SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

USER'S GUIDE FOR LSSOL (VERSION 1.0)[†]: A FORTRAN PACKAGE FOR CONSTRAINED LINEAR LEAST-SQUARES AND CONVEX QUADRATIC PROGRAMMING

by

Philip E. Gill, Sven J. Hammarling[‡], Walter Murray,
Michael A. Saunders and Margaret H. Wright

TECHNICAL REPORT SOL 86-1

January 1986

[†]LSSOL is available from the Stanford Office of Technology Licensing, 350 Cambridge Avenue, Suite 250, Palo Alto, California 94306, USA.

[‡]Numerical Algorithms Group, Ltd., 256 Banbury Road, Oxford, United Kingdom.

Research and reproduction of this report were partially supported by the U.S. Department of Energy Contract DE-AA03-76SF00326, PA# DE-AS03-76ER72018; National Science Foundation Grants DCR-8413211 and ECS-8312142; the Office of Naval Research Contract N00014-85-K-0343 and the U.S. Army Research Office Contract DAAG29-84-K-0156.

User's Guide for LSSOL (Version 1.0)†: a Fortran Package for Constrained Linear Least-Squares and Convex Quadratic programming

Philip E. Gill, Sven J. Hammarling‡, Walter Murray,
Michael A. Saunders and Margaret H. Wright
Systems Optimization Laboratory
Department of Operations Research
Stanford University
Stanford, California 94305

January 1986

ABSTRACT

This report forms the user's guide for Version 1.0 of LSSOL, a set of Fortran 77 subroutines for linearly constrained linear least-squares and convex quadratic programming. The method of LSSOL is of the two-phase, active-set type, and is related to the method used in the package SOL/QPSOL (Gill *et al.*, 1984b). Two main features of LSSOL are its exploitation of convexity and treatment of singularity.

LSSOL may also be used for linear programming, and to find a feasible point with respect to a set of linear inequality constraints. LSSOL treats all matrices as dense, and hence is not intended for large sparse problems.

† LSSOL is available from the Stanford Office of Technology Licensing, 350 Cambridge Avenue, Suite 250, Palo Alto, California 94306, USA.

‡ Numerical Algorithms Group, Ltd., 256 Banbury Road, Oxford, United Kingdom.

The material contained in this report is based upon research supported by the U.S. Department of Energy Contract DE-AA03-76SF00326, PA No. DE-AS03-76ER72018; National Science Foundation Grants DCR-8413211 and ECS-8312142; the Office of Naval Research Contract N00014-85-K-0343; and the U.S. Army Research Office Contract DAAG29-84-K-0156.

CONTENTS

1. PURPOSE

LSSOL is a collection of Fortran 77 subroutines designed to solve a class of quadratic programming problems that are assumed to be stated in the following general form:

$$\text{LCLS} \qquad \underset{x \in \Re^n}{\text{minimize}} \quad F(x) \qquad \text{subject to} \quad \ell \le \begin{Bmatrix} x \\ Cx \end{Bmatrix} \le u,$$

where C is $m_L \times n$ (m_L may be zero) and $F(x)$ is one of the following objective functions:

FP:	*None*	(find a feasible point for the constraints)
LP:	c^Tx	
QP1:	$\frac{1}{2}x^TAx$	A symmetric and positive semi-definite,
QP2:	$c^Tx + \frac{1}{2}x^TAx$	A symmetric and positive semi-definite,
QP3:	$\frac{1}{2}x^TA^TAx$	A $m \times n$ upper-trapezoidal,
QP4:	$c^Tx + \frac{1}{2}x^TA^TAx$	A $m \times n$ upper-trapezoidal,
LS1:	$\frac{1}{2}\|b - Ax\|^2$	A $m \times n$,
LS2:	$c^Tx + \frac{1}{2}\|b - Ax\|^2$	A $m \times n$,
LS3:	$\frac{1}{2}\|b - Ax\|^2$	A $m \times n$ upper-trapezoidal,
LS4:	$c^Tx + \frac{1}{2}\|b - Ax\|^2$	A $m \times n$ upper-trapezoidal,

with c an n-vector and b an m-vector. The specific objective function to be minimized is selected using the optional parameter Problem Type (see Section 4.2). In all that follows, problems of type "LP", "QP" and "LS" will be referred to as linear programming, quadratic programming and constrained least-squares problems respectively.

The constraints involving C will be called the *general* constraints. Note that upper and lower bounds are specified for all the variables and for all the general constraints. An *equality* constraint is specified by setting $\ell_i = u_i$. If certain bounds are not present, the associated elements of ℓ or u can be set to special values that will be treated as $-\infty$ or $+\infty$. (See the description of the optional parameter Infinite Bound in Section 4.2.)

The constant second-derivative matrix of $F(x)$ is defined as H, the *Hessian matrix.* In the LP case, $H = 0$. In QP cases 1 and 2, $H = A$; and in QP cases 3 and 4, $H = A^TA$. In all LS cases, $H = A^TA$. Problems of type QP3 or QP4 with A not in trapezoidal form should be solved as type LS1 or LS2 with $b = 0$. When considering problems of type LS, we shall refer to A as the *least-squares matrix* and to b as the *vector of observations.*

The user must supply an initial estimate of the solution. If the Hessian matrix is non-singular, LSSOL will obtain the unique (global) minimum. If H is singular, the solution may still be a global minimum if all active constraints have nonzero Lagrange multipliers. Otherwise, the solution obtained will either be a *weak minimum* (i.e., with a unique optimal objective value, but an infinite set of optimal x), or else the objective function is unbounded below in the feasible region. The last case can occur only when $F(x)$ contains an explicit linear term (as in problems of type LP, QP2, QP4, LS2 and LS4).

The LSSOL package contains approximately 6000 lines of ANSI Fortran 77, of which about 50% are comments.

2. DESCRIPTION OF THE ALGORITHM

Here we briefly summarize the main features of the method of LSSOL. Where possible, explicit reference is made to the names of variables that are parameters of subroutine LSSOL or appear in the printed output.

The method of LSSOL is a two-phase (primal) quadratic programming method (see Gill *et al.*, 1984b) with features to exploit the convexity of the objective function. (In the full-rank case, the method is related to that of Stoer, 1971.) The two phases of the method are: finding an initial feasible point by minimizing the sum of infeasibilities (the *feasibility phase*), and minimizing the quadratic objective function within the feasible region (the *optimality phase*). The computations in both phases are performed by the same subroutines. The two-phase nature of the algorithm is reflected by changing the function being minimized from the sum of infeasibilities to the quadratic objective function. The feasibility phase does *not* perform the standard simplex method (i.e., it does not necessarily find a vertex), except in the LP case when $m_L \le n$. Once any iterate is feasible, all subsequent iterates remain feasible.

In general, an iterative process is required to solve a quadratic program. (For simplicity, we shall always consider a typical iteration and avoid reference to the index of the iteration.) Each new iterate $\bar{x}$ is defined by

$$\bar{x} = x + \alpha p, \tag{1}$$

where the *step length* α is a non-negative scalar, and p is called the *search direction.*

At each point x, a *working set* of constraints is defined to be a linearly independent subset of the constraints that are satisfied "exactly" (to within the tolerance defined by the optional parameter "`Feasibility Tolerance`"; see Section 4.2). The working set is the current prediction of the constraints that hold with equality at a solution of LCLS. The search direction is constructed so that the constraints in the working set remain *unaltered* for any value of the step length. For a bound constraint in the working set, this property is achieved by setting the corresponding component of the search direction to zero. Thus, the associated variable is *fixed*, and specification of the working set induces a partition of x into *fixed* and *free* variables. During a given iteration, the fixed variables are effectively removed from the problem: since the relevant components of the search direction are zero, the columns of C corresponding to fixed variables may be ignored.

Let m_W denote the number of general constraints in the working set and let n_{FX} denote the number of variables fixed at one of their bounds (m_W and n_{FX} are the quantities "`Lin`" and "`Bnd`" in the printed output from LSSOL). Similarly, let n_{FR} ($n_{FR} = n - n_{FX}$) denote the number of free variables. At every iteration, *the variables are re-ordered so that the last n_{FX} variables are fixed*, with all other relevant vectors and matrices ordered accordingly. The order of the variables is indicated by the list of indices KX, a parameter of LSSOL.

Let C_{FR} denote the $m_W \times n_{FR}$ submatrix of general constraints in the working set corresponding to the free variables, and let p_{FR} denote the search direction with respect to the free variables only. The general constraints in the working set will be unaltered by any move along p if

$$C_{FR}p_{FR} = 0. \tag{2}$$

In order to compute p_{FR}, the *TQ factorization* of C_{FR} is used:

$$C_{FR}Q_{FR} = (\,0 \quad T\,), \tag{3}$$

where T is a nonsingular $m_W \times m_W$ reverse-triangular matrix (i.e., $t_{ij} = 0$ if $i + j < m_W$), and the non-singular $n_{FR} \times n_{FR}$ matrix Q_{FR} is the product of orthogonal transformations (see Gill *et al.*, 1984a). If the columns of Q_{FR} are partitioned so that

$$Q_{FR} = (\,Z \quad Y\,), \tag{4}$$

where Y is $n_{FR} \times m_W$, then the n_Z ($n_Z = n_{FR} - m_W$) columns of Z form a basis for the null space of C_{FR}. Thus, p_{FR} will satisfy (2) only if

$$p_{FR} = Zp_Z \tag{5}$$

for some vector p_Z.

Let Q denote the $n \times n$ matrix

$$Q = \begin{pmatrix} Q_{FR} & \\ & I_{FX} \end{pmatrix}, \tag{6}$$

where I_{FX} is the identity matrix of order n_{FX}. Let R denote an $n \times n$ upper-triangular matrix (the *Cholesky factor*) such that

$$Q^T HQ = R^T R, \tag{7}$$

and let the matrix of first n_Z rows and columns of R be denoted by R_Z. (Recall that H in (7) will in general have been re-ordered.)

The definition of p_Z in (5) depends on whether or not the matrix R_Z is singular at x. In the non-singular case, p_Z satisfies the equations

$$R_Z^T R_Z p_Z = -g_Z, \tag{8}$$

where g_Z denotes the vector $Z^T g_{FR}$ and g denotes the objective gradient. (The norms of g_{FR} is the printed quantity Norm Gf.) When p_Z is defined by (8), $x + p$ is the minimizer of the objective function subject to the constraints (bounds and general) in the working set treated as equalities. In general, a vector f_Z is available such that $R_Z^T f_Z = -g_Z$, which allows p_Z to be computed from a single back-substitution $R_Z p_Z = f_Z$. For example, when solving problem LS1, f_Z comprises the first n_Z elements of the *transformed residual vector*

$$f = P(b - Ax), \tag{9}$$

which is recurred from one iteration to the next, where P is an orthogonal matrix.

In the singular case, p_Z is defined such that

$$R_Z p_Z = 0 \quad \text{and} \quad g_Z^T p_Z < 0. \tag{10}$$

This vector has the property that the objective function is linear along p and may be reduced by any step of the form $x + \alpha p$, $\alpha > 0$.

The vector $Z^T g_{FR}$ is known as the *projected gradient* at x. If the projected gradient is zero, x is a constrained stationary point in the subspace defined by Z. During the feasibility phase, the projected gradient will usually be zero only at a vertex (although it may be zero at non-vertices in the presence of constraint dependencies). During the optimality phase, a zero projected gradient implies that x minimizes the quadratic objective when the constraints in the working set are treated as equalities. At a constrained stationary point, Lagrange multipliers λ_C and λ_B for the general and bound constraints are defined from the equations

$$C_{FR}^T \lambda_C = g_{FR} \quad \text{and} \quad \lambda_B = g_{FX} - C_{FX}^T \lambda_C. \tag{11}$$

Given a positive constant δ of the order of the machine precision, the Lagrange multiplier λ_j corresponding to an inequality constraint in the working set is said to be *optimal* if $\lambda_j \le \delta$ when the associated constraint is at its *upper bound*, or if $\lambda_j \geq -\delta$ when the associated constraint is at its *lower bound*. If a multiplier is non-optimal, the objective function (either the true objective or the sum of infeasibilities) can be reduced by deleting the corresponding constraint (with index Jdel; see Section 5) from the working set.

If optimal multipliers occur during the feasibility phase and the sum of infeasibilities is nonzero, there is no feasible point, and LSSOL will continue until the minimum value of the sum of infeasibilities has been found. At this point, the Lagrange multiplier λ_j corresponding to an inequality constraint in the working set will be such that $-(1+\delta) \leq \lambda_j \leq \delta$ when the associated constraint is at its *upper bound*, and $-\delta \leq \lambda_j \leq 1+\delta$ when the associated constraint is at its *lower bound*. Lagrange multipliers for equality constraints will satisfy $|\lambda_j| \leq 1+\delta$.

The choice of step length is based on remaining feasible with respect to the satisfied constraints. If R_Z is nonsingular and $x + p$ is feasible, α will be taken as unity. In this case, the projected gradient at $\bar{x}$ will be zero, and Lagrange multipliers are computed. Otherwise, α is set to α_M, the step to the "nearest" constraint (with index Jadd; see Section 5), which is added to the working set at the next iteration.

If A is not input as a triangular matrix, it is overwritten by a triangular matrix R satisfying (7) obtained using the Cholesky factorization in the QP case, or the QR factorization in the LS case. Column interchanges are used in both cases, and an estimate is made of the rank of the triangular factor. Thereafter, the dependent rows of R are eliminated from the problem.

Each change in the working set leads to a simple change to C_{FR}: if the status of a general constraint changes, a *row* of C_{FR} is altered; if a bound constraint enters or leaves the working set, a *column* of C_{FR} changes. Explicit representations are recurred of the matrices T, Q_{FR} and R; and of vectors Q^Tg, Q^Tc and f, which are related by the formulae

$$f = Pb - \begin{pmatrix} R \\ 0 \end{pmatrix} Q^T x \qquad (b \equiv 0 \quad \text{for the QP case}),$$

and

$$Q^T g = Q^T c - R^T f.$$

Note that the triangular factor R associated with the Hessian of the original problem is updated during both the optimality *and* the feasibility phases.

The treatment of the singular case depends critically on the following feature of the matrix updating schemes used in LSSOL: if a given factor R_Z is non-singular, it can become singular during subsequent iterations only when a constraint leaves the working set, in which case only its *last* diagonal element can become zero. This property implies that a vector satisfying (10) may be found using the single back-substitution $\bar{R}_Z p_Z = e_Z$, where $\bar{R}_Z$ is the matrix R_Z with a unit last diagonal, and e_Z is a vector of all zeros except in the last position. If H is singular, the matrix R (and hence R_Z) may be singular at the start of the optimality phase. However, R_Z will be non-singular if enough constraints are included in the initial working set. (The null matrix is positive definite by definition, corresponding to the case when C_{FR} contains n_{FR} constraints.) The idea is to include as many general constraints as necessary to ensure a non-singular R_Z.

At the beginning of each phase, an upper-triangular matrix R_1 is determined that is the largest non-singular leading submatrix of R_Z. The use of interchanges during the factorization of A tends to maximize the dimension of R_1. (The rank of R_1 is estimated using the optional parameter **Rank Tolerance**; see Section 4.2.) Let Z_1 denote the columns of Z corresponding to R_1, and let Z be partitioned as $Z = (\ Z_1 \quad Z_2\ )$. A working set for which Z_1 defines the null space can be obtained by including *the rows of* Z_2^T as "artificial constraints". Minimization of the objective function then proceeds within the subspace defined by Z_1.

The artificially augmented working set is given by

$$\bar{C}_{FR} = \begin{pmatrix} C_{FR} \\ Z_2^T \end{pmatrix}, \tag{12}$$

so that p_{FR} will satisfy $C_{FR}p_{FR} = 0$ *and* $Z_2^T p_{FR} = 0$. By definition of the TQ factorization, $\bar{C}_{FR}$ *automatically* satisfies the following:

$$\bar{C}_{FR}Q_{FR} = \begin{pmatrix} C_{FR} \\ Z_2^T \end{pmatrix} Q_{FR} = \begin{pmatrix} C_{FR} \\ Z_2^T \end{pmatrix} (\, Z_1 \quad Z_2 \quad Y\,) = (\, 0 \quad \bar{T}\,),$$

where

$$\bar{T} = \begin{pmatrix} 0 & T \\ I & 0 \end{pmatrix},$$

and hence the TQ factorization of (12) requires no additional work.

The matrix Z_2 need not be kept fixed, since its role is purely to define an appropriate null space; the TQ factorization can therefore be updated in the normal fashion as the iterations proceed. No work is required to "delete" the artificial constraints associated with Z_2 when $Z_1^T g_{FR} = 0$, since this simply involves repartitioning Q_{FR}. When deciding which constraint to delete, the "artificial" multiplier vector associated with the rows of Z_2^T is equal to $Z_2^T g_{FR}$, and the multipliers corresponding to the rows of the "true" working set are the multipliers that would be obtained if the temporary constraints were not present.

The number of columns of Z and Z_1, the Euclidean norm of $Z_1^T g_{FR}$, and the condition estimator of R_1 appear in the printed output as `Nz`, `Nz1`, `Norm Gz1` and `Cond Rz1` (see Section 5).

Although the algorithm of LSSOL does not perform simplex steps in general, there is one exception: a linear program with fewer general constraints than variables (i.e., $m_L \le n$). (Use of the simplex method in this situation leads to savings in storage.) At the starting point, the "natural" working set (the set of constraints exactly or nearly satisfied at the starting point) is augmented with a suitable number of "temporary" bounds, each of which has the effect of temporarily fixing a variable at its current value. In subsequent iterations, a temporary bound is treated as a standard constraint until it is deleted from the working set, in which case it is never added again.

One of the most important features of LSSOL is its control of the conditioning of the working set, whose nearness to linear dependence is estimated by the ratio of the largest to smallest diagonals of the TQ factor T (the printed value `Cond T`; see Section 5). In constructing the initial working set, constraints are excluded that would result in a large value of `Cond T`. Thereafter, LSSOL allows constraints to be violated by as much as a user-specified **`Feasibility Tolerance`** (see Section 4.2) in order to provide, whenever possible, a *choice* of constraints to be added to the working set at a given iteration. Let α_M denote the maximum step at which $x + \alpha_M p$ does not violate any constraint by more than its feasibility tolerance. All constraints at distance α ($\alpha \le \alpha_M$) along p from the current point are then viewed as acceptable candidates for inclusion in the working set. The constraint whose normal makes the largest angle with the search direction is added to the working set. In order to ensure that the new iterate satisfies the constraints in the working set as accurately as possible, the step taken is the exact distance to the newly added constraint. As a consequence, negative steps are occasionally permitted, since the current iterate may violate the constraint to be added by as much as the feasibility tolerance.

LSSOL has been designed to be efficient when used to solve a *sequence* of related problems · for example, within a sequential quadratic programming method for nonlinearly constrained optimization (e.g., the NPSOL package of Gill *et al.*, 1986). In particular, the user may specify an initial working set (the indices of the constraints believed to be satisfied exactly at the solution); see the discussion of the optional parameter `Warm Start` in Section 4.2.

3. SPECIFICATION OF SUBROUTINE LSSOL

The formal specification of LSSOL is the following:

```
    SUBROUTINE LSSOL ( M, N,
                       NCLIN, NROWC, NROWA,
                       C, BL, BU, CVEC,
                       ISTATE, KX, X, A, B,
                       INFORM, ITER, OBJ, CLAMDA,
                       IW, LENIW, W, LENW )

    INTEGER            M, N, NCLIN,
                       NROWC, NROWA, INFORM, ITER, LENIW, LENW
    INTEGER            ISTATE(N+NCLIN), KX(N), IW(LENIW)
    REAL               OBJ
    REAL               C(NROWC,*), BL(N+NCLIN), BU(N+NCLIN),
                       CVEC(*), X(N), A(NROWA,*),
                       B(*), CLAMDA(N+NCLIN), W(LENW)
```

Note: Here and elsewhere, the specification of a parameter as REAL should be interpreted as *working precision*, which may be DOUBLE in some installations.

3.1. Formal parameters

M — **(Input)** The number of rows in the array A. If the problem is specified as type FP or LP (see Section 4), M is not referenced and is assumed to be zero.

If the problem is of type QP, M will usually be N, the number of variables. However, a value of M less than N is appropriate for QP3 or QP4 if A is an upper-trapezoidal matrix with M rows. Similarly, M may be used to define the dimension of a leading block of non-zeros in the Hessian matrices of QP1 or QP2, in which case the last $N - M$ rows and columns of A are assumed to be zero. In the QP case, M should not be greater than N; if it is, the last $M - N$ rows of A are ignored.

If the problem is specified as type LS1, LS2, LS3 or LS4, M is also the dimension of the array B. Note that all possibilities ($M < N$, $M = N$ and $M > N$) are allowed.

N — **(Input)** The number of variables, i.e., the dimension of X. (N must be positive.)

NCLIN — **(Input)** The number of general linear constraints in the problem. (NCLIN may be zero.)

NROWC — **(Input)** The declared row dimension of C. (NROWC must be at least 1 and at least NCLIN.)

NROWA — **(Input)** The declared row dimension of the array A. (NROWA must be at least 1 and at least M.)

C — **(Input)** A real array of declared dimension (NROWC,*), where the second dimension must be at least N. The i-th row of C contains the coefficients of the i-th general constraint, $i = 1$ to NCLIN. If NCLIN is zero, C is not accessed; the actual parameter may then be any convenient array or an array with dimension (1,1).

BL — **(Input)** A real array of dimension at least N + NCLIN that contains the lower bounds for all the constraints, in the following order (which is also observed for BU, ISTATE, and CLAMDA): the first N elements of BL contain the lower bounds on the variables; if NCLIN > 0, the next NCLIN elements of BL contain the lower bounds for the general linear constraints. In order for the problem specification to be meaningful, it is required that BL(j) $\leq$ BU(j) for all j. To specify a non-existent lower bound (i.e., $l_j = -\infty$), the value used must satisfy BL(j) $\leq -$BIGBND, where BIGBND is the value of the optional parameter Infinite Bound, whose default value is 10^{10} (see Section 4.2). To specify the j-th constraint as an *equality*, the user must set BL(j) = BU(j) = β, say, where $|\beta| <$ BIGBND.

BU (Input) A real array of dimension at least N + NCLIN that contains the upper bounds for all the constraints, in the same order described above under BL. To specify a non-existent upper bound (i.e., $u_j = \infty$), the value used must satisfy BU(j) $\geq$ BIGBND.

CVEC (Input) A real array of dimension at least N containing the coefficients of the explicit linear term of the objective function. If the problem is of type FP, QP1, QP3, LS1 or LS3, CVEC is not accessed; CVEC may then be declared to be of dimension (1), or the actual parameter may be any convenient array.

ISTATE (Input) An integer array of dimension at least N + NCLIN. ISTATE need not be initialized if Cold Start (the default) is specified. For a Warm Start, ISTATE specifies the desired status of the constraints at the start of the feasibility phase. The ordering of ISTATE is the same as that described above for BL, i.e., the first N components of ISTATE refer to the upper and lower bounds on the variables, and components N + 1 through N + NCLIN refer to the upper and lower bounds on Cx. Possible values for ISTATE are:

ISTATE(j)	**Meaning**
0	The corresponding constraint should *not* be in the initial working set.
1	The constraint should be in the initial working set at its lower bound.
2	The constraint should be in the initial working set at its upper bound.
3	The constraint should be in the initial working set as an equality. This value must not be specified unless BL(j) = BU(j). The values 1, 2 or 3 all have the same effect when BL(j) = BU(j).

Other values of ISTATE are also acceptable. In particular, if LSSOL has been called previously with the same values of N and NCLIN, ISTATE already contains satisfactory information.

(Output) If LSSOL exits with INFORM = 0, 1 or 3, the values in the array ISTATE indicate the status of the constraints in the active set at the solution. Otherwise, ISTATE indicates the composition of the working set at the final iterate. The significance of each possible value of ISTATE(j) is as follows:

ISTATE(j)	**Meaning**
−2	The constraint violates its lower bound by more than the feasibility tolerance.
−1	The constraint violates its upper bound by more than the feasibility tolerance.
0	The constraint is satisfied to within the feasibility tolerance, but is not in the working set.

1	This inequality constraint is included in the working set at its lower bound.
2	This inequality constraint is included in the working set at its upper bound.
3	The constraint is included in the working set as an equality. This value of ISTATE can occur only when BL(j) = BU(j).

KX **(Input)** An integer array of dimension at least N. KX *must be defined on input* for problems QP3, QP4, LS3 or LS4, i.e., problems in which A is specified as an upper-trapezoidal matrix. KX must define the order of the columns of the matrix A with respect to the ordering of X. Thus, if KX(1) = 5, column 1 of A is the column associated with variable X(5). For problems of type FP, LP, QP1, QP2, LS1 or LS2, KX need not be initialized.

(Output) KX gives the order of the columns of A with respect to the ordering of X, as described above.

X **(Input)** A real array of dimension at least N. X contains the initial estimate of the solution.

(Output) X is the last iterate of LSSOL. If INFORM = 0, 1 or 3, X will be an estimate of the solution.

A **(Input)** A real array of dimension (NROWA,*), where the second dimension must be at least N. A defines the data matrix A in LCLS.

If the problem is of type FP or LP, A is not accessed and may be dimensioned (1,1).

If the problem is of type QP1 or QP2, the first M rows and columns of A must contain the leading M by M rows and columns of the symmetric Hessian matrix. Only the diagonal and upper-triangular elements of the leading M rows and columns of A are referenced. The remaining elements are assumed to be zero and need not be assigned.

For problems QP3, QP4, LS3 or LS4, the first M rows of A must contain an M by N upper-trapezoidal factor of either the Hessian matrix or the least-squares matrix, *ordered according to the* KX *array* (see above). The factor need not be of full rank, i.e., some of the diagonals may be zero. However, as a general rule, the larger the dimension of the leading non-singular submatrix of A, the fewer iterations will be required. Elements outside the upper-triangular part of the first M rows of A are assumed to be zero and need not be assigned.

If a constrained least-squares problem contains a very large number of observations, storage limitations may prevent storage of the entire least-squares matrix. In such cases, the user should transform the original A into a triangular matrix *before* the call to LSSOL and solve the problem as type LS3 or LS4.

(Output) If the problem is of type LS or QP, A contains the upper-triangular matrix R of (7), with columns ordered as indicated by KX (see above). This matrix may be used to obtain the variance-covariance matrix or to recover the upper-triangular factor of the original least-squares matrix.

B **(Input)** A real array of dimension at least M. If the problem is of type FP, LP or QP, B is not accessed and may be dimensioned (1). If the problem is of type LS, B must contain the vector of observations b in problem LCLS.

(Output) On exit from a problem of type LS, B contains the transformed residual vector (9).

INFORM **(Output)** An integer that indicates the result of LSSOL. (If `Print Level` > 0, a short description of INFORM is printed.) The possible values of INFORM are:

INFORM	**Meaning**
0	X is a strong local minimum. (The projected gradient is negligible, the Lagrange multipliers are optimal, and R_z is non-singular.)
1	X is a weak local minimum. (The projected gradient is negligible, the Lagrange multipliers are optimal, but R_z is singular or there is a small multiplier.) This means that the final X is not unique.
2	The solution appears to be unbounded. This value of INFORM implies that a step as large as `Infinite Bound` would have to be taken in order to continue the algorithm. This situation can occur only when A is singular, there is an explicit linear term, and at least one variable has no upper or lower bound.
3	No feasible point was found, i.e., it was not possible to satisfy all the constraints to within the feasibility tolerance. In this case, the constraint violations at the final X will reveal a value of the tolerance for which a feasible point will exist for example, if the feasibility tolerance for each violated constraint exceeds its `Residual` at the final point. The modified problem (with an altered feasibility tolerance) may then be solved using a `Warm Start` (see Section 4).
4	The limiting number of iterations (determined by the parameters `Feasibility Phase Iterations` and `Optimality Phase Iterations`) was reached before normal termination occurred.
5	The algorithm could be cycling, since a total of 50 changes were made to the working set without altering X.
6	An input parameter is invalid.

ITER **(Output)** An integer that gives the total number of iterations performed in the feasibility phase and the optimality phase.

OBJ **(Output)** The value of the objective function at X if X is feasible, or the sum of infeasibilities at X otherwise. If the problem is of type FP and X is feasible, OBJ is zero.

CLAMDA **(Output)** A real array of dimension at least N + NCLIN that contains the Lagrange multiplier for every constraint with respect to the current working set. The ordering of CLAMDA follows the convention given above under BL, i.e., the first N components contain the multipliers for the bound constraints on the variables, and the remaining components contain the multipliers for the general linear constraints. If ISTATE(j) $= 0$ (i.e., constraint j is not in the working set), CLAMDA(j) is zero. If X is optimal, CLAMDA(j) should be non-negative if ISTATE(j) $= 1$ and non-positive if ISTATE(j) $= 2$.

3.2. Workspace parameters

IW **(Input)** An integer array of dimension LENIW that provides integer workspace for LSSOL.

`LENIW`	**(Input)** The dimension of `IW`. `LENIW` must be at least `N`.
`W`	**(Input)** A real array of dimension `LENW` that provides real workspace for LSSOL.
`LENW`	**(Input)** The dimension of `W`. If the problem is of type `FP` and $\mathtt{N} \le \mathtt{NCLIN}$, `LENW` must be at least $2\,\mathtt{N}^2 + 6\,\mathtt{N} + 6\,\mathtt{NCLIN}$. If the problem is of type `FP` and $0 < \mathtt{NCLIN} < \mathtt{N}$, `LENW` must be at least $2\,(\mathtt{NCLIN} + 1)^2 + 6\,\mathtt{N} + 6\,\mathtt{NCLIN}$. If $\mathtt{NCLIN} = 0$, `LENW` must be at least $6\,\mathtt{N}$. If the problem is of type `LP` and $\mathtt{N} \le \mathtt{NCLIN}$, `LENW` must be at least $2\,\mathtt{N}^2 + 7\,\mathtt{N} + 6\,\mathtt{NCLIN}$. If the problem is of type `LP` and $\mathtt{N} > \mathtt{NCLIN} > 0$, `LENW` must be at least $2\,(\mathtt{NCLIN} + 1)^2 + 7\,\mathtt{N} + 6\,\mathtt{NCLIN}$. If the problem is of type `LP` and $\mathtt{NCLIN} = 0$, `LENW` must be at least $7\,\mathtt{N}$. For problems `QP1`, `QP3`, `LS1` and `LS3`, `LENW` must be at least $2\,\mathtt{N}^2 + 9\,\mathtt{N} + 6\,\mathtt{NCLIN}$ if $\mathtt{NCLIN} > 0$, and at least $9\,\mathtt{N}$ if $\mathtt{NCLIN} = 0$. For problems `QP2`, `QP4`, `LS2` and `LS4`, `LENW` must be at least $2\,\mathtt{N}^2 + 10\,\mathtt{N} + 6\,\mathtt{NCLIN}$ if $\mathtt{NCLIN} > 0$, and at least $10\,\mathtt{N}$ if $\mathtt{NCLIN} = 0$.

If `Print Level` > 0, the amounts of workspace provided and required are printed. As an alternative to computing `LENIW` and `LENW` from the formulas given above, the user may prefer to obtain appropriate values from the output of a preliminary run with a positive value of `Print Level` and `LENIW` and `LENW` set to 1. (LSSOL will then terminate with $\mathtt{INFORM} = 6$.)

4. OPTIONAL INPUT PARAMETERS

Several optional parameters in LSSOL define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of LSSOL, these optional parameters have associated *default values* (see Section 4.2) that are appropriate for most problems. Therefore, the user need specify only those parameters whose values are to be different from their default values. The remainder of this section can be skipped by users who wish to use the default values for *all* optional parameters.

Each optional parameter is defined by a single character string of up to 72 characters, containing one or more *items*. The items associated with a given option must be separated by spaces or equal signs (=). Alphabetic characters may be upper or lower case. An example of an optional parameter is the string

```
Print level = 5
```

For each option, the string contains the following items.

1. The *keyword* (required for all options).
2. A *phrase* (one or two words) that qualifies the keyword (only for some options).
3. A *number* that specifies either an INTEGER or a REAL value (only for some options). Such numbers may be up to 16 contiguous characters in Fortran 77's I, F, E or D formats, terminated by a space.

Blank strings and comments are ignored and may be used to improve readability. A *comment* begins with an asterisk (*) and all subsequent characters are ignored. If the string is not a comment and is not recognized, a warning message is printed on the specified output device (see Section 7.5). Synonyms are recognized for some of the keywords, and abbreviations may be used.

The following are examples of valid option strings for LSSOL:

```
NOLIST
warm start
COLD START
Problem type = Least Squares
Problem type = LP
Problem Type QP4
Feasibility tolerance 1.0E-8 * for IBM in double precision
CRASH TOLERANCE = .002
* This string will be completely ignored.
Feasibility phase iteration limit 100
Optimality phase iteration limit = 10 *
```

4.1. Specification of the optional parameters

Optional parameters may be specified in two ways, as follows.

• Using subroutine LSFILE and an external file

The subroutine LSFILE provided with the LSSOL package will read options from an external *options file*, and should be called *before* a call to LSSOL. Each line of the options file defines a single optional parameter. The file must begin with Begin and end with End. (An options file consisting only of these two lines corresponds to supplying no options.)

The specification of LSFILE is

```
SUBROUTINE LSFILE( IOPTNS, INFORM )
INTEGER            IOPTNS, INFORM
```

IOPTNS must be the unit number of the options file, in the range [0, 99], and is unchanged on exit from LSFILE. INFORM need not be set on entry. On return, INFORM will be 0 if the file is a valid options file and IOPTNS is in the correct range. INFORM will be set to 1 if IOPTNS is out of range, and will be set to 2 if the file does not begin with **Begin** or end with **End**.

An example of a valid options file is

```
Begin
   Print level = 5
   Problem type LP
End
```

If the options file is on unit number 5, it can be read by the call

```
CALL LSFILE( 5, INFORM )
```

• **Using subroutine LSOPTN**

The second method of setting the optional parameters is through a series of calls to the subroutine LSOPTN provided with the LSSOL package. The specification of LSOPTN is

```
SUBROUTINE LSOPTN( STRING )
CHARACTER*(*)      STRING
```

STRING must be a single valid option string (see above), and will be unchanged on exit. LSOPTN must be called once for every optional parameter to be set. An example of a call to LSOPTN is

```
CALL LSOPTN( 'Print level = 5' )
```

• **Use of the Nolist and Defaults option**

In general, each user-specified optional parameter is printed as it is read or defined. By using the special parameter **Nolist**, the user may suppress this printing for a given call of LSSOL. To take effect, **Nolist** must be the first parameter specified in the options file; for example,

```
Begin
   Nolist
   Problem type LP
End
```

Alternatively, the first call to LSOPTN, before or after a call to LSSOL, must be

```
CALL LSOPTN( 'Nolist' ).
```

All parameters not specified by the user are automatically set to their default values. Any optional parameters that are set by the user are not altered by LSSOL, and hence changes to the options are *cumulative*. For example, calling LSOPTN('Print level = 5') sets the print level to 5 for all subsequent calls to LSSOL until it is reset by the user. The only exception to this rule is permitted by the special optional parameter `Defaults`, whose effect is to reset all optional parameters to their default values. For example, in the following situation

```
          CALL LSSOL ( ...  )
C
          CALL LSOPTN( 'Print level 5' )
          CALL LSOPTN( 'Iteration limit = 100' )
          CALL LSSOL ( ...  )
C
          CALL LSOPTN( 'Defaults' )
          CALL LSSOL ( ...  )
```

the first and last runs of LSSOL will occur with the default parameter settings, but in the second run, the print level and iteration limit are altered.

4.2. Description of the optional parameters

The following list (in alphabetical order) gives the valid options. For each option, we give the keyword, any essential optional qualifiers, the default value, and the definition. The minimum abbreviation of each keyword is underlined. If no characters of an optional qualifier are underlined, the qualifer may be omitted. The letter a denotes a phrase (character string) that qualifies an option. The letters i and r denote INTEGER and REAL values required with certain options. The number ϵ is a generic notation for machine precision.

`Cold Start` Default = `Cold Start`
`Warm Start`

This option specifies how the initial working set is chosen. With a cold start, LSSOL chooses the initial working set based on the values of the variables and constraints at the initial point. Broadly speaking, the initial working set will include equality constraints and bounds or inequality constraints that violate or "nearly" satisfy their bounds (to within `Crash Tolerance`; see below).

With a warm start, the user must provide a valid definition of every element of the array ISTATE (see Section 3 for the definition of this array). LSSOL will override the user's specification of ISTATE if necessary, so that a poor choice of the working set will not cause a fatal error. A warm start will be advantageous if a good estimate of the initial working set is available—for example, when LSSOL is called repeatedly to solve related problems.

`Crash Tolerance` r Default = .01

This value is used in conjunction with the optional parameter `Cold Start` (the default value) when LSSOL selects an initial working set. If $0 \le r \le 1$, the initial working set will include bounds or general inequality constraints that lie within r of their bounds. In particular, a constraint of the form $c_j^T x \ge l$ will be included in the initial working set if $|c_j^T x - l| \le r(1 + |l|)$. If $r < 0$ or $r > 1$, the default value is used.

`Feasibility Phase Iteration Limit` i_1 Default = $\max(50, 5(n + m_L))$
`Optimality Phase Iteration Limit` i_2 Default = $\max(50, 5(n + m_L))$

The scalars i_1 and i_2 specify the maximum number of iterations allowed in the feasibility and optimality phases. `Optimality Phase Iteration Limit` is equivalent to `Iteration Limit`. Setting $i_1 = 0$ and `Print Level` > 0 means that the workspace needed will be computed and printed, but no iterations will be performed.

`Feasibility Tolerance` r Default $= \sqrt{\epsilon}$

If $r > 0$, r defines the maximum acceptable *absolute* violation in each constraint at a "feasible" point: i.e., a constraint is considered satisfied if its violation does not exceed r. For example, if the variables and the coefficients in the general constraints are of order unity, and the latter are correct to about 6 decimal digits, it would be appropriate to specify r as 10^{-6}. If $r \leq 0$, the default value is used.

`Infinite Bound Size` r Default $= 10^{10}$

If $r > 0$, r defines the "infinite" bound BIGBND in the definition of the problem constraints. Any upper bound greater than or equal to BIGBND will be regarded as plus infinity (and similarly for a lower bound less than or equal to $-$BIGBND). If $r \leq 0$, the default value is used.

`Infinite Step Size` r Default $= \max(\text{BIGBND}, 10^{10})$

If $r > 0$, r specifies the magnitude of the change in variables that will be considered a step to an unbounded solution. (Note that an unbounded solution can occur only when the Hessian is singular and the objective contains an explicit linear term.) If the change in x during an iteration would exceed the value of `Infinite Step`, the objective function is considered to be unbounded below in the feasible region. If $r \leq 0$, the default value is used.

`Iteration Limit` i Default $= \max(50, 5(n + m_L))$
`Iters`
`Itns`

See `Feasibility Phase Iteration Limit` above.

`Optimality Phase Iteration Limit` i Default $= \max(50, 5(n + m_L))$

See `Feasibility Phase Iteration Limit` above.

`Print Level` i Default $= 10$

The value of i controls the amount of printout produced by LSSOL, as indicated below.

i	Output
0	No output.
1	*The final solution only.*
5	One line of output for each iteration (no printout of the final solution).
≥ 10	The final solution and one line of output for each iteration.
≥ 20	At each iteration, the Lagrange multipliers, the variables x, the constraint values Cx and the constraint status.
≥ 30	At each iteration, the diagonal elements of the matrix T associated with the TQ factorization (3) of the working set, and the diagonal elements of the triangular matrix R.

`Problem Type` a Default = `LS1`

This option specifies the type of objective function to be minimized during the optimality phase. The following are the ten optional keywords and the dimensions of the arrays that must be specified to define the objective function:

FP	A, B and CVEC not accessed;
LP	A and B not accessed, CVEC(N);
QP1	A(NROWA,N) symmetric, B and CVEC not referenced;
QP2	A(NROWA,N) symmetric, B not referenced, CVEC(N);
QP3	A(NROWA,N) upper-trapezoidal, KX(N), B and CVEC not referenced;
QP4	A(NROWA,N) upper-trapezoidal, KX(N), B not referenced, CVEC(N);
LS1	A(NROWA,N), B(M), CVEC not referenced;
LS2	A(NROWA,N), B(M), CVEC(N);
LS3	A(NROWA,N) upper-trapezoidal, KX(N), B(M), CVEC not referenced;
LS4	A(NROWA,N) upper-trapezoidal, KX(N), B(M), CVEC(N).

The options Least Squares and LSQ are equivalent to the default option LS1. The options Linear program and Quadratic program are equivalent to LP and QP2 respectively. If $A = 0$, i.e., the objective function is purely linear, the efficiency of LSSOL may be increased by specifying a as LP (or Linear Program).

Rank Tolerance $\quad r \quad$ Default $= \sqrt{\epsilon}$

If $0 \le r < 1$, r enables the user to control the estimation of the rank of A and the triangular factor R_1 (see Section 2). If ρ_i denotes the function $\rho_i = \max\{|R_{11}|, |R_{22}|, \ldots, |R_{ii}|\}$, the rank of R is defined to be smallest index i such that $|R_{i+1,i+1}| \le r|\rho_{i+1}|$. If $r < 0$ or $r \ge 1$, the default value is used.

4.3. Optional parameter checklist and default values

For easy reference, the following sample LSOPTN list shows all valid *keywords* and their *default values*. The default options Feasibility Tolerance and Rank Tolerance depend upon ϵ, the relative precision of the machine being used. The values given here correspond to double precision arithmetic on IBM 360 and 370 systems and their successors ($\epsilon \approx 2.22 \times 10^{-16}$). Similar values would apply to any machine having about 16 decimal digits of precision.

* List of optional parameters.		
Cold Start		*
Crash Tolerance	.01	*
Feasibility Tolerance	1.1E-8	* $\sqrt{\epsilon}$
Infinite Bound	1.0E+10	* Plus infinity
Infinite Step	1.0E+10	*
Feasibility Phase Iteration Limit	50	* or $5(n + m_L)$
Optimality Phase Iteration Limit	50	* or $5(n + m_L)$
Print Level	10	*
Problem Type	Least squares	* or LS1
Rank Tolerance	1.1E-8	* $\sqrt{\epsilon}$

5. DESCRIPTION OF THE PRINTED OUTPUT

This section describes the intermediate printout produced by LSSOL. To aid interpretation of the printed results, we repeat the convention for numbering the constraints: indices 1 through N refer to the bounds on the variables, and indices N + 1 through N + NCLIN refer to the general constraints. When the status of a constraint changes, the index of the constraint is printed, along with the designation "L" (lower bound), "U" (upper bound), "E" (equality), "T" (temporary bound) or "Z" (artificial constraint).

When Print Level $\geq$ 5, the following line of output is produced at every iteration. In all cases, the values of the quantities printed are those in effect *on completion* of the given iteration.

Itn	is the iteration count.
Jdel	is the index of the constraint deleted from the working set. If **Jdel** is zero, no constraint was deleted.
Jadd	is the index of the constraint added to the working set. If **Jadd** is zero, no constraint was added.
Step	is the step taken along the computed search direction. If a constraint is added during the current iteration (i.e., **Jadd** is positive), **Step** will be the step to the nearest constraint. During the optimality phase, the step can be greater than one only if the factor R_z is singular.
Ninf	is the number of violated constraints (infeasibilities). This number will be zero during the optimality phase.
Sinf/Objective	is the value of the current objective function. If X is not feasible, **Sinf** gives a weighted sum of the magnitudes of constraint violations. If X is feasible, **Objective** is the value of the objective function of LCLS. The output line for the final iteration of the feasibility phase (i.e., the first iteration for which **NINF** is zero) will give the value of the true objective at the first feasible point. During the optimality phase, the value of the objective function will be non-increasing. During the feasibility phase, the number of constraint infeasibilities will not increase until either a feasible point is found, or the optimality of the multipliers implies that no feasible point exists. Once optimal multipliers are obtained, the number of infeasibilities can increase, but the sum of infeasibilities will either remain constant or be reduced until the minimum sum of infeasibilities is found.
Bnd	is the number of simple bound constraints in the current working set.
Lin	is the number of general linear constraints in the current working set.
Nz	is the number of columns of Z (see Section 2). The value of **Nz** is the number of variables minus the number of constraints in the working set; i.e., **Nz** = N − (**Bnd** + **Lin**). A zero value of **Nz** implies that x lies at a vertex of the feasible region.
Nz1	is the number of columns of Z_1 (see Section 2). **Nz1** is the dimension of the subspace in which the objective function is currently being minimized. If **Nz1** is less than **Nz**, the current R_z is singular.
Norm Gf	is the Euclidean norm of the gradient of the objective function with respect to the free variables, i.e., variables not currently held at a bound.
Norm Gz1	is $\|Z_1^T g_{FR}\|$, the Euclidean norm of the projected gradient with respect to Z_1. During the optimality phase, this norm will be approximately zero after a unit step.

`Cond T`	is a lower bound on the condition number of the working set.
`Cond Rz1`	is a lower bound on the condition number of the triangular factor R_1 (the first `Nz1` rows and columns of the factor R_z. If the problem is specified to be of type LP, or the estimated rank of the data matrix A is zero, `Cond Rz1` is not printed.

When `Print Level` = 1 or `Print Level` $\geq$ 10, the summary printout at the end of execution of LSSOL includes a listing of the status of every variable and constraint. Note that default names are assigned to all variables and constraints.

The following describes the printout for each variable.

`Variable`	gives the name (VARBL) and index j (j = 1 to N) of the variable.
`State`	gives the state of the variable (FR if neither bound is in the working set, EQ if a fixed variable, LL if on its lower bound, UL if on its upper bound). If `Value` lies outside the upper or lower bounds by more than the feasibility tolerance, `State` will be "++" or "--" respectively.
`Value`	is the value of the variable at the final iteration.
`Lower bound`	is the lower bound specified for the variable. ("`None`" indicates that BL(j) $\leq$ −BIGBND.)
`Upper bound`	is the upper bound specified for the variable. ("`None`" indicates that BU(j) $\geq$ BIGBND.)
`Lagr multiplier`	is the value of the Lagrange multiplier for the associated bound constraint. This will be zero if `State` is FR. If X is optimal, the multiplier should be non-negative if `State` is LL, and non-positive if `State` is UL.
`Residual`	is the difference between the variable "`Value`" and the nearer of its bounds BL(j) and BU(j).

The meaning of the printout for general constraints is the same as that given above for variables, with "variable" replaced by "constraint", with the following change in the heading:

`Linear constr`	is the name (LNCON) and index i (i = 1 to NCLIN) of the constraint.

6. ERROR RECOVERY

Termination	Recommended Action
Underflow	A single underflow will always occur if machine constants are computed automatically (as in the distributed version of LSSOL; see Section 7). Other floating-point underflows may occur occasionally, but can usually be ignored.
Overflow	If the printed output before the overflow error contains a warning about serious ill-conditioning in the working set when adding the j-th constraint, it may be possible to avoid the difficulty by increasing the magnitude of the optional parameter `Feasibility Tolerance` and rerunning the program. If the message recurs even after this change, the offending linearly dependent constraint (with index "j") must be removed from the problem. If a warning message did not precede the fatal overflow, contact the authors at Stanford University.
`INFORM` = 3	LSSOL has terminated without finding a feasible point, which means that no feasible point exists for the given feasibility tolerance. The user should check that there are no constraint redundancies. If the data for the constraints are accurate only to the absolute precision σ, the user should ensure that the value of the optional parameter `Feasibility Tolerance` is *greater* than σ. For example, if all elements of `C` are of order unity and are accurate only to three decimal places, the optional parameter `Feasibility Tolerance` should be at least 10^{-3}.
`INFORM` = 4	The value of the optional parameter `Iteration Limit` may be too small. If the method appears to be making progress (e.g., the objective function is being satisfactorily reduced), increase the iterations limit and rerun LSSOL (possibly using the warm start facility to specify the initial working set). If the iteration limit is already large, but some of the constraints could be nearly linearly dependent, check the output for a repeated pattern of constraints entering and leaving the working set. (Near-dependencies are often indicated by wide variations in size in the diagonal elements of the T matrix, which will be printed if `Print Level` ≥ 30.) In this case, the algorithm could be cycling (see the comments for `INFORM` = 5).
`INFORM` = 5	This value will occur if 50 iterations are performed without changing `X`. The user should check the printed output for a repeated pattern of constraint deletions and additions. If a sequence of constraint changes is being repeated, the iterates are probably cycling. (LSSOL does not contain a method that is *guaranteed* to avoid cycling: such a method would be combinatorial in nature.) Cycling may occur in two circumstances: at a constrained stationary point where there are some small or zero Lagrange multipliers; or at a point (usually a vertex) where the constraints that are satisfied exactly are nearly linearly dependent. In the latter case, the user has the option of identifying the offending dependent constraints and removing them from the problem, or restarting the run with a larger value of the optional parameter `Feasibility Tolerance`. If LSSOL terminates with `INFORM` = 5, but no suspicious pattern of constraint changes can be observed, it may be worthwhile to restart with the final `X` (with or without the warm start option).

7. IMPLEMENTATION INFORMATION

7.1. Format of the distribution tape

The source code and example program for LSSOL are distributed on a magnetic tape containing 7 files. The tape characteristics are described in a document accompanying the tape; normally they are 9 track, 1600 bpi, unlabeled, ASCII, 80-character records (card images), 4800-character blocks.

The following is a list of the files and a summary of their contents. For reference purposes we give a name to each file. However, the names will not be recorded on unlabeled tapes. The MACH and LSCODE files are composed of several smaller source files described in Section 7.3.

File	*Name*	*Type*	*Cards*†	*Description*
1.	DPMACH	FORTRAN	450	Double-precision source file 1: MCSUBS
2.	DPLSCODE	FORTRAN	8250	Double precision source files 2-5: BLAS, . . . , OPSUBS
3.	DPLSMAIN	FORTRAN	260	Double-precision source file LSMAIN
4.	LSMAIN	DATA	6	Options file for LSMAIN
5.	SPMACH	FORTRAN	450	Single-precision source file 1
6.	SPLSCODE	FORTRAN	8250	Single-precision source files 2–5
7.	SPLSMAIN	FORTRAN	260	Single-precision version of file 3

† Approximate figure.

One MACH and one LSCODE file should be selected for any given installation. DPMACH and DPLSCODE are intended for machines that generally require double precision computation. Examples include IBM Systems 360, 370, 3033, 3081, etc.; Amdahl 470, Facom, Fujitsu, Hitachi, and other systems analogous to IBM; DEC VAX systems; Data General MV/8000; ICL 2900 series; recent PRIME systems; DEC Systems 10 and 20; Honeywell systems; and the Univac 1100 series.

SPMACH and SPLSCODE are intended for machines for which single precision is suitably accurate for numerical computation. Examples include the Burroughs 6700 and 7700 series; the CDC 6000 and 7000 series and their Cyber counterparts; and the Cray-1.

7.2. Installation procedure

1. Obtain the appropriate MACH and LSCODE files from the tape.
2. If necessary, edit the subroutine MCHPAR according to Section 7.5.
3. Decide whether or not to split the LSCODE file into files BLAS through OPSUBS as suggested in Section 7.3.
4. Compile all the routines that were originally in the LSCODE files together with those from MACH. Run them in conjunction with the main program LSMAIN from either File 3 or File 7 and the options given in file LSMAIN DATA. Check the output against that shown in Section 8.

7.3. Source files

LSSOL has been written in ANSI (1977) Fortran and tested on an IBM 3081K computer using the IBM Fortran 77 compiler VS Fortran. Certain unavoidable machine dependencies are confined to the routine MCHPAR.

The source code is divided into 5 logical parts. For ease of handling, these are combined into the MACH and LSCODE files on the distribution tape, but for subsequent maintenance we recommend that 5 separate files be kept. In the description below we suggest a name for each file and summarize its purpose. We then list the names of the Fortran subroutines and functions involved. The naming convention used should minimize the risk of a clash with user-written routines.

File 1. `MCSUBS` *Computes machine-dependent constants.*

```
MCHPAR  MCEPS  MCENV1  MCENV2  MCSTOR
```

File 2. `BLAS` *Basic Linear Algebra Subprograms* (a subset).

```
DASUM   DAXPY   DCOPY   DDOT    DNRM2   DSWAP   DSCAL   IDAMAX
```

These routines are functionally similar to members of the BLAS package (Lawson *et al.*, 1979). If possible they should be replaced by authentic BLAS routines. Versions may exist that have been tuned to your particular machine.

```
DGEMV   DGER1
```

These routines are functionally similar to members of the Level 2 BLAS packages (Dongarra et al., 1985).

```
DCOND   DDIV    DDSCL   DLOAD   DNORM   DSSQ    DSWAP   ICOPY
IDRANK  ILOAD
```

These are additional utility routines that could be tuned to your machine. DLOAD is used the most frequently, to load a vector with a constant value.

```
DROT3   DROT3G  DGEAPQ  DGEQR   DGEQRP  DGRFG
```

These linear algebra routines are used to compute and update various matrix factorizations in LSSOL.

File 3. `CMSUBS` *General utility routines.*

```
CMALF   CMALF1  CMCHK   CMFEAS  CMPRT   CMQMUL  CMRSOL  CMRSWP
CMR1MD  CMTSOL
```

File 4. `LSSUBS` *Least-squares routines.*

```
LSADD   LSADDS  LSBNDS  LSCHOL  LSCORE  LSCRSH  LSDEL   LSDFLT
LSFEAS  LSFILE  LSGETP  LSGSET  LSKEY   LSLOC   LSMOVE  LSMULS
LSOPTN  LSPRT   LSSETX  LSSOL
```

File 5. `OPSUBS` *Option string handling routines.*

```
OPFILE  OPLOOK  OPNUM   OPSCAN  OPTOKN  OPUPPR
```

7.4. Common blocks

Certain Fortran COMMON blocks are used in the LSSOL source code to communicate between subroutines. Their names are listed below.

```
CMDEBG  LSDEBG  LSPAR1  LSPAR2  SOL1CM  SOL3CM  SOL4CM  SOL5CM
SOL6CM  SOLMCH  SOL1LS  SOL3LS
```

7.5. Machine-dependent subroutines

The routine MCHPAR in the MACH file may require modification to suit a particular machine or a non-standard application.

At the beginning of LSSOL, MCHPAR is called to assign the machine-dependent constants and the standard input and output unit numbers. These parameters are stored in the array WMACH(15) in the labeled COMMON block SOLMCH, and are defined as follows.

WMACH(1)	is NBASE, the base of floating-point arithmetic.
WMACH(2)	is NDIGIT, the number of NBASE digits of precision.
WMACH(3)	is EPS, the floating-point precision.
WMACH(4)	is RTEPS, the square root of EPSMCH.
WMACH(5)	is RMIN, the smallest positive floating-point number.
WMACH(6)	is RTMIN, the square root of RMIN.
WMACH(7)	is RMAX, the largest positive floating-point number.
WMACH(8)	is RTMAX, the square root of RMAX.
WMACH(10)	is NIN, the file number for the input stream.
WMACH(11)	is NOUT, the file number for the output stream.

Within routine MCHPAR, the machine constants are set one of two ways, depending upon the value of the logical variable HDWIRE, which is set in-line.

If HDWIRE is .FALSE. (the value set for the distributed copy of MCHPAR), the machine constants are computed *automatically* for the machine being used. If HDWIRE is .TRUE., machine constants appropriate for the IBM 360 Series are assigned directly to the elements of WMACH.

Before selecting the method of assigning the machine constants, you should note the following. The computation of the machine constants will always generate a single arithmetic underflow, and hence some appropriate remedial action may need to be taken if your machine traps underflow.

If you wish to implement the in-line assignment of machine constants for a machine other than one from the IBM 360/370 Series, MCHPAR must be modified as follows.

1. Change the in-line assignment of HDWIRE from .FALSE. to .TRUE..
2. Set the values of WMACH appropriate for the machine and precision being used. The values of NBASE, NDIGIT, EPSMCH, RMIN and RMAX for several machines are given in the following table, for both single and double precision; RTEPS, RTMIN and RTMAX may be computed using Fortran statements. The values NIN and NOUT depend on the machine installation.

   For each precision, we give two values for EPSMCH, RMIN and RMAX. The first value is a Fortran decimal approximation of the exact quantity; use of this value in MCHPAR should cause no difficulty except in extreme circumstances. The second value is the exact mathematical representation.

Table of machine-dependent parameters

	IBM 360/370 Single	CDC 6000/7000 Single	DEC 10/20 Single	Univac 1100 Single	DEC Vax Single
NBASE	16	2	2	2	2
NDIGIT	6	48	27	27	24
EPS	9.54E-7 16^{-5}	7.11E-15 2^{-47}	7.46E-9 2^{-27}	1.50E-8 2^{-26}	1.20E-7 2^{-23}
RMIN	1.0E-78 16^{-65}	1.0E-293 2^{-975}	1.0E-38 2^{-129}	1.0E-38 2^{-129}	1.0E-38 2^{-128}
RMAX	1.0E+75 $16^{63}(1-16^{-6})$	1.0E+322 $2^{1070}(1-2^{-48})$	1.0E+38 $2^{127}(1-2^{-27})$	1.0E+38 $2^{127}(1-2^{-27})$	1.0E+38 $2^{127}(1-2^{-24})$

	IBM 360/370 Double	CDC 6000/7000 Double	DEC 10/20 Double	Univac 1100 Double	DEC Vax Double
NBASE	16	2	2	2	2
NDIGIT	14	96	62	61	56
EPS	2.22D-16 16^{-13}	2.53D-29 2^{-95}	2.17D-19 2^{-62}	8.68D-19 2^{-60}	2.78D-17 2^{-55}
RMIN	1.0D-78 16^{-65}	1.0D-293 2^{-975}	1.0D-38 2^{-129}	1.0D-308 2^{-1025}	1.0D-38 2^{-128}
RMAX	1.0D+75 $16^{63}(1-16^{-14})$	1.0D+322 $2^{1070}(1-2^{-96})$	1.0D+38 $2^{127}(1-2^{-62})$	1.0D+307 $2^{1023}(1-2^{-61})$	1.0D+38 $2^{127}(1-2^{-56})$

8. EXAMPLE PROBLEMS

This section describes a linear least-squares problem and a quadratic program; the sample main program LSMAIN that calls LSSOL and the output are given in the Appendix.

The first problem is a constrained least-squares problem of type LS1 with nine variables and three general linear constraints. The least-squares matrix and vector of observations are given by

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 & 1 & 2 & 0 & 0 \\ 1 & 1 & 3 & 1 & 1 & 1 & -1 & -1 & -3 \\ 1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 3 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & -1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 2 & 2 & 3 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 2 & 2 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

The least-squares matrix has rank 6. Let ℓ in LCLS be partitioned into two sections: the first n components (denoted by ℓ_B), corresponding to the bound constraints; and the last m_L components (denoted by ℓ_L), corresponding to the linear constraints. The vector u is partitioned in a similar fashion. Using this notation, the upper and lower bounds on the variables are given by

$$\begin{aligned} \ell_B &= (-2,\ -2,\ -\infty,\ -2,\ -2,\ -2,\ -2,\ -2,\ -2)^T \\ u_B &= (\ 2,\ \ 2,\ \ 2,\ \ 2,\ \ 2,\ \ 2,\ \ 2,\ \ 2,\ \ 2)^T, \end{aligned}$$

and the general constraints are given by

$$\ell_L = \begin{pmatrix} 2 \\ -\infty \\ -4 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 \\ 1 & 2 & 3 & 4 & -2 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad u_L = \begin{pmatrix} \infty \\ -2 \\ -2 \end{pmatrix}.$$

The starting point x_0 is

$$x_0 = (\ .1,\ .5,\ .3333,\ .25,\ .2,\ .1667,\ .1428,\ .125,\ .1111\ )^T,$$

and $F(x_0) = 9.4746$ (to five figures). The optimal solution (to five figures) is

$$x^* = (\ 2.0000,\ 1.5719,\ -1.4454,\ -.037003,\ .546685,\ .17512,\ -1.6567,\ -.39477,\ .31002\ )^T,$$

and $F(x^*) = 1.390587$. All three general linear constraints are satisfied exactly at x^*. The Lagrange multiplier associated with the third general constraint is of the order of the machine precision, and therefore the point x^* is a weak minimum, i.e., the optimal objective function is unique, but is achieved for infinitely many values of x.

The second problem is a quadratic programming problem of type QP2 with a semi-definite Hessian matrix and linear term given by

$$A = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad c = \begin{pmatrix} -4 \\ -1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -.1 \\ -.3 \end{pmatrix}.$$

(Note that by setting `M = 5`, we need not assign the last four rows and columns of `A` to zero.)

The upper and lower bounds on the variables are given by

$$\ell_B = (-2, \ -2, \ -2, \ -2, \ -2, \ -2, \ -2, \ -2, \ -2)^T$$
$$u_B = (\ 2, \ \ 2, \ \ 2, \ \ 2, \ \ 2, \ \ 2, \ \ 2, \ \ 2, \ \ 2)^T,$$

and the general constraints are given by

$$\ell_L = \begin{pmatrix} -2 \\ -2 \\ -2 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 \\ 1 & 2 & 3 & 4 & -2 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad u_L = \begin{pmatrix} 1.5 \\ 1.5 \\ 4 \end{pmatrix}.$$

The starting point x_0 is the zero vector, at which $F(x_0) = 0$. The optimal solution (to five figures) is

$$x^* = (\ 2.0, \ -.23333, \ -.26667, \ -.3, \ -.1, \ 2.0, \ 2.0, \ -1.7777, \ -.45555\ )^T,$$

and $F(x^*) = -8.067778$. The first two linear constraints are satisfied exactly at the solution, as are the upper bounds on variables x_1, x_6 and x_7. Note that, although the Hessian matrix is positive semi-definite, the point x^* is unique.

9. REFERENCES

Dongarra, J. J., Du Croz, J. J., Hammarling, S. J. and Hanson, R. J. (1985). A proposal for an extended set of Fortran basic linear algebra subprograms. *SIGNUM Newsletter* 20, 1, 2 18.

Gill. P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1984a). Procedures for optimization problems with a mixture of bounds and general linear constraints, *ACM Transactions on Mathematical Software* **10**.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1984b). User's guide for QPSOL (Version 3.2): a Fortran package for quadratic programming, Report SOL 84-6, Department of Operations Research, Stanford University, California.

Gill. P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1986). User's guide for NPSOL (Version 4.0): a Fortran package for nonlinear programming, Report SOL 86-2, Department of Operations Research, Stanford University, California.

Gill. P. E.. Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press, London and New York.

Lawson, C. L., Hanson, R. J., Kincaid, D. R., and Krogh, F. T. (1979). Basic linear algebra subprograms for Fortran usage, *ACM Transactions on Mathematical Software* **5**, pp. 308–325.

Stoer, J. (1971). On the numerical solution of constrained least-squares problems, *SIAM J. Numer. Anal.* **8**, pp. 382 411.

APPENDIX. SAMPLE PROGRAM AND OUTPUT

```
************************************************************************
*     FILE LSMAIN FORTRAN
*
*     Sample program for  Version 1.0 January 1986.
************************************************************************

      IMPLICIT           DOUBLE PRECISION(A-H,O-Z)

*     Set the declared array dimensions.
*     NROWC  = the declared row dimension of  C.
*     NROWA  = the declared row dimension of  A.
*     MAXN   = maximum no. of variables allowed for.
*     MAXM   = maximum no. of observations allowed for.
*     MAXBND = maximum no. of variables + linear constraints.
*     LIWORK = the length of the integer work array.
*     LWORK  = the length of the double precision work array.

      PARAMETER         (NROWC  =   3, NROWA =  10,
     $                   MAXN   =   9, MAXM  =  10,
     $                   LIWORK =  60, LWORK = 900,
     $                   MAXBND = MAXN + NROWC )

      INTEGER            KX(MAXN), ISTATE(MAXBND)
      INTEGER            IWORK(LIWORK)
      DOUBLE PRECISION   C(NROWC,MAXN), B(MAXM)
      DOUBLE PRECISION   BL(MAXBND), BU(MAXBND), CLAMDA(MAXBND)
      DOUBLE PRECISION   CVEC(MAXN)
      DOUBLE PRECISION   A(NROWA,MAXN), X(MAXN)
      DOUBLE PRECISION   WORK(LWORK)

      DOUBLE PRECISION   BIGBND
      CHARACTER*10       CBGBND

      INTRINSIC          FLOAT

      PARAMETER        ( POINT1=0.1D+0, POINT3=0.3D+0, ONEPT5=1.5D+0 )
      PARAMETER        ( ZERO  =0.0D+0, ONE   =1.0D+0, TWO   =2.0D+0 )
      PARAMETER        ( THREE =3.0D+0, FOUR  =4.0D+0, FIVE  =5.0D+0 )
      PARAMETER        ( SIX   =6.0D+0                               )

      BIGBND =  1.0D+15
      CBGBND = '1.0D+15'

*     ========================================
*     Example 1.  A linear least-squares problem.
*     ========================================
*     Set the actual problem dimensions.
*     M      = the number of observations (rows of A)   (may be 0).
*     N      = the number of variables.
*     NCLIN  = the number of general linear constraints (may be 0).

      M      = 10
      N      = 9
      NCLIN  = 3
      NBND   = N + NCLIN
```

```

*     ------------------------------------------------------------------
*     Assign file numbers and problem data.
*     NOUT   = the unit number for printing.
*     IOPTNS = the unit number for reading the options file.
*     A      = the least-squares matrix.
*     B      = the vector of observations.
*     C      = the general constraint matrix.
*     BL     = the lower bounds on  x  and  C*x.
*     BU     = the upper bounds on  x  and  C*x.
*     X      = the initial estimate of the solution.
*     ------------------------------------------------------------------
      IOPTNS = 5
      NOUT   = 6

      DO 120 J = 1, N
         DO 110 I = 1, M
            A(I,J) = ONE
            B(I)   = ONE
  110    CONTINUE
  120 CONTINUE

      A(2 ,2) =    TWO
      A(10,2) =    ZERO

      A(3,3) =    THREE
      A(6,3) =    TWO
      A(9,3) =    ZERO

      A(4,4) =    FOUR
      A(5,4) =    THREE
      A(8,4) =    ZERO

      A(7,5) =    ZERO

      A(6,6) =    ZERO

      A(2 ,7) =   TWO
      A(3 ,7) = - ONE
      A(6 ,7) =   ZERO
      A(9 ,7) =   TWO
      A(10,7) =   ZERO

      A(2 ,8) =   ZERO
      A(3 ,8) = - ONE
      A(6 ,8) =   ZERO
      A(9 ,8) =   TWO
      A(10,8) =   TWO

      A(2 ,9) =   ZERO
      A(3 ,9) = - THREE
      A(6 ,9) = - ONE
      A(9 ,9) =   THREE
      A(10,9) =   TWO

      DO 140 J = 1, N
         DO 130 I = 1, NCLIN
            C(I,J) = ONE
  130    CONTINUE
  140 CONTINUE

      C(1,9) =   FOUR

      C(2,2) =   TWO
      C(2,3) =   THREE
      C(2,4) =   FOUR
      C(2,5) = - TWO

      C(3,2) = - ONE
      C(3,4) = - ONE

      DO 150 J = 1, N
         BL(J) = - TWO
         BU(J) =   TWO
  150 CONTINUE
      BL( 3) = - BIGBND

*     Set the ranges for the general constraints.

      BL(N+1) =   TWO
      BU(N+1) =   BIGBND
      BL(N+2) = - BIGBND
      BU(N+2) = - TWO
      BL(N+3) = - FOUR
      BU(N+3) = - TWO

      DO 170 J = 1, N
         X(J) = ONE / FLOAT(J)
  170 CONTINUE


*     ------------------------------------------------------------------
*     Read the options file.
*     Add a single option using a call to LSOPTN.
*     ------------------------------------------------------------------

      CALL LSFILE( IOPTNS, INFORM )
      IF (INFORM .NE. 0) THEN
         WRITE (NOUT, 3000) INFORM
         STOP
      END IF

      CALL LSOPTN( 'Infinite Bound size ='//CBGBND )

*     ------------------------------------------------------------------
*     Solve the problem.
*     ------------------------------------------------------------------

      CALL LSSOL ( M, N,
     $             NCLIN, NROWC, NROWA,
     $               C, BL, BU, CVEC,
     $               ISTATE, KX, X, A, B,
     $               INFORM, ITER, OBJ, CLAMDA,
     $               IWORK, LIWORK, WORK, LWORK )

*     Test for an error condition.

      IF (INFORM .GT. 1) GO TO 999

*     ================================================
*     Example 2.  A QP with Hessian bordered by zeros.
*     ================================================
*     Set the new problem dimensions.
*     M      = the number of rows (and columns) of A  (may be 0).
*     N      = the number of variables.
*     NCLIN  = the number of general linear constraints (may be 0).
*     CVEC   = the linear part of the objective function.

         M      = 5
         N      = 9
         NCLIN  = 3
         NBND   = N + NCLIN

         DO 220 J = 1, M
            DO 210 I = 1, J-1
               A(I,J) = ONE
  210       CONTINUE
  220    CONTINUE

         DO 230 I = 1, M
            A(I,I) = TWO
  230    CONTINUE

         DO 260 J = 1, N
            BL(J) = - TWO
            BU(J) =   TWO
  260    CONTINUE

         BL(N+1) = - TWO
         BU(N+1) =   ONEPT5
         BL(N+2) = - TWO
         BU(N+2) =   ONEPT5
         BL(N+3) = - TWO
         BU(N+3) =   FOUR

         DO 270 J = 1, N
            CVEC(J) = - ONE
  270    CONTINUE
         CVEC(1) = - FOUR
         CVEC(8) = - POINT1
         CVEC(9) = - POINT3

         DO 280 J = 1, N
            X(J) = ZERO
  280    CONTINUE
```

```


*     ------------------------------------------------------------------
*     Assign some new options.
*     ------------------------------------------------------------------

      CALL LSOPTN( 'Defaults            ' )
      CALL LSOPTN( 'Problem type  QP2' )
      CALL LSOPTN( 'Rank tolerance  =  1.0E-10' )
      CALL LSOPTN( 'Feasibility tolerance = 1.0E-10' )

*     ------------------------------------------------------------------
*     Solve the QP problem.
*     ------------------------------------------------------------------

      CALL LSSOL ( M, N,
     $             NCLIN, NROWC, NROWA,
     $             C, BL, BU, CVEC,
     $             ISTATE, KX, X, A, B,
     $             INFORM, ITER, OBJ, CLAMDA,
     $             IWORK, LIWORK, WORK, LWORK )

*     Test for an error condition.

      IF (INFORM .GT. 1) GO TO 999
      STOP

*     Error condition.

  999 WRITE (NOUT, 3010) INFORM
      STOP

 3000 FORMAT(/ '  LSFILE terminated with  INFORM =', I3)
 3010 FORMAT(/ '  LSSOL  terminated with  INFORM =', I3)

*     End of the example program for LSSOL.

      END
```

```
OPTIONS file
------------

     BEGIN  Options for LSSOL 1.0 Sample problem.

        Iterations Limit          =    25
        Problem Type              =    Least squares

     End

Calls to LSOPTN
---------------

     Infinite Bound size =1.0D+15




                              SOL/LSSOL  ---  Version 1.0  Feb 1986
                              ===================================


Parameters
----------

Problem type...........        LS1
Linear constraints.....          3        Feasibility tolerance..  1.49E-08        COLD start.............
Variables..............          9        Infinite bound size....  1.00E+15        Crash tolerance........  1.00E-02
Objective matrix rows..         10        Infinite step size.....  1.00E+15        Rank tolerance.........  1.49E-08

EPS (machine precision)  2.22E-16        Feasibility phase itns.        60        Print level............        10
                                          Optimality  phase itns.        25

Workspace provided is     IW(    60),  W(   900).
To solve problem we need  IW(     9),  W(   261).

Rank of the objective function data matrix =      6


 Itn Jdel  Jadd       Step Ninf  Sinf/Objective  Bnd  Lin     Nz    Nz1   Norm Gf  Norm Gz1   Cond T Cond Rz1
   0    0     0  0.0E+00    2    9.474603E+00    0    0      9      0  6.86E+00  0.00E+00  1.0E+00  0.0E+00
   1   1Z   10L  1.2E+00    2    5.987698E+00    0    1      8      0  6.86E+00  0.00E+00  1.0E+00  0.0E+00
   2   1Z   11U  4.1E-01    1    4.990079E+00    0    2      7      0  3.00E+00  0.00E+00  1.1E+00  0.0E+00
   3   1Z   12U  3.7E+00    0    4.959041E+01    0    3      6      6  5.60E+01  4.13E+01  2.3E+00  2.2E+01
   4    0    1U  3.0E-01    0    2.429930E+01    1    3      5      5  3.89E+01  2.85E+01  2.4E+00  4.8E+00
   5    0     0  1.0E+00    0    1.390587E-01    1    3      5      5  6.55E-01  1.59E-15  2.4E+00  4.8E+00

Exit from LS problem after    5 iterations.  INFORM =  1


Variable          State       Value        Lower bound       Upper bound  Lagr multiplier      Residual

VARBL  1            UL     2.000000       -2.000000         2.000000       -0.1191932            0.0000E+00
VARBL  2            FR     1.571959       -2.000000         2.000000        0.0000000E+00       0.4280
VARBL  3            FR    -1.445403          None           2.000000        0.0000000E+00        3.445
VARBL  4          FR  -0.3700275E-01    -2.000000        2.000000        0.0000000E+00        1.963
VARBL  5          FR   0.5466858        -2.000000        2.000000        0.0000000E+00        1.453
VARBL  6          FR   0.1751236        -2.000000        2.000000        0.0000000E+00        1.825
VARBL  7          FR  -1.656704         -2.000000        2.000000        0.0000000E+00       0.3433
VARBL  8          FR  -0.3947742        -2.000000        2.000000        0.0000000E+00        1.605
VARBL  9          FR   0.3100290        -2.000000        2.000000        0.0000000E+00        1.690

Linear constr   State       Value      Lower bound     Upper bound  Lagr multiplier     Residual

LNCON  1          LL    2.000000       2.000000          None         0.3973107E-01     -0.3553E-14
LNCON  2          UL   -2.000000         None          -2.000000     -0.1191932         -0.4219E-14
LNCON  3          UL   -2.000000      -4.000000        -2.000000      0.2086660E-15     -0.4441E-15

Exit LSSOL - Weak LS solution.

Final LS objective value =    0.1390587


Calls to LSOPTN
---------------

     Defaults
     Problem type  QP2
     Rank tolerance  =  1.0E-10
     Feasibility tolerance = 1.0E-10



                     SOL/LSSOL  ---  Version 1.0  Feb 1986
                     ==================================


Parameters
----------

Problem type...........       QP2
Linear constraints.....         3     Feasibility tolerance..  1.00E-10      COLD start.............
Variables..............         9     Infinite bound size....  1.00E+10      Crash tolerance........  1.00E-02
Objective matrix rows..         5     Infinite step size.....  1.00E+10      Rank tolerance.........  1.00E-10

EPS (machine precision)  2.22E-16     Feasibility phase itns.        60      Print level............        10
                                      Optimality  phase itns.        60

Workspace provided is     IW(    60),  W(    900).
To solve problem we need  IW(     9),  W(    270).

Rank of the objective function data matrix =      5


 Itn Jdel   Jadd       Step Ninf  Sinf/Objective  Bnd  Lin     Nz    Nz1    Norm Gf   Norm Gz1    Cond T  Cond Rz1
   0    0      0   0.0E+00    0    0.000000E+00    0    0      9      5   4.70E+00   4.47E+00   2.4E+00   1.3E+00
   1    0     1U   7.5E-01    0   -4.375000E+00    1    0      8      4   1.53E+00   5.00E-01   2.4E+00   1.3E+00
   2    0      0   1.0E+00    0   -4.400000E+00    1    0      8      4   1.45E+00   3.67E-17   2.4E+00   1.3E+00
   3   5Z    10U   3.0E-01    0   -4.700000E+00    1    1      7      4   1.45E+00   8.94E-01   1.0E+00   1.0E+00
   4    0      0   1.0E+00    0   -5.100000E+00    1    1      7      4   2.47E+00   1.29E-17   1.0E+00   1.0E+00
   5   7Z    12U   5.4E-01    0   -6.055714E+00    1    2      6      4   2.47E+00   1.73E+00   2.0E+00   1.3E+00
   6    0     6U   1.1E-02    0   -6.113326E+00    2    2      5      3   2.2[illegible]+00   1.64E+00   2.0E+00   1.7E+00
   7    0   11U  1.1E-01   0   -6.215049E+00   2   3   4   2  2.03E+00  1.18E+00  2.1E+00  1.5E+00
   8    0    0   1.0E+00   0   -6.530008E+00   2   3   4   2  1.10E+00  2.22E-16  2.1E+00  1.5E+00
   9   3Z    0   1.0E+00   0   -6.567373E+00   2   3   4   3  1.07E+00  2.23E-16  2.1E+00  2.7E+00
  10   4Z   7U   1.7E+00   0   -8.055612E+00   3   3   3   3  3.83E-01  2.82E-01  2.1E+00  3.7E+00
  11    0    0   1.0E+00   0   -8.067718E+00   3   3   3   3  4.38E-01  1.05E-16  2.1E+00  3.7E+00
  12  12U    0   1.0E+00   0   -8.067778E+00   3   2   4   4  4.31E-01  1.05E-16  1.2E+00  5.8E+00

Exit from QP problem after   12 iterations.  INFORM =  0

Variable        State      Value        Lower bound     Upper bound  Lagr multiplier     Residual

VARBL  1          UL     2.000000     -2.000000       2.000000      -0.8000000          0.0000E+00
VARBL  2          FR   -0.2333333     -2.000000       2.000000       0.0000000E+00       1.767
VARBL  3          FR   -0.2666667     -2.000000       2.000000       0.0000000E+00       1.733
VARBL  4          FR   -0.3000000     -2.000000       2.000000       0.0000000E+00       1.700
VARBL  5          FR   -0.1000000E+00 -2.000000       2.000000       0.0000000E+00       1.900
VARBL  6          UL     2.000000     -2.000000       2.000000      -0.9000000          0.0000E+00
VARBL  7          UL     2.000000     -2.000000       2.000000      -0.9000000          0.0000E+00
VARBL  8          FR    -1.777778     -2.000000       2.000000       0.0000000E+00      0.2222
VARBL  9          FR   -0.4555556     -2.000000       2.000000       0.0000000E+00       1.544

Linear constr   State      Value        Lower bound     Upper bound  Lagr multiplier     Residual

LNCON  1          UL     1.500000     -2.000000       1.500000      -0.6666667E-01     -0.3553E-14
LNCON  2          UL     1.500000     -2.000000       1.500000      -0.3333333E-01      0.2220E-15
LNCON  3          FR     3.933333     -2.000000       4.000000       0.0000000E+00      0.6667E-01

Exit LSSOL - Optimal QP solution.

Final QP objective value =   -8.067778
```

INDEX

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 86-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) User's Guide for LSSOL (Version 1.0): A Fortran Package for Constrained Linear Least-Squares & Convex Quadratic Programming		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Philip E. Gill, Sven J. Hammarling, Walter Murray, Michael A. Saunders & Margaret H. Wright		8. CONTRACT OR GRANT NUMBER(s) N00014-85-K-0343 DAAG29-84-K-0156
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-064
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217 U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE January 1986 13. NUMBER OF PAGES 38 pp. 15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) This document has been approved for public release and sale; its distribution is unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) optimization; quadratic programming; linearly constrained linear least-squares; mathematical software; linear programming		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) See next page.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT: USER'S GUIDE FOR LSSOL (VERSION 1.0): A FORTRAN PACKAGE FOR CONSTRAINED LINEAR LEAST-SQUARES AND CONVEX QUADRATIC PROGRAMMING by Philip E. Gill, Sven J. Hammarling, Walter Murray, Michael A. Saunders and Margaret H. Wright.

This report forms the user's guide for Version 1.0 of LSSOL, a set of Fortran 77 subroutines for linearly constrained linear least-squares and convex quadratic programming. The method of LSSOL is of the two-phase, active-set type, and is related to the method used in package SOL/QPSOL (Gill, et al., 1984b). Two main features of LSSOL are its exploitation of convexity and treatment of singularity.

LSSOL may also be used for linear programming, and to find a feasible point with respect to a set of linear inequality constraints. LSSOL treats all matrices as dense, and hence is not intended for large sparse problems.