IBM

Application Program

H20-0345-2

1130 Linear Programming– Mathematical Optimization Subroutine System (1130 LP-MOSS) (1130-CO-16X) Program Reference Manual

This manual contains all the information needed to use the LP-MOSS/1130 Linear Programming System. Chapter 1 is written in tutorial form to present concepts to new users. The rest of the manual describes the system procedures, data formats, and operating instructions.

Kristofer Sweger

Third Edition

This edition, H20-0345-2, is a major revision obsoleting H20-0345-1 and H20-0345-0.

Significant changes or additions to the specifications contained in this publication will be reported in subsequent revisions or Technical Newsletters.

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to:
IBM, Technical Publications Department, 112 East Post Road, White Plains, N. Y. 10601

CONTENTS

READER'S GUIDE

This manual is divided into tutorial, procedure, format, and operational sections. An appendix is provided to serve as a handy reference for system usage. The tutorial section, Chapter 1, presents system concepts and provides a background for the usage of 1130 LPS. It defines a simple blending application, shows the mathematical development, and illustrates data preparation and setup for the computer run. The output reports are also shown and described. Chapter 1 continues with a natural extension of the original problem to illustrate data maintenance features of the system. It concludes with a discussion of the usage of starting solutions, conditional control, and data generation.

Chapter 2 describes and illustrates the functions carried out by the various LP procedures — that is, the functions required to solve a linear programming problem and to provide the necessary data maintenance and control functions for LP application data processing.

Chapter 3 describes the LPS input data and unit record formats.

Chapter 4 contains the operating procedures and system and error messages.

The Appendix contains a summary of the formulation rules and the unit record formats and other data, and serves as a handy reference for system usage.

COMPARISON WITH OTHER SYSTEMS*

General

The 1130 LPS contains a row variable feature in place of the more common and less flexible slack and/or artificial variable feature. The row variable feature has been incorporated to simplify problem formulation and to provide a more efficient and natural problem representation. The following are some of the advantages of this feature as expressed in slack variable terms:

1. Rows can be added together, independently of row limits. This feature can be used to reduce substantially the number of coefficients in the problem data for many applications.
2. Slack costs are allowed.
3. Slacks can replace transfer variables.

*While the experienced user of other LP systems will find this comparison useful, the new user should *omit it* and proceed to Chapter 1.

The row variables of 1130 LPS which do not contain other coefficients (for row summing, slack costs, etc.) appear similar to negative slacks, except that the solution activity of these variables is constrained to be positive, negative, free, according to the problem limits expressed by bounds, RHS, RANGES, etc.

The 1130 LPS row variables are shown on the left-hand side of an equation to highlight the difference in relationship between the LPS formulation and traditional LP usage of right-hand sides.

The LPS row variables conform to the common mathematical usage of defining a variable in relationship to other variables; hence, in LPS notation, Yi=F(X, Y), where F is a linear function of variables X and Y excluding Yi. In 1130 LPS notation, a variable is a row variable if it is defined on the left-hand side of an equation. A variable is a column variable if it is not defined on the left-hand side of any equation.

All row and column variables are assumed to be restricted to nonnegative solution activities unless otherwise specified.

All row and column variables may have type designations — for example, any variable may have a fixed solution activity, or may be restricted to nonpositive solution activities, or may be designated as a free variable, or may be the objective variable.

Limits on the solution activities of 1130 LPS variables are normally specified by an upper bound and/or by a lower bound. An explicit, user-specified lower bound is required only if the lower limit on the solution activity is different from the standard (the standard is zero unless a type has been specified for a variable). Similarly, an explicit upper bound is required to specify an upper limit on solution activity, unless a type is specified for a variable.

The traditional LP ROW.ID or ROWS file defines the type or relationship of a row or linear function F(X) to constant A or constants A, B, specified by a RHS entry and, for MPS/360, by a RANGE entry. The traditional LP slack then provides a conversion of these various types to equations:

Row Type	Relationship to RHS	Equation
Free	F(X)-A	F(X)+FREE=A
Equation	F(X)=A	F(X)+ARTIFICIAL=A
Greater than	F(X) ≥ A	F(X)-SLACK=A
Less than	F(X) ≤ A	F(X)+SLACK=A
Range	A ≤ F(X) ≤ B	F(X)-SLACK=A O ≤ SLACK ≤ B-A

The 1130 LPS does not use the traditional slack variable technique for the conversion of row types to equations. The 1130 LPS will accept the MPS/360 input data within the limitations described at the end of Chapter 3, under "Compatibility with MPS/360". The ROWS, RHS, and RANGE information is translated internally as required to specify upper bounds and lower bounds on the variables.

MPS/360

The 1130/LPS is conceptually more similar to this system than to other systems. The obvious advantage is the simple transition from the 1130/LPS to MPS/360. Additional information about MPS/360 compatibility is contained in Chapter 3.

LP/1620

The 1130/LPS provides many features not included in the LP/1620 system. Certain functions operate in a different manner from the LP/1620 system to provide greater flexibility. A careful reading, therefore, is highly recommended.

Input. Both systems provide for unit record input and disk data maintenance. The data formats are different on the two systems. Data conversion can be made by using MPS/360 TRNSLATE to convert LP/1620 problem decks to MPS/360 problem decks which can be used as input data for 1130 LPS.

CAUTION: The 1130 LPS requires unique row and column names. Referenced data (feature of LP/1620) is not required, since multiple bound sets and a complete REVISE are available with 1130 LPS.

Revise. The 1130 LPS REVISE allows new elements, rows, columns, bound sets, right-hand sides, and ranges to be added, and allows the removal of variables. This additional flexibility imposes a far greater need for data checking than was required for LP/1620. The revision data should be verified, checked, and listed before using 1130 LPS REVISE.

If there are a substantial number of revisions, or if there are inadequate checking facilities, the 1130 LPS user is recommended to use MERGE, instead of, or in conjunction, with REVISE. This topic is further discussed in Chapter 1, under "Data Maintenance".

Basis is optional INPUT on LP/1620. The user can optionally INSERT a basis (advanced starting solution) with LPS 1130.

Optimize. All data (coefficients, bounds, right-hand sides, etc.) required in the optimization of a problem must be in the problem data. The current problem data must be recalled and the processing parameters must be set before optimization. When a start from a starting solution is desired, it must be RESTOREd before optimization. The control sequence is as follows:

```
*********************************************************************
MOVE
    DATA            LPSAMPLE
    MINIMIZE        COST
    BOUNDS          ALLOY1
ENDATA
RESTORE
    DATA            LPSAMPLE
OPTIMIZE
*********************************************************************
```

Exhibit A. Re-optimization from saved solution

We shall present a typical (though simplified) production problem as a basis for the development of an LP model that will illustrate the methods and capabilities of the 1130 Linear Programming System (LPS).

An aluminum alloy smelter wishes to produce 2000 lbs. of a particular alloy at minimum cost. The alloy must, however, meet certain chemical constraints. The smelter has available various scrap materials and some industrially pure aluminum and silicon. Five scrap materials, of known chemical composition and in specific quantities, are available for the blend, while the pure aluminum and silicon will be purchased as needed. The cost of each of these seven ingredients is known. We shall assume that chemical substances are neither lost from nor added to the raw materials during the process.

Tabular representations of all the relevant information are provided in Figures 1-4. Figure 1 provides the chemical specifications of the desired alloy; Figure 2, scrap metal inventories; Figure 3, a chemical analysis of the available scrap metals; and Figure 4, the price per pound of each of the raw materials.

The problem is to determine which combination of raw materials will produce the specified alloy at the least cost. The LPS is designed to solve a set of simultaneous equations so that the value of one of the variables in the equation system, which we shall call the objective variable, is either maximized or minimized (depending on whether we seek the cost or the profit associated with some process). In this problem we shall minimize the objective variable COST. The other variables in the system of equations — the weight of each raw material used, the weight of each chemical element in the alloy, etc. — will be assigned values that result in the lowest value for the variable COST which meets the specifications of the problem.

In order to formulate a linear programming problem for solution by the LPS, three varieties of data must be available:

1. A system of equations defining the relationships among all the problem variables.

2. A set of bounds defining the limits on the values of the problem variables (such as inventory limitations or chemical specification constraints).

CHEMICAL	ALLOY CONTENT IN LBS MAXIMUM	MINIMUM
IRON (FE)	60	
COPPER (CU)	100	
MANGANESE (MN)	40	
MAGNESIUM (MG)	30	
ALUMINUM(AL)		1500
SILICON (SI)	300	250
FE+CU (BASE)	120	

Figure 1. Chemical specifications for 2000 pounds of alloy

BIN NUMBER	INVENTORY (LBS)
1 (BIN.1)	200
2 (BIN.2)	750
3 (BIN.3)	800
4 (BIN.4)	700
5 (BIN.5)	1500

Figure 2. Scrap metal inventories

CHEMICAL	SCRAP METALS						
	BIN.1	BIN.2	BIN.3	BIN.4	BIN.5	ALUMINUM	SILICON
FE	.15	.04	.02	.04	.02	.01	.03
CU	.03	.05	.08	.02	.06	.01	
MN	.02	.04	.01	.02	.02		
MG	.02	.03			.01		
AL	.70	.75	.80	.75	.80	.97	
SI	.02	.06	.08	.12	.02	.01	.97

Figure 3. Chemical analysis of the raw materials in pounds per pound of alloy

RAW MATERIAL	COST PER LB
BIN•1	•03
BIN•2	•08
BIN•3	•17
BIN•4	•12
BIN•5	•15
ALUMINUM	•21
SILICON	•38

Figure 4. Raw material costs

3. The nature and name of the objective variable (that is, the variable whose value will be either minimized or maximized).

Thus, for a sample problem designed to discover what blend of raw materials will produce a specified aluminum alloy at least cost, the LPS will require:

1. A system of equations establishing the relationships among all the problem variables in terms of the cost, chemical composition, and weight of alloy desired.
2. A set of bounds establishing the inventory availability of each raw material, the limits on the weight of each chemical substance in the desired alloy (that is, the specifications), and the total weight of alloy to be produced.
3. The name of the objective variable — in this case, COST, which, in this sample problem, will be minimized.

The LPS will find the minimum value for the objective variable (COST) which is consistent with the chemical specifications of the desired alloy and the available scrap inventories. The solution will (1) indicate the cost of the desired alloy and the weight of each of the materials required, (2) produce a variety of reports that may alert the producer to relationships profoundly affecting the total cost of the end metal, and (3) suggest methods for reducing the cost.

THE PROBLEM EQUATIONS

The system of equations that must be input in order to solve this problem will contain the following variables:

1. The weight of each raw material required to produce the alloy
2. The weight of each of the chemical contents of the desired alloy
3. The total cost of the raw materials required to produce the alloy (this is the objective variable and will be minimized)
4. The weight of the desired alloy (this will later be limited to a single value — 2000 lbs.)

Since we have assumed that no chemical substances will be lost or added during the smelting process, we can easily formulate an equation that provides the total weight of the desired alloy:

$$\begin{aligned}\text{WEIGHT} = {} & 1.0\ \text{BIN.1} + 1.0\ \text{BIN.2} + 1.0\ \text{BIN.3} \\ & + 1.0\ \text{BIN.4} + 1.0\ \text{BIN.5} \\ & + 1.0\ \text{ALUMINUM} + 1.0\ \text{SILICON}\end{aligned}$$

where WEIGHT is the weight of the desired alloy, and BIN.1, BIN.2, etc., are variables representing the weights of the raw materials used.

Similarly, we can formulate equations for the chemical content of the desired alloy. As indicated in Figure 3, the scrap in BIN.1 contains .02 magnesium by weight. Hence, .02 BIN.1 expresses the weight of magnesium from that source which will appear in the desired alloy. It follows that the total weight of magnesium in the desired alloy may be expressed as:

$$\text{MG} = .02\ \text{BIN.1} + .03\ \text{BIN.2} + .01\ \text{BIN.5}$$

Again referring to Figure 3, we can formulate the remainder of the chemical content equations in the same fashion:

$$\begin{aligned}\text{FE} = {} & .15\ \text{BIN.1} + .04\ \text{BIN.2} + .02\ \text{BIN.3} \\ & + .04\ \text{BIN.4} + .02\ \text{BIN.5} \\ & + .01\ \text{ALUMINUM} + .03\ \text{SILICON}\end{aligned}$$

$$\begin{aligned}\text{CU} = {} & .03\ \text{BIN.1} + .05\ \text{BIN.2} + .08\ \text{BIN.3} \\ & + .02\ \text{BIN.4} + .06\ \text{BIN.5} \\ & + .01\ \text{ALUMINUM}\end{aligned}$$

$$\begin{aligned}\text{MN} = {} & .02\ \text{BIN.1} + .04\ \text{BIN.2} + .01\ \text{BIN.3} \\ & + .02\ \text{BIN.4} + .02\ \text{BIN.5}\end{aligned}$$

$$\begin{aligned}\text{AL} = {} & .70\ \text{BIN.1} + .75\ \text{BIN.2} + .80\ \text{BIN.3} \\ & + .75\ \text{BIN.4} + .80\ \text{BIN.5} \\ & + .97\ \text{ALUMINUM}\end{aligned}$$

$$\begin{aligned}\text{SI} = {} & .02\ \text{BIN.1} + .06\ \text{BIN.2} + .08\ \text{BIN.3} \\ & + .12\ \text{BIN.4} + .02\ \text{BIN.5} \\ & + .01\ \text{ALUMINUM} + .97\ \text{SILICON}\end{aligned}$$

One more equation relevant to the chemical content of the desired alloy must be formulated which reflects the specification (see Figure 1) regarding the maximum total weight of iron and copper in the alloy:

$$\text{BASE} = 1.0\ \text{FE} + 1.0\ \text{CU}$$

A final equation is required which defines the objective variable for this problem. The price of each ingredient per pound, multiplied by the quantity in pounds of each ingredient used, will provide the cost:

COST = .03 BIN.1 + .08 BIN.2 + .17 BIN.3
+ .12 BIN.4 + .15 BIN.5
+ .21 ALUMINUM + .38 SILICON

Gathering all these equations into a system, then, provides the framework of a model for the aluminum alloy problem:

WEIGHT = 1.0 BIN.1 + 1.0 BIN.2 + 1.0 BIN.3
+ 1.0 BIN.4 + 1.0 BIN.5 (1)
+ 1.0 ALUMINUM + 1.0 SILICON

COST = .03 BIN.1 + .08 BIN.2 + .17 BIN.3
+ .12 BIN.4 + .15 BIN.5 (2)
+ .21 ALUMINUM + .38 SILICON

FE = .15 BIN.1 + .04 BIN.2 + .02 BIN.3
+ .04 BIN.4 + .02 BIN.5 (3)
+ .01 ALUMINUM + .03 SILICON

CU = .03 BIN.1 + .05 BIN.2 + .08 BIN.3
+ .02 BIN.4 + .06 BIN.5 (4)
+ .01 ALUMINUM

MN = .02 BIN.1 + .04 BIN.2 + .01 BIN.3
+ .02 BIN.4 + .02 BIN.5 (5)

MG = .02 BIN.1 + .03 BIN.2
+ .01 BIN.5 (6)

AL = .70 BIN.1 + .75 BIN.2 + .80 BIN.3
+ .75 BIN.4 + .80 BIN.5 (7)
+ .97 ALUMINUM

SI = .02 BIN.1 + .06 BIN.2 + .08 BIN.3
+ .12 BIN.4 + .02 BIN.5 (8)
+ .01 ALUMINUM + .97 SILICON

BASE = 1.0 FE + 1.0 CU (9)

BOUNDS

The set of equations we have formulated is not a complete model for the alloy smelting problem, because it does not provide a set of bounds restricting the solution within the constraints imposed by inventory limitations and chemical specifications. The bounds, of course, are determined by the availability of scrap inventory (Figure 2) and the chemical specifications for the desired alloy (Figure 1). The final weight (WEIGHT) is made equal to 2000 lbs., and COST, which will be minimized by the LPS, may take any value.

The inventories given in Figure 2 provide the upper bounds for the raw materials used; the lower bounds are 0 (since it would make no sense to use a negative weight of scrap). Similarly, the specifications given in Figure 1 provide bounds for the chemical contents of the alloy. There must, for instance, be between 250 and 300 lbs. of silicon in the alloy, no less than 1500 lbs. of aluminum, no more than 30 lbs. of magnesium, etc. Figure 5 provides a tabular representation of the set of bounds required in this problem.

The provision of a set of bounds completes the mathematical model of our problem. The 1130 LPS, operating upon the data here formulated, will produce a solution that satisfies all the problem bounds and print or type a solution report containing:

1. The weight of each raw material required to make the alloy at the least possible cost
2. The weight of alloy made
3. The cost of the alloy made
4. The chemical contents of the alloy made

DATA PREPARATION

A mathematical model of a linear programming problem is, for our purposes, not useful until it is converted into input for the LPS. The principal purpose of this manual is to acquaint the user with the vocabulary and syntax of the 1130 LPS so that he may convert the mathematical models of LP problems into input data and control the manipulations of

	VARIABLE	UPPER BOUND	LOWER BOUND
RAW MATERIALS	BIN.1	200	0
	BIN.2	750	0
	BIN.3	800	0
	BIN.4	700	0
	BIN.5	1500	0
	ALUMINUM	NONE	0
	SILICON	NONE	0
ALLOY WEIGHT	WEIGHT	2000	2000
COST	COST	NONE	NONE
ALLOY CHEMICAL CONTENT	FE	60	0
	CU	100	0
	MN	40	0
	MG	30	0
	AL	NONE	1500
	SI	300	250
	BASE	120	0

Figure 5. Bounds on the sample problem variables

that data in order to obtain the LP solution and the various additional reports that the 1130 LPS provides. Before discussing the program control features provided by 1130 LPS, let us consider the methods for inputting the problem data contained in the mathematical model of our LP aluminum alloy sample problem.

PROBLEM DATA

Mathematical models of LP problems formulated for solution by the 1130 LPS must consist of equations of the form:

$$RV = CV1 + CV2 + CV3 \ldots + CVn$$

We shall call those variables that appear on the left-hand side of such problem equations "row variables", and any variables which do not appear on the left-hand side of any equation, "column variables". Note that row variables may sometimes appear on the right-hand side of some problem equations, but column variables, by definition, never appear on the left-hand side of any problem equation. Further, problem equations must be formulated according to the following rules:

1. Only one variable may appear on the left-hand side of a problem equation (that is, equations of the form RV1 + RV2 = CV1 + CV2 may not be input to the 1130 LPS).
2. The same variable may not appear on the left-hand side of more than one equation.
3. The coefficient of the variable on the left-hand side of a problem equation must be 1.0 (that is, equations of the form xRV = CV1 + CV2 ... + CVm, where x is a number other than 1, may not be input to the 1130 LPS).

Given the preceding nomenclature and formulation rules, consider now the methods for preparing a problem data deck designed to input the problem equations and bounds for the sample alloy problem we have been considering.

A problem data deck may contain records prepared in three different formats depending on the specific purpose of the record:

1. Indicator record format (used for a variety of functions, such as naming problem files in which data is stored and signaling the end of a data deck)
2. Element record format (used to input the problem equations and bounds)
3. Comment record format (used to document the problem for user's convenience — records in comment format in the data deck are ignored by 1130 LPS)

The format for input records is discussed in detail in Chapter 3 of this manual. Since several different problems may be simultaneously stored on the disk, each problem data deck should be introduced by a NAME indicator record naming the problem file into which the problem data will be stored. Such problem file names should consist of from 1 to 8 alphameric characters. It is recommended that the name be left-justified and that it not contain embedded blanks. If no name is provided by the user, the 1130 LPS will assign the problem file a name consisting of eight blanks. Figure 6 provides the format for indicator records and illustrates the NAME indicator record for the problem LPSAMPLE, which we are here formulating. Every input data deck must end with an ENDATA indicator record signaling the end of the data deck.

Following the NAME indicator record, will be a number of element records prepared according to the format defined in Figure 7. Such records are required to input the elements of the equations we formulated for our sample problem.

The first equation defined the weight of the desired alloy:

WEIGHT = 1.0 BIN.1 + 1.0 BIN.2 + 1.0 BIN.3
+ 1.0 BIN.4 + 1.0 BIN.5
+ 1.0 ALUMINUM + 1.0 SILICON

In this equation, WEIGHT is the name of the row variable. To input the elements of this equation, we must prepare an element record for each variable on the right-hand side of the equation. Each record will include the name of a variable on the right-hand side of the equation in the first name field (cc 5-12), the name of the row variable (WEIGHT) in the second name field (cc 15-22), and

Positions	Field
1-14	Indicator name
15-22	Name

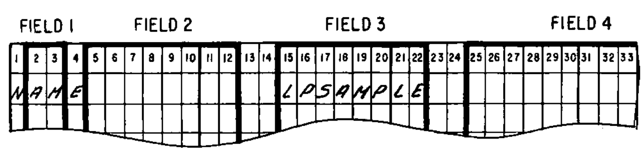


Figure 6. Indicator record format and NAME indicator for problem file LPSAMPLE

```
*************************************************************************************
*  RECORD     *    CONTENTS- COMMENTS                                               *
* POSITIONS   *                                                                     *
*************************************************************************************
*************************************************************************************
*    1-4      * TYPE FIELD- MUST BE BLANK FOR EQUATION ELEMENT RECORDS              *
*************************************************************************************
*    5-12     * FIRST NAME FIELD- THE NAME OF A VARIABLE ON THE RIGHT HAND          *
*             *                   SIDE OF AN EQUATION                               *
*************************************************************************************
*   13-14     * MUST BE BLANK                                                       *
*************************************************************************************
*   15-22     * SECOND NAME FIELD- THE ROW VARIABLE FOR AN EQUATION                 *
*************************************************************************************
*   23-24     * MUST BE BLANK                                                       *
*************************************************************************************
*   25-36     * VALUE FIELD- COEFFICIENT OF THE VARIABLE IN THE FIRST NAME          *
*             *              FIELD IN THE EQUATION CORRESPONDING TO THE             *
*             *              VARIABLE IN THE SECOND NAME FIELD                      *
*************************************************************************************
```

Figure 7. Equation element record format

the coefficient of the variable on the right-hand side in the first value field (cc 25-36). The worksheet for the records that input the WEIGHT equation would appear as in Figure 8. In this manual, worksheets are used instead of the Mathematical Programming Input Form (X20-1761), which is recommended for input data preparation.

Similarly, the worksheet for the records which input the equation elements for the cost equation:

COST = .03 BIN.1 + .08 BIN.2 + .17 BIN.3
+ .12 BIN.4 + .15 BIN.5
+ .21 ALUMINUM + .38 SILICON

would appear as in Figure 9.

The equations representing the chemical composition may be input in precisely the same fashion. Figure 10 provides a printout of a portion of the input deck, beginning with the indicator record naming the problem LPSAMPLE, and including all the nonzero elements.

It is now necessary to input the set of bounds indicating the weight of the desired alloy, the inventory limitations, and the chemical specifications. To this end the user must define a name for the set of bounds. This name (ALLOY1 here) may be thought of as a special sort of name and it will appear in the first name field on each of the element records that input the set of bounds for the problem at hand. Note that several bound sets may be input for one problem through the simple agency of defining a different name for each set. Consequently the same LP equations may be solved with a variety

FIELD 1	FIELD 2	FIELD 3	FIELD 4
	BIN.1	WEIGHT	1.0
	BIN.2	WEIGHT	1.0
	BIN.3	WEIGHT	1.0
	BIN.4	WEIGHT	1.0
	BIN.5	WEIGHT	1.0
	ALUMINUM	WEIGHT	1.0
	SILICØN	WEIGHT	1.0

Figure 8. Worksheet for weight equation

FIELD 1	FIELD 2	FIELD 3	FIELD 4
	BIN.1	CØST	0.03
	BIN.2	CØST	0.08
	BIN.3	CØST	0.17
	BIN.4	CØST	0.12
	BIN.5	CØST	0.15
	ALUMINUM	CØST	0.21
	SILICØN	CØST	0.38

Figure 9. Worksheet for cost equation

```
*COST EQUATION
      BIN.1      COST          0.03
      BIN.2      COST          0.08
      BIN.3      COST          0.17
      BIN.4      COST          0.12
      BIN.5      COST          0.15
      ALUMINUM   COST          0.21
      SILICON    COST          0.38
* WEIGHT EQUN
      BIN.1      WEIGHT        1.00
      BIN.2      WEIGHT        1.00
      BIN.3      WEIGHT        1.000
      BIN.4      WEIGHT        1.000
      BIN.5      WEIGHT        1.000
      ALUMINUM   WEIGHT        1.000
      SILICON    WEIGHT        1.000
* CHEMICAL CONTENT EQUNS
      BIN.1      FE            0.150000
      BIN.2      FE            0.040000
      BIN.3      FE            0.020000
      BIN.4      FE            0.040000
      BIN.5      FE            0.020000
      ALUMINUM   FE            0.01
      SILICON    FE            0.03
      BIN.1      CU            0.030000
      BIN.2      CU            0.050000
      BIN.3      CU            0.080000
      BIN.4      CU            0.020000
      BIN.5      CU            0.060000
      ALUMINUM   CU            0.01
      BIN.1      MN            0.020000
      BIN.2      MN            0.040000
      BIN.3      MN            0.010000
      BIN.4      MN            0.020000
      BIN.5      MN            0.020000
      BIN.1      MG            0.020000
      BIN.2      MG            0.030000
      BIN.5      MG            0.010000
      BIN.1      AL            0.700000
      BIN.2      AL            0.750000
      BIN.3      AL            0.800000
      BIN.4      AL            0.750000
      BIN.5      AL            0.800000
      ALUMINUM   AL            0.97
      BIN.1      SI            0.020000
      BIN.2      SI            0.060000
      BIN.3      SI            0.080000
      BIN.4      SI            0.120000
      BIN.5      SI            0.020000
      ALUMINUM   SI            0.01
      SILICON    SI            0.97
      FE         BASE          1.0
      CU         BASE          1.0
```

Figure 10. Portion of input deck

of different bounds by using (later during the solution process) a new single control record to name a different bound set to be used in the computation.

The element records that input bounds differ from equation element records in that bound element records always indicate what sort of bound is being defined. For our purposes, at the moment, we need to define four principal bound types: upper, lower, fixed, and free. For each of these types, LPS provides a specific symbol which must appear in cc 2 and 3 of the bound element data record. Consider the bound element record in Figure 11. The UB in cc 2-3 signifies that the record will input an upper bound. ALLOY1 in the first name field names the bound set, and BIN.1 in the second name field names the variable to be bounded. The figure 200.0 in the coefficient field indicates that the upper bound on the material symbolized by BIN.1 is 200 lbs. The lower bound is automatically set at zero by the 1130 LPS.

The record in Figure 12 illustrates an element data card defining a lower bound (LB) on the quantity of ALUMINUM which must appear in the solution employing bound set ALLOY1.

To the obvious designations UB and LB, signifying upper bound and lower bound, we must add the designations FX and FR. FX signifies a fixed level, and may be thought of as an equal sign. The record in Figure 13 indicates that the total weight of the desired alloy must be equal to 2000 lbs. Similarly, the FR on the record in Figure 14 signifies that COST is free, unlimited (that is, COST is not really bounded at all). Since, in this problem, we wish the objective variable minimized, the LPS will compute

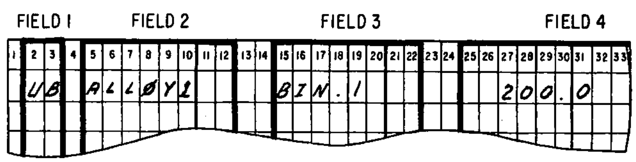


Figure 11. Bound element record — upper bound

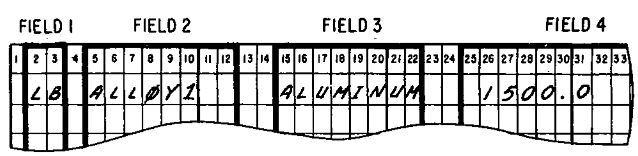


Figure 12. Bound element record — lower bound

the lowest possible value for COST consistent with the other bounds defined in the bound set ALLOY1. Were the problem designed to maximize profit, the variable representing profit would be maximized in the solution. Figure 15, then, provides the worksheet for the deck of bound element records required for our problem.

FIELD 1	FIELD 2	FIELD 3	FIELD 4
FX	ALLØY1	WEIGHT	2000.0

Figure 13. Bound element record — fixed bounds

FIELD 1	FIELD 2	FIELD 3	FIELD 4
FR	ALLØY1	CØST	

Figure 14. Bound element record — free

FIELD 1	FIELD 2	FIELD 3	FIELD 4
FR	ALLØY1	CØST	
FX	ALLØY1	WEIGHT	2000.0
UB	ALLØY1	BIN.1	200.0
UB	ALLØY1	BIN.2	750.0
UB	ALLØY1	BIN.3	800.0
UB	ALLØY1	BIN.4	700.0
UB	ALLØY1	BIN.5	1500.0
UB	ALLØY1	FE	60.0
UB	ALLØY1	CU	100.0
UB	ALLØY1	MN	40.0
UB	ALLØY1	MG	30.0
LB	ALLØY1	AL	1500.0
LB	ALLØY1	SI	250.0
UB	ALLØY1	SI	300.0
UB	ALLØY1	BASE	120.0

Figure 15. Worksheet for bounds set ALLOY1, problem file LPSAMPLE

Having prepared records which name the problem file (LPSAMPLE), describe the equation elements, and name and describe the set of bounds, we need only one further record to complete the input data deck: a second indicator to indicate that the data deck is now concluded. A standard ENDATA record beginning in cc 1, as in Figure 16, must be the final record in an input data deck.

FIELD 1	FIELD 2	FIELD 3	FIELD 4
ENDATA			

Figure 16. End-of-data indicator record

PROGRAM CONTROL

We have considered the format of records required to input the problem data and constructed an input data deck. We need, now, to consider a number of 1130 LPS procedure control records that provide means for actually inputting and processing the problem data. The format for a procedure control record is quite simple. Punch the name of the procedure beginning in cc 1 of the record. The name of the procedure that introduces a data deck into the computer storage is INPUT, and Figure 17 illustrates an INPUT procedure control record. Such a record, followed by the problem data deck, results in the storage of the data in a problem file on the disk.

FIELD 1	FIELD 2	FIELD 3	FIELD 4
INPUT			

Figure 17. Input procedure call record

Assume that several problem files, in addition to LPSAMPLE, are stored on the disk. We wish, now, to produce a solution for the problem LPSAMPLE. To accomplish this, the parameters for the problem must be assigned; that is, the data contained in the problem file LPSAMPLE must be retrieved, the bounds set to be used (ALLOY1 in this case) must be named, and the objective variable to be optimized (minimize COST in this case) must be named before a solution can be undertaken. The 1130 LPS procedure that sets the parameters for each computation is named MOVE, and a procedure control record punched MOVE beginning in cc 1 will call the MOVE procedure. But MOVE must be supplied with its own data records indicating which problem file, which bounds set, and which objective variable should be used in the computation. The records which provide that information to the MOVE procedure are called "data control records". These data control records are similar in format to input data element records, but differ in that they supply information to the 1130 LPS procedures rather than supply problem data. The format for data control records is given in Figure 18, and Figure 19 illustrates the data control records that would follow the procedure control record MOVE in order to set parameters for the solution of problem LPSAMPLE. The first of these, a DATA record, names the problem file containing the data for computation; the second record establishes that COST is the objective variable and that it should be MINIMIZED; the third names the set of BOUNDS to be employed in the current computation. An ENDATA indicator is necessary to signal the end of the data control records relevant to the MOVE procedure control record.

Once the problem data has been input, and the parameters set, the single procedure control record LPSOLUTION will result in a solution to the problem and produce an output report detailing that solution. Figure 20 provides a schematic illustration of the entire deck required for the solution of the aluminum blend problem.

```
****************************************
*  RECORD      *    CONTENTS           *
* POSITIONS    *                       *
****************************************
****************************************
*    1-4       * MUST BE BLANK         *
*    5-12      * NAME 1                *
*   13-14      * MUST BE BLANK
*   15-22      * NAME 2                *
*   23-24      * MUST BE BLANK         *
*   25-36      * VALUE                 *
****************************************
```

Figure 18. Data control record format

```
MOVE
     DATA         LPSAMPLE
     MINIMIZE     COST
     BOUNDS       ALLOY1
ENDATA
LPSOLUTION
```

Figure 19. Move and data control records required to solve LPSAMPLE

OUTPUT REPORTS

The 1130 LPS will produce the output report reproduced in Figure 21 for the alloy blending problem we formulated. Across the top of the report appears a series of headings describing the tabular information contained in the report. Under the heading "Variable" is a list of all the variables that figure in the problem. "Type" heads a column which indicates,

```
******************************************************************************
INPUT
NAME            LPSAMPLE
*****************************
*                           *
*                           *
*   INPUT DATA RECORDS      *
*   LPSAMPLE                *
*                           *
*                           *
*                           *
*****************************
ENDATA
MOVE
    DATA        LPSAMPLE
    MINIMIZE    COST
    BOUNDS      ALLOY1
ENDATA
LPSOLUTION
******************************************************************************
```

Figure 20. Schematic layout for LPSAMPLE input and solution

for each variable, whether the value of the variable in the solution lies at an upper limit (UL), a lower limit (LL), or an intermediate level (B*). The heading "Entries" appears over a column of figures indicating the number of equation elements input for each variable. The column headed "Solution Activity" indicates the total COST of the alloy blend, the total final WEIGHT of the alloy blend, the quantities of each raw material required, and the quantities of each of the chemical ingredients in the final blend. The next two columns, headed "Upper Bound" and "Lower Bound" respectively, indicate for each variable the upper and lower bound as defined in the set of bounds named ALLOY1 which controlled the solution of this LP problem. The next-to-last column, headed "Current Cost", provides, where relevant, the cost of the available raw materials.

The last column in the output report requires a separate treatment, since its significance is less obvious. Headed "Reduced Cost", it provides a significant figure for all those variables represented in the solution at a bound — either an upper or a lower bound. When the solution of a least-cost LP program indicates that some variable, either a raw material ingredient or a specified chemical, is present at a bound, it is fairly safe to assume that if the particular bound constraining the activity level of that variable were removed, the cost associated with the process would decrease. The reduced cost figure indicates how much less per unit measure (in this case, pounds) the cost of the final blend would be if the bound limiting each particular variable were relaxed — that is, made higher if the variable were at an upper bound, or made lower if the variable were at a lower bound. For example, the chemical specifications require that the final blend contain between 250 and 300 lbs. of silicon. In fact, the optimal solution contains 250 lbs., the lower limit. If that minimum of 250 lbs. could be relaxed, lowered somewhat, the reduced cost indicates that a saving of 24¢ per lb. would be realized. But note that this figure is valid only in the neighborhood of the solution. The user cannot assume that a specification of 100 lbs. less silicon will save $24. In effect, the reduced cost figure represents a rate of change (actually a slope) which does not hold over all values, but only in the immediate vicinity of the calculated solution. In any case the reduced cost figure gives the user a good indication of the cost of specification quality and the cost of inventory limitations, and can suggest at which points the user should introduce changes in order to operate more profitably.

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	301.434	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.242
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.288
BIN.2	B*	8	606.508	750.000	0.000	0.079	0.000
BIN.3	LL	7	0.000	800.000	0.000	0.170	-0.001
BIN.4	B*	7	554.733	700.000	0.000	0.120	0.000
BIN.5	B*	8	232.248	1500.000	0.000	0.150	0.000
ALUMINUM	B*	6	464.497	***********	0.000	0.210	0.000
SILICON	B*	4	142.011	***********	0.000	0.379	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-2.951
CU	B*	1	60.000	100.000	0.000	0.000	0.000
MN	UL	0	40.000	40.000	0.000	0.000	-0.908
MG	B*	0	20.517	30.000	0.000	0.000	0.000
AL	B*	0	1507.292	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.241
BASE	UL	0	120.000	120.000	0.000	0.000	-0.245

Figure 21. LPSOLUTION report

If the control deck includes a record calling LPANALYSIS after the LPSOLUTION procedure, an additional report in two parts, as illustrated in Figures 22 and 23, will be printed out for the aluminum blend problem. One section of the LP analysis report is titled "Variables at Upper Bound or Lower Bound" and the other is titled "Variables at Intermediate Level". Across the top of the LP analysis report appears a series of headings describing the tabular information contained in the report. Note that most columns have two headings, and each variable listed is followed by two rows of information. In each case the top heading identifies the information in the first row associated with each variable, and the bottom heading identifies the information in the second row.

Essentially, the LP analysis report is designed to indicate the effect that price changes, inventory availability changes, or specification changes would have on the cost of the final blend, and over what range the effect would be valid. To illustrate, consider the information output for the variable ALUMINUM in the report in Figure 23. The first three headings are self-explanatory. The variable is ALUMINUM, and the B* under "Type" indicates that it is at an intermediate level in the solution. "Solution Activity" gives the weight of ALUMINUM in the final blend, and "Current Cost" gives the input cost of ALUMINUM in the problem LPSAMPLE. There is no upper bound on ALUMINUM, and the lower bound, as indicated, is 0.

The next three columns of data require explanation. The figure which appears under "Cost/Unit Increase" indicates that for each additional pound of aluminum used, at the current cost, the cost of the alloy would increase by .1¢. The next column indicates that such a price change in the final blend would occur until 552.8 pounds of aluminum were used. Similarly, the figure under "Cost/Unit Decrease" indicates that if less aluminum were used in the final blend, the cost of the alloy would increase by 2.4¢ per pound until the amount of aluminum used was reduced to 393.8 lbs. Whenever a variable is present in the optimal solution at an intermediate level, it follows that any forced change, either increasing or decreasing the activity of that variable, will produce a rise in the price of the final blend. This report reveals not only the

VARIABLES AT UPPER BOUND OR LOWER BOUND

VARIABLE / TYPE	SOLUTION ACTIVITY / CURRENT COST	UPPER BOUND / LOWER BOUND	COST/UNIT INCREASE / COST/UNIT DECREASE	INCREASED ACTIVITY / DECREASED ACTIVITY	LOWEST COST / HIGHEST COST
WEIGHT	2000.000	2000.000	0.242	2179.629	-0.242
EQ	0.000	2000.000	-0.242	1993.060	***********
BIN.1	0.000	200.000	0.288	22.319	-0.258
LL	0.030	0.000	-0.288	-13.678	***********
BIN.3	0.000	800.000	0.001	100.512	0.168
LL	0.170	0.000	-0.001	-52.513	***********
FE	60.000	60.000	-2.951	62.033	***********
UL	0.000	0.000	2.951	55.198	2.951
MN	40.000	40.000	-0.908	41.579	***********
UL	0.000	0.000	0.908	38.059	0.908
SI	250.000	300.000	0.241	257.596	-0.241
LL	0.000	250.000	-0.241	160.727	***********
BASE	120.000	120.000	-0.245	122.318	***********
UL	0.000	0.000	0.245	114.360	0.245

Figure 22. LPANALYSIS variables at bounds

VARIABLES AT INTERMEDIATE LEVEL

VARIABLE	SOLUTION ACTIVITY	UPPER BOUND	COST/UNIT INCREASE	INCREASED ACTIVITY	LOWEST COST
TYPE	CURRENT COST	LOWER BOUND	COST/UNIT DECREASE	DECREASED ACTIVITY	HIGHEST COST
COST	301.434	***********	***********	301.434	***********
B*	-1.000	***********	***********	301.434	***********
BIN.2	606.508	750.000	0.000	743.597	0.079
B*	0.079	0.000	0.010	430.237	0.090
BIN.4	554.733	700.000	0.012	851.851	0.107
B*	0.120	0.000	0.001	462.548	0.121
BIN.5	232.248	1500.000	0.015	342.490	0.134
B*	0.150	0.000	0.000	-10.845	0.150
ALUMINUM	464.497	***********	0.001	552.816	0.208
B*	0.210	0.000	0.024	393.873	0.234
SILICON	142.011	***********	0.203	151.001	0.176
B*	0.379	0.000	0.091	140.524	0.471
CU	60.000	100.000	2.951	64.801	-2.951
B*	0.000	0.000	0.245	54.360	0.245
MG	20.517	30.000	0.076	22.307	-0.076
B*	0.000	0.000	0.421	16.332	0.421
AL	1507.292	***********	0.009	1521.251	-0.009
B*	0.000	1500.000	0.251	1459.289	0.251

Figure 23. LPANALYSIS variables at intermediate levels

per-unit price rise but also the range over which the price rise holds.

The last column ("Lowest Cost" and "Highest Cost") indicates that if aluminum drops to a price of 20.8¢ per lb., the activity level of aluminum in the solution will rise to 552.816 lbs. Further, if the price of aluminum rises to 23.4¢ per lb., the activity level of aluminum will fall from 464.497 lbs. to 393.873 lbs. Thus we have an indication that the quantity of aluminum in the final blend will remain constant at 464.497 lbs. as long as the price of aluminum remains between 20.8¢ and 23.4¢ per lb.

As a further illustration of the usefulness of the LP analysis report, consider the variable FE in the report pictured in Figure 22. Iron is present in the optimal solution at its upper level, 60 lbs. The cost per unit increase is -2.951, which means that for every additional pound of iron in the final blend, the cost of the blend would drop $2.951 per pound until 62.033 lbs. of iron were present. On the other hand, if less than 60 lbs. of iron were present, the cost of the final blend would increase $2.951 per lb., until the iron content was reduced to 55.198 lbs. Since the variable FE is a specification variable rather than a raw material variable, the highest-cost figure is not relevant.

Again, consider the variable BIN.3 in Figure 22. This material which costs 17¢ per pound is not used in the optimal solution. The cost per unit increase figure, together with the increased-activity figure, indicates that 100.5 lbs. of the material in BIN.3 could be used at an increased cost of only about .1¢ per lb. for the final blend. The lowest-cost figure associated with BIN.3 indicates that at an approximate price of 16.8¢ (only about .2¢ less than its price in LPSAMPLE*), over 100 lbs. would be used in the final blend. Such information provides the user with an excellent basis for purchasing and inventory maintenance decisions.

---

*The difference between the two values (.1¢ and .2¢) is due to output truncation.

We have examined the method for formulating a simple LP problem and the method for punching the input records that load the data on the computer and enable the 1130 LPS to solve the problem and produce an output report. We have examined the LPSOLUTION and LPANALYSIS solution reports. We need, now, to examine what capabilities the 1130 LPS provides to explore the effects of changes to several variables simultaneously, or to investigate changes that transcend the limits described by the LPANALYSIS report.

Assume that the price of the material in BIN. 2 and BIN. 5, depending on local market conditions, will fluctuate from the original cost of 8¢ and 15¢ per pound up to 11¢ and 18¢ per pound respectively. Assume, also, that the price of pure silicon might drop from 38¢ to 32¢ during these same market fluctuations. What effect would these particular price fluctuations have on the optimum solution? The 1130 LPS provides a procedure which enables us to discover the effects of such price changes. We can formulate a problem file that names the variables in which we are interested and indicates what changes of coefficients we would like to investigate. Assume that we would like to investigate the effect of these price fluctuations at each 1¢ increase in the cost of BIN. 2 and BIN. 5 and 2¢ decrease in SILICON cost. To accomplish this, we first input the problem file containing the change coefficients.

```
INPUT
NAME                    PARCOST
         BIN. 2         COST          .01
         BIN. 5         COST          .01
         SILICON        COST         -.02
ENDATA
```

Note that these records are the same type which are used to input the original problem data. First an 1130 LPS procedure named INPUT signals that data follows. Then an indicator record names the problem file (PARCOST) in which the data is to be stored. The data itself, then, appears in the form of equation element records indicating a cost increase of 1¢ for the materials in bins 2 and 5, and a cost change of -2¢ for silicon, since we wish to determine the effect of a price reduction in silicon. Finally an ENDATA indicator record signals the end of the data stream.

Figure 24 illustrates the data and control sequence required for this run. The data control record following the MOVE record indicates the number of solution REPORTS required.

For example:

```
MOVE
    PARAMET       REPORTS  3.0
ENDATA
```

The ENDATA indicator record signals the end of the data control stream. We have at this point loaded the computer with all the data required to compute a series of output reports of the same form as those output by LPSOLUTION, which will provide new solutions to the original problem showing the effects of the price variations in which we are interested. As shown in Figure 24, an LPPARAMETRIC procedure control record and a DATA record are required to call the appropriate procedure and to specify the data to be used respectively. No ENDATA indicator is required, since the LPPARAMETRIC procedure requires exactly one data control record. These two records will initiate the series of re-solutions as defined in the PARAMET data control records and the change data stored in PARCOST. Solution reports will be provided for each change level or interval, in our example, at the costs shown in Figure 25. The solution reports are shown in Figures 26, 27, and 28.

```
INPUT
NAME              PARCOST
     BIN.2        COST                0.01
     BIN.5        COST                0.01
     SILICON      COST          -     0.02
ENDATA
MOVE
     DATA         LPSAMPLE
     MINIMIZE     COST
     BOUNDS       ALLOY1
ENDATA
LPSOLUTION
MOVE
     PARAMET      REPORTS             3.0
ENDATA
LPPARAMETRIC
     DATA         PARCOST
```

Figure 24. Parametric cost setup and input data

```
*************************************
*        *       *        *         *
* REPORT * BIN.2* BIN.5  * SILICON  *
*        * COST * COST   * COST     *
*        *       *        *         *
*************************************
*************************************
*   1    * .09  * .16    * .36      *
*   2    * .10  * .17    * .34      *
*   3    * .11  * .18    * .32      *
*************************************
```

Figure 25. Parametric cost changes

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	306.197	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.244
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.312
BIN.2	B*	8	743.597	750.000	0.000	0.089	0.000
BIN.3	B*	7	100.512	800.000	0.000	0.170	0.000
BIN.4	B*	7	462.548	700.000	0.000	0.120	0.000
BIN.5	LL	8	0.000	1500.000	0.000	0.160	-0.003
ALUMINUM	B*	6	552.816	***********	0.000	0.210	0.000
SILICON	B*	4	140.524	***********	0.000	0.360	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-3.197
CU	B*	1	60.000	100.000	0.000	0.000	0.000
MN	UL	0	40.000	40.000	0.000	0.000	-0.464
MG	B*	0	22.307	30.000	0.000	0.000	0.000
AL	B*	0	1521.251	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.225
BASE	UL	0	120.000	120.000	0.000	0.000	-0.239

Figure 26. LPPARAMETRIC cost change — first report

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	310.767	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.247
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.337
BIN.2	B*	8	483.476	750.000	0.000	0.099	0.000
BIN.3	B*	7	212.117	800.000	0.000	0.170	0.000
BIN.4	UL	7	700.000	700.000	0.000	0.120	-0.000
BIN.5	LL	8	0.000	1500.000	0.000	0.170	-0.007
ALUMINUM	B*	6	485.679	***********	0.000	0.210	0.000
SILICON	B*	4	118.727	***********	0.000	0.340	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-3.418
CU	B*	1	60.000	100.000	0.000	0.000	0.000
MN	B*	0	35.460	40.000	0.000	0.000	0.000
MG	B*	0	14.504	30.000	0.000	0.000	0.000
AL	B*	0	1528.410	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.209
BASE	UL	0	120.000	120.000	0.000	0.000	-0.255

Figure 27. LPPARAMETRIC cost change — second report

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	313.227	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.243
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.278
BIN.2	B*	8	483.476	750.000	0.000	0.109	0.000
BIN.3	B*	7	212.117	800.000	0.000	0.170	0.000
BIN.4	UL	7	700.000	700.000	0.000	0.120	-0.009
BIN.5	LL	8	0.000	1500.000	0.000	0.180	-0.015
ALUMINUM	B*	6	485.679	***********	0.000	0.210	0.000
SILICON	B*	4	118.727	***********	0.000	0.320	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-2.963
CU	B*	1	60.000	100.000	0.000	0.000	0.000
MN	B*	0	35.460	40.000	0.000	0.000	0.000
MG	B*	0	14.504	30.000	0.000	0.000	0.000
AL	B*	0	1528.410	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.179
BASE	UL	0	120.000	120.000	0.000	0.000	-0.286

Figure 28. LPPARAMETRIC cost change — third report

The parametric data can include any of the equation coefficients and can also include bound entries. It is highly recommended that bound parametric data apply only to explicit bound entries. Parametric bound data for implicit bound data (for example, the upper bound of ALUMINUM is an implicit bound of infinity) can yield unexpected and undesired results.

Figure 29 shows an example that would produce a series of solution reports corresponding to iron specification changes from the input upper bound value of 60 lbs., at two-pound intervals, to a final value of 64 lbs. Figures 30 and 31 show the series of solution reports produced by this example.

During an LPPARAMETRIC run, the data files formed and used by the LPS optimization routines are actually altered to reflect the parametric changes specified by the user. Thus, in the previous example, at the end of the parametric run, the upper bound of the variable FE would actually be 64 lbs. The original "input" data files are not altered, and can be used again for processing.

```
INPUT
NAME                PARFE
 UB ALLOY1          FE                2.0
ENDATA
MOVE
     DATA           LPSAMPLE
     MINIMIZE       COST
     BOUNDS         ALLOY1
ENDATA
LPSOLUTION
MOVE
     PARAMET        REPORTS           2.0
ENDATA
LPPARAMETRIC
     DATA           PARFE
```

Figure 29.

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	295.532	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.242
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.288
BIN.2	B*	8	546.745	750.000	0.000	0.079	0.000
BIN.3	LL	7	0.000	800.000	0.000	0.170	-0.001
BIN.4	B*	7	697.633	700.000	0.000	0.120	0.000
BIN.5	B*	8	208.875	1500.000	0.000	0.150	0.000
ALUMINUM	B*	6	417.751	***********	0.000	0.210	0.000
SILICON	B*	4	128.994	***********	0.000	0.379	0.000
FE	UL	1	62.000	62.000	0.000	0.000	-2.951
CU	B*	1	58.000	100.000	0.000	0.000	0.000
MN	UL	0	40.000	40.000	0.000	0.000	-0.908
MG	B*	0	18.491	30.000	0.000	0.000	0.000
AL	B*	0	1505.603	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.241
BASE	UL	0	120.000	120.000	0.000	0.000	-0.245

Figure 30. LPPARAMETRIC bounds change — first report

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	291.791	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.219
BIN.1	B*	8	7.167	200.000	0.000	0.030	0.000
BIN.2	B*	8	628.857	750.000	0.000	0.079	0.000
BIN.3	B*	7	70.237	800.000	0.000	0.170	0.000
BIN.4	UL	7	700.000	700.000	0.000	0.120	-0.030
BIN.5	LL	8	0.000	1500.000	0.000	0.150	-0.011
ALUMINUM	B*	6	472.312	***********	0.000	0.210	0.000
SILICON	B*	4	121.425	***********	0.000	0.379	0.000
FE	UL	1	64.000	64.000	0.000	0.000	-0.624
CU	B*	1	55.999	100.000	0.000	0.000	0.000
MN	UL	0	40.000	40.000	0.000	0.000	-2.546
MG	B*	0	19.009	30.000	0.000	0.000	0.000
AL	B*	0	1515.992	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.192
BASE	UL	0	120.000	120.000	0.000	0.000	-0.272

Figure 31. LPPARAMETRIC bounds change — second report

We shall illustrate the data maintenance and editing features by using the alloy blend problem LPSAMPLE and the parametric problem data sets PARCOST and PARFE.

As a result of our optimization, we have used 606.5 lbs. of the original 750 lbs. of BIN.2, 554.7 lbs. of the original 700 lbs. of BIN.4, and 232.2 lbs. of the original 1500 lbs. of BIN.5. In addition, the quantities required of ALUMINUM and SILICON have been purchased and ALLOY1 has been produced.

Suppose that we now must produce, from our reduced inventory, ALLOY2. To run this new problem, we could prepare a new INPUT deck or REVISE the current problem data LPSAMPLE. Figure 32 gives the specifications of ALLOY2.

As a consequence of our ALLOY1 production, we can remove the bound set ALLOY1 and add a new bound set ALLOY2. To indicate our new inventory situation and the new ALLOY2, we shall REVISE the data of a NAMEd problem, LPSAMPLE, as shown in Figure 33.

This revision illustrates a new data specification type, the variable removal type. The effect of this type is to "X out" (cross out) the named variable or bound set. This element record requires an X in record position 2 and the bound set or variable name in the first name field (record position 5-12).

In our example we required only inventory and alloy specification changes. This was done by X'ing out the previous bound set, ALLOY1, and introducing a new bound set, ALLOY2. Normal day-to-day processing will probably require (1) new column variables to represent materials purchased, (2) price changes in current inventory variables to reflect market cost changes, (3) column variable removals as inventory variables are used up, and (4) row variable additions and removals as the model definition changes.

REVISE is the only way to alter data in a problem file and the <u>only</u> way to add or remove variable or bound sets.

The data order or sequence in REVISE is immaterial; the number of variables and bound sets to be X'd out is not limited; and the data removal element records may appear anywhere in the REVISE unit record data file.

NOTE: The same name cannot be X'd out and added during the same revision. For example, do not X out BIN.1 and then define a new BIN.1. Bound revisions should be made with care. The bound types FX and FR generate internal UB and LB entries. The FX type generates the same internal data as distinct UB and LB entries with identical coefficients for the same variable. Revision of an FX bound type by a UB bound type entry only does not alter the LB entry generated during input; and, conversely, revising by an LB entry only will not alter the UB entry. The FR bound type will generate a UB entry of plus infinity and an LB entry of minus infinity. Revision by a single LB or UB entry will have no effect on the other limit. After the REVISE data has been processed, the old problem data no longer exists. It is recommended, therefore, that all REVISE data be listed and checked before making the revision.

VARIABLE	UPPER BOUND	LOWER BOUND
WEIGHT	2000	2000
COST	NONE	NONE
FE	60	0
CU	100	0
MN	50	0
MG	25	0
AL	NONE	1500
SI	350	300
BASE	150	0

Figure 32. Alloy 2 specifications

```
REVISE
NAME            LPSAMPLE
 X   ALLOY1
 FR  ALLOY2     COST
 FX  ALLOY2     WEIGHT        2000.0
 UB  ALLOY2     BIN.1          200.0
 UB  ALLOY2     BIN.2          143.5
 UB  ALLOY2     BIN.3          800.0
 UB  ALLOY2     BIN.4          145.3
 UB  ALLOY2     BIN.5         1267.8
 UB  ALLOY2     FE              60.0
 UB  ALLOY2     CU             100.0
 UB  ALLOY2     MN              50.0
 UB  ALLOY2     MG              25.0
 UB  ALLOY2     SI             350.0
 LB  ALLOY2     SI             300.0
 LB  ALLOY2     AL            1500.0
 UB  ALLOY2     BASE           150.0
ENDATA
```

Figure 33.

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	366.265	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	0.124
BIN.1	B*	8	136.051	200.000	0.000	0.030	0.000
BIN.2	UL	8	143.500	143.500	0.000	0.079	-0.061
BIN.3	UL	7	800.000	800.000	0.000	0.170	-0.012
BIN.4	UL	7	145.300	145.300	0.000	0.120	-0.053
BIN.5	B*	8	13.575	1267.800	0.000	0.150	0.000
ALUMINUM	B*	6	553.921	***********	0.000	0.210	0.000
SILICON	B*	4	207.651	***********	0.000	0.379	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-0.656
CU	B*	1	84.516	100.000	0.000	0.000	0.000
MN	B*	0	19.638	50.000	0.000	0.000	0.000
MG	B*	0	7.161	25.000	0.000	0.000	0.000
AL	LL	0	1500.000	***********	1500.000	0.000	-0.346
SI	LL	0	300.000	350.000	300.000	0.000	-0.540
BASE	B*	0	144.516	150.000	0.000	0.000	0.000

Figure 34. The LPSOLUTION for alloy 2

DATA REVISION AND MERGE

The ability to add new elements, rows and column variables, and bound sets to an existing problem imposes certain responsibilities on the user. The revision data should be verified, listed, and checked before REVISEion to ensure protection of the existing data. If there are a substantial number of revisions, or if there are inadequate checking facilities, the MERGE procedure should be used to protect the original files. The MERGE procedure can be used to copy a disk problem data file or to combine one or more disk problem data files. (The use of MERGE in data generation is discussed later in this section.)

We shall first illustrate the use of MERGE to combine two data files (saving both), to form a new file. This conforms to standard data processing file protection methods. We may INPUT the changes into a problem file CHANGES on the disk. We may then form a new problem, LPSAMPL2 on the disk by MERGEing the DATA LPSAMPLE and the DATA CHANGES. Figure 35 lists the input and control records for this use of MERGE. Figure 36

```
INPUT
NAME               CHANGES
 FR ALLOY2         COST
 FX ALLOY2         WEIGHT          2000.0
 UB ALLOY2         BIN.1            200.0
 UB ALLOY2         BIN.2            143.5
 UB ALLOY2         BIN.3            800.0
 UB ALLOY2         BIN.4            145.3
 UB ALLOY2         BIN.5           1267.8
 UB ALLOY2         FE                60.0
 UB ALLOY2         CU               100.0
 UB ALLOY2         MN                50.0
 UB ALLOY2         MG                25.0
 UB ALLOY2         SI               350.0
 LB ALLOY2         SI               300.0
 LB ALLOY2         AL              1500.0
 UB ALLOY2         BASE             150.0
ENDATA
MERGE
    NAME           LPSAMPL2
    DATA           LPSAMPLE
    DATA           CHANGES
ENDATA
```

Figure 35.

illustrates the output log. Note one restriction — namely, that a variable or a bound set cannot be removed.

```
INPUT
NAME             CHANGES
ENDATA
PROBLEM 'CHANGES ' CONTAINS
     0 ROWS
     0 SELECTED ROWS
     0 VARIABLES
     0 SELECTED COLUMNS
     1 BOUNDS
     0 RHS'S
     0 RANGES
     0 COLUMN ELEMENTS
     4 LOWER BOUND ELEMENTS
    13 UPPER BOUND ELEMENTS
     0 RHS ELEMENTS
     0 RANGE ELEMENTS
MERGE
     NAME        LPSAMPL2
     DATA        LPSAMPLE
     DATA        CHANGES
ENDATA
PROBLEM 'LPSAMPL2' CONTAINS
     9 ROWS
     0 SELECTED ROWS
    16 VARIABLES
     0 SELECTED COLUMNS
     2 BOUNDS
     0 RHS'S
     0 RANGES
    50 COLUMN ELEMENTS
     8 LOWER BOUND ELEMENTS
    26 UPPER BOUND ELEMENTS
     0 RHS ELEMENTS
     0 RANGE ELEMENTS
```

Figure 36.

MERGE may also be used to prepare a copy of the problem file before revision. We may first use MERGE to form a new copy, which we NAME LPSAMPL2 from the DATA LPSAMPLE. We then REVISE the problem LPSAMPLE, leaving the copy LPSAMPL2 untouched. We may later list the revisions if there are processing problems. Figure 37 lists the input and control records for this use of MERGE. Figure 38 illustrates the output log.

```
MERGE
     NAME        LPSAMPL2
     DATA        LPSAMPLE
ENDATA
MOVE
     DATA        LPSAMPLE
ENDATA
REVISE
 X   ALLOY1
 FR  ALLOY2      COST
 FX  ALLOY2      WEIGHT        2000.0
 UB  ALLOY2      BIN.1          200.0
 UB  ALLOY2      BIN.2          143.5
 UB  ALLOY2      BIN.3          800.0
 UB  ALLOY2      BIN.4          145.3
 UB  ALLOY2      BIN.5         1267.8
 UB  ALLOY2      FE              60.0
 UB  ALLOY2      CU             100.0
 UB  ALLOY2      MN              50.0
 UB  ALLOY2      MG              25.0
 UB  ALLOY2      SI             350.0
 LB  ALLOY2      SI             300.0
 LB  ALLOY2      AL            1500.0
 UB  ALLOY2      BASE           150.0
ENDATA
```

Figure 37.

```
MERGE
     NAME        LPSAMPL2
     DATA        LPSAMPLE
ENDATA
PROBLEM 'LPSAMPL2' CONTAINS
     9 ROWS
     0 SELECTED ROWS
    16 VARIABLES
     0 SELECTED COLUMNS
     1 BOUNDS
     0 RHS'S
     0 RANGES
    50 COLUMN ELEMENTS
     4 LOWER BOUND ELEMENTS
    13 UPPER BOUND ELEMENTS
     0 RHS ELEMENTS
     0 RANGE ELEMENTS
MOVE
     DATA        LPSAMPLE
ENDATA
REVISE
ENDATA
PROBLEM 'LPSAMPLE' CONTAINS
     9 ROWS
     0 SELECTED ROWS
    16 VARIABLES
     0 SELECTED COLUMNS
     2 BOUNDS
     0 RHS'S
     0 RANGES
    50 COLUMN ELEMENTS
     8 LOWER BOUND ELEMENTS
    26 UPPER BOUND ELEMENTS
     0 RHS ELEMENTS
     0 RANGE ELEMENTS
```

Figure 38.

In addition, since we have produced ALLOY1, we are no longer interested in the parametric data PARCOST. Consequently, we shall DELETE this problem DATA named PARCOST. Since we may wish to DELETE several obsolete DATA files, this procedure is also terminated by an ENDATA indicator.

```
DELETE
    DATA            PARCOST
ENDATA
```

When these records are processed, the DATA named PARCOST ceases to be available to the user. The area on the disk occupied by the problem data will not be made available to the user until the disk is EDITed.

In general, when a procedure requires a single data control card, no ENDATA indicator is required. When a procedure may be used with one or more data control cards, an ENDATA indicator is required to signify end of data control cards for the procedure.

The EDIT procedure makes available for additional problem files the disk space that has been occupied by deleted files. EDIT automatically calls the DICTIONARY procedure, which causes a report to be printed out listing the names of all the remaining problem files on the disk, the number of disk sectors used by each, and the number of the next disk sector available for use, as illustrated below:

PROBLEM NAME	NO. SECTORS
LPSAMPLE	3
NEXT AVAILABLE SECTOR IS	9

STARTING SOLUTIONS

The time required to find the optimal solution may depend heavily on the solution used to start the process. The 1130 LPS will perform a number of iterations (changes in solution), using each iteration as a solution base to find an improved solution. This process is continued until the optimum solution is computed.

The 1130 LPS will begin by setting the row and column variables at more or less arbitrary levels (lower bound, upper bound, or intermediate level) unless the user RESTOREs advanced solution levels from a disk solution file. Often the user can provide good starting solution levels based on past experience. Indeed, most optimizations are not the first for a particular model. Usually the same model is optimized many times, the subsequent optimizations being required by changes in some of the equation coefficients and/or some changes in the problem bounds. Thus significant optimization processing time may be saved by using the first solution of a model as an advanced solution for the second optimization, the second solution as an advanced solution for the third, and so forth. The procedure SAVESOLUTION is recommended to save the current optimization solution levels in a NAMEd file for models that are kept and maintained on the disk, as shown in Figure 39. The PUNCH and INSERT procedures are recommended for models that will be maintained in unit record form, as shown in Figure 40.

The SAVESOLUTION procedure requires a single data control record to designate a NAME for the starting solution level file.

The RESTORE procedure requires a single data control record to specify which disk-stored DATA file contains the starting solution levels.

```
SAVESOLUTION
    NAME        LPSOL

RESTORE
    DATA        LPSOL
```

```
****************************************
MOVE
    DATA       LPSAMPLE
ENDATA
REVISE
*****************************
*                           *
* MAINLY COEFFICIENT,BOUND  *
*        CHANGES            *
*                           *
*                           *
*****************************
MOVE
    BOUNDS     ALLOY1
    MINIMIZE   COST
ENDATA
RESTORE
    DATA       LPSAMPLE
LPSOLUTION
SAVESOLUTION
    NAME       LPSAMPLE
****************************************
```

Figure 39. Use of SAVESOLUTION and RESTORE. This illustrates a subsequent optimization of the model. During the first optimization, the data would be INPUTted and (usually) no advance solution would be available for RESTOREation.

```
****************************************
INPUT
******************************
*                            *
*                            *
*   DATA IN UNIT RECORD FORM *
*                            *
*                            *
******************************
INSERT
******************************
*                            *
*   PREVIOUS SOLUTION RUN    *
*         LEVELS             *
*                            *
******************************
MOVE
     DATA        LARGEPRB
     BOUNDS      LIMITSPB
     MAXIMIZE    PROFIT
ENDATA
RESTORE
     DATA        LARGEPRB
LPSOLUTION
PUNCH
******************************
*                            *
*   BLANK CARDS FOR NEW      *
*   SOLUTION LEVELS          *
*                            *
******************************
DELETE
     DATA        LARGEPRB
ENDATA
EDIT
****************************************
```

Figure 40. Use of INSERT, RESTORE, and PUNCH (subsequent optimization)

CONDITIONAL CONTROL

So far we have assumed perfection. That is, we have assumed that the input data was correctly prepared, that the problem definition provides a feasible answer (it is possible to satisfy all of the bounds of the problem), and that the solution is bounded (there is a finite cost). The problem, however, may be infeasible (that is, no solution may exist which satisfies all of the bounds of the bound set used in optimization), or the problem may be unbounded (that is, a feasible solution may exist but there may be no lowest cost or no highest profit), or the 1130 LPS has encountered a processing error during optimization and cannot continue optimization.

Each of these conditions is a possibility during any optimization and is usually due to faulty formulation or incorrect input data. These conditions can usually be cured by reformulation, or by REVISEion of input data.

Many experienced users of LP systems expect to find some trouble in the first optimization of a model. The experienced user will often INPUT a problem during one LP run and check the INPUT statistics for clues to obvious data preparation errors before the first try at optimization. When the problem is sufficiently small, or if the model and most of the data have previously been run, the experienced user may include optimization. The use of LPANALYSIS and/or LPPARAMETRIC should generally be limited to checked-out models.

Suppose, now, that we have loaded a large alloy blend problem named BLEND, and that the model and data are pretty well checked out with two bound sets (BOUND1 and BOUND2) which reflect somewhat different specifications. Either blend is acceptable, but we prefer the blend defined by BOUND1, if this bound set provides a feasible solution.

BOUND1 has tighter specifications (smaller upper bounds or larger lower bounds) and is more likely to be infeasible than BOUND 2. Hence we wish, at a certain point in the LPS procedure sequence, to OPTIMIZE the blend defined by BOUND1 and, IF the solution is INFEASible, to bypass LPANALYSIS.

To provide these additional functions, we introduce the procedures: OPTIMIZE, which obtains an LPSOLUTION but does not print an LPSOLUTION report, and IF, which provides a conditional control of the LPS procedures on the conditions and in the order specified by one or more data control records.

IF — data control records. The first name field specifies the condition. If the condition is true (for example, IF the problem is INFEASible), a search is performed for the LABEL record whose name or label is specified by the second name field (for example, 1REOPTIM). If the condition is not true, the next data control record is read until an ENDATA indicator is encountered.

LABEL records. These are unit records with a label (instead of a procedure name) in position 1-8 of the record. The user must use a numeric character for the first character of a label.

Figure 41 illustrates a control sequence that will provide the conditional control desired for our large BLEND problem with the tighter specifications in BOUND1 than in BOUND2.

The procedure execution sequence is identical whether BOUND1 is infeasible or not, except that the LPANALYSIS procedure preceding 1REOPTIM will be bypassed IF BOUND1 is INFEASible.

The various conditions that can be tested through the agency of the IF and IF NOT procedures are listed in Chapter 2. In every case, if the condition tests negatively, the next data control record is read. For example, suppose BOUND1 provides a feasible solution — this is a negative test on the condition INFEAS. If all IF data control records have been read, and if all test negative, the ENDATA indicator signifies end of testing and the next procedure control record is read. If the condition tests positive, the procedure results in a scan until the appropriate label is encountered, whereupon the run recommences.

```
MOVE
     DATA        BLEND
     MINIMIZE    COST
     BOUNDS      BOUND1
ENDATA
OPTIMIZE
IF
     INFEAS      1REOPTIM
ENDATA
LPANALYSIS
1REOPTIM
MOVE
     BOUNDS      BOUND2
ENDATA
LPSOLUTION
LPANALYSIS
```

Figure 41.

DATA GENERATION AND MERGE

The sample problem has shown how to formulate and prepare data for a single blend of material. In many blending applications more than one product (for example, several alloys) must be made from the same inventory.

In the previous example, we supposed that only one alloy could be produced. Let us see what would be required to prepare a multiblend problem, where instead of producing one alloy from the ingredients, we shall produce two alloys.

First, we must provide a new formulation, complete with new names for the variables. The new names are required to differentiate between materials in the first blend and materials in the second blend. A common technique to simplify name generation and result interpretation is to assign a specific character or characters (for example, 1, 2, A, or B) to appear in some part of the name. If we arbitrarily choose the letter A to replace the current character appearing in position 6 of the column variable names, each name will be unique — BIN.1A, BIN.2A, BIN.3A, BIN.4A, BIN.5A, ALUMIAUM, SILICAN — and this representation can be used to distinguish the materials to be used in the first blend. Similarly, we may arbitrarily choose the letter B in position 6 of the column variable names to provide a unique representation of variables used in the second blend — BIN.2B,..., SILICBN.

The change of column names is the first step in the new formulation; we must also uniquely change the row names to differentiate between the alloys. Again we may arbitrarily select the numbers 1 and 2 to change the third position of the row variables for blends 1 and 2 respectively. Hence, the new formulation (in part) of the problem would be:

First alloy. Equations 1-8 as shown on page 5, except that the column variable names would contain the letter A as the sixth character of the name, and the row variable names would contain the digit 1 as the third character of the name.

Second alloy. Equations 9-16 as shown by equations 1-8 on page 5, except that column variable names would incorporate the letter B in the sixth position and row variable names would incorporate the digit 2 in the third position.

These equations do not complete the equation part of the formulation, since the material usage is not limited. It will not be sufficient to place an upper bound on BIN.2A or BIN.2B, since we are concerned with the total usage of the material rather than the individual ingredient-alloy usage. Hence, to express the total material usage and the total cost, we also include the equations:

BIN.1T = 1.0 BIN.1A + 1.0 BIN.1B (19)

BIN.2T = 1.0 BIN.2A + 1.0 BIN.2B (20)

BIN.3T = 1.0 BIN.3A + 1.0 BIN.3B (21)

BIN.4T = 1.0 BIN.4A + 1.0 BIN.4B (22)

BIN.5T = 1.0 BIN.5A + 1.0 BIN.5B (23)

ALUMITUM = 1.0 ALUMIAUM + 1.0 ALUMIBUM (24)

SILICTN = 1.0 SILICAN + 1.0 SILICBN (25)

COTT = 1.0 CO1T + 1.0 CO2T (26)

These equations (19 to 26) are usually called linking equations and are often shown as the first equations in a schematic or diagrammatic display of the model, as shown in Figure 42. The equations used to represent the total aluminum, ALUMITUM, and total silicon, SILICTN, are included only for convenience, since no limits are imposed on their usage. This convenience should be avoided when the number of such unnecessary equations is large and when processing time is limited. The bounds on the problem are shown in Figure 43.

It remains now to discuss the provisions within the 1130 LPS which may be used to simplify the data preparation in problems of this sort. First, we shall INPUT the data corresponding to the linking equations and the required bounds. This INPUT, LINKEQ is shown in Figure 44. The MERGE procedure provides the ability to combine one or more problems (stored on the disk) to form a new problem. We may specify a fixed variation in the names of the column variables, and a different fixed variation in the names of the rows variables.

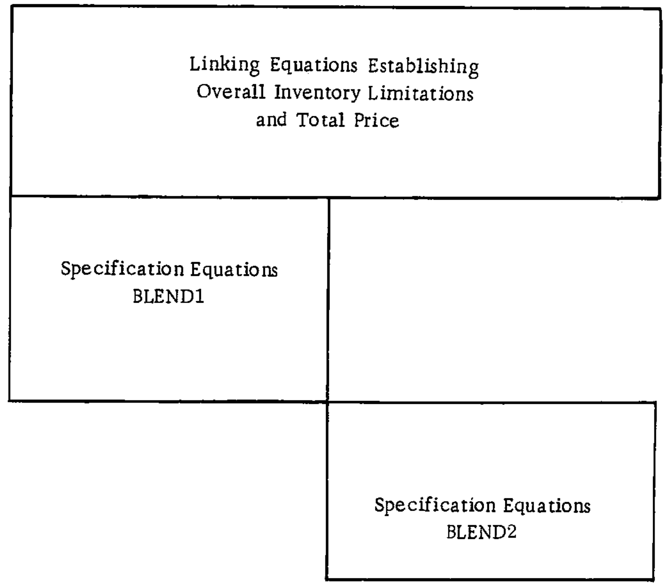


Figure 42. Schematic model for multiblend problem

The new problem file is established by a NAME data control record. We now wish to change the variable names as described in the multiblend formulation.

Generally a MERGE procedure control record requires data control records naming the problem files to be merged. However, the system also provides the feature of changing the names of the existing problem files (in the MERGE process but without actually changing the original data) through the agency of data control records.

An optional ROWS data control record will cause whatever nonblank characters appear in the name field (15-22) to be superimposed on the row variable names. For example, a 1 in the third position of the name field (record position 17) will cause the third position of each row variable name to contain the character 1.

Similarly, an optional COLS data control record with an A in the sixth position of the name field (record position 20) will cause the sixth position of each column variable name to contain the character A.

The sequence of data control records is shown in Figure 45, the run setup in Figure 46, and the output in Figure 47.

	VARIABLE	UPPER BOUND	LOWER BOUND
RAW MATERIALS	BIN•1T	200	0
	BIN•2T	750	0
	BIN•3T	800	0
	BIN•4T	700	0
	BIN•5T	1500	0
	ALUMITUM	NONE	0
	SILICTN	NONE	0
FIRST ALLOY	WE1GHT	2000	2000
	CO1T	NONE	NONE
	FE1	60	0
	CU1	100	0
	MN1	40	0
	MG1	30	0
	AL1	NONE	1500
	SI1	300	250
	BA1E	120	0
SECOND ALLOY	WE2GHT	2000	2000
	CO2T	NONE	NONE
	FE2	60	0
	CU2	100	0
	MN2	50	0
	MG2	25	0
	AL2	NONE	1500
	SI2	350	300
	BA2E	150	0
TOTAL COST	COTT	NONE	NONE
ALLOY MATERIALS	BIN• A	NONE	0
	ALUMIAUM	NONE	0
	SILICAN	NONE	0
	BIN• B	NONE	0
	ALUMIBUM	NONE	0
	SILICBN	NONE	0

Figure 43. Bounds on the multiblend problem

```
INPUT
NAME             LINKEQ
    CO1T         COTT            1.0
    CO2T         COTT            1.0
    BIN.1A       BIN.1T          1.0
    BIN.1B       BIN.1T          1.0
    BIN.2A       BIN.2T          1.0
    BIN.2B       BIN.2T          1.0
    BIN.3A       BIN.3T          1.0
    BIN.3B       BIN.3T          1.0
    BIN.4A       BIN.4T          1.0
    BIN.4B       BIN.4T          1.0
    BIN.5A       BIN.5T          1.0
    BIN.5B       BIN.5T          1.0
    ALUMIAUM     ALUMITUM        1.0
    ALUMIBUM     ALUMITUM        1.0
    SILICAN      SILICTN         1.0
    SILICBN      SILICTN         1.0
 UB ALLOYMU      BIN.1T        200.0
 UB ALLOYMU      BIN.2T        750.0
 UB ALLOYMU      BIN.3T        800.0
 UB ALLOYMU      BIN.4T        700.0
 UB ALLOYMU      BIN.5T       1500.0
 FX ALLOYMU      WE1GHT       2000.0
 UB ALLOYMU      FE1            60.0
 UB ALLOYMU      CU1           100.0
 UB ALLOYMU      MN1            40.0
 UB ALLOYMU      MG1            30.0
 LB ALLOYMU      AL1          1500.0
 UB ALLOYMU      SI1           300.0
 LB ALLOYMU      SI1           250.0
 UB ALLOYMU      BA1E          120.0
 FX ALLOYMU      WE2GHT       2000.0
 UB ALLOYMU      FE2            60.0
 UB ALLOYMU      CU2           100.0
 UB ALLOYMU      MN2            50.0
 UB ALLOYMU      MG2            25.0
 LB ALLOYMU      AL2          1500.0
 UB ALLOYMU      SI2           350.0
 LB ALLOYMU      SI2           300.0
 UB ALLOYMU      BA2E          150.0
ENDATA
```

Figure 44. LINKEQ input

```
MERGE
    NAME        MULTIBLE
    ROWS          1
    COLS              A
    DATA        LPSAMPLE
    ROWS          2
    COLS              B
    DATA        LPSAMPLE
    DATA        LINKEQ
ENDATA
REVISE
 X  ALLOY2
ENDATA
```

Figure 45. The MERGE data control cards to generate the multiple blend problem MULTIBLE

```
INPUT

NAME            LINKEQ
******************************
*                            *
*        LINKING EQUATIONS   *
*                            *
*        INVENTORY           *
*        SPECIFICATION       *
*        BOUNDS              *
*                            *
******************************
ENDATA

MERGE
    NAME        MULTIBLE
    ROWS          1
    COLS              A
    DATA        LPSAMPLE
    ROWS          2
    COLS              B
    DATA        LPSAMPLE
    DATA        LINKEQ
ENDATA
REVISE
 X  ALLOY2
ENDATA
MOVE
    MINIMIZE    COTT
    BOUNDS      ALLOYMU
    PUNCH       PRINTER
ENDATA
BCDOUT
LPSOLUTION
LPANALYSIS
```

Figure 46. The run setup for MULTIBLE

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
CO1T	B*	1	301.716	***********	0.000	1.000	0.000
WE1GHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.235
BIN.1A	LL	9	0.000	***********	0.000	0.000	-0.098
BIN.2A	B*	9	578.338	***********	0.000	0.000	0.000
BIN.3A	LL	8	0.000	***********	0.000	0.000	-0.028
BIN.4A	B*	8	577.949	***********	0.000	0.000	0.000
BIN.5A	B*	9	249.866	***********	0.000	0.000	0.000
ALUMIAUM	B*	7	453.210	***********	0.000	0.000	0.000
SILICAN	B*	5	140.635	***********	0.000	0.000	0.000
FE1	UL	1	60.000	60.000	0.000	0.000	-1.014
CU1	B*	1	60.000	100.000	0.000	0.000	0.000
MN1	B*	0	39.689	40.000	0.000	0.000	0.000
MG1	B*	0	19.848	30.000	0.000	0.000	0.000
AL1	B*	0	1506.723	***********	1500.000	0.000	0.000
SI1	LL	0	250.000	300.000	250.000	0.000	-0.207
BA1E	UL	0	120.000	120.000	0.000	0.000	-0.865
CO2T	B*	1	365.788	***********	0.000	1.000	0.000
WE2GHT	EQ	0	2000.000	2000.000	2000.000	0.000	0.226
BIN.1B	B*	9	135.776	***********	0.000	0.000	0.000
BIN.2B	B*	9	171.661	***********	0.000	0.000	0.000
BIN.3B	B*	8	799.999	***********	0.000	0.000	0.000
BIN.4B	B*	8	122.050	***********	0.000	0.000	0.000
BIN.5B	LL	9	0.000	***********	0.000	0.000	-0.014
ALUMIBUM	B*	7	561.517	***********	0.000	0.000	0.000
SILICBN	B*	5	208.993	***********	0.000	0.000	0.000
FE2	UL	1	60.000	60.000	0.000	0.000	-0.466
CU2	B*	1	84.712	100.000	0.000	0.000	0.000
MN2	B*	0	20.023	50.000	0.000	0.000	0.000
MG2	B*	0	7.865	25.000	0.000	0.000	0.000
AL2	LL	0	1500.000	***********	1500.000	0.000	-0.448
SI2	LL	0	300.000	350.000	300.000	0.000	-0.639
BA2E	B*	0	144.712	150.000	0.000	0.000	0.000
COTT	B*	0	667.505	***********	0.000	-1.000	1.000
BIN.1T	B*	0	135.776	200.000	0.000	0.000	0.000
BIN.2T	UL	0	750.000	750.000	0.000	0.000	-0.049
BIN.3T	UL	0	800.000	800.000	0.000	0.000	-0.003
BIN.4T	UL	0	700.000	700.000	0.000	0.000	-0.047
BIN.5T	B*	0	249.866	1500.000	0.000	0.000	0.000
ALUMITUM	B*	0	1014.727	***********	0.000	0.000	0.000
SILICTN	B*	0	349.628	***********	0.000	0.000	0.000

Figure 47. MERGE problem MULTIBLE LPSOLUTION report

Chapter 2: THE LPS PROCEDURES

CONTROLLING LPS

This chapter provides a description of the LPS procedures. Each procedure carries out a specific task — for example:

1. INPUT the problem data
2. MOVE the processing specifications or parameters, such as bound set name, objective
3. OPTIMIZE the specified problem
4. LPSOLUTION report preparation and writing

This chapter is intended to serve as a reference for procedure usage. System concepts, formulation, and general description have been given in Chapter 1.

It is highly recommended that the user read the appropriate procedure description in this chapter before using a new or unfamiliar procedure.

The new LPS user should begin by formulating one or more small practice problems. It is advisable to begin with a simple procedure sequence, gradually increasing familiarity with the procedures that appear to be most relevant to the application.

Figure 48 provides a table of the functions and procedures of 1130 LPS.

ERROR PHILOSOPHY

The system will continue the procedure sequence, except for conditional control bypass, even if errors are made in the problem description or in the specification of processing parameters. Whenever possible, corrective action is taken by the procedures to continue operation.

LPS distinguishes two types of errors: minor errors, which allow the procedure to continue processing, and major errors, which cause the procedure to discontinue processing.

Minor errors. These errors are usually due to incorrect data. A message is printed describing the error. The procedure will also print any corrective action taken.

Major errors. These errors are usually due to invalid procedure or data control sequence. An appropriate error message is printed, and in many of the procedures, the procedure will discontinue processing. The next procedure requested will be read and called to perform its task.

Function	Procedure	Purpose
Data Maintenance	INPUT	Reads unit record data to form problem file, then calls STATISTICS
	STATISTICS	Writes tabular summary of current problem file
	REVISE	Reads unit record data to alter problem file, then calls STATISTICS
	MERGE	Combines (variations) of problem files to form new file, then calls STATISTICS
	BCDOUT	Writes unit record data from problem file
	DELETE	Removes obsolete problem files
	EDIT	Frees disk space previously used by deleted problems, calls DICTIONARY
	DICTIONARY	Lists names of problem files stored on disk
Processing Specification	MOVE	Sets or alters problem and system parameters
Problem Solution	OPTIMIZE	Finds solution to linear programming problem
	LPSOLUTION	Writes unit record LP solution report calls, OPTIMIZEs if optimization not done
	RESTART	Provides OPTIMIZE recovery procedure
Postoptimal Analysis	LPANALYSIS	Call OPTIMIZE if optimization not done. Writes unit record LP analysis report
	LPPARAMETRIC	Performs series of optimizations with LPSOLUTION reports as LP problem file is modified by change data
Starting Solutions	SAVESOLUTION	Saves current solution status file for next reoptimization
	RESTORE	Restores a previous solution for new optimization
	PUNCH	Writes current solution status records
	INSERT	Forms a previous solution file from unit records
Conditional Control	IF	Bypasses part of procedure sequence if condition true
	IFNOT	Bypasses part of procedure sequence if condition false
Solution Simultaneous Equations	SOLVE	Writes solution to set of simultaneous equations
Program Termination	END	Terminates LP-MOSS run

Figure 48.

RECORD FORMATS

The LPS uses three record formats: indicator/ procedure call (Figure 49), data/data control (Figure 50), and comment (Figure 51).

PROCEDURE CALL RECORD FORMAT

As illustrated in Figure 49, the procedure call format is:

Position 1 -12 Procedure name. Must be left-justified and must not contain embedded blanks.

Position	Contents
1-12	Indicator or procedure name
15-22	Name indicator only, the problem names
Other	Must be blank

Figure 49. Indicator/procedure call format

DATA CONTROL RECORD FORMAT

As illustrated in Figure 50, the data control format is:

Positions 1-4 Must be blank.

Positions 5-12 Contain the data control name — for example, PARAMET, DATA

Positions 15-22 Contain an I/O designation (discussed later), or a processing specification name (for example, INTERVAL), or a user-defined name (for example, LPSAMPLE).

Positions 25-36 Contains the value assigned for processing specifications.

System-defined names, such as data control names, must be left-justified and must not contain embedded blanks.

User-defined names consist of one to eight alphameric characters. It is recommended that names be left-justified and not contain embedded blanks.

A number in a value field may consist of from one to eleven digits. The number must contain a decimal point. The number must not contain embedded blanks. A negative number is denoted by an 11-punch in position 25. Punching the decimal point in position 30 will simplify data checking. If a decimal point is not punched in a value field, the decimal point is assumed to lie between the sixth and seventh field positions.

An invalid value field may cause an 1130 MONITOR stop displaying F003 in the accumulator.

Position	Contents
1-4	Blank except for bounds record
5-12	First system-defined name or user-defined name
15-22	Second system-defined name or user-defined name
25-36	Value field
Other	Field positions should be blank in Data Control Format

Figure 50. Data/data control format

Position	Contents
1-24	Comment, asterisk must be in column 1
25-End	Must be blank if in input or revise data

Figure 51. Comment format

The ENDATA Indicator

Some procedures, such as SAVESOLUTION, read only one data control record. Others (for example, DELETE) read a variable number. All the procedures that read a variable number of data control records require an ENDATA indicator record to mark the end of the particular data control deck.

Procedures requiring an ENDATA indicator are:

INPUT	IF and IFNOT INSERT
REVISE	MERGE
MOVE	DELETE

Procedures reading data control records and not requiring an ENDATA indicator are:

SAVESOLUTION

RESTORE

LPPARAMETRIC

INPUT

The INPUT procedure reads input data records and forms a problem file containing the data on the disk. The input data may represent a complete problem or may be a partial set of data for later use with MERGE or LPPARAMETRIC procedures.

Input Data

The format and contents of the records in an input data deck must conform to the specifications in Chapter 1, under "Data Preparation". Such a data deck must contain:

1. A NAME indicator record. (This record names the problem file. If no NAME indicator is provided, the problem file will be assigned a name consisting of eight blanks. If more than one NAME indicator appears within the data deck, the last name encountered will be used. Any problem file already on the disk and having the same name as the current problem will automatically be deleted.)
2. The input data for the problem
3. An ENDATA indicator record

A /LISTON (record positions 1-7) can be used to begin listing the input data on the typewriter. Data records following the /LISTON will be listed on the typewriter until a /LISTOFF (record positions 1-8) or ENDATA record. The input may contain several /LISTON and /LISTOFF records for selective listing of input data.

Output

After INPUT has sorted the problem file and checked the data for duplicates, the STATISTICS procedure is called to provide a summary of the data.

Major Errors

If an invalid indicator record is found, a major error is recorded and the indicator is treated as an ENDATA indicator. If some of the problem data has been bypassed because of the invalid indicator, the bypassed data may be added to the partially INPUT problem file by using the REVISE procedure.

STATISTICS

The STATISTICS procedure provides a statistical summary of the current problem. This procedure is called automatically by INPUT, REVISE, and MERGE.

The user may obtain the STATISTICS of a previously stored problem by first using MOVE to obtain the required DATA.

Output

The summary contains one line stating the name of the problem and one line for each of the types of data indicating the number of elements contained in each set.

Example

```
MOVE
      DATA            LPSAMPLE
ENDATA
STATISTICS
```

The tabular summary for the problem LPSAMPLE discussed earlier would be logged as in Figure 52. The logged information is interpreted as follows:

PROBLEM 'LPSAMPLE' CONTAINS
- 9 ROWS (the number of row variables in the problem)
- 0 SELECTED ROWS (the number of row variables selected by a ROWS file)
- 16 VARIABLES (the total number of variables in the problem)
- 0 SELECTED COLUMNS (the number of column variables selected by a COLS file)
- 1 BOUNDS (the number of bound sets input)
- 0 RHS'S (the number of right-hand-side sets input)*
- 0 RANGES (the number of range sets input)*
- 50 COLUMN ELEMENTS (the number of equation elements input for the problem)
- 4 LOWER BOUND ELEMENTS (the number of lower bound elements input for the problem)
- 13 UPPER BOUND ELEMENTS (the number of upper bound elements input for the problem)
- 0 RHS ELEMENTS (the number of RHS elements input for the problem)*
- 0 RANGE ELEMENTS (the number of range elements input for the problem)*

*Optional MPS/360 compatibility data types discussed at the end of Chapter 3.

Major Errors

None

```
PROBLEM 'LPSAMPLE' CONTAINS
     9 ROWS
     0 SELECTED ROWS
    16 VARIABLES
     0 SELECTED COLUMNS
     1 BOUNDS
     0 RHS'S
     0 RANGES
    50 COLUMN ELEMENTS
     4 LOWER BOUND ELEMENTS
    13 UPPER BOUND ELEMENTS
     0 RHS ELEMENTS
     0 RANGE ELEMENTS
```

Figure 52. LPSAMPLE statistics log

REVISE

REVISE reads input data specifying deletions, changes, or additions to the data contained in a problem file stored on the disk. To revise an existing file, either (1) MOVE the problem DATA before calling REVISE, or (2) REVISE a NAMEd problem data file.

Input Data

The format and rules for compiling a data deck to be read by REVISE are the same as those for INPUT, except that (1) if the first card of a revision deck is a NAME card, the NAMEd problem will be revised, (2) if the first card of a revision deck is not a NAME card, the current problem will be revised, (3) if a NAME card appears after the first card of a revision deck, the NAME card will be treated as an invalid indicator, and (4) removal of specific variables may be accomplished by punching an X in either the second or third position of the record, and the name of the variable, bound set, etc., to be removed in the first name field (positions 5-12).

Thus, the following records would enable the user to remove the variable BIN.1 from the problem file LPSAMPLE:

```
REVISE
NAME              LPSAMPLE
 X  BIN.1
ENDATA
```

/LISTON and /LISTOFF may be used for selective listing of revisions.

In order to revise the coefficients of variables, the values in bounds sets, etc., simply prepare element data records with the new values. The new values will be incorporated into the problem file, and the old values will be replaced automatically. Similarly, the addition of new variables or coefficients is accomplished by preparing new element data records to be read by the REVISE procedure. Only additions, changes, or deletions must be input. The remainder of the problem file is unchanged.

Output

After the problem has been updated as specified by the data cards read by the REVISE procedure, the STATISTICS procedure is automatically called for to provide a data summary. Figure 53 illustrates the STATISTICS log after the REVISE procedure which eliminated BIN.1 from the problem file LPSAMPLE.

Major Errors

If an invalid indicator record is found, a major error is recorded and the indicator is treated as an ENDATA record. Any remaining revisions can be made by again employing the REVISE procedure.

```
PROBLEM 'LPSAMPLE' CONTAINS
     9 ROWS
     0 SELECTED ROWS
    15 VARIABLES
     0 SELECTED COLUMNS
     1 BOUNDS
     0 RHS'S
     0 RANGES
    42 COLUMN ELEMENTS
     4 LOWER BOUND ELEMENTS
    12 UPPER BOUND ELEMENTS
     0 RHS ELEMENTS
     0 RANGE ELEMENTS
```

Figure 53. LPSAMPLE statistics after deleting BIN.1

NOTE: When removing obsolete data, do not include a new variable with the same name as a removed variable. If you do, the new variable will be removed. The variable may subsequently be REVISEd into the problem in a later REVISEion. Note also that some of the bound types generate an upper bound and a lower bound entry. An UB (UP) or an LB (LO) bound entry will revise one and only one bound. If the previous variable bound type (see Figure 78) defined two bounds (FX, E, FR, N, G, and L all define both a UB and an LB), a single UB or LB will change only one bound.

MERGE

The MERGE procedure forms a new disk problem file from problem file(s) previously stored on the disk. The MERGE procedure makes it possible to alter the names of row variables and/or column variables as they are combined to form the new disk problem file. This provides a useful model generation capability for many application areas. The source problem files (the problem files previously stored on the disk) are not altered by MERGE. The new MERGEd file can be used in every way as if it had been formed by unit record INPUT; it may be REVISEd, OPTIMIZEd, SOLVEd, MERGEd, DELETEd, etc.

The source problem files are designated by DATA control records. Each source problem DATA record can be preceded by a column variable mask COLS data control record, and/or by a row variable mask ROWS data control record. The column (row) variable name change is specified by a user-specified mask in a COLS (ROWS) data control record. The mask specifies the character(s) and the position(s) in the name field that the character(s) are to occupy. For example, a ROWS mask of bb1bbbbb will cause the row variable names originating from the next specified DATA to be identical to the row variable names in the source file, except that the third position of each row variable name will be 1, FE changed to FE1, COST changed to CO1T.

If the source problem files contain variables or sets (BOUNDS, RHS, RANGE) with the same name, the new file will contain the combined variable or set. If the same element is contained in more than one of the files, the element coefficient of the last source problem specified in the MERGE DATA records will be used; the others will be eliminated.

Data Control Records

MERGE reads a series of data control records.

1. The NAME data control record names the new problem file. The new problem name must be given in the second name field.

2.,3. COLS (ROWS) records are optional. These may be used to specify a variation in the column (row) variable names of the new problem from column (row) variables originating in the next source DATA file. The name variation is specified by a mask in the second name field. The COLS (ROWS) variation applies only to the next source DATA file.

4. DATA records designate the source problem file. The second name field specifies the name of the source problem. No variation of names is required; COLS and/or ROWS variations may be specified.

5. ENDATA indicator must follow the last DATA record.

IF the data control records include no NAME, the new problem file will be assigned a name of eight blanks. If there are multiple NAMEs, the last NAME will be used. Any problem file with the same NAME is automatically DELETEd.

It is recommended that the user MERGE problems with like selection files (see Chapter 3). If all source problems have ROW selection files, the new problem will have a ROW selection file that combines the source problem ROW selection files. If some source problems do not have ROW selection files, and some do, the new problem ROW selection file will often contain only a subset of the problem desired, since the unselected row variables will be omitted. A similar comment applies to COLS selection files.

The following records result in the creation of a new file named MULTIBLE. The new file merges the data contained in LPSAMPLE and LINKEQ. Further, it will contain, instead of the original variable names in LPSAMPLE, new variable names in which row names containing the numeral 1 in the third position will be defined and row names containing the numeral 2 in the third position will be defined; similarly, the column names of LPSAMPLE will be altered and doubled in number through the agency of the template, which defines new names with A in the sixth position and new names with B in the sixth position.

```
MERGE
          NAME          MULTIBLE
          ROWS            1
          COLS               A
          DATA          LPSAMPLE
          DATA          LINKEQ
          ROWS            2
          COLS               B
          DATA          LPSAMPLE
ENDATA
```

The last MERGE control data record must be followed by an ENDATA record.

Output

After the data in the new problem file has been sorted and checked for duplicates, MERGE automatically calls STATISTICS to log a summary of the data in the new file. Figure 54 illustrates the problem coefficients and bounds generated in the formation of MULTIBLE.

Major Errors

If there is no problem file on the disk with a name specified on a data control record read by MERGE, a major error is recorded. The remaining MERGE data control records are processed normally.

If an invalid data control record is read, a major error is recorded and the record is treated as an ENDATA record.

BCDOUT

BCDOUT will write the data in a problem file on the PUNCH unit. The PUNCH unit is assigned during system generation and may be reassigned by MOVE. PUNCH may be assigned to PAPERTAPE, CARD, PRINTER, or TYPEWRITER. The problem file to be output must first have been retrieved by a MOVE procedure reading a DATA control data record naming the problem file. BCDOUT will output problem data only. INSERTed and SAVESOLUTION data will not be output. See PUNCH procedure for output of starting solution data.

Input

If the output of the BCDOUT procedure is to be in the form of punched cards, a sufficient number of blank cards must follow the procedure call in the hopper of the card reader.

```
MOVE
    DATA        MULTIBLE
    PUNCH       PRINTER
ENDATA
REVISE
ENDATA
PROBLEM 'MULTIBLE' CONTAINS
   26 ROWS
    0 SELECTED ROWS
   40 VARIABLES
    0 SELECTED COLUMNS
    1 BOUNDS
    0 RHS'S
    0 RANGES
  116 COLUMN ELEMENTS
    6 LOWER BOUND ELEMENTS
   19 UPPER BOUND ELEMENTS
    0 RHS ELEMENTS
    0 RANGE ELEMENTS


BCDOUT

NAME          MULTIBLE

COLUMNS

    CO1T       COTT         1.000000
    BIN.1A     CO1T         0.030000
    BIN.1A     WE1GHT       1.000000
    BIN.1A     FE1          0.150000
    BIN.1A     CU1          0.030000
    BIN.1A     MN1          0.020000
    BIN.1A     MG1          0.020000
    BIN.1A     AL1          0.700000
    BIN.1A     SI1          0.020000
    BIN.1A     BIN.1T       1.000000
    BIN.2A     CO1T         0.080000
    BIN.2A     WE1GHT       1.000000
    BIN.2A     FE1          0.040000
    BIN.2A     CU1          0.050000
    BIN.2A     MN1          0.040000
    BIN.2A     MG1          0.030000
    BIN.2A     AL1          0.750000
    BIN.2A     SI1          0.060000
    BIN.2A     BIN.2T       1.000000
    BIN.3A     CO1T         0.170000
    BIN.3A     WE1GHT       1.000000
    BIN.3A     FE1          0.020000
    BIN.3A     CU1          0.080000
    BIN.3A     MN1          0.010000
    BIN.3A     AL1          0.800000
    BIN.3A     SI1          0.080000
    BIN.3A     BIN.3T       1.000000
    BIN.4A     CO1T         0.120000
    BIN.4A     WE1GHT       1.000000
    BIN.4A     FE1          0.040000
    BIN.4A     CU1          0.020000
    BIN.4A     MN1          0.020000
    BIN.4A     AL1          0.750000
    BIN.4A     SI1          0.120000
    BIN.4A     BIN.4T       1.000000
    BIN.5A     CO1T         0.150000
    BIN.5A     WE1GHT       1.000000
    BIN.5A     FE1          0.020000
    BIN.5A     CU1          0.060000
    BIN.5A     MN1          0.020000
    BIN.5A     MG1          0.010000
    BIN.5A     AL1          0.800000
    BIN.5A     SI1          0.020000
    BIN.5A     BIN.5T       1.000000
    ALUMIAUM   CO1T         0.210000
    ALUMIAUM   WE1GHT       1.000000
    ALUMIAUM   FE1          0.010000
    ALUMIAUM   CU1          0.010000
    ALUMIAUM   AL1          0.970000
    ALUMIAUM   SI1          0.010000
    ALUMIAUM   ALUMITUM     1.000000
    SILICAN    CO1T         0.380000
    SILICAN    WE1GHT       1.000000
    SILICAN    FE1          0.030000
    SILICAN    SI1          0.970000
    SILICAN    SILICTN      1.000000
    FE1        BA1E         1.000000
    CU1        BA1E         1.000000
    CO2T       COTT         1.000000
    BIN.1B     CO2T         0.030000
    BIN.1B     WE2GHT       1.000000
    BIN.1B     FE2          0.150000
    BIN.1B     CU2          0.030000
    BIN.1B     MN2          0.020000
    BIN.1B     MG2          0.020000
    BIN.1B     AL2          0.700000
    BIN.1B     SI2          0.020000
    BIN.1B     BIN.1T       1.000000
    BIN.2B     CO2T         0.080000
    BIN.2B     WE2GHT       1.000000
    BIN.2B     FE2          0.040000
    BIN.2B     CU2          0.050000
    BIN.2B     MN2          0.040000
    BIN.2B     MG2          0.030000
    BIN.2B     AL2          0.750000
    BIN.2B     SI2          0.060000
    BIN.2B     BIN.2T       1.000000
    BIN.3B     CO2T         0.170000
    BIN.3B     WE2GHT       1.000000
    BIN.3B     FE2          0.020000
    BIN.3B     CU2          0.080000
    BIN.3B     MN2          0.010000
    BIN.3B     AL2          0.800000
    BIN.3B     SI2          0.080000
    BIN.3B     BIN.3T       1.000000
    BIN.4B     CO2T         0.120000
    BIN.4B     WE2GHT       1.000000
    BIN.4B     FE2          0.040000
    BIN.4B     CU2          0.020000
    BIN.4B     MN2          0.020000
    BIN.4B     AL2          0.750000
    BIN.4B     SI2          0.120000
    BIN.4B     BIN.4T       1.000000
    BIN.5B     CO2T         0.150000
    BIN.5B     WE2GHT       1.000000
    BIN.5B     FE2          0.020000
    BIN.5B     CU2          0.060000
    BIN.5B     MN2          0.020000
    BIN.5B     MG2          0.010000
    BIN.5B     AL2          0.800000
    BIN.5B     SI2          0.020000
    BIN.5B     BIN.5T       1.000000
    ALUMIBUM   CO2T         0.210000
    ALUMIBUM   WE2GHT       1.000000
    ALUMIBUM   FE2          0.010000
    ALUMIBUM   CU2          0.010000
    ALUMIBUM   AL2          0.970000
    ALUMIBUM   SI2          0.010000
    ALUMIBUM   ALUMITUM     1.000000
    SILICBN    CO2T         0.380000
    SILICBN    WE2GHT       1.000000
    SILICBN    FE2          0.030000
    SILICBN    SI2          0.970000
    SILICBN    SILICTN      1.000000
    FE2        BA2E         1.000000
    CU2        BA2E         1.000000

BOUNDS

 FX ALLOYMU    WE1GHT    2000.000000
 UP ALLOYMU    FE1         60.000000
 UP ALLOYMU    CU1        100.000000
 UP ALLOYMU    MN1         40.000000
 UP ALLOYMU    MG1         30.000000
 LO ALLOYMU    AL1       1500.000000
 LO ALLOYMU    SI1        250.000000
 UP ALLOYMU    SI1        300.000000
 UP ALLOYMU    BA1E       120.000000
 FX ALLOYMU    WE2GHT    2000.000000
 UP ALLOYMU    FE2         60.000000
 UP ALLOYMU    CU2        100.000000
 UP ALLOYMU    MN2         50.000000
 UP ALLOYMU    MG2         25.000000
 LO ALLOYMU    AL2       1500.000000
 LO ALLOYMU    SI2        300.000000
 UP ALLOYMU    SI2        350.000000
 UP ALLOYMU    BA2E       150.000000
 UP ALLOYMU    BIN.1T     200.000000
 UP ALLOYMU    BIN.2T     750.000000
 UP ALLOYMU    BIN.3T     800.000000
 UP ALLOYMU    BIN.4T     700.000000
 UP ALLOYMU    BIN.5T    1500.000000

ENDATA
```

Figure 54. MULTIBLE problem data generated by MERGE,output by the BCDOUT procedure

Output

The PUNCH unit can be CARD, PRINTER, PAPER-TAPE, or TYPEWRITER. PUNCH assignment can be made by a data control record read by MOVE before BCDOUT. If PRINTER output is desired, the following records will accomplish that end:

```
MOVE
      DATA          MULTIBLE
      PUNCH         PRINTER
ENDATA
BCDOUT
```

If a deck of punched cards is desired, the following records are appropriate:

```
MOVE
      DATA          MULTIBLE
      PUNCH         CARD
ENDATA
BCDOUT
```

If the output is on cards or paper tape, the BCDOUT output deck can be used as input data to be read by INPUT. Figure 54 illustrates the output produced by BCDOUT on the printer for the problem file MULTIBLE.

Major Errors

If the output is on cards, BCDOUT reads the next card in the deck being read and checks that the first 36 positions of the card are blank. If they are not blank, a major error is recorded and the procedure is terminated.

DELETE

DELETE is used to remove one or more obsolete problem files from the disk. DATA control records designate the obsolete problem(s). The disk space previously occupied by the deleted files can be made available for further use by the EDIT procedure.

Data Control

The DELETE procedure deletes from the disk the problem files named on DATA control records which follow it:

```
DELETE
         DATA          LPSAMPLE

ENDATA
```

Since DELETE may read any number of data control records, the last DATA record must be followed by an ENDATA indicator record.

Output

Each DATA control record and the ENDATA record are logged.

Major Errors

If an invalid data control record is found, a major error is recorded. The remainder of the data control records are processed normally.

EDIT

EDIT packs the current problem files into consecutive disk sectors, thus freeing disk space previously used by deleted problems.

Output

EDIT automatically calls the DICTIONARY procedure after it has packed the problem data, thus producing the DICTIONARY output report.

DICTIONARY

DICTIONARY lists the names of the problem files stored on the disk, the number of disk sectors used by each one, and the location number of the next disk sector available for use.

Output

The DICTIONARY report contains a heading line, one line for each problem file stored on the disk, and a line showing the location number of the next disk sector available for use. Figure 55 illustrates the output produced by DICTIONARY.

```
PROBLEM NAME    NO. RECORDS

 LPSAMPLE             3


NEXT AVAILABLE SECTOR IS      9
```

Figure 55. DICTIONARY output

PROCESSING SPECIFICATIONS

MOVE

The MOVE procedure is used to reset or alter processing parameters for problem definition, device assignment, and LPPARAMETRIC and OPTIMIZE parameters. MOVE reads data control records which specify the parameter changes for the current computation. If the record is being typed, a complete record name must be typed, including trailing blanks in the second name field if necessary, before pressing the EOF key to end the record. For example,

bbbbDATAbbbbbbFILENMbb

would retrieve the problem file FILENM for use by OPTIMIZE and the other LPS procedures. The types of data control record (positions 5-12) and their functions are discussed in this section.

Problem Definition

These records are used to retrieve a problem file and define the problem to be optimized — the objective, bound set to be used, etc. The name of the problem file, bound set, etc., appears in positions 15-22 of the record.

- DATA — retrieves the problem file named in positions 15-22 of the record and makes its problem data available to the other LPS procedures. The problem file to be used in an optimization or with REVISE or BCDOUT must be called before any other procedures may be called (except when using RESTART). Once called, a problem file need not be called again except after INPUT, MERGE, EDIT, DICTIONARY, or INSERT. The INSERT, INPUT, REVISE, MERGE, DICTIONARY, and EDIT procedures, and the DATA control record in MOVE, reset the objective, bounds, RHS, and ranges set names. The user must MOVE the appropriate names after these procedures, or following the MOVE DATA control record.

- MINIMIZE (MAXIMIZE) — names the variable to be minimized or maximized in an optimization. The name of the variable appears in positions 15-22.

- BOUNDS — names the bound set to be used in an optimization. The name of the bound set appears in positions 15-22. If, in a series of optimizations using the same problem file, a bound set is used for one optimization but no bound set is to be used for the next, this must be signified by a BOUNDS data control record containing **NONE** in positions 15-22.

- RHS — names the RHS set to be used in an optimization. Its use is exactly the same as a BOUNDS data control record.

- RANGES — names the ranges set to be used in an optimization. Its use is exactly the same as a BOUNDS data control record.

Device Assignment

These data control records are used to change the physical I/O devices assigned to the I/O functions. The initial setting of the I/O units is done by the operator (see the Application Directory) during system generation. The name of the I/O device assigned to a particular function appears in positions 15-24 of the record. The I/O device name must be left-justified, with trailing blanks, if required, to complete the device name.

- INPUT — the device to be used to read input data records (by INPUT, REVISE, and INSERT). May be CARD or PAPERTAPE.

- CONTROL — the device to be used to read control records (procedure call and data control records). May be CARD, PAPERTAPE, or TYPEWRITER.

- LOG — the device upon which control records, OPTIMIZE iteration log, DICTIONARY report, and procedure messages are printed. May be PRINTER, TYPEWRITER, or **NONE** if no log or error messages are required.

- REPORT — the device used to output the LPSOLUTION, LPANALYSIS, and LPPARAMETRIC reports. May be PRINTER, TYPEWRITER, CARD, or PAPERTAPE.

- PUNCH — the device used to output unit record files by BCDOUT and PUNCH. May be CARD, PAPERTAPE, PRINTER, or TYPEWRITER.

LPPARAMETRIC Parameters

These records are used to set parameters used to control the LPPARAMETRIC procedure. In each case the word PARAMET must appear in positions 5-11 of the data control record, the word REPORTS or INTERVAL left-justified in positions 15-22, and the value of the parameter in positions 25-36. Initial values are provided for LPPARAMETRIC and OPTIMIZE parameters. The initial values will be used unless user-assigned values are MOVEd.

- REPORTS — sets the parameter which controls the number of reports printed during an LPPARAMETRIC run to the integer part of the number in the value field. The sign of the number is ignored. If this parameter is not MOVEd, the initial value will be used. This parameter is initially set to 5.

- INTERVAL — sets the interval parameter of LPPARAMETRIC to the number contained in the value field. This parameter is initially set to 1.0.

Optimization Parameters

All of the parameters are initialized by the system. Only resetting MAXIMUM ITERATIONS and MAXIMUM PRICING will generally be useful, even to the experienced LP user. Resetting the other parameters may cause serious processing difficulties and should be done with great care only by the experienced 1130 LPS user.

- BETWEEN INVERTS — sets the parameter which controls the number of minor iterations between inversions. This will cause reinversion at the first major iteration when the number of minor iterations exceeds the specified number. Inversions are also called for by OPTIMIZE when an internally calculated processing error is exceeded. Too low a value may result in failure to complete the optimization; too high a value may result in extremely long processing times (or may have little or no effect, depending on internal processing error calculation inversion checks). This parameter is initially set to 40.

- AFTER INVERT — sets the parameter which controls the number of iterations allowed since invert when an optimum solution is found. When this number is exceeded, reinversion occurs and optimization continues. Too low a value may result in failure to complete the optimization process; too high a value may result in processing error. This parameter is initially set to 30.

- MAXIMUM ITERATIONS — is used to limit the number of iterations that may be performed for this DATA (since this DATA was MOVEd and including all subsequent optimizations). This parameter is sometimes set to a small number (for example, 5) for the first, partial optimization of a large, new, or complex model. This checkout run of the model will test the procedure sequence and, if the run includes an LPSOLUTION report, may also indicate data and/or formulation errors. The MAXIMUM ITERATIONS are checked at the end of each major iteration, and the optimization will end when the number of iterations taken is equal to or exceeds the maximum. This parameter is initially set to 32767.

- MAXIMUM PRICING — governs the size of the subset of variables chosen in each major iteration. This parameter is effective only if it is less than the limitation imposed by available storage. The subset size will be the smaller of MAXIMUM PRICING and 747/(ROW VARIABLES + 8). Since this parameter is initially set to 32767, the real limit will be imposed by the problem size unless the MAXIMUM PRICING is set by the user.

- TOLERANCE FEASIBLE — sets the allowable margin for solution activity bound match. A variable will be considered to be feasible if the computed activity does not exceed either bound by the feasibility tolerance. Too large a value may lead to unrecognized infeasible solutions (the solution activity may be too far below the lower bound, or too far above the upper bound). Too small a value may lead to unnecessary iterations or to failure to find a solution when an acceptable solution exists. This parameter is initially set to .0001 and applies to the internally computed and scaled activities. Use extra caution, therefore, when increasing this value.

- TOLERANCE PIVOT — sets the pivot tolerance. Too large a value may lead to a false unbounded solution. Too small a value may lead to computational error. This parameter is initially set to .0005.

- TOLERANCE ELEMENT — sets the threshold for computed elements. A computed element is set to zero if its magnitude is less than the element tolerance. Too small a value may lead to unnecessary computation time. Too large a value may lead to computation error. This parameter is initially set to .000 000 000 5.

- TOLERANCE SCREEN — sets the scaled input data threshold. After the input data has been scaled, any number whose magnitude is below this tolerance is ignored. Too high a screen tolerance may result in incorrect solution. Too low a screen tolerance may result in computation error and/or excessive computation time. This tolerance is initially set to the value .001.

- TOLERANCE ERROR — sets the optimization process error threshold. During optimization, pricing checks are made. When the computation error exceeds the error tolerance, a reinversion is performed. If, after a reinversion, the computational error is still greater than the error tolerance, the optimization is abandoned. Too high an error tolerance can result in failure to detect computation errors and may result in invalid results. Too low an error tolerance can result in failure to continue optimization. This tolerance is initially set to the value .01.

PROBLEM SOLUTION

OPTIMIZE

OPTIMIZE finds the solution to an LP problem. The problem to be solved is defined by MOVE data control records naming the problem file to be used, the objective variable of the problem, and the BOUNDS, RHS, and RANGE sets to be used, if any.

When OPTIMIZE is first called following a MOVE, DATA, it scales the problem data to provide greater processing accuracy. OPTIMIZE will automatically generate a somewhat arbitrary set of starting solution levels or status to begin optimization except when (1) a starting solution is RESTOREd (see "Starting Solutions", later in this chapter), or (2) a second or subsequent optimization is performed without an intervening MOVE, DATA.

The procedure will call invert to compute the solution activities at the beginning of the first optimization. Then the procedure performs a number of minor iterations (changes in solution) using each minor iteration as a base to find an improved solution. Each major iteration is a selection of a subset of the problem variables for suboptimization by the minor iterations. OPTIMIZE periodically calls for new inversions in the course of solution, to improve processing accuracy, or to reduce the size of the computation files. The optimization is complete when no subset of variables can be found which may improve the current solution.

Data Control

The optimize problem definition and processing parameters are specified by MOVE data control records.

Problem Definition

- DATA — names and retrieves the problem file containing the data to be used.

- MINIMIZE (or MAXIMIZE) — names the variable whose solution value is to be minimized (or maximized).

- BOUNDS — names the bound set to be used, if any.

- RHS — names the right-hand-side set to be used, if any.

- RANGE — names the range set to be used, if any.

The DATA and MINIMIZE (or MAXIMIZE) records are always required to begin a series of optimizations. The remaining records are used only if a corresponding set is to be used. Thus, if only a bound set is defined, only a BOUND record is required. No RHS or RANGE record need be provided.

If several optimizations are to be made using the same problem file, only changes of objective, BOUNDS, or RHS or RANGE should be specified by MOVE control records.

A typical set of records designed to optimize an LP problem using bounds set ALLOY1, and then using bound set ALLOY2, would appear as follows:

```
MOVE
      DATA            LPSAMPLE
      MINIMIZE        COST
      BOUNDS          ALLOY1
ENDATA
OPTIMIZE
LPSOLUTION
MOVE
      BOUNDS          ALLOY2
ENDATA
OPTIMIZE
```

Optional Control of the Optimization Process

The parameter settings are initialized by the system. Only the settings MAXIMUM ITERATIONS and MAXIMUM PRICING will generally be useful, even to the experienced LP user. Resetting the other options may cause serious processing difficulties and should be used with great care only by the experienced 1130 LPS user.

- MAXIMUM ITERATIONS — is used to limit the number of iterations that may be performed for this DATA (since this DATA was MOVEd and including all subsequent optimizations). This parameter is sometimes set to a small number (for example, 5) for the first, partial optimization of a large, new, or complex model. This checkout run of the model will test the procedure sequence and, if the run includes an LPSOLUTION report, may also indicate data and/or formulation errors. The MAXIMUM ITERATIONS are checked at the end of each major iteration, and the optimization will end when the number of iterations taken is equal to or exceeds the maximum. This parameter is initially set to 32767.

- MAXIMUM PRICING — governs the size of the subset of variables chosen in each major iteration. This parameter is effective only if it is less than the limitation imposed by available storage. The subset size will be the smaller of MAXIMUM PRICING and 747/(ROW VARIABLES + 8). Since this parameter is initially set to 32767, the real limit will be imposed by the problem size unless the MAXIMUM PRICING is set by the user.

- Others — The remaining options have been discussed under "MOVE", earlier in this chapter.

Output

While it is performing iterations to find the optimal solution to a problem, OPTIMIZE prints an iteration log, on the LOG device, to show how the optimization is progressing. When OPTIMIZE has found the optimal solution or terminates its operation for some other reason, it logs a message stating the reason for its termination.

The iteration log prints two heading lines, then one line for each major iteration — that is, each updating of a group of variables to the current solution for possible change in solution status. Figure 56 shows an example of an iteration log. The headings are:

ITERATION NUMBER	— The number of minor iterations so far. A line repeating the current iteration number is printed after an inversion.
INFEASIBILITY COUNT	— The number of variables whose solution value exceeds the bounds on the variable.
VALUE OF 'AAAAAAAA'	— The current value of the objective variable. The name of the variable is printed between the quotation marks in the second line.

Possible termination messages are:

- SOLUTION OPTIMUM. — An optimum solution to the problem has been found.

- SOLUTION INFEASIBLE. — No solution can be found in which the values of all the problem variables lie within their bounds. The infeasible variables are indicated by messages on the LPSOLUTION report.

- SOLUTION UNBOUNDED. — The objective variable may be decreased (MINIMIZE) or increased (MAXIMIZE) without limit. The variable causing this unboundedness is indicated by a message on the LPSOLUTION report.

- INCOMPLETE OPTIMIZATION. — The maximum numbers of iterations allowed have been taken without finding an optimal solution.

- MAXIMUM PROCESSING ERROR XXXXXX'XXXXXX. — A computational error has been found that is greater than the error tolerance. The maximum error found is printed. The problem should be examined to attempt to reduce greatly the number of elements. A reduction in the number of row variables is also desirable.

Major Errors

If there is no problem data in the problem file retrieved by the MOVE DATA record, or if no file has been retrieved, an error message is logged and a major error recorded. If this condition occurs, OPTIMIZE returns control to the LP-MOSS Monitor regardless whether OPTIMIZE was called by the user, by LPSOLUTION, or by any other procedure.

```
ITERATION  INFEASIBILITY  VALUE OF
 NUMBER        COUNT      'COST    '
    0            2          487.761
    8            0          301.434
SOLUTION  OPTIMUM
```

Figure 56. Iteration log

LPSOLUTION

LPSOLUTION automatically calls OPTIMIZE if the latter has not already been called. At the end of OPTIMIZE, LPSOLUTION prints a report listing the solution values for all the variables in the problem, as well as additional information useful for postoptimal analysis.

Data Control

Before LPSOLUTION can be called, the parameters of the problem must be set by a MOVE procedure call and the appropriate data control records (see "OPTIMIZE", just preceding this section).

Output

The LPSOLUTION report is a tabular representation of the solution containing two heading lines and one line for each variable in the problem. Figure 57 illustrates the report produced by LPSOLUTION for the problem LPSAMPLE, and should be interpreted as follows:

1. VARIABLE — the name of each variable in the problem.
2. TYPE — the status of each variable in the solution, in terms of its bounds. One of the following status codes will appear:

   LL (variable is at lower limit)
   UL (variable is at upper limit)
   EQ (variable has a fixed value)
   FR (a free variable is at zero level)
   B* (variable is at an intermediate level)
3. ENTRIES — the number of equation elements input for each variable, very useful for checking data.
4. SOLUTION ACTIVITY — the value of each variable in the current solution.
5. UPPER BOUND — the effective upper bound defined for each variable value in the input data. The effective upper bound may be due to STANDARD BOUNDS, BOUNDS set, RHS, or RANGE set.
6. LOWER BOUND — the effective lower bound defined for each variable value in the input data.
7. CURRENT COST — the coefficient of each variable in the equation defining the objective variable, if the objective variable is a row variable. It is zero if the objective variable is a column variable.
8. REDUCED COST — for variables in the solution at a <u>bound</u>, a value indicating what change per unit measure of the objective variable value would result if the bound on this specific variable were relaxed. The reduced-cost value is valid only in the neighborhood of the solution, but provides a good indication of the cost of inventory limitations and specification constraints.

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	301.434	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.242
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.288
BIN.2	B*	8	606.508	750.000	0.000	0.079	0.000
BIN.3	LL	7	0.000	800.000	0.000	0.170	-0.001
BIN.4	B*	7	554.733	700.000	0.000	0.120	0.000
BIN.5	B*	8	232.248	1500.000	0.000	0.150	0.000
ALUMINUM	B*	6	464.497	***********	0.000	0.210	0.000
SILICON	B*	4	142.011	***********	0.000	0.379	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-2.951
CU	B*	1	60.000	100.000	0.000	0.000	0.000
MN	UL	0	40.000	40.000	0.000	0.000	-0.908
MG	B*	0	20.517	30.000	0.000	0.000	0.000
AL	B*	0	1507.292	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.241
BASE	UL	0	120.000	120.000	0.000	0.000	-0.245

Figure 57. LPSOLUTION report

If the solution is found to be unbounded, the line

VARIABLE CAUSES UNBOUNDED SOLUTION

is printed after the variable that caused the unbounded solution. If the problem is found to be infeasible, messages are printed after each infeasible variable. These messages are:

- INFEASIBLE**UPPER BOUND. — The solution value of the variable exceeds its upper bound.

- INFEASIBLE**LOWER BOUND. — The solution value of the variable exceeds its lower bound.

- INFEASIBLE**XXXXXXXXX.XXX ARTIFICIAL. — An artificial variable, generated during the optimization process, is at the nonzero level shown. This also indicates a genuine infeasibility.*

The number of infeasibilities is printed as the last line of the report:

********** PROBLEM CONTAINS XXXXX INFEASIBILITIES

RESTART

If, during OPTIMIZEation of a problem, a computer malfunction occurs or the computer has to be used for other purposes, the RESTART procedure allows processing to be continued from the point of processing interrupt. For example, if OPTIMIZE is interrupted at some time after the LOG headings have been printed out, RESTART reinitializes core so that the problem can be restarted from approximately the point of interrupt, as long as the 1130 LPS disk has not been used for another job. The user must use an OPTIMIZE, LPSOLUTION, or LPANALYSIS procedure call record following RESTART. Operating instructions are given under "RESTART Optimization Operating Procedure".

Output

RESTART logs the name of the problem file, as illustrated in Figure 58.

```
RESTART
    DATA        LPSAMPLE
```

Figure 58. RESTART log

*This variable, in effect, transforms the original statement of the problem to be RV-ARTIFICIAL= ACV1+BCV2+..., where RV is the row variable preceding the message.

POSTOPTIMAL ANALYSIS

LPANALYSIS

After the optimal solution is found, LPANALYSIS prints a report in two parts. The first part of the report lists all the variables in the solution at a bound, and indicates for each variable the type of bound (upper, lower, or fixed), the solution activity, the current cost, the bound values, the per-unit additional cost produced by increasing or decreasing the activity of each variable by one unit, the cost range for each variable at which its activity in the final solution remains unchanged, and the new activity of each variable at the point where that cost range is exceeded. The second part of the LPANALYSIS report lists the same information for all variables in the solution at an intermediate level.

Output

The LPANALYSIS report is a tabular representation of the effect of price changes on variable activity and the effect of variable activity changes on price. Figures 59 and 60 illustrate, respectively, the first and second part of the report produced by LPANALYSIS for the problem LPSAMPLE. Each part of the report contains two rows of headings, and each variable in the report occupies two rows as well. The upper heading identifies the information contained in the first row associated with each variable, and the lower heading identifies the information contained in the second row associated with each variable. The report should be interpreted as follows:

1. VARIABLE — the name of each variable in the solution at a bound (Figure 59) and the name of each variable in the solution at an intermediate level (Figure 60). In other respects the two parts of the report provide the same type of data.
2. TYPE — the status of each variable in terms of its bounds. The status codes used are as follows:

    LL (variable is at lower limit)
    UL (variable is at upper limit)
    EQ (variable is a fixed variable)
    FR (variable is a free variable at 0 level)
    B* (variable is at intermediate level)

3. SOLUTION ACTIVITY — the value of the variable in the current solution.
4. CURRENT COST — the current cost per unit of the variable.
5. UPPER BOUND — the upper bound on the value that the variable may take.
6. LOWER BOUND — the lower bound on the value that the variable may take.

-------------------VARIABLES AT UPPER BOUND OR LOWER BOUND-------------------

VARIABLE	SOLUTION ACTIVITY	UPPER BOUND	COST/UNIT INCREASE	INCREASED ACTIVITY	LOWEST COST
TYPE	CURRENT COST	LOWER BOUND	COST/UNIT DECREASE	DECREASED ACTIVITY	HIGHEST COST
WEIGHT	2000.000	2000.000	0.242	2179.629	-0.242
EQ	0.000	2000.000	-0.242	1993.060	***********
BIN.1	0.000	200.000	0.288	22.319	-0.258
LL	0.030	0.000	-0.288	-13.678	***********
BIN.3	0.000	800.000	0.001	100.512	0.168
LL	0.170	0.000	-0.001	-52.513	***********
FE	60.000	60.000	-2.951	62.033	***********
UL	0.000	0.000	2.951	55.198	2.951
MN	40.000	40.000	-0.908	41.579	***********
UL	0.000	0.000	0.908	38.059	0.908
SI	250.000	300.000	0.241	257.596	-0.241
LL	0.000	250.000	-0.241	160.727	***********
BASE	120.000	120.000	-0.245	122.318	***********
UL	0.000	0.000	0.245	114.360	0.245

Figure 59. LPANALYSIS variables at bounds

----------------------VARIABLES AT INTERMEDIATE LEVEL--------------------

VARIABLE	SOLUTION ACTIVITY	UPPER BOUND	COST/UNIT INCREASE	INCREASED ACTIVITY	LOWEST COST
TYPE	CURRENT COST	LOWER BOUND	COST/UNIT DECREASE	DECREASED ACTIVITY	HIGHEST COST
COST	301.434	***********	***********	301.434	***********
B*	-1.000	***********	***********	301.434	***********
BIN.2	606.508	750.000	0.000	743.597	0.079
B*	0.079	0.000	0.010	430.237	0.090
BIN.4	554.733	700.000	0.012	851.851	0.107
B*	0.120	0.000	0.001	462.548	0.121
BIN.5	232.248	1500.000	0.015	342.490	0.134
B*	0.150	0.000	0.000	-10.845	0.150
ALUMINUM	464.497	***********	0.001	552.816	0.208
B*	0.210	0.000	0.024	393.873	0.234
SILICON	142.011	***********	0.203	151.001	0.176
B*	0.379	0.000	0.091	140.524	0.471
CU	60.000	100.000	2.951	64.801	-2.951
B*	0.000	0.000	0.245	54.360	0.245
MG	20.517	30.000	0.076	22.307	-0.076
B*	0.000	0.000	0.421	16.332	0.421
AL	1507.292	***********	0.009	1521.251	-0.009
B*	0.000	1500.000	0.251	1459.289	0.251

Figure 60. LPANALYSIS variables at intermediate levels

7. COST/UNIT INCREASE — the per-unit additional cost produced by increasing the activity of the variable by one unit, valid up to the activity shown under INCREASED ACTIVITY.

8. COST/UNIT DECREASE — the per-unit additional cost produced by decreasing the activity of the variable by one unit, valid down to the activity shown under DECREASED ACTIVITY.

9. INCREASED ACTIVITY — the activity level of the variable which would be produced by a change in its cost to the LOWEST COST.

10. DECREASED ACTIVITY — the activity level of the variable which would be produced by a change in its cost to the HIGHEST COST.

11. LOWEST COST & HIGHEST COST — the cost range within which the activity level of the variable remains unchanged. When the variable cost falls below the LOWEST COST, its activity increases to the value listed under INCREASED ACTIVITY. When the variable cost exceeds HIGHEST COST, its activity decreases to the value listed under DECREASED ACTIVITY. The values listed under INCREASED ACTIVITY and DECREASED ACTIVITY are valid only in the neighborhood of the values listed under LOWEST COST and HIGHEST COST.

If the solution is infeasible or unbounded, LPANALYSIS will call LPSOLUTION in order to avoid using machine time to produce an essentially meaningless LPANALYSIS report.

LPPARAMETRIC

The LPPARAMETRIC procedure performs a series of optimizations on successive variations of the problem data. LPPARAMETRIC can be used to determine the effect on the problem solution activities due to cumulative changes in the formulation data. This procedure requires a disk-stored parametric data problem file in addition to the problem formulation data file. An LPSOLUTION report is printed for each variation of the problem data. The problem data and the parametric files are not altered by this procedure.

Parametric Data

The parametric data file specifies the change data, which may include equation elements, bound entries, RHS elements, and RANGE elements. The parametric file must be stored before problem optimization.

Data Control

A DATA control record must follow the LPPARAMETRIC procedure call to specify the parametric problem file. No ENDATA indicator is required.

Optional MOVE Data Control

PARAMET REPORTS can be used to specify the number of successive variations for which LPSOLUTION reports are required. Since the number of reports is initialized to 5., five reports will be output unless otherwise specified by the user.

PARAMET INTERVAL provides a means of altering the scale of the change data so that the same parametric data can be used for both gross-scale and fine-scale parametric investigations. The interval is initialized to 1., which will not scale the parametric data used. An INTERVAL of 2.0 would have the same effect as a multiplication of every coefficient (including bounds, RHS, ranges) in the parametric data by 2.0 for a more gross-scale investigation. An INTERVAL of .25 would similarly leave the same effect as a multiplication of every coefficient in the parametric data by .25 for a finer-scale study.

Output

LPPARAMETRIC will produce five LPSOLUTION reports unless a PARAMET REPORTS has been MOVEd to set a different value. Each report will reflect the effect of the cumulative changes on the problem data by the parametric data. The DATA control record is logged.

Figure 61 shows the desired parametric changes for an LPSAMPLE problem run. Figure 62 is a listing of the data, procedure calls, and data control records for the LPPARAMETRIC study. Figure 63 is the solution report at the first level of data change. Figure 64 is the solution report at the second level or interval. Figure 65 is the solution report at the third level.

```
***********************************
*        *      *       *         *
* REPORT * BIN.2* BIN.5 * SILICON *
*        * COST * COST  * COST    *
*        *      *       *         *
***********************************
***********************************
*    1   * .09  *  .16  *  .36    *
*    2   * .10  *  .17  *  .34    *
*    3   * .11  *  .18  *  .32    *
***********************************
```

Figure 61. Parametric cost changes

```
INPUT
NAME            PARCOST
    BIN.2       COST            0.01
    BIN.5       COST            0.01
    SILICON     COST      -     0.02
ENDATA
MOVE
    DATA        LPSAMPLE
    MINIMIZE    COST
    BOUNDS      ALLOY1
ENDATA
LPSOLUTION
MOVE
    PARAMET     REPORTS         3.0
ENDATA
LPPARAMETRIC
    DATA        PARCOST
```

Figure 62. Parametric cost setup and input data

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	306.197	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.244
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.312
BIN.2	B*	8	743.597	750.000	0.000	0.089	0.000
BIN.3	B*	7	100.512	800.000	0.000	0.170	0.000
BIN.4	B*	7	462.548	700.000	0.000	0.120	0.000
BIN.5	LL	8	0.000	1500.000	0.000	0.160	-0.003
ALUMINUM	B*	6	552.816	***********	0.000	0.210	0.000
SILICON	B*	4	140.524	***********	0.000	0.360	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-3.197
CU	B*	1	60.000	100.000	0.000	0.000	0.000
MN	UL	0	40.000	40.000	0.000	0.000	-0.464
MG	B*	0	22.307	30.000	0.000	0.000	0.000
AL	B*	0	1521.251	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.225
BASE	UL	0	120.000	120.000	0.000	0.000	-0.239

Figure 63. LPPARAMETRIC cost change — first report

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	310.767	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.247
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.337
BIN.2	B*	8	483.476	750.000	0.000	0.099	0.000
BIN.3	B*	7	212.117	800.000	0.000	0.170	0.000
BIN.4	UL	7	700.000	700.000	0.000	0.120	-0.000
BIN.5	LL	8	0.000	1500.000	0.000	0.170	-0.007
ALUMINUM	B*	6	485.679	***********	0.000	0.210	0.000
SILICON	B*	4	118.727	***********	0.000	0.340	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-3.418
CU	B*	1	60.000	100.000	0.000	0.000	0.000
MN	B*	0	35.460	40.000	0.000	0.000	0.000
MG	B*	0	14.504	30.000	0.000	0.000	0.000
AL	B*	0	1528.410	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.209
BASE	UL	0	120.000	120.000	0.000	0.000	-0.255

Figure 64. LPPARAMETRIC cost change – second report

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
COST	B*	0	313.227	***********	***********	-1.000	1.000
WEIGHT	EQ	0	2000.000	2000.000	2000.000	0.000	-0.243
BIN.1	LL	8	0.000	200.000	0.000	0.030	-0.278
BIN.2	B*	8	483.476	750.000	0.000	0.109	0.000
BIN.3	B*	7	212.117	800.000	0.000	0.170	0.000
BIN.4	UL	7	700.000	700.000	0.000	0.120	-0.009
BIN.5	LL	8	0.000	1500.000	0.000	0.180	-0.015
ALUMINUM	B*	6	485.679	***********	0.000	0.210	0.000
SILICON	B*	4	118.727	***********	0.000	0.320	0.000
FE	UL	1	60.000	60.000	0.000	0.000	-2.963
CU	B*	1	60.000	100.000	0.000	0.000	0.000
MN	B*	0	35.460	40.000	0.000	0.000	0.000
MG	B*	0	14.504	30.000	0.000	0.000	0.000
AL	B*	0	1528.410	***********	1500.000	0.000	0.000
SI	LL	0	250.000	300.000	250.000	0.000	-0.179
BASE	UL	0	120.000	120.000	0.000	0.000	-0.286

Figure 65. LPPARAMETRIC cost change – third report

STARTING SOLUTIONS

SAVESOLUTION

SAVESOLUTION saves the current status of the current optimization problem variables in a disk problem file. The saved solution can be used as an advanced starting solution for subsequent optimizations of this problem, or for a subsequent optimization of a different problem.

Data Control

SAVESOLUTION reads a NAME data control record naming the problem file into which the solution is to be saved. If the named problem file is already on the disk, the solution is saved into it; if not, a new problem file is created. Typical records that save a current solution would appear as follows:

```
SAVESOLUTION
     NAME            LPSOL
```

Since SAVESOLUTION reads only one data control record, no ENDATA indicator is required.

Output

The NAME data control record is logged.

Major Errors

If a NAME data control record does not immediately follow the SAVESOLUTION call, a major error is recorded and the procedure is terminated. If there is no current solution — that is, if neither OPTIMIZE nor RESTORE has been called for this problem — a major error is recorded and the procedure terminated.

PUNCH

PUNCH saves the current status of the current optimization problem variables in a unit record (PUNCH device) file. The PUNCH procedure output can later be INSERTed into a new or existing disk problem file, and can then be used as an advanced starting solution for this problem, or for a subsequent optimization of a different problem.

Output

The first record of the output deck produced by the PUNCH procedure is a NAME indicator record containing the name of the current problem. The last record of the deck is an ENDATA indicator. The remaining records are status records in the format required by INSERT (see "INSERT Data" in Chapter 3). Status records are output only for variables that are not at standard initial status. Hence:

1. row variables at upper or lower levels are output (standard initial — intermediate level),
2. column variables at intermediate level (standard initial — at lower level),
3. column variables at upper level.

Major Errors

If a PUNCH output is to be on cards, PUNCH reads the next card in the hopper and checks that the first 36 positions are blank. If they are not, a major error is recorded and the procedure is terminated. If there is no current solution — that is, if neither OPTIMIZE nor RESTORE has been called for this problem, a major error is recorded and the procedure is terminated.

INSERT

INSERT reads a unit record PUNCH file and stores the status of the variables in a NAMEd disk problem file. The INSERTed disk file can be used to provide an advanced starting solution in subsequent optimizations.

Input Data

The INSERT input data deck must be as described under "INSERT Data" in Chapter 3. The solution is stored in the problem file named by a NAME indicator record, which should be the first record of the data deck. If no problem file with this name is stored on the disk, one is created. If the data deck does not contain a NAME indicator, the name is assumed to consist of eight blanks. If the deck contains more than one NAME indicator, the last one read is used.

Output

The ENDATA record is logged.

Major Errors

If an indicator other than NAME or ENDATA is found in the deck, a major error is recorded and the indicator is treated as an ENDATA record.

RESTORE

RESTORE applies the status of the variables as stored in a problem file by SAVESOLUTION or INSERT. If the status of a variable is not specified in the file, the standard starting status is used:

1. Intermediate level, for row variables
2. Lower level, if the variable is a column variable with a finite lower bound
3. Upper level, otherwise

Data Control

RESTORE reads a DATA record naming the problem file containing the problem solution to be used. Figure 66 illustrates RESTORE used in conjunction with SAVESOLUTION. Figure 67 illustrates RESTORE used in conjunction with INSERT. Since RESTORE reads only one data control record, no ENDATA indicator is required.

Output

The DATA record is logged.

Major Errors

If a DATA record does not immediately follow the RESTORE call, a major error is recorded and the procedure is terminated.

```
****************************************
MOVE
     DATA         LPSAMPLE
ENDATA
REVISE
******************************
*                            *
* MAINLY COEFFICIENT,BOUND   *
*         CHANGES            *
*                            *
*                            *
******************************
MOVE
     BOUNDS       ALLOY1
     MINIMIZE     COST
ENDATA
RESTORE
     DATA         LPSAMPLE
LPSOLUTION
SAVESOLUTION
     NAME         LPSAMPLE
****************************************
```

Figure 66. Use of SAVESOLUTION and RESTORE. This illustrates a subsequent optimization of the model. During the first optimization, the data would be INPUTted and (usually) no advance solution would be available for RESTOREation.

```
****************************************
INPUT
******************************
*                            *
*                            *
*   DATA IN UNIT RECORD FORM *
*                            *
*                            *
******************************
INSERT
******************************
*                            *
*   PREVIOUS SOLUTION RUN    *
*         LEVELS             *
*                            *
******************************
MOVE
     DATA         LARGEPRB
     BOUNDS       LIMITSPB
     MAXIMIZE     PROFIT
ENDATA
RESTORE
     DATA         LARGEPRB
LPSOLUTION
PUNCH
******************************
*                            *
*   BLANK CARDS FOR NEW      *
*   SOLUTION LEVELS          *
*                            *
******************************
DELETE
     DATA         LARGEPRB
ENDATA
EDIT
****************************************
```

Figure 67. Use of INSERT, RESTORE, and PUNCH (subsequent optimization)

IF

The IF procedure provides a conditional bypass of part or all of the remaining procedure sequence. The IF procedure reads a data control record that specifies a condition and a search label.

If the condition is untrue, the next data control record is read, the IF procedure is terminated by an ENDATA indicator following the last condition-label data control record.

If the condition is true, the procedure begins a search for the label record on the CONTROL device. The search and the procedure are completed when the label record or an END call record is read. Control is returned to the LP-MOSS Monitor, which reads the procedure call record following the label record.

Data Control

See "IFNOT".

Label Records

See "IFNOT".

IFNOT

The IFNOT procedure provides a conditional bypass of part or all of the remaining procedure sequence. IF a condition is NOT true, the IFNOT procedure reads a condition-label data control record.

IF the condition specified by the data control record is NOT untrue, the next data control record is used. This procedure is also terminated by an ENDATA indicator following the last condition-label data control record.

IF the condition specified by the data control record is NOT true, the procedure begins a search for the label on the CONTROL device. The search and procedure are completed when the label record or an END call record is read. Control is returned to LP-MOSS, which reads the next procedure call record.

Data Control

The first name field (positions 5-12) of the data control records read by IF and IFNOT specify the condition to be tested. The second name field (positions 15-22) contains the label to be searched for if the test is positive. The data control records must be followed by an ENDATA indicator. The conditions that may be tested are:

- OPTIMUM. — An optimum and feasible solution to an LP problem has been obtained by OPTIMIZE.

- MAJOR. — A major error has been detected during the preceding run. The major error switch is set to off if it is on.

- MINOR. — A minor error, usually indicating input data error, has been detected during the preceding run. The minor error switch is set to off after it has been tested successfully as being on.

- NORMAL. — No errors, major or minor, have been detected during the preceding run. Both the minor and major error switches are set off after the test.

- UNBOUND. — The current LP problem has no finite solution; the value of the objective variable is unbounded.

- INFEAS. — The current LP problem has no feasible solution; the problem is infeasible.

- ANY. — The test for ANY allows the user to employ an unconditional break in order to bypass a portion of the program sequence.

Label Records

These records must be prepared in indicator format (label in positions 1-8). The label must contain a digit in the first position.

Output

Data control records read by IF and IFNOT are logged. Of the records read during the search phase, only the record ending the search is logged.

SOLVE

SOLVE determines the solution to a set of simultaneous equations. The inversion method used is designed to maintain accuracy when inverting sparse matrices. SOLVE should not be used to solve large systems of equations with many coefficients.

Data Preparation

The problem must be defined as a set of equations, as in the LP formulation. The row (variable) names must be distinct from column (variable) names. Figure 68 illustrates data preparation and control sequence for the solution of a set of simultaneous equations. Figures 69 and 70 show the solution reports.

Data Control

SOLVE is used in precisely the same way as OPTIMIZE (see "OPTIMIZE"), except that no objective variable need be specified. The data file containing the equation data, previously formed by INPUT or MERGE, must have been retrieved by a MOVE DATA record, and the name of the bound set, RHS and/or range set to be used must have been specified by MOVE BOUNDS, RHS, and/or RANGE control data records.

Output

The output report is exactly the same as the LPSOLUTION report (see "LPSOLUTION"). The report shows the solution value for each variable of the problem. The user is cautioned to observe the solution type of column variables. An incomplete and invalid solution is shown by one or more column (variables) at LL (lower level). This indicates either non-independence of row or column variables, or numerical processing difficulties.

Major Errors

If there is no equation data in the problem file retrieved by the MOVE DATA record, or if no file was retrieved, a major error is recorded and the procedure is terminated.

```
INPUT
NAME                SIMULTEQ
*
* DESCRIPTION
*
*       COLUMNS     R-H-S
*       X1 X2 X3    B1 B2
*
*ROWS
*   Y1   1  1      = 7  9
*   Y2   2     -1  = 1  9
*   Y3      3  -2  = 2  6
*
*MATRIX COEFFICIENTS
     X1             Y1                    1.
     X2             Y1                    1.
     X1             Y2                    2.
     X3             Y2               -    1.
     X2             Y3                    3.
     X3             Y3               -    2.
RHS
*FIRST R-H-S
     B1             Y1                    7.
     B1             Y2                    1.
     B1             Y3                    2.
*SECOND R-H-S
     B2             Y1                    9.
     B2             Y2                    9.
     B2             Y3                    6.
ENDATA
MOVE
     RHS            B1
ENDATA
SOLVE
MOVE
     RHS            B2
ENDATA
SOLVE
```

Figure 68.

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
X1	B*	2	3.000	***********	0.000	0.000	0.000
Y1	LL	0	7.000	***********	7.000	0.000	0.000
X2	B*	2	4.000	***********	0.000	0.000	0.000
Y2	LL	0	1.000	***********	1.000	0.000	0.000
X3	B*	2	5.000	***********	0.000	0.000	0.000
Y3	LL	0	2.000	***********	2.000	0.000	0.000

Figure 69. Solution of simultaneous equations, first R-H-S

VARIABLE	TYPE	ENTRIES	SOLUTION ACTIVITY	UPPER BOUND	LOWER BOUND	CURRENT COST	REDUCED COST
X1	B*	2	5.571	***********	0.000	0.000	0.000
Y1	LL	0	9.000	***********	9.000	0.000	0.000
X2	B*	2	3.428	***********	0.000	0.000	0.000
Y2	LL	0	9.000	***********	9.000	0.000	0.000
X3	B*	2	2.142	***********	0.000	0.000	0.000
Y3	LL	0	6.000	***********	6.000	0.000	0.000

Figure 70. Solution of simultaneous equations, second R-H-S

PROGRAM TERMINATION

END

The END procedure is used to terminate an LP-MOSS run and return control to the IBM 1130 Monitor. The run will be terminated when the END procedure is called. It is usually called by the user via an LP-MOSS Monitor control record but may also be called by the IF and IFNOT procedures if an END control record is read during a bypass operation.

Chapter 3: LPS INPUT DATA FORMATS

This chapter provides a discussion of the LPS formats. The system provides a very flexible problem definition capability for the advanced LP user, together with a minimum of data preparation detail for all users. Certain input data options are designed expressly for the 1130 LPS user who is also an MPS/360 user. Some portions of this chapter discuss features that may be of little interest to the new LP user; other portions are directed primarily to the joint 1130 LPS and MPS/360 user.

The section entitled "Basic Input Record Formats" provides a discussion of input from card and paper tape, the general formats and rules governing data preparation. The precise format of each type of record is illustrated and described.

The section entitled "Input Data" completes the additional details for the preparation of basic data required for problem definition. The beginner should skip the subsections "Standard Bounds" and "Selection Files — ROWS and COLS". The remainder of "Input Data", except for "ENDFILE Indicator", is oriented to the joint 1130 LPS and MPS/360 user.

The section entitled "Revise Data" discusses the LPS disk problem data maintenance functions. This topic should be thoroughly read and understood before the REVISE procedure is used.

The section entitled "Insert Data" is provided for completeness, since the data under discussion is almost always prepared by the PUNCH procedure.

Chapter 3 concludes with a discussion of compatibility with MPS/360.

INPUT DATA PREPARATION

The Mathematical Programming Input Form (X20-1761) is recommended for input data preparation.

BASIC INPUT RECORD FORMATS

The 1130 LPS input procedures INPUT, REVISE, and INSERT will accept input data in the form of 80-column cards or 61-position paper-tape records. The input device is specified by the user when the system is generated (see "Initializing LP-MOSS, the COMM1 Routine" in the Application Directory), but can be changed during a run by use of the MOVE procedure.

All input data records are in one of three basic formats:

1. Element format (Figure 71), used for actual problem data. The data control records also use this format.
2. Indicator format (Figure 72), used to indicate the beginning and end of sets of data. The procedure call records also use this format.

Positions	Field
1	Must be blank
2-3	Type
4	Must be blank
5-12	Name field 1
13-14	Must be blank
15-22	Name field 2
23-24	Must be blank
25-36	Value field 1
37-39	Must be blank
40-47	Name field 3
48-49	Must be blank
50-61	Value field 2

Figure 71. Element format

3. Comments format (Figure 73), used to include comments records in the unit record data. Comments records are not printed out when included in the data deck. Comments records in this format may also be placed in the sequence of procedure call records, and will be printed out.

/LISTON (record positions 1-7) can be used to begin listing input data on the typewriter. All element, indicator, and comment cards following the /LISTON record will be listed until an ENDATA or a /LISTOFF (record positions 1-8).

Blank records may appear anywhere in an input unit record file (not in a sequence of data control records) and are ignored.

The following rules apply to the construction of all name and value fields:

1. User-defined names consist of 1 to 8 alphameric characters. It is recommended that names be left-justified and not contain embedded blanks.
2. System-defined names, such as ENDATA, must be left-justified and must not contain embedded blanks.
3. A number in a value field may consist of from 1 to 11 digits. The number must contain a decimal point. The number must not contain embedded blanks. A negative number is denoted by an 11-punch in position 25. Punching the decimal point in position 30 is recommended to simplify data checking. (See figure 74.)

An invalid value field may cause an 1130 MONITOR F003 (accumulator display) halt, or, if the decimal point is missing, the position of the decimal point is assumed to lie between the 6th and 7th positions of the value field. This usually results in wasted computer processing time.

Positions	Field
1-14	Indicator name
15-22	Name

Figure 72. Indicator format

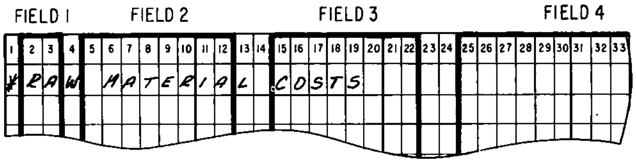


Figure 73. Typical comments record. Record position 1 must contain an asterisk. Comments records must contain valid number fields (positions 25-36, 50-61) when placed within unit record input or revision data files.

INPUT DATA

The NAME indicator

A NAME indicator is used to give a name to the problem and should be the first record of the INPUT unit record file. Its format is shown in Figure 75. If no NAME indicator is provided, the problem is given a name consisting of eight blanks. If there is already a disk problem file with the specified name, it is automatically deleted.

Problem Equation Data

The problem equations are input by records in element format. The contents of these records are shown in Figure 76. Name field 3 and coefficient field 2 may be used to input two equation coefficients on one record. If so desired, the second coefficient represents a second row variable entry for the first name variable.

A record is required to represent each of the nonzero coefficients on the right-hand side of the problem equations (see "Formulation Rules") in the Appendix.

No indicator record is required to input the problem equation data, and the data may appear in any sequence. For compatibility with MPS/360, however, a set of records consisting entirely of problem equation data may be preceded by a COLUMNS indicator. All records following such an indicator are assumed to be problem equation data until another indicator record is found.

Record Position	25 50	26 - 29 51 - 54	30 55	31 - 36 56 - 61
Contents	Sign		Dec. Pt.	
	-	xxxx	.	xxxxxx

Figure 74. Suggested value field format

Field	Contents
Indicator name Name	Name Name of the problem

Figure 75. Name indicator

The Bounds on the Variables

The bounds on the problem variables are input to LPS by element records defining the upper and lower bounds on each bounded variable. The bounds are input as a set that has a user-defined name. One unit record input file may contain several distinctly named bound sets.

Figure 77 shows the format of a bound element record, and Figure 78 shows the bounds defined by the various bound types. The alternate bound type designations provide compatibility with MPS/360.

Bound element records are required for:

1. All nonzero lower bounds. A variable that is given no lower bound by the user is given a lower bound of 0.
2. All finite upper bounds. A variable that is given no upper bound by the user is given an upper bound of 10^{10}, which is defined as LPS INFINITY.

No indicator record is required to input the bound data, and the data may appear in any sequence. For compatibility with MPS/360, however, a set of records consisting only of bound data may be preceded by a BOUNDS indicator record. All records following such an indicator are assumed to be bound data records until another indicator record is found.

The ENDATA Indicator

The last record of the input deck must be an ENDATA indicator record.

Field	Contents
Type	Must be blank
Name 1	The name of a variable appearing on the right-hand side of an equation(s)
Name 2	The name of a row variable
Value 1	The coefficient of the variable in name field 1 in the equation corresponding to the row variable in name field 2
Name 3	The name of a row variable
Value 2	The coefficient of the variable in name field 1 in the equation corresponding to the row variable in name field 3

Figure 76. Equation element record

Field	Contents
Type	Bound type
Name 1	Name of the bound set
Name 2	Name of a variable
Value 1	Bound value
Name 3	Must be blank
Value 2	Must be blank

Figure 77. Bound element record

A Basic INPUT Data Deck

The contents of a basic INPUT data deck are shown in Figure 79. They consist of:

1. A NAME indicator record
2. Problem equation and bound data records in any sequence
3. An ENDATA indicator record

Standard Bounds

If no lower bound is input in a particular bound set for a variable, it is given a lower bound of 0. If no upper bound is input for a variable in a particular bound set it is given an upper bound of INFINITY. These implicit bounds (0, INFINITY) may be changed for a particular variable by a standard bound record, as shown in Figure 80. The possible bound types are the same as those for bound data records, as shown in Figure 78. A blank in the bound type field is equivalent to a G type.

Standard bound records may appear anywhere in the INPUT data deck except in files headed by COLUMNS, BOUNDS, RHS's, or RANGES indicator records.

Selection Files — ROWS and COLS

Files of records headed by ROWS or COLS indicator records are used to select row or column variables for use in optimization. A ROWS file is a list of the row (variables) to be used in optimization. It is necessary only if not all the row variables are to be used in the optimization. A COLS file is a list of the column variables to be used in optimization. It is necessary only if not all column variables are to be used.

ROWS and/or COLS files can be used to select a subset of the problem for optimization. The selection of a subset of the problem data for optimization is a traditional feature of LP systems. The primary utility of the selection capability for the 1130 LPS user is compatibility with MPS/360 input ROWS file and simplification of problem conversion from other systems to 1130 LPS.

The records following a ROWS (COLS) indicator are in standard bound record format (see Figure 80) and may be used to set standard bounds if so desired. Each record names a row (column) variable to be used during optimization. Any row (column) variable not so named will be ignored. All records following a ROWS (COLS) indicator are assumed to be naming row (column) variables for selection until another indicator record is found.

Bound Type	Upper bound	Lower bound
UP or UB	Bound value	No effect
LO or LB	No effect	Bound value
FX or E	Bound value	Bound value
PL	INFINITY	No effect
MI	No effect	-INFINITY
FR or N	INFINITY	-INFINITY
G	INFINITY	Bound value
L	Bound value	-INFINITY

Figure 78. The bound types

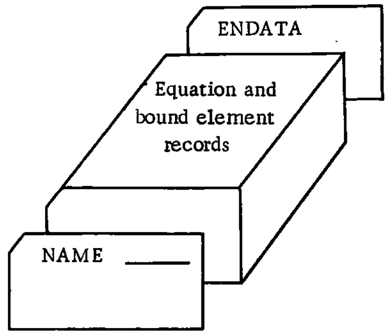


Figure 79. A basic INPUT data deck

RHS's and Ranges

The bounds on a variable may be specified by RHS and range values, rather than by an explicit bound value.

Field	Contents
Type	Bound type (if any)
Name 1	The name of the variable being bounded or selected
Name 2	Blank
Value 1	The bound value (if any)
Name 3	Blank
Value 2	Blank

Figure 80. Standard bound record and selection file record

The bounds on a variable generated by RHS and range values depend on the standard bounds of the variable. The determination of the bounds generated by the general case is quite complex. However, the bounds generated in 1130 LPS by MPS/360 input provide an equivalent problem definition.

The 1130 LPS does not use the traditional slack variable technique for the conversion of row types to equations. The 1130 LPS will accept the MPS/360 input data within the limitations described under "Compatibility with MPS/360". The ROWS, RHS, and RANGE information is translated internally, as required, to specify upper bounds and lower bounds on the variables. The technical details of the conversion are given in the Appendix.

Right-Hand-Side Files — RHS and RHS's

A right-hand-side (RHS) file must be headed by an RHS or RHS's indicator record. The format of a right-hand-side record is shown in Figure 81. All records following an RHS or RHS's indicator are assumed to be right-hand-side records until another indicator is found.

Range Files — RANGES

A range file must be headed by a RANGES indicator record. The format of a range record is shown in Figure 81. All records following a RANGES indicator are assumed to be range records until another indicator is found.

The ENDFILE Indicator

The ENDFILE indicator may be used to terminate a set of records headed by another indicator (for example, ROWS). An ENDFILE indicator record is used to terminate such a set of records only if the set of records is not immediately followed by another indicator record and its associated data.

Field	Contents
Type	Blank
Name 1	Name of the RHS or range set
Name 2	Name of a variable
Value 1	RHS or range element for the variable in Name 2
Name 3	Name of a variable
Value 2	RHS or range element for the variable in Name 3

Figure 81. RHS and range records. The general bound format (Figure 77) gives the record positions for each field.

REVISE DATA

A REVISE data deck is identical in format and makeup to an INPUT data deck, except that:

1. A NAME indicator, if used, must be the first card.
2. Remove (X-out) records may be used to remove obsolete variables, bound sets, etc.

Remove Records

Any variable, bound set, RHS set or range set may be removed from a problem file by a remove record (Figure 82). Remove records may appear anywhere in a REVISE data deck. When removing obsolete data, do not include a new variable that bears the same name as a removed variable; if you do, it will be removed. The variable may subsequently be REVISEd into the problem in a later REVISEion.

Also, note that some of the bound types generate an upper bound and a lower bound entry. An UB (UP) or an LB (LO) bound entry will revise one and only one bound. If the previous variable bound type (see Figure 78) defined two bounds (FX, EQ, E, FR, N, G, and L all define both a UB and an LB), a single UB or LB will change only one bound.

Field	Contents
Type	bX or Xb
Name 1	The name of the variable, bound set, etc., to be deleted
Name 2	Blank
Value 1	Blank
Name 3	Blank
Value 2	Blank

Figure 82. REVISE delete record format

INSERT DATA

The NAME Indicator

A NAME indicator is used to name the problem file into which the starting solution is to be stored. If no NAME indicator is provided, the name will be assumed to consist of eight blanks.

If there is already a problem file with the specified name, the starting solution is stored in it. If not, a problem file is created. The 1130 LPS will accept an MPS/360 PUNCHed solution status file.

Status records

The format of a status record is shown in Figure 83, and the possible status types are illustrated in Figure 84.

Positions	Field
1	Blank
2-3	Type
4	Blank
5-12	Name 1
13-14	Blank
15-22	Name 2
23-36	Blank

Figure 83. INSERT data format

The ENDATA Indicator

The last record of an INSERT data deck must be an ENDATA indicator.

An INSERT Data Deck

The contents of an INSERT unit record file are shown in Figure 85. They consist of a NAME indicator, status records, and an ENDATA indicator.

Type	Meaning
LL	The variable in Name 1 is at lower level
UL	The variable in Name 1 is at upper level
EQ	The variable in Name 1 is at lower level
BS	The variable in Name 1 is at intermediate level
**	The variable in Name 1 is at intermediate level
XL	Variable Name 2 at lower level, Name 1 intermediate
XU	Variable Name 2 at upper level, Name 1 intermediate

Figure 84. INSERT data types

```
PUNCH
NAME              LPSAMPLE
 EQ WEIGHT
 BS BIN.2
 BS BIN.4
 BS BIN.5
 BS ALUMINUM
 BS SILICON
 UL FE
 UL MN
 LL SI
 UL BASE
ENDATA
```

Figure 85.

COMPATIBILITY WITH MPS/360

LPS has been designed to be compatible with MPS/360. Wherever possible, LPS uses the names and formats used in MPS/360. With the few exceptions listed below, LPS will accept input data decks prepared for MPS/360. The LPS input procedures, however, are designed to provide facility and flexibility rather than great efficiency. For this reason data decks prepared for 1130 LPS will not necessarily be acceptable to MPS/360. Therefore, if compatibility between MPS/360 and 1130 LPS is essential, data decks should be prepared as described in MPS/360 (360A-CO-14X) Linear Programming User's Manual (H20-0291) rather than as described in the relevant chapters of this manual. Though the LPS procedure BCDOUT will punch a data deck that is usually acceptable to MPS/360, this is sometimes not possible, because of the additional features allowed by the 1130 LPS input procedures.

The following MPS/360 input data is not recognized by LPS:

1. The D row type of MPS/360 is not allowed by 1130 LPS. 1130 LPS allows "rows" (row variables) to be added together without the use of any special feature (see "Formulation Rules" in the Appendix).
2. The MPS/360 MARKER records will generate an extraneous row by 1130 LPS which is not used during the optimization, and which will have no effect on the optimal solution.
3. The MPS/360 SCALE records will generate an extraneous row by 1130 LPS which is not used in the optimization and which will have no effect on the optimal solution. LPS automatically scales problem data to increase the accuracy of the solution found.
4. A dollar sign ($) in field 3 or field 5 indicating comments in the remainder of the card is not recognized by LPS. Such records in the ROWS file will have no effect. If, however, they appear in the COLUMNS, RHS's, or RANGES files, an extraneous row will be generated which is not used in the optimization and which will not affect the optimal solution. The $ sign must not appear in a value field; it will cause an F003 Monitor FORTRAN format halt.
5. LPS considers a blank as a valid character in a name field. It does not automatically left-justify names or squeeze out embedded blanks.
6. Row names and column names in LPS must be distinct, unless the column is to be the "slack" variable for the row of the same name (see "Formulation Rules" in the Appendix).
7. The input data for the 1130 LPS REVISE is the same as that for the 1130 LPS INPUT. 1130 LPS does not use MPS/360 REVISE functions MODIFY, DELETE, BEFORE, and AFTER records.
8. 1130 LPS coefficients or value fields must contain a decimal point for correct data input. If no decimal point is punched, a decimal point is assumed to lie between the sixth and seventh field positions.

CALLING THE SYSTEM

To call LP-MOSS, insert the disk cartridge on which the system has been loaded, and turn the FILE switch ON. When the FILE READY light comes on, use the appropriate procedure given below.

CARD SYSTEM

The deck required for an LP-MOSS run consists of:

1. An IBM 1130 Disk Monitor Cold Start card
2. //bJOB
3. //bXEQbMOSS
4. The LP-MOSS control and/or input deck for this run. If there is no deck, place a blank card after the XEQ card.

To begin the run:

1. Press START on the printer (if any).
2. Run all cards out of the card reader by pressing NPRO on the card reader.
3. Place the deck in the hopper of the card reader.
4. Press START on the card reader.
5. Press IMM STOP, RESET, and then PROGRAM LOAD on the console.

NOTE: Remove the LP-MOSS disk after the run to protect the data files.

PAPER TAPE SYSTEM

The paper tape records required for an LP-MOSS run are:

1. An IBM 1130 Disk Monitor Cold Start record
2. //bJOB
3. //bXEQbMOSS
4. The LP-MOSS control and/or input records for this run (if any)

To begin the run:

1. Press START on the printer (if any).
2. Put the cold start paper tape record into the reader; position any of the delete codes after the program name in the tape leader under the read starwheels.
3. Press IMM STOP, RESET, and then PROGRAM LOAD on the console.
4. Put the remaining records into the reader, always positioning a delete code under the read starwheels. Press START on the console to read the next record.

NOTE: Remove the LP-MOSS disk after the run to protect the data files.

USING THE TYPEWRITER

If the typewriter is assigned as the control device when a control record is required by the system, the KB light on the console will come on. Type in the required record and press the EOF key to indicate that the record is ended. If a mistake is made in typing, press the ERASE key and begin the record again.

INTERRUPTION, ERROR, AND MALFUNCTION RECOVERY

RECOVERY OPERATING PROCEDURE M06 NO PROGRAM ERROR

To restart after an invalid procedure call resulting in an M06 NO PROGRAM has been read by the LP-MOSS Monitor, the following procedure can be used (provided the 1130 has not been used for another job and core storage has not been altered).

Card System

1. Run all cards out of the card reader.
2. Place the following deck in the hopper of the card reader:
   a. //bJOB
   b. //bXEQbMOSSM
   c. The corrected procedure call card (or a blank card if some other CONTROL device is used)
   d. The remainder of the deck for this run
3. Press START on the card reader.
4. Press START on the console.

Paper Tape System

1. Remove the current paper tape from the paper tape reader.
2. Place the following records in the paper tape reader, beginning with a delete code under the starwheels:
   a. //bJOB
   b. //bXEQbMOSSM
   c. The corrected procedure call record (if the paper tape reader is the control device)
   d. The remainder of the paper tape for this run

3. Press START on the console.

RESTART OPTIMIZATION OPERATING PROCEDURE

To restart after a machine malfunction (for example, an F102 disk read error or an operator interruption during OPTIMIZE), the following procedure can be used (provided the disk containing LP-MOSS has not been used since the interruption).

Card System

1. Run all cards out of the card reader.
2. Place the following card deck in the hopper of the card reader:
   a. An IBM 1130 Disk Monitor Cold Start card
   b. //bJOB
   c. //bXEQbMOSS
   d. RESTART
   e. The remainder of the procedure (and data cards) including the interrupted procedure
3. Press START on the card reader and printer (if any).
4. Press IMM STOP, RESET, and PROGRAM LOAD on the console.

Paper Tape System

1. Remove the current paper tape from the paper tape reader.
2. Place the following records in the paper tape reader:
   a. An IBM 1130 Disk Monitor Cold Start record
   b. //bJOB
   c. //bXEQbMOSS
   d. RESTART
3. Press IMM STOP, RESET, and then PROGRAM LOAD on the console.
4. Put the remaining records, including the interrupted procedure called, into the reader, always positioning a delete code under the read starwheels. Press START on the console to read the next record.

INTERRUPTING OPTIMIZE

To interrupt the OPTIMIZE procedure, simply press PROGRAM STOP on the console. Remove the LP-MOSS disk before beginning the next job. To begin the next IBM 1130 Disk Monitor System job, use the cold start operating procedures detailed in C26-3750.

CARRIAGE CONTROL TAPE

The carriage control tape must contain:
Channel 1 punch to define the first line on a page.
Channel 12 punch to define the last line on a page.

HALTS AND MESSAGES

HALTS

During normal operation of the system the computer should not halt except when waiting for a record to be typed on the console typewriter. In this case the KB SELECT light on the console is on. To continue, type the required record and press the EOF key on the typewriter.

Halts may also occur if:

1. An I/O unit is not ready. To continue, ready the unit and press PROGRAM START on the console.
2. An error is detected by the 1130 FORTRAN I/O routines. Such an error causes control to return to the 1130 Disk Monitor System. For a complete list of FORTRAN I/O errors, see C26-3750. The following are 1130 Disk Monitor System input/output subroutine halts:
   a. Disk read error F102, use RESTART if optimization is in process.
   b. Format input error F003, correct.
   c. Disk overflow F101, problem too large or data area inadequate for this additional problem, may or may not be improved by EDIT.

If any unexplained halts occur, submit an Authorized Programming Analysis Report (APAR) through your local IBM systems engineer. In addition to the report send:

1. A sheet showing the status of the console lights.
2. A core dump taken at the point at which the system halts.
3. The log of the run up to that point.
4. Input decks for any problems processed during the run.

These materials will enable any program error to be found and corrected as quickly as possible.

ERROR PHILOSOPHY

There are two types of error: major errors, which prevent a procedure from completing its task as required by the user, and minor errors, which are associated with problem data. Whenever possible, corrective action will be taken by the procedures, as specified below, and the processing of the current problem will be continued.

PROGRAM AND ERROR MESSAGES

The following list shows every message logged by the system, the probable cause of the error, and the corrective action taken by the system, or to be taken by the user:

- BOUND SET 'NNNNNNNN' MISSING
  Minor error. The bound set named is not in the problem file. The name of the bound set used is logged on the next line.

- CARD NOT BLANK
  Major error. The first card of a deck to be used for punching is not blank. The procedure is terminated.

- COLUMN 'NNNNNNNN' EMPTY
  Minor error. The column variable named has no equation elements (that is, it does not appear in any of the problem equations). The variable will have no effect on the optimal solution.

- DUPLICATE ELEMENTS
  Minor error. Duplicate elements have been found in an input data deck. The elements used are listed below the message.

- INCOMPLETE OPTIMIZATION
  The maximum number of iteration specified have been taken by OPTIMIZE without reaching an optimal solution.

- INFEASIBLE BOUNDS NAME UPPER BOUND LOWER BOUND
  Major error. The lower bound on the variables listed are greater than the upper bound. The name, upper bound, and lower bound on each variable are listed. The optimization is abandoned.

- INVALID FORMAT
  Minor error. An input data record has been found which has a nonblank character in one of the fields that must be blank. The record is logged immediately above the message and ignored.

- INVALID INDICATOR
  Major error. An invalid indicator name has been found. The record is logged immediately above the message and treated as an ENDATA indicator.

- INVALID RECORD
  Major error. An invalid data control record has been found. The record is logged immediately above the message. The action taken depends on the procedure logging the message.

- INVALID TYPE
  Minor error. An input record with invalid bound or status type has been found in an input deck. The record is logged immediately above the message and ignored.

- MAXIMUM PROCESSING ERROR
  XXXXXX.XXXXXX
  OPTIMIZE has been terminated because of uncorrectable computational error. The maximum error found is shown.

- M06 NO PROGRAM
  This 1130 Disk Monitor System message indicates that an invalid procedure call has been read by the LP-MOSS monitor. Use the recovery operating procedure to continue (see "Recovery Operating Procedure M06 No Program Error" in this chapter.

- NO DATA
  Major error. There is no problem data in the problem file retrieved by a MOVE DATA record or no file was retrieved.

- NO OBJECTIVE
  Minor error. No objective variable was specified for an optimization. The name of the variable used will be logged on the next line.

- NO SOLUTION
  Major error. No current solution is available to be saved by SAVESOLUTION or PUNCH. The procedure is terminated.

- NONE USED
  No bound set (RHS, range set) is in the problem file to replace one specified.

- OBJECTIVE 'NNNNNNNN' MISSING
  Minor error. The objective variable named by a MOVE MINIMIZE or MAXIMIZE record is not in the problem file. The name of the variable used is logged on the next line.

- RANGE SET 'NNNNNNNN' MISSING
  Minor error. The range set named by a MOVE RANGE record is not in the problem file. The name of the range set used is logged on the next line.

- RHSIDE SET 'NNNNNNNN' MISSING
  Minor error. The RHSide set named by a MOVE RHSIDE record is not in the problem file. The name of the bound set used is logged on the next line.

- ROW 'NNNNNNNN' EMPTY
  Minor error. The row variable named has no equation elements. The variable will have no effect on the problem solution.

- SOLUTION INFEASIBLE
  No solution to the problem can be found satisfying all the variable bounds. The variables whose values exceed their bounds are so noted on the LPSOLUTION report.

- SOLUTION OPTIMUM
  An optimum solution has been found.

- SOLUTION UNBOUNDED
  The value of the objective variable can be improved without limit. The variable causing the unboundedness is so noted on the LPSOLUTION report.

- TOO MANY PROBLEMS
  Major error. More than 64 problem files were specified to be combined by MERGE. The procedure continues as if an ENDATA record had been read. The file so far formed can be combined with the remaining files in another MERGE operation.

- 'NNNNNNNN' MISSING FROM DICTIONARY
  Major error. The problem file named on a DATA control data record is not in the dictionary.

- 'NNNNNNNN' USED
  The name of the bound set, RHS, Range Set, or Objective used instead of the one named by the user.

- XXXXX ROWS TOO MANY
  Major errors. There are more than 700 rows in a problem. The number of rows in excess of 700 is logged.

FORMULATION RULES

1. The relationships between the problem variables are expressed as a set of equations. The equations must obey the following rules:
   a. Each equation may have only one variable on the left-hand side of the equal sign.
   b. The coefficient of this variable must be 1.0.
   c. No variable may appear on the left-hand side of more than one equation.

A variable appearing on the left-hand side of an equation is called the row variable for that equation. It may appear on the right-hand side of other equations. Any variable not appearing on the left-hand side of any equation is called a column variable.

2. Restrictions on the values that the problem variables may take are expressed as bounds on the variables.

3. The optimization procedures will determine values for the problem variables which:
   a. Satisfy the problem equations.
   b. Satisfy the bounds on the problem variables.
   c. Minimize or maximize the value of one of the variables.

INPUT DATA FORMAT SUMMARY

INPUT and REVISE

Figure 86 is a summary of the input data formats used by the INPUT and REVISE procedures.

Symbols Used

AAAAAAAA — The name of the problem file in which data will be stored
BBBBBBBB — A bound set name
NNNNNNNN — The name of a variable, bound set, RHS set, or range set
RRRRRRRR — The name of a row variable
SSSSSSSS — The name of a RHS or range set
TT — Bound type
VVVVVVVV — The name of a variable

Bound Types

Figure 87 shows all the possible bound types and the bounds they generate.

INSERT

Figure 88 is a summary of the input data formats used by the INSERT procedure.

Symbols Used

IIIIIIII — The name of a variable that is to be at intermediate level (XL and XU status types only)
OOOOOOOO — The name of a variable that is to be at a bound (XL and XU status types only)
SS — Status type
VVVVVVVV — The name of a variable

Status Types

Figure 89 shows all possible status types and their meanings.

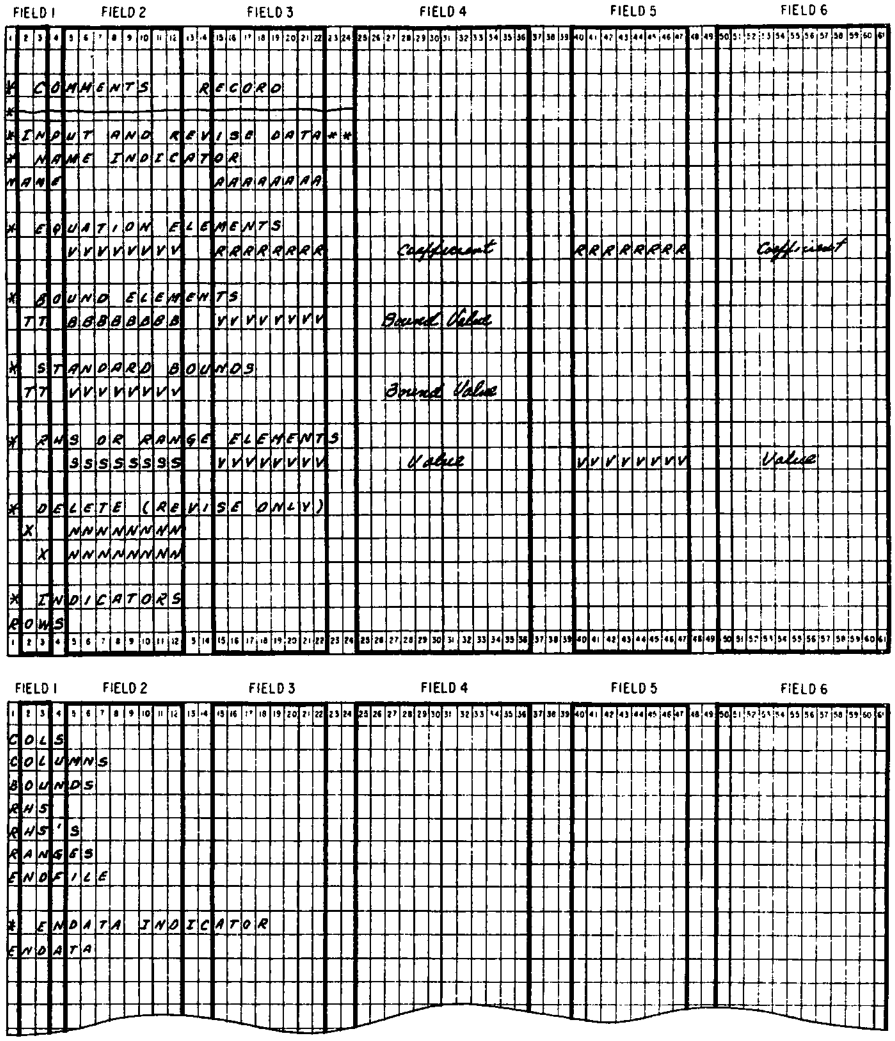


Figure 86.

Bound Type	Lower Bound	Upper Bound
LB LO	Bound value	No effect
UB UP	No effect	Bound value
FX Eb bE	Bound value	Bound value
FR Nb bN	(-)Infinity	(+)Infinity
Gb bG	Bound value	(+)Infinity
Lb bL	(-)Infinity	Bound value

Figure 87. INPUT and REVISE bound types

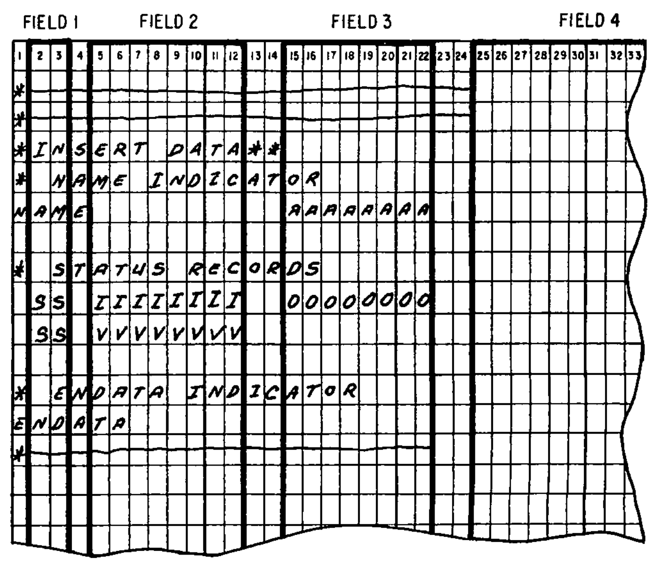


Figure 88.

Type	Status of variable in	
	5-12	15-22
XL	Intermediate level	Lower bound
XU	Intermediate level	Upper bound
LL	Lower bound	
WL	Upper bound	
BS	Intermediate level	

Figure 89. INSERT status types

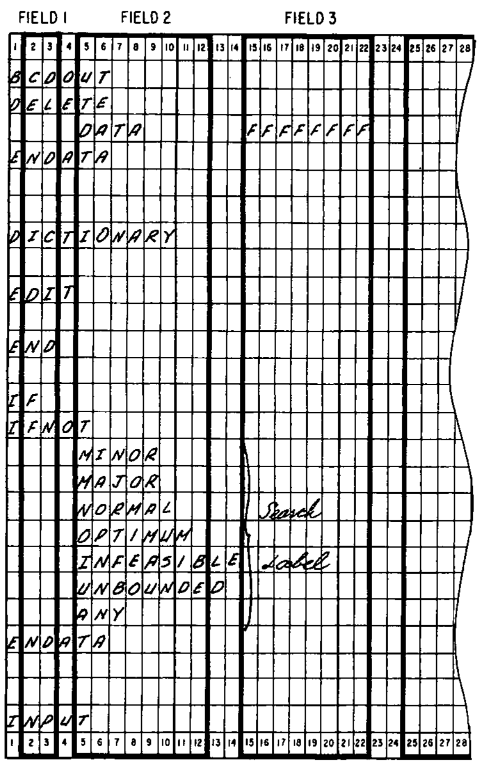
CONTROL RECORD FORMATS

Figure 90 shows all the procedure call control records, and the data control records that may be read by each procedure. For the MOVE control data records requiring a numeric value, the initial value of the corresponding parameter is shown in the value field.

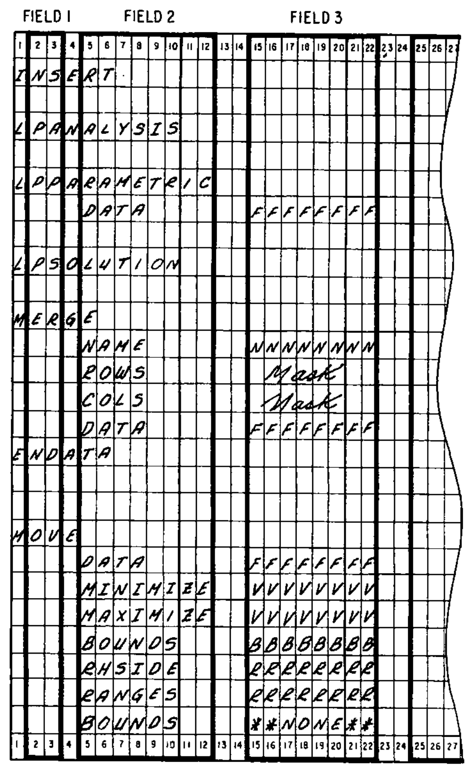
DATA CONTROL

Symbols Used

BBBBBBBB — The name of a bound set
FFFFFFFF — The name of a problem file containing data to be used
NNNNNNNN — The name of a problem file into which data is to be stored
RRRRRRRR — The name of an RHS set, or range set, as appropriate
VVVVVVVV — The name of a variable



a.



b.

Figure 90

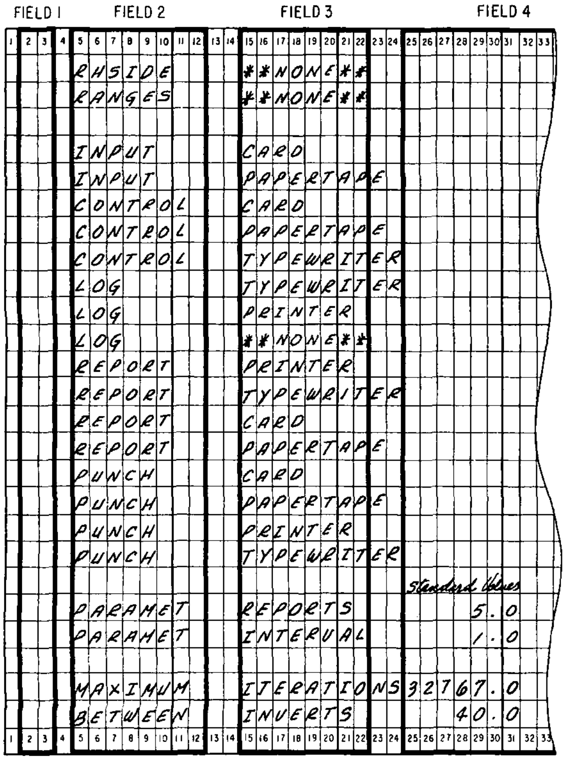
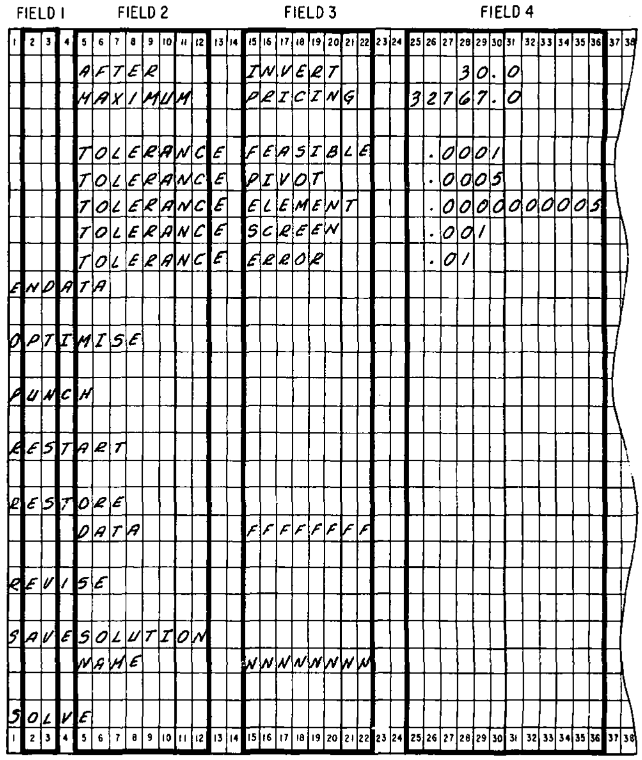


Figure 90. (Cont) c.



d.

BOUND GENERATION

The rules for finding the bounds generated by LPS from RHS and range entries are as follows:

1. If there are entries for the variable in a bound set specified for use during optimization, the RHS and range entries are ignored.
2. If not, the lower and upper bound are set to their standard values — 0 and INFINITY — unless otherwise specified by a standard bound record.
3. If both the standard lower and upper bounds are finite but not equal, the RHS and range entries are ignored.
4. If both the standard lower and upper bounds are infinite, the RHS and range entries are ignored.
5. Otherwise, if there is an RHS entry, set all finite bounds equal to the RHS value.
6. Then, if there is a range entry, it is applied as shown in Figure 91.

Bounds after applying RHS value	Final bounds after applying Range value r
UB=LB	r < 0 -- LB=UB-abs(r) r ≥ 0 -- UB=LB+abs(r)
UB=INFINITY LB=FINITE	UB=LB+abs(r)
UB=FINITE LB=INFINITY	LB=UB-abs(r)

Figure 91. RANGE values when RHS sets are used. This provides compatibility with MPS/360. Abs(r) means the absolute (unsigned) value of r.

PROBLEM CAPACITY

LPS capacity is restricted by row variables, disk capacity, and numerical accuracy. The maximum number of row variables is 700. The maximum disk capacity is 794 sectors. The following formulas can be utilized to find the approximate number of disk sectors used in storing and processing a problem:

If a problem is to be input and processed, and the solution output taken, the approximate number of disk sectors used is $(R+C)(1/5+N^x/25)$.

If the problem is to be input only, the approximate number of disk sectors used is $[(R+C)/25] + [CN/50]$.

Where:

- R is the number of rows in the problem
- C is the number of columns (including RHS, range, and bound sets)
- N is the average number of nonzero elements per column
- x lies between 1 and 1.5, depending on problem characteristics

Examples:

Rows	Columns	Nonzero Elements per Column	Number of Sectors Used: Input Only	Number of Sectors Used: Total
100	200	6	36	132 to 162
100	200	12	60	204 to 245
250	500	10	130	550 to 650
700	1000	5	168	680 to 833

PRECISION AND ACCURACY

Numerical accuracy depends on many factors, including problem size and the average number of column elements. The system scaling procedures and inversion methods are designed to produce accurate, reliable solutions within the limits of a 31-bit mantissa.

TIMING

It is impossible to predict the solution time for a particular LP problem, even if the solution time for a similar problem is known. However, as a rough guide to estimating throughput using LP-MOSS/1130, the table below shows the approximate solution times for various problem sizes. The times shown are for the initial solution and for solution beginning at an advanced starting solution.

Number of: ROWS	Number of: COLUMNS	Approximate Solution Time (hours): Initial	Approximate Solution Time (hours): Advanced Starting Solution
50	75	.5 - 1.3	.2 - .4
100	150	3 - 8	1 - 2
250	300	7.5 - 20	2 - 5
400	500	12 - 32	3 - 8
500	650	24 - 60	6 - 15

MACHINE AND SYSTEM CONFIGURATION

- 1130 Model 2B with 8192 words of core storage and one disk storage drive
- 1442 Card Read Punch and/or 1134 Paper Tape Reader and 1055 Paper Tape Punch
- 1132 Printer (optional).

The recommended 1130 system for best performance and simplest operation includes a 1442 Card Read Punch with an 1132 Printer.

Programming System

LP-MOSS/1130 operates under control of the IBM 1130 Monitor System, Version 1. The source language is IBM 1130 FORTRAN.

BIBLIOGRAPHY

IBM PUBLICATIONS

Introduction to Linear Programming (E20-8171)

IBM 1130 Disk Monitor Reference Manual (C26-3750)

Aluminum Alloy Blending (E20-0127)

Electric Arc Furnace Steelmaking (E20-0147)

Feed Manufacturing (E20-0148)

Ice Cream Blending (E20-0156)

Blast Furnace Burdening (E20-0160)

Cotton Blending (E20-0164)

Gasoline Blending (E20-0168)

IBM SYSTEM/360 Mathematical Programming System (360A-C0-14x) Linear Programming User's Manual (H20-0291)

OTHER REFERENCES

Charnes, A. and W. W. Cooper, Management Models and Industrial Applications of Linear Programming, John Wiley & Sons, Inc., New York, 1961.

Dantzig, George B., Linear Programming and Extensions, Princeton University Press, 1963.

Garvin, Walter W., Introduction to Linear Programming, McGraw-Hill Book Company, Inc., New York, 1960.

Gass, Saul I., Linear Programming, Method and Applications, McGraw-Hill Book Company, Inc., New York, 1958.

Orchard-Hays, William, Matrices, Elimination and the Simplex Method, C-E-I-R, Inc., Arlington, Virginia, October 1961.

Riley, Vera and Robert Loring Allen, Interindustry Economic Studies, Operations Research Office, Johns Hopkins Press, Baltimore, Maryland, May 1955.

Riley, Vera and S. I. Gass, Bibliography on Linear Programming and Related Techniques, Johns Hopkins Press, Baltimore, Maryland, 1958.

Wolfe, Philip (ed.), The RAND Symposium on Mathematical Programming, Santa Monica, March 1959. Proceedings published by the RAND Corporation, R-351, 1960.

Printed in U.S.A. H20-0345-2

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)