I-1765

(1)

SANDIA REPORT SAND81-0297 • Unlimited Release • UC-32

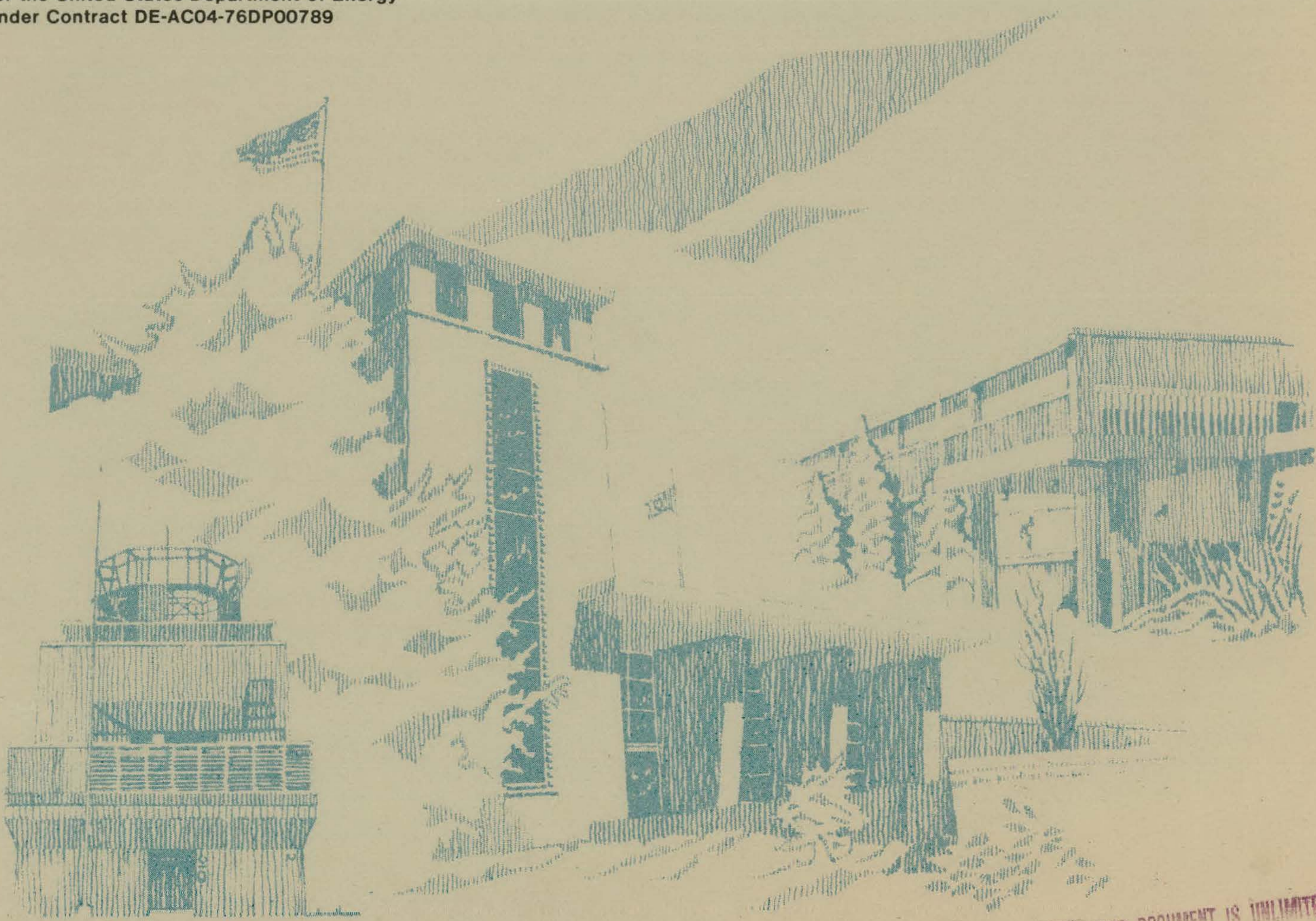
Printed December 1981

MASTER

A Sparse Linear-Programming Subprogram

Richard J. Hanson, Kathie L. Hiebert

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789



SF2900Q(8-81)

DISCLAIMER

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

UC-32
SAND81-0297
Unlimited Release
Printed December 1981

SAND--81-0297

DE82 011783

A Sparse Linear-Programming Subprogram

Richard J. Hanson and Kathie L. Hiebert
Numerical Mathematics Division 5642

Abstract

This report describes a subprogram, SPLP( ), for solving linear programming problems. The package of subprogram units comprising SPLP( ) is written in Fortran 77.

The subprogram SPLP( ) is intended for problems involving at most a few thousand constraints and variables. The subprograms are written to take advantage of sparsity in the constraint matrix.

A very general problem statement is accepted by SPLP( ). It allows upper, lower, or no bounds on the variables. Both the primal and dual solutions are returned as output parameters. The package has many optional features. Among them is the ability to save partial results and then use them to continue the computation at a later time.

Table of Contents

1. Introduction

This report discusses the subprogram SPLP( ) that computes solutions to linear programming (LP) problems of "modest size." An LP problem seeks the optimum of a linear function of unknowns. The unknowns also must satisfy certain given linear constraints. The constraints are usually stated as equations, inequalities, or bounds on the unknowns.

Typically LP problems have sparse constraint matrices, and effectively dealing with this sparseness is one of our goals. We have also concentrated on achieving easy portability across various machine lines.

How does the subprogram SPLP( ) compare with some other LP software packages? To partially answer this question we list five of the primary features of our package as seen from a user's viewpoint.

1. Up-to-date and robust methods are used in the implementation of the revised simplex method.
2. Generally, the user's problem can be accepted by the package without modification.
3. The software is portable. Virtually no change is required on any computer system that supports Fortran 77.
4. The software is in the public domain.
5. The package is easy to use, yet flexible.

We consider three alternate mathematical programming packages: XMP [17], MINOS [18], and MPS [19]. In our opinion these packages admirably satisfy 1. and 2., and provide other features that we have not mentioned. Yet each one of these packages does not satisfy at least one of these five desirable features:

- XMP does not satisfy 5.
- MINOS does not satisfy 3.
- MPS does not satisfy 3. or 4.

The subprogram SPLP( ) does satisfy all five features and should be a welcome addition to the mathematical programming software "repertoire."

Readers who are primarily concerned with using SPLP( ) should read Sections 1.1 and 3 for some background information and then turn to Section 4 for an example and Section 6 for the usage documentation. Section 2 details our implementation of the revised simplex method. Section 5 describes how to install the package.

1.1 Statement of the LP Problem

The subprogram SPLP( ) nominally solves the following linear optimization problem.

Problem LP

$$\text{minimize } c_1x_1 + \dots + c_{NVARS}x_{NVARS} \equiv (\underset{\sim}{c}^T\underset{\sim}{x})$$

subject to

$$A\underset{\sim}{x} = \underset{\sim}{w} \tag{1}$$

and bounds on the unknowns $\underset{\sim}{x}$ and $\underset{\sim}{w}$.

The matrix A has MRELAS rows and NVARS columns. The bounds for $\underset{\sim}{x}$ and $\underset{\sim}{w}$ can be any one of the four types:

a) lower and upper bounds,

b) lower bound only,

c) upper bound only, and

d) no bound at all.

Variables of type (d) will be called free variables.

The input to SPLP( ) consists of the problem dimensions MRELAS and NVARS, the vector of "costs," $\underset{\sim}{c}$, the matrix A and the bounds for $\underset{\sim}{x}$ and $\underset{\sim}{w}$. The output from SPLP( ) normally includes the primal vectors $\underset{\sim}{x}$ and $\underset{\sim}{w}$, and the dual variables, $\underset{\sim}{d}$, for the equations $A\underset{\sim}{x} = \underset{\sim}{w}$ and for the constraints expressed as bounds on the components of $\underset{\sim}{x}$. A discussion of the dual variables and their calculations is given in Section 3.

Alternate output from SPLP( ) includes indications of usage errors or that the stated problem has no solution in the usual sense. Usage errors are described in Section 6 and the situation of no solution is discussed in Section 3.

1.2 Reposing Other Forms of Problem LP

We use the novel problem statement of Eq. (1) because it incorporates most common linear programming problems. It has some attributes that are not immediately obvious. These will be discussed in the next section. The reader should note that our form of the standard problem statement is close to that of Ref. [2], p. 11-17.

Two problem statements that many LP codes use are as follows.

Problem LP-A

$$\text{minimize } c_1x_1 + \ldots + c_Nx_N \equiv (\underset{\sim}{c}^T\underset{\sim}{x}), \quad (\underset{\sim}{c} \text{ known})$$

subject to

$$\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{b} \quad , \quad (A_{M\times N},\ b_{M\times 1} \text{ known})$$

$$\underset{\sim}{x} \geq \underset{\sim}{0}$$

Problem LP-B

$$\text{minimize } c_1x_1 + \ldots + c_Nx_N \equiv (\underset{\sim}{c}^T\underset{\sim}{x}), \quad (\underset{\sim}{c} \text{ known})$$

subject to

$$\underset{\sim}{A}\underset{\sim}{x}\ \underset{\sim}{\rho}\ \underset{\sim}{b} \quad (\underset{\sim}{A}_{M\times N},\ \underset{\sim}{b}_{M\times 1} \text{ known})$$

$$\underset{\sim}{\alpha} \leq \underset{\sim}{x} \leq \underset{\sim}{\beta} \quad (\underset{\sim}{\alpha},\underset{\sim}{\beta} \text{ known})$$

(The symbol $\underset{\sim}{\rho} = (\rho_1,\ldots,\rho_M)^T$ is a set of relations: each ρ_j is either "$\leq$", "=", or "$\geq$".)

It is well-known that the Problem LP-A suffices (mathematically) for linear optimization problems; see, e.g., Ref. [1], p. 94. This form of the problem statement has several unfortunate side effects. For example, each simple bound (on the unknowns) becomes a new constraint equation and introduces a new variable into the problem. This is unnecessary; Problem LP-B avoids this objection.

Each of these two problem statements are easily reposed in terms of our statement of Eq. (1). For Problem LP-A we define NVARS = N, MRELAS = M, and $\underset{\sim}{w} = \underset{\sim}{A}\underset{\sim}{x}$. The bounds for $\underset{\sim}{x}$ and $\underset{\sim}{w}$ are $0 \leq x_j$, $j = 1,\ldots,$NVARS, and $b_i \leq w_i \leq b_i$ (i.e., $b_i = w_i$), $i = 1,\ldots,$MRELAS. For Problem LP-B we define NVARS = N, MRELAS = M, and $\underset{\sim}{w} = \underset{\sim}{A}\underset{\sim}{x}$. The bounds for $\underset{\sim}{x}$ and $\underset{\sim}{w}$ are $\alpha_j \leq x_j \leq \beta_j$, $j = 1,\ldots,$NVARS, and $w_i \leq b_i$, if ρ_i is "$\leq$"; $b_i \leq w_i$, if ρ_i is "$\geq$"; $b_i \leq w_i \leq b_i$, if ρ_i is "=".

Problem LP-B has its own unfortunate side effects. One example is an LP problem with generalized bounds on some of the constraints: $v_i < \Sigma a_{ij}x_j < u_i$. Users normally write this as two constraints: $v_i < \Sigma a_{ij}x_j$, $\Sigma a_{ij}x_j < u_i$. This is unnecessary when our problem statement is used. To put the generalized bounds into a form compatible with our Problem LP, define $w_i = \Sigma a_{ij}x_j$ with the bounds $v_i < w_i < u_i$. This provides an alternative to writing pairs of inequalities for each generalized bound.

A potential difficulty with any LP problem, no matter which statement is used, is inconsistent constraints. Suppose a user has a constraint $\Sigma a_{ij}x_j = b_i$ and a second (almost) redundant constraint $\Sigma a_{ij}x_j = b_i + \delta$. This second equation makes the resulting constraint matrix (nearly) rank deficient. This can be a serious problem and it may not be easy for the user to identify the offending constraints. The subprogram SPLP( ) can be used to eliminate this problem. First, when this condition exists, SPLP( ) will return with information to identify the offending constraint. Then, because of our specific problem statement, the redundant equation can be eliminated by reclassifying it as a free variable.

2. Details of the Implementation

Basically we are using the revised simplex algorithm, Ref. [1], p. 195, to solve Problem LP. In this section we will discuss modifications to the user's problem made by the SPLP( ) subprograms, modifications to the basic simplex algorithm and some numerical aspects of the implementation.

2.1 Normalizing and Scaling the Original Problem

Within the subprogram SPLP( ) both the independent variables, $\underset{\sim}{x}$, and the dependent variables, $\underset{\sim}{w}$, are translated or reflected so that each variable that is not free has a lower bound of zero. By extending the costs vector, $\underset{\sim}{c}$, and rewriting the constraint equations as

$$[A:-I]\begin{pmatrix}\underset{\sim}{x}\\\underset{\sim}{w}\end{pmatrix} = \underset{\sim}{0} \qquad (1')$$

we can abbreviate Eq. (1) as

$$\min c'^T x'$$
$$A'x' = 0 \qquad (2)$$

where

$$c'^T = [c^T : 0^T] \ ,$$
$$A' = [A : -I] \ ,$$
$$x' = [x^T, w^T]^T \ .$$

The components of c' and columns of A' that correspond to the dependent variables, w, are not explicitly stored in SPLP( ).

Next the unknowns x' are separated into four groups. The classification into these groups is determined by the bounds on x'. Eq. (2) can be written as

$$A'x' = A_b x_b + A_v x_v + A_u x_u + A_f x_f = 0 \ .$$

Bounds on x_b, x_v and x_u, are as follows:

$$\gamma \le x_b \le \delta$$
$$\alpha \le x_v$$
$$x_u \le \beta$$
$$x_f \text{ is free} \ .$$

These constraints are normalized by translating and reflecting the variables with the transformations

$$x_b' = x_b - \gamma$$
$$x_v' = x_v - \alpha$$
$$x_u' = \beta - x_u \ .$$

The transformed minimization problem is

$$\begin{aligned} &\text{minimize} && c_b^T x_b' + c_v^T x_v' + (-c_u^T) x_u' + c_f^T x_f \\ &\text{subject to} && A_b x_b' + A_v x_v' + (-A_u) x_u' + A_f x_f = h \ , \\ & && h = -(A_b \gamma + A_v \alpha + A_u \beta) \end{aligned}$$

$$
\begin{aligned}
\text{and bounds}\quad & 0 < x_b' < \varepsilon = \delta - \gamma \\
& 0 < x_v' \\
& 0 < x_u' \\
& \quad x_f \text{ is free} \quad . \qquad\qquad (3)
\end{aligned}
$$

Thus the problem has been normalized so that all of the variables have zero as their lower bound or they are free. The variables x_b' also have a nonnegative upper bound.

We rewrite Eq. (3), in the abbreviated form

$$
\begin{aligned}
&\text{minimize}\quad \hat{c}^T x \\
&\text{subject to constraints} \\
&\qquad \hat{A}x = h \quad , \qquad \hat{A}_{M\times N},\ h_{M\times 1}, (N > M) \quad , \\
&\text{and bounds for each } j \\
&\qquad 0 < x_j \quad , \\
&\text{or} \qquad 0 < x_j < e_j \quad , \\
&\text{or} \qquad x_j \text{ is free} \quad . \qquad\qquad (3')
\end{aligned}
$$

Scaling of the columns of $\hat{A}$ and the vector $\hat{c}$ is also performed within SPLP( ). Column scaling amounts to the change of variables $x = Dy$ where each diagonal term d_i of the diagonal matrix D is nonnegative. Normally D is computed so that each nonzero column of $\hat{A}$ has maximum magnitude equal to one. For columns of $\hat{A}$ that are zero, $d_i = 1$. The nonnegative scale factor, σ , for the vector c is chosen so that if $\hat{c}^T D$ is nonzero, $\sigma\hat{c}^T D$ has maximum magnitude equal to one. If $\hat{c}^T D$ is zero, $\sigma = 1$. Those users who think this particular choice of scaling is inappropriate can (optionally) specify other nonnegative choices for D and σ.

With this rescaling of the variables and the objective function, we have the problem

$$
\begin{aligned}
&\text{minimize}\quad \sigma(\hat{c}^T D)y \\
&\text{subject to constraints} \\
&\qquad (\hat{A}D)y = h \\
&\text{and bounds; for each } i \\
&\qquad 0 < y_i
\end{aligned}
$$

or $\qquad 0 \le d_i y_i \le e_i$

or $\qquad y_i$ is free . (4)

The condition $d_i = 0$ implies that $y_i = 0$. For $d_i > 0$ the upper bounds for the y_i are $e_i d_i^{-1}$.

This problem (Eq. 4) is the actual problem solved within SPLP( ). The user does not need to perform any of the translations, reflections or scaling of the variables. Eq. (4) is solved using the input data of Eq. (1).

Finding a solution to Eq. (4), $\underset{\sim}{y} = \hat{\underset{\sim}{y}}$, determines a solution to the normalized problem, Eq. (3'), $x' = \hat{x} = D\hat{y}$. A translation or reflection of the components of $\hat{\underset{\sim}{x}}$ is required to yield the solution vectors for $\underset{\sim}{x}$ and $\underset{\sim}{w}$ of Problem LP, Eq. (1). The dual variables $\underset{\sim}{p}'$, that satisfy the optimality conditions $D\hat{A}^T\underset{\sim}{p}' \ge \sigma D\hat{\underset{\sim}{c}}$, are rescaled to yield the dual variables for the constraint equations of Eq. (1): $\underset{\sim}{p} = \sigma^{-1}\underset{\sim}{p}'$.

2.2 Modifications to the Revised Simplex Algorithm

First we review the revised simplex algorithm, as usually presented. This form of the algorithm is based on two assumptions: 1) all the variables are nonnegative and have no upper bound and 2) the starting value for the variables is feasible (satisfies all the constraints). Given the description of this algorithm we are then able to discuss where modifications have been made.

In this section we refer to the notation of Eq. (3') but with the understanding that scaling of the data has been applied as in Eq. (4). Therefore consider problem statement Eq. (3') with every $x_i \ge 0$. At each step of the algorithm we assume that the columns are permuted so that

$$\hat{A} = [\overbrace{A_1}^{M} : \overbrace{A_2}^{N-M}]\}M \quad ,$$

$$\hat{\underset{\sim}{c}}^T = [\underset{\sim}{c}_1^T : \underset{\sim}{c}_2^T] \quad ,$$

$$\underset{\sim}{x}^T = [\underset{\sim}{x}_1^T : \underset{\sim}{x}_2^T]$$

where $\qquad \underset{\sim}{x}_1 = A_1^{-1}\underset{\sim}{h} \quad , \quad \underset{\sim}{x}_2 = \underset{\sim}{0} \quad , \qquad$ with $\qquad \underset{\sim}{x}_1 \ge \underset{\sim}{0}$.

The M by M nonsingular matrix A_1 is the <u>basis matrix</u>; the vector $\underset{\sim}{x}_1$ is the set of <u>basic variables</u>, and the vector $\underset{\sim}{x}_2$ is the set of <u>nonbasic variables</u>. The revised simplex algorithm, as presented below, consists of a main loop with three possible exits in steps (A.2), (A.5) and (A.9).

<u>Loop</u>

Compute reduced costs $z_i = c_i - \underset{\sim}{p}^T \underset{\sim}{a}_i$, $i = M+1,\ldots,N$.

(Here the c_i are components of $\underset{\sim}{c}_2$ and $A_1^T \underset{\sim}{p} = \underset{\sim}{c}_1$.) (A.1)

If $z_i \geq 0$ for $i = M+1,\ldots,N$, then exit Loop with an optimal solution. (A.2)

Else

Choose a nonbasic column vector, $\underset{\sim}{a}_q$, from those $z_i < 0$, $M < i \leq N$. (A.3)

Compute search direction $\underset{\sim}{w} = A_1^{-1} \underset{\sim}{a}_q$. (A.4)

If $w_i \leq 0$ for $i = 1,\ldots,M$, then exit Loop with an indication of an unbounded solution. (A.5)

Else

Determine the step length and the column vector to leave the basis matrix:

$\theta = x_p/w_p = \min[x_i/w_i : w_i > 0,\ i = 1,\ldots,M]$ (A.6)

Update the solution, $\hat{\underset{\sim}{x}}_1 = \underset{\sim}{x}_1 - \theta \underset{\sim}{w}$, $\hat{x}_p = \theta$. (A.7)

Exchange column p and q of A_1. (A.8)

Exit Loop if too many iterations have been taken. (A.9)

<u>End Loop</u>

Figure 1. Statement of the Revised Simplex Algorithm

2.2.1 General Bounds and Free Variables

In order to incorporate general bounds and free variables as stated in Eq. (3') we made three types of modifications to the revised simplex algorithm as shown in Fig. 1.

The first modification is only a remark: additional information must be kept regarding types of bounds that the variables have, and whether a nonbasic variable is zero or at its upper bound.

The second modification is necessary because in the algorithm of Fig. 1, nonbasic variables can only increase. In the more general case of solving Eq. (3') nonbasic variables with upper bounds may decrease from their upper bounds and nonbasic free variables can either increase or decrease. Therefore in step (A.1) of Fig. 1, the reduced costs $\{z_i\}$ are replaced by modified reduced costs $\{\hat{z}_i\}$, where

$$\hat{z}_i = \begin{cases} -|z_i|, & \text{if } x_i \text{ is free} \\ -z_i, & \text{if } x_i = e_i = \text{upper bound} \\ z_i, & \text{if } x_i = 0 \end{cases} .$$

This modification also implies that if the nonbasic variable chosen in Step (A.3) is at its upper bound or is free with $z_i > 0$, then the search direction of Steps (A.4) and (A.7) has a sign change, $\underset{\sim}{w} := -\underset{\sim}{w}$.

The third major modification of the algorithm involves the step length parameter θ of Step (A.6). The usual restriction is that for $w_i > 0$ and $x_i > 0$, θ must not exceed x_i/w. A new restriction is that a variable with an upper bound must not exceed its upper bound. Therefore, if $w_i < 0$ and x_i has an upper bound e_i, then θ must not exceed $(x_i-e_i)/w_i$. The column exchanged is, of course, the one that restricts θ the most severely. If this corresponds to a variable $x_p = e_p$, then the variable is reflected, $\hat{x}_p = e_p - x_p$. This involves updating the right-hand side, $\ddot{\underset{\sim}{h}} = \underset{\sim}{h} - \underset{\sim}{a}_p e_p$ and recording that variable number p is nonbasic at its upper bound.

The variable being exchanged can, in fact, be the variable entering the basis. This situation implies that the active basis remains fixed for this step. Only the right-hand side is then changed.

2.2.2 Feasibility

At the outset we check the bounds on the variables $\underset{\sim}{x}$ and $\underset{\sim}{w}$ for consistency. Those variables that have both lower and upper bounds must have the lower bound smaller than or equal to the upper bound.

The revised simplex algorithm of Fig. 1, as modified for bounds and free variables, requires that the solution of the system $A_1 \underset{\sim}{x}_1 = \underset{\sim}{h}$ satisfy the required bounds for each component, i.e., $\underset{\sim}{x}_1$ is feasible. Choosing the initial A_1 to define such a feasible $\underset{\sim}{x}_1$ can be a problem. A common approach to solving this problem is a two-phase process. The first phase consists of solving a related LP problem for which an initial feasible $\underset{\sim}{x}_1$ is easy to define and whose optimum provides a feasible point for the given problem. A description of this two-phase process is found in Ref. [1], p. 102.

Our approach to achieving feasibility is similar to the so-called "Big M" or penalty function method, Ref. [4], p. 112. Specifically, we modify the objective function $\underset{\sim}{c}^T\underset{\sim}{x}$ by adding the penalty function $\underset{\sim}{M}^T\underset{\sim}{x}$. The components of $\underset{\sim}{M}$ may change at each iteration and are defined as:

$$M_j = \begin{cases} -\lambda, & \text{if } x_j < 0 \\ 0, & \text{if } 0 \leq x_j \leq e_j \text{ or } x_j \text{ is free} \\ +\lambda, & \text{if } x_j > e_j \end{cases}$$

where

$$\lambda = \begin{cases} 2\|\hat{\underset{\sim}{c}}\|, & \hat{\underset{\sim}{c}} \neq \underset{\sim}{0} \\ 1, & \underset{\sim}{c} = \underset{\sim}{0} \end{cases} .$$

(The value of λ is chosen after the constraint matrix and objective function has been rescaled.)

Normally the initial basis consists of the columns corresponding to the dependent variables, $\underset{\sim}{w}$, in Eq. (1). In this case, the basis matrix is simply an identity matrix with possible sign changes. Given an initial basis, the value for the variables can be calculated and the M_j defined. In this formulation only the basic variables will have nonzero M coefficients. As a variable is exchanged from the basis (it has become feasible and nonbasic), its corresponding M value is set to zero. Once an $M_j = 0$, it remains zero for all remaining iterations. In order to preserve continuity of the modified objective function, we allow an $x_j < 0$ to increase only to zero and an $x_j > e_j$ to decrease only to e_j. A (slight) possibility remains that when the optimality conditions are met in step (A.2) of Fig. 1, the resulting solutions will not be feasible. If it is not feasible we enter a two-phase form of the algorithm by (formally) redefining $\lambda = +\infty$. The actual mechanics involve the scale factors for the vectors $\underset{\sim}{c}$ and $\underset{\sim}{M}$ used in subprogram SPLP( ). This has the effect of eliminating numerical difficulties due to disparate scaling that can arise in the "Big M" or penalty function method.

The "Big M" method, as modified, was used because of its efficiency. Frequently users are interested in changes to the bounds or other problem data after one solution has been computed. When an LP problem is well posed, small perturbations in the problem data should cause small changes in the solution. If the perturbations cause the original solution to be an infeasible starting point for the new problem, there is a problem with respect to efficiency for two-phase algorithms. In this case, a two-phase LP code cannot take advantage of knowing the original solution and must start over with Phase 1. On the other hand, the "Big M" method, as implemented by SPLP( ), will typically involve only a few new nonzero M_i. Almost always the "Big M" method will reach an optimal solution with fewer iterations.

2.2.3 Pricing Strategies

Step (A.3) of the algorithm of Fig. 1 indicates that for some i, $M < i \leq N$ with $z_i < 0$, the column vector $\underset{\sim}{a_i}$ enters the (active) basis. The method for choosing i is often called the "pricing strategy." The most common pricing strategy is to choose the i which corresponds to the minimum of all reduced costs $z_i < 0$, Ref. [1], p. 156. A more elaborate and more expensive technique is to choose i corresponding to the minimum of all weighted reduced costs $z_i/y_i < 0$. The $y_i > 0$ are weights such that the $\underset{\sim}{a_i}$ chosen to enter the basis corresponds to an edge of steepest descent, Ref. [5]. Numerical results based on certain test problems indicate that this technique often results in significantly fewer iterations than the "minimum reduced cost" method. For this reason our nominal pricing strategy is the steepest edge strategy. However, we do allow the user to optionally revert to the "minimum reduced cost" method.

Calculating all of the reduced costs, z_i, for Step (A.3) can be very expensive. Therefore, we allow the user to choose a "partial pricing" strategy. Instead of calculating all of the reduced costs for Step (A.3), enough z_i's are calculated in order to find a specific number of negative values. The reduced costs are calculated in a "circular" order so that none are missed. Partial pricing is a suboptimal strategy;

the "best" move might not be taken at each iteration. However, especially for problems with many variables, partial pricing may reduce the total amount of work because it reduces so much work at the early stages; see Ref. [2], p. 113-114. Nominally, all the reduced costs are computed for step (A.3).

2.3 Numerical Aspects

2.3.1 Sparsity Considerations

The revised simplex algorithm, Fig. 1, requires solutions of linear algebraic systems of the form $A_1\underset{\sim}{x} = \underset{\sim}{h}$ and $A_1^T\underset{\sim}{p} = \underset{\sim}{c}_1$. These computations are followed by an exchange step wherein one column of A_1 is replaced. For these processes we rely on the Bartels-Golub algorithm as implemented in Ref. [6]. The LA05 package of this reference achieves a good balance between exploiting sparsity of A_1 (to reduce data storage requirements) and the preservation of numerical stability in computing the LU factors, Ref. [3], p. 313. Beside solving the pair of linear algebraic equations $A_1\underset{\sim}{x}_1 = \underset{\sim}{b}$ and $A_1^T\underset{\sim}{p} = \underset{\sim}{c}_1$, operations also must be performed on the nonbasic columns of the constraint matrix, A_2. From the standpoint of efficiency, the most important operation is the dot product $z_i = c_i - \underset{\sim}{a}_i^T\underset{\sim}{p}$. Thus one of our chief concerns is dealing effectively and efficiently with sparsity in the consraint matrix A. To this end, the matrix A is stored using a fairly standard method: the nonzero elements of A are stored sequentially <u>by columns</u>. The indices of the rows within each column, together with their values, are stored in ascending order.

Although SPLP( ) stores the matrix A by columns, the user does not have to define A by columns. There are two distinct ways for the user to define this matrix. The first and the simplier way is to provide SPLP( ) an array defined in a specific format which contains information about each nonzero entry of A. A second way is to provide a user-written subroutine that describes the nonzero entries of A. These matters are dealt with in detail in Section 6.

If the high-speed memory storage for A is too small, the subprogram SPLP( ) automatically stores the data on a direct access file, Ref. [10]. (The user may change the amount of high-speed memory allocated for the storage of A. Nominally there is enough storage allocated for an average of four nonzero entries per column of A.)

Special care has been taken to minimize "page thrashing." The idea is to access columns in the same order as they are stored on the direct access file.

After a "normal" return from SPLP( ) (see Section 6, INFO = 1) the user might be interested in performing calculations involving the matrix A. This is possible because A is not overwritten by SPLP( ) and there are two subprograms PNNZRS( ) and PCHNGS( ) for retrieving and storing entries of the matrix A. These subprograms are slightly modified versions of LNNZRS( ) and LCHNGS( ) of Ref. [10]. The usage of PNNZRS( ) and PCHNGS( ) remains unchanged.

2.3.2 Tolerances

The successful implementation of the revised simplex algorithm of Fig. 1, with the modifications discussed in Section 2.3, requires classification of values according to their signs. Specifically, in Steps (A.2)-(A.3) and (A.5)-(A.6) we must make tests "$z_i > 0$" and "$w_i < 0$." As shown in Ref. [9], robust tests for these conditions can be based on estimating the uncertainty in the numerical solutions of the two linear algebraic systems $A_1 \underset{\sim}{x}_1 = \underset{\sim}{b}$ and $A_1^T \underset{\sim}{p} = \underset{\sim}{c}_1$. We estimate the uncertainty in both systems by the use of row and column check sums. The columns of A_1 and the rows of A_1 are summed to form right-hand side vectors $\hat{\underset{\sim}{h}}$ and $\hat{\underset{\sim}{c}}$ such that the true solution of both systems is $\underset{\sim}{x}_1 = \underset{\sim}{p} = (1,\ldots,1)^T$. The absolute error in each component of the approximate solution $\hat{\underset{\sim}{x}}_1$ and $\hat{\underset{\sim}{p}}$ of these systems is used as an estimate of the relative error in each component of the respective systems $A_1 \underset{\sim}{x}_1 = \underset{\sim}{b}$ and $A_1^T \underset{\sim}{p} = \underset{\sim}{c}_1$.

This relatively simple idea has an attractive side-effect: the accuracy of the machine is automatically reflected in the error estimates because of the check sum approximate solution. So if a user is satisfied with the point of view of obtaining as accurate a solution as possible (in that precision), then no further adjustment of tolerances is required. In particular, this feature obviates the need to do any local "tuning" of tolerances. This fact is fundamental in achieving portability.

2.3.3 Finite Convergence

One concern about LP software is the convergence of the revised simplex algorithm in a reasonable number of steps. There are, in fact, examples by Hoffman and Beale, Ref. [1], p. 229-230, that show that cycling (no convergence) can occur in the algorithm.

Another class of examples presented by Klee and Minty, Ref. [3], show that the algorithm may traverse the main loop exponentially often. Since this number is so large for even modest values of MRELAS and NVARS, the algorithm has virtually failed in this case.

Ideally LP software based on the revised simplex algorithm should alleviate both of these potential types of counterexamples. We have not achieved this ideal but we offer the following remarks.

Our observation about both of these counterexamples is that they are sensitive to normalized scaling of the data, i.e., proper scaling alleviates the difficulty. This type of scaling is done automatically by SPLP( ).

An ad hoc but effective additional technique that we use in SPLP( ) to prevent cycling is to note, at Step (A.6) of Fig. 1, whenever $\theta \doteq 0$. Those variables that are exchanged with $\theta \doteq 0$ are not allowed to reenter the basis until either $\theta > 0$ at some future step, or at Step (A.2) all eligible $z_i > 0$. While this appears to be a viable method for the virtual prevention of cycling, it will not prevent cycling from occuring on Beale's example, Ref. [1], p. 230, when no data scaling is performed.

As an additional safeguard, we have an iteration counter that can cause an exit from the loop at Step (A.9). Nominally the maximum number of iterations allowed is 3(NVARS + MRELAS). The user can reset this value as an option. In fact this feature may prove useful in other ways such as performing a fixed number of iterations and then saving the partial results for completion at a later time.

3. The Solution Returned

The subprogram SPLP( ) can return to the user for several reasons, most importantly when it has found a solution. In this case the primal solution, the dual solution, and the indices of the basic variables are returned to the user.

The primal solution is calculated by first solving for the values of the MRELAS basic variables and then appropriately rescaling and translating and reflecting to return to the original form of the problem. The nonbasic variables are set to either their upper or lower bound. Because of the generality of the problem, the dual solution needs more explanation.

To derive the primal-dual relationship for Eq. (1) we follow Ref. [1], p. 126. In order to use the development given there it is necessary to group the variables $\underset{\sim}{x}$ and $\underset{\sim}{w}$ into four natural categories. Let $\underset{\sim}{x} = (\underset{\sim}{x}_1^T, \underset{\sim}{x}_2^T, \underset{\sim}{x}_3^T, \underset{\sim}{x}_4^T)^T$ and $\underset{\sim}{w} = (\underset{\sim}{w}_1^T, \underset{\sim}{w}_2^T, \underset{\sim}{w}_3^T, \underset{\sim}{w}_4^T)^T$. The $\underset{\sim}{x}_j$ and $\underset{\sim}{w}_i$ have the following bounds: $\underset{\sim}{x}_1 > \underset{\sim}{\alpha}_1$, $\underset{\sim}{x}_2 < \underset{\sim}{\beta}_1$, $\underset{\sim}{\gamma}_1 < \underset{\sim}{x}_3 < \underset{\sim}{\delta}_1$, $\underset{\sim}{x}_4$ free; $\underset{\sim}{w}_1 > \underset{\sim}{\alpha}_2$, $\underset{\sim}{w}_2 < \underset{\sim}{\beta}_2$, $\underset{\sim}{\gamma}_2 < \underset{\sim}{w}_3 < \underset{\sim}{\delta}_2$, $\underset{\sim}{w}_4$ free. With these categories of $\underset{\sim}{w}_j$ and $\underset{\sim}{w}_i$ naturally partitioning the constraints $\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{w}$, the primal-dual relations are as follows.

<u>Primal Problem Statement</u>

minimize $\underset{\sim}{c}_1^T\underset{\sim}{x}_1 + \underset{\sim}{c}_2^T\underset{\sim}{x}_2 + \underset{\sim}{c}_3^T\underset{\sim}{x}_3 + \underset{\sim}{c}_4^T\underset{\sim}{x}_4$

subject to

$$
\begin{array}{lllllllll}
A_{11}\underset{\sim}{x}_1 & + A_{12}\underset{\sim}{x}_2 & + A_{13}\underset{\sim}{x}_3 & + A_{14}\underset{\sim}{x}_4 & - \underset{\sim}{w}_1 & & & & = \underset{\sim}{0} \\
A_{21}\underset{\sim}{x}_1 & + A_{22}\underset{\sim}{x}_2 & + A_{23}\underset{\sim}{x}_3 & + A_{24}\underset{\sim}{x}_4 & & - \underset{\sim}{w}_2 & & & = \underset{\sim}{0} \\
A_{31}\underset{\sim}{x}_1 & + A_{32}\underset{\sim}{x}_2 & + A_{33}\underset{\sim}{x}_3 & + A_{34}\underset{\sim}{x}_4 & & & - \underset{\sim}{w}_3 & & = \underset{\sim}{0} \\
A_{41}\underset{\sim}{x}_1 & + A_{42}\underset{\sim}{x}_2 & + A_{43}\underset{\sim}{x}_3 & + A_{44}\underset{\sim}{x}_4 & & & & - \underset{\sim}{w}_4 & = \underset{\sim}{0} \\
\underset{\sim}{x}_1 & & & & & & & & > \underset{\sim}{\alpha}_1 \\
 & - \underset{\sim}{x}_2 & & & & & & & > -\underset{\sim}{\beta}_1 \\
 & & \underset{\sim}{x}_3 & & & & & & > \underset{\sim}{\gamma}_1 \\
 & & - \underset{\sim}{x}_3 & & & & & & > -\underset{\sim}{\delta}_1 \\
 & & & & \underset{\sim}{w}_1 & & & & > \underset{\sim}{\alpha}_2 \\
 & & & & & - \underset{\sim}{w}_2 & & & > -\underset{\sim}{\beta}_2 \\
 & & & & & & \underset{\sim}{w}_3 & & > \underset{\sim}{\gamma}_2 \\
 & & & & & & \underset{\sim}{w}_3 & & > \underset{\sim}{\delta}_2
\end{array}
$$

<u>Dual Problem Statement</u>

maximize

$$
\begin{aligned}
& \underset{\sim}{\alpha}_1^T\underset{\sim}{v}_5 - \underset{\sim}{\beta}_1^T\underset{\sim}{v}_6 + \underset{\sim}{\gamma}_1^T\underset{\sim}{v}_7 - \underset{\sim}{\delta}_1^T\underset{\sim}{v}_8 \\
+ & \underset{\sim}{\alpha}_2^T\underset{\sim}{v}_9 - \underset{\sim}{\beta}_2^T\underset{\sim}{v}_{10} + \underset{\sim}{\gamma}_2^T\underset{\sim}{v}_{11} - \underset{\sim}{\delta}_2^T\underset{\sim}{v}_{12}
\end{aligned}
$$

subject to

$$
\begin{array}{l}
A_{11}^T v_1 + A_{21}^T v_2 + A_{31}^T v_3 + A_{41}^T v_4 + v_5 = c_1 \\
A_{12}^T v_1 + A_{22}^T v_2 + A_{32}^T v_3 + A_{42}^T v_4 - v_6 = c_2 \\
A_{13}^T v_1 + A_{23}^T v_2 + A_{33}^T v_3 + A_{43}^T v_4 + v_7 - v_8 = c_3 \\
A_{14}^T v_1 + A_{24}^T v_2 + A_{34}^T v_3 + A_{44}^T v_4 = c_4 \\
v_9 - v_1 = 0 \\
v_{10} + v_2 = 0 \\
v_{11} - v_{12} - v_3 = 0 \\
v_4 = 0
\end{array}
$$

v_1, v_2, v_3 are free; $v_4 = 0$; $v_5, \ldots, v_{12} \geq 0$.

Due to the simple dependence of $v_9, v_{10}, v_{11}, v_{12}$ on v_1, v_2, v_3, it is clear that only the dual variables v_i, $i = 1, \ldots, 8$ need to be computed. Also note that the variables v_7, v_8 (as well as v_{11}, v_{12}) are complementary, i.e., v_{7_i} and v_{8_i} cannot simultaneously be nonzero.

The subprogram SPLP( ) computes the MRELAS + NVARS vector $(v_1^T, v_2^T, v_3^T, 0^T, -v_6^T, v_7^T - v_8^T, 0^T)^T$ as output parameters. The first group of zeros in this vector corresponds to $v_4 = 0$. The second group of zeros corresponds to the variables x_4 that are free.

If the stated problem is infeasible (there are no values of x and w such that $Ax = w$ and the bounds on x and w are simultaneously satisfied) an offending set of components is indicated. See the subprogram documentation in Section 6. for details. Another possibility is that the problem is unbounded. There is no finite optimum for $c^T x$ within the constraint set for x and w. In this case the offending set of variables in x is indicated. Further details about this situation are also in Section 6.

4. An Example Solving Related LP Problems

In Ref. [11], p. 9, a sequence of three linear programming problems is discussed and solved:

1. Minimize $-5x_1 - 8x_2 - 5x_3 - 6x_4 - 7x_5$

Subject to
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & -1 & -3 & -5 \\ 2 & 2 & 3 & 0 & 0 \\ -3 & 0 & 4 & 5 & 6 \\ -9 & 3 & -3 & 0 & -1 \\ -4 & 0 & -2 & -1 & 5 \end{bmatrix} \underset{\sim}{x} < \begin{bmatrix} 4 \\ 6 \\ 4 \\ 6 \\ 9 \\ 4 \end{bmatrix}$$

$$\underset{\sim}{x} > \underset{\sim}{0}$$

2. Add the constraint to the problem 1:

$$5x_1 + 8x_2 + 5x_3 + 6x_4 + 7x_5 < 23 \quad .$$

3. Add a new column to the constraint matrix, $\underset{\sim}{a}_6 = (1,0,0,0,0,0,0)^T$. The cost coefficient is -30.

(Restart problems 2 and 3 from the solutions of 1 and 2, respectively.)

Our approach to solving these problems is to consider the enlarged problem:

Minimize $-5x_1 - 8x_2 - 5x_3 - 6x_4 - 7x_5 - 30x_6$

Subject to
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & -1 & -3 & 5 & 0 \\ 2 & 2 & 3 & 0 & 0 & 0 \\ -3 & 0 & 4 & 5 & 6 & 0 \\ -9 & 3 & -3 & 0 & -1 & 0 \\ -4 & 0 & -2 & -1 & 5 & 0 \\ 5 & 8 & 5 & 6 & 7 & 0 \end{bmatrix} \underset{\sim}{x} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} = \underset{\sim}{w} \quad .$$

To solve problems 1, 2, and 3 above, we modify the bounds on the vectors $\underset{\sim}{x}$ and $\underset{\sim}{w}$:

1'. $\underset{\sim}{x} > 0$, $x_6 = 0$, $w_1 < 4$, $w_2 < 6$, $w_3 < 4$, $w_4 < 6$, $w_5 < 9$, $w_6 < 4$, w_7 free.

2'. $\underset{\sim}{x} > \underset{\sim}{0}$, $x_6 = 0$, same bounds as 1' for w_i, $i = 1,\ldots,6$, $w_7 < 23$.

3'. $\underset{\sim}{x} > \underset{\sim}{0}$, same bounds as 2' for $\underset{\sim}{w}_i$, $i = 1,\ldots,7$.

Note that problem 1' with w_7 free has the additional constraint of problem 2 effectively ignored. The constraint $x_6 = 0$ yields a solution for problem 1. By bounding $w_7 < 23$ we obtain a solution for problem 2. Finally by replacing the constraint $x_6 = 0$ by $x_6 > 0$ we obtain a solution for problem 3. The output primal and dual solutions are listed in Fig. 2. A program unit for solving these problems is listed in Fig. 3. Note that the source language used is SFTRAN3, Ref. [12], a Fortran preprocessor language.

	Problem 1'				Problem 2'				Problem 3'			
	$\underset{\sim}{x}$	$\underset{\sim}{w}$	duals		$\underset{\sim}{x}$	$\underset{\sim}{w}$	duals		$\underset{\sim}{x}$	$\underset{\sim}{w}$	duals	
1.	4/3	4	-1		1.54	4	0		0	4	-30	
2.	2/3	-2	0		0.3175	-2.524	0		0	0	0	
3.	0	4	-3.5		0.09524	4	0		0	0	0	
4.	2	6	-1	v_2	2.048	6	0	v_2	0	0	0	v_2
5.	0	-10	0		0	-13.19	0		0	0	0	
6.	0	$-7\frac{1}{3}$	0		0	-8.397	0		4	0	0	
7.		24	0	v_4		23	-1			0	0	
8.			0				0				25	
9.			0				0				22	
10.			10.5	v_5			0	v_5			25	
11.			0				0				24	v_5
12.			0				0				23	
13.			-29	$-v_8$			-30	$-v_8$			0	

Figure 2. Primal and Dual Solutions for the Sequence of Example LP Problems

```
      PROGRAM MATMOD
      DIMENSION A(7,6),DATTRV(69),PRGOPT(020)
      DIMENSION COSTS(06),BL(13),BU(13),IND(13),PRIMAL(13)
      DIMENSION DUALS(13),IBASIS(13),WORK(167),IWORK(226)
      EXTERNAL USRMAT
      DATA ZERO,MRELAS,NVARS,LW,LIW /0.E0,7,6,167,226/
C
C     DEFINE COMPONENTS OF THE COSTS(*) ARRAY.
      DATA (COSTS(J),J=1,6) /-5.0,-8.0,-5.0,-6.0,-7.0,-30.0/
C
C     DEFINE THE CONSTRAINT MATRIX OF THE EXAMPLE.
      DATA ((A(I,J),J=1,6),I=1,7) /6*1.0, 2*2.0,-1.0,-3.0,5.0,0.0,
     *2*2.0,3.0,3*0.0, -3.0,0.0,4.0,5.0,
     *6.0,0.0, -9.0,3.0,-3.0,0.0,-1.0,0.0,
     *-4.0,0.0,-2.0,-1.0,5.0,0.0,
     *5.0,8.0,5.0,6.0,7.0,0.0 /
C
C     DEFINE BOUNDS FOR THE INDEPENDENT (X) AND THE DEPENDENT (W)
C     VARIABLES.
      DATA (BL(J),J=1,6) /6*0.0/
      DATA (BU(I),I=7,13) /4.0,6.0,4.0,6.0,9.0,4.0,23.0/
C
C     DEFINE THE PROBLEM DIMENSIONS AND MOVE DATA FROM
C     FULL MATRIX REPRESENTATION TO "SPARSE" FORM.
      IP=0
      DO FOR J=1,NVARS
      :  IP=IP+1
      :  DATTRV(IP)=-J
      :  DO FOR I=1,MRELAS
      :  :  IF(A(I,J).NE.ZERO) THEN
      :  :  :  DATTRV(IP+1)=I
      :  :  :  DATTRV(IP+2)=A(I,J)
      :  :  :  IP=IP+2
      :  :  END IF
      :  END FOR
      END FOR
      DATTRV(IP+1)=ZERO
C
C     DEFINE THE OPTIONS TO SAVE AND THEN RESTART AT
C     THE PREVIOUS SOLUTION.
      PRGOPT(01)=4
C
C     KEY=55 IS THE CONTINUAIIUN OPTION.
      PRGOPT(02)=55
      PRGOPT(04)=7
C
C     KEY=57 IS THE SAVE (AFTER COMPLETION) OPTION.
      PRGOPT(05)=57
      PRGOPT(06)=1
      PRGOPT(07)=10
C
C     OBTAIN SUMMARY PRINTED OUTPUT ON FILE NUMBER=I1MACH(2).
C     KEY=51 IS THE PRINT OPTION. KPRINT IS THE LEVEL OF OUTPUT.
      PRGOPT(08)=51
      KPRINT=1
      PRGOPT(09)=KPRINT
      PRGOPT(10)=1
```

```
C
C     EACH X( ) VARIABLE HAS ZERO AS A LOWER BOUND.
      DO FOR J=1,NVARS
      :   IND(J)=1
      END FOR
C
C     EACH W( ) VARIABLE MAY HAVE AN UPPER BOUND.  THE BOUNDS
C     ARE DEFINED IN A DATA STATEMENT.
      DO FOR I=NVARS+1,NVARS+MRELAS
      :   IND(I)=2
      END FOR
C
C     SOLVE PROBLEMS 1., 2. AND 3. BY CHANGING THE BOUNDS ON THE
C     SOLUTION VARIABLES X AND W.
      DO FOR IPROB=1,3
      :  DO CASE (IPROB,3)
      :  CASE 1
C     :  :
C     :  :  NOTE THAT THIS FIRST CASE DOES NOT RESTART.
      :  :  PRGOPT(03)=0
C     :  :
C     :  :  CONSTRAIN X(NVARS) TO HAVE ZERO AS AN UPPER BOUND.
      :  :  BU(NVARS)=ZERO
      :  :  IND(NVARS)=3
C     :  :
C     :  :  REMOVE CONSTRAINT EQUATION NO. MRELAS BY LETTING W(MRELAS)
C     :  :  BE A FREE VARIABLE.
      :  :  IND(NVARS+MRELAS)=4
      :  CASE 2
C     :  :
C     :  :  RESTART THE 2ND PROBLEM FROM THE SOLUTION OF THE FIRST.
      :  :  PRGOPT(03)=1
C     :  :
C     :  :  ADD THE CONSTRAINT EQUATION NO. MRELAS BY LETTING W(MRELAS)
C     :  :  HAVE AN UPPER BOUND.
      :  :  IND(NVARS+MRELAS)=2
C     :  :
C     :  :  NONE OF THE MATRIX ENTRIES CHANGE FOR PROBLEMS 2. AND 3.
      :  :  DATTRV(1)=ZERO
      :  CASE 3
C     :  :
C     :  :  LET X(NVARS) NO LONGER BE FIXED AT ZERO.  LET IT BECOME
C     :  :  OPTIMAL AND NONNEGATIVE.
      :  :  IND(NVARS)=1
      :  END CASE
      :  CALL SPLP(USRMAT,MRELAS,NVARS,COSTS,PRGOPT,DATTRV,
     *:  BL,BU,IND,INFO,PRIMAL,DUALS,IBASIS,WORK,LW,IWORK,LIW)
C     :
C     :  NEED TO UNLOAD PAGE STORAGE FILE WHEN DOING RESTARTS IN SAME JOB.
      :  IPAGEF=1
      :  CLOSE(UNIT=IPAGEF,STATUS='DELETE')
      END FOR
      STOP
      END
```

5. Acquiring and Installing SPLP( )

Figure 4 gives a list of the subprograms in the SPLP( ) package. These codes are written in the Fortran preprocessor language SFTRAN3, Ref. [12]. They are also available in Fortran 77, which is output from the SFTRAN3 processor. There are various documents, test drivers and data sets that are useful when installing the SPLP( ) system on a machine. Because of the wide variety of types of textual material associated with this package, we recommend that the physical distribution be accomplished using the TES system, Ref. [13]. Other arrangements can be made by contacting the authors.

Name	Purpose
SPLP	Interface to main worker subprogram SPLPMN.
SPLPMN	Manages other subprograms in package and allocates working storage.
SPOPT	Process option array.
SPLPUP	Process matrix and bound data.
SPINIT	Initialize values for SPLP package.
SPLPDM	Decompose basis matrix using modified LA05 package.
SPLPCE	Estimate error in primal and dual systems.
SPLPCW	Compute steepest edge weights and initialize reduced costs.
SPLPFE	Find variable to enter basis.
SPLPFL	Find variable to leave basis.
SPLPMU	Update primal solution, edge weights, reduced costs, and matrix composition.
USRMAT	Decodes sparse matrix if provided by user in DATTRV(*).
PINITM, IPLOC, PRWPGE, PRWVIR, PNNZRS, PCHNGS, OPENM, CLOSM, READM, WRITM	Sparse matrix storage and retrieval package.
SVOUT, IVOUT	Vector output subprogrms.
LA05AS, LA05BS, LA05CS, LA05ES, MC20AS	Sparse matrix decomposition and updating package, Ref. [6].
I1MACH, R1MACH	Environmental parameter subprograms, Ref. [15]. (These are the only routines that require modification if Fortran 77 is used.)
SCOPY, SASUM, SDOT	Subprograms from the BLAS package, Ref. [18].
XERROR, XERRWV, + several others	Error message handling package, Ref. [14].

Figure 4. Subprograms in the SPLP( ) Package.

6. Documentation and Error Messages for Using SPLP( )

The following document gives complete usage instructions for the linear programming subprogram SPLP( ). There is an abbreviated version of this documentation (not shown here) that is provided with the complete package. This shorter document may be useful in machine-readable library systems or in reminding the user of features of SPLP( ) that are intricate.

Machine Readable Documentation

```
C      SUBROUTINE SPLP(USRMAT,MRELAS,NVARS,COSTS,PRGOPT,DATTRV,
C     *            BL,BU,IND,INFO,PRIMAL,DUALS,IBASIS,WORK,LW,IWORK,LIW)
C
C      |------------|
C      |Introduction|
C      |------------|
C      The subprogram SPLP( ) solves a linear optimization problem.
C      The problem statement is as follows
C
C                           minimize (transpose of costs)*x
C                           subject to A*x=w.
C
C      The entries of the unknowns x and w may have simple lower or
C      upper bounds (or both), or be free to take on any value.  By
C      setting the bounds for x and w, the user is imposing the con-
C      straints of the problem.
C
C      (The problem may also be stated as a maximization
C      problem.  This is done by means of input in the option array
C      PRGOPT(*).)  The matrix A has MRELAS rows and NVARS columns.  The
C      vectors costs, x, and w respectively have NVARS, NVARS, and
C      MRELAS number of entries.
C
C      The input for the problem includes the problem dimensions,
C      MRELAS and NVARS, the array COSTS(*), data for the matrix
C      A, and the bound information for the unknowns x and w, BL(*),
C      BU(*), and IND(*).
C
C      The output from the problem (when output flag INFO=1) includes
C      optimal values for x and w in PRIMAL(*), optimal values for
C      dual variables of the equations A*x=w and the simple bounds
C      on x in  DUALS(*), and the indices of the basic columns in
C      IBASIS(*).
C
C  |-------------------------------|
C  |Fortran Declarations Required:|
C  |-------------------------------|
C
C      DIMENSION COSTS(NVARS),PRGOPT(*),DATTRV(*),
C     *BL(NVARS+MRELAS),BU(NVARS+MRELAS),IND(NVARS+MRELAS),
C     *PRIMAL(NVARS+MRELAS),DUALS(MRELAS+NVARS),IBASIS(NVARS+MRELAS),
C     *WORK(LW),IWORK(LIW)
C
C      EXTERNAL USRMAT (or 'NAME', if user provides the subprogram)
C
```

```
C     The dimensions of PRGOPT(*) and DATTRV(*) must be at least 1.
C     The exact lengths will be determined by user-required options and
C     data transferred to the subprogram USRMAT( ) ( or 'NAME').
C
C     The values of LW and LIW, the lengths of the arrays WORK(*)
C     and IWORK(*), must satisfy the inequalities
C
C               LW .GE. 4*NVARS+ 8*MRELAS+LAMAT+  LBM
C               LIW.GE.   NVARS+11*MRELAS+LAMAT+2*LBM
C
C     It is an error if they do not both satisfy these inequalities.
C     (The subprogram will inform the user of the required lengths
C     if either LW or LIW is wrong.)  The values of LAMAT and LBM
C     nominally are
C
C               LAMAT=4*NVARS+7
C     and       LBM  =8*MRELAS
C
C     These values will be as shown unless the user changes them by
C     means of input in the option array PRGOPT(*).  The value of LAMAT
C     determines the length of the sparse matrix "staging" area.
C     For reasons of efficiency the user may want to increase the value
C     of LAMAT.  The value of LBM determines the amount of storage
C     available to decompose and update the active basis matrix.
C     Due to exhausting the working space because of fill-in,
C     it may be necessary for the user to increase the value of LBM.
C     (If this situation occurs an informative diagnostic is printed
C     and a value of INFO=-28 is obtained as an output parameter.)
C
C  |------|
C  |Input:|
C  |------|
C
C     MRELAS,NVARS
C     ------------
C     These parameters are respectively the number of constraints (the
C     linear relations A*x=w that the unknowns x and w are to satisfy)
C     and the number of entries in the vector x.  Both must be .GE. 1.
C     Other values are errors.
C
C     COSTS(*)
C     --------
C     The NVARS entries of this array are the coefficients of the
C     linear objective function.  The value COSTS(J) is the
C     multiplier for variable J of the unknown vector x.  Each
C     entry of this array must be defined.  This array can be changed
C     by the user between restarts.  See options with KEY=55,57 for
C     details of checkpointing and restarting.
C
C     USRMAT
C     ------
C     This is the name of a specific subprogram in the SPLP( ) package
C     that is used to define the matrix entries when this data is passed
C     to SPLP( ) as a linear array.  In this usage mode of SPLP( )
C     the user gives information about the nonzero entries of A
C     in DATTRV(*) as given under the description of that parameter.
C     The name USRMAT must appear in a Fortran EXTERNAL statement.
C     User's who are passing the matrix data with USRMAT( ) can skip
C     directly to the description of the input parameter DATTRV(*).
C
```

```
C     If the user chooses to provide a subprogram 'NAME'( ) to
C     define the matrix A, then DATTRV(*) may be used to pass floating
C     point data from the user's program unit to the subprogram
C     'NAME'( ). The content of DATTRV(*) is not changed in any way.
C
C     The subprogram 'NAME'( ) can be of the user's choice
C     but it must meet Fortran standards and it must appear in a
C     Fortran EXTERNAL statement.  The first statement of the subprogram
C     has the form
C
C            SUBROUTINE 'NAME'(I,J,AIJ, INDCAT, PRGOPT, DATTRV, IFLAG)
C
C    The variables I,J, INDCAT, IFLAG(10) are type INTEGER,
C            while  AIJ, PRGOPT(*),DATTRV(*) are type REAL.
C
C     The user interacts with the contents of IFLAG(*) to
C     direct the appropriate action.  The algorithmic steps are
C     as follows.
C
C            Test IFLAG(1).
C
C               IF(IFLAG(1).EQ.1) THEN
C
C                 Initialize the necessary pointers and data
C                 for defining the matrix A.  The contents
C                 of IFLAG(K), K=2,...,10, may be used for
C                 storage of the pointers.  This array remains intact
C                 between calls to 'NAME'( ) by SPLP( ).
C                 RETURN
C
C               END IF
C
C               IF(IFLAG(1).EQ.2) THEN
C
C                 Define one set of values for I,J,AIJ, and INDCAT.
C                 Each nonzero entry of A must be defined this way.
C                 These values can be defined in any convenient order.
C                 (It is most efficient to define the data by
C                 columns in the order 1,...,NVARS; within each
C                 column define the entries in the order 1,...,MRELAS.)
C                 If this is the last matrix value to be
C                 defined or updated, then set IFLAG(1)=3.
C                 (When I and J are positive and respectively no larger
C                 than MRELAS and NVARS, the value of AIJ is used to
C                 define (or update) row I and column J of A.)
C                 RETURN
C
C               END IF
C
C                 END
C
C     Remarks:    The values of I and J are the row and column
C                 indices for the nonzero entries of the matrix A.
C                 The value of this entry is AIJ.
C                 Set INDCAT=0 if this value defines that entry.
C                 Set INDCAT=1 if this entry is to be updated,
C                              new entry=old entry+AIJ.
C                 A value of I not between 1 and MRELAS, a value of J
C                 not between 1 and NVARS, or a value of INDCAT
C                 not equal to 0 or 1 are each errors.
C
```

```
C               The contents of IFLAG(K), K=2,...,10, can be used to
C               remember the status (of the process of defining the
C               matrix entries) between calls to 'NAME'( ) by SPLP( ).
C               On entry to 'NAME'( ), only the values 1 or 2 will be
C               in IFLAG(1).  More than 2*NVARS*MRELAS definitions of
C               the matrix elements is considered an error because
C               it suggests an infinite loop in the user-written
C               subprogram 'NAME'( ).  Any matrix element not
C               provided by 'NAME'( ) is defined to be zero.
C
C               The REAL arrays PRGOPT(*) and DATTRV(*) are passed as
C               arguments directly from SPLP( ) to 'NAME'( ).
C               The array PRGOPT(*) contains any user-defined program
C               options.  In this usage mode the array DATTRV(*) may
C               now contain any (type REAL) data that the user needs
C               to define the matrix A.  Both arrays PRGOPT(*) and
C               DATTRV(*) remain intact between calls to 'NAME'( )
C               by SPLP( ).
C     Here is a subprogram that communicates the matrix values for A,
C     as represented in DATTRV(*), to SPLP( ).  This subprogram,
C     called USRMAT( ), is included as part of the SPLP( ) package.
C     This subprogram 'decodes' the array DATTRV(*) and defines the
C     nonzero entries of the matrix  A for SPLP( ) to store.  This
C     listing is presented here as a guide and example
C     for the users who find it necessary to write their own subroutine
C     for this purpose.  The contents of DATTRV(*) are given below in
C     the description of that parameter.
C
C     SUBROUTINE USRMAT(I,J,AIJ, INDCAT,PRGOPT,DATTRV,IFLAG)
C    (THIS IS FORTRAN 77 CODING.)
C     DIMENSION PRGOPT(*),DATTRV(*),IFLAG(10)
C
C     IF(IFLAG(1).EQ.1) THEN
C
C     THIS IS THE INITIALIZATION STEP.  THE VALUES OF IFLAG(K),K=2,3,4,
C     ARE RESPECTIVELY THE COLUMN INDEX, THE ROW INDEX (OR THE NEXT COL.
C     INDEX), AND THE POINTER TO THE MATRIX ENTRY'S VALUE WITHIN
C     DATTRV(*).  ALSO CHECK (DATTRV(1)=0.) SIGNIFYING NO DATA.
C          IF(DATTRV(1).EQ.0.) THEN
C          I = 0
C          J = 0
C          IFLAG(1) = 3
C          ELSE
C          IFLAG(2)=-DATTRV(1)
C          IFLAG(3)= DATTRV(2)
C          IFLAG(4)= 3
C          END IF
C
C          RETURN
C     ELSE
C          J=IFLAG(2)
C          I=IFLAG(3)
C          L=IFLAG(4)
C          IF(I.EQ.0) THEN
C
C     SIGNAL THAT ALL OF THE NONZERO ENTRIES HAVE BEEN DEFINED.
C               IFLAG(1)=3
C               RETURN
C          ELSE IF(I.LT.0) THEN
C
```

```
C     SIGNAL THAT A SWITCH IS MADE TO A NEW COLUMN.
C               J=-I
C               I=DATTRV(L)
C               L=L+1
C          END IF
C
C          AIJ=DATTRV(L)
C
C     UPDATE THE INDICES AND POINTERS FOR THE NEXT ENTRY.
C          IFLAG(2)=J
C          IFLAG(3)=DATTRV(L+1)
C          IFLAG(4)=L+2
C
C     INDCAT=0 DENOTES THAT ENTRIES OF THE MATRIX ARE ASSIGNED THE
C     VALUES FROM DATTRV(*).  NO ACCUMULATION IS PERFORMED.
C          INDCAT=0
C          RETURN
C     END IF
C     END
C
C     DATTRV(*)
C     ---------
C     If the user chooses to use the provided subprogram USRMAT( ) then
C     the array DATTRV(*) contains data for the matrix A as follows:
C     Each column (numbered J) requires (floating point) data con-
C     sisting of the value (-J) followed by pairs of values.  Each pair
C     consists of the row index immediately followed by the value
C     of the matrix at that entry.  A value of J=0 signals that there
C     are no more columns.  (See "Example of SPLP( ) Usage," below.)
C
C     If the Save/Restore feature is in use (see options with
C     KEY=55,57 for details of checkpointing and restarting)
C     USRMAT( ) can be used to redefine entries of the matrix.
C     The matrix entries are redefined or overwritten.  No accum-
C     ulation is performed.
C     Any other nonzero entry of A, defined in a previous call to
C     SPLP( ), remain intact.
C
C     BL(*),BU(*),IND(*)
C     ------------------
C     The values of IND(*) are input parameters that define
C     the form of the bounds for the unknowns x and w.  The values for
C     the bounds are found in the arrays BL(*) and BU(*) as follows.
C
C     For values of J between 1 and NVARS,
C          if IND(J)=1, then X(J) .GE. BL(J); BU(J) is not used.
C          if IND(J)=2, then X(J) .LE. BU(J); BL(J) is not used.
C          if IND(J)=3, then BL(J) .LE. X(J) .LE. BU(J),(BL(J)=BU(J) ok)
C          if IND(J)=4, then X(J) is free to have any value,
C          and BL(J), BU(J) are not used.
C
C     For values of I between NVARS+1 and NVARS+MRELAS,
C          if IND(I)=1, then W(I-NVARS) .GE. BL(I); BU(I) is not used.
C          if IND(I)=2, then W(I-NVARS) .LE. BU(I); BL(I) is not used.
C          if IND(I)=3, then BL(I) .LE. W(I-NVARS) .LE. BU(I),
C          (BL(I)=BU(I) is ok).
```

```
C           if IND(I)=4, then W(I-NVARS) is free to have any value,
C           and BL(I), BU(I) are not used.
C
C     A value of IND(*) not equal to 1,2,3 or 4 is an error.  When
C     IND(I)=3, BL(I) must be .LE. BU(I).  The condition BL(I).GT.
C     BU(I) indicates infeasibility and is an error.  These
C     arrays can be changed by the user between restarts.  See
C     options with KEY=55,57 for details of checkpointing and
C     restarting.
C
C     PRGOPT(*)
C     ---------
C     This array is used to redefine various parameters within SPLP( ).
C     Frequently, perhaps most of the time, a user will be satisfied
C     and obtain the solutions with no changes to any of these
C     parameters.  To try this, simply set PRGOPT(1)=1.E0.
C
C     For users with more sophisticated needs, SPLP( ) provides several
C     options that may be used to take advantage of more detailed
C     knowledge of the problem or satisfy other utilitarian needs.
C     The complete description of how to use this option array to
C     utilize additional subprogram features is found under the
C     heading "Usage of SPLP( ) Subprogram Options."
C
C     Briefly, the user should note the following value of the parameter
C     KEY and the corresponding task or feature desired before turning
C     to that section.
C
C     Value       Brief Statement of Purpose for Option
C     of KEY
C     ------      ------------------------------------
C     50          Change from a minimization problem to a
C                 maximization problem.
C     51          Change the amount of printed output.
C                 Normally, no printed output is obtained.
C     52          Redefine the line length and precision used
C                 for the printed output.
C     53          Redefine the values of LAMAT and LBM that
C                 were dicussed above under the heading
C                 Fortran Declarations Required.
C     54          Redefine the unit number where pages of the sparse
C                 data matrix A are stored.  Normally, the unit
C                 number is 1.
C     55          A computation, partially completed, is
C                 being continued.  Read the up-to-date
C                 partial results from unit number 2.
C     56          Redefine the unit number where the partial results
C                 are stored.  Normally, the unit number is 2.
C     57          Save partial results on unit 2 either after
C                 maximum iterations or at the optimum.
C     58          Redefine the value for the maximum number of
C                 iterations.  Normally, the maximum number of
C                 iterations is 3*(NVARS+MRELAS).
C     59          Provide SPLP( ) with a starting (feasible)
C                 nonsingular basis.  Normally, SPLP( ) starts
C                 with the identity matrix columns corresponding
C                 to the vector w.
```

```
C     60         The user has provided scale factors for the
C                columns of A.  Normally, SPLP( ) computes scale
C                factors that are the reciprocals of the max. norm
C                of each column.
C     61         The user has provided a scale factor
C                for the vector costs.  Normally, SPLP( ) computes
C                a scale factor equal to the reciprocal of the
C                max. norm of the vector costs after the column
C                scaling for the data matrix has been applied.
C     62         Size parameters, namely the smallest and
C                largest magnitudes of nonzero entries in
C                the matrix A, are provided.  Values noted
C                outside this range are to be considered errors.
C     63         Redefine the tolerance required in
C                evaluating residuals for feasibility.
C                Normally, this value is set to SQRT(EPS),
C                where EPS = relative precision of the arithmetic.
C     64         Change the criterion for bringing new variables
C                into the basis from the steepest edge (best
C                local move) to the minimum reduced cost.
C     65         Redefine the value for the number of iterations
C                between recalculating the error in the primal
C                solution.  Normally, this value is equal to ten.
C     66         Perform "partial pricing" on variable selection.
C                Redefine the value for the number of negative
C                reduced costs to compute (at most) when finding
C                a variable to enter the basis.  Normally this
C                value is set to NVARS.  This implies that no
C                "partial pricing" is used.
C
C   |---------------|
C   |Working Arrays:|
C   |---------------|
C
C      WORK(*),LW,
C      IWORK(*),LIW
C      ============
C      The arrays WORK(*) and IWORK(*) are respectively floating point
C      and type INTEGER working arrays for SPLP( ) and its
C      subprograms.  The lengths of these arrays are respectively
C      LW and LIW.  These parameters must satisfy the inequalites
C      noted above under the heading "Fortran Declarations Required."
C      It is an error if either value is too small.
C
C   |----------------------------|
C   |Input/Output files required:|
C   |----------------------------|
C
C      Fortran unit 1 is used by SPLP( ) to store the sparse matrix A
C      out of high-speed memory.  This direct access file is opened
C      within the package under the following two conditions.
C      1. When the Save/Restore feature is used.  2. When the
C      constraint matrix is so large that storage out of high-speed
C      memory is required.  The user may need to close unit 1
C      (with deletion from the job step) in the main program unit
C      when several calls are made to SPLP( ).  A crude
```

```
C      upper bound for the amount of information written on unit 1
C      is 6*nz, where nz is the number of nonzero entries in A.
C      The unit number may be redefined to any other positive value
C      by means of input in the option array PRGOPT(*).
C
C      Fortran unit 2 is used by SPLP( ) only when the Save/Restore
C      feature is desired.  Normally this feature is not used.  It is
C      activated by means of input in the option array PRGOPT(*).
C      On some computer systems the user may need to open unit
C      2 before executing a call to SPLP( ).  This file is type
C      sequential and is unformatted.
C
C      Fortran unit=I1MACH(2) (check local setting) is used by SPLP( )
C      when the printed output feature (KEY=51) is used.  Normally
C      this feature is not used.  It is activated by input in the
C      options array PRGOPT(*).  For many computer systems I1MACH(2)=6.
C
C   |-------|
C   |Output:|
C   |-------|
C
C      INFO,PRIMAL(*),DUALS(*)
C      -----------------------
C      The integer flag INFO indicates why SPLP( ) has returned to the
C      user.  If INFO=1 the solution has been computed.  In this case
C      X(J)=PRIMAL(J) and W(I)=PRIMAL(I+NVARS).  The dual variables
C      for the equations A*x=w are in the array DUALS(I)=dual for
C      equation number I.  The dual value for the component X(J) that
C      has an upper or lower bound (or both) is returned in
C      DUALS(J+MRELAS).  The only other values for INFO are .LT. 0.
C      The meaning of these values can be found by reading
C      the diagnostic message in the output file, or by looking for
C      error number = (-INFO) under the heading "List of SPLP( ) Error
C      and Diagnostic Messages."
C      The diagnostic messages are printed using the error processing
C      subprograms XERROR( ) and XERRWV( ) with error category LEVEL=1.
C      See the document "Brief Instr. for Using the Sandia Math.
C      Subroutine Library," SAND79-2382, Nov., 1980, for further inform-
C      ation about resetting the usual response to a diagnostic message.
C
C      BL(*),BU(*),IND(*)
C      ------------------
C      These arrays are output parameters only under the (unusual)
C      circumstances where the stated problem is infeasible, has an
C      unbounded optimum value, or both.  These respective conditions
C      correspond to INFO=-1,-2 or -3.  For INFO=-1 or -3 certain
C      components of the vectors x or w will not satisfy the input bounds.
C      If component J of x or component I of w does not satisfy its input
C      bound because of infeasibility, then IND(J)=-4 or IND(I+NVARS)=-4,
C      respectively.  For INFO=-2 or -3 certain
C      components of the vector x could not be used as basic variables
C      because the objective function would have become unbounded.
C      In particular if component J of x corresponds to such a variable,
C      then IND(J)=-3.  Further, if the input value of IND(J)
C                          =1, then BU(J)=BL(J);
C                          =2, then BL(J)=BU(J);
```

```
C                       =4, then BL(J)=0.,BU(J)=0.
C
C     (The J-th variable in x has been restricted to an appropriate
C     feasible value.)
C     The negative output value for IND(*) allows the user to identify
C     those constraints that are not satisfied or those variables that
C     would cause unbounded values of the objective function.  Note
C     that the absolute value of IND(*), together with BL(*) and BU(*),
C     are valid input to SPLP( ).  In the case of infeasibility the
C     sum of magnitudes of the infeasible values is minimized.  Thus
C     one could reenter SPLP( ) with these components of x or w now
C     fixed at their present values.  This involves setting
C     the appropriate components of IND(*) = 3, and BL(*) = BU(*).
C
C     IBASIS(I),I=1,...,MRELAS
C     ------------------------
C     This array contains the indices of the variables that are
C     in the active basis set at the solution (INFO=1).  A value
C     of IBASIS(I) between 1 and NVARS corresponds to the variable
C     X(IBASIS(I)).  A value of IBASIS(I) between NVARS+1 and NVARS+
C     MRELAS corresponds to the variable W(IBASIS(I)-NVARS).
C
C     Computing with the Matrix A after Calling SPLP( )
C     -------------------------------------------------
C     Following the return from SPLP( ), nonzero entries of the MRELAS
C     by NVARS matrix A are available for usage by the user.  The method
C     for obtaining the next nonzero in column J with a row index
C     strictly greater than I in value, is completed by executing
C
C         CALL PNNZRS(I,AIJ,IPLACE,WORK,IWORK,J)
C
C     The value of I is also an output parameter.  If I.LE.0 on output,
C     then there are no more nonzeroes in column J.  If I.GT.0, the
C     output value for component number I of column J is in AIJ.  The
C     parameters WORK(*) and IWORK(*) are the same arguments as in the
C     call to SPLP( ).  The parameter IPLACE is a single INTEGER
C     working variable.
C
C     The data structure used for storage of the matrix A within SPLP( )
C     corresponds to sequential storage by columns as defined in
C     SAND78-0785.  Note that the names of the subprograms LNNZRS(),
C     LCHNGS(),LINITM(),LLOC(),LRWPGE(), and LRWVIR() have been
C     changed to PNNZRS(),PCHNGS(),PINITM(),IPLOC(),PRWPGE(), and
C     PRWVIR() respectively.  The error processing subprogram LERROR()
C     is no longer used; XERROR() is used instead.
C
C  |-------------------------------|
C  |Subprograms Required by SPLP( )|
C  |-------------------------------|
C     Called by SPLP() are SPLPMN(),SPLPUP(),SPINIT(),SPOPT(),
C                          SPLPDM(),SPLPCE(),SPINCW(),SPLPFL(),
C                          SPLPFE(),SPLPMU().
C
C     Error Processing Subprograms XERROR(),XERRWV(),I1MACH(),R1MACH()
C
C     Sparse Matrix Subprograms PNNZRS(),PCHNGS(),PRWPGE(),PRWVIR(),
```

```
C                                  PINITM(),IPLOC()
C
C      Mass Storage File Subprograms OPENM(),CLOSM(),READM(),WRITM()
C
C      Basic Linear Algebra Subprograms SCOPY(),SASUM(),SDOT()
C
C      Sparse Matrix Basis Handling Subprograms LA05AS(),LA05BS(),
C                                          LA05CS(),LA05ED(),MC20AS()
C
C      Vector Output Subprograms SVOUT(),IVOUT()
C
C      Machine-sensitive Subprograms I1MACH( ),R1MACH( ),
C                                 OPENM( ),CLOSM( ),READM( ),WRITM( ).
C      COMMON Block Used
C      -----------------
C      /LA05DS/ SMALL,LP,LENL,LENU,NCP,LROW,LCOL
C      See the document AERE-R8269 for further details.
C     |-----------------------|
C     |Example of SPLP( ) Usage|
C     |-----------------------|
C      PROGRAM LPEX (FILES=TAPE1,TAPE6 MAY BE USED)
C      THE OPTIMIZATION PROBLEM IS TO FIND X1, X2, X3 THAT
C      MINIMIZE X1 + X2 + X3, X1.GE.0, X2.GE.0, X3 UNCONSTRAINED.
C
C      THE UNKNOWNS X1,X2,X3 ARE TO SATISFY CONSTRAINTS
C
C         X1 -3*X2 +4*X3 = 5
C         X1 -2*X2     .LE.3
C             2*X2 - X3.GE.4
C
C      WE FIRST DEFINE THE DEPENDENT VARIABLES
C           W1=X1 -3*X2 +4*X3
C           W?=X1- 2*X2
C           W3=    2*X2 -X3
C
C      WE NOW SHOW HOW TO USE SPLP( ) TO SOLVE THIS LINEAR OPTIMIZATION
C      PROBLEM.  EACH REQUIRED STEP WILL BE SHOWN IN THIS EXAMPLE.
C      DIMENSION COSTS(03),PRGOPT(01),DATTRV(18),BL(06),BU(06),IND(06),
C     *PRIMAL(06),DUALS(06),IBASIS(06),WORK(079),IWORK(103)
C
C      EXTERNAL USRMAT
C      MRELAS=3
C      NVARS=3
C
C      DEFINE THE ARRAY COSTS(*) FOR THE OBJECTIVE FUNCTION.
C      COSTS(01)=1.
C      COSTS(02)=1.
C      COSTS(03)=1.
C
C      PLACE THE NONZERO INFORMATION ABOUT THE MATRIX IN DATTRV(*).
C      DEFINE COL. 1:
C      DATTRV(01)=-1
C      DATTRV(02)=1
C      DATTRV(03)=1.
C      DATTRV(04)=2
C      DATTRV(05)=1.
```

```
C
C     DEFINE COL. 2:
C     DATTRV(06)=-2
C     DATTRV(07)=1
C     DATTRV(08)=-3.
C     DATTRV(09)=2
C     DATTRV(10)=-2.
C     DATTRV(11)=3
C     DATTRV(12)=2.
C
C     DEFINE COL. 3:
C     DATTRV(13)=-3
C     DATTRV(14)=1
C     DATTRV(15)=4.
C     DATTRV(16)=3
C     DATTRV(17)=-1.
C
C     DATTRV(18)=0
C
C     CONSTRAIN X1,X2 TO BE NONNEGATIVE. LET X3 HAVE NO BOUNDS.
C     BL(1)=0.
C     IND(1)=1
C     BL(2)=0.
C     IND(2)=1
C     IND(3)=4
C
C     CONSTRAIN W1=5,W2.LE.3, AND W3.GE.4.
C     BL(4)=5.
C     BU(4)=5.
C     IND(4)=3
C     BU(5)=3.
C     IND(5)=2
C     BL(6)=4.
C     IND(6)=1
C
C     INDICATE THAT NO MODIFICATIONS TO OPTIONS ARE IN USE.
C     PRGOPT(01)=1
C
C     DEFINE THE WORKING ARRAY LENGTHS.
C     LW=079
C     LIW=103
C     CALL SPLP(USRMAT,MRELAS,NVARS,COSTS,PRGOPT,DATTRV,
C    *BL,BU,IND,INFO,PRIMAL,DUALS,IBASIS,WORK,LW,IWORK,LIW)
C
C     CALCULATE VAL, THE MINIMAL VALUE OF THE OBJECTIVE FUNCTION.
C     VAL=SDOT(NVARS,COSTS,1,PRIMAL,1)
C
C     STOP
C     END
C     |-----------------------|
C     |End of Example of Usage |
C     |-----------------------|
C
C     |-------------------------------------|
C     |Usage of SPLP( ) Subprogram Options.|
C     |-------------------------------------|
C
```

```
C     Users frequently have a large variety of requirements for linear
C     optimization software.  Allowing for these varied requirements
C     is at cross purposes with the desire to keep the usage of SPLP( )
C     as simple as possible. One solution to this dilemma is as follows.
C     (1) Provide a version of SPLP( ) that solves a wide class of
C     problems and is easy to use. (2) Identify parameters within SPLP()
C     that certain users may want to change.  (3) Provide a means
C     of changing any selected number of these parameters that does
C     not require changing all of them.
C
C     Changing selected parameters is done by requiring
C     that the user provide an option array, PRGOPT(*), to SPLP( ).
C     The contents of PRGOPT(*) inform SPLP( ) of just those options
C     that are going to be modified within the total set of possible
C     parameters that can be modified.  The array PRGOPT(*) is a linked
C     list consisting of groups of data of the following form
C
C          LINK
C          KEY
C          SWITCH
C          data set
C
C     that describe the desired options.  The parameters LINK, KEY and
C     switch are each one word and are always required.  The data set
C     can be comprised of several words or can be empty.  The number of
C     words in the data set for each option depends on the value of
C     the parameter KEY.
C
C     The value of LINK points to the first entry of the next group
C     of data within PRGOPT(*).  The exception is when there are no more
C     options to change.  In that case, LINK=1 and the values for KEY,
C     SWITCH and data set are not referenced.  The general layout of
C     PRGOPT(*) is as follows:
C          ...PRGOPT(1)=LINK1 (link to first entry of next group)
C          .  PRGOPT(2)=KEY1 (KEY to the option change)
C          .  PRGOPT(3)=SWITCH1 (on/off switch for the option)
C          .  PRGOPT(4)=data value
C          .       .
C          .       .
C          .       .
C          ...PRGOPT(LINK1)=LINK2 (link to first entry of next group)
C          .  PRGOPT(LINK1+1)=KEY2 (KEY to option change)
C          .  PRGOPT(LINK1+2)=SWITCH2 (on/off switch for the option)
C          .  PRGOPT(LINK1+3)=data value
C          ...     .
C          .       .
C          .       .
C          ...PRGOPT(LINK)=1 (no more options to change)
C
C     A value of LINK that is .LE.0 or .GT. 10000 is an error.
C     In this case SPLP( ) returns with an error message, INFO=-14.
C     This helps prevent using invalid but positive values of LINK that
C     will probably extend beyond the program limits of PRGOPT(*).
C     Unrecognized values of KEY are ignored.  If the value of SWITCH is
C     zero then the option is turned off.  For any other value of SWITCH
C     the option is turned on.  This is used to allow easy changing of
```

```
C     options without rewriting PRGOPT(*).  The order of the options is
C     arbitrary and any number of options can be changed with the
C     following restriction.  To prevent cycling in processing of the
C     option array PRGOPT(*), a count of the number of options changed
C     is maintained.  Whenever this count exceeds 1000 an error message
C     (INFO=-15) is printed and the subprogram returns.
C
C     In the following description of the options, the value of
C     LATP indicates the amount of additional storage that a particular
C     option requires.  The sum of all of these values (plus one) is
C     the minimum dimension for the array PRGOPT(*).
C
C     If a user is satisfied with the nominal form of SPLP( ),
C     set PRGOPT(1)=1 (or PRGOPT(1)=1.E0).
C
C     Options:
C
C-----KEY = 50.  Change from a minimization problem to a maximization
C     problem.
C     If SWITCH=0  option is off; solve minimization problem.
C              =1  option is on; solve maximization problem.
C     data set =empty
C     LATP=3
C
C-----KEY = 51.  Change the amount of printed output.  The nominal form
C     of SPLP( ) has no printed output.
C     The first level of output (SWITCH=1) includes
C
C     (1) Minimum dimensions for the arrays COSTS(*),BL(*),BU(*),IND(*),
C         PRIMAL(*),DUALS(*),IBASIS(*), and PRGOPT(*).
C     (2) Problem dimensions MRELAS,NVARS.
C     (3) The types of and values for the bounds on x and w,
C         and the values of the components of the vector costs.
C     (4) Whether optimization problem is minimization or
C         maximization.
C     (5) Whether steepest edge or smallest reduced cost criteria used
C         for exchanging variables in the revised simplex method.
C
C     Whenever a solution has been found, (INFO=1),
C
C     (6) the value of the objective function,
C     (7) the values of the vectors x and w,
C     (8) the dual variables for the constraints A*x=w and the
C         bounded components of x,
C     (9) the indices of the basic variables,
C    (10) the number of revised simplex method iterations,
C    (11) the number of full decompositions of the basis matrix.
C
C     The second level of output (SWITCH=2) includes all for SWITCH=1
C     plus
C
C    (12) the iteration number,
C    (13) the column number to enter the basis,
C    (14) the column number to leave the basis,
C    (15) the length of the step taken.
C
```

```
C      The third level of output (SWITCH=3) includes all for SWITCH=2
C      plus
C     (16) critical quantities required in the revised simplex method.
C           This output is rather voluminous.  It is intended to be used
C           as a diagnostic tool in case of a failure in SPLP( ).
C
C      If SWITCH=0  option is off; no printed output.
C               =1  summary output.
C               =2  lots of output.
C               =3  even more output.
C      data set =empty
C      LATP=3
C
C-----KEY = 52.  Redefine the parameter, IDIGIT, which determines the
C      format and precision used for the printed output.  In the printed
C      output, at least ABS(IDIGIT) decimal digits per number is printed.
C      If IDIGIT.LT.0, 72 printing columns are used.  IF IDIGIT.GT.0, 133
C      printing columns are used.
C      If SWITCH=0  option is off; IDIGIT=-4.
C               =1  option is on.
C      data set =IDIGIT
C      LATP=4
C
C-----KEY = 53.  Redefine LAMAT and LBM, the lengths of the portions of
C      WORK(*) and IWORK(*) that are allocated to the sparse matrix
C      storage and the sparse linear equation solver, respectively.
C      LAMAT must be .GE. NVARS+7 and LBM must be positive.
C      If SWITCH=0  option is off; LAMAT=4*NVARS+7
C                                  LBM  =8*MRELAS.
C               =1  option is on.
C      data set =LAMAT
C                LBM
C      LATP=5
C
C-----KEY = 54.  Redefine IPAGEF, the file number where the pages of the
C      sparse data matrix are stored.  IPAGEF must be positive and
C      different from ISAVE (see option 56).
C      If SWITCH=0  option is off; IPAGEF=1.
C               =1  option is on.
C      data set =IPAGEF
C      LATP=4
C
C-----KEY = 55.  Partial results have been computed and stored on unit
C      number ISAVE (see option 56), during a previous run of
C      SPLP( ).  This is a continuation from these partial results.
C      The arrays COSTS(*),BL(*),BU(*),IND(*) do not have to have
C      the same values as they did when the checkpointing occurred.
C      This feature makes it possible for the user to do certain
C      types of parameter studies such as changing costs and varying
C      the constraints of the problem.  This file is rewound both be-
C      fore and after reading the partial results.
C      If SWITCH=0  option is off; start a new problem.
C               =1  option is on; continue from partial results
C                   that are stored in file ISAVE.
C      data set = empty
C      LATP=3
```

```
C
C-----KEY = 56.  Redefine ISAVE, the file number where the partial
C      results are stored (see option 57).  ISAVE must be positive and
C      different from IPAGEF (see option 54).
C      If SWITCH=0  option is off; ISAVE=2.
C              =1  option is on.
C      data set =ISAVE
C      LATP=4
C
C-----KEY = 57.  Save the partial results after maximum number of
C      iterations, MAXITR, or at the optimum.  When this option is on,
C      data essential to continuing the calculation is saved on a file
C      using a Fortran binary write operation.  The data saved includes
C      all the information about the sparse data matrix A.  Also saved
C      is information about the current basis.  Nominally the partial
C      results are saved on Fortran unit 2.  This unit number can be
C      redefined (see option 56).  If the save option is on,
C      this file must be opened (or declared) by the user prior to the
C      call to SPLP( ).  A crude upper bound for the number of words
C      written to this file is 6*nz.  Here nz= number of nonzeros in A.
C      If SWITCH=0  option is off; do not save partial results.
C              =1  option is on; save partial results.
C      data set = empty
C      LATP=3
C
C-----KEY = 58.  Redefine the maximum number of iterations, MAXITR, to
C      be taken before returning to the user.
C      If SWITCH=0  option is off; MAXITR=3*(NVARS+MRELAS).
C              =1  option is on.
C      data set =MAXITR
C      LATP=4
C
C-----KEY = 59.  Provide SPLP( ) with exactly MRELAS indices which
C      comprise a feasible, nonsingular basis.  The basis must define a
C      feasible point: values for x and w such that A*x=w and all the
C      stated bounds on x and w are satisfied.  The basis must also be
C      nonsingular.  The failure of either condition will cause an error
C      message (INFO=-23 or =-24, respectively).  Normally, SPLP( ) uses
C      identity matrix columns which correspond to the components of w.
C      This option would normally not be used when restarting from
C      a previously saved run (KEY=57).
C      In numbering the unknowns,
C      the components of x are numbered (1-NVARS) and the components
C      of w are numbered (NVARS+1)-(NVARS+MRELAS).  A value for an
C      index .LE. 0 or .GT. (NVARS+MRELAS) is an error (INFO=-16).
C      If SWITCH=0  option is off; SPLP( ) chooses the initial basis.
C              =1  option is on; user provides the initial basis.
C      data set =MRELAS indices of basis; order is arbitrary.
C      LATP=MRELAS+3
C
C-----KEY = 60.  Provide the scale factors for the columns of the data
C      matrix A.  Normally, SPLP( ) computes the scale factors as the
C      reciprocals of the max. norm of each column.
C      If SWITCH=0  option is off; SPLP( ) computes the scale factors.
C              =1  option is on; user provides the scale factors.
C      data set =scaling for column J, J=1,NVARS; order is sequential.
```

```
C      LATP=NVARS+3
C
C-----KEY = 61.  Provide a scale factor, COSTSC, for the vector of
C      costs.  Normally, SPLP( ) computes this scale factor to be the
C      reciprocal of the max. norm of the vector costs after the column
C      scaling has been applied.
C      If SWITCH=0  option is off; SPLP( ) computes COSTSC.
C               =1  option is on; user provides costsc.
C      data set =COSTSC
C      LATP=4
C
C-----KEY = 62.  Provide size parameters, ASMALL and ABIG, the smallest
C      and largest magnitudes of nonzero entries in the data matrix A,
C      respectively.  When this option is on, SPLP( ) will check the
C      nonzero entries of A to see if they are in the range of ASMALL and
C      ABIG.  If an entry of A is not within this range, SPLP( ) returns
C      an error message, INFO=-22. Both ASMALL and ABIG must be positive
C      with ASMALL .LE. ABIG.  Otherwise,  an error message is returned,
C      INFO=-17.
C      If SWITCH=0  option is off; no checking of the data matrix is done
C               =1  option is on; checking is done.
C      data set =ASMALL
C                ABIG
C      LATP=5
C
C-----KEY = 63.  Redefine the relative tolerance, TOLLS, used in
C      checking if the residuals are feasible.  Normally,
C      TOLLS=SQRT(EPS), where EPS is the machine precision.
C      If SWITCH=0  option is off; TOLLS=SQRT(EPS).
C               =1  option is on.
C      data set =TOLLS
C      LATP=4
C
C-----KEY = 64.  Use the minimum reduced cost pricing strategy to choose
C      columns to enter the basis.  Normally, SPLP( ) uses the steepest
C      edge pricing strategy which is the best local move.  The steepest
C      edge pricing strategy generally uses fewer iterations than the
C      minimum reduced cost pricing, but each iteration costs more in the
C      number of calculations done.  The steepest edge pricing is
C      considered to be more efficient.  However, this is very problem
C      dependent.  That is why SPLP( ) provides the option of either
C      pricing strategy.
C      If SWITCH=0  option is off; steepest option edge pricing is used.
C               =1  option is on; minimum reduced cost pricing is used.
C      data set =empty
C      LATP=3
C
C-----KEY = 65.  Redefine MXITBR, the number of iterations between
C      recalculating the error in the primal solution.  Normally, MXITBR
C      is set to 10.  The error in the primal solution is used to monitor
C      the error in solving the linear system.  This is an expensive
C      calculation and every tenth iteration is generally often enough.
C      If SWITCH=0  option is off;MXITBR=10.
C               =1  option is on.
C      data set =MXITBR
C      LATP=4
C
```

```
C-----KEY = 66.  Redefine NPP, the number of negative reduced costs
C      (at most) to be found at each iteration of choosing
C      a variable to enter the basis.  Normally NPP is set
C      to NVARS which implies that all of the reduced costs
C      are computed at each such step.  This "partial
C      pricing" may very well increase the total number
C      of iterations required.  However it decreases the
C      number of calculations at each iteration.
C      therefore the effect on overall efficiency is quite
C      problem-dependent.
C
C      if SWITCH=0 option is off; NPP=NVARS
C                =1 option is on.
C      data set =NPP
C      LATP=4
C
C     |-----------------------------|
C     |Example of Option array Usage|
C     |-----------------------------|
C      To illustate the usage of the option array, let us suppose that
C      the user has the following nonstandard requirements:
C
C           a) Wants to change from minimization to maximization problem.
C           b) Wants to limit the number of simplex steps to 100.
C           c) Wants to save the partial results after 100 steps on
C              Fortran unit 2.
C
C      After these 100 steps are completed the user wants to continue the
C      problem (until completed) using the partial results saved on
C      Fortran unit 2.  Here are the entries of the array PRGOPT(*)
C      that accomplish these tasks.  (The definitions of the other
C      required input parameters are not shown.)
C
C      CHANGE TO A MAXIMIZATION PROBLEM; KEY=50.
C      PRGOPT(01)=4
C      PRGOPT(02)=50
C      PRGOPT(03)=1
C
C      LIMIT THE NUMBER OF SIMPLEX STEPS TO 100; KEY=58.
C      PRGOPT(04)=8
C      PRGOPT(05)=58
C      PRGOPT(06)=1
C      PRGOPT(07)=100
C
C      SAVE THE PARTIAL RESULTS, AFTER 100 STEPS, ON FORTRAN
C      UNIT 2; KEY=57.
C      PRGOPT(08)=11
C      PRGOPT(09)=57
C      PRGOPT(10)=1
C
C      NO MORE OPTIONS TO CHANGE.
C      PRGOPT(11)=1
C      The user makes the CALL statement for SPLP( ) at this point.
C      Now to restart, using the partial results after 100 steps, define
C      new values for the array PRGOPT(*):
C
```

```
C      AGAIN INFORM SPLP( ) THAT THIS IS A MAXIMIZATION PROBLEM.
C      PRGOPT(01)=4
C      PRGOPT(02)=50
C      PRGOPT(03)=1
C
C      RESTART, USING SAVED PARTIAL RESULTS; KEY=55.
C      PRGOPT(04)=7
C      PRGOPT(05)=55
C      PRGOPT(06)=1
C
C      NO MORE OPTIONS TO CHANGE.  THE SUBPROGRAM SPLP( ) IS NO LONGER
C      LIMITED TO 100 SIMPLEX STEPS BUT WILL RUN UNTIL COMPLETION OR
C      MAX.=3*(MRELAS+NVARS) ITERATIONS.
C      PRGOPT(07)=1
C      The user now makes a CALL to subprogram SPLP( ) to compute the
C      solution.
C     |------------------------------------------|
C     |End of Usage of SPLP( ) Subprogram Options.|
C     |------------------------------------------|
C
C      |-----------------------------------------------|
C      |List of SPLP( ) Error and Diagnostic Messages.|
C      |-----------------------------------------------|
C       This section may be required to understand the meanings of the
C      =-INFO  that may be returned from SPLP( ).
C
C-----1. There is no set of values for x and w that satisfy A*x=w and
C      the stated bounds.  The problem can be made feasible by ident-
C      ifying components of w that are now infeasible and then rede-
C      signating them as free variables.  Subprogram SPLP( ) only
C      identifies an infeasible problem; it takes no other action to
C      change this condition.  Message:
C      SPLP( ). THE PROBELM APPEARS TO BE INFEASIBLE.
C      ERROR NUMBER =         1
C
C      2. One of the variables in either the vector x or w was con-
C      strained at a bound.  Otherwise the objective function value,
C      (transpose of costs)*x, would not have a finite optimum.
C      Message:
C      SPLP( ). THE PROBLEM APPEARS TO HAVE NO FINITE SOLN.
C      ERROR NUMBER =         2
C
C      3.  Both of the conditions of 1. and 2. above have occurred.
C      Message:
C      SPLP( ). THE PROBLEM APPEARS TO BE INFEASIBLE AND TO
C      HAVE NO FINITE SOLN.
C      ERROR NUMBER =         3
C
C-----4.  The REAL and INTEGER working arrays, WORK(*) and IWORK(*),
C      are not long enough. The values (I1) and (I2) in the message
C      below will give you the minimum length required.  Also redefine
C      LW and LIW, the lengths of these arrays.  Message:
C      SPLP( ). WORK OR IWORK IS NOT LONG ENOUGH. LW MUST BE (I1)
C      AND LIW MUST BE (I2).
C                IN ABOVE MESSAGE, I1=         0
C                IN ABOVE MESSAGE, I2=         0
```

```
C      ERROR NUMBER =          4
C
C-----5. and 6.  These error messages often mean that one or more
C      arguments were left out of the call statement to SPLP( ) or
C      that the values of MRELAS and NVARS have been over-written
C      by garbage.  Messages:
C      SPLP( ). VALUE OF MRELAS MUST BE .GT.O. NOW=(I1).
C                 IN ABOVE MESSAGE, I1=          0
C      ERROR NUMBER =          5
C
C      SPLP( ). VALUE OF NVARS MUST BE .GT.O. NOW=(I1).
C                 IN ABOVE MESSAGE, I1=          0
C      ERROR NUMBER =          6
C
C-----7.,8., and 9.  These error messages can occur as the data
C      matrix is being defined by either USRMAT( ) or the user-supplied sub-
C      program, 'NAME'( ). They would indicate a mistake in the contents of
C      DATTRV(*), the user-written subprogram or that data has been over-written.
C      Messages:
C      SPLP( ). MORE THAN 2*NVARS*MRELAS ITERS. DEFINING OR UPDATING
C      MATRIX DATA.
C      ERROR NUMBER =          7
C
C      SPLP( ). ROW INDEX (I1) OR COLUMN INDEX (I2) IS OUT OF RANGE.
C                 IN ABOVE MESSAGE, I1=          1
C                 IN ABOVE MESSAGE, I2=         12
C      ERROR NUMBER =          8
C
C      SPLP( ). INDICATION FLAG (I1) FOR MATRIX DATA MUST BE
C      EITHER O OR 1.
C                 IN ABOVE MESSAGE, I1=         12
C      ERROR NUMBER =          9
C
C-----10. and 11.  The type of bound (even no bound) and the bounds
C      must be specified for each independent variable. If an independent
C      variable has both an upper and lower bound, the bounds must be
C      consistent.  The lower bound must be .LE. the upper bound.
C      Messages:
C      SPLP( ). INDEPENDENT VARIABLE (I1) IS NOT DEFINED.
C                 IN ABOVE MESSAGE, I1=          1
C      ERROR NUMBER =         10
C
C      SPLP( ).  LOWER BOUND (R1) AND UPPER BOUND (R2) FOR INDEP.
C      VARIABLE (I1) ARE NOT CONSISTENT.
C                 IN ABOVE MESSAGE, I1=          1
C                 IN ABOVE MESSAGE, R1=    O.
C                 IN ABOVE MESSAGE, R2=    -.1000000000E+01
C      ERROR NUMBER =         11
C
C-----12. and 13.  The type of bound (even no bound) and the bounds
C      must be specified for each dependent variable.  If a dependent
C      variable has both an upper and lower bound, the bounds must be
C      consistent. The lower bound must be .LE. the upper bound.
C      Messages:
C      SPLP( ). DEPENDENT VARIABLE (I1) IS NOT DEFINED.
C                 IN ABOVE MESSAGE, I1=          1
```

```
C     ERROR NUMBER =         12
C
C     SPLP( ).  LOWER BOUND (R1) AND UPPER BOUND (R2) FOR DEP.
C      VARIABLE (I1) ARE NOT CONSISTENT.
C               IN ABOVE MESSAGE, I1=         1
C               IN ABOVE MESSAGE, R1=     0.
C               IN ABOVE MESSAGE, R2=     -.1000000000E+01
C     ERROR NUMBER =         13
C
C-----14. - 21.  These error messages can occur when processing the
C     option array, PRGOPT(*), supplied by the user.  They would
C     indicate a mistake in defining PRGOPT(*) or that data has been
C     over-written.  See heading Usage of SPLP( )
C     Subprogram Options, for details on how to define PRGOPT(*).
C     Messages:
C     SPLP( ). THE USER OPTION ARRAY HAS UNDEFINED DATA.
C     ERROR NUMBER =         14
C
C     SPLP( ). OPTION ARRAY PROCESSING IS CYCLING.
C     ERROR NUMBER =         15
C
C     SPLP( ). AN INDEX OF USER-SUPPLIED BASIS IS OUT OF RANGE.
C     ERROR NUMBER =         16
C
C     SPLP( ). SIZE PARAMETERS FOR MATRIX MUST BE SMALLEST AND LARGEST
C     MAGNITUDES OF NONZERO ENTRIES.
C     ERROR NUMBER =         17
C
C     SPLP( ). THE NUMBER OF REVISED SIMPLEX STEPS BETWEEN CHECK-POINTS
C     MUST BE POSITIVE.
C     ERROR NUMBER =         18
C
C     SPLP( ). FILE NUMBERS FOR SAVED DATA AND MATRIX PAGES MUST BE
C     POSITIVE AND NOT EQUAL.
C     ERROR NUMBER =         19
C
C     SPLP( ). USER-DEFINED VALUE OF LAMAT (I1)
C     MUST BE .GE. NVARS+7.
C               IN ABOVE MESSAGE, I1=         1
C     ERROR NUMBER =         20
C
C     SPLP( ). USER-DEFINED VALUE OF LBM MUST BE .GE. 0.
C     ERROR NUMBER =         21
C
C-----22.  The user-option, number 62, to check the size of the matrix
C     data has been used.  An element of the matrix does not lie within
C     the range of ASMALL and ABIG, parameters provided by the user.
C     (See the heading: Usage of SPLP( ) Subprogram Options,
C     for details about this feature.)  Message:
C     SPLP( ). A MATRIX ELEMENT'S SIZE IS OUT OF THE SPECIFIED RANGE.
C     ERROR NUMBER =         22
C
C-----23.  The user has provided an initial basis that is singular.
C     In this case, the user can remedy this problem by letting
C     subprogram SPLP( ) choose its own initial basis.  Message:
C     SPLP( ). A SINGULAR INITIAL BASIS WAS ENCOUNTERED.
```

```
C      ERROR NUMBER =          23
C
C-----24.  The user has provided an initial basis which is infeasible.
C      The x and w values it defines do not satisfy A*x=w and the stated
C      bounds.  In this case, the user can let subprogram SPLP( )
C      choose its own initial basis.  Message:
C      SPLP( ). AN INFEASIBLE INITIAL BASIS WAS ENCOUNTERED.
C      ERROR NUMBER =          24
C
C-----25. Subprogram SPLP( ) has completed the maximum specified number
C      of iterations.  (The nominal maximum number is 3*(MRELAS+NVARS).)
C      The results, necessary to continue on from
C      this point, can be saved on Fortran unit 2 by activating option
C      KEY=57.  If the user anticipates continuing the calculation, then
C      the contents of Fortran unit 2 must be retained intact.  This
C      is not done by subprogram SPLP( ), so the user needs to save unit
C      2 by using the appropriate system commands.  Message:
C      SPLP( ). MAX. ITERS. (I1) TAKEN. UP-TO-DATE RESULTS
C      SAVED ON FILE (I2). IF(I2)=0, NO SAVE.
C                 IN ABOVE MESSAGE, I1=       500
C                 IN ABOVE MESSAGE, I2=         2
C      ERROR NUMBER =          25
C
C-----26.  This error should never happen.  Message:
C      SPLP( ). MOVED TO A SINGULAR POINT. THIS SHOULD NOT HAPPEN.
C      ERROR NUMBER =          26
C
C-----27.  The subprogram LA05A( ), which decomposes the basis matrix,
C      has returned with an error flag (R1).  (See the document,
C      "Fortran subprograms for handling sparse linear programming
C      bases", AERE-R8269, J.K. Reid, Jan.,1976, H.M. Stationery Office,
C      for an explanation of this error.)  Message:
C      SPLP( ). LA05A( ) RETURNED ERROR FLAG (R1) BELOW.
C                 IN ABOVE MESSAGE, R1=    -.5000000000E+01
C      ERROR NUMBER =          27
C
C-----28.  The sparse linear solver package, LA05*( ), requires more
C      space.  The value of LBM must be increased.  See the companion
C      document, Usage of SPLP( ) Subprogram Options, for details on how
C      to increase the value of LBM.  Message:
C      SPLP( ). SHORT ON STORAGE FOR LA05*( ) PACKAGE. USE PRGOPT(*)
C      TO GIVE MORE.
C      ERROR NUMBER =          28
C
C      !----------------------------------------------------!
C      !End of List of SPLP( ) Error and Diagnostic Messages. !
C      !----------------------------------------------------!
C
```

References

1. Dantzig, G. B., Linear Programming and Extensions. Princeton University Press, Princeton, NJ, (1963).

2. Orchard-Hays, W., Advanded Linear Programming Computing Techniques. McGraw-Hill, New York, NY, (1968).

3. Klee, V., Minty, G., "How Good is the Simplex Algorithm?," Inequalities 3, p. 159-175, (1972).

4. Wagner, H., Principles of Operations Research. Prentice-Hall, Inc., Englewood Cliffs, NJ, (1969).

5. Goldfarb, D., Reid, J., "A Practical Steepest-Edge Simplex Algorithm," Math. Programming 12, p. 361-371, (1977).

6. Reid, J. "Fortran Subroutines for Handling Sparse Linear Programming Bases." AERE-R8269, Comp. Sci. and Systems Div., AERE Harwell, Oxfordshire, England.

7. Gay, D., "On Combining the Schemes of Reid and Saunders for Sparse LP Bases," Sparse Matrix Proceedings, 1978. Editors Duff, I. and Stewart, G., SIAM Publications, Philadelphia, PA, (1979).

8. Hopper, M., Harwell Subroutine Library, A Catalogue of Subroutines. AERE-R7477, Suppl. 2, August, (1977).

9. Hanson, R., Wisniewski, J., "A Mathematical Programming Updating Method Using Modified Givens Transformations and Applied to LP Problems," Comm. ACM, Vol. 22, No. 4, (1979), p. 245-251.

10. Hanson, R., Wisniewski, J., "A Single Set of Software that Implements Various Storage Methods for Sparse Matrices," Sandia National Laboratories Rpt. SAND78-0785, (1978).

11. Preckel, P., "Modules for Use with MINOS/AUGMENTEN in Solving Sequences of Mathematical Programs." Tech. Rpt. SOL80-15, Stanford Univ., Nov., (1980).

12. Lawson, C., "SFTRAN3 - Programmer's Reference Manual," JPL Document No. 1846-98. Rev. A., April, (1981).

13. Snyder, W., Hanson, R., "Text Exchange System. A Transportable System for Management and Exchange of Programs and Other Text." Submitted to Trans. Math. Software, Sept., (1981). (Contains TES system as an ACM Algorithm.)

14. Jones, R., "SLATEC Common Mathematical Library Error Handling Package," Sandia National Laboratories Rpt. SAND78-1189, Sept., (1978).

15. Fox, P., Hall, A., Schryer, N., "Framework for a Portable Library," ACM Algorithm 528, Trans. Math. Software. Vol. 4, No. 2, June, (1978), p. 177-188.

16. Lawson, C., Hanson, R., Kincaid, D., Krogh, F., "Basic Linear Algebra Subprograms for Fortran Usage," Trans. Math. Software, Vol. 5, No. 3, Sept., (1979), p. 308-323.

17. Marsten, R. E., "The Design of the XMP Linear Programming Library," Univ. of Arizona, MIS Tech. Rpt. No. 80-2, Tucson, AZ 85721, (1980).

18. Saunders, M. A., "MINOS System Manual," Systems Optim. Lab. Tech. Rpt. SOL77-31, Dept. of Oper. Res., Stanford Univ., Dec., (1977).

19. IBM Corporation, IBM Math. Prog. System Extended, MPSX/370 Ref. Manual, No. SH19-1095-1, (1976).

Distribution

T. J. Aird
IMSL, Inc.
7500 Bellaire Blvd.
Houston, TX 77036

Richard Allen
Dept. of Math. and Statistics
University of New Mexico
Albuquerque, NM 87131

Richard Bartels
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA

Carol Burton
Bendix Field Engineering
P. O. Box 1569
Grand Junction, CO 81501

Billy L. Buzbee
Los Alamos National Labs
P. O. Box 808
Los Alamos, NM 87545

Roy Danchick
The Rand Corp.
1700 Main St.
Santa Monica, CA 90406

George Dantzig
Dept. of Operations Research
Stanford University
Stanford, CA 94305

John E. Dennis, Jr.
Dept. of Math. Sciences
Rice University
Houston, TX 77001

Albert M. Erisman
Boeing Comp. Services
565 Andover Park W. 9C-01
Tukwila, WA 98188

Kirby Fong
Lawrence Livermore Laboratory
P. O. Box 808
Livermore, CA 94550

Brian Ford
NAG, Ltd.
7 Banbury Rd.
Oxford OX2 6NN
ENGLAND

Fred N. Fritsch
Lawrence Livermore Laboratory
P. O. Box 808
Livermore, CA 94550

Pat Gaffney
Oak Ridge National Laboratory
P. O. Box Y
Oak Ridge, TN 37830

David Gay
Bell Laboratories
600 Mountai Rd.
Murray Hill, NJ 07974

Phillip Gill
Stanford University
Dept. of Operations Research
Stanford, CA 94305

Donald Goldfarb
City College of New York
Dept. of Computer Science
New York, NY 10031

Ronald D. Grisell
4 New Hyde Park
Franklin Square, NY 11010

Richard V. Helgason
Southern Methodist University
Dept. of Operations Research
Dallas TX 75275

Bill Kamp
Phillips Petroleum
71C PRC
Bartlesville, OK 74004

Dieter Kraft
D-8031 Oberpfaffenhofen
Inst. f. Dynamik d. Flugsysteme
Wessling, Oberbay
GERMANY

Fred T. Krogh
Jet Propulsion Laboratories
4800 Oak Grove Drive
Pasadena, CA 91103

Charles L. Lawson
Jet Propulsion Laboratories
4800 Park Grove Drive
Pasadena, CA 91103

David W. Lyon
Farmbank Services
12000 E. 47th Ave., Box 39088
Denver, CO 80239

Tom A. Manteuffel, C-3
Los Alamos Nastional Laboratories
P. O. Box 808
Los Alamos, NM 87545

Roy E. Marsten
Dept. of Management Info. Systems
University of Arizona
Tucson, AZ 85718

Mike Minkoff
Argonne National Laboratories
9700 S. Cass Ave.
Argonne, IL 60439

Cleve Moler
Dept. of Computer Science
University of New Mexico
Albuquerque, NM 87131

Jorge J. Moré
Argonne National Laboratories
9700 Cass Ave.
Argonne, IL 60439

Alfred Morris
Naval Surface Weapons Center
DK-74
Dahlgren, VA 22448

Walter Murray
Dept. of Operations Research
Stanford University
Stanford, CA 94305

K. G. Murty
Dept. of Industrial Engr.
University of Michigan
Ann Arbor, MI 48109

Ron Nickel
University of North Carolina
Smith Bldg. 128a
Chapel Hill, NC 27514

Dan O'Reilly
Data Resources Inc.
29 Hartwell Ave.
Lexington, MA 02173

Allen Pope
NOAA/NOS/NGS/OA/C-12
6001 Executive Blvd.
Rockville, MD 20854

John Reid
AERE Harwell
Comp. Sci. & Systems Div.
Oxford, OX11 ORA
ENGLAND

Report Section
Dept. of Computer Science
Royal Inst. of Technology
Stockholm
SWEDEN

Raymond C. Roan
Dept. of Mathematics
University of Idaho
Moscow, ID 83843

Gordon Sande
Stat. of Canada
Business Survey Div. Tunney's Pasture
Ottowa, Ontario K1AOT6
CANADA

Mike Saunders
Dept. of Operations Research
Stanford University
Stanford, CA 94305

Ron Schroder
Dept. of Math. and Stat.
University of New Mexico,
Albuquerque, NM 87131

Don Schick
Dept. of Indust. Engineering
Ohio University
Athens, OH 45701

John D. Schmitz
Strategic Information
8C Henshaw St.
Woburn, MA 01801

Jerry Simon
Exxon Corp.
CCS, Bldg. F102, Rm. D132
Florham Park, NJ 07932

Aarne Sipila
Computer Center
Helsinki University of Technology
Otakaara 1 SF02150
FINLAND

Peter J. Slater
Dept. of Mathematics
University of Alabama
Huntsville, AL 35899

C. A. Steele, Jr.
Computer Science Dept.
University of Lowell
Lowell, MA 01854

Jim Sutton
Hercules Corp.
Wilmington, Deleware

Richard A. Tapia
Dept. of Math. Sciences
Rice University
Houston, TX 77001

R. P. Tewarson
Dept. of Appl. Math. & Stat.
SUNY at Stony Brook
Long Island, NY 11794

Steve Thomson
Computing Center
University of Kentucky
Lexington, KY 40506

R. M. Thrall
Dept. of Math. Sciences
Rice University
Houston, TX 77001

Homer Walker
Dept. of Math. & Stat.
University of New Mexico
Albuquerque, NM 87131

Robert Ward
Oak Ridge Natl. Laboratories
P. O. Box Y
Oak Ridge, TN 37830

Rick Whitaker
McDonnell Douglas Automation
Box 516
St. Louis, MO 63156

Jimmie D. Woods, Superintendent
U.S. Coast Guard Academy
New London, CT 06320

Margaret Wright
Dept. of Operations Research
Stanford University
Stanford, CA 94305

1111 S. D. Stearns
1222 C. F. Diegert
1223 K. V. Kiegert
1223 P. G. Kaestner
1223 R. R. Prairie
2113 J. A. Wisniewski
2614 A. R. Iacoletti
2614 R. E. Jones
2644 D. C. Chiglia
2646 M. R. Scott
2646 E. A. Aronson
2646 K. H. Haskell
2646 W. A. Vandevender
4231 F. Biggs
4231 P. J. McDaniel
4413 D. R. Strip
4416 L. M. Grady
4416 L. D. Chapman
4737 L. C. Bartel
5000 J. K. Galt
5600 D. B. Shuster
Attn: A. A. Lieber, 5610
M. M. Newsom, 5620
R. C. Maydew, 5630
D. J. Rigali, 5650
5640 G. J. Simmons
5640 M. J. Norris
5641 R. J. Thompson
5642 L. F. Shampine
5642 R. J. Hanson (10)
5642 K. L. Hiebert (10)
5642 B. L. Hulme
8214 M. A. Pound
8331 R. J. Kee
8332 R. E. Huddleston
8332 J. F. Lathrop
3141 L. J. Erickson (5)
3151 W. L. Garner (3)
3154-3 C. H. Dalin (25)
(For DOE/TIC)

Org.	Bldg.	Name	Rec'd by	Org.	Bldg.	Name	Rec'd by

Sandia National Laboratories