be exactly satisfied)} \\ A\mathbf{x} &\cong \mathbf{b} && \text{(equations to be approximately satisfied, least squares sense)} \\ G\mathbf{x} &\geq \mathbf{h} && \text{(inequality constraints that the solution must satisfy)} \end{aligned} \tag{2}$$

In both problems the matrices E and A are real and of respective dimensions m_E by n and m_A by n. For Problem NNLSE, the variables $x_1, \ldots, x_l$ are free to have either sign. For Problem LSEI, the (real) inequality constraint matrix G is m_G by n. The right-side vectors **f**, **b**, and **h** that appear in the two problem statements have, respectively, m_E, m_A, and m_G components. The (unknown) solution vector **x** has n components.

While Problem LSEI of eq. (2) appears to be a more general problem than Problem NNLSE of eq. (1), it really is not. In fact, there are a number of ways to

---

Editing for this algorithm was supervised by Associate Editor W.J Cody, Jr
Received 9 February 1981, revised 19 March 1982, accepted 1 April 1982
Authors' addresses: R.J. Hanson, Numerical Mathematics Division 5642, Sandia National Laboratories, Albuquerque, NM 87185, K H. Haskell, Applied Mathematics Division 2646, Sandia National Laboratories, Albuquerque, NM 87185.

transform Problem LSEI into one of the forms of Problem NNLSE. Three ways of doing this are discussed in [3]. The method we have implemented is described on pages 101–102. The successful implementation of an algorithm for solving Problem NNLSE is the key computational process. Nevertheless, it is important for applications such as constrained curve fitting [2] to have a subprogram that solves Problem LSEI of eq. (2) directly. We provide FORTRAN subprograms **WNNLS( )** and **LSEI( )** that solve the respective problems in eqs. (1) and (2).

In Section 2 we review mathematical and numerical analysis details pertinent to solving Problem LSEI. In Section 3 we review some necessary details for understanding our methods for solving Problem NNLSE. In Section 4 we summarize some features and advantages of the codes. These features include changing tolerances, scaling of data matrices, and optional computation of the covariance matrix. Section 5 presents a test subprogram **CLSTP( )**, which is included with the package. It solves the test problem with both subprograms. Section 6 contains installation guidelines and remarks.

2. SOLVING PROBLEM LSEI

In this section, we briefly review mathematical and algorithmic details needed to solve Problem LSEI of eq. (2) [3, pp. 101–102]. The overall process consists of four main parts.

Step 1 Problem LSEI is reduced to a subproblem with possibly fewer unknown variables and with all explicitly stated equality constraints removed.

Step 2 The problem resulting from step 1 is reduced to a new problem where the least squares matrix is a simple projection matrix and the right-side vector is zero.

Step 3 The problem resulting from step 2 is solved by reposing it as a dual problem. This dual problem consists of two special cases of Problem NNLSE, eq. (1).

Step 4 The solution obtained in step 3 is transformed to the solution of the original problem using translations, matrix multiplications, and the solution of triangular linear algebraic systems.

3. SOLVING PROBLEM NNLSE

The theoretical development for solving problem NNLSE of eq. (1) is presented in [3]. The fundamental point of this method involves a numerically stable implementation of a penalty function approach. The least squares equations are each weighted by a small parameter ϵ, chosen in the subprogram **WNNLS( )**. The augmented and weighted least squares system of eq. (3) is then solved.

$$
\begin{gathered}
\begin{bmatrix} E \\ \epsilon A \end{bmatrix} \mathbf{x} \equiv D \begin{bmatrix} E \\ A \end{bmatrix} \mathbf{x} \cong D \begin{bmatrix} \mathbf{f} \\ \mathbf{b} \end{bmatrix} \equiv \begin{bmatrix} \mathbf{f} \\ \epsilon \mathbf{b} \end{bmatrix} \\
D = \operatorname{diag}(\overbrace{1, \ldots, 1}^{m_E}, \overbrace{\epsilon, \ldots, \epsilon}^{m_A}) \\
\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{w} \end{bmatrix} \begin{matrix} \}l \\ \}n-l \end{matrix} \qquad \begin{matrix} \mathbf{y} \ \text{unconstrained} \\ \mathbf{w} \geq \mathbf{0}. \end{matrix}
\end{gathered} \tag{3}
$$

Part of the theoretical development in [3] shows that solutions of the weighted problem of eq. (3) converge to solutions of Problem NNLSE (if it is consistent) as $\epsilon \to 0$. Within the subprogram **WNNLS( )** eq. (3) is solved only once with a value of ϵ that is chosen to achieve full working accuracy in the solution. The value used in **WNNLS( )** is defined by

$$\epsilon^2 = \frac{10^{-4}\eta}{\gamma}, \tag{4}$$

where $\gamma = \left\|\begin{smallmatrix}E\\A\end{smallmatrix}\right\|$, ($\|\cdot\|$ = subordinate matrix norm of l_∞ vector norm), and η = machine relative arithmetic precision.

The algorithm for solving eq. (3) with ϵ as defined in eq. (4) proceeds in two main steps. First we compute a (minimum-length) solution for the unconstrained variables in terms of the constrained variables. Solving for the unconstrained variables is primarily a triangularization operation. In the second main step of the process we solve for the constrained variables. This is an iterative process, that is, it is Algorithm NNLS of [7, Chap. 23]. Certain crucial differences in numerical tests are needed because of the penalty parameter ϵ that multiplies the least squares equations. These tests are discussed in [3].

4. USAGE SUGGESTIONS AND SUBPROGRAM OPTIONS

In Sections 2 and 3 we have outlined solution methods for solving Problem LSEI of eq. (2) and Problem NNLSE of eq. (1). As shown in [3], computing the solution of Problem NNLSE can be regarded as the core computation in solving constrained linear least squares problems.

The most satisfactory method from the standpoint of accuracy and stability is to introduce slack variables into the inequality constraints of Problem LSEI [3]. This problem is then solved using subprogram **WNNLS( )**. The results of solving a bounded variable Hilbert matrix problem summarized in [3] suggest that subprogram **WNNLS( )** continues to compute acceptable solutions even as the problems become increasingly ill-conditioned.

The use of subprogram **WNNLS( )** with the slack variable formulation does have a disadvantage compared to subprogram **LSEI( )**. For most problems, **WNNLS( )** will require more computing time and storage than **LSEI( )**. This is due to the larger number of problem variables in the slack variable formulation. The advantage of efficiency with **LSEI( )** may be countered by the simultaneous occurrence of poor conditioning and rounding errors. (This can occur with a poorly conditioned least squares problem.) Owing to the poor conditioning and rounding error, the feasible constraint region can be mapped to one that is infeasible. Instances of this are shown in the results of solving the bounded variable Hilbert matrix problem summarized in [3].

The choice between the two subprograms is a time and storage versus stability trade-off. Specifically, in the case of a poorly conditioned least squares problem, **WNNLS( )** might obtain a solution when **LSEI( )** cannot. As illustrated in [3], subprogram **WNNLS( )** can also be used to extend the notion of solution for problems with infeasible constraints.

Occasionally, a user of subprogram **LSEI( )** will need the covariance matrix of the least squares solution variables of minimum length. This is returned as an output matrix if the user wants it. It is an unbiased estimate of the covariance matrix for the minimum-length solution of an equality constrained least squares problem *with no inequalities.* This is developed in [4] and [6].

When inequalities are included, certain additional mathematical problems must be considered. These have to do with the behavior of the set of inequalities chosen by the algorithm to be equalities. The question is as follows: What is the sensitivity of these equalities as the data are allowed to vary within its uncertainty? Inequalities may move from being satisfied as equalities to strict inequalities as the data are perturbed. The covariance matrix computed by **LSEI( )** is based on the assumption that the set of equalities does not change when the solution is perturbed. No comprehensive theory is known to the authors for determining the matrix when the set of equalities does change. The user must keep these facts in mind when interpreting the covariance matrix for Problem LSEI with inequalities.

The remainder of this section describes parameters within **LSEI( )** and **WNNLS( )** which can optionally be changed by the user. These options fall into the three following groups.

(A) Computation of the covariance matrix.
(B) Column scaling of the data matrix.
(C) Redefinition of tolerances used for determining ranks of problem matrices.

Changes to any number of these parameters can be specified as the linked-list input in the array **PRGOPT(*)**. Precise instructions for defining **PRGOPT(*)** are found in the usage prologues for **LSEI( )** and **WNNLS( )**. If the user is satisfied with the nominal subprogram features, it is only necessary to set **PRGOPT(1)=1.**

Remarks about A: Nominally the covariance matrix is not computed by **LSEI( )**.

Remarks about B: Column scaling of the form $\mathbf{x} = D\mathbf{y}$ is always performed by **LSEI( )** and **WNNLS( )**. Nominally D is the identity matrix. Another option here is a choice for D such that each nonzero column of the entire scaled data matrix has length one. The user can also specify an arbitrary D.

Remarks about C: The user can change tolerances t_E and t_A in **LSEI( )** and tolerance t_W in **WNNLS( )**. The nominal values of t_E, t_A, and t_W are $\eta^{1/2}$, where η is the relative arithmetic precision of the machine. The parameter t_E is used in approximating the rank of the equality constraint matrix E of eq. (2). Its role is discussed near the end of [3, Sec. 1].

The parameter t_A is used in approximating the rank of the least squares matrix that results from eliminating the equality constraints from eq. (2). It is used to compute the factor τ, which is t_A times the norm of this reduced least squares matrix. Then τ is used in Algorithm HFTI [7, Chap. 14].

The parameter t_W is used by **WNNLS( )** to compute the rank of the row-scaled least squares matrix as discussed in [3, Sec. 3.1].

5. REMARKS ON THE TESTING SUBPROGRAM **CLSTP( )**

The subprogram **CLSTP (KLOG, COND, ISTAT)** constructs and solves a constrained least squares problem that has a known solution and known condition numbers [7, Chap. 9]. The problem generated is stated in eq. (2). The matrices A, E, and G are computed using formulas

$$A = U_1 S_1 V_1^{\mathrm{T}}$$

$$E = U_2 S_2 V_2^{\mathrm{T}}$$

and

$$G = U_3 S_3 V_3^{\mathrm{T}}.$$

The problem dimensions are specified by using five integer parameters k_A, k_E, k_G, k_I, and k_n to compute $m_A = 2^{k_A}$, $m_E = 2^{k_E}$, $m_G = 2^{k_G}$, and $n = 2^{k_n}$. The integer $m_I = 2^{k_I}$ denotes the number of inequality constraints that are to be satisfied as strict inequalities. These five integers are passed to **CLSTP( )** in the array **KLOG(*)** in the order indicated. If any of the values k_A, k_E, k_G, k_I, or k_n are less than zero, the respective values m_A, m_E, m_G, m_I, or m_n are set to zero. No computation is performed if $n = 0$.

Arrays within **CLSTP( )** currently have fixed dimensions that require k_A, k_E, k_G, k_I, and k_n to all be less than or equal to 5. Instructions for increasing the array dimensions are given as comments within **CLSTP( )**.

The matrices U_J and V_J are symmetric orthogonal Hadamard matrices of dimension $n = 2^k$ generated by the recursion

$$
\begin{aligned}
&n := 1\\
&U := 1\\
&\text{For} \quad l = 1, \ldots, k\\
&\qquad U := \begin{bmatrix} U & : & U \\ U & : & -U \end{bmatrix}\\
&\qquad n := n + n\\
&\text{End For}\\
&U := n^{-1/2} U
\end{aligned}
$$

The matrices S_J, $J = 1, 2, 3$, are rectangular diagonal matrices. The extreme diagonal terms are κ_ι and 1, where $\kappa_J \equiv$ **COND(J)**. The intermediate diagonal terms are generated in the open interval $(1, \kappa_J)$ using the random number generator **RAN( )**. The output value of $t =$ **RAN(ISEED)** satisfies $0 < t < 1$. The intermediate diagonal terms are successively computed as $1 + t(\kappa_J - 1)$. Initially, **ISEED** is set to 100001 in **CLSTP( )**.

The n-vector $\hat{\mathbf{x}} = (1, \ldots, 1)^{\mathrm{T}}$ is used to generate the vectors

$$\mathbf{f} = E\hat{\mathbf{x}}$$

$$\hat{\mathbf{b}} = A\hat{\mathbf{x}}$$

and

$$\hat{\mathbf{h}} = G\hat{\mathbf{x}}.$$

We add a vector $\tilde{\mathbf{b}}$ to $\hat{\mathbf{b}}$ that is orthogonal to the column space of A. This is given by

$$\tilde{\mathbf{b}} = U_1(0, \ldots, 0, g_{n+1}, \ldots, g_{m_A}),$$

where

$$g_i = \mathbf{RAN(ISEED)} \cdot \|\hat{\mathbf{b}}\| \cdot \sigma, \qquad i = n+1, \ldots, m_A.$$

The value of σ is specified by the variable **ANSR** in **CLSTP( )**. It is currently set to 0.01. The right-side vector for the least squares equations in eq. (2) is $\mathbf{b} \equiv \hat{\mathbf{b}} + \tilde{\mathbf{b}}$.

The right-side vector for the inequality constraints is constructed by making the first m_I constraints strict inequalities. This is done by defining the right-side vector as $\mathbf{h} = \hat{\mathbf{h}} - \tilde{\mathbf{h}}$, where

$$\tilde{\mathbf{h}} = (h_1, \ldots, h_{m_I}, 0, \ldots, 0^{\mathrm{T}}),$$

$$h_i = \mathbf{RAN(ISEED)} \cdot \|\hat{\mathbf{h}}\|, \qquad i = 1, \ldots, m_I.$$

These techniques for generating problems with known solutions are similar to those discussed in [9, pp. 6–9]. One might obtain different sets of test problems on machines with differing arithmetic characteristics. Part of this is due to a different sequence of numbers generated by **RAN( )**.

We have found that column scaling is sometimes required for solving eqs. (1) and (2). In particular, when using 32-bit floating-point arithmetic, problems generated by **CLSTP( )** using the published test data occasionally failed to pass the tests when no column scaling was done. Thus the option array input for calls to both **LSEI( )** and **WNNLS( )** are set so that unit length column scaling is performed on all the tests.

After subprogram **LSEI( )** has computed an approximate solution $\mathbf{x}'$ for this particular form of eq. (2), and subprogram **WNNLS( )** has solved for an approximate solution $\mathbf{x}''$ of the system

$$E\mathbf{x} = \mathbf{f}$$

$$A\mathbf{x} \cong \mathbf{b}$$

$$G\mathbf{x} - \mathbf{h} = \mathbf{w}$$

for the unknown $(\mathbf{x}^{\mathrm{T}}, \mathbf{w}^{\mathrm{T}})^{\mathrm{T}}$, we compute the differences $\mathbf{dx}_1 = \mathbf{x}' - \hat{\mathbf{x}}$ and $\mathbf{dx}_2 = \mathbf{x}'' - \hat{\mathbf{x}}$. A test is made on the value of $\|\mathbf{dx}_i\|$ to ensure that $\mathbf{x}'$ or $\mathbf{x}''$ is as accurate as it deserves to be. The test of the subprogram has failed if the corresponding $\|\mathbf{dx}_i\|$ is too large. Otherwise the test has passed and $\mathbf{x}'$ or $\mathbf{x}''$ is an acceptable approximation of $\mathbf{x}$. With

$$\rho = \|\tilde{\mathbf{b}}\| / \|\hat{\mathbf{b}}\|$$

$$\kappa = \kappa_1 = \text{condition number of } A$$

$$\eta = \text{relative arithmetic precision}$$

$$\mu = \max(m_A, n)$$

$$\nu = \min(m_A, n)$$

$$\phi = 100$$

each test has passed if and only if

$$\frac{\|\mathbf{dx}_i\|}{\|\hat{\mathbf{x}}\|} \le \kappa(1 + \kappa\rho)\eta[(6\mu - 3\nu)\nu]\phi.$$

The output value of **ISTAT** is set as follows:

ISTAT = 1 means both **LSEI( )** and **WNNLS( )** failed.
= 2 means **WNNLS( )** passed but **LSEI( )** failed.
= 3 means **LSEI( )** passed but **WNNLS( )** failed.
= 4 means both **LSEI( )** and **WNNLS( )** passed.

This measure for $\|\mathbf{dx}_i\|$ is based on combining the estimate for the norm of the matrix H of the nearby problem that $\mathbf{x}'$ solves (without constraints), $(A + H)\mathbf{x}' \cong \mathbf{b}$, [7, Chap. 13], together with the perturbation bounds of [7, Chap. 9].

It may be necessary to increase the value of ϕ slightly on some machines.

A short main program, **CLSTST**, is provided with the algorithm. Also provided are 11 data cards that are read by **CLSTST** from FORTRAN unit = 5. Each pair of the first 10 cards specifies a distinct test case. The last (eleventh) card terminates the program execution.

The subprogram **CLSTP( )** prints the computed values of the least squares residual vector length and the vectors $\mathbf{dx}_i$ for both **WNNLS( )** and **LSEI( )**. Also printed in **CLSTP( )** are the computed ranks of the equality constraint and reduced least squares matrices returned by **LSEI( )**. The arrays **KLOG(I), I = 1 to 5**, and **COND(I), J = 1 to 3**, and the value of **ISTAT** returned from **CLSTP( )** are printed by **CLSTST**. Printing is done on FORTRAN unit = 6.

6. INSTALLATION GUIDELINES AND REMARKS

This section contains information for installing subprograms **LSEI( )** and **WNNLS( )**.

Included in the package are seven groups of subprograms.

(1) **LSEI, LSI, LPDP**
(2) **WNNLS, WNLSM, WNLIT**
(3) **HFTI, H12, DIFF** from [7]
(4) **SDOT, SSCAL, SASUM, SAXPY, SNRM2, SCOPY, SSWAP, ISAMAX, SROTM, SROTMG** from [8]. (For double-precision usage **DDOT, DSCAL, DASUM, DAXPY, DNRM2, DCOPY, DSWAP, IDAMAX, DROTM, DROTMG.**)
(5) **XERROR, XERRWV, XERABT, XERCLR, XERCTL, XERDMP, XERMAX, XERPRT, XERSAV, XGETF, XGETUA, XGETUN, XSETF, XSETUA, XSETUN, FDUMP, J4SAVE, S88FMT, NUMXER** from [5]. (The subprogram **NUMXER** is included for completeness but is not used in this package.)
(6) **I1MACH** based on [1]
(7) **CLSTST, CLSTP, RAN** (test package)

All of the subprograms are written in 1966 American National Standard portable FORTRAN. The only machine-sensitive subprogram is **I1MACH( )**. It provides two environmental parameters required by the error-handling subprograms **XERROR( )** and **XERRWV( )**. This will require modification of **I1MACH( )** at each host site. FORTRAN **DATA** statements defining the values of all the required constants are available for many machines in comments within the subprogram. The appropriate set of commented statements must be activated. If the values for your machine are not there, they should be provided in the order corresponding to the description near the beginning of **I1MACH( )**. Machines for which these constants are provided are Honeywell 600/6000, IBM 360/370, Xerox Sigma, CDC 6000/7000, PDP-10 (KA and KI processors), PDP-11 (16- and 32-bit arithmetic), Burroughs 5700/6700/7700, UNIVAC 1100, Data General Eclipse, Harris, VAX, and CRAY. In addition, the user must open or declare the FORTRAN unit, designated in **I1MACH(4)**, where any error messages will be written.

We strongly recommend that calls to the error-handling subprograms **XERROR( )** and **XERRWV( )** be left intact. If the size or complexity of the error-handling package presents a problem on a particular machine, we suggest that the subprograms **XERROR( )** and **XERRWV( )** be replaced by shorter, machine-sensitive versions. These replacements should, minimally, print the character string comprising the error message and the specified data values. Usage of the full error-handling package is discussed in [5].

To convert the package for double-precision usage, follow the editing instructions at the beginning of each subprogram in groups 1, 2, 3, and 7 above. Use the double-precision version of the BLAS in group 4. No conversion is required for subprograms in groups 5 and 6.

ACKNOWLEDGMENTS

We thank two anonymous referees for providing suggestions that clarified the presentation of this algorithm.

REFERENCES

1. Fox, P.A , Hull, A.D , and Schryer, N.D. The PORT mathematical subroutine library. *ACM Trans. Math. Softw. 4*, 2 (June 1978), 104–126.
2. Hanson, R J *Constrained Least Squares Curve Fitting to Discrete Data Using B-splines—A User's Guide* Available as Sandia National Laboratories Tech Rep. SAND78-1291, Sandia National Laboratories, Albuquerque, N Mex , Feb. 1979.
3. Haskell, K.H., and Hanson, R J. An algorithm for linear least squares problems with equality and nonnegativity constraints *Math Program. 21* (1981), 98–118.
4 Haskell, K.H., and Hanson, R.J Selected algorithms for the linearly constrained least squares problem—A user's guide Tech Rep. SAND78-1290, Sandia National Laboratories, Albuquerque, N.Mex., 1979.
5. Jones, R.E. SLATEC common mathematical library error handling package. Tech. Rep. SAND78-1189, Sandia National Laboratories, Albuquerque, N Mex., 1978.
6. Lawson, C L. *The Covariance Matrix for the Solution Vector of an Equality-Constrained Least-Squares Problem.* Available as Jet Propulsion Laboratories Tech Memo 33-807, Jet Propulsion Laboratories, Pasadena, Calif., Dec. 1976.
7. Lawson, C L., and Hanson, R.J. *Solving Least Squares Problems* Prentice-Hall, Englewood Cliffs, N J , 1974.
8 Lawson, C L., Hanson, R.J., Kincaid, D.R., and Krogh, F.T Basic linear algebra subprograms for Fortran usage *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 308–323.
9. Wampler, R.H Problems used in testing the efficiency and accuracy of the modified Gram-Schmidt least squares algorithm. NBS Tech. Note 1126, Aug. 1980.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service (see page 335 for order form).]

```
      SUBROUTINE LSEI(W, MDW, ME, MA, MG, N, PRGOPT, X, RNORME, RNORML,
     * MODE, WS, IP)
C
C     DIMENSION W(MDW,N+1),PRGOPT(*),X(N),
C     WS(2*(ME+N)+K+(MG+2)*(N+7)),IP(MG+2*N+2)
C     ABOVE, K=MAX(MA+MG,N).
C
C     ABSTRACT
C
C     THIS SUBPROGRAM SOLVES A LINEARLY CONSTRAINED LEAST SQUARES
C     PROBLEM WITH BOTH EQUALITY AND INEQUALITY CONSTRAINTS, AND, IF THE
C     USER REQUESTS, OBTAINS A COVARIANCE MATRIX OF THE SOLUTION
C     PARAMETERS.
C
C     SUPPOSE THERE ARE GIVEN MATRICES E, A AND G OF RESPECTIVE
C     DIMENSIONS ME BY N, MA BY N AND MG BY N, AND VECTORS F, B AND H OF
C     RESPECTIVE LENGTHS ME, MA AND MG.  THIS SUBROUTINE SOLVES THE
C     LINEARLY CONSTRAINED LEAST SQUARES PROBLEM
C
C                   EX = F, (E ME BY N) (EQUATIONS TO BE EXACTLY
C                                       SATISFIED)
C                   AX = B, (A MA BY N) (EQUATIONS TO BE
C                                       APPROXIMATELY SATISFIED,
C                                       LEAST SQUARES SENSE)
C                   GX.GE.H,(G MG BY N) (INEQUALITY CONSTRAINTS)
C
C     THE INEQUALITIES GX.GE.H MEAN THAT EVERY COMPONENT OF THE PRODUCT
C     GX MUST BE .GE. THE CORRESPONDING COMPONENT OF H.
C
C     IN CASE THE EQUALITY CONSTRAINTS CANNOT BE SATISFIED, A
C     GENERALIZED INVERSE SOLUTION RESIDUAL VECTOR LENGTH IS OBTAINED
C     FOR F-EX. THIS IS THE MINIMAL LENGTH POSSIBLE FOR F-EX.
C
C
C     ANY VALUES ME.GE.0, MA.GE.0, OR MG.GE.0 ARE PERMITTED.  THE
C     RANK OF THE MATRIX E IS ESTIMATED DURING THE COMPUTATION. WE CALL
C     THIS VALUE KRANKE. IT IS AN OUTPUT PARAMETER IN IP(1) DEFINED
C     BELOW. USING A GENERALIZED INVERSE SOLUTION OF EX=F, A REDUCED
C     LEAST SQUARES PROBLEM WITH INEQUALITY CONSTRAINTS IS OBTAINED.
C     THE TOLERANCES USED IN THESE TESTS FOR DETERMINING THE RANK
C     OF E AND THE RANK OF THE REDUCED LEAST SQUARES PROBLEM ARE
C     GIVEN IN SANDIA TECH. REPT. SAND 78-1290. THEY CAN BE
C     MODIFIED BY THE USER IF NEW VALUES ARE PROVIDED IN
C     THE OPTION LIST OF THE ARRAY PRGOPT(*).
C
C     THE EDITING REQUIRED TO CONVERT THIS SUBROUTINE FROM SINGLE TO
C     DOUBLE PRECISION INVOLVES THE FOLLOWING CHARACTER STRING CHANGES.
C     USE AN EDITING COMMAND (CHANGE) /STRING-1/(TO)STRING-2/.
C     (START EDITING AT LINE WITH C++ IN COLS. 1-3.)
C     /REAL (12 BLANKS)/DOUBLE PRECISION/,/SASUM/DASUM/,/SDOT/DDOT/,
C     /SNRM2/DNRM2/,/ SQRT/ DSQRT/,/ ABS/ DABS/,/AMAX1/DMAX1/,
C     /SCOPY/DCOPY/,/SSCAL/DSCAL/,/SAXPY/DAXPY/,/SSWAP/DSWAP/,/E0/D0/,
C     /, DUMMY/,SNGL(DUMMY)/,/SRELPR/DRELPR/
C
```

```
C     WRITTEN BY R. J. HANSON AND K. H. HASKELL.  FOR FURTHER MATH.
C     AND ALGORITHMIC DETAILS SEE SANDIA LABORATORIES TECH. REPTS.
C     SAND 77-0552, (1978), SAND 78-1290, (1979), AND
C     MATH. PROGRAMMING, VOL. 21, (1981), P.98-118.


      SUBROUTINE WNNLS(W, MDW, ME, MA, N, L, PRGOPT, X, RNORM, MODE,
     * IWORK, WORK)
C
C     DIMENSION W(MDW,N+1),PRGOPT(*),X(N),IWORK(M+N),WORK(M+5*N)
C
C     ABSTRACT
C
C     THIS SUBPROGRAM SOLVES A LINEARLY CONSTRAINED LEAST SQUARES
C     PROBLEM.  SUPPOSE THERE ARE GIVEN MATRICES E AND A OF
C     RESPECTIVE DIMENSIONS ME BY N AND MA BY N, AND VECTORS F
C     AND B OF RESPECTIVE LENGTHS ME AND MA.  THIS SUBROUTINE
C     SOLVES THE PROBLEM
C
C               EX = F, (EQUATIONS TO BE EXACTLY SATISFIED)
C
C               AX = B, (EQUATIONS TO BE APPROXIMATELY SATISFIED,
C                        IN THE LEAST SQUARES SENSE)
C
C               SUBJECT TO COMPONENTS L+1,...,N NONNEGATIVE
C
C     ANY VALUES ME.GE.0, MA.GE.0 AND 0.LE. L .LE.N ARE PERMITTED.
C
C     THE PROBLEM IS REPOSED AS PROBLEM WNNLS
C
C               (WT*E)X = (WT*F)
C               (   A)    (   B), (LEAST SQUARES)
C               SUBJECT TO COMPONENTS L+1,...,N NONNEGATIVE.
C
C     THE SUBPROGRAM CHOOSES THE HEAVY WEIGHT (OR PENALTY PARAMETER) WT.
C
C     THE PARAMETERS FOR WNNLS ARE
C
C     INPUT..
C
C     W(*,*),MDW,   THE ARRAY W(*,*) IS DOUBLE SUBSCRIPTED WITH FIRST
C     ME,MA,N,L     DIMENSIONING PARAMETER EQUAL TO MDW.  FOR THIS
C                   DISCUSSION LET US CALL M = ME + MA.  THEN MDW
C                   MUST SATISFY MDW.GE.M.  THE CONDITION MDW.LT.M
C                   IS AN ERROR.
C
C                   THE ARRAY W(*,*) CONTAINS THE MATRICES AND VECTORS
C
C                        (E  F)
C                        (A  B)
C
C                   IN ROWS AND COLUMNS 1,...,M AND 1,...,N+1
C                   RESPECTIVELY.  COLUMNS 1,...,L CORRESPOND TO
C                   UNCONSTRAINED VARIABLES X(1),...,X(L).  THE
C                   REMAINING VARIABLES ARE CONSTRAINED TO BE
C                   NONNEGATIVE.  THE CONDITION L.LT.0 .OR. L.GT.N IS
C                   AN ERROR.
C
```

```
C      PRGOPT(*)    THIS ARRAY IS THE OPTION VECTOR.                        WNN  530
C                   IF THE USER IS SATISFIED WITH THE NOMINAL               WNN  540
C                   SUBPROGRAM FEATURES SET                                 WNN  550
C                                                                           WNN  560
C                   PRGOPT(1)=1 (OR PRGOPT(1)=1.0)                          WNN  570
```