Application Program

1620-1311 Linear Programming System

(1620-CO-04X) Program Reference Manual

This is a general-purpose programming system providing sophisticated mathematical techniques to determine the most efficient use of various resources in carrying out alternative activities. A number of programs are stored on the 1311. Each routine is called into core storage by procedural control cards when it is to be executed. The sequence of control cards defines the solution procedure.

Besides direct optimization, the program provides extensive post-optimal analysis to indicate the effect of changes in constraints, costs, or technology. Automatic procedures have been incorporated for determining the frequency of inversion and appropriate data accuracy, but both may be adjusted by the user to accomplish specialized purposes.

CONTENTS

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

PROGRAMMING SYSTEM ABSTRACT

The IBM 1620-1311 Linear Programming System is a general-purpose programming system designed to provide the 1620-1311 user with sophisticated mathematical techniques to determine the most efficient use of various resources in carrying out alternative activities. The system is composed of a number of programs which are stored on the 1311. Each program routine is called into core storage by procedure control cards (or agendum cards) when that program routine is to be executed. The sequence of the control cards defines the solution procedure.

Besides direct optimization, the program provides extensive post-optimal analysis to indicate the effect of changes in constraints, costs, or technology. Automatic procedures have been incorporated for determining frequency of inversion and appropriate data accuracy, but both may be adjusted by the user to accomplish specialized purposes.

GENERAL DESCRIPTION OF PROGRAMMING SYSTEM

Linear programming (LP) is a mathematical technique for determining the optimum use of various resources (such as capital, raw materials, manpower, and plant or other facilities) to attain a particular objective (such as minimum cost or maximum profit), when there are alternate business activities to be carried out. LP can be used to allocate, assign, schedule, select or evaluate the uses of limited resources for various jobs: blending, mixing, distributing, controlling, budgeting, ordering, bidding, cutting, trimming, pricing, purchasing, planning, and transportation of goods from plants to warehouses to outlets.

The IBM 1620-1311 Linear Programming (LP) System optimizes an objective function subject to the constraints of a system of independent linear equations. For the minimum 1620 configuration, the system will accept problems having up to 100 equations (including alternate objective functions) and up to 1,000 columns (including structural variables and multiple right-hand sides). The revised simplex algorithm with special features is used to obtain a two-phase solution. Source language for the system is IBM 1620 SPS II-D.

The IBM 1620 Monitor I System controls the calling in of the LP System, and its I/O operations. Upon completion of an LP run, control is returned to Monitor. This means that a linear programming problem can be stacked with other jobs run under the 1620 Monitor.

The IBM 1620-1311 LP System is composed of a number of programs which are stored on the disk. Each program is called in only when it is to be executed. The cards used to control this procedure are called agendum cards. These cards are prepared by the user and entered into the system through the card reader. As the agendum cards are read sequentially by the computer, the appropriate agendum routines are called in from the disk and executed to perform the input, solution, output, and utility functions required for an application.

The input data originates on cards and may be stored and maintained on disk for subsequent reprocessing. Data revision is accomplished easily and does not require that all of the data files be reloaded onto the disk. Problems can be run using selected subsets of data, with the required rows and columns being referred to by name. The data files can include multiple objective functions (also called cost rows) and right-hand sides; these are also selected by name when a problem is run.

Processing may be interrupted during calculation, and later continued from the point of interruption.

FEATURES

The specification of input data is simple and provides the following flexibility:

- Variables, right-hand sides, objective functions, and constraints may be selected from card or disk input by item name.
- Data is maintained on disk in compact form.
- Revision of an entry on the disk does not require that the entire file be re-entered on the disk.
- Several problems may reference different subsets of the same card or disk data files.
- Slacks are automatically generated and named.
- A starting basis may be given; if it is not, a slack basis is used.
- The use of SHARE standard matrix card format provides substantial data compatibility with other linear programming systems.

The following advanced mathematical methods are used to increase processing speed and accuracy:

- Revised simplex (product form of inverse) with a choice of automatic or user control of inversion frequency.
- User may maximize or minimize any objective function.
- Assignment and variation of floating-point length and tolerances during solution processing may be under automatic or user control.
- Minimum floating-point length is five digits; maximum is 18 digits.
- The bounded variable algorithm is applied to range ($\geq$ and $\leq$) constraints (and bounded variables). In alloy scheduling (blending), for example, the silicon composition of an alloy (product) may be constrained to the range of 5% to 7.5% by one range constraint.
- Input data files are maintained in compressed floating-point form. This decreases calculation to about 30% of normal floating-point time.

Restart is incorporated into the system as follows:

- Processing may be interrupted; a subsequent run can continue from the point of interruption.

The following extensive checking features are incorporated:

- Input is checked for duplicate row identifications, split or duplicate columns and right-hand sides, and duplicate coefficient entries.
- Each time a reinversion is to be done during solution, the problem is checked for accuracy; if error tolerance is exceeded, the mantissa length and tolerances are adjusted automatically.

Any output may be selective or include all pertinent slacks and variables. All output quantities are in fixed-point form, with a specified number of decimals. Output is optional, and the output unit may be selected as follows:

- The user may specify output device for the iteration log as well as iteration frequency. During solution a sense switch can be used to inhibit output of the log for certain iterations.

- The output device for other reports, which may also be specified, is independent of that used for the iteration log. The output options include: solution amounts for variables and slacks, or both solution amounts and optimum price range for variables and slacks; opportunity (shadow) prices for nonbasis variables and slacks; sensitivity (restriction coefficient shadow price) values for nonbasis variables; and error check for solution using rounded output values.

Post-optimal analysis capabilities include:

- Marginal value -- the change in the objective function per unit increase in the corresponding right-hand-side constant, assuming that a change of basis is not required to maintain feasibility.

- Reduced cost -- the amount by which the cost of a non-basic structural variable would have to be reduced before this variable would tie the optimum basis. A reduction in excess of this amount would push the variable into the basis.

- Cost range -- the objective function interval of a basic solution structural variable in which the basis is optimum; a change in the objective function of the variable which is outside that range would force a change in the basis to preserve optimality.

- Submatrix summary -- the partial sum of the coefficients of a matrix, evaluated at the current solution; the solution contribution of a subset of the variables on a subset of the rows.

After optimum solution, reoptimization can be efficiently accomplished with the following features:

- Using a new right-hand side (or sides)

- Changing values in the right-hand side

- Changing values in the objective function

- Changing values of coefficients in the matrix

MATHEMATICAL METHODS SUMMARY

The 1620-1311 Linear Programming System uses the bounded variable, product form of the inverse, simplex method. The simplex method is based upon the fact that if there are m equations (or rows) in the constraint matrix and they are linearly independent, then there is a set of m columns (variables or vectors) which are also linearly independent. Hence, any right-hand side can be expressed in terms of these m columns (called a basis). The simplex method works with these basic solutions, stepping from one to another (by adding one column and removing one column from the basis on each step or iteration), until a solution (called a basic feasible solution) is obtained that meets all of the criteria, including the requirement that all the column values be non-negative. The bounded variable feature allows the user to specify limits on the activity levels for any or all of the variables. Either or both upper and lower bounds may be specified.

After a basic feasible solution is found, the simplex method steps along, examining a series of basic feasible solutions, to find one that satisfies the requirement that the value of the functional (or objective) row be a maximum or minimum; this is called the optimal solution. Not all linear programming problems have an optimal solution. If there is no solution at all in non-negative variables, or none which keeps the variables within their specified bounds, that LP problem is said to be infeasible. If a feasible solution is found, but the constraint rows do not confine the value of the functional row to finite values, the LP problem is said to be unbounded.

In the process of trying to find a feasible solution, row transformations are made on the constraint rows of the matrix. Then an optimal solution is made on the objective function row. The transformed coefficients in the objective function row (also called cost row, or functional row) are the relative cost factors, or d_j's. It has been found advantageous to modify the method to obtain solutions faster. The simplex method uses the d_j's to determine which vector to bring into the basis. The product form of the inverse is used to update this vector in terms of the basis. The product form of the inverse is a means of representing the inverse as a product of matrices of a special form. Actually, only one column of each such matrix is required to represent the matrix. A new matrix (column), called an eta, is produced each iteration. It is used on all succeeding iterations until the matrix is inverted again. When reinversion is done, the etas are started again; this reduces the number of etas (and also the density of the etas) that must be processed during each iteration.

In reinversion, the column selected for pivoting is that which "minimizes the number of coefficients modified at each step (excluding those which are eliminated at the particular step)".* This is essentially a revised tableau simplex iteration, with the pivot choice determined by the number of ensuing operations. The tableau begins with the current basis variables in their original input data form (not updated). The tableau is updated by simplex iterations on the remaining variables, and the selected variable is pivoted and entered as an eta. This method is particularly suited to a data processing machine with high input/output speeds relative to compute speeds.

MACHINE CONFIGURATION AND PROBLEM SIZE

The 1620-1311 Linear Programming System will operate on a 1620 Model 1 or Model 2.

*Bibliography, reference 11.

1620 Model 1

Minimum configuration:

Core storage of 20,000 digits
Indirect Addressing (Feature 4650)
Automatic Divide (Feature 1285)
Additional Instructions (strip, fill, move flag) (Feature 1021)
1622 Card Read Punch
1311 Disk Storage Drive

Optional:

Automatic Floating-Point Operations (Feature 1288)
1443 Printer
Core Storage of 40,000 or 60,000 digits
Additional 1311 Disk Storage Drives

1620 Model 2

Minimum configuration:

Core storage of 20,000 digits
1622 Card Read Punch
1311 Disk Storage Drive

Optional:

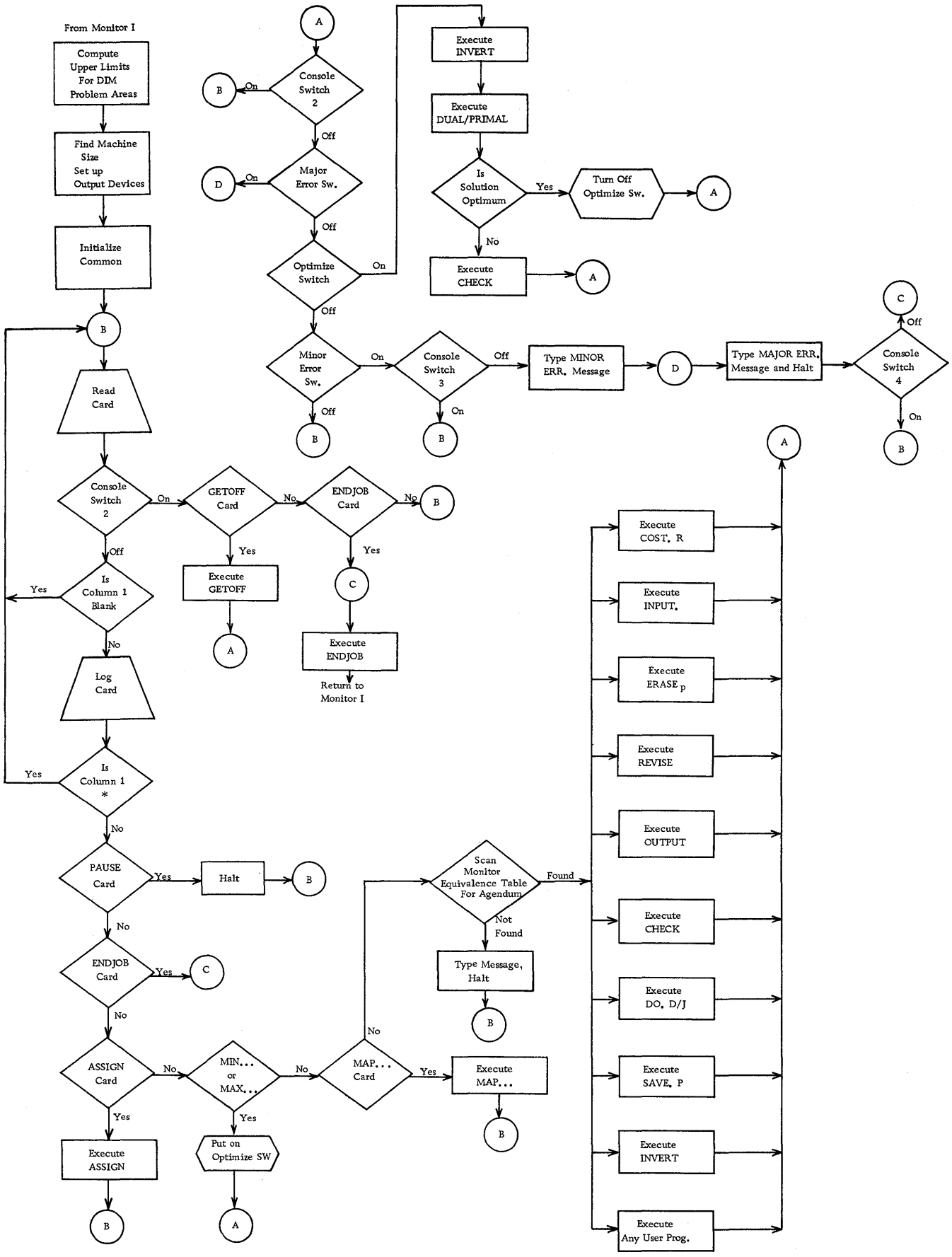
Automatic Floating-Point Operations (Feature 1289)
1443 Printer
Core storage of 40,000 or 60,000 digits
Additional 1311 Disk Storage Drives

Machine Size	Max. No. of Rows	Max. No. of Columns
20K	100	1000
40K	250	2000
60K	400	3000

SYSTEM CONFIGURATION

The source language used for the 1620-1311 Linear Programming System is IBM 1620 SPS II-D. The system is called in under control of the IBM 1620 Monitor I System. (See Bibliography, reference 12.) The I/O operations in the LP System are under Monitor control. At the completion of an LP run, the LP System returns control to Monitor. Therefore, an LP problem can be stacked with other jobs to be run under Monitor.

GENERAL SYSTEM CHART



DATA INPUT, AGENDA AND REPORTS

AGENDA

User control of the 1620-1311 LP system is exercised through special procedure control cards which are known as agendum cards (agenda is the plural). Each agendum card causes a particular program (or set of programs, where program size is a factor) to be read from disk storage into main storage. That program is then executed; it will in turn call for additional data input if required (either from cards or disk storage) or even call other agendum programs. Certain programs will produce selected output on cards for LP solution restart or printed reports for user analysis.

The various agenda accomplish four principal functions; they are explaned in detail in "1620-1311 LP Agenda" in the order listed below:

Data Preparation

INPUT.	Calls the input program to read in data from cards, set up the problem on disk, or to reference existing information on disks. Problems may contain alternate objective functions and multiple right-hand sides.
REVISE	Calls the revise program to change identification information or data values already stored on disk. Changes can be made to the row and column identification files, the matrix and right-hand-side element files, and the stored basis.

Optimization

MAX. . .	Calls the solution program (Dual and Primal with use of Invert) to obtain the optimal solution to produce the maximum value for the selected objective function.
MIN. . .	Calls the solution program (Dual and Primal with use of Invert) to obtain the optimal solution to produce the minimum value for the selected objective function.
INVERT	Calls a program which obtains an inverse of the current basis; this is seldom done by the user since it is handled automatically by 1620-1311 LP.

Report Preparation

OUTPUT	Calls the output program to report the optimal solutions on cards, typewriter, or printer, and gives basic structural and logical variables activity level.
DO. D/J	Calls the programs needed to determine the relative values of structural variables in terms of entering the basis, and indicates the relative value of changing the constraint limits.
COST. R	Calls the programs needed to determine the cost range for each structural variable which would not cause a change in the solution basis.

CHECK.	Calls the check program to determine solution accuracy, or to obtain selected submatrix sums.

Control and Data Maintenance

ASSIGN	Establishes certain control values for mantissa length, inversion frequency, error tolerances, use of output devices, and iteration printout frequency.
ENDJOB	Transfers control back to the 1620 Monitor.
ERASEp	Eliminates the index value which permits the use of a stored basis.
GETOFF	Punches a restart deck when processing needs to be interrupted prior to obtaining an optimal solution.
MAP . . .	Calls a program which lists the size of the problem (number of rows and columns), and the location of the input data on disk.
PAUSE.	Interrupts proper processing at the end of a particular agendum to permit user review and insertion of changes.
SAVE.p	Stores the optimal solution as a basis for later use; the current inverse can also be saved.

DATA INPUT

There is a variety of data inputs required or useful in solving LP problems. The data input cards are the means by which the problem data are placed in disk storage or referenced from previous problems already in disk storage. The data inputs always follow an INPUT. or a REVISE agendum card.

There are input indicator cards, input identification cards, and input value cards. They are discussed in detail under "Data Preparation" in the order listed below:

ROW. ID indicator	Introduces the row identification file.
Row identification	Contains the row name and row type (objective function, equality, less-than, greater-than, or range constraint) for each row in the matrix. Omitted rows are indicated.
COL. ID indicator	Introduces the column identification file. If not used, the 1620-1311 LP System will generate column identification from the matrix element cards.
Column identification	Contains the column names and column usage (active or omitted).
MATRIX indicator	Introduces the matrix element data value file.

Matrix elements	Contains the actual data values and structural variable bounds (upper, lower, or both) for each active or potentially active matrix element. Both the column name and row name are indicated for each value.
FIRST. B indicator	Introduces the first or only right-hand-side element constraint value file.
NEXT. B indicator	Introduces each additional right-hand-side element constraint value file.
Right-hand-side element	Contains the actual limit values for each constraint row, e. g. , the constraint to which the row is equal, or the upper limit, lower limit or range constraint for an inequality.
BASIS. indicator	Introduces an advanced initial basis. Usually recycling of OUTPUT or GETOFF generated cards.
VARBLS indicator	Introduces the basic structural variable file.
Basic structural variables identification	Contains the names of the structural variable (column name) to be included in the initial basis together with a basis entry type to indicate whether it is to be entered at its upper bound or at an intermediate value.
SLACKS indicator	Introduces the nonbasic logical variables files.
Nonbasic logical variable identification	Contains the names of the rows corresponding to the logical variables which are to be omitted from the initial basis. A basis entry type indicates whether the variable left at its lower intermediate or upper bound.
ENDATA or EOF indicator	Concludes the data input files.

REPORTS

The user can receive various reports from the 1620-1311 LP System that provide useful information concerning the solution, the sensitivity of the solution to changes in the values, and accuracy of the results. These reports are produced by the OUTPUT, CHECK. , COST. R and DO. D/J agenda. Under control of ASSIGN, the report information may be on the typewriter, on cards, or on a printer.

The following list summarizes the reports which can be issued; details are covered in "1620-1311 LP Agenda" under "Report Preparation".

OUTPUT	Contains as the first card (or a line of print on the typewriter or printer) a BASIS. indicator card. This deck can be used as an advanced basis in solving subsequent related problems when placed in the input deck since the format is compatible. For the basic structural variables, the activity levels are given. The activity levels are given for basic logical variables, and the simplex multipliers for the nonbasic logical variables.

CHECK.	Gives the row errors, the maximum error (the row error of the row having the largest error), and includes the specified row limit or limits and the solution row activity level.
COST. R	Contains the cost (or profit) interval over which the objective function coefficient may range and the structural variable still remains in the optimal basis. In addition to the upper and lower limit of the range, the report gives the variable that would enter if the upper bound were exceeded (by a cost change), and the change in basis that would occur if the cost were to go below the lower bound.
DO. D/J	Contains the reduced costs (D_j's for nonbasic structural variables and for basic logical variables which are forced into the basis because of upper or lower constraint limits. This report also gives the current cost and the basis value (difference between current cost and reduced cost). Each of the rows in this problem is given the type of row (+, -, =, RANGE), the row name, and the marginal value.
GETOFF	Punches certain output cards which can be used as an initial basis.

USING THE 1620-1311 LINEAR PROGRAMMING SYSTEM

The basic ideas of the IBM 1620-1311 Linear Programming System are easy to learn. This section presents the Linear Programming System as if it were being taught in the classroom. Sample problems are used to illustrate the various features of the system; after each problem there is a discussion of the routines used therein.

This section does not discuss all of the features available with the Linear Programming System. The complete information with all of the options appears later under "Data Preparation" and "1620-1311 LP Agenda". You may choose to refer to the appropriate parts of those sections as you study this section, or you may give this section a reading for the broad outline before becoming concerned with all the details. The latter course is recommended for new linear programming users. The very experienced linear programming user should skim through this section for terminology and general format; he should progress quickly to the succeeding sections.

The Linear Programming System is a high-level, problem-oriented system. It is a very specialized system in that it will solve only linear programming problems. The use of the system involves the same three basic steps as the solution of any problem in data processing. These are: input (concerned with placing the data that is to be processed into memory), compute or process (concerned with processing the data in predetermined ways to arrive at the answers), and output (concerned with recording the answers).

Control of this system is accomplished by the use of control cards, called agendum cards. These cards initiate the execution of routines of the Linear Programming System.

Five examples in this section explain the program features. These examples are based on the sample problem recorded later in the manual. They progressively introduce the various agenda with their options, going from simple formats and uses to the more complex. Each sample problem is legitimate and complete in itself. Given the initial conditions stated for the problem, the problem solution can be keypunched to produce results using the 1620-1311 Linear Programming System. All the information presented here assumes that the user has carefully read the IBM manual "An Introduction to Linear Programming" (E20-8171). This manual contains a glossary which should be used to define any unfamiliar terms.

Before tackling the first sample problem let us review the construction principles underlying the 1620-1311 Linear Programming System:

1. The system has been designed for ease of use and is simple to prepare and modify.

2. The system is disk-oriented so that moderate-size problems can be handled most efficiently; data and solution logic can be saved between runs; reference can be made to previously solved problems.

3. It is efficient to perform multiple runs with slightly modified input data at a fraction of the cost of the original run. Special post-optimal analyses can be performed conveniently to determine the extent to which data input can be changed without requiring major solution changes.

4. All operation is under the 1620 Monitor system.

5. The actual optimization is controlled automatically to produce accurate results efficiently. The sophisticated user may override the automatic controls on mantissa length, tolerances, and inversion frequency through use of ASSIGN and INVERT. (These two agenda are discussed under "1620-1311 LP Agenda" and are not covered here).

6. A large number of input and computational errors are identified and appropriate messages displayed, but the system solution continues with appropriate assumptions unless certain major errors occur.

SAMPLE PROBLEM 1

The following problem is concerned with blending certain raw materials to produce a desired end product. This first sample problem shows how the information can be formulated to describe the situation and placed in disk storage for later solution.

> An aluminum alloy smelter wishes to produce 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 materials. There are five scrap materials which are stored in separate bins; each scrap has been chemically analyzed. The inventory of each bin is known, as is the cost of each scrap material. Two of the bins contain powdered scrap; since these have the desirable property of oxidizing rapidly in the furnace, it is required that at least a certain amount from each of these bins be used in the solution.

The information stated above gives a framework in which to associate the actual problem data.

1. The use of each of the seven raw materials is bounded by the available inventory (an upper bound) and by the minimum required usage (lower bound).

   - Bin 3 has 800 pounds in inventory and must have 400 pounds used for effective oxidation. Therefore bin 3 has an upper and lower bound.

   - The aluminum ingot (a pure material) has an unlimited supply available, and no minimum requirement. Therefore the pure aluminum usage is unbounded.

   A series of mathematical statements is required to explicitly indicate any applicable bounds. It is always implied that the minimum usage is zero unless otherwise stated (e. g., we can't use a negative amount of aluminum).

   $400 \leq \text{BIN3} \leq 800$

   $0 \leq \text{ALUM}$ (need not be stated explicitly since 1620-1311 LP will do this automatically)

   BIN3 and ALUM are the names of the structural variables whose value is the amount of usage that particular raw material will get.

2. The cost of each raw material will be multiplied by the usage of that raw material to determine the total cost of the planned mixture.

   - The material in BIN3 costs $. 17 per pound.
   - Aluminum ingot costs $. 21 per pound. etc.

These values are combined into an objective function statement.

.03 (BIN1) + .08 (BIN2) + .17 (BIN3)

+.12 (BIN4) + .15 (BIN5) +

.21 ALUM + .38 SILCON = VALUE

VALUE represents the value of the objective function

3. A material balance equation states that what comes in must go out. Therefore, the sum of the weight of the raw materials used must equal the end product weight. 2000 pounds of the alloy are required.

BIN1 + BIN2 + . . . + ALUM + SILCON=2000

4. There are six elements whose presence in the alloy is to be controlled. Each raw material has been analyzed as to its composition (by weight) in terms of these elements. The amount of a particular element in the alloy will depend on the usage of each raw material and its content of that element.

- BIN2 has
  - .04 Iron (FE)
  - .05 Copper (CU)
  - .04 Manganese (MN)
  - .03 Magnesium (MG)
  - .75 Aluminum (AL)
  - .06 Silicon (SI)

- The total must be less than or equal to 1.00, since the composition is on a per-pound basis. In this case it is less than 1.00, indicating that other elements are present. The amount of such other elements is not of interest in the solution of this problem.

- There must be no more than 3% Iron (FE) in the end product. Therefore the alloy cannot contain more than 60 pounds of this element.

$$FE \leq 60$$

- It is required that all composition values be zero or positive so unless otherwise stated there is an implied lower constraint of zero. So iron has an upper constraint limit of 60 and an implied lower constraint limit of zero.

- There must be at least 75% of the element aluminum in the alloy.

$$AL \geq 1500$$

- The amount of aluminum actually present is the sum of the amount of the element in each raw material multiplied by its aluminum content.

$$.70\ BIN1 + .75BIN2 + \ldots + .97ALUM \geq 1500$$

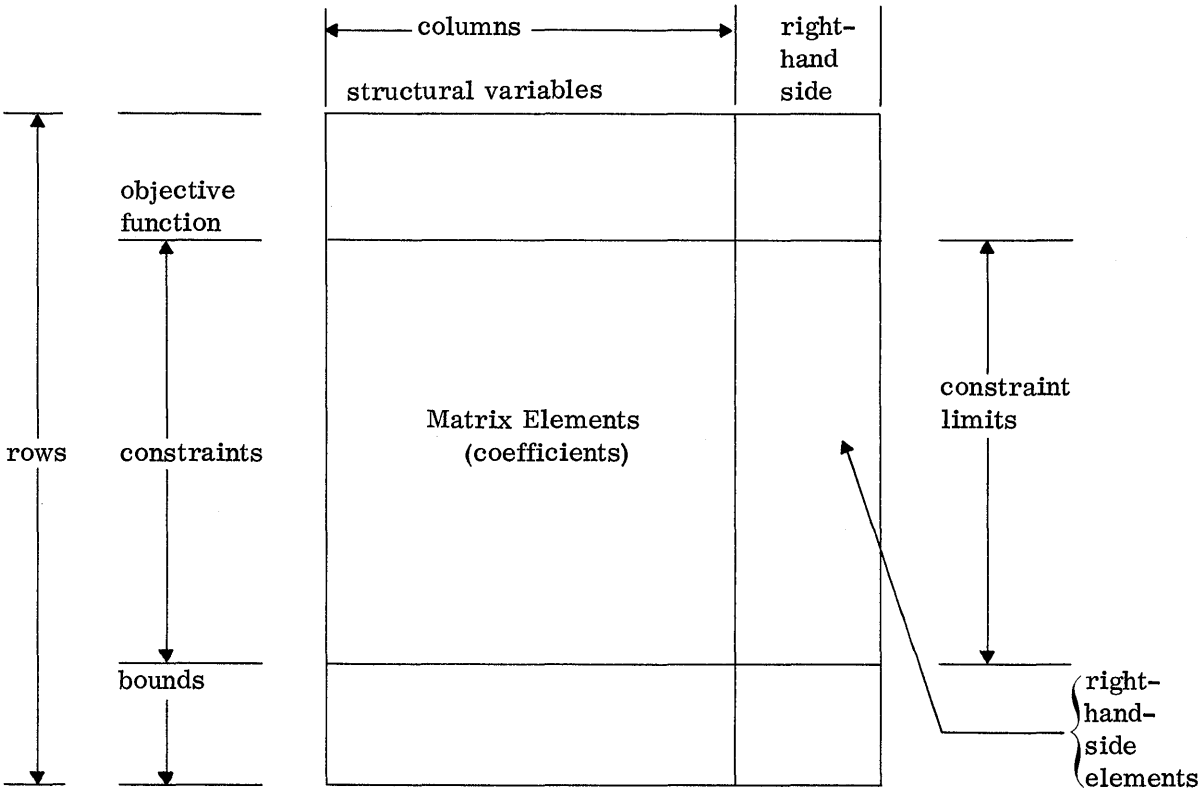
- A similar constraint statement would be written for each element. Some elements might have a range of permissible values, e.g., both an upper constraint limit and a lower constraint limit.

To summarize, these are the following explicit statements required to describe the alloy smelting problem:

- Bounds on the structural variables (the raw materials)
- An objective function (cost of each structural variable)
- Constraint and balance equations or inequalities (these are collectively called constraints)

Let us see how the user places the alloy blending problem in disk. The first step requires the user to write a series of equations and inequalities to express material balance and constraints; he also must write one or more objective functions; he must record the upper and lower bounds for each structural variable. This information then should be represented in matrix form for clarity and compactness. From the matrix the actual data input cards are prepared.

The general form of the matrix is shown below:



To see how this actually works, consider the following matrix, which represents one formulation of the alloy blending problem.

		COLUMN NAME							RHS NAME	
		BIN1	BIN2	BIN3	BIN4	BIN5	ALUM	SILCON	ALOY1	
	ROW NAME									
Obj. function	VALUE	.03	.08	.17	.12	.15	.21	.38		
Constraints	YIELD	1	1	1	1	1	1	1	=2000	Limits
	FE (Iron)	.15	.04	.02	.04	.02	.01	.03	≤ 60	
	CU (Copper)	.03	.05	.08	.02	.06	.01		≤ 100	
	MN (Manganese)	.02	.04	.01	.02	.02			≤ 40	
	MG (Magnesium)	.02	.03			.01			≤ 30	
	AL (Aluminum)	.70	.75	.80	.75	.80	.97		≥ 1500	
	SI (SI (Silicon)	.02	.06	.08	.12	.02	.01	.97	250 to 300	
Bounds	Upper (Inventory)	200	2500	800	700	1500				
	Lower (min req'd)			400	100					

The matrix shows the coefficients of the variables from the equations and inequalities. For instance, look at the row called CU (Copper). These coefficients represent the following inequality:

$$.03\ BIN1 + .05\ BIN2 + .08\ BIN3 + .02\ BIN4 + .06\ BIN5 + .01\ ALUM \leq 100$$

The blank under the column SILCON for this row means that there is no measurable amount of copper in the pure silicon ingot.

So the second step in preparing the data input for an LP solution is to prepare a matrix of coefficients from the various equations and inequalities. This is the time for names of rows, columns, and right-hand sides to be standardized.

The third step is to prepare the data input forms and the fourth step is to keypunch and verify the information from these forms.

The data input forms on the next pages fully represent the information in the preceding matrix. Notice these important conventions:

1. All information must appear in defined columns.

2. All names and data must be written exactly as indicated, with correct spelling and spacing.

3. Letters and numbers must be carefully handwritten to avoid confusion (e. g., the number 0 and the letter O, the number 1 and the letter I, the number 2 and the letter Z). In typing, be careful to avoid confusion between the number 1 and the letter l. To avoid mistakes, periods are often used as spacers instead of blanks (e. g., MIN...)

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. _____ JOB TITLE _____ ANALYST _____ DATE _____ PAGE 1 OF 6

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
ROW.ID						
		VALUE				
	0	YIELD				
	+	FE				
	+	CU				
	+	MN				
	+	MG				
	-	AL				
	+	SI				
COL.ID						
		BIN1				
		BIN2				
		BIN3				
		BIN4				
		BIN5				
		ALUM				
		SILCON				

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. _____ JOB TITLE _____ ANALYST _____ DATE _____ PAGE 2 OF 6

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
MATRIX			.	.	.	
	BIN1	VALUE	.03	.	.	
	BIN1	YIELD	1.	2000.	.	
	BIN1	FE	.15	.	.	
	BIN1	CU	.03	.	.	
	BIN1	MN	.02	.	.	
	BIN1	MG	.02	.	.	
	BIN1	AL	.70	.	.	
	BIN1	SI	.02	.	.	
			.	.	.	
	BIN2	VALUE	.03	.	.	
	BIN2	YIELD	1.	2500.	.	
	BIN2	FE	.04	.	.	
	BIN2	CU	.05	.	.	
	BIN2	MN	.04	.	.	
	BIN2	MG	.03	.	.	
	BIN2	AL	.75	.	.	
	BIN2	SI	.06	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. _____ JOB TITLE _____ ANALYST _____ DATE _____ PAGE 3 OF 6

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
	BIN3	VALUE	.17	.	.	
	BIN3	YIELD	1.	800.	400.	
	BIN3	FIE	.02	.	.	
	BIN3	CIU	.08	.	.	
	BIN3	MIN	.01	.	.	
	BIN3	ALL	.80	.	.	
	BIN3	SI	.08	.	.	
			.	.	.	
	BIN4	VALUE	.12	.	.	
	BIN4	YIELD	1.	700.	100.	
	BIN4	FIE	.04	.	.	
	BIN4	CIU	.02	.	.	
	BIN4	MIN	.02	.	.	
	BIN4	ALL	.75	.	.	
	BIN4	SI	.12	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	
			.	.	.	

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. _____ JOB TITLE _____ ANALYST _____ DATE _____ PAGE 4 OF 6

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
	BIN5	VALUE	.15			
	BIN5	YIELD	1.	1500.		
	BIN5	FE	.02			
	BIN5	CU	.06			
	BIN5	MN	.02			
	BIN5	MG	.01			
	BIN5	AL	.80			
	BIN5	SI	.02			
	ALUM	VALUE	.21			
	ALUM	YIELD	1.			
	ALUM	FE	.01			
	ALUM	CU	.01			
	ALUM	AL	.97			
	ALUM	SI	.01			

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. _____ JOB TITLE _____ ANALYST _____ DATE _____ PAGE 5 OF 6

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
	SILCØN	VALUE	.38			
	SILCØN	YIELD	1.			
	SILCØN	FIE	.03			
	SILCØN	SI	.97			

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. _____ JOB TITLE _____ ANALYST _____ DATE _____ PAGE 6 OF 6

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
FIRST,	BALØY1					
		YIELD,	2000.			
		FE,	60.			
		CU,	100.			
		MN,	40.			
		MG,	30.			
		AL,	1500.			
		SI,	300.	250.		
ENDATA						

Let us review each type of card shown (the b is used to represent a blank or space):

ROW. ID — This is the ROW. ID indicator card, which precedes the row identification cards.

bbbbbbbbbbbbVALUE
bbbbbbbbbbb0YIELD
bbbbbbbbbbb+FE

:

bbbbbbbbbbb-AL
bbbbbbbbbbb+SI — These are the row identification cards. One is required for each row. In column 12 is a code which indicates the type of row:

b (blank) means objective function
0 (zero) means an equality
+ (plus) means a less-than inequality or a range constraint
- (minus) means a greater-than inequality

To determine the row type, look at the right-hand side of the matrix shown earlier. If there is no number, then it is an objective function (put a blank in column 12). If there is an equal sign, put a zero in column 12. If there is a less-than-or-equal-to sign ($\leq$), put a plus (12 punch) in column 12. If there is a greater-than-or-equal-to sign ($\geq$), put a minus (11 punch) in column 12. If there is a range of values (A to B), put a plus (12 punch) in column 12.

The name of the row is also given (e. g., VALUE, YIELD, etc.). The name is placed in a six-position alphameric field with all special symbols permitted except the record mark and group mark.

COL. ID — This is the COL. ID indicator card, which precedes the column identification cards. If the column identification file is omitted, it will be produced automatically by the 1620-1311 LP System.

bbbbbbbbbbbbBIN1

.
:

bbbbbbbbbbbbSILCON — These are the column identification cards. One is required for each column. In the twelfth position on the card may be a code (an asterisk) to indicate that this structural variable should not be included as an active item in the solution of the problem. The name occupies a six-position alphameric field with all special symbols permitted except record mark and groupmark.

MATRIX — This is the matrix indicator card, which precedes the matrix element cards that contain the data values and the bounds of the structural variables.

```
bbbbbbBIN1bbVALUEb+bbbb.03bbbb
bbbbbbBIN1bbYIELDb+bbb1.bbbbbb+b200.bbbbbb
        :
        :
bbbbbbSILCONSIbbbb+bbbb.97bbbb
```

These are the matrix element cards. Each contains the data value for one row-column intersection in the matrix. This data value is the coefficient for one term in one equation or inequality. The matrix element cards must be grouped in column sequence; that is, all data values for a given column must be together (e.g., all rows for BIN1, then all rows for BIN2, finally all rows for SILCON). The row order within a column is not significant and need not be consistent from column to column (the row names are VALUE, YIELD, (....), SI). Obviously, it is safer to record the columns (and rows within the columns) in the same order as in the matrix itself. Only nonzero entries need to be recorded; however, if the user feels he may need to insert a data value at some later time for this row-column intersection, he should prepare a matrix element card with a value of zero. The row names and column names must correspond exactly to the names used in the row identification cards and in the column identification cards. If no COL. ID indicator and column identification cards were used, then the column names which appear in the matrix element cards will be assumed to be the correct names, and a column identification file will be constructed from these names with all of the structural variables considered as active.

Note the recommended format for the data values: ±xxxx.xxxxxx. Either blanks or zeros may be used when the number does not require all the positions, though blanks are more usual. The plus sign is not required for positive numbers, but the minus sign is required for negative numbers. Integers do not require a decimal point. In the "Matrix Element" subsection of "Data Preparation", there are more detailed rules on additional flexibility permitted in data value format.

The matrix element cards have one additional function: they are used to specify the upper and lower bounds of the structural variables. These bounds must both be on the same matrix element card and only one matrix element card should contain the bounds. Many users will find it convenient to consistently use the first (or last) matrix element card for a column to specify the appropriate structural bounds. The format for showing the bounds is the same as for the data values: ±xxxx.xxxxxx with the same construction rules. The upper bound, if any, appears in position 31 - 42 while the lower bound is in positions 43 - 54. For instance, in this matrix element card:

```
bbbbbbBIN3bbYIELDb+bbb1.bbbbbb+b800.bbbbbb+b400.bbbbbb
```

BIN3 is the column name, YIELD is the row name, the matrix element value is 1., the upper bound is 800., and the

lower bound is 400. Unless specified, all lower bounds are implicity zero. Where proper (because of the nature of the equalities or inequalities), either positive or negative bounds may be used (1620 - 1311 LP actually transforms these statements so that the lower bound is effectively zero during calculation and is adjusted correctly before reports are prepared). The upper bound must be algebraically larger than the lower bound. If a negative upper bound is used, a lower bound is required.

FIRST. B ALOY1

This is the indicator card for the first right-hand side. Its name is ALOY1. The FIRST. B indicator card precedes the right-hand-side element cards. If there is only one right-hand side it may be given a name of all blanks (bbbbb).

```
bbbbbbbbbbbbYIELDb+2000. bbbbbb
bbbbbbbbbbbbFEbbbb+bb60. bbbbbb
        .
        .
        .
bbbbbbbbbbbbSIbbbb+b250. bbbbbb+b300. bbbbbb
```

These are the right-hand-side element cards. There is one for each row, except for those rows with a zero limit (or no limit, as in an objective function row). The row names are YIELD, FE, ..., SI. If there is only one limit, it is placed in positions 19-30. If there is a range limit, the upper constraint is placed in positions 19-30 and the lower constraint is placed in positions 31-42. The format for these limits is ±xxxx.xxxxxx, just like the matrix element data values and bounds. The first value may be an equality constraint or a lower limit or an upper limit, if there is only one constraint limit. The relationship (equality, less than, greater than) is specified through the row identification card by the row type code.

ENDATA

This is the data termination indicator card; it concludes the data input.

Now that the principal data inputs have been covered, this information can be placed into disk storage. The following instructions (agenda and data) store the data ready for problem solution:

```
*-*XEQbLP1620
INPUT. C02000ALLOY
```

data input:

```
ROW. ID indicator
row identification (8 cards)
COL. ID indicator
column identification (7 cards)
MATRIX indicator
matrix elements (48 cards)
FIRST. B indicator
right-hand-side elements (7 cards)
ENDATA
```

```
ENDJOB
```

Each of the lines on the LP form is keypunched into an individual card. These cards are inserted in the card reader of the 1620-1311 system. The start button is depressed and three minutes later the eight-row-by-eight-column matrix is stored on disk ready for later use.

Whenever the entire problem data must be read in, a substantial number of cards is required. This is why so much emphasis is placed on disk maintenance of problem data. Each input card is now examined:

++XEQbLP1620	This is the 1620 Monitor program execute card, which instructs the 1620 Monitor to call in from disk storage the Linear Programming Initialization routine. This routine sets up a common communication area for use by the various LP routines. The Initialization routine in turn calls the Linear Programming Supervisor routine, which is the control or executive program for all other LP routines. The LP Supervisor is then ready to read another card. The exact form shown must always be used to initiate an LP solution. The ++XEQb indicates a program execute instruction (in the 1620 Monitor language) where LP1620 is the established name for the LP program. The 1620 Monitor has a record of where the LP program is stored.
INPUT. C02000ALLOY	This is the first LP agendum card. It serves to call in the routine which will set up the input data for problem solution. INPUT. is the agendum name; all agendum names are six characters long, so the period is used to make this name fill out the full space allotted. The C indicates that the data for this problem is on cards: it is to be stored on disk starting at location 02000 (a five-position numeric field). The name of the problem stored at 02000 is ALLOY; the problem name occupies a six-position alphameric field.
data input deck	These 75 cards contain the entire problem data which is to be stored for later solution.
ENDJOB	The final agendum card indicates to the LP Supervisor routine that the LP problem has now been solved, that the appropriate internal storage areas should be cleared and that control should be returned to the 1620 Monitor program. The monitor then is ready to initiate processing of whatever job is next in the card reader. The problem data, of course, remains in disk storage for use at some later time. This termination card is used at the end of all LP processing; it should not be used between different LP jobs.

SAMPLE PROBLEM 2

The LP program has been stored on a 1311 under control of the 1620 Monitor. The data and control information for the particular LP problem has also been stored starting at disk location 02000. This problem is concerned with blending certain raw materials to produce a desired end product. On disk have been stored (1) the data concerning the available inventories of the raw materials and their composition, (2) the demand for the end product, and its composition constraints, and (3) the cost of each available raw material (ingredient). The purpose now is to solve this problem using linear programming: to determine the best

(optimum) combination of ingredients to use, to produce the end product, within the stated constraints at a minimum cost.

The following five instructions will accomplish this goal:

```
ǂǂXEQbLP1620
INPUT. D02000ALLOY
MIN...
OUTPUTbbbbbbbbbbbb23
ENDJOB
```

ǂǂXEQbLP1620 This is the 1620 Monitor program execute card, which brings into the main memory the first LP routine.

INPUT. D02000ALLOY The INPUT. agendum card indicates that the data for this problem has already been stored on disk starting at location 02000. The original problem name was ALLOY.

MIN... This agendum card calls in the routines which will develop an optimum solution to the LP problem stored at 02000. The solution is a two-step process: first, a feasible solution is found, e. g., one in which all the constraints and bounds are satisfied; second, the optimal solution is determined, e. g., one which is feasible and for which the costs are minimized. Costs may be considered in dollars, process time, or any other suitable measure of value.

OUTPUTbbbbbbbbbbbb23 The third agendum card calls in the output reporting routine from disk. This will cause appropriate output cards to be punched so that a report on the solution to the LP problem can be printed. These output cards state specifically how much of each raw material is used in producing the desired end product. The number 2 indicates that these answers will be given to two decimal places, e. g., ±xxxx.xx . In addition to the raw materials usage (those materials that are used are called the basic structural variables), the output routine also produces a card for each constraint row indicating the amount by which the end product differs from the constraint limitation on that row; e. g., if the amount of iron must be less than 60 pounds out of 2000 pounds of end product, and only 40 pounds were used, then there would be a card which would note that the end product contained 20 pounds less than the allowed limit. The number 3 on the agendum card means that these values will be shown to three decimal places, e. g., ±xxxx.xxx . A card is also punched for the objective function row giving the cost of the optimum solution.

ENDJOB This agendum card turns control back to the 1620 Monitor.

For this simple eight-row-by-eight-column example the 1620 LP System produces 17 cards stating the solution to the stored problem. The cards are so arranged as to provide appropriate headers for the columns of data. The answers can be listed with a regular 80-column print board on a 407 or by a listing program on a 1440, 1401 or other appropriate machine. Through use of an ASSIGN agendum card (see "1620-1311 LP Agenda") the answers can be printed on the console typewriter (practical only for very small problems) or listed by the 1620 on an online printer.

While the problem is being solved, certain information is printed at the console typewriter. The 1620 Monitor notes operating data, while the LP supervisor lists each agendum card and key internal LP operations, like assignment of mantissa length and tolerances, use of the optimization and inversion routines, and, of course, minor or major errors encountered.

The agenda which have been covered in Sample Problem 2 are the fundamental agenda needed to prepare, solve, and present the results of an LP problem. While many special options and additional agenda are still to be covered, the routines called in solving problem 2 are the workhorses. These routines are always required to solve an LP problem.

SAMPLE PROBLEM 3

In Sample Problem 3 the matrix element values to be used are stored on disk from a previous problem. The row names are used as an index to get at the matrix information efficiently. A new alloy is to be made which has different constraints from the previous alloy. It is desired to find the minimum cost (solution) for making this new alloy. It is also desirable to be in a position to rapidly reanalyze the solution with changes in costs or constraints. The following instructions will solve this problem.

```
                ++XEQbLP1620
                INPUT. C03000ALLOYA
                ROW. IDD02000ALLOYA
                MATRIXD02000
                FIRST. BALOY2
                b__________bVALUE
data            b__________bYIELDb-D2000bbbbbb
input           b__________bFEbbbb-Dbb60
                          CU        140
                          MN         40
                          MG         40
                          AL       1500
                          SI        300          bbb250
                ENDATA

                MIN....ALOY2VALUE
                OUTPUTbbbbbbbbbbbb23
                SAVE. B. 10000ALLOYX
                ENDJOB
```

Quite a few cards are required in this problem since some of the input data is being specified explicitly rather than just through reference to previously stored information on disk. Notice that agendum cards are combined in the same deck with data cards and that the order of the cards is significant. Let us review each card to note its format and usage.

++XEQbLP1620 — The same 1620 Monitor program execute card which initiates the LP system.

INPUT. C03000ALLOYA — The INPUT. agendum card calls in the input routine. In this case the C indicates that some input information is on cards and will directly follow the INPUT. agendum card. The user wants the problem stored beginning at location 03000 and the problem name to be assigned is ALLOYA.

ROW. IDD02000ALLOY — Following an INPUT. C agendum card there will be data input cards. Since different kinds of information can be inserted, it is necessary to have an indicator card to tell what kind of data will follow. ROW. ID is the name of an indicator card which precedes the row identification cards (i. e., the cards containing the names of the objective function and constraint rows). In this

case the D02000 indicates that the row names are already stored on disk at location 02000; the problem name where they are stored is ALLOY. The name will be used as a double check before the new problem picks up the information from the referenced problem. The row names are used to index the matrix element values and right-hand-side element values.

MATRIXD02000ALLOY — This is another input indicator card. MATRIX is the card name. The D02000ALLOY indicates that the matrix element values are already stored on disk at location 02000 and the referenced problem name is ALLOY. The matrix element values referenced (e. g., the cost coefficients and the raw material compositions) are those that will be used in solving the new problem. The information is not recopied; only appropriate linkages are made to the data.

FIRST. BALOY2 — The third indicator card is called FIRST. B. This is the card which introduces the right-hand-side element values. The B has been used historically to refer to these right-hand-side values; they are frequently called B values. ALOY2 is the name given to this special column. The right-hand-side name must be only five alphameric characters.

b____bVALUE, etc. — These are the right-hand-side element cards. They give the actual constraint limits for each row in the matrix. The row name in the first card is VALUE: a six-position alphameric name is permitted. In the second card the name is YIELD. The 2000 indicates that the row called YIELD has a constraint limit of 2000. The row YIELD takes the following form:

BIN1 + BIN2 + BIN3 + BIN4 +
BIN5 + ALUM + SILCON = 2000

So in this case the limit is an equality; i. e., the sum of the weights used of each raw material must equal the required weight of the end product.

The third through sixth rows (FE, CU, MN, and MG) also have constraint limits (60, 140, 40, 40 respectively). The constraint formula for FE takes this form:

.15(BIN1) + .04(BIN2) + .02(BIN3) + .04(BIN4)
+ .02(BIN5) + .01(ALUM) + .03(SILCON) $\leq$ 60

The 60 in this case is an upper constraint limit. The same is true for the next three rows covered (CU, MN and MG).

The next row, AL, states a lower constraint limit, 1500; i. e., the amount of aluminum must equal or exceed 1500 pounds.

Notice that regardless of whether the constraint limit is an equality, an upper constraint limit, or a lower constraint limit, the value is always placed in the same place on the right-hand-side element card.

In the last row there is a different situation. For SI, two values are given: 300 and 250. The 300 represents the upper constraint limit for SI, while 250 represents the lower constraint limit. This is called a range constraint. In the rows with upper limits a lower limit of zero is implied. If there is an explicit upper and lower limit, the upper limit is placed first and the lower limit second.

The field used for stating the limits is twelve positions wide. It is recommended that the following format be consistently followed:

±xxxx.xxxxxx

If the value is positive, no sign is required although a plus sign is often useful. For integer values the decimal point is not needed, though it too is often used. This arrangement tends to keep the values in a reasonable range for most efficient LP solution.

ENDATA — The indicator card which states the input data is now concluded.

MIN....ALOY2VALUE — MIN... is the agendum name which calls in the routine to determine the optimum mix of raw materials. In this case ALOY2 refers to the right-hand-side name of the values being used. If there were different alloys being smelted, it would be necessary to state which particular set of constraint limits should apply to the current problem solution. VALUE is the name of the objective function row used in evaluating the various possible solutions to pick the optimum. Since alternate objective function rows may be used to state different cost criteria, the particular objective function row to be used in this problem solution must be named.

OUTPUTb b23 — The OUTPUT agendum card produces the same types of cards as indicated in Sample Problem 1: a card for each structural variable and one for each row indicating the difference between total usage and the constraint limits (to two decimal places).

SAVE. B. 10000ALLOYX — This is another agendum card. It is called SAVE. B to note that the user wishes to retain the current solution information in disk storage for possible later use. This solution is known as a basis; if relatively minor revisions are made to the matrix elements or to the right-hand-side values or to the objective function, a previous basis can provide a head start in solving the new problem. 10000 is the location where the problem information and the basis is to be stored, and ALLOYX is the name given to this stored version of the problem.

Essentially, the output data is stored to form a possible starting basis.

ENDJOB — This agendum card concludes the LP solution process and returns control to the 1620 Monitor.

This sample problem has shown how some data from a previous problem may be used in conjunction with new constraint limits to set up a new problem. It has also indicated how data is entered into disk storage for later use.

SAMPLE PROBLEM 4

Usually the LP user is working with a previous problem, but has to make revisions to reflect changing inventories, costs, material composition, or constraints. 1620-1311 LP has been designed to allow changes to be made readily. The fourth sample problem illustrates how some of these changes can be introduced.

After analyzing the ALLOYA solution from Sample Problem 3, the aluminum smelter wants to determine whether it will pay to purchase 2000 pounds at $. 19 per pound or 2500 pounds at $. 18 per pound of a new scrap raw material; he is given its composition. He wishes to consider producing the original alloy (from Sample Problem 2) using the new material as a substitute for BIN5.

He will examine the cost and usage of the new material. The problem is solved with the following set of input cards:

```
++XEQbLP1620
INPUT.D02000ALLOY
REVISE
MATRIX
b_____bBIN5bbVALUEbbbbbb.19bbbbbb2000
      BIN5    FE              .05
      BIN5    CU              .08
      BIN5    MN              .01
      BIN5    MG              .01
      BIN5    AL              .84
      BIN5    SI              .01

ENDATA
MIN....ALOY1VALUE
OUTPUTb         b23
SAVE.B
REVISE
MATRIX
b      bBIN5bbVALUEbbbbb.18bbbb2500
ENDATA
MIN....ALOY1VALUE
OUTPUTb       b23
ERASEBD02000ALLOY
ENDJOB
```

++XEQbLP1620 — The 1620 Monitor program execute card for 1620-1311 LP.

INPUT.D02000ALLOY — The INPUT. agendum card sets up the problem which is stored on disk at 02000 for re-solution.

REVISE — The agendum card, REVISE, serves to bring in a routine which permits the introduction of changes to the information stored for a particular problem. Only information in storage can be changed. No new information (new rows or columns, or additional elements) can be added. REVISE must come only after

an INPUT. It always revises the problem currently set up for solution.

MATRIX — The MATRIX indicator card introduces the changed matrix element values. Only elements already in disk storage can be modified.

b_____b BIN5bbVALUEbbbbbb. 19bbbbbbbbb2000, etc.

These are the matrix element cards. BIN5 is the structural variable. VALUE is the name of the objective function row. The element value is .19. The upper bound on this structural variable is 2000, i.e., the quantity available to be purchased. If there were a lower bound it would follow the upper bound. The element value and the upper and lower bounds are all shown in the following form:

±xxxx.xxxxxx

The bounds for a single structural variable are shown on one matrix element card.

The other matrix element cards state the composition of the proposed raw material; one card is required for each nonzero entry in a constraint row.

ENDATA — This indicator card ends the data input for the REVISE agendum.

MIN....ALOY1VALUE — The MIN...agendum card optimizes the revised problem using the right-hand side called ALOY1 and the objective function called VALUE. The optimization starts from a previously saved basis to speed up the solution. This is not indicated explicitly, but MIN... will always start from any basis stored with the problem.

OUTPUTb b23 — The OUTPUT agendum prepares the reports on raw material usage and total costs.

SAVE.B — The SAVE.B agendum stores the new solution basis with the current problem for later revise.

REVISE — The REVISE agendum is used again to make further changes in the actual material values. This is done in order to compare the usage of the BIN5 material at $.18 per pound and an availability of 2500 pounds with the 2000 pounds at $.19 per pound.

MATRIX — The MATRIX indicator card is used to introduce the value change for the matrix elements.

BIN5bbVALUEbbbbb. 18bbbbbb2500

The matrix element card changes the data value for the BIN5 column in the row named VALUE (the objective function row); the data value is set at $.18 pound. The upper bound is also changed for this structural variable to show an availability of 2500 pounds.

ENDATA — The data input termination indicator card ends the revised data.

MIN....ALOY.1VALUE — The revised matrix is now optimized. The same right-hand side, ALOY1, and the same objective function, VALUE, are used. However, since the cost in the objective function row has been changed, it is possible that a different answer will result.

OUTPUT.b_____b23 — The OUTPUT agendum prepares another report on raw material usage. The cost of this solution will be compared with the previous solution to indicate the advisability of purchasing either 2000 or 2500 pounds of the new raw material that is available. These results will also be compared with the previous solution (with the old BIN5 material) to decide whether the raw material should be purchased at all.

ERASEB.02000ALLOY — The ERASEB agendum is the means by which a previous solution basis can be eliminated from disk storage. Any inverse that has been stored will of course also be deleted. This is done only when the user expects that there will be such a significant change in the right-hand-side or matrix elements that it would be better to start over again rather than use the previous results. If the basis is erased, the user cannot insert a revised basis. He would have to go through a normal input procedure in order to get an initial basis.

ENDJOB — This agendum card ends the solution of this program and returns control to the 1620 Monitor.

SAMPLE PROBLEM 5

Often the most valuable aspect of using a good LP system is the ability to gain substantial insight into alternate solutions or sensitivity of the solution to changes in cost, composition or constraints. For efficiency, this type of analysis (sometimes called parametric programming) can be handled through special agenda cards. These cards can be used only after a MIN... or MAX... has been carried out.

Let us assume that the alloy smelter decided to buy 2,500 pounds of the new raw material at $.18 per pound. He now wants to examine in depth the use of this new raw material. The following instructions are one approach to this problem:

```
++XEQbLP1620
INPUT.D02000ALLOY
MIN...
COST.R
DO.D/J
CHECK
ENDJOB
```

Each of these cards will be reviewed:

++XEQbLP1620 — This is the 1620 Monitor program execute card.

INPUT.D02000ALLOY — The input agendum card sets up the problem stored at 02000 (with the problem name of ALLOY) for solution.

MIN...	The minimum value solution is determined.
COST.R	This agendum card calls in the program which determines the cost ranges for the structural variables which are in the optimal solution (called the optimal basis). For each such variable, COST.R determines the highest and lowest values of the objective function coefficient which do not require any change in the solution activity level. In other words, this is the range over which the cost of the structural variable may change without changing the solution. The report also indicates what would happen if the cost of each variable were raised or lowered so as to be outside the stated cost range; it indicates which other variable would be affected by the change. This report, in effect, tells the user that as long as the costs remain within the stated boundaries he need not resolve the problem since he knows that the present answer will still be satisfactory.
DO.D/J	This agendum card calls in a program which will provide information on the economic effect of changes in the bounds and limits on certain structural variables and the rows. For each structural variable in the solution at its upper bound, it indicates how much the objective function value could be improved if the upper bound were increased one unit. It also indicates for each structural variable in the solution at its lower bound how much the objective function value could be improved if the lower bound were reduced one unit. For each row, the report indicates the additional profit to be obtained by increasing the upper constraint limit; similarly, the report states the value of decreasing the lower constraint limit. The DO.D/J report gives the user substantial additional insight into the sensitivity of the solution to changes in the bounds and limits.
CHECK.	This agendum card calls in a program which checks the accuracy of the output which has been obtained. It does this by comparing the calculated sum for each row (coefficients times activity levels) and determining the difference between this total and the constraint limit. The largest row error is identified. This routine is used automatically in computing to make sure that error limits are not being exceeded. It may also be called for by the user to assist him in judging whether or not restarting or rescaling some rows may not provide an accurate solution in less time.
ENDJOB	This agendum card indicates the end of the LP work and turns control back to the 1620 Monitor.

DATA PREPARATION

The 1620-1311 Linear Programming System expects two input streams -- the control stream and the data stream. The control stream contains agendum cards which direct the LP system in processing a particular problem. The data stream contains the data values to be operated on. Both types of input are introduced in the card reader.

There are three main types of data input cards:

- Indicator cards that announce the type of data following
- Identification cards that either state the names of rows and columns used in the particular problem or indicate the structural variable to be incorporated in the initial solution basis
- Actual data value cards that introduce the values of matrix elements and right-hand-side elements

The method of presentation is to describe the format and usage of each indicator card and then the identification cards and data value cards which follow it. The indicator cards are described in the recommended input sequence.

To solve a linear programming problem, it is required that at least ROW.ID, MATRIX, FIRST.B, and ENDATA (or EOF) cards are all present in that order. The other indicator cards are optional. If the problem is concerned only with preparing output from a previously solved problem, then only a ROW.ID or a COL.ID and an ENDATA card are needed.

The following states the sequence in which the data input cards are discussed:

ROW.ID indicator

Row identification

COL.ID indicator

Column identification

MATRIX indicator

Matrix elements

FIRST.B and NEXT.B indicators

Right-hand-side elements

BASIS. indicator

VARBLS indicator

Basic structural variables identification

SLACKS indicator

Nonbasic logical variables identification

ENDATA or EOF indicator

The 1620-1311 LP data input forms have been designed to be compatible with many currently used linear programming systems:

1. SHARE standard matrix element cards will be accepted.

2. 7040/7044 Linear Programming System (7040-CO-08X) data input decks will be accepted with four exceptions:

   a. On row identification cards 7040/44 LP permits a blank or zero for type of constraint for both equality rows and objective function rows. In 1620-1311 LP each objective function row must have a blank in col. 12, and each structural row with an equality constraint must have a zero in col. 12.

   b. 7040/44 LP permits up to five row names per row identification card; only one row name per card is permitted in 1620-1311 LP.

   c. The 7040/44 LP basis cards have a completely different format and cannot be accepted.

   d. Certain data input cards in 7040/44 LP are not acceptable to 1620-1311 LP: SUM, CNT, CURTAIN, PARTITION, and SLACK.

3. 1410 Linear Programming System (1410-CO-01X, 06X, 07X, 09X) matrix element cards will be accepted.

1620/1311 LP data input will be accepted by the 7040/7044 LP system if:

a. The cost rows are first in the ROW.ID deck.

b. No bounds are used.

c. No COL.ID's are used.

d. No period appears in ROW.ID, FIRST.B, NEXT.B.

e. The first characters of the row and column names are blank or numeric.

f. The right-hand-side names are numeric, and in ascending, but not necessarily consecutive, sequence.

g. No basis indicator or identification cards are used.

Since the data values are stored in a compressed form in disk storage, it is necessary to identify all matrix elements and right-hand-side elements that are expected to be used in later solutions of the problem. For instance, if the problem may need to be solved using different (but not yet defined) objective functions, extra objective function rows should be inserted so that appropriate values can be established later -- similarly with alternate right-hand sides or initially zero-valued matrix elements.

Two of the agendum cards produce output cards which can be recycled through the system. These agenda are OUTPUT and GETOFF. Some of the information on these recycled cards is ignored by the INPUT. and REVISE agenda.

For a more complete discussion on how the data input cards are used to set up a problem for linear programming solution, refer to INPUT. and REVISE under "1620-1311 LP Agenda".

A comments card which appears within a data input deck will not be typed.

ROW.ID INDICATOR

Format

	Indicator Name	Input Data Source	Starting Sector Location	Problem Name	Comments
cc	1 6	7	8 12	13 18	19 80
	ROW.ID ROW.ID	D	sssss	nnnnnn	

Indicator Name: ROW.ID; the period in column 4 is optional.

Input Data Source: b (blank) indicates that row identification cards will follow.
D indicates that the row identification file should be referenced from a previous problem.

Starting Sector Address: A five-position numeric field which identifies the established location for the problem containing the row identification information desired.

Problem Name: A six-position alphameric field containing the problem name which is checked against the problem name stored at the referenced sector location.

Comments: Any desired information

Usage

The ROW.ID indicator card introduces the row names which are on the row identification cards or in disk storage.

Only one ROW.ID card may be used per INPUT. card. It is always required to follow an INPUT.C card if MIN... or MAX... are to be executed for the problem. The ROW.IDb card must always be followed by the appropriate row identification cards. If there is a MATRIX card there must be a ROW.ID card preceding it. If ROW.IDD is used, the current problem refers to information stored with a previous problem. If that previous problem is destroyed (through writing a new problem over it), the second problem will not be usable.

ROW IDENTIFICATION

Format

	Blank	Row Type	Row Name	Comments
cc	1 11	12	13 18	19 80
	b b	t	rrrrrr	

Blank: Must be all blanks

Row Type: The type of logical variable to be generated for this row:

- b (blank) for objective function
- 0 (zero) for equation
- + (12 punch) for a less-than-or-equal-to (≤) relation or a row with both an upper and lower constraint.
- - (11 punch) for a greater-than-or-equal-to (≥) relation or a row with both an upper and lower constraint.
- * (asterisk) for an omitted row (it can be revised later).

Row Name: A six-position alphameric field with special characters (other than record mark and groupmark) permitted. Blanks are significant characters; therefore, it is recommended that a period be used for internal spacing, for example, OB.ROW, not OBbROW. The names should all be left-justified.

Comments: Any desired information

Usage

The row identification cards are used to define the type of row relationship (equality, less than or equal to, etc.), thereby specifying the type of logical variable (slack or artificial) to be generated. They are also used to name the rows (both objective function and constraint) for user reference. The row identification cards permit the selection of rows from a matrix by asterisking the unwanted row identification cards; this will omit them from the matrix. The order of row entry is immaterial, and objective function rows may be interspersed with constraint rows.

A row identification card may be omitted rather than specifying an asterisk for it. This avoids using unnecessary storage. However, a row which is not defined, rather than being defined with an asterisk as an omitted row, cannot be later entered under the REVISE agendum.

One row identification card is used for each row in the full matrix; every row must have a row identification card containing its row name. Omission of a row identification card will cause a minor error message when a matrix element card is read in for that row.

The row type is a method for having 1620-1311 LP generate the artificial and slack columns by creating logical variables. These make it an equality; they are called artificial variables when placed in the objective function or in a constraint equality. They are required to provide the basis (one column for each row) needed to start the linear programming solution process. An artificial variable can never re-enter the basis once it has left.

Row type	Col. 12 code	Logical variable generated
objective function	b (blank)	a free artificial unrestricted as to sign
equality constraint	0 (zero)	a zero-valued artificial
less-than constraint $[A_1X_1+\ldots \leq B]$	+ (12 punch)	a positive slack $[A_1X_1+\ldots+S=B]$ where $S \geq 0$

Row type	Col. 12 code	Logical variable generated
greater-than constraint $[A_1X_1+\ldots \geq B]$	- (11 punch)	a negative slack $[A_1X_1+\ldots -S=B]$ where $S \geq 0$
range constraint	+ (12 punch)	treated as a positive slack with upper bound specified
	or	
	- (11 punch)	treated as a negative slack with upper bound specified
omitted row	* (asterisk)	no logical variable

COL.ID INDICATOR

Format

cc	Indicator Name	Input Data Source	Starting Sector Location	Problem Name	Comments
	1 6	7	8 12	13 18	19 80
	COL.ID COL.ID	D	sssss	nnnnnn	

Indicator Name: COL.ID; the period in column 4 is optional.

Input Data Source: b (blank) indicates that column identification cards will follow.
D indicates that the column identification file should be referenced from a previous problem.

Starting Sector Address: A five-position numeric field which identifies the established location for the problem containing the column identification information desired.

Problem Name: A six-position alphameric field containing the problem name which is checked against the problem name stored at the referenced storage location.

Comments: Any desired information

Usage

The COL.ID indicator card introduces the column names which are on the column identification cards or in disk storage. Only one COL.ID card may be used per INPUT. card. The column identification file (COL.ID indicator card and column identification cards) is strictly optional. It is required only if the user wishes to eliminate certain columns from the matrix for a particular problem solution. If there is a COL.ID indicator card, every column must have a column identification card. If no column file is introduced, the 1620-1311 LP system will make one using the names in the matrix element files.

If COL.IDD is used, the current problem refers to information stored with a previous problem. If that previous problem is destroyed (through writing a new problem over it), the second problem will not be usable.

COLUMN IDENTIFICATION

Format

	Blank	Column Usage	Column Name	Comments
cc	1 11	12	13 18	19 80
	b b	t	ccccccc	

Blank: Must be all blanks

Column Usage: b (blank) indicates that the column is to be used; it is an active structural variable.

* (asterisk) indicates that the column is not to be used; it is omitted from the solution; it can be revised later to be considered in the solution.

Column Name: A six-position alphameric field with special characters (other than record mark and groupmark) permitted. Since blanks are significant characters, it is recommended that a period be used for internal spacing, for example, ST.COL, not STbCOL. For consistency and accuracy the names should be all left-justified.

Comments: Any desired information

Usage

The column identification cards are used to select certain columns (or structural variables) which will be in the current matrix. This permits the solution of various problems from one large matrix. The order of column entry is immaterial. Column identification cards are also used to set for selective analysis reports.

One column identification card is used for each column in the matrix; if the column identification file (COL.ID indicator card and column identification cards) is used, every column in the matrix must be represented. Omission of a column identification card will cause a minor error message at the time that a matrix element card is read in for that column.

The use of the asterisk makes it convenient to omit certain columns. By replacing the asterisked card with a blank card, the column can be included in the matrix.

Note that column identification cards are not needed for the automatically generated logical variables, nor are they used for the right-hand sides.

MATRIX INDICATOR

Format

	Indicator Name	Input Data Source	Starting Sector Location	Problem Name	Comments
cc	1 6	7	8 12	13 18	19 80
	MATRIX MATRIX	D	sssss	nnnnnn	

Indicator Name: MATRIX

Input Data Source: b (blank) indicates that matrix element cards will follow
D indicates that the matrix element data values are to be referenced from a previous problem

Starting Sector Location: A five-position numeric field which identifies the established location for the problem containing the desired matrix element data values.

Problem Name: A six-position alphameric field containing the problem name which is checked against the problem name stored at the referenced storage location.

Comments: Any desired information

Usage

The MATRIX indicator card introduces the matrix data values which are on the matrix element cards or in disk storage. Only one MATRIX card may be used per INPUT. card. The MATRIX file (MATRIX indicator card and matrix element cards, or MATRIXD indicator card) is required if a problem solution is to be obtained. If MATRIXD is used, any change to the referenced matrix may make the current problem unsolvable.

MATRIX ELEMENT

Format

	Blank	Column Name	Row Name	Data Value	Upper Bound	Lower Bound	Comments
cc	1 6	7 12	13 18	19 30	31 42	43 54	55 80
	b b	cccccc	rrrrrr	±xxxx.xxxxxx	±yyyy.yyyyyy	±zzzz.zzzzzz	
	b b	cccccc	rrrrrr	±xxxx.xxxxxx	±yyyy.yyyyyy		
	b b	cccccc	rrrrrr	±xxxx.xxxxxx		±zzzz.zzzzzz	
	b b	cccccc	rrrrrr	±xxxx.xxxxxx			

Blank: Must be all blanks

Column Name: A six-position alphameric field containing the column name. Blanks are significant characters. Record mark and groupmark cannot be used. It is recommended that the name be left-justified and that periods be used for internal spacing.

Row Name: A six-position alphameric field containing the row name as included in the row identification file

Data Value: A twelve-position field permitting a ten-position numeric value with space for a sign (optional for positive numbers) and a decimal point. The format shown is recommended since it is used for SHARE standard input and 7040/7044 LP system (7040-CO-08X). The recommended format aids in obtaining reasonable scaling of values for more accurate solutions. All blanks are interpreted as zeros. There is no checking for alphabetic information in any data field. The user may vary the construction of this field provided the following rules are observed:

- One to ten contiguous digits with or without a decimal point. The decimal point may appear prior to the first digit, between two digits, or after the last digit. The decimal point is not required for an integer.

- A minus sign (11 punch) is required for all negative numbers. The plus sign (12 punch) is optional for positive numbers. The sign may appear anywhere before the first digit or after the last digit and need not be contiguous with the number. In an objective function row, cost data values are usually entered as positive numbers and profit data values entered as negative numbers, but they may be reversed if the user is careful to use MAX... instead of MIN...

Upper Bound: A twelve-position field permitting a ten-position numeric value with space for a sign and a decimal point. Same rules for construction as in data value. Usage is optional. An asterisk in the first position will delete a previously stored value.

Lower Bound: A twelve-position field permitting a ten-position numeric value with space for a sign and a decimal point. Same construction rules as for data value. A lower bound is required if there is a negative upper bound; otherwise usage is optional. An asterisk in the first position will delete a previously stored value.

Comments: Any desired information

Usage

There are four variations of matrix element cards: data value only; data value and upper bound; data value and lower bound; data value and both upper and lower bound. Every matrix element with a nonzero data value (coefficient) must have a matrix element card. Every matrix element which the user may wish to have inserted in the matrix at a later time must have a matrix element card when the problem is set up. The principal purpose of the matrix element cards is to establish the coefficients for the matrix to be solved. To do this it is required that the matrix element cards be arranged in column name sequence; for example, all matrix elements cards for a given column must be consecutive. The order of the rows within the column is not significant. All constraint rows and objective function rows are entered. The right-hand sides are not entered on the matrix element cards.

A structural variable (i.e., the column name refers to the structural variable) may have its activity level bounded by the user:

$$Z_1 \leq X_1 \leq Y_1$$

where: X_1 represents a particular structural variable.
Y_1 represents an upper bound on that structural variable.
Z_1 represents a lower bound on that structural variable.

The 1620-1311 LP System takes any explicit lower bound and adds it to the value of the structural variable and to its upper bound (if any):

$$\text{if} \quad Z_1 \leq X_1 \leq Y_1$$

$$\text{then} \quad 0 \leq X_1' \leq (Y_1 + Z_1) \quad \text{where } X_1' = X_1 + Z_1$$

and the amount Z_1 times the matrix element data value is added to the right-hand side for each constraint in which that structural variable occurs.

Both lower and upper bounds may be negative or positive, but the lower bound must, of course, be algebraically less than the upper bound (e.g., $Z_1 \leq Y_1$); however, the system does not check for this.

Only one matrix element card for a given column name should contain bounds. However, the bounds may appear on any one card for a particular structural variable. The first matrix element card for a column which contains a bound is assumed to correctly define both bounds for that structural variable. If no bounds are used, the variable is treated as having a lower bound of zero and no defined upper bound.

If just the upper bound is used, it must be a positive value since the lower bound is assumed to be zero and the upper bound must be greater than the lower bound. If a negative upper bound is used, a negative lower bound is also required.

If just the lower bound is used, the structural variable is treated as having no defined upper bound. The lower bound may be either positive or negative.

If both bounds are used, both may be negative, both positive, or the lower negative and the upper positive. Extreme care must be taken in establishing upper and lower bounds to ensure that the signs are properly set. Normally, negative bounds are used only when there are both sending and receiving types of activities (receipts and disbursements, loans and payments, deposits and withdrawals). In these cases the user should set up the convention early as to which is the positive activity. There should be a careful check each time for consistency.

For the objective function data values the user must remember that the practice has been to give costs a plus sign and profits a minus sign (just the opposite of what might seem natural). The reasons for this practice are historical: the original simplex algorithms were aimed at minimizing the value of the objective function, that is, to make the objective function less positive or more negative. So if costs are plus values and profits minus values, minimization (the agendum, MIN...) would operate in the proper direction. The user may, however, reverse this usage if he is careful to use the MAX... agendum instead of the MIN... agendum. In this case the user may make profits positive values and costs negative values.

There may be multiple objective functions so that the same matrix may be evaluated against different criteria. While the discussion has represented the objective function as containing costs, it may use any consistent measure of value, e.g., time used or quantity produced. In the use of MIN... and MAX... agenda a particular objective function is selected as the criteria for optimization.

In revising an existing matrix the values of the previous upper and lower bounds may be changed as well as the previous data value. The 1620-1311 LP System assumes that the only

values to be changed are those that are explicitly stated on the matrix element card following the REVISE agendum card. For instance, if there were originally both an upper and lower bound and the revised matrix element card only contained an upper bound value, only the upper bound would be changed. If the user wishes to delete a previous bound, he should place an asterisk in the first position of the corresponding field on the revised matrix element card; e.g., an asterisk in column 31 would delete a previously stored upper bound.

FIRST.B and NEXT.B INDICATORS

Format

	Indicator Name	Right-Hand-Side Name	Comments
cc	1 7	8 12	13 80
	FIRST.B FIRST.B NEXT.B	 hhhhh hhhhh	

Indicator Name: FIRST.B represents the first (or only) right-hand side. The period in column 6 is optional.
NEXT.B represents any additional right-hand side.

Right-Hand-Side Name: A five-position alphameric field; blanks are significant. Record mark and groupmark cannot be used. The first (or only) right-hand side does not need to have a name; it is given a name of bbbbb (all blanks) if not otherwise specified.

Comments: Any desired information

	Indicator Name	Input Data Source	Starting Sector Location	Problem Name	Comments
cc	1 6	7	8 12	13 18	19 80
	FIRST.	D	sssss	nnnnnn	

Indicator Name: FIRST. represents the first (or only) right-hand side.

Input Data Source: D indicates that the right-hand-side data values are to be referenced from a previous problem.

Starting Sector Location: A five-position numeric field indicating the assigned location for the referenced right-hand-side data values.

Problem Name: A six-position alphameric field stating the name of the referenced problem. This is checked against the problem name stored at the referenced location for accuracy.

Comments: Any desired information

Usage

The FIRST.B indicator card is used to introduce the first (or only) right-hand side values. The first right-hand side may be named bbbbb (all blanks). The NEXT.B indicator card introduces alternate right-hand-side values. Only one right-hand side will be used in a particular solution. Each additional right-hand side must have a name.

There must be a FIRST.B or FIRST.D indicator card if the problem is to be solved. If there is a FIRST.B card, all of the data value cards for that right-hand side must follow behind it. If there are NEXT.B cards, each one must be followed by its appropriate data value cards.

The FIRST.D indicator card references the data values stored with a previous problem. If the previous problem is overwritten, any problem which references this data will not be usable. With FIRST.D, all right-hand sides for the previous problem are referenced so that any of the previous right-hand sides may be used in the MIN... or MAX... agenda.

The MIN... and MAX... agenda specify the particular right-hand side to be used in determining feasibility of the solution.

RIGHT-HAND-SIDE ELEMENT

Format

	Blank	Row Name	Constraint Value	Lower Range Limit	Comments
cc	1 12	13 18	19 30	31 42	43 80
	b b	rrrrrr	±uuuu.uuuuuu		
	b b	rrrrrr	±uuuu.uuuuuu	±vvvv.vvvvvv	

Blank: Must be all blanks

Row Name: A six-position alphameric field denoting the exact row name as specified in the row identification cards.

Constraint Value: A twelve-position field permitting a ten-position numeric value with space for a sign (optional for positive number) and a decimal point. The format shown is recommended. Rules on construction are the same as those under "Matrix Element", data value. This field should be used to enter the constraint value if there is only one (whether it be an equality, a lower limit, or an upper limit). It must be used for an upper range limit. An asterisk in the first position will delete any previous entry for that field.

Lower Range Limit: A twelve-position field permitting a ten-position numeric value with space for a sign (optional for positive numbers) and a decimal point. Construction rules are the same as for Constraint Value. This field must be used for a lower range limit. It may be used (though not recommended) for any individual constraint value. An asterisk in the first position will delete any previous entry for that field.

Comments: Any desired information

Usage

A right-hand-side element card is required for every row where the value of the constraint is not equal to zero. There are five types of constraints which correspond to the row types specified on the row identification card:

Type of Constraint	Row Type
objective function	b (blank)
equality	0 (zero)
less than	+ (12 punch)
greater than	- (11 punch)
range	+ (12 punch) or - (11 punch)

For objective functions it is not necessary to have a right-hand-side element card unless it is desired to insert a constant term in the value determination. The objective function row has the following construction:

$$C_1X_1 + C_2X_2 \ldots C_nX_n + K = Z$$

Assume that the costs (C_1, C_2, etc.) are positive values for costs and negative values for profits. K represents a constant term. If it is a cost, it would be positive, and negative if it is a profit. But only coefficients for structural variables may be stated on matrix element cards, so the constant term cannot be covered. The constant term is therefore subtracted from both sides of the objective function equation, yielding:

$$C_1X_1 + C_2X_2 + \ldots C_nX_n = Z - K$$

The K value would therefore be shown on the right-hand-side element card for the objective function row, but the sign would be the reverse of a coefficient on the matrix element card. For example, if costs are + and profits -, the constant cost would be - and a constant profit + and MIN... would be used to solve the problem; but if costs are - and profits are +, a constant cost would be + and a constant profit - and MAX... would be used to solve the problem.

For an equality, less-than, or greater-than constraint, the value should be inserted in the constraint value field.

For a range constraint the upper limit should be placed in the Constraint Value field and the lower limit in the Lower Range Limit field. If the type of row (on the row identification card) indicates either a less-than or a greater-than constraint, it may be treated as a range constraint simply by inserting both an upper and a lower limit. If the lower and upper limits are equal, the constraint operates as an equality but the solution process is less efficient than if the row type were indicated as an equality. Care must be taken to ensure that the lower limit is algebraically smaller than the upper limit; however, the system does not check for this. By proper choice of values, a less-than constraint can be changed effectively into a greater-than constraint without changing the row identification card. For a row classified as + (less-than constraint) with a value of 3. in the constraint value field, by inserting a large negative number in the lower range limit field, the 3. will be interpreted as an upper bound with effectively an infinite lower bound.

The right-hand-side element cards are placed directly behind the appropriate FIRST.B or NEXT.B indicator card. Row order is not significant. Right-hand sides can be added during INPUT. even when the row identification and matrix files are referenced from disk storage.

It is also possible to insert additional right-hand sides during the original input and then REVISE these values as desired for additional problem solutions. To modify a right-hand-side element value, a new right-hand-side element card is needed. It will change the value only of those fields explicitly stated on the new card. If there were previously both an upper and lower constraint limit and the new card contained only an upper limit, just the upper limit would be modified. If the user wishes to delete a value he inserts an asterisk in the first position of the corresponding field on the new card.

BASIS. INDICATOR

Format

	Indicator Name	Input Data Source	Starting Sector Location	Problem Name	Comments
cc	1 6	7	8 12	13 18	19 80
	BASIS.				
	BASIS.	D	sssss	nnnnnn	

Indicator Name: BASIS. The period in column 6 is optional.

Input Data Source: b (blank) indicates that basis cards will follow.
D indicates that the basis will be referenced from a previous problem.

Starting Sector Location: A five-position numeric field indicating the disk storage location of the problem containing the desired basis.

Problem Name: A six-position alphameric field containing the problem name which is checked against the problem name stored at the referenced storage location.

Blank: Must be all blanks

Comments: Any desired information

Usage

Basis cards are used to provide an advanced start in solving a linear programming problem. The basis is a statement of the particular structural variables which are in the initial solution, plus information as to which logical variables are to be omitted from the initial solution. Normally, basis cards are available from previous LP solutions either of the entire problem or of portions of the total problem. However, with care a user can insert an initial basis directly.

Basis cards for recycling come from OUTPUT and GETOFF, two of the agenda in 1620-1311 LP. The BASIS. indicator card must precede both the VARBLS indicator card, which introduces the structural variables to be placed in the initial solution, and the SLACKS indicator card, which introduces the logical variables to be omitted from the basis.

The BASIS. indicator card provided by OUTPUT contains information on mantissa length and tolerances which is used by the program described in "1620-1311 LP Agenda" under "Output".

If the user is preparing the BASIS, indicator card manually, he may provide this information by means of an ASSIGN agendum card which is covered later under "ASSIGN". Experience with certain problems will indicate which values may lead to efficient solution.

The BASIS.D card references a basis established in a previous problem. No other basis cards may follow a BASIS.D card.

VARBLS INDICATOR

Format

	Indicator Name	Comments
cc	1 6	7 80
	VARBLS	

Indicator Name: VARBLS

Comments: Any desired information

Usage

The VARBLS indicator card precedes the basic structural variable cards. Only one VARBLS indicator is permitted per BASIS. indicator card. The VARBLS indicator card provided by OUTPUT has column headings in the comments field (e.g., TYPE, NAME, ACTIVITY LEVEL).

BASIC STRUCTURAL VARIABLES IDENTIFICATION

Format

	Blank	Basis Entry Type	Column Name	Comments
cc	1 11	12	13 18	19 80
	b b	e	cccccc	

Blank: Must be all blanks

Basis Entry Type: There are three levels at which a structural variable may be entered into a basis:

W - The structural variable is entered at its lower bound; these are not required for basic structural variables. Remember that there is always an implied lower bound of zero unless otherwise specified.

G - The structural variable is entered at its upper bound.

F - The structural variable is entered at an intermediate level; an unbounded variable could be noted only as type F.

Column Name: A six-position alphameric field stating the name of a structural variable which is to be in the initial basis.

Comments: Any desired information

Usage

If the basis cards come from OUTPUT, they will also contain the solution activity level in columns 20-33. W-type cards (structural variables in at their lower bound) will also be included unless there is an implicit lower bound of zero. These W cards are ignored in establishing the initial basis.

If the user is trying to manually provide an advanced starting basis, he should select those structural variables which he feels quite certain will be present in any reasonable solution to the problem. A selected structural variable might be a low-cost ingredient, or an item which is the only source for a certain component of the final product, or a limiting resource (a particular machine or process). If an improperly formed basis is entered, a poor starting solution may result. It is therefore suggested that the new LP user avoid manually preparing a basis until he has gained considerable experience in the characteristics of the problem with which he will be dealing.

The activity level, other than upper (G) or intermediate (F), is not needed since it is calculated by 1620-1311 LP System automatically after the problem is entered. If basis cards have been produced by more than one previous problem, they can be combined by inserting all the basic structural variables cards behind one VARBLS indicator card, and nonbasic logical variables cards behind the SLACKS indicator card.

SLACKS INDICATOR

Format

	Indicator Name	Comments
cc	1 6	7 80
	SLACKS	

Indicator Name: SLACKS

Comments: Any desired information

Usage

The SLACKS indicator card precedes the nonbasic logical variable cards. Only one SLACKS indicator is permitted per BASIS. indicator card; it must come after the basic structural variables cards. The SLACKS indicator card provided by OUTPUT has column headings in the Comments field (for example, TYPE, NAME, ACTIVITY LEVEL, SIMPLEX MULT.).

NONBASIC LOGICAL VARIABLES IDENTIFICATION

Format

	Blank	Basis Entry Type	Row Name	Comments
cc	1 11	12	13 18	19 80
	b b	e	rrrrrr	

Blank: Must be all blanks

Basis Entry Type: W - A logical variable at its lower bound.

G - A logical variable at its upper bound.

F - A logical variable at an intermediate level.

Row Name: A six-position alphameric field containing the row name corresponding to the logical variable which is to be removed from the basis.

Comment: Any desired information

Usage

For every structural variable inserted in the basis, a corresponding logical variable must be removed from the basis. Removing the logical variable is accomplished by naming the constraint row in which that logical variable occurs. The row named should be one in which the basic structural variable has a nonzero coefficient.

The OUTPUT and GETOFF agenda produce nonbasic logical variable cards which can be recycled. These cards also contain certain values (activity level and simplex multiplier); these values will be ignored when the cards are recycled. All F-type cards will be ignored.

ENDATA OR EOF INDICATOR

Format

	Indicator Name	Comments
cc	1 6	7 80
	ENDATA EOF	

Indicator Name: ENDATA or EOF.

Comments: Any desired information

Usage

ENDATA or EOF is used as a terminal indicator. It must be placed at the end of a set of data input cards. Before any agendum card is used, following the data input cards, either an ENDATA or EOF indicator card is required.

This section of the manual gives the details of each of the agendum statements. Thus, it is more of a reference than a tutorial section. It contains four major categories, each with an introduction, followed by the agendum statements:

1. Data Preparation

   INPUT.
   REVISE

2. Optimization

   MAX...
   MIN...
   INVERT

3. Report Preparation

   OUTPUT
   DO.D/J
   COST.R
   CHECK.

4. Control and Data Maintenance

   ASSIGN
   ENDJOB
   ERASEp
   GETOFF
   MAP...
   PAUSE.
   SAVE.p

DATA PREPARATION

It is necessary to provide data in order to initiate problem solution. There are two agendum statements to accomplish this: INPUT. and REVISE. INPUT. brings new data into disk storage or references data already in storage. REVISE changes information in disk storage. These agenda have the data cards following them. The data input cards have been explained in detail in "Data Preparation".

INPUT.

Format

	Agendum Name	Data Source	Starting Sector Address	Problem Name	Comments
cc	1 6	7	8 12	13 18	19 80
	INPUT. INPUT.	C D	sssss sssss	nnnnnn nnnnnn	

Agendum Name: INPUT.

Data Source: C indicates that input data cards will follow.
D indicates that the input should be taken from a previous problem stored on disk.

Starting Sector Address: A five-position numeric field identifying the initial sector location for storing (or retrieving) the problem data. The address must be 01805 or greater.

Problem Name: A six-position alphameric field; any symbols may be used except record mark and groupmark. Blanks are permitted but not recommended as internal characters of a name. For disk input the problem name must correspond exactly with the original problem name at the referenced location.

Comments: Any desired information

Examples

```
INPUT. C03000BLENDG
INPUT. D12000ALLOC
```

Description

Portions of disk module 0 are used for table, program, and monitor residence, and are not available for storage of problem data. Sectors 01805 through 04799 are available. If additional room is necessary, the user must specify, on an ASSIGN card, which sectors beyond the programs and below the monitor should be made available. A description of the ASSIGN card is given later in this section, and a list of the unavailable areas on module 0 appears immediately under "Disk Storage Utilization".

The INPUT. agendum is used to set up the linear programming model for solution or to set up data for special LP analysis or output. This agendum will establish the input files in disk storage from card input, or will prepare to work with the referenced input files in disk storage. INPUT. will reference a previously stored problem, or will input one or more card files. These files are:

- Row identification file, preceded by ROW.ID indicator card
- Column identification file, preceded by COL.ID indicator card. This file is optional and is used primarily to select a subset of variables from the problem to be used during optimization or for selective output.
- Matrix data file, preceded by a MATRIX indicator card. This file defines the data value (coefficients) and bounds of the structural variables. If no column identification file is given, a column identification file is generated from the matrix file.
- Right-hand-side file is preceded by a FIRST. B indicator card. This establishes the data value (coefficients) of the principal or only right-hand side. Additional right-hand sides can be defined; each is preceded by a NEXT. B indicator card.
- Basis identification files. These files are preceded by a BASIS. indicator card. These files are optional. They are used to establish an advanced starting basis. The structural variable file is preceded by a VARBLS indicator card; the logical variable file is preceded by a SLACKS indicator card. This file was probably generated by a previous OUTPUT agendum.

- Input data is terminated by an EOF or ENDATA indicator card.

An input card which references disk storage (INPUT. D) completely defines the input; no indicator card or card file can follow this card.

If any input files are on cards, an INPUT. C card must be used. Data may also be inserted through special disk cards, e.g., ROW.IDD, COL.IDD, MATRIXD, FIRST. D, and BASIS. D. All the problem data may be read in from cards, but it is usually more efficient to define a problem wholly or in part by reference to data stored on disk. The data file for the variables need be read only once. Selective column and row identification files then serve to select the data as required for each particular problem. Right-hand sides and objective functions can also be varied. By this approach, data maintenance is simplified since one master data value file is maintained.

The following illustration shows a complete set of card input identification and data files. Unless noted otherwise, all cards are required to set up a new problem. If the indicator card is present, the file cards following the indicator card are required unless the indicator card refers to information in disk storage. Each of the cards is described in detail in "Data Preparation".

REVISE

Format

cc	Agendum Name	Comments
	1 6	7 80
	REVISE	

Agendum Name: REVISE

Comments: Any desired information

Example

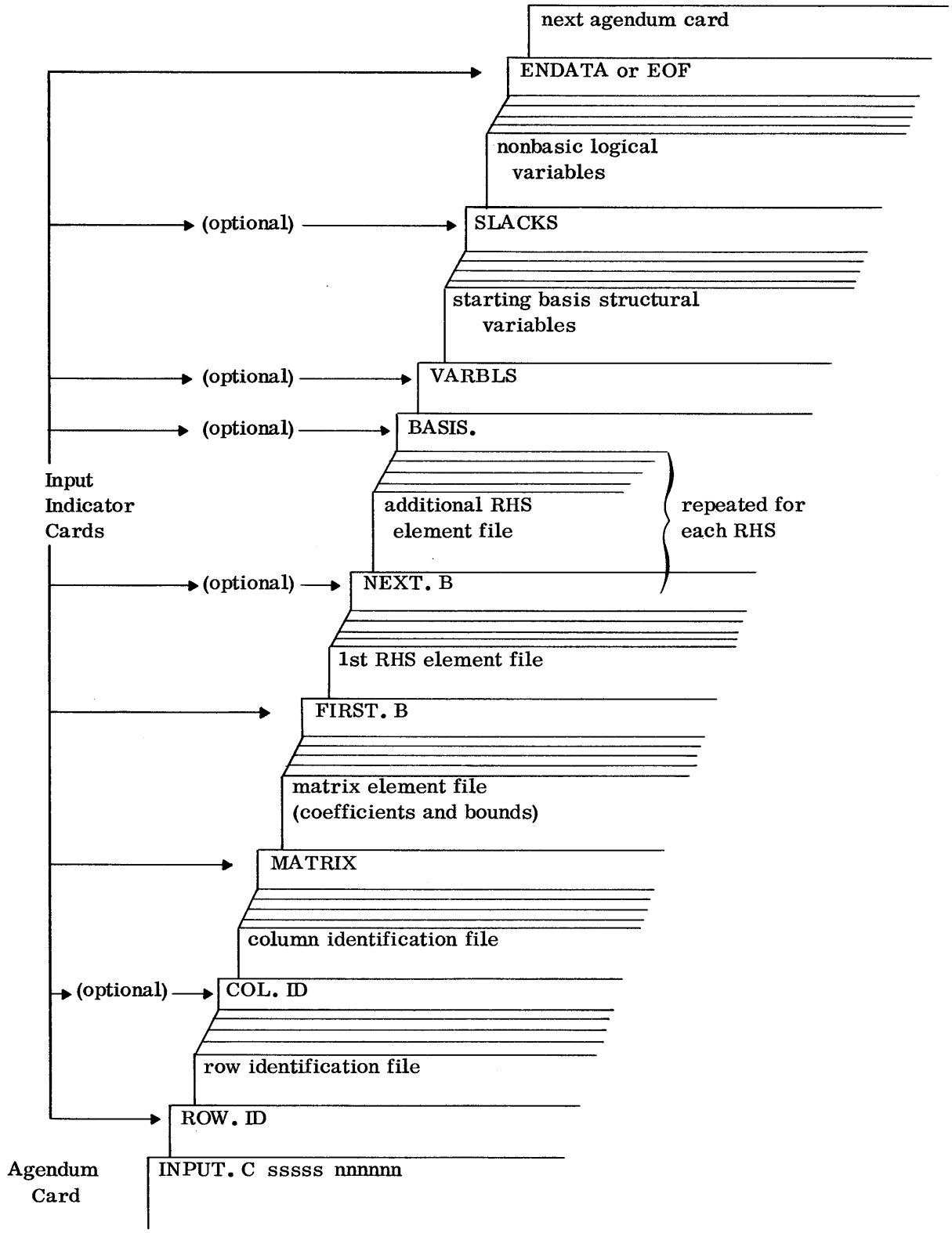
REVISE

Description

The REVISE agendum permits the modification of existing problem data. REVISE works only on the current problem, so it must be preceded by an INPUT. card, though other cards may intervene. For example, INPUT.C, MIN..., OUTPUT and then REVISE, is an acceptable sequence.

REVISE may be followed by any of the input indicator cards, in turn followed by the desired revision data and identification cards: ROW.ID, row identification, COL.ID, column identification, MATRIX, matrix elements, FIRST. B, right-hand-side elements, BASIS., VARBLS, basic structural variables, SLACKS, nonbasic logical variables, EOF or ENDATA.

The order and use of the data input cards is the same as with INPUT., except that no disk references may be made (for example, no ROW.IDD or MATRIXD entries).


next agendum card
ENDATA or EOF
nonbasic logical variables
(optional)
SLACKS
starting basis structural variables
(optional)
VARBLS
(optional)
BASIS.
Input Indicator Cards
additional RHS element file
repeated for each RHS
(optional)
NEXT. B
1st RHS element file
FIRST. B
matrix element file (coefficients and bounds)
MATRIX
column identification file
(optional)
COL. ID
row identification file
ROW. ID
Agendum Card
INPUT. C sssss nnnnnn

Since REVISE works on the current problem, it must be preceded at some previous point by INPUT. REVISE actually changes the identification information and data stored for a particular problem, so if any other problem uses that data, the second problem will in turn be affected by the data.

REVISE can only change information already in disk storage. It will alter any entry in any of the files of the current problem, but it cannot add any new row or column or elements unless provision has been previously made for the entry.

- Row identification file. -- Any row type can be revised. No rows can be added. Rows can be omitted by using an asterisk (*) on a row identification card. The type of generated logical variable can be changed by varying the row type. Rows can be inserted by changing an asterisk row type to any other row type.

- Column identification file. -- Any structural variable can be revised and inserted in the matrix by putting in a column identification card with a blank for column usage. A column may be omitted by putting an asterisk in for column usage. No column names can be added.

- Matrix element file. -- Any entered data value or bound can be revised. It is not possible to revise an implicit zero data value or to specify a bound for a previously unbounded variable. All values for a given matrix element card must be included in the revised data input.

- Right-hand-side element file. -- Any entered constraint value can be revised. It is also possible to change a simple inequality to a range constraint. The first indicator card must be a FIRST. B indicator card. The right-hand-side name can be the name of any right-hand side.

- Basis identification file. -- (1) If basis was saved by SAVE. B or SAVE. I, basis entry type (F and G only) of structural variables can be revised. Similarly, basis entry type (W and G only) of logical variables can be revised. (2) If basis was inserted as part of an input deck, any of the entries may be revised.

- Data revision must be terminated by an EOF or ENDATA indicator card.

Based on information from the various output-producing agenda, the user may want to revise various problem elements. Or, as the result of changing conditions, various factors may have to be changed.

The user may omit rows or columns that are not highly significant to the solution. This is done by inserting a card with the particular row or column name with an asterisk for row type or column usage. The bounds on a structural variable may need to be changed to reflect a different inventory position or technological requirement. The matrix element values may have to be modified to reflect changing use of resources to accomplish different activities. The right-hand-side element values may change to recognize different capacities or demands. The objective function values may need to be changed to consider different prices or resource costs.

For solution efficiency it may be desirable to modify a previous basis to compensate for changes in the matrix elements, bounds, or constraints.

REVISE provides a method for changing right-hand-side values and objective functions. This multiple examination can also be made through the use of alternate right-hand-sides and objective functions in the original problem input.

Revised problems can usually be solved rapidly by using the basis from a previous solution of the problem. The basis can be retained for later use by the SAVE. B agendum. Then this basis will be automatically used as the starting basis for re-solution of the problem. If the user does not want to start from a saved basis, he must use the ERASEB agendum before beginning the solution.

OPTIMIZATION

The optimization phase of a linear programming problem takes the data provided by the input programs or the revise programs and determines the solution which has the maximum profit or the minimum cost, whichever is specified by the user. This phase of processing takes considerably longer than the input or output phases except for very small problems. The number of rows and the number of nonzero elements will have a significant effect upon the time required to solve a problem, as does the variation in the size of the numbers. It is extremely difficult to estimate the running time for a problem, even when the number of rows and variables and the density are known.

Unless a starting basis is provided by the user, the 1620-1311 Linear Programming System will start from an all-slack basis which it generates. Providing a good basis from a previously solved variation of the problem will usually dramatically reduce the solution time.

The 1620-1311 LP System uses the Dual algorithm to produce a feasible solution. The Primal algorithm then automatically takes over to obtain an optimal feasible solution. A sophisticated matrix inversion routine is also used. Inversion is called for automatically every 15 iterations unless the user chooses to change the inversion frequency with an ASSIGN card. In addition, INVERT will be called when the Dual or Primal algorithms consider it necessary to maintain accuracy.

The optimization procedure, called by a MIN... or MAX... agendum card, will automatically calculate and assign a mantissa length according to the problem it is to solve. During the course of the calculation, when the CHECK. routine is executed before each inversion, the mantissa length will automatically be increased if necessary to maintain the accuracy required. The user may override the mantissa length the system selects, by using an ASSIGN card. The user may also stipulate the accuracy he requires by assigning a maximum error tolerance (see "ASSIGN" page 2.15.39). All these features are automatic within the LP system, unless the user chooses to override them because of special knowledge about the problem or special requirements.

In addition to reducing the size of the numbers used in the calculation by using an appropriate mantissa length to fit the problem, a further speed increase is made by calculating, when possible, with the normally smaller original input numbers, maybe two or three digits, even when the mantissa length may be eight or ten. This gain is made possible by the nature of the machine and use of the revised simplex method, product form of the inverse. Additionally, the inversion method is designed to maintain sparsity, thus reducing the number of calculations. See "Mathematical Description of 1620-1311 LP" for a detailed mathematical description. An additional feature, the bounded variable algorithm, significantly improves performance for problems which have bounded variables and/or range constraints.

MAX..., MIN...

Format

	Agendum Name	Spacer	Right-Hand-Side Name	Objective Function Name	Comments
cc	1 6	7	8 12	13 18	19 80
	MAX... MAX...	.	hhhhh	ffffff	
	MIN... MIN...	.	hhhhh	ffffff	

Agendum Name: MAX... or MIN...

Spacer: An unused position. A period is suggested to ensure column alignment.

Right-Hand-Side Name: The name of the right-hand side to be used in the optimization. The right-hand-side name has a maximum of five characters. If the field is left blank, the first (or only) right-hand side is used.

Objective Function Name: The name of the objective function row to be used in the optimization. Whenever the field is left blank, the first (or only) objective function is used.

Examples

```
MAX...
MAX....ALOY2
MIN....BEEFbCOST2
MAX....bbbbbPROFIT
```

Description

MAX... or MIN... (maximize or minimize) calls the main solution procedures to compute the optimal value of the selected objective function. MAX... is generally used with profits expressed as positive numbers and costs expressed as negative numbers in the objective function; the user wishes to maximize profit. MIN... is used to minimize cost and the signs of the objective function would be reversed.

The 1620-1311 Linear Programming System permits the user to include multiple right-hand sides and objective functions. For any optimization, the user selects the right-hand side and objective function he desires by name. Note that right-hand sides may have only five-character names. If the user does not specify a right-hand side, the first one will be used; the same is true for the objective function row name. Neither, either, or both may be specified.

INVERT

Format

	Agendum Name	Spacer	Right-Hand-Side Name	Objective Function Name	Comments
cc	1 6	7	8 12	13 18	19 80
	INVERT	.	hhhhh	ffffff	

Agendum Name: INVERT

Spacer: An unused position. A period is suggested to ensure column alignment.

Right-Hand-Side Name: The name of the right-hand side to be used in the inversion. The right-hand-side name has a maximum of five characters. If the field is left blank, the first (or only) right-hand side is used.

Objective Function Name: The name of the objective function row to be used in the inversion. If the field is left blank, the first (or only) objective function row is used.

Examples

```
INVERT
INVERT.ALOY2
INVERT.ALOY2COST2
INVERT.bbbbbPROFIT
```

Description

The INVERT agendum is used to solve a set of simultaneous equations. It is never necessary for the 1620-1311 LP user to place an INVERT card in a linear programming agenda. If it is used in a linear program agenda, that is, preceding a MIN... or MAX... agendum card, it will force an initial pass through the invert routine. This pass is then bypassed when the MIN... or MAX... card is read. If no INVERT agendum precedes the MIN... or MAX... card, the LP system will take an initial pass through the invert routine.

The purpose of the INVERT agendum is:

1. To solve simultaneous equations

2. To be compatible with other linear programming systems which require the INVERT agendum to precede their optimization agendum when a basis is given

The solution to a set of simultaneous equations requires:

1. An INPUT. agendum and input data

   a. Row identification file, including an objective function row

   b. Matrix

c. Right-hand side(s)

d. Basis*

2. An INVERT agendum card

3. An OUTPUT agendum card

4. (Optional) A CHECK. agendum to test inversion accuracy

DO. D/J and COST. R should not be used (they are meaningless in this case).

REPORT PREPARATION

Various reports are available from the IBM 1620-1311 Linear Programming System. These reports contain, among other things, the objective function value, activity levels, simplex multipliers, reduced costs (D_j's), marginal values, row errors, submatrix summaries, and cost ranges. The particular reports are called by agendum cards (OUTPUT, DO. D/J, COST. R, CHECK.). Any or all of these reports can be obtained after an optimal solution is reached. They may be obtained in any order desired by the user. In addition, an iteration log is maintained to show how the solution is proceeding.

Before describing the reports and how they are obtained, a few basic nonmathematical definitions are presented. For fuller definitions, see the glossary of the IBM manual "An Introduction to Linear Programming" (E20-8171).

Activity level -- the number of units of a structural variable or a slack in the current solution.

Objective function value -- the total cost (or profit) of the current solution. The algebraic sum of the unit costs and/or profits times their activity level yields the objective function value.

Simplex multiplier -- the objective function row of the basis inverse. This row, except for signs, is the set of marginal values. The simplex multipliers are used internally to calculate the reduced costs.

Marginal value -- the amount the objective function value would increase if one more unit of the desirable right-hand-side element were allowable; or, for undesirable right-hand-side restrictions, the amount the objective function value would increase if one less unit were required.

Reduced cost (D_j) -- the objective function coefficient change required to alter the solution activity level of a variable. (1) For a variable without bounds, it is the reduction in cost (or increase in profit) required to tie for a position in the solution. A change in objective function coefficient exceeding the reduced cost will force the variable into the optimum basis. (2) For variables in the solution at a specified lower bound, the same definition applies relative to an increase in solution activity level. The reduced cost of a variable at lower bound can also be interpreted as the unit cost of forcing an additional unit into the solution. (3) For a variable at its upper bound, it is the unit profit of allowing an additional unit into the solution.

* If the basis does not include all of the variables, see the description of the INVERT agendum in "Mathematical Description of 1620-1311 LP" for an explanation of the results.

Row error -- the computational error associated with a row, caused by accumulated roundoff. The row error is calculated by summing the products of each row coefficient by its activity level and subtracting from the right-hand side element. Excessive row errors, when not handled automatically by the system, can generally be handled by increasing the floating-point precision.

Submatrix summary -- the product of the solution level of selected variables with row coefficients summed for the row. One or more rows may be selected. For example, in a feed blending problem this could be used to determine the protein and fat due to vegetable ingredients.

Cost range -- the upper and lower limits which an objective function element may take on and remain in the optimal basis. If the cost (or profit) coefficient changes to a point outside the range, a basis change will occur resulting in an increase (or decrease) in solution value of the variable being analyzed.

There are two report modes: (1) where full reports are desired, and (2) where reports of only selected portions are desired. Full or partial results can be obtained from each of the reports. While the details of each of the reports follow, a quick summary is given to show what values are available and how to get them.

The full reports are obtained by simply placing the agendum card(s) in the agenda deck following the MIN... or MAX... cards (other cards may intervene). The four primary reports are called for as follows and contain the significant values indicated.

```
1       6
OUTPUT
```

The OUTPUT report contains as the first card (or a line of print on the typewriter or printer) a BASIS. indicator card. This deck can be used as an advanced basis in solving subsequent related problems when placed in the input deck, since the format is compatible. For the basic structural variables, the activity levels are given. For the logical variables (slacks and artificials), the activity levels are given for basic variables, and the simplex multipliers for the nonbasic variables. For details see "Output", later in this section.

```
1       6
DO.D/J
```

The DO.D/J report contains the reduced costs (D_j's) for nonbasic variables and basic variables which are forced into the basis because of upper or lower bounds. This report also gives the current cost and the basis value (difference between current cost and reduced cost). Each of the rows in the problem is given the type of row (+, -, =, RANGE), the row name, and the marginal value. For details see "DO. D/J" later in this section.

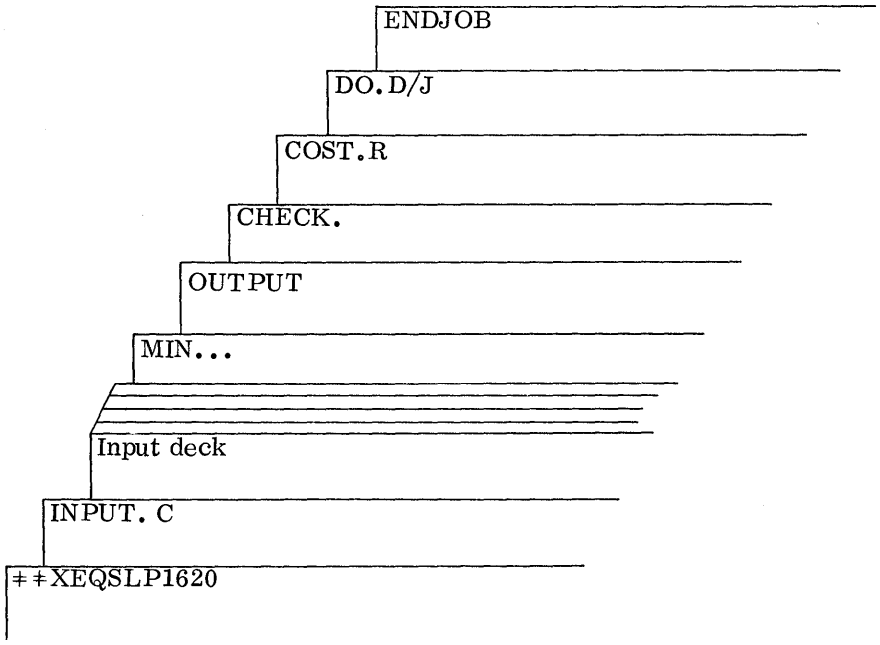
```
1       6
COST.R
```

The COST.R report contains the cost (or profit) interval over which the objective function coefficient may range and remain in the optimal basis. In addition to the upper and lower limit of the range, the report will give the variable that will enter if the upper bound were exceeded (by a cost change), and the change in basis that would occur if the cost were to go below the lower bound. For details see "COST.R" later in this section.

```
1      6
CHECK.
```

The CHECK. report will give the row errors and the maximum error (the row error of the row having the largest error); in addition, the CHECK. report includes the specified row limit or limits and the solution row activity level. For details see "CHECK." later in this section.

The illustration below shows how a user might set up his deck when he desires the full reports. The order of the report agendum cards is not significant.



The format (and method of obtaining selective output from the four output reports) is similar and is briefly summarized. A more detailed explanation is given with each agendum description.

cc	Agendum Name	Data Source	Starting Sector Location	Problem Name	Decimal Digits Field 1	Decimal Digits Field 2	Comments
	1 6	7	8 12	13 18	19	20	21 80
	OUTPUT	D	sssss	nnnnnn	d_1	d_2	
	CHECK.	D	sssss	nnnnnn	d_1	d_2	
	DO.D/J	D	sssss	nnnnnn	d_1		
	COST.R	D	sssss	nnnnnn	d_1		

Agendum Name: The agendum call card OUTPUT, CHECK., DO.D/J, or COST.R.

Data Source: D indicates that the referenced identification files have been previously placed on the disk by INPUT..

Starting Sector Location: This five-digit field is used to give the sector address of the row and/or row column identification files containing the names of the rows and/or columns (a subset of the entire row and/or column identification files for the problem) for which output is to be prepared for the report. The address must correspond to the address given on the INPUT.C card for the desired file.

Problem Name: Used to specify the name of the subproblem to be reported on. The name must correspond to the name given on the INPUT.C card for the desired file. If the name does not take the full six characters, the names should line up on a character-by-character basis; for example, PROD1b is not the same as bPROD1.

Decimal Digits: The number of desired decimal digits to follow the decimal point for the fixed-point output.

Field 1 blank: Three decimal digits are given as standard for activity level, reduced cost, row solution level, and all costs for OUTPUT, DO.D/J, CHECK., and COST.R respectively. A digit in this position specifies the number of decimal digits desired.

Field 2 specifies the number of decimal digits for row error and simplex multipliers in CHECK. and OUTPUT respectively. Standard for row error and simplex multipliers is 8 and 3, which is given if the field is left blank. Does not apply to DO.D/J and CHECK.

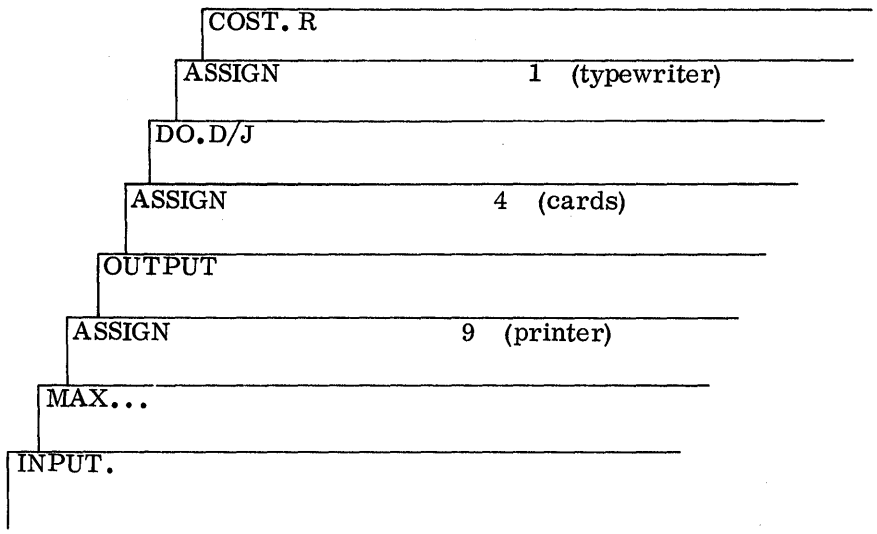
The various output reports available to the 1620-1311 linear program user may be obtained on the typewriter, on cards, or on an online printer. The standard output device is the card punch. If another device is desired, the user may assign it using the ASSIGN agendum card (see "ASSIGN"). For example, to cause the output results to be printed on an online printer, the user would enter the following agendum card:

```
                OUTPUT
          ASSIGN                    9
cc        1     6                   16
```

To put the results on the typewriter the user would precede any output agendum cards with an assign card, as follows:

```
                DO.D/J
          ASSIGN                    1
cc        1     6                   16
```

To reassign to the card punch would call for an ASSIGN card with a 4 punch in column 16.



The agenda above would put the normal OUTPUT on the printer, the D_j's (reduced costs) on cards, and the cost ranges on the typewriter. Note that any INPUT. card will reinitialize the output unit to card. For assigning output devices for the iteration log and the agenda log see "ASSIGN".

The Iteration Log, unlike the other output reports, is not controlled by agendum cards. At the completion of each simplex iteration a one-line item is prepared to indicate what happened on that iteration. Through the use of the ASSIGN card, the user can obtain this information directly on the typewriter, on the printer, or punch a card deck. If the user chooses not to log every iteration, he can, through the ASSIGN card, assign the frequency of logging that he chooses. Additional control of the iteration log is provided through console switch 1. By putting console switch 1 ON, every iteration will be logged on the typewriter. When it is turned OFF, the normal logging frequency is maintained. The standard frequency for logging is every iteration. The standard output device for the iteration log is the typewriter.

The headings and the meaning of the values to which they apply follow:

ITER: The iteration number.

VARBL. EXIT: Name of the variable leaving the basis. "U.B." indicates that the variable is leaving at its upper bound.

VARBL. ENTR: Name of the variable entering the basis. "U.B." or "L.B." indicates that the variable is entering at either its upper or lower bound.

CNT: While in the dual algorithm, the number indicates the number of infeasibilities at the beginning of the iteration. After the solution has become feasible and the primal algorithm takes over, the field indicates the number of desirable variables on the last iteration.

OBJ. FUNCTION: The current value of the objective function.

OUTPUT

Format

	Agendum Name	Data Source	Starting Sector Location	Problem Name	Decimal Digits Field 1	Decimal Digits Field 2	Comments
cc	1 6	7	8 12	13 18	19	20	21 80
	OUTPUT OUTPUT	D	sssss	nnnnnn	d_1 d_1	d_2 d_2	

Agendum Name: OUTPUT

Data Source: blank - complete output of all basic variables and all slacks plus simplex multipliers for all nonbasic slacks. If Data Source is blank, the Sector Location and Problem Name fields will also be blank.

D-Output will be selected from a file specified on the disk.

Starting Sector Location: The address specified on the INPUT.C card for the row and/or column identification files which provide the names of the desired basic structural variables and/or the names of the desired rows for the output report.

Problem Name: Provides the name of the file specified on the INPUT.C card which gives the names of the desired variables and/or rows for the output report. The problem name is used to check that the user has specified the correct Starting Sector Location. An error condition will be indicated if there is not an exact match.

Decimal Digits: If either Field 1 or Field 2 is blank, three decimal digits will follow the decimal point.

Field 1 -- The digit specified will indicate the number of desired digits following the decimal point for the activity level field in the output report.

Field 2 -- The digit specified will indicate the desired number of decimal digits to follow the decimal point in the simplex multiplier field on the output report.

Examples

```
OUTPUT
OUTPUT                  25
OUTPUTD03000SHIPNG
OUTPUTD02500PROD1b16
```

Description

The OUTPUT report contains three primary pieces of information: (1) the activity level for basic variables and slacks, (2) the values of the simplex multiplier for the rows containing nonbasic slacks, and (3) the current value of the objective function. The activity level indicates the number of units of the variable in the optimal solution. The activity level of a slack indicates the difference between the sum of activity levels for a row and the right-hand-side value. For example, in a resource availability constraint row, the slack would indicate the amount of the resource not used. The simplex multipliers (the row evaluators) have the same absolute value as the marginal values and indicate how uneconomic it is to bring the corresponding slack variable into the basis.

In addition to displaying the primary solution to a linear programming problem, the OUTPUT agendum also performs another valuable function. If the output is placed on cards, the same cards may later be reintroduced as input to provide an advanced starting basis for re-solution. Starting with a good advanced basis will substantially reduce the time to resolve the problem even when some significant changes have been made to the objective function, right-hand side, or matrix. To accomplish this, three input indicator cards are produced by the OUTPUT routine. These cards also provide the headings for the OUTPUT report. The indicator cards are BASIS., VARBLS, and SLACKS.

The output deck can then be taken intact and placed behind the matrix deck when inputting a subsequent problem.

If the output is placed on cards, sequence numbers are punched in columns 76-80.

Sample Report

```
OUTPUT
BASIS.              MANTISSA 10    TOLERANCES  05 05 05 05 05
VARBLS  TYPE  NAME  ACTIVITY LEVEL
              FBIN2       190.678
              WBIN3       500.000
              FBIN4       233.051
              FALUM       961.864
              FSILCON     114.407
SLACKS  TYPE  NAME  ACTIVITY LEVEL  SIMPLEX MULT.
              FVALUE      373.686
              FVALUE2     373.686-
              OGYIELD        .000          .264-
              +WFE                         2.843
              +WCU                         2.754
              +FMN          22.712
              +FMG          24.280
              -FAL          54.619
              -WSI                         .208-
```

Report Description

The report resulting from the OUTPUT agendum appears in two sections. Each section of the report is preceded by its indicator/heading card. In addition there is a BASIS. indicator card which precedes the entire OUTPUT report. It is this indicator card that is recognized by the INPUT agendum card when the output deck is being recycled to provide an advanced starting basis. In addition, this card indicates the mantissa length and tolerances which were set during input processing or during the calculation. (See "FORMAT" for definitions of these lengths and tolerances.)

Since the OUTPUT report, when punched on cards, can be recycled as input, significant card columns will be indicated.

BASIS.	MANTISSA xx	TOLERANCES t_1t_1 t_2t_2 t_3t_3 t_4t_4 $t_5t_5$xxxxx
col. 1-6	BASIS.	Indicator/heading
col. 20-27	MANTISSA	Title
col. 29-30	xx	Mantissa length
col. 35-44	TOLERANCES	Title
col. 47-48	t_1	The element tolerance
col. 50-51	t_2	The pivot tolerance
col. 53-54	t_3	The feasibility tolerance
col. 56-57	t_4	The objective function tolerance
col. 59-60	t_5	The row error tolerance
col. 76-80	xxxxx	The output card sequence number

After the BASIS. indicator card, the variables section of the output report appears preceded by its indicator card. Except for the indicator VARBLS in columns 1-6, the card contains just column headings.

VARBLS: Indicates that the structural variables follow.

TYPE: Indicates the type of structural variable. Type W indicates a variable at its lower bound (implicit lower bound of zero excepted). Type G indicates a variable in the basis at its upper bound. Type F indicates a variable at an intermediate level.

NAME: Contains the name of the structural variable.

ACTIVITY LEVEL: Indicates the number of units of the named variable in the current solution. The activity level will be to three decimal digits unless otherwise specified in column 19 of the OUTPUT agendum card.

Card Count: The output cards are numbered sequentially.

The second section of the report is preceded by a SLACKS indicator in positions 1-6. This section contains information about the logical or slack variables.

SLACKS: Indicates that the slack variables follow.

TYPE: Indicates the type of slack variable--type F when at an intermediate level, type W when at the lower bound, and type G when at the upper bound. Range constraints and equations can be type G.

NAME: Contains the name of the row for the slack (logical) variable.

ACTIVITY LEVEL: Indicates the number of units of the slack variable in the basis. First entry in this section of the report generally contains the value of the objective function. There will not be a value in this field for type W slacks.

SIMPLEX MULT: This field contains the value of the row simplex multiplier. SLACKS at an intermediate value, type F, since they are in the basis, will not have a simplex multiplier value. The simplex multiplier for a logical (slack) variable indicates how uneconomical it would be to bring the slack variable, identified by its row name, into the basis. The simplex multipliers for the implicit constraints expressed by the bounds are available from DO. D/J.

Card Count: The output cards are numbered sequentially.

The OUTPUT agendum is sometimes used when an optimal solution has not been obtained. This might happen when the problem has not been formulated properly or some unexpected difficulty has occurred. The output cards can be recycled to restart the solution from an advanced basis.

DO. D/J

Format

	Agendum Name	Data Source	Starting Sector Location	Problem Name	Decimal Digits	Comments
cc	1 6	7	8 12	13 18	19	20 80
	DO. D/J DO. D/J	D	sssss	nnnnnn	d d	

Agendum Name: DO. D/J

Data Source: Blank.--All D_j's (reduced costs) will be produced. If Data Source is blank then Starting Sector Location and Problem Name will also be blank,
D - D_j's will be selected from a file specified on disk.

Starting Sector Location: The address specified on the INPUT. C card which contains the identifications of the desired variables and/or rows.

Problem Name: The name of the file specified on the INPUT. C card. The name of the file on the disk must match the name on the DO. D/J card or an error condition will exist.

Decimal Digits: The number of digits the user wishes following the decimal point for the reduced costs. If blank, three digits will be given.

Examples

```
DO. D/J
DO. D/J                6
DO. D/JD03000SHIPNG
DO. D/JD02700DISTRB2
```

Description

D_j's (the reduced costs) indicate the uneconomic change in the value of the objective function if a unit of the undesirable activity (variable) is brought into the solution. Or to look at it another way, the reduced cost indicates the difference between the current cost of a variable and the lower cost at which it would, if in the basis, yield the same optimal objective function value. This lower cost is called the basis value in the report. If the cost were reduced still more, the variable would enter the optimal basis and the objective function value would be decreased (when minimizing). When applied to profit rather than cost, the D_j's indicate the increase in profitability required to make the variable tie for a position in the basis. If the profitability is increased more it will enter the optimal solution and the maximum profit will be increased.

Sample Report

```
DO.D/J
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
 VBLS    TYPE  NAME   CURRENT COST   REDUCED COST    BASIS VALUE
             WBIN1          .030           .254           .224-
             WBIN5          .150           .015           .135
 ROWS    TYPE  NAME   INCR B VALUE   DECR B VALUE
             OYIELD          .014
             +FE            2.568
             +CU
             +MN             .544
             +MG
             -AL                            .252
          RANGE+SI                          .485
```

Report Description

The report resulting from the DO. D/J agendum appears in two sections, each with appropriate report headings.

The first section contains the structural variables which are not in the basis and the structural variables which are uneconomically in the basis at either their lower or upper bound. The output sequence is the same as the input sequence.

VARBLS: Indicates that the variable group follows.

TYPE: Type W indicates the type of a structural variable which is either in the basis at its lower bound or is not in the basis. Type G indicates a structural variable in the basis at its upper bound.

NAME: Contains the name of the structural variable.

REDUCED COST: This field contains the reduced cost. For nonbasic variables (Type W) the meaning given above applies. For a basic variable at its lower bound (also type W), the economic interpretation is the cost of forcing one unit of the variable into the solution by increasing the bound. For a variable in the basis at its upper bound (type G), the economic interpretation is the reduction in the objective function (when minimizing) that would be obtained if one more unit of the variable were permitted in the solution. When applied to profit items, the opposite economic interpretation holds. The reduced cost will be shown to three decimal digits unless otherwise specified in column 19 of the DO. D/J agendum card.

BASIS VALUE: This field contains the value that a variable must attain in order to economically tie for entry in the optimal solution. It is computed as BASIS VALUE=CURRENT COST-REDUCED COST.

Card Count: The output cards are numbered sequentially.

The second section of the report provides the incremental or decremental marginal values. The incremental marginal value tells how much the objective function will increase corresponding to an increase of one unit of the right-hand side (within a certain range). The range is not provided. The decremental marginal value indicates the increase in the objective function value for each unit decrease in the right-hand-side constraint.

ROWS: Indicates that the row group follows.

TYPE: Identifies the type of row: + for a ≤ constraint row, - for a ≥ constraint row, and 0 for an equality row. If the row also has a range specification on the input cards, the word RANGE will precede the indicator.

NAME: The row name corresponding to the logical variable is given.

INCR B VALUE: The marginal value prescribing the amount by which the objective function would be increased if the right-hand-side restriction were increased by one unit.

DECR B VALUE: The marginal value prescribing the amount by which the objective function would be increased if the right-hand-side restriction were decreased by one unit.

Card Count: The output cards are numbered sequentially.

COST. R

Format

	Agendum Name	Data Source	Starting Sector Location	Problem Name	Decimal Digits	Comments
cc	1 6	7	8 12	13 18	19	20 80
	COST. R COST. R	D	sssss	nnnnnn	d d	

Agendum Name: COST. R

Data Source: blank. - Cost ranges for all objective function coefficients will be produced. When Data Source is blank, so also will be Starting Sector Location and Problem Name.
D - Cost ranges will be given only for those variables specified in the referenced file on the disk.

Starting Sector Location: The address specified on the INPUT. C card which provides the names of the variables for which the user desires cost ranges. Only those variables in the file which are in the basic solution will have cost ranges calculated.

Problem Name: Specifies the name of the file which was on the INPUT. card. If the name specified on the COST. R agendum card does not match the name on the sector specified, an error condition will result.

Decimal Digits: The number of digits the user wishes following the decimal point. If blan' three digits will be given.

Examples

```
COST. R
COST. R                6
COST. RD03000SHIPNG2
```

Description

Cost ranges have meaning only for structural variables that are in the basis. The ranges apply to the coefficients of the objective functions and indicate the upper and lower limit for prices which would yield the same activity level as the optimal solution. If the price were to go above the upper limit or below the lower limit, the activity level of that variable will change. It provides a good measure of the sensitivity of the optimal solution to price changes.

Sample Report

```
COST. R
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
COST.R  NAME    CURRENT COST    HIGHEST COST  HI-VAR   LO-VAR   LOWEST COST
        BIN2        .080            .089      MN       BIN1        .017
        BIN3        .170            .179      BIN5     MN          .160
        BIN4        .120            .147      BIN1     MN          .109
        ALUM        .210            .226      MN       AL          .189
        SILCON      .380            .467      MN       GSI         .148
```

Report Description

The report resulting from the COST. R agendum contains a heading line and a series of detail lines corresponding to all, or a subset, of the basic structural variables. The output sequence will be the same as the input sequence. This page describes the report if COST. R follows a MIN....

COST. R: Indicates that the following report contains the cost ranges

NAME: The name of the structural variable to which the cost ranges apply. The cost ranges apply only to variables in the basis at an intermediate level (type F variables). Cost range information about other variables is obtained from DO. D/J.

CURRENT COST: Current value of the objective function coefficient.

HIGHEST COST: The highest value the cost of the variable may take on and still yield the current activity level.

HI-VAR: The basis type if at its upper bound, and the name of the variable which limits the highest cost of this variable; that is, if the cost of variable whose cost range we have obtained increases above the upper cost limit, its activity level will decrease, and the activity level for the variable named in this field will increase. *

LO-VAR: The basis type if at its upper bound, and the name of the variable which limits the lowest cost; that is, the variable named in this field will increase* if the variable whose range we are obtaining goes below the lowest cost.

LOWEST COST: The lowest value the cost of the variable may attain and still yield the current activity level.

Card Count: The output cards are numbered sequentially.

*If the variable is already at its upper bound, it will decrease rather than increase.

If COST.R follows a MAX..., then highest cost is equivalent to lowest profit, and lowest cost is equivalent to highest profit.

CHECK.

Format

	Agendum Name	Data Source	Starting Sector Location	Problem Name	Decimal Digit Field 1	Decimal Digit Field 2	Comments
cc	1 6	7	8 12	13 18	19	20	21 80
	CHECK. CHECK.	D	sssss	nnnnnn	d_1 d_1	d_2 d_2	

Agendum Name: CHECK.

Data Source: blank. - Complete CHECK. of all rows and columns. If Data Source is blank, the Starting Sector Location and Problem Name field will also be blank.

D - The row and column identification files will be obtained from disk.

Starting Sector Location: The address specified on the INPUT.C agendum card which contains the names of the desired rows and columns.

Problem Name: Provides the name to correspond with that on the INPUT.C agendum card which contains the names of the desired rows and variables. The name of the file on the disk must match the name on the CHECK. card or an error condition will exist.

Decimal Digits:

Field 1 - the number of digits desired following the decimal point for the upper limit, solution value, and lower limit field. If blank, three digits will be given.

Field 2 - the number of digits desired for the row error. If blank, eight digits will be given.

Examples

```
CHECK.
CHECK.D02500SHIPNGb6
CHECK.D03000PROTEN15
```

Description

The CHECK. agendum is used in three different ways: (1) to provide a row check of the solution accuracy for the optimal solution, (2) to provide row summaries for selected variables, and (3) internally by the system to ensure that the solution is proceeding within the defined accuracy.

The report contains a line item for each row of the problem or a subset of the rows if so directed by a row identification file selected through the agendum card. The report contains the row name, the upper limit and/or lower limit for the row as indicated by the right-hand side or range constraint, the computed solution value, and the row error. The row error is computed by taking the sum of the products of the activity level times the coefficients and subtracting from the row limit.

A submatrix summary can be obtained by indicating the desired row and column identification (from row and column identification files) for the desired results. In this case the summation will occur only for the columns selected, that is, all points of intersection in the matrix.

Sample Report

```
CHECK.
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
CHECK.   ROW  NAME    UPPER LIMIT    SOL. VALUE    LOWER LIMIT    ROW ERROR
           VALUE                      296.217          .000       .00000000
           VALUE2                     296.217-         .000       .00000000
           YIELD      2000.000       2000.000                     .00002000
           FE           60.000         60.000                     .00000010-
           CU          100.000         83.968                     .00000050
           MN           40.000         40.000                     .00000010-
           MG           30.000         19.960                     .00000010=
           AL                        1500.000       1500.000      .00001000
           SI          300.000        250.000        250.000       .00000300
* MAX ERROR =         .00002000
```

Report Description

CHECK: Identifies the type of report

ROW NAME: Contains the name of the row.

UPPER LIMIT: The upper limit of the row, either the right-hand-side value in an equal-to-or-less-than row, or the range constraint.

SOL. VALUE: The sum of the products of the row coefficients times the activity levels for the selected columns. If all columns are used -- that is, if a submatrix summary is not being obtained -- the value should differ from the upper or lower limit by the amount of the row error (if any).

LOWER LIMIT: The lower limit for the row -- either the right-hand side for an equal-to-or-greater-than row or the lower limit of a range constraint. R-H-S objective function entries are reported in this field.

ROW ERROR: The total accumulated roundoff error for the row, calculated as the difference between the right-hand side and the sum of the products of row coefficients times the activity levels.

CONTROL AND DATA MAINTENANCE

In addition to the main functions of solving a linear programming problem --input, optimization, output-- a number of other functions are needed in the control language to make a complete system. These include various control statements and data maintenance facilities. Included in this section, in alphabetic order, are:

ASSIGN
ENDJOB
ERASEp
GETOFF

MAP. . .
PAUSE.
SAVE. p

ASSIGN

Format

Card Columns:

1-6 ASSIGN - agendum name

7-8 Mantissa length: standard= as calculated; may be assigned values 05-18.

9 Master tolerance: if used, all tolerances will be set to this value; as for all tolerances, it is the negative power of 10 desired.

10 Element tolerance: standard = as calculated

11 Pivot tolerance: standard = as calculated

12 Feasibility tolerance: standard = as calculated

13 Objective function tolerance: standard = as calculated

14 Maximum error tolerance: standard = as calculated

15 Iteration log output unit: standard = typewriter: 1 for typewriter, 4 for punch, 9 for printer

16 Reports output unit: standard = punch: 1 for typewriter, 4 for punch, 9 for printer

17 Agenda log unit: standard = typewriter: 1 for typewriter, 4 for punch, 9 for printer.

18-22 Sector address-DIM: where DIM table, programs and data terminate

23-27 Sector address upper limit: usable for linear programming data up to this location

28-29 Inversion frequency: standard = 15

30-32 Iteration log frequency: standard = 1

33-37 Common computation area sector address

38-80 Comments

Agendum Name: ASSIGN

Mantissa Length: The size of the fractional part of a floating-point number. The system will initially calculate a mantissa length based upon the problem to be solved and the range of numbers encountered by the INPUT. agendum. As its solution proceeds, a recalculation is made after the CHECK. routine is executed prior to each inversion. If the error buildup

warrants it, the mantissa length is increased automatically. The dual and primal optimization routines continually check for error buildup; if it becomes excessive, control will also revert to a routine to recompute a proper mantissa length. If the user chooses, he may override this automatic mechanism and specify a mantissa length before the optimization is initiated by the MIN... or MAX... agendum card. To do this, the ASSIGN card should be placed after the INPUT. card.

Master Tolerance: A single control is provided to allow the setting of all tolerances to a single value. It would give the same results as writing the number in this field in the next five fields. All tolerances are calculated by the system automatically, and generally are changed by the user only when the user has special requirements or special knowledge about the problem, or when he encounters difficulty in solving a particular problem.

Element Tolerance: When any computed matrix value becomes smaller than the element tolerance, it is treated as "essentially zero" and given a value of zero.

Pivot Tolerance: Whenever a candidate pivot element is found to be lower than tolerance level, it is treated as zero and ignored.

Feasibility Tolerance: The feasibility tolerance is the amount a right-hand-side element may be negative and still be considered to yield a feasible result.

Objective Function Tolerance: If during the "pricing" operation, the pricing criterion values becomes less than this tolerance level, it is considered "essentially zero" and rejected.

Maximum Error Tolerance: If a row error exceeds this amount when automatically calculated by the CHECK. routine, the mantissa length will be increased prior to further iterations.

Iteration Log Output Unit: The output unit to receive the iteration log; the standard unit is the typewriter. A digit 1, 4, or 9 will put the log on the typewriter, punch, or printer respectively. In addition, the user may get every iteration logged on the typewriter during MIN. . . or MAX. . . by placing console switch 1 in the ON position.

Reports Output Unit: The standard output unit for reports is the card punch. They may be placed on the typewriter, card punch, or printer by assigning a 1, 4, or 9 respectively in this position.

Agenda Log Unit: The agenda and comments cards in the agendum stream are always typed. To place this information on the printer or card punch, assign a 4 or 9 respectively in this position.

Sector Address - DIM: This field is used to specify the lower limit where LP data may be placed following the DIM table and user programs. This field should be used to provide a lower limit for LP data following the work cylinders and the DIM table. The user must explicitly indicate available space above the work cylinders, DIM tables, and other user information before the LP system will use the area. He must also make an entry in the next field.

Sector Address-Upper Limit: This field is used to specify the highest sector address usable by the LP system for data storage. If this and the prior field are not specified, the LP system will terminate if the problem data exceeds the work cylinders. Use of this field and the preceding field allows the user to file-protect part of the disk.

Inversion Frequency: The user may specify how often he wants to take inversions. If not specified, inversion will take place every 15 iterations. Only those iterations in which a variable leaves the basis are counted.

Iteration Log Frequency: The user may specify the desired frequency of iteration logging. Standard logging frequency is 1 and, unless otherwise specified, it will be on the typewriter. If console switch 1 is ON, every iteration will be logged on the typewriter, in addition to the unit specified on the ASSIGN card.

Common Computation Area Sector Address: A considerable saving of disk space can be achieved when many problems are being stored and solved by specifying a single common computation area. All computational results will be started at this location if an entry is made.

Examples

```
ASSIGN097bbbbb191bbbbbbbbbb20005bbbbb
ASSIGNbbbbbbbb999080001600006500
ASSIGNbbbbbbb5
```

Description

The 1620-1311 Linear Programming System has been designed to allow simple hands-off operation. Many automatic features have been implemented to accomplish this goal. However, there are times when it is desirable for the user to be able to control the system more directly. The ASSIGN agendum allows him to do this by giving him access to a number of the internal controls. These controls include the mantissa length, inversion frequency, tolerances, output devices, and the reservation of disk storage.

A number of things should be kept in mind:

- The system will operate with standard settings unless otherwise directed by ASSIGN.
- INPUT. C will reintialize to the standard settings except for agenda log unit, lower and upper available disk storage, and the start of the common computational area.
- The contents of an ASSIGN card will stay in effect until another ASSIGN card changes these, or another INPUT. C agendum is executed which will initialize all values except those noted above.
- When the SAVE. p agendum is used to save a problem, the contents of the ASSIGN card in effect will be saved along with the problem. If an INPUT. D agendum later references the saved problem, the ASSIGN controls which were saved will be restored. Thus, the problem can be reoptimized without another ASSIGN card being used.
- Any or all of the fields on an ASSIGN card may be filled in.
- Multiple ASSIGN cards are permitted and they may be interspersed among the agendum cards.

ENDJOB

Format

cc	Agendum Name	Comments
	1 6	7 80
	ENDJOB	

Agendum Name: ENDJOB

Comments: Any desired information

Example: ENDJOB

Description

This card is used to terminate the linear programming run(s). Control will be returned to the 1620 Monitor.

ERASEp

Format

cc	Agendum Name	Storage Code	Starting Sector Location	Problem Name	Comments
	1 6	7	8 12	13 18	19 80
	ERASEB ERASEB ERASEI ERASEI	 D D	 sssss sssss	 nnnnnn nnnnnn	

Agendum Name: ERASEB is used to delete a previously saved basis or a previously saved basis and inverse.

ERASEI is used to delete a previously saved inverse (the basis is still retained).

Storage Code: blank - erases current problem basis and/or inverse
D - erases basis and/or inverse which is at specified disk location.

Starting Sector Location: A five- position numeric field which identifies the disk storage location where the referenced basis (and inverse) is stored.

Problem Name: A six-position alphameric field containing the problem name which is checked against the problem name stored at the referenced sector location.

Comments: Any desired information

Examples

ERASEB 04300PRODMX
ERASEI 08750DISTBR

Description

ERASEp is the means by which a previously retained basis (and inverse) may be deleted. If the basis will no longer be useful, it must be erased before re-solution of the problem. If just the basis has been saved (through a SAVE. B), an ERASEB is all that is needed. If a SAVE. I has been used to save both basis and inverse, the user has to decide whether the basis might still be of value. If not, then an ERASEB is used which eliminates both the basis and the inverse. The ERASEp agendum is not used very often, since a new basis and inverse can be saved right over an old one.

If a previously retained inverse is destroyed by another input or usage of the disk area, this destruction must be indicated to the system by an ERASEI before its attempted use in optimization.

GETOFF

Format

cc	Agendum Name	Comments
	1 6	7 80
	GETOFF	

Agendum Name: GETOFF

Comments: Any desired information

Example

GETOFF

Description

GETOFF provides a means of interrupting a linear programming solution for a higher-priority job or to provide a restart basis if problem solution information (from iteration log) indicates that solution is not progressing satisfactorily. GETOFF is meaningful only if a basis has been established. It will generate basis cards (similar to OUTPUT except that values are not inserted):

BASIS.
VARBLS
Basic structural variables
SLACKS
Nonbasic logical variables

If sense switch 2 is ON, all cards will be bypassed until either a GETOFF or ENDJOB card is read.

MAP. . .

Format

	Agendum Name	Data Source	Starting Sector Location	Problem Name	Comments
cc	1 6	7	8 12	13 18	19 80
	MAP. . .				
	MAP. . .	D	sssss	nnnnnn	

Agendum Name: MAP. . .

Data Source: blank. - The MAP. . . report is desired for the current problem.
D - The MAP. . . report is desired for a specified problem which is stored on the disk.

Starting Sector Location: The address specified on the INPUT. C card for the row and/or column identification files.

Problem Name: The name of the problem specified on the INPUT. C card. This name must match the name of the problem at the Starting Sector Location, or an error condition will exist.

Examples

```
MAP. . .
MAP. . . D02000ALLOYA
```

Description

The MAP report specifies the disk storage that contains the current problem or a previous problem. The report always contains (1) the starting and final sector locations occupied by the input, (2) the problem name, and (3) the number of entries in the row and column identification files. In addition, if the input data includes both a matrix and a right-hand side, the report contains the current mantissa length and the number of sectors required for the inverse and beta file computation area (computation file area). This area is in addition to the input area.

If the report is requested before optimization for a problem with which no basis was input, the input area listed does not include that which would be necessary for saving a basis. If the report is requested after optimization, or if a basis was input with the problem, the input area does include the basis.

Format of Report

Columns

1-6 MAP. . . - agendum name

7 S

8-12 Starting sector location occupied by the input

13 /

14-18 Final sector location occupied by the input

19 b

20-25 Problem name

26-27 .b

28-30 Number of entries in the ROW.ID file

31-34 ROW,

35-38 Number of entries in the COL.ID file

39-41 COL

42-48 .MANSA-

49-50 Current mantissa length

51-59 -COMPUTE-

60-65 Number of sectors occupied by the computation file area

66-74 -SECTORS.

Columns 42-74 appear only if the input includes a matrix and right-hand side.

Sample Report

```
MAP...D03600ALLOYC
MAP...S03600/03605 ALLOYC.  009ROW,0007COL.MANSA-10-COMPUTE-000086-SECTORS.
MAP...D04400ALLOYD
MAP...S04400/04406 ALLOYD.  000ROW,0002COL
```

The compute area varies with machine size. The illustration given above is for a 40K machine. For a 20K machine the area would be less; for a 60K machine it would be greater.

The following chart was developed from the MAP. . . reports for several problems. It gives an indication of the number of sectors required for the files of problems of various size problems.

Several of the MAP. . . reports were produced before optimization; several after. This is indicated by the columns titled Number of Sectors--No basis, and Incl. basis, respectively.

Problem	Rows	Cols.	Number of Sectors			Mantissa Length
			No basis	Incl. basis	Compute Area	
Share 21	245	352	971		8958	10
Share 5	399	344	1175		17156	9
	63	22	188		426	11
	39	13	96	104	232	11
	28	13	104	110	201	11
	42	107	264		361	9
	6	3		25	142	9

PAUSE.

Format

cc

Agendum Name	Comments
1 6	7 80
PAUSE.	

Agendum Name: PAUSE.: period in column 6 is not required.

Comments: Any desired information

Example

PAUSE.

Description

This agendum serves to stop the computer, and leaves it ready to continue LP processing when the start key is depressed. At this point, the next agendum is read and processing continues. It might be used between two LP jobs when it is desirable to look at the results of the first, before the second is started. If it were decided to conclude LP processing, an ENDJOB card would be placed in the reader, and upon depressing the start key, control would be returned to Monitor I.

SAVE. p

Format

cc	Agendum Name	Storage Code	Starting Sector Location	Problem Name	Comments
	1 6	7	8 12	13 18	19 80
	SAVE. B SAVE. B SAVE. I SAVE. I	D D	sssss sssss	nnnnnn nnnnnn	

Agendum Name: SAVE. B indicates that the current basis should be retained in disk storage. SAVE. I indicates that the current basis and its inverse should be retained in disk storage.

Storage Code: blank. - Saves basis (and inverse) with current problem.
D - Saves basis (and inverse) at specified disk location.

Starting Sector Location: A five-position numeric field which identifies the disk storage location where the current basis should be stored.

Problem Name: A six-position alphameric field containing the problem name which is used to identify the problem when referenced at a later time.

Comments: Any desired information

Examples

SAVE. B04600PROB1
SAVE. I04200BLENDG

Description

SAVE. p is a method for retaining an advanced basis in disk storage and, if desired, the latest inverse from which to begin a subsequent solution. If the matrix data values are revised between runs, it is not worth storing the inverse; but unless the changes are very significant, it is often valuable to store the basis for subsequent use. If the basis is not saved on disk or recorded on cards through OUTPUT or GETOFF, any subsequent re-solution must start from an initial basis of all logical variables.

To be able to use BASIS. D it is necessary to have used SAVE. B or SAVE. I when that problem was previously solved, or to have previously input a basis.

In executing SAVE. p, the 1620-1311 LP system stores the entire communication region at the referenced location. Therefore the entire problem is effectively stored at the new location. This problem can then be prepared for re-solution through an INPUT. D agendum card. The ASSIGN controls that were in effect when the problem was saved will be restored by the INPUT. D agendum.

FORMAT SUMMARY

The next three pages contain an agendum card summary (see "1620-1311 LP Agenda" for detailed explanation). Following this is a two-page summary of data input cards (see "Data Preparation" for detailed explanation). Below is an index of all the symbols used in the summaries.

SYMBOL EXPLANATION

ccccccc — Column name

d1 — Number of decimal places in primary output value field

d2 — Number of decimal places in secondary output value field

e — Basis classifications

W	Variable basic at lower bound
G	Variable basic at upper bound
F	Variable basic at intermediate level

ffffff — Name of objective function row. If this field is blank the primary (first) objective function will be used during optimization.

hhhhh — Right-hand-side name

nnnnnn — Problem name

rrrrrr — Row name

sssss — Disk sector of data to be stored, referenced, or deleted

t — Type or classification.

Row classifications

b (blank)	identifies an objective function
0 (zero)	identifies an equation
+ (plus)	identifies a less-than inequality or a range constraint
- (minus)	identifies a greater-than inequality
* (asterisk)	identifies an omitted row

Column classifications

b (blank)	identifies a structural variable to be included in the matrix
* (asterisk)	identifies a structural variable to be omitted from the matrix

±uuuu.uuuuuu — Only entry of a single limit right-hand side, or upper row limit of a range constraint

AGENDUM CARDS SUMMARY

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE		COMMENTS
1	7	13	19 24	31 36 41 46 51	56 61 66 73 80
‡,‡,X,E,Q,	L,P,/,6,2,0				
X,					COMMENTS CARD. CØMMENTS IN COL 2-80
I,N,P,U,T,.	C,s,s,s,s,s	n,n,n,n,n,n			INPUT AGENDUM. INPUT INDICATØR CARDS FOLLOW
I,N,P,U,T,.	D,s,s,s,s,s	n,n,n,n,n,n			INPUT AGENDUM. DISK
R,E,V,I,S,E					REVISE AGENDUM. INPUT INDICATØR CARDS FOLLOW
M,I,N,....	.h,h,h,h,h	f,f,f,f,f			MIN AGENDUM. MINIMIZE ØBJECTIVE FUNCTION.
M,I,N,....	.				MIN AGENDUM. MINIMIZE ØBJECTIVE FUNCTIØN
M,A,X,....	.h,h,h,h,h	f,f,f,f,f			MAX AGENDUM. MAXIMIZE ØBJECTIVE FUNCTION
M,A,X,....	.				MAX AGENDUM. MAXIMIZE ØBJECTIVE FUNCTION.
Ø,U,T,P,U,T			dude		ØUTPUT AGENDUM.
Ø,U,T,P,U,T	D,s,s,s,s,s	n,n,n,n,n,n	dude		ØUTPUT AGENDUM. ØUTPUT SELECTED FROM DISK.
D,Ø,.,D,/,J			dll		DØ|D/J AGENDUM. ØUTPUT ALL REDUCED COSTS.
D,Ø,.,D,/,J	D,s,s,s,s,s	n,n,n,n,n,n	dll		DØ|D/J AGENDUM. ØUTPUT SELECTED FROM DISK
C,Ø,S,T,.,R			dll		CØST.R AGENDUM. ØUTPUT ALL CØST RANGES
C,Ø,S,T,.,R	D,s,s,s,s,s	n,n,n,n,n,n	dll		CØST.R AGENDUM. ØUTPUT SELECTED FROM DISK
C,H,E,C,K,.			dude		CHECK. AGENDUM. CHECK ALL RØWS
C,H,E,C,K,.	D,s,s,s,s,s	n,n,n,n,n,n	dude		CHECK. AGENDUM. ROW. ID FILE FROM DISK.

AGENDUM CARDS SUMMARY (Continued)

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE					COMMENTS
1	7	13	19 24	31	36	41	46 51	56 61 66 73 80
E,N,D,J,Ø,B								TERMINATES LP RUNS. END JOB AGENDUM
S,A,V,E,.,B	D,s,s,s,s,s	n,n,n,n,n,n						SAVE AGENDUM. CURRENT BASIS
S,A,V,E,.,B								SAVE AGENDUM. CURRENT BASIS
S,A,V,E,.,I	D,s,s,s,s,s	n,n,n,n,n,n						SAVE AGENDUM. CURRENT BASIS & INVERSE
S,A,V,E,.,I								SAVE AGENDUM. CURRENT BASIS & INVERSE
E,R,A,S,E,B	D,s,s,s,s,s	n,n,n,n,n,n						ERASE AGENDUM. BASIS (AND INVERSE)
E,R,A,S,E,B								ERASE AGENDUM. BASIS (AND INVERSE)
E,R,A,S,E,I	D,s,s,s,s,s	n,n,n,n,n,n						ERASE AGENDUM. INVERSE
E,R,A,S,E,I								ERASE AGENDUM. INVERSE
I,N,V,E,R,T	.,h,h,h,h,h	f,f,f,f,f,f						INVERT AGENDUM. SOLVES SIMULTANEOUS EQUATIONS
G,E,T,Ø,F,F								GETOFF AGENDUM. LP INTERRUPTION
M,A,P,.,.,.								MAP AGENDUM. CURRENT PROBLEM
M,A,P,.,.,.	D,s,s,s,s,s	n,n,n,n,n,n						MAP AGENDUM. PROBLEM SELECTED FROM DISK
P,A,U,S,E,.								PAUSE AGENDUM

AGENDUM CARDS SUMMARY (Continued)

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE		COMMENTS
1	7	13	19 24	31 36 41 46 51	56 61 66 73 80
ASSIGN					ASISIGN AGENDUM
	XX				MANTISSA LENGTH
	X				MASTER TOLERANCE
	X				ELEMENT TOLERANCE
	X				PIVOT TOLERANCE
	X				FEASIBILITY TOLERAINCE
		X			OBJECTIVE FUNCTION TOLERANCE
		X			MAXIMUM ERROR TOLERANCE
		X			OUTPUT UNIT - ITERATION LOG
		X			OUTPUT UNIT - REPORTS
		X			AGENDA LOG UNIT
		X	XXXXX		SECTOR ADDRESS - END OF DIM TABLE, PROGRAMS, DATA
			XXXXX		UPPER DISK LIMIT - LP DATA
			XX		INVERSION FREQUENCY
			X	XX	ITERATION LOG FREQUENCY
				XXXXX	COMMON COMPUTATION AREA SECTOR ADDRESS

DATA INPUT CARDS SUMMARY

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
ROW.ID						ROW ID INDICATOR. ROW IDENTIFICATION CARDS FOLLOW
ROW.ID	DIS,S,S,S,S	NNNNNN				ROW ID INDICATOR. DISK
	t	rrrrrr				ROW IDENTIFICATION
COL.ID						COL ID INDICATOR. COL IDENTIFICATION CARDS FOLLOW
COL.ID	DIS,S,S,S,S	NNNNNN				COL ID INDICATOR. DISK
	t	CCCCCC				COL IDENTIFICATION
MATRIX						MATRIX INDICATOR. MATRIX ELEMENT CARDS FOLLOW.
MATRIX	DIS,S,S,S,S	NNNNNN				MATRIX INDICATOR DISK
	C	CCCCCC	±XXXXX.XXXXXX	±YYYY.YYYYYY	±ZZZZZ.ZZZZZZ	MATRIX ELEMENT
	C	CCCCCC	±XXXXX.XXXXXX	±YYYY.YYYYYY		MATRIX ELEMENT
	C	CCCCCC	±XXXXX.XXXXXX		±ZZZZZ.ZZZZZZ	MATRIX ELEMENT
	C	CCCCCC	±XXXXX.XXXXXX			MATRIX ELEMENT
FIRST.B	B					FIRST RIGHT HAND SIDE INDICATOR
FIRST.B	B	hhhhhh				FIRST RIGHT HAND SIDE INDICATOR
NEXT.B	B	hhhhhh				ADDITIONAL RIGHT HAND SIDE INDICATOR
FIRST.D	DIS,S,S,S,S	NNNNNN				FIRST RIGHT HAND SIDE INDICATOR. DISK
		rrrrrr	±UUUUU.UUUUUU			RIGHT HAND SIDE ELEMENT
		rrrrrr	±UUUUU.UUUUUU	±VVVV.VVVVVV		RIGHT HAND SIDE ELEMENT
BASIS.						BASIS INDICATOR. BASIS CARDS FOLLOW
BASIS.	DIS,S,S,S,S	NNNNNN				BASIS INDICATOR DISK
VARBLS						VARIABLES INDICATOR
	e	CCCCCC				BASIC STRUCTURAL VARIABLES IDENTIFICATION.

DATA INPUT CARDS SUMMARY (Continued)

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE		COMMENTS
1	7	13	19 24	31 36 41 46 51	56 61 66 73 80
SLACKS					SLACKS INDICATOR
	e				NON-BASIC LOGICAL VARIABLES IDENTIFICATIO
ENDATA					TERMINATES INPUT DATA
EOF					TERMINATES INPUT DATA

±vvvv.vvvvvv	Lower row limit of a range constraint
±xxxx.xxxxxx	Data value of a structural variable
±yyyy.yyyyyy	Upper bound of a structural variable
±zzzz.zzzzzz	Lower bound of a structural variable

SAMPLE PROBLEM

The sample problem is a blending problem. It is constructed to illustrate the use of the various agenda and data input formats in the 1620-1311 LP system.

STATEMENT OF PROBLEM

An aluminum alloy smelter wishes to produce a particular alloy at minimum cost. The alloy must, however, meet certain chemical constraints. The smelter has available various scrap materials and some pure metal stocks. There are five scrap materials which are stored in separate bins; each scrap material has been chemically analyzed. The inventory of each bin is known, as is the cost of each scrap material. The pure metal has a stated cost and can be purchased as required. Two of the bins contain powdered scrap; since these oxidize rapidly it is required that at least a certain amount from each of these bins be used in the solution.

The original matrix is displayed on the next page. To assist in understanding the problem, consider these illustrations:

> BIN3, for example, has an inventory of 800 lbs. The end product must include 400 lbs. of this scrap. The composition of the scrap is 2% Fe (iron), 8% Cu (copper), 1% Mn (manganese), no measurable Mg (magnesium), 80% Al (aluminum), and 8% Si (silicon). The market value is estimated at $.17 per pound.
>
> A ton of the end product is required. The composition of the end product (ALOY1) must be less than or equal to: 3% Fe, 5% Cu, 2% Mn, 1.5% Mg. ALOY1 must contain at least 75% aluminum. It must have between 12.5% and 15% silicon.

DESCRIPTION OF SAMPLE PROBLEM INPUT

The sample problem consists of five segments which together use virtually every program in 1620-1311 LP. Each segment is briefly explained below. (These explanations should be examined in conjunction with a review of the input forms which directly follow them.)

Segment 1 ALLOYA

Input (a row identification, matrix element, and one right-hand-side element file) is read from cards and stored on disk. The solution minimizes the objective function, VALUE, for right-hand side, ALOY1. The solution basis is stored on disk. The solution values are reported. Check is performed on all the rows to determine the row computational error. DO. D/J is performed outputting structural variables at their lower and upper bounds, and marginal values for each row.

Segment 2 ALLOYA

The current problem files are revised: the row type of SI is changed from a range constraint (positive slack) to a negative slack; BIN1 is eliminated from the matrix. The aluminum (AL) content of BIN2 is changed to .77, and the inventory set to 3000. The BIN2 copper content (CU) is changed to .03. Change the cost of BIN5 material in objective functions VALUE and VALUE2 to $.13 and -.13 respectively, and lower its inventory to 1000. Raise the required amount of BIN3 to 500. Change the right-hand-side value of FE to 50, and indicate a lower bound of 275 for SI. Take BIN4 out of the basis and put AL into the basis.

		STRUCTURAL VARIABLES							RIGHT-HAND SIDES		
		BIN1	BIN2	BIN3	BIN4	BIN5	ALUM	SILCON	ALOY1	ALOY2	ALOY3
Objective Functions	VALUE	.03	.08	.17	.12	.15	.21	.38			
	VALUE2	-.03	-.08	-.17	-.12	-.15	-.21	-.38			
Constraints	YIELD	1	1	1	1	1	1	1	2000	2000	2000
	FE	.15	.04	.02	.04	.02	.01	.03	≤60	≤40	≤50
	CU	.03	.05	.08	.02	.06	.01		≤100	≤60	≤60
	MN	.02	.04	.01	.02	.02			≤40	≤40	≤35
	MG	.02	.03			.01			≤30	≤30	≤35
	AL	.70	.75	.80	.75	.80	.97		≥1500	≥1600	≥1500
	SI	.02	.06	.08	.12	.02	.01	.97	250 to 300	200 to 300	300 to 400
Bounds	Inventory (Upper)	200	2500	800	700	1500					
	Minimum Required (Lower)			400	100						

Segment 3 ALLOYD

Input from cards a reference column file for later use as selective output.

Segment 4 ALLOYB

Input a new column file. Input the revised matrix files from the disk, and input three right-hand sides. Maximize objective function VALUE2 for right-hand-side ALOY2. Save the solution inverse and basis. Output the solution values. Minimize objective function VALUE for right-hand-side ALOY1. Output the solution value. Input from disk the saved files, and minimize the objective function VALUE for right-hand-side ALOY3 using the saved basis and inverse. Output the solution values and perform a check on the problem.

Segment 5 ALLOYC

Input a set of files from previous problems saved on the disk. Minimize objective function VALUE using right-hand-side ALOY1. Perform a selected output and do D_j using the reference file for segment 3, ALLOYD. Punch out a set of solution basis cards and return to monitor. The basis is erased because major subsequent changes to the matrix are expected to be made.

The actual input sheets for the sample problem are on the next seven pages. Following these pages is the typewriter output which was obtained during the running of the sample problem. The sheets are annotated to describe what the information means. If different ASSIGN cards had been used, the output reports would have appeared on cards or on an online printer.

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. 727 JOB TITLE SAMPLE PROBLEM - SEGMENT 1 ANALYST _____ DATE 7/10 PAGE 1 OF 7

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE				COMMENTS
1	7	13	19 24	31 36 41	46 51		56 61 66 73 80
3,4,0,0,0,3	2,1,0,0,7,0,1	3,1,6,0,0,0,3	2,1,0,0,7,0,2,4,9,0,2,4,0	2,5,1,1,9,6,3,6,1,1,3,0	0,1,0,2,		MONITOR COLD START
,,J,O,B,	5						JOB CONTROL
,,X,E,Q,	L,P,1,6,2,0						EXECUTE LP1620 PROGRAM
I,N,P,U,T,	C,1,2,0,0,0	A,L,L,O,Y,A					
R,O,W,.,I,D							
		V,A,L,U,E,					ROW IDENTIFICATION FILE
		V,A,L,U,E,2					
	0	Y,I,E,L,D,					
	+	F,E					
	+	C,U					
	+	M,N					
	+	M,G					
	-	A,L					
	+	S,I					
M,A,T,R,I,X							
	B,I,N,1	V,A,L,U,E,	.0,3				MATRIX ELEMENT FILE
	B,I,N,1	Y,I,E,L,D,	1.	2,0,0.			
	B,I,N,1	F,E	.1,5				
	B,I,N,1	C,U	.0,3				
	B,I,N,1	M,N	.0,2				
	B,I,N,1	M,G	.0,2				
	B,I,N,1	A,L	.7,0				
	B,I,N,1	S,I	.0,2				
	B,I,N,1	V,A,L,U,E,2	-.0,3				
	B,I,N,2	V,A,L,U,E,	.0,8				
	B,I,N,2	Y,I,E,L,D,	1.	2,5,0,0.			
	B,I,N,2	F,E	.0,4				
	B,I,N,2	C,U	.0,5				

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. 727 JOB TITLE SAMPLE PROBLEM · SEGMENT 1 ANALYST _____ DATE 7/10 PAGE 2 OF 7

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
	BIN2	MIN	.04			
	BIN2	MIG	.03			
	BIN2	AIL	.75			
	BIN2	SI	.06			
	BIN2	VIALUE2	-.08			
	BIN3	VIALUE	.17			
	BIN3	YIELD	1.	800.	400.	
	BIN3	FIE	.02			
	BIN3	CIU	.08			
	BIN3	MIN	.01			
	BIN3	AIL	.80			
	BIN3	SI	.08			
	BIN3	VIALUE2	-.17			
	BIN4	VIALUE	.12			
	BIN4	YIELD	1.	700.	100.	
	BIN4	FIE	.04			
	BIN4	CIU	.02			
	BIN4	MIN	.02			
	BIN4	AIL	.75			
	BIN4	SI	.12			
	BIN4	VIALUE2	-.12			
	BIN5	VIALUE	.15			
	BIN5	YIELD	1.	1500.		
	BIN5	FIE	.02			
	BIN5	CIU	.06			
	BIN5	MIN	.02			
	BIN5	MIG	.01			
	BIN5	AIL	.80			

IBM 1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. 727 JOB TITLE SAMPLE PROBLEM - SEGMENT 1 ANALYST _____ DATE 7/10 PAGE 3 OF 7

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
	BINS	SI	.02			
	BINS	VALUE2-	.15			
	ALUM	VALUE	.21			
	ALUM	YIELD	1.			
	ALUM	FE	.01			
	ALUM	CU	.01			
	ALUM	AL	.97			
	ALUM	SI	.01			
	ALUM	VALUE2-	.21			
	SILCON	VALUE	.38			
	SILCON	YIELD	1.			
	SILCON	FE	.03			
	SILCON	SI	.97			
	SILCON	VALUE2-	.38			
FIRST.	BALØY1					RIGHT HAND SIDE ELEMENT FILE
		YIELD	2000.			
		FE	60.			
		CU	100.			
		MN	40.			
		MG	30.			
		AL	1500.			
		SI	300.	250.		
ENDATA						END ØF INPUT DATA
ASSIGN	0935	1	0	001		
MIN....	ALØY1	VALUE				
SAVE.B						SAVE BASIS
ØUTPUT						
CHECK.						

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. 727 JOB TITLE SAMPLE PROBLEM-SEGMENT 2,3 ANALYST _____ DATE 7/10 PAGE 4 OF 7

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
COST.R						
DØ..D/J			5			ØUTPUT ITØ 5 IDECIMALIS
*. CHECK ØUTPUT AND MAKE REVISIONS TØ INPUT DATA						COMMENTS CARD
PAUSE.						
REVISE						REVISE ICURRENTI PRØBLEIM FILES FØR SEGMENT 2
ROW..ID						
	-	SI				
COL..ID						
	X	BIN1				
MATRIX						
	BIN2	AL	.77	3000.		
	BIN2	CU	.03			
	BIN5	VALUE	.13	1000.		
	BIN3	YIELD			500.	
	BIN5	VALUE2	-.13			
FIRST.	BALØY1					
		FE	50.			
		SI	275.	*		
BASIS.						
VARBLS						
	W	BIN4				
SLACKS						
	F	AIL				
EØF						END ØF INPUT DATA
INPUT.	C0440	ALLØYD				STØRE COLUMN FILE FØR SEGMENT 3
COL..ID						
		ALUM				
		SILCØN				

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. 727 JOB TITLE SAMPLE PROBLEM - SEGMENT 4 ANALYST _____ DATE 7/10 PAGE 5 OF 7

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
E,Ø,F,						END ØF INPUT DATA
I,N,P,U,T,	C,1,0,2,8,0,0	A,L,L,Ø,Y,B,				
R,Ø,W,.,I,D	D,1,0,2,0,0,0	A,L,L,O,Y,A,				RØW ID FILE DISK REFERENCE
C,Ø,L,.,I,D,						
		B,I,N,1,				
		B,I,N,2,				
		B,I,N,3,				
		B,I,N,4,				
		B,I,N,5,				
		A,L,U,M,				
		S,I,L,C,Ø,N				
M,A,T,R,I,X	D,1,0,2,0,0,0	A,L,L,Ø,Y,A,				MATRIX ELEMENT FILE DISK REFERENCE
F,I,R,S,T,.	B,A,L,Ø,Y,1					
		Y,I,E,L,D,	2,0,0,0,.			
		F,E,	6,0,.			
		C,U,	1,0,0,.			
		M,N,	4,0,.			
		M,G,	3,0,.			
		A,L,	1,5,0,0,.			
		S,I,	3,0,0,.	2,5,0,.		
N,E,X,T,.,B,	A,L,Ø,Y,2,					
		Y,I,E,L,D,	2,0,0,0,.			
		F,E,	4,0,.			
		C,U,	6,0,.			
		M,N,	4,0,.			
		M,G,	3,0,.			
		A,L,	1,6,0,0,.			
		S,I,	3,0,0,.	2,0,0,.		

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. 727 JOB TITLE SAMPLE PROBLEM-SEGMENT 4 ANALYST _____ DATE 7/10 PAGE 6 OF 7

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
NEXT.B	.ALØY3					
		YIELD	2000.			
		FE	50.			
		CU	60.			
		MN	35.			
		MG	35.			
		AL	15000.			
		SI	400.	300.		
BASIS.	D02000	ALLØYA				
EØF						END ØF INPUT DATA
ASSIGN	105	1	12000	001		
MAX.....	.ALØY2	VALUE2				
SAVE.I	D10000	INVERS				SAVE INVERSE AND BASIS ØN DISK
ØUTPUT						
MIN.....	.ALØY1	VALUE				
ØUTPUT						
INPUT.	D10000	INVERS				INPUT PRØBLEM DATA FRØM DISK
MIN.....	.ALØY3	VALUE				
ØUTPUT						
CHECK.						
INPUT.	C03600	ALLØYC				START ØF SEGMENT 5
RØW.ID	D02000	ALLØYA				
CØL.ID	D02800	ALLØYB				
MATRIX	D02000	ALLØYA				
FIRST.	D02000	ALLØYA				
BASIS.	D10000	INVERS				
EØF						END ØF INPUT DATA
ASSIGN	0935	1	0	001		

IBM

1620/1311 LINEAR PROGRAMMING SYSTEM

INPUT DATA FORMAT

JOB NO. 727 JOB TITLE SAMPLE PROBLEM - SEGMENT 5 ANALYST _____ DATE 7/10 PAGE 7 OF 7

KEY PUNCH: PUNCH DEC. PT. IN COL. 24

INDICATOR CARDS	COLUMN NAME	ROW NAME	VALUE			COMMENTS
1	7	13	19 24	31 36 41	46 51	56 61 66 73 80
MIN.....	.ALØY1	VIALUE				
ØUTPUT	D04400	ALLØYD				
DØ.D/J	D04400	ALLØYD				
CØST..R	D04400	ALLØYD				
GETØFF						PUNCH ØUT SØLUTION BASIS ON CARDS
ERASEB						ERASE BASIS
MAP.....	D02000	ALLØYA				
MAP.....	D02800	ALLØYB				
MAP.....	D03600	ALLØYC				
MAP.....	D04400	ALLØYD				
ENDJØB						END ØF PRØBLEM

TYPEWRITER OUTPUT - SAMPLE PROBLEM

```
‡‡JOB 5                                                    JOB CONTROL

‡‡XEQ LP1620                                               EXECUTE LP1620 PROGRAM
EXECUTION

NOISE DIGIT 0                                              PRODUCED BY
MANTISSA LENGTH 08                                           1620 MONITOR
SUBR SET 02
UPPER LIMIT DIM = 07799
LP1620-1621

INPUT.CO2000ALLOYA                                         SEGMENT 1
INPUT.
ROW.ID
MATRIX
FIRSTB
ASSIGN0935     1              001
MIN....ALOY1VALUE
LP1621 TO INVERT

NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
LP1621 TO DUAL
NOISE DIGIT 0
MANTISSA LENGTH 09                                         ITERATION LOG
SUBR SET 02
 ITER  VARBL. EXIT  VARBL. ENTR  CNT  OBJ. FUNCTION
 0001              BIN1   U.B.  003        86.000
 0002  YIELD  U.B.  BIN2         003       190.000
 0003  SI     U.B.  SILCON       005       230.879
 0004              BIN4   U.B.  004       243.010
 0005              BIN1   L.B.  004       250.373
 0006  AL          ALUM         004       288.007
 0007  FE          BIN4         001       295.298
 0008  MN          BIN3         001       296.216
FEASIBLE
OPTIMUM
SAVE.B                                                     SAVE BASIS
OUTPUT
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
```

OUTPUT REPORT

```
OUTPUT
BASIS.              MANTISSA 09    TOLERANCES  05 03 03 03 03
VARBLS  TYPE  NAME  ACTIVITY LEVEL
           FBIN2          665.343
           FBIN3          490.253
           FBIN4          424.188
           FALUM          299.639
           FSILCON        120.578
SLACKS  TYPE  NAME  ACTIVITY LEVEL  SIMPLEX MULT.
           FVALUE         296.217
           FVALUE2        296.217-
          OGYIELD            .000          .014
          +WFE                            2.568
          +FCU             16.032
          +WMN                             .544
          +FMG             10.040
          -WAL                             .252-
          +GSI             50.000          .485-
CHECK.
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
```

CHECK REPORT

```
CHECK.    ROW  NAME    UPPER LIMIT    SOL. VALUE    LOWER LIMIT     ROW ERROR
             VALUE                      296.217            .000     .00000000
             VALUE2                     296.217-           .000     .00000000
             YIELD       2000.000      2000.000                     .00002000
             FE            60.000        60.000                     .00000010-
             CU           100.000        83.968                     .00000050
             MN            40.000        40.000                     .00000010-
             MG            30.000        19.960                     .00000010-
             AL                         1500.000       1500.000     .00001000
             SI           300.000       250.000        250.000      .00000300
* MAX ERROR =            .00002000
COST.R
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
```

COST.R REPORT

```
COST.R  NAME    CURRENT COST    HIGHEST COST  HI-VAR   LO-VAR   LOWEST COST
      BIN2              .080            .089  MN       BIN1             .017
      BIN3              .170            .179  BIN5     MN               .160
      BIN4              .120            .147  BIN1     MN               .109
      ALUM              .210            .226  MN       AL               .189
      SILCON            .380            .467  MN       GSI              .148
```

```
DO.D/J
NOISE DIGIT 0
MANTISSA LENGTH 09                                          DO.D/J REPORT
SUBR SET 02
 VBLS    TYPE  NAME   CURRENT COST   REDUCED COST    BASIS VALUE
            WBIN1            .030           .254          .224-
            WBIN5            .150           .015          .135
 ROWS    TYPE  NAME   INCR B VALUE   DECR B VALUE
            OYIELD           .014
            +FE             2.568
            +CU
            +MN              .544
            +MG
            -AL                             .252
         RANGE+SI                           .485
                                                          SEGMENT 2
*   CHECK OUTPUT AND MAKE REVISIONS TO INPUT DATA       COMMENTS CARD
PAUSE.
REVISE

REVROW
REVCOL
REVMAT
REVFST
REVBAS
REVSLK

INPUT.C04400ALLOYD                                        SEGMENT 3
INPUT.
COL.ID

INPUT.C02800ALLOYB                                        SEGMENT 4
INPUT.
ROWDSK
COL.ID
MTXDSK
FIRSTB
BASDSK
ASSIGN105       1       12000   001
```

```
MAX....ALOY2VALUE2
LP1621 TO INVERT

NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
LP1621 TO DUAL
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
```

ITERATION LOG

ITER	VARBL. EXIT	VARBL. ENTR	CNT	OBJ. FUNCTION
0001		SI L.B.	003	499.560-
0002		BIN5 U.B.	003	423.560-
0003	BIN3	MN	003	335.755-
0004	AL	BIN5	003	354.631-
0005	CU	AL	001	371.486-
0006	BIN5	BIN4	001	373.686-

```
FEASIBLE
OPTIMUM
SAVE.ID10000INVERS
OUTPUT
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
OUTPUT
```

OUTPUT REPORT

BASIS. MANTISSA 10 TOLERANCES 05 05 05 05 05

VARBLS	TYPE	NAME	ACTIVITY LEVEL	SIMPLEX MULT.
		FBIN2	190.678	
		WBIN3	500.000	
		FBIN4	233.051	
		FALUM	961.864	
		FSILCON	114.407	
SLACKS	TYPE	NAME	ACTIVITY LEVEL	SIMPLEX MULT.
		FVALUE	373.686	
		FVALUE2	373.686-	
		OGYIELD	.000	.264-
		+WFE		2.843
		+WCU		2.754
		+FMN	22.712	
		+FMG	24.280	
		-FAL	54.619	
		-WSI		.208-

```
MIN....ALOY1VALUE
LP1621 NEW RHS
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
LP1621 TO DUAL
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
 ITER  VARBL. EXIT  VARBL. ENTR  CNT  OBJ. FUNCTION
 0001  BIN4         BIN5         005       267.586
 0002  ALUM         CU           003       273.623
 0003  AL           ALUM         002       287.986
 0004  MN           BIN4         001       292.607
FEASIBLE
OPTIMUM
OUTPUT
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
OUTPUT
BASIS.              MANTISSA 10    TOLERANCES  05 05 05 05 05
VARBLS  TYPE  NAME  ACTIVITY LEVEL
            FBIN2          574.696
            WBIN3          500.000
            FBIN4          469.636
            FBIN5          130.972
            FALUM          206.680
            FSILCON        118.016
SLACKS  TYPE  NAME  ACTIVITY LEVEL  SIMPLEX MULT.
            FVALUE         292.607
            FVALUE2        292.607-
           OGYIELD             .000          .036
           +WFE                             2.324
           +FCU             23.441
           +WMN                              .769
           +FMG             11.449
           -WAL                              .273-
           -WSI                              .501-
INPUT.D10000INVERS
INPUT.
MIN....ALOY3VALUE
LP1621 NEW RHS
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
```

ITERATION LOG

OUTPUT REPORT

```
LP1621 TO DUAL
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
 ITER  VARBL. EXIT  VARBL. ENTR  CNT  OBJ. FUNCTION
 0001  BIN4   U.B.  BIN1          002       368.091
 0002  BIN2         FE            001       369.010
 0003  BIN1         BIN4          001       369.205
FEASIBLE
OPTIMUM
OUTPUT
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
OUTPUT
BASIS.                MANTISSA 10    TOLERANCES  05 05 05 05 05
VARBLS  TYPE  NAME  ACTIVITY LEVEL
          WBIN3          500.000
          FBIN4          677.570
          FALUM          644.860
          FSILCON        177.570
SLACKS  TYPE  NAME  ACTIVITY LEVEL  SIMPLEX MULT.
          FVALUE         369.206
          FVALUE2        369.206
          OGYIELD            .000          .307-
          +FFE              1.121
          +WCU                            9.822
          +FMN             16.449
          +FMG             35.000
          -FAL             33.692
          -WSI                             .075-
CHECK.
NOISE DIGIT 0
MANTISSA LENGTH 10
SUBR SET 02
CHECK.   ROW  NAME   UPPER LIMIT    SOL. VALUE   LOWER LIMIT     ROW ERROR
           VALUE                      369.206          .000      .00000140-
           VALUE2                     369.206-         .000      .00000140
           YIELD      2000.000       2000.000                   .00000600-
           FE           50.000         48.879                   .00000015-
           CU           60.000         60.000                   .00000084-
           MN           35.000         18.551                   .00000005-
           MG           35.000           .000                   .00000002
           AL                         1533.692      1500.000     .00000537-
           SI          400.000        300.000       300.000     .00000090-
* MAX ERROR =        .00000600-
```

ITERATION LOG

OUTPUT REPORT

CHECK REPORT

```
INPUT.C03600ALLOYC                                          SEGMENT 5
INPUT.
ROWDSK
COLDSK
MTXDSK
FSTDSK
BASDSK
ASSIGN0935    -1              001
MIN,...ALOY1VALUE
LP1621 TO INVERT

NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
LP1621 TO DUAL
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
 ITER  VARBL. EXIT  VARBL. ENTR  CNT  OBJ. FUNCTION        ITERATION LOG
 0001  BIN4         BIN5         005       316.236
 0002  AL           CU           001       333.611
FEASIBLE
OPTIMUM
OUTPUTD04400ALLOYD
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
OUTPUT                                                  OUTPUT REPORT
BASIS.              MANTISSA 09    TOLERANCES  05 03 03 03 03
VARBLS   TYPE  NAME ACTIVITY LEVEL
           FALUM         416.612
           FSILCON       187.993
DO.D/JD04400ALLOYD                                      DO.D/J REPORT
NOISE DIGIT 0
MANTISSA LENGTH 09
SUBR SET 02
 VBLS    TYPE  NAME  CURRENT COST  REDUCED COST   BASIS VALUE
 ROWS    TYPE  NAME  INCR B VALUE  DECR B VALUE
COST.RD04400ALLOYD
NOISE DIGIT 0                                           COST.R REPORT
MANTISSA LENGTH 09
SUBR SET 02
COST.R  NAME    CURRENT COST    HIGHEST COST  HI-VAR   LO-VAR  LOWEST COST
      ALUM              .210            .230  BIN3     AL              .158
      SILCON            .380            .480  BIN3     SI              .174-
```

```
GETOFF
GETOFF
ERASEB
-REVSLK  —                                                    MAP REPORT
MAP...D02000ALLOYA
MAP...S02000/02036 ALLOYA. 009ROW,0007COL.MANSA-09-COMPUTE-000086-SECTORS.
MAP...D02800ALLOYB
MAP...S02800/02813 ALLOYB. 009ROW,0007COL.MANSA-09-COMPUTE-000086-SECTORS.
MAP...D03600ALLOYC
MAP...S03600/03605 ALLOYC. 009ROW,0007COL.MANSA-10-COMPUTE-000086-SECTORS.
MAP...D04400ALLOYD
MAP...S04400/04406 ALLOYD. 000ROW,0002COL
ENDJOB
END OF JOB                                                    END OF SAMPLE
```

PROGRAM DECK

The 1620-1311 Linear Programming System is supplied to the user in the form of a self-loading card deck, which is to be loaded onto disk under control of the 1620 Monitor System. To load the program, place the deck in the card reader, press RESET on the console, and then press LOAD on the card reader.

The program deck consists of a series of object decks for the individual routines included in the 1620-1311 LP System. The first three cards of each object deck are:

Monitor	++JOB 5 card
Monitor	++DUP card
Monitor	*DLOAD card

See Bibliography Reference 12, for Monitor card format.

These cards call in Monitor, which will load the individual routine on disk and place the appropriate entries into the Monitor DIM Table and Equivalence Table. The entire 1620-1311 LP System program deck is arranged as follows:

Monitor cold start card ++JOB 5 card ++DUP card *DLABL 11111 card	These four cards MUST appear here.
Object deck, INPUT.	
Object deck, ROW.ID	
Object deck, COL.ID	
Object deck, MATRIX	
Object deck, FIRSTB	
Object deck, BASIS.	
Object deck, INPUTA	
Object deck, INPUTB	
Object deck, INPUTC	
Object deck, INPUTD	
Object deck, MTXDSK	
Object deck, REVISE	
Object deck, REVROW	
Object deck, REVCOL	
Object deck, REVMAT	
Object deck, REVSLK	
Object deck, REVFST	
Object deck, REVBAS	
Object deck, SAVE.B	
Object deck, OUTPUT	
Object deck, DO.D/J	
Object deck, GETOFF	
Object deck, COST.R	
Object deck, CHECK.	
Object deck, INVRT1	
Object deck, INV002	
Object deck, INV003	
Object deck, INV004	

Object deck, INV005
Object deck, IVLAST
Object deck, NEWRHS
Object deck, LP1620
Object deck, LP1621
Object deck, 1DUAL
Monitor ++++ card

SYSTEM ORGANIZATION

All the routines included in the 1620-1311 LP System reside on the disk; they are called into memory individually as required. The sequence in which routines are called in and executed depends completely on the control cards which form the input deck to an LP run; these cards are called agendum cards.

The executive program, called LP1621, reads an agendum card and calls in the required routine or series of routines. For example, when the DO.D/J agendum card is used, the DO.D/J routine is executed, and LP1621 reads in the next agendum cards. When MIN... (or MAX...) is used, however, LP1621 initiates an optimizing loop consisting of INVERT, DUAL-PRIMAL, CHECK. The loop continues until optimum solution is reached (unless the problem goes infeasible or unbounded, or a major error occurs); it is not until optimum solution is reached that another agendum card is read in.

The agendum cards are always entered through the card reader. The data cards may be entered through the card reader, in which case the INPUT. agendum card will read INPUT.C in columns 1-7. However, the data used in a particular run may already be on the disk, in which case an INPUT.D agendum card is used. When input data is on disk, selected values may be changed through use of the REVISE agendum card without requiring the entire data file to be re-entered from the card reader. This means that the same basic data files stored on disk may be used in a number of LP runs.

INPUT/OUTPUT ROUTINES

All the programs of the 1620-1311 LP System use the GET and PUT statements of the 1620 Monitor System for input/output. Thus the Monitor System I/O routines are in memory at all times, regardless of which LP System routine is being executed. All I/O messages are Monitor messages.

OPERATING INSTRUCTIONS

LOADING THE SYSTEM

Before an LP problem can be run, the 1620-1311 Linear Programming System must be on the disk; once the system has been loaded, it need not be reloaded before each LP run.

To load the system onto disk:

- Load IBM 1620-1311 Monitor 1 on the disk
- Place the LP actual deck in the card reader
- Set all console sense switches OFF

- Press RESET on the console
- Press LOAD on the card reader

When the LP System has been loaded, the message END OF JOB will be typed out.

RUNNING AN LP PROBLEM

1. Place the LP deck in the card reader. The deck should be arranged as follows:

When LP deck is first in the stack of Monitor jobs

Cold start card
++JOB 5 card
++XEQ LP1620 card
:
ENDJOB card
++ PAUS (optional)

When LP deck is stacked behind another Monitor job

++JOB 5 card
++XEQ LP1620 card
:
ENDJOB card
++ PAUS (optional)
++JOB 5 card
:
(next Monitor job)

These control cards (except ++ PAUS) are required, regardless of whether the data is in the card reader or on disk. When the 1620-1311 LP System reads the ENDJOB card, it will return control to the 1620 Monitor System, which then types END OF JOB. If the ++ PAUS card is used, there will be a halt; if not, 1620 Monitor will then try to read another card from the card reader. (See Bibliography Reference 12, for Monitor card format.)

2. Set the console sense switches.

SWITCH	IF ON	IF OFF
1	Each iteration will be logged on the typewriter. This switch setting may be changed as often as desired during problem run. Iterations will also be logged per ASSIGN control.	Iterations will be logged per ASSIGN control.
2	Used to interrupt optimization procedure. (See previous description of GETOFF procedure.)	Normal operation.
3	Will not halt or type MINOR ERROR if minor errors occur. Will supress ***** NOT IN COL FILE and ***** ROW NOT IN FILE messages which occur in INPUT when selecting columns and rows respectively from MATRIX and RHS files.	Will type MINOR ERROR and halt if minor error occurs. The position of switch 4 controls the succeeding action.

SWITCH	IF ON	IF OFF
4	Will type MAJOR ERROR and halt if major error occurs. If START is then pushed, will read in next control card and attempt to continue.	Will type MAJOR ERROR and halt if major error occurs. If START is then pushed, will execute ENDJOB.

3. Set all console check switches to PROGRAM.

4. Press RESET on console.

5. Press LOAD on card read punch if LP deck is first in the stack of Monitor jobs.

6. After a ++ PAUS card, set console switches as desired and press START.

Some fields on the Monitor control cards which are needed to describe the 1620 system used are listed below. (The Monitor control cards are fully described in Bibliography Reference 12.)

++JOB

Col.	1 - 5	++JOB	
	7	5	must always be present
	8 - 11	blank	if 1 disk system
		01	if 2 disk system
		012	if 3 disk system
		0123	if 4 disk system

++XEQ

Col.	1 - 5	++XEQ	
	7 - 12	LP1620	
	31 - 32	03	if system has floating-point hardware
		blank	otherwise

*DFINE

Col.	1 - 6	*DFINE
	18	The number of disk storage drives on the system. May be one, two, three, or four. One drive is assumed if an *DFINE card is not present. See Reference 12 for a description of where the *DFINE card should be placed.

HALTS AND MESSAGES

The following list contains all messages which will be typed out during a 1620-1311 LP System run, except for the following:

As each agendum card is read, it is logged (on unit ASSIGNed for logging).

All comments cards (* in column 1) in the agendum stream are logged.

All disk I/O operations are under control of the Monitor I/O System, and any disk I/O messages are Monitor messages. If any disk I/O errors occur, a message is typed out for appropriate action. (See Reference 12.)

The two program switches in the 1620-1311 LP System are the minor and major error switches. When any program finds an error condition, it types out an appropriate message, sets the program error switch, continues processing if possible, and then returns control to the executive program (LP1621). If the error is minor, the action of the executive program depends upon the setting of console switch 3. If ON, the error is ignored and the LP run continues without a halt. If OFF, the minor and major messages are typed and the program halts. If the error is major, the major error message is always typed and the program halts.

The action that the user should take when an error halt occurs depends upon the cause of the error. On the following pages are listed all the error messages. The Error Restart procedures referenced are described later under "Error and Interrupt Restart Procedures".

There are several messages for which no restart procedure is given. These are due to program errors. If the user has modified the programs, he must correct his modifications.

*MAX ERROR = xxxx.xxxxxxxx- CHECK.

Information message only. Maximum row error generated during the preceding set of iterations. A CHECK and reinversion is automatically taken as specified by the inversion frequency. This message occurs at the end of CHECK. and yields the maximum error detected. This error is used to adjust the mantissa length and tolerances.

****** COL NO LOWER BOUND REVMAT

Minor error. ****** is the name of a column without a lower bound. It is not possible to revise the lower bound of a variable unless the variable is input with an explicit lower bound. The Revise Matrix element card contains a lower bound. If the card is mispunched use Error Restart B. If the lower bound is required use Error Restart C or D.

****** COL NO UPPER BOUND REVMAT

Minor error. Same as above except that this message is to the upper bound.

****** COL NOT IN FILE REVCOL

Minor error. ****** is a column name which is not in the COL. ID file. If the Revise column identification card is misspelled, use Error Restart B. If the column identification is to be added, use Error Restart C or D.

****** COL NOT IN MATRIX REVMAT

Minor error. ****** is the name of a column which is not in the MATRIX file. The revise matrix element card identifies a column not in the matrix. If the column name is misspelled, use Error Restart B. If a column is to be added to the matrix, use Error Restart C or D.

****** DUP BASIS NAME BASIS.

Minor error. ****** is the name of a variable (structural or logical) that has already been input. A second basis identification card names a previously read variable. The entry is correct if the first of the duplicate names specifies the correct basis type. In that case use Error Restart F. If the first of the duplicate basis cards specifies an incorrect basis type, use Error Restart G or A.

****** DUP COL COL.ID

Minor error. See above, except that reference is to column name. A second column identification card names a previously read variable. The entry is correct if the first of the duplicate names specifies the correct basis type. In that case use Error Restart F. If the first of the duplicate basis cards specifies an incorrect column type, use Error Restart A or E.

****** DUP COL NAME INPUT B

Minor error. ****** is the name of two columns in the MATRIX file. Detected in the generation of a COL.ID file from the MATRIX file. This is usually due to: cards out of order in the MATRIX deck, a blank card in the MATRIX deck, two variables with the same name, two sets of coefficients for the same variable, or a misspelled variable name. Use Error Restart A.

****** DUP NAME IN AJ FILE INPUT C

Minor error. ****** is the name of two columns in the AJ VAR or MATRIX file. Detected in initialization of the MATRIX file from the COL.ID file. See preceding message for cause and action.

****** DUP NAME IN AJ SLK FILE INPUT D

Minor error. ****** is the name of two logical variables. Detected in initialization of AJ SLACK file from BASIS SLACK file. The program which generated the ROW.ID file, or the program which generated the logical variables from the ROW.ID file is in error. This error may occur due to user modification or replacement of input routines. The second logical variable will be classified as an omit (*) variable regardless of the row type.

****** DUP NAME IN AJ VAR FILE INPUT D

Minor error. ****** is the name of two columns in the MATRIX file. Detected in the initialization of the AJ VAR (MATRIX) file from the BASIS VARIABLE file. See ****** DUP COL NAME message for cause and action.

****** DUP ROW ROW.ID

Minor error. ****** is the name of a row that has already been input. A second row identification card names a previously read row. The entry is correct if the first of the duplicate names specifies the correct row type. In that case use Error Restart F. If the first of the duplicates specifies an incorrect row type, use Error Restart G or A.

****** G SLK NO UPPER BD OUTPUT

Major error. The logical variable generated for row ****** has no upper bound. This variable is classified as a type G variable in the AJ SLACK file. The input or revise basis cards specified that the logical variable was a type G variable. This classification is valid only for equation rows and range constraints. Use Error Restart B if revised incorrectly. Use Error Restart A or G if input incorrectly. This error may also occur due to modification of optimization programs.

****** G VAR NO UPPER BD OUTPUT

Major error. The structural variable named ****** has been classified as a type G variable in the AJ VARIABLE or MATRIX file. This variable does not have an upper bound. The MATRIX file has been misused by an optimization program. This may be due to modification of an optimization program.

nnnnnn* IN AJ SLK FILE INPUT D

Minor error. nnnnnn is the name of a row classified as an OMIT (*) row. The logical variable for this row must not appear in the basis slack file. A basis identification card named the logical variable for an omit row. The row remains classified as an omit row and the basis classification is ignored. If that is the desired result, use Error Restart F. To enable this row to be used as a constraint, correct the row identification card and use Error Restart A or E.

nnnnnn* IN AJ VAR FILE INPUT D

Minor error. nnnnnn is the name of a variable classified as an omit (*) in the MATRIX and COL.ID files. A basis identification card has been read for this variable. Omitted columns cannot be specified in the basis variable cards. The basis classification is ignored and the column remains omitted. If that is desired, use Error Restart F. If the variable is to be included in the problem, prepare a column identification card and use Error Restart A or E.

****** INP ROW NOT IN MTRX ROW FILE MTXDSK

Major error. ****** is the name of a row in the ROW.ID file for the problem which is not in the ROW.ID file of the matrix. A row identification card is mispunched. Use Error Restart A.

****** IS NOT IN EQU TABLE LP1621

Major error. The agendum name ****** col. 1-6 of the current agenda card does not correspond to any of the subroutines of LP1621 and this name is not in the IBM 1620 Monitor Equivalence (program name) table. The agendum card is misspelled (DObD/J instead of DO.D/J for example), or cards are out of order in the agenda deck, or there are duplicate EOF or ENDATA cards. Use Error Restart H.

****** NO ENTRY IN RHS nnnnn REVFST

Minor error. The input RHS nnnnn did not have an entry for row ******. A right-hand-side Revise element card has been read for this coordinate. If no entry is required, use Error Restart F. If this entry is required, use Error Restart C or D. If the card is misspelled, use Error Restart B.

****** NO ROW ENTRY IN COL nnnnnn REVMAT

Minor error. The input column nnnnnn did not have an entry for row ******. A Revise matrix element card specifies this coordinate. If no entry is required, use Error Restart F. If the card is misspelled, use Error Restart B. If this entry is required, use Error Restart C or D.

****** NOT IN BASIS YYY FILE REVBAS

Minor error. ****** is the name of a variable which is not in a basis file. YYY = SLK, basis slack file. YYY = VAR, basis variable file. A Revise basis identification card names a variable not in the basis file. If the card is misspelled, use Error Restart B. If this variable must be reclassified from the initial status (W - structural variables; F - logical variables), use Error Restart C or D.

****** NOT IN COL FILE INPUT C

Minor error. ****** is the name of a variable which is in the MATRIX file but not in the column identification file. This message is suppressed if console switch 3 is on during INPUT. The input COL.ID file did not include this variable, which is in the MATRIX. This column is classed omit (*). If that is desired, use Error Restart F. If the variable is to be included in the problem, use Error Restart A.

***** RHS NOT IN B FILE REVFST

Minor error. There is no RHS named ***** in the B (right-hand-side) file. The right-hand side specified by the FIRST.B or NEXT.B Revise indicator card does not exist. The right-hand-side element cards following this card are ignored. If the FIRST.B or NEXT.B card is misspelled, use Error Restart B. If a right-hand side is to be added, use Error Restart C or D.

****** ROW NOT IN BETA 1 OUTPUT

Major error. ****** is the name of a variable classified as a type F in the MATRIX or AJ VARIABLE FILE (structural), or the AJ SLACK FILE (logical). This name does not appear in the solution list BETA 1. This error is probably due to file destruction or misuse by an optimization or output program: 1 DUAL, INVERT, CHECK, OUTPUT, DO.D/J, or COST.R. It may be due to user program modification or replacement.

****** ROW NOT IN FILE FIRST B
MATRIX

Minor error. Row named ****** is not in the ROW.ID file and is specified in a RHS row coordinate or on a MATRIX element card. This message is suppressed if console switch 3 is on during INPUT. (Note: If switch 3 is off and there is a blank card in the MATRIX cards, the message will appear to be "bbbbbb ROW NOT IN FILE".) The input ROW.ID file did not include this variable. This card is bypassed and no entry occurs. If that is desired, use Error Restart F. If the row is desired, use Error Restart A.

****** ROW NOT IN FILE REVROW

Minor error. The input ROW.ID file did not include a row named ******. If the REVISE row identification card is misspelled, use Error Restart B. If a row is to be added, use Error Restart C or D.

****** ROW NOT IN ROW.ID FILE — REVFST, REVMAT

Minor error. Row ****** is not in the ROW.ID file. The Revise matrix element or right-hand-side element card specifies a row name not in the ROW.ID file. If the card is misspelled, use Error Restart B. If a row is to be added to the MATRIX (and RHS), use Error Restart C or D.

****** SLK NOT IN ROW.ID FILE SEQUENCE — REVSLK

Minor error. The logical variable generated for row ****** is not in the same order as the ROW.ID file. The AJ SLACK file has been generated incorrectly. This may be due to modification of input programs.

A AND/OR SLACK FILE NOT DEFINED — CHECK., SAVE.p, DO.D/J, NEWRHS

Major error. The AJ VAR (MATRIX) or the AJ SLACK file does not exist. This file(s) is required for processing the agendum. The agenda is out of order. Use Error Restart H.

AGENDA NAME NOT EQUAL CNTL. FILE NAME — DO.D/J

Major error. The name on the DO.D/J agendum card does not match the specified control file name. The agendum card is mispunched, either the problem name and/or the control file location is incorrect. Use Error Restart H.

BASIS FILE FULL — BASIS.

Minor error. The number of variables in the BASIS variable file exceeds the number of columns in the matrix, or the number of logical variables exceeds the number of rows in the ROW.ID file. Remove the unnecessary basis cards. Use Error Restart A.

BETA FILE NOT DEFINED — CHECK.

Major error. The CHECK. agendum is called for before the Beta file (names and activity levels of intermediate-level basis variables) has been formed. The agendum cards are out of order. Use Error Restart H.

CANNOT FIND MATCHING RHS — CHECK.

Major error. The RHS name specified in columns 8-12 of the INVERT, MIN, or MAX agendum card is not the name of an R-H-S in the RHS file. Correct the agendum card and use Error Restart H.

CANT FIND NAMED RHS — NEWRHS

Major error. The RHS name specified in columns 8-12 of the INVERT, MIN, or MAX agendum card is not the name of an R-H-S in the RHS file. Correct the agendum card and use Error Restart H.

CANT CONTINUE. INV FILE GOING BEYOND USER AREA. — INVERT

Major error. The upper limit ASSIGNed by the user has been exceeded.
1. Preferred method. Repunch the ASSIGN card with a higher upper limit or a lower common computation area sector address. Use Error Restart H.
2. Specify a lower starting sector address on the INPUT card. Use Error Restart A. Depending upon the size of the problem, this method may or may not yield a successful run.

CANT CONTINUE. INV FILE GOING INTO DIM TABLE. — INVERT

Major error. The inverse file is about to exceed the disk storage area allowed. If the program had not halted, the Monitor DIM table required by the system would be destroyed.
1. Preferred method. Prepare an ASSIGN card with
   a. Sector address -- DIM
   b. Sector address -- UL
   c. Common computation area sector address specifying an area above the DIM table large enough for the inverse file.

   Use Error Restart H.
2. Specify a lower starting sector address on the INPUT card. Use Error Restart A. Depending upon the size of the problem, this method may or may not yield a successful run.

CANT FIND OBJ FCT IN ROWID — LP1621 NEWRHS

Major error. The objective function specified on the MIN or MAX card is not in the ROW.ID file. The agendum card is mispunched. Use Error Restart H.

CDP ERR $\bar{X}$XXXX

Monitor error message; program halts. May be due to invalid disk sector addresses (multiple drives). The disks on drive n (as specified on the Monitor JOB card) may have invalid sector addresses. Usually occurs when a disk defined for use on drive 0 (sector addresses 00000-19999) is placed on drive n, $n \neq 0$. May also be due to program error.

CNTRL CARD NAME DOES NOT MATCH PRB CTL FILE — INPUT A

Major error. The input control card (ROW.ID, BASIS., MATRIX, FIRST.D, COL.ID) problem name does not match the specified control file name. The control card is mispunched. Use Error Restart A.

COL FILE BEING MADE IS OVER LIMIT — INPUT B

Minor error. The number of columns in the MATRIX file exceeds capacity. The remaining columns are not entered in the COL.ID file and will subsequently be classed omit (*) in the MATRIX file. The problem is too large for the machine size. Use Error Restart A.

COL FILE OVER LIMIT — COL.ID

Minor error. The number of column names in the column identification file exceeds capacity. The remaining column names are bypassed. The problem is too large for the machine size. Use Error Restart A.

DATA ERROR CANNOT OPERATE WITH MATRIX FILE — 1 DUAL, COST.R

Major error. Vector product exponent exceeds 99. Program cannot operate with inverse and/or data files. The INVERSE file has been destroyed by a disk operation, or the MATRIX file has been incorrectly formed or has subsequently been destroyed. If the INVERSE is destroyed, use Error Restart I. If the MATRIX files have been destroyed, use Error Restart C or D. Incorrect MATRIX file formation may be due to modification or replacement of input routines.

DATA ERROR, TYPE F NOT IN BETA — COST.R

Major error. A variable is classified as a type F variable in the AJ VAR (MATRIX) file. This variable is not in the Beta file (list of type F variables). The error is probably due to file destruction or misuse by an optimization or output program: 1DUAL, INVERT, CHECK., OUTPUT, DO.D/J, or COST.R. It may be due to user modification or replacement.

DSK ERR $\bar{0}\,6\,\bar{0}\,7\,\bar{1}\,6\,\bar{1}\,7\,\bar{3}\,6\,\bar{3}\,7\,\bar{3}\,8$ — MONITOR

Monitor error message. Machine halts. Error is probably due to incorrect sequence of agendum cards. Use Error Restart A.

DUP B FILE — INPUT A

Major error. A FIRST.B input control card is read when a B or RHS file has already been input. The input deck is invalid. Use Error Restart A.

DUP BASIS FILE — INPUT A

Major error. A BASIS. input control card is read when a BASIS. file has already been input. The input deck is invalid. Use Error Restart A.

DUP COL FILE — INPUT A

Major error. A COL.ID input control card is read when a COL.ID file has already been input. The input deck is invalid. Use Error Restart A.

DUP MATRIX FILE — INPUT A

Major error. A MATRIX input control card is read when a MATRIX file has already been input. The input deck is invalid. Use Error Restart A.

DUP ROW FILE — INPUT A

Major error. A ROW.ID input control card is read when a ROW.ID file has already been input. The input deck is invalid. Use Error Restart A.

DUPL ENTRY — FIRSTB; MATRIX

Minor error. A duplicate data entry for RHS or MATRIX element card is found. The duplicate card is punched out. The first element is stored in the disk MATRIX, the second is ignored. If the first element is correct, use Error Restart F. If the first element is not correct, use Error Restart A or E.

ENT ERROR $\overline{0}\,6\,\overline{0}\,7\,\overline{1}\,6\,\overline{1}\,7\,\overline{3}\,6\,\overline{3}\,7\,\overline{3}\,8$ MONITOR

Monitor error message. Computation continues. A preceding I/O operation is erroneous and caused an error indicator to be turned on. Error may be due to machine malfunction or program error. If machine malfunction, use Error Restart K.

ERROR IN INPUT CONTROL CARD INPUT.

Major error. Card column 7 of the INPUT. agendum card is not C or D. Use Error Restart A.

ETA11 FILE NOT DEFINED CHECK.
NEWRHS

Major error. The ETA11 file does not exist. The agendum cards are out of order or a program error resulted in file destruction. Use Error Restart H if the agendum cards are out of order.

FEASIBLE 1 DUAL

Information message; no halt occurs. The current solution is now feasible.

FILE MISSING xxxxxx COST.R
1DUAL

Major error. xxxxxx File is required for processing and has not been input or formed. (Can be Beta, MATRIX, AJ SLACK, ETA 11, or INVERSE file.) Agendum cards are out of order, no (or inadequate) input data exists, or COST.Ranging precedes optimization. Use Error Restart H.

FILE OVER LIMIT BASIS.

Minor error. The number of structural (logical) variables in the BASIS VARIABLES (SLACK) file exceeds machine-file capacity. The problem is too large for machine-size capacity. Use Error Restart A.

G W/O UPBND INVERT

Major error. A variable has been classified as a type G (in basis at upper bound), but the variable has no upper bound. Use a GETOFF agendum to determine the variables and types, correct the types and use Error Restart G.

ILLEGAL MANTISSA LENGTH - xx INVERT

Major error. The mantissa length specified on the ASSIGN card is not in the allowable range (5-18). Correct the ASSIGN card. Use Error Restart H.

INFEASIBLE VARIABLE xxxxxx 1 DUAL

Minor error/major error. Variable xxxxxx is infeasible and cannot be removed from the basis. This is a minor error the first time it occurs; system will automatically enter CHECK. and then INVERT to see whether this was caused by digital difficulties. If the situation recurs in 1 DUAL, it is a major error and the system halts because of infeasible problem formulation. Use Error Restart A.

INPUT CNTRL CARD NAME DOES NOT MATCH PRB CTL FILE — INPUT.

Major error. The INPUT. agendum card problem name does not match the specified control file name. The agendum card is mispunched. Use Error Restart A.

INPUT ROW FILE GREATER THAN MATRIX ROW FILE — MTXDSK

Major error. The number of rows in the ROW.ID file for the problem is greater than the number of rows in the ROW.ID file of the matrix. If it is desirable to add rows to the matrix, use Error Restart C or D. If adding rows to the matrix is not desired, use Error Restart A.

INVERSE FILE ENTERS DIM. — 1DUAL

Major error. The inverse file is about to exceed the disk storage area allowed. If the program had not halted, the Monitor DIM table required by the system would have been destroyed.

1. Preferred method. Prepare an ASSIGN card with
   a. Sector address -- DIM
   b. Sector address -- UL
   c. Common computation area sector address specifying an area above the DIM table large enough for the inverse file.

   Use Error Restart H.
2. Specify a lower starting sector address on the INPUT card. Use Error Restart A. Depending upon the size of the problem, this method may or may not yield a successful run.

INVERSE FILE EXCEEDS ULPRB. — 1DUAL

Major error. The upper limit ASSIGNed by the user has been exceeded.

1. Preferred method. Repunch the ASSIGN card with a higher upper limit or a lower common computation area sector address. Use Error Restart H.
2. Specify a lower starting sector address on the INPUT card. Use Error Restart A. Depending upon the size of the problem, this method may or may not yield a successful run.

LP 1621 AUTO ASSIGN NEW MANSA IS XX — LP 1621

Information message; no halt occurs. The problem mantissa is now xx.

LP 1621 NEW OBJ FCT — LP1621

Information message; no halt occurs. Optimization subroutine to change objective function row.

LP 1621 TO RRRRRR — LP 1621

Information message; no halt occurs. The next routine in the optimization sequence is RRRRRR.
RRRRRR = DUAL for iterations.
RRRRRR = PRIMAL for iterations.
RRRRRR = CHECK. to check accuracy.
RRRRRR = INVERT to (re-)invert basis.
RRRRRR = NEWRHS to change R-H-S.
RRRRRR = INVERT CLEANUP when the basis to the INVERT program is singular.

MAJ. ERROR. SW 4 ON TO CONTINUE OR OFF TO ENDJOB — LP 1621

Information message; program halts. Major error has occurred. See preceding logical error messages to determine the cause of the major error. Consult message list to clarify the cause of the error and to determine the corrective action required.

MAP REFERENCE NAME DOES NOT AGREE WITH CONTROL FILE REFERENCE — LP 1621

Major error. The MAP... agenda card problem name does not match the specified control file name. The agendum card is mispunched. Use Error Restart A.

MATRIXD AND PRB CTL FILE NAMES DO NOT MATCH — MTXDSK

Major error. The name specified in columns 13-18 of the MATRIXD indicator card does not match the name in the sector given by columns 8-12 of the MATRIXD card. If the indicator card is mispunched, use Error Restart A. If the indicator card is not mispunched, the referenced file has probably been destroyed. A matrix must be entered on cards. Use Error Restart A.

MINOR ERROR WITH SW 3 OFF MAJ. ERROR. SW 4 ON TO CONTINUE OR OFF TO ENDJOB — LP 1621

Information message; program halts. A minor error has occurred. This message is suppressed and no halt occurs if console switch 3 is ON. See preceding logged error messages to determine cause of error. Consult message list to clarify the cause of the error and to determine the corrective action required.

MOD ERR $\bar{x}$xxxx — MONITOR

Monitor error message; machine halts. Invalid disk address due to failure to define, by a Monitor *DFINE card, the number of disk storage drives to MONITOR, or invalid ++JOB card, or invalid sector addresses on drive 1, 2, or 3.

NO AREA SET UP FOR SAVE.p — SAVE.p

Major error. No area has been set aside for the basis file. Agendum cards are out of order, SAVE.p precedes optimization. Use Error Restart H.

NO xxxxxx FILE — REVISE

Major error. No xxxxxx file exists but revise control card xxxxxx has been read. The data cards are bypassed. The file may be BASIS., COL.ID, FIRST.B (R-H-S), MATRIX, ROW.ID. This is an attempt to revise a file which has not been INPUT or SAVEd. If the file is not required, use Error Restart F. If the file is required, use Error Restart C or D.

NO DATA EXISTS FOR MIN, MAX, OR INVERT — LP 1621

Major error. A file or files required for processing have not been INPUT. Agenda deck or input deck out of order or inadequate. Use Error Restart A.

NO DATA FILES - CANNOT OUTPUT — OUTPUT

Major error. Data files are nonexistent or invalid. The agendum cards are probably out of order. Use Error Restart A.

NO IND CARD — BASIS.
INPUT A

Major error. The card which follows the INPUT. agendum or the BASIS. data indicator card is not a data indicator card. The data cards are bypassed until an indicator card is read. The INPUT. agendum must be followed by either a ROW.ID or a COL.ID data indicator card. The BASIS. data indicator card must be followed by a VARBLS data indicator card. Use Error Restart A.

NO INPUT CONTROL CARD — INPUT A

Major error. The indicator card read (cols 1-3 not blank, col 1 not *) is not an INPUT. data indicator card. The card in question is punched out. Error is due to mispunched indicator card or missing ENDATA or EOF card. Use Error Restart A.

NO INVERSE EXISTS. ONLY VARBLS AND SLKS SAVED — SAVE.p

Informative message; no halt occurs. The agendum card specified SAVE.I (save inverse), but no inverse file exists. The BASIS VARIABLE and BASIS SLACK files are formed and saved.

NO MATRIX -- CANNOT ACCEPT BASIS — INPUTA

Major error. The BASIS. indicator card has been read, but the MATRIX file has not been read. The data cards are bypassed. The input deck does not contain a MATRIX, or the MATRIX follows the BASIS. Use Error Restart A.

NO MATRIX -- CANNOT FORM B FILE — INPUT A

Major error. The FIRST.B indicator card has been read, but the MATRIX file has not been read. The data cards are bypassed. The input deck does not contain a MATRIX, or the MATRIX follows the BASIS. Use Error Restart A.

NO MATRIX FILE TO REV B FILE — REVISE

Major error. The MATRIX file does not exist. A B or R-H-S file is valid only if a MATRIX file exists. The data cards are bypassed. The data file for this problem is invalid and must be corrected and re-INPUT. before use. Use Error Restart C or D. May also be due to modification or replacement of input programs.

NO OBJ. FUNCTION ROW IN FILE — INVERT

Major error. There is no row classified as an objective function in the ROW.ID file. The objective function row identification card is either missing or mispunched. If it is missing, use Error Restart A. If it is misclassified, use Error Restart A or E.

NO REV IND CARD — REVISE

Major error. The card(s) which follows the REVISE agendum card is not a data indicator card (or a comments card * in col. 1). These cards are bypassed until an indicator card is read. A data indicator card must follow the REVISE agendum card. Cards are missing or out of order. Use Error Restart B.

NO ROOM FOR PROB. ON DISK — BASIS., COL.ID, FIRSTB, INPUT., INPUTB, MATRIX, MTXDSK, ROW.ID, SAVE.p

Major error. The files created by the program at the sector address specified in the INPUT. or SAVE.p agendum card (col. 8-12) exceed the upper limit for the problem. An ASSIGN card can be used to modify this limit. A mispunched sector address in the agendum card or a missing ASSIGN card causes this error. Use Error Restart H.

NO ROW FILE-CANNOT FORM MATRIX — INPUT A

Major error. The MATRIX data indicator is read before formation of the ROW.ID file. The ROW.ID file is missing or does not precede the MATRIX. Use Error Restart A.

NO ROW FILE TO REV MATRIX — REVISE.

Major error. The current problem file does not contain a ROW.ID file. The revision is being made to the wrong file (incorrect INPUT agenda), the files have been destroyed by subsequent disk usage, or by trying to add a MATRIX to a file.

NO SLACKS IND CARD — BASIS., REVBAS

Major error. The BASIS. deck does not have a SLACKS indicator card. The SLACKS indicator card is misplaced or missing. Use Error Restart A if the error occurred during input of a BASIS. Use Error Restart B if the error occurred during REVision of a BASis.

NO SPACE AVAILABLE ABOVE DIM — LP1620

Informative message only; not an error. The area above the DIMension table (cylinder 24) to the monitor program area (cylinder 80) is Monitor-protected. This area contains programs and data under Monitor control and must not be used for LP1620 data storage or computation. The only area available for LP data use is sector 01805 to sector 04799, if the system has one IBM 1311 disk device.

NO UPPER BOUND FOR TYPE G — CHECK.

Major error. A variable is classified as a type G variable in the AJ VAR (MATRIX) file. This variable has no upper bound. Error is probably due to file destruction or misuse by an optimization or output program: 1DUAL, INVERT, CHECK., OUTPUT, DO.D/J, or COST.R. It may be due to user program modification or replacement.

NO VARBLS IND CARD — BASIS., REVBAS

Major error. The BASIS. deck does not have a VARBLS indicator card. The VARBLS data indicator card is misplaced or missing. Use Error Restart A if the error occurred during input of a basis. Use Error Restart B if the error occurred during revision of a basis.

NO VARBLS OR SLACKS CARD BASIS.

Major error. The first data indicator card following the BASIS. card in the BASIS deck is not VARBLS and is not SLACKS. The input deck is out of order or indicator cards are missing. Use Error Restart A.

NO VBLS LEFT IN BETA 1 CHECK.

Major error. Program error in CHECK. program. Error is probably due to user modification.

NOT A REVISE CONTROL CARD REVISE

Major error. The indicator card (column 1-3 not blank and column 1 not *) is not a data indicator card: ROW.ID, COL.ID, BASIS., MATRIX, FIRST.B, EOF, or ENDATA. The indicator card is punched out. The REVISE deck is out of order or cards are missing. Use Error Restart B.

NOT ENOUGH DISK FOR BETA LP1621

Major error. The disk area remaining is inadequate to allow room for a BETA file. This area is part of the compute area. The starting sector address for the compute area can be specified on an ASSIGN card. Prepare an ASSIGN card and use Error Restart H.

NOT ENOUGH ROOM FOR SAVED VBL AND SLACK NAMES NXTDSK = xxxxx LP1621

Major error. The disk area remaining is inadequate to allow room for a BASIS file. The BASIS area is space for a BASIS SLACK file and a BASIS VARIABLE file. This area is part of the input area. It is set aside prior to optimization (MIN or MAX). The ASSIGN card (see "1620-1311 LP Agenda") can be used to increase the available area. A lower sector address on the INPUT agendum card, if possible, will increase the available disk area. Use Error Restart A if the INPUT agendum is changed. Use Error Restart H if the ASSIGN agendum is used.

OBJ. FUNCTION OR OMIT ROW HAS CLASS G IN SLACK FILE INVERT

Major error. The BASIS SLACK file classifies an objective function or an omit row as a class G variable. The ROW.ID file or the BASIS SLACK file should be revised to correct the discrepancy. Use Error Restart E or G respectively.

OPTIMUM 1 DUAL

Information message only. The current basis is optimum.

OUTPUT AGDM CARD NAME DOES NOT MATCH PRB CTL FILE OUTPUT

Major error. The name specified on columns 13-18 of the OUTPUT agendum does not match the name in sector given by columns 8-12 of the OUTPUT agendum. If the name or sector address is mispunched, use Error Restart H. If the name and sector address are correct, the referenced file has probably been destroyed. Prepare an OUTPUT agendum card with blanks in column 7-18. Use Error Restart H and manually select output. Examine INPUT and SAVE agenda and use MAP. . . agenda to determine which agenda caused destruction of the referenced file.

PROB. NAME ON CHECK CARD DOES NOT MATCH CTL. FILE — CHECK.

Major error. Same as preceding except that agenda is CHECK.

PROBLEM COMMON CONFLICTS WITH DISK LIMITS — INPUT.

Major error. The sector address specified in columns 8-12 of the INPUT agendum card is not in the available disk area. An ASSIGN card should precede the INPUT card, making available the area specified by the sector address. Use Error Restart A.

PROC. ERROR EXCEEDS TOL. FOR VARIABLE ****** — 1DUAL

Information message only. Processing error for variable ****** selected to enter the basis in this iteration exceeds the error tolerance. The system will increase the mantissa length (if possible) and change tolerances, INVERT, and continue optimization.

RECORD MARK ON AGENDUM CARD UNINTERPRETABLE AGENDUM CARD — LP 1621

Major error. The agendum card has record marks in columns 1-6. Use Error Restart H.

ROW.ID FILE NOT DEFINED — NEWRHS

Major error. The ROW.ID file does not exist. The agendum cards are out of order or a program error resulted in file destruction. Use Error Restart H if the agendum cards are out of order.

RHS FILE NOT DEFINED — CHECK., NEWRHS

Major error. The R-H-S file does not exist. The agendum cards are out of order or a program error resulted in file destruction. Use Error Restart H if the agendum cards are out of order.

RHS NAME NOT FOUND IN FILE — INVERT

Major error. The RHS name specified in columns 8-12 of the INVERT, MIN, or MAX agendum card is not the name of a R-H-S in the RHS file. Correct the agendum card and use Error Restart H.

ROW FILE OVER LIMIT — ROW.ID

Minor error. The number of rows in the INPUT ROW.ID file exceeds machine capacity. The excess rows are bypassed. Use Error Restart A.

ROW = ≠ — INVERT, NEWRHS

Major error. There are two entries in the right-hand side for a row defined as an equation in the ROW.ID file. Use Error Restart E to either change the row type or to delete one (or both) entries in the right-hand side.

ROW NOT IN FILE — FIRSTB, MATRIX, REVROW

Minor error. See error message ****** ROW NOT IN FILE. Probably a blank or mispunched card in the R-H-S or MATRIX element cards. Use Error Restart A.

S*****CONTROL FILE NOT YYYYYY — COST.R

Major error. The name on the COST.R agendum card does not match the specified control file name. The agendum card is mispunched. Either the problem name and/or the control file location are incorrect. Use Error Restart H.

SLACK HAS INVALID CLASS — INVERT

Major error. A basis identification card contains an invalid slack class. Turn console switches 2 and 4 on, remove all cards from read hopper, place GETOFF agendum card in the read hopper, press START, turn console switches 2 and 4 off, list GETOFF output cards, correct invalid slack class(es). Use Error Restart G.

THIS IS NOT A COST ROW — INVERT

Major error. The objective function name xxxxxx, specified on columns 13-18 of the INVERT, MAX, or MIN card, is not a row classified as an objective function in the ROW.ID file. If the row type is incorrect in the ROW.ID file, use Error Restart A or E. If the row name is incorrect in the INVERT, MAX, or MIN agendum card, use Error Restart H.

THIS ROW NOT FOUND IN FILE — INVERT

Major error. The objective function name xxxxx, specified on columns 13-18 of the INVERT, MAX, or MIN card, is not in the ROW.ID file. The agendum card is probably mispunched. Use Error Restart H.

TRIED TO FLOAT INVALID NUMBER — FIRSTB, MATRIX, REVFST, REVMAT

Minor error. The element card contains an invalid data field. The data is bypassed and the erroneous card is punched out. The card is probably mispunched. Use Error Restart A if the error occurred in data INPUT. Use Error Restart B if the error occurred in data REVISION.

TWO CONSECUTIVE FAILURES IN OPTIMIZING — LP1621

Major error. 1 DUAL has found that the problem is infeasible or unbounded; or that the mantissa and tolerances are inadequate. The problem status (infeasible or unbounded) and the name of the (infeasible or unbounded) variable are given by the preceding error message: INFEASIBLE VARIABLE xxxxxx or UNBOUNDED VARIABLE xxxxxx. If the problem is incorrectly formulated, use Error Restart A or, if possible, E. If the problem is correctly formulated, turn console switches 2 and 4 on, remove cards from read hopper, place a GETOFF agendum card in the read hopper, press START, turn console switches 2 and 4 off, list GETOFF output cards, adjust mantissa and/or tolerances by an ASSIGN card preceding the MIN or MAX agendum, and use Error Restart H. (See ASSIGN card format shown earlier.) The tolerances which may required adjustment are: element tolerance (set this to 9 unless the BASIS. card in the GETOFF output is larger than 9); pivot tolerance (increase this by 1 or 2, that is, from 4 to 5 or 6); the objective function tolerance -- if the solution is unbounded only -- (decrease this by 1 or 2, that is, from 6 to 5 or 4). Increase the mantissa to 18 (or

less). If the element tolerance is changed and the mantissa is not changed, the ERASE I (erase the inverse) agendum should precede the MIN or MAX agendum card.

TYPE F YYYYY NOT IN BETA 1 — CHECK. NEWRHS

Major error. YYYYY = SLACK - AJ SLACK file. YYYYY = VBL. - AJ VAR (MATRIX) file. A variable is classified as a type F variable in the AJ VAR or AJ SLACK file. This variable does not appear in the solution list BETA 1. Error is probably due to file destruction or misuse by an optimization or output program: 1DUAL, INVERT, CHECK., OUTPUT, DO.D/J, or COST.R. It may be due to user program modification or replacement.

UNBOUNDED VARIABLE xxxxxx — 1 DUAL

Information message/major error. Variable xxxxxx has been chosen to enter the basis but has been found to be unbounded. The system will CHECK. the current basis for accuracy, increase the mantissa and tolerances if accuracy (max and min row error above maximum error tolerance) is inadequate, and re-INVERT the current basis to minimize possibility of digital difficulty. If the variable remains unbounded, the message repeats, this time as a major error. See TWO CONSECUTIVE FAILURES IN OPTIMIZING message for correction and restart procedure.

UNKNOWN CODE, A VBL FILE — INVERT

Major error. A basis identification card for a structural variable contains an invalid class. Turn console switches 2 and 4 on, remove all cards from read hopper, place GETOFF agendum card in the read hopper, press START, turn console switches 2 and 4 off, list GETOFF output cards, correct invalid class(es), use Error Restart G.

UPPER LIMIT DIM = $\bar{N}$NNNN — LP1620

Information message. The last Monitor-protected sector on disk drive 0 is NNNNN. Sectors 04800 to NNNNN are Monitor-reserved for DIM, EQUIVALENCE tables and programs. The area from NNNNN to 15999 may be used for LP data if an ASSIGN specifies that the area can be used. This ASSIGN card must precede the use (INPUT., SAVE) of the area above NNNNN.

DISK STORAGE UTILIZATION

AREAS UNAVAILABLE ON MONITOR DISK

The following areas on the Monitor disk are not available for LP files:

Sectors	00000-01804	--	Temporary Storage, Common File, LP System
"	04800-04999	--	DIM Table
"	05000-05199	--	Equivalence Table
"	05200-NNNNN	--	Linear Programming and User Programs. The sector address NNNNN is given by the Message UPPER LIMIT DIM = $\bar{N}$NNNN
"	16000-19999	--	Monitor

Sectors 00000-04799 are used by Monitor as working storage; if the Monitor disk is being used for other jobs as well as LP runs, any files to be saved must be placed above the DIM table, Equivalence table, and programs. At the beginning of each LP run, a message is typed out giving the sector address of the first available sector above the program area.

If the Monitor disk is being used only for LP runs, any files to be saved may be stored in the following sectors:

sectors 01805 through 04799
first sector beyond programs through 15999

On multidrive systems, the second and succeeding disks may be used in their entirety for saving LP files.

PROTECTION OF DIM TABLE

When files are created, they are placed on the disk beginning at the sector address specified on the INPUT. agendum card. If they require too much area to fit below the DIM table, Equivalence table, and programs, they will be continued beyond the program area, so that the DIM table, etc., will be protected. This is true of all files except the Inverse File, where this technique is not possible. If the Inverse File is started below the DIM table and does not fit in that area, a message is typed out to that effect and a major error halt occurs.

With a large problem, the Inverse File will utilize by far the greatest amount of disk storage; thus it is to be expected that the Inverse File will not always fit below the DIM table. As stated above, the files will be placed on the disk beginning at the sector address specified on the INPUT. agendum card; the ASSIGN agendum card may be used to place the Inverse and Beta Files on a different disk area. Thus, INPUT. may specify a sector address low enough to allow all other files to reside on the disk below the DIM table, while ASSIGN is used to place the Inverse and Beta Files beyond the program area. This technique may also be used with a multidrive system, where the Inverse and Beta Files are placed on the second disk, while the other files are put on the first disk. (An example of this usage is included in the paragraph on additional disk drives, later in this section.)

LP DISK FILES

The number and size of the generated files depends on the input data. The size of the Inverse and Beta Files (see below) depends also on the mantissa length.

The files generated from the input data or formed in processing are:

1. Program Control File, containing location, status, and current processing information.
2. Identification Files
   a. ROW.ID file, formed from card input
   b. COL.ID file, formed from card input or generated from the Matrix file
   c. Basis slack file, formed from card input or SAVEd from Matrix (A_j Variable) file
   d. Basis Variable file, formed from card input or SAVEd from A_j Slack file
3. Data Files
   a. A_j Variable (Matrix) file, formed from card input
   b. A_j Slack file, generated from ROW.ID file
   c. B (R-H-S) file, generated from card input.

4. Solution Level File: Beta file -- solution levels and bounds of type F (intermediate-level) basis variables.

5. Transformation Files

   a. Eta 11 Logical basis transformation. A single-digit-per-row coding to indicate the type of logical variable formed from the ROW.ID file.

   b. Inverse file -- iteration transformation vectors, formed after inversion and by iteration process

   c. Alpha file -- generated during inversion procedure. Partially transformed basis vectors

LP FILE LOCATION AND SIZE

The Problem Control File begins at the sector specified by the INPUT. or SAVE.p agendum. This file is five sectors in size. The location of all other files is given by the Problem Control File. The applicable fields are all in the first two sectors as follows:

Sector	Positions in sector	Length	Description
1	01-12	12	Problem name
	15-23	9	Disk drive, sector address, amount of sectors for ROW.ID File
	29-37	9	Disk drive, sector address, amount of sectors for ETA11 File
	43-51	9	Disk drive, sector address, amount of sectors for COL.ID File
	57-65	9	Disk drive, sector address, amount of sectors for MATRIX File
	86-90	5	First available sector above DIM table, Equivalence table, and programs
	91-95	5	Upper limit of user's area on disk
2	01-09	9	Disk drive, sector address, amount of sectors for Right-Hand-Side File
	15-23	9	Disk drive, sector address, amount of sectors for Basis Variable File
	29-37	9	Disk drive, sector address, amount of sectors for Basis Slack File
2	43-51	9	Disk drive, sector address, amount of sectors for Inverse File
	57-65	9	Disk drive, sector address, amount of sectors for Beta File

Sector	Positions in sector	Length	Description
	71-75	5	Sector address of first A_j group record
	76-80	5	Sector address of first sector beyond last A_j group record
	81-85	5	Sector address of first A slack group record
	86-90	5	Sector address of first sector beyond last A group record
	91-95	5	Sector address of first Right-Hand-Side group record
	96-100	5	Sector address of first sector beyond last Right-Hand-Side group record

EXAMPLE OF LP FILE ARRANGEMENT

A typical input deck is shown below, with the files which would be created for the problem. Note that the files start at the sector address specified on the INPUT. agendum card.

Card Deck

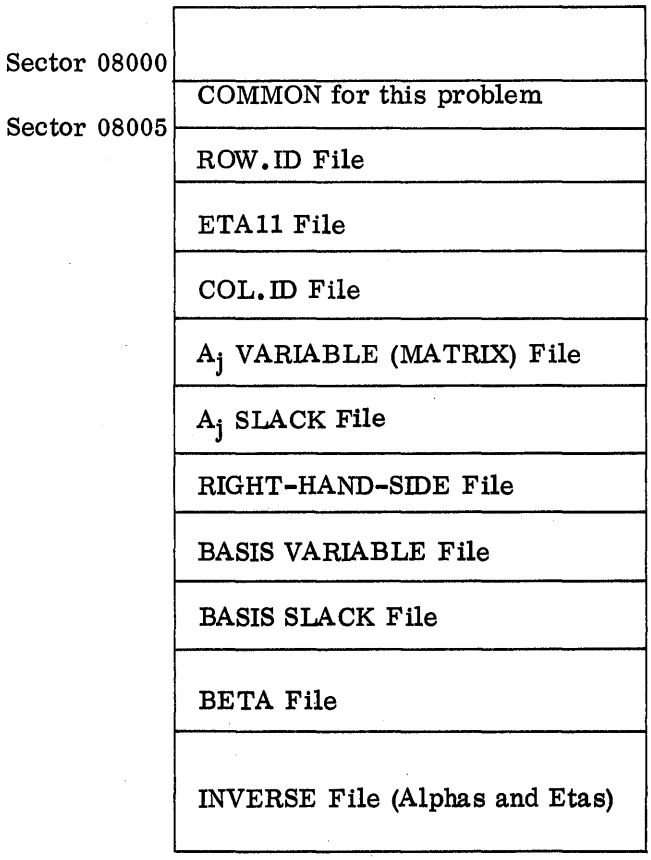
```
INPUT.C08000

ROW.ID
   :
COL.ID
   :
MATRIX
   :
FIRST.B
   :
NEXT.B
   :
BASIS
   :
ENDATA
```

If no COL.ID data is input, a COL.ID File is created internally and put:

1. directly after the RHS File if no BASIS cards are input, or
2. directly after the BASIS File if BASIS cards are input.

Disk Storage



FILE SIZE

With the exception of COMMON, all files vary in size according to the problem size. To find the number of sectors used for a file, take the results of the formula and use the next higher integer (for example, 2024/100 = 21 sectors).

COMMON	5 sectors	
ROW.ID File	$\frac{R \times 14}{100}$	R = number of rows
ETA11 File	$\frac{R}{100}$	R = number of rows
COL.ID File	$\frac{C \times 14}{100}$	C = number of columns

A_j VARIABLE (MATRIX) FILE AND THE A_j SLACK FILE

Each of these is composed of group records. A group record contains as many variables as possible within the maximum group record size to reduce disk usage. The maximum size of a group record depends on machine size.

	max. size of a group record:	20K -- 20 sectors 40K -- 50 sectors 60K -- 100 sectors
MATRIX File A_j file	$\frac{15 + (57C + 15N)}{100}$	C = number of columns (variables) N = average number of nonzero entries per column, including bounds
MATRIX File A_j Slack file	$\frac{15 + 72R}{100}$	R = number of rows in ROW.ID File

RIGHT-HAND-SIDE FILE

The RHS File also consists of group records, whose maximum size is shown below. However, when more than one group record is needed, one right-hand side is never split; that is, if a right-hand side will not all fit in a group record, it is put into the next group record.

	max. size of a group record:	20K -- 31 sectors 40K -- 80 sectors 60K -- 155 sectors
RHS group record	$\frac{15 + (15H + 30N)}{100}$	H = number of right-hand sides in this group record N = number of nonzero row entries in this group record

BASIS Variables File	$\frac{V \times 14}{100}$	V = number of variables
BASIS Slack File	$\frac{S \times 14}{100}$	S = number of slacks
BETA File	$\frac{R \times 44}{100}$	R = number of rows

INVERSE File maximum size

$$[(24 + (M + 2) * R)/100]_{\mu} * R + \max (I, G * [F/E]_{\mu})$$

where:

$$E = [(G * 100 - 20)/((M + 5) * R + 11)]_{d}$$

$[\ ]_{\mu}$ is the smallest integer $\geq [\ ]$

$[\ ]_{d}$ is the largest integer $\leq [\ ]$

R = number of rows
M = problem mantissa length
F = inversion frequency
E = min. number of Eta's per group record
G = ETA group record size
I and G vary according to machine size:

size	I	G
20K	32	24
40K	68	50
60K	104	100

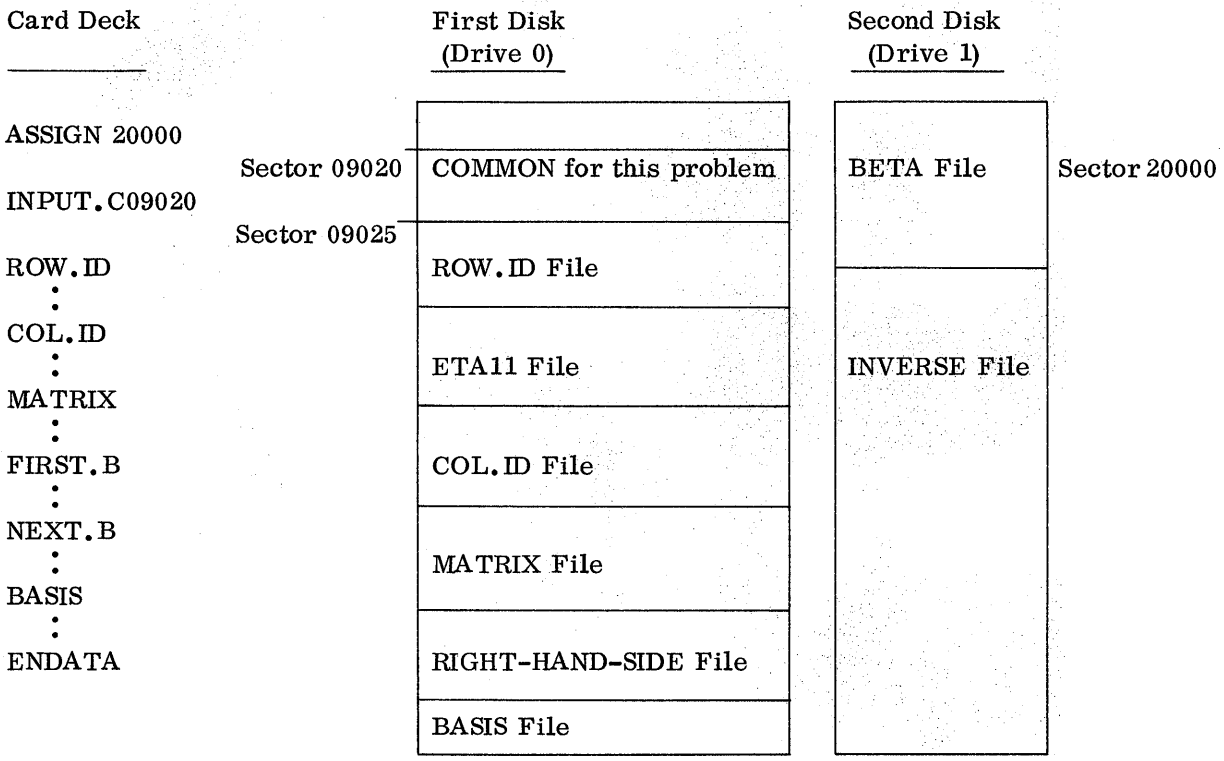
ADDITIONAL DISK DRIVES

If more than one disk drive is used, the Monitor would be on the first drive (drive 0); the second (drive 1) and succeeding drives could then be used in their entirety for LP problems. With two disk drives, three different arrangements are possible for an LP problem:

- Problem files entirely on first disk
- Problem files entirely on second disk
- Problem files on both disks

To have the problem files entirely on one disk, use the appropriate sector address on the INPUT. agendum card. Thus, to begin the problem files on the first sector of the second disk, specify sector address 20000.

It is possible to have the files for one problem on both disks; this is recommended when large problems are run. The method for accomplishing this is to use the second disk for the BETA and Inverse Files, and the first disk for all others. This is done by using a sector address of the first disk on the INPUT. agendum card, and then a sector address of the second disk on an ASSIGN agendum card (columns 33-37). For example:



CORE STORAGE UTILIZATION

Because each routine in the 1620-1311 LP System is loaded and executed individually and works with different files, there is no standard core storage arrangement. Communication from routine to routine is accomplished by the use of COMMON, which is stored in sectors 01800-01804 on the disk and is called in by each routine.

Monitor makes use of the memory area below 02402; above that area are located program and files, and floating-point subroutines in those programs which require them.

Not only do routines differ from one another in the utilization of memory, but the organization in the memory of a specific routine may vary from one machine size to another in order to accommodate the files which are increased in size when core storage is increased.

ERROR AND INTERRUPT RESTART PROCEDURES

Procedures A through I describe input data and formulation restart procedures. These procedures are referenced by error message action. In case of multiple input errors, the highest-priority restart procedure (lowest number) should be used.

Procedure J is the standard GETOFF interruption resumption.

Procedure K is the standard malfunction restart procedure.

ERROR RESTART PROCEDURES

A. Correction of INPUT data. Priority 1.
   1. Correct card(s).
   2. Put corrected deck in card reader followed by interrupted agenda.
   3. Turn on console switch 4.
   4. Push START.
   5. Turn off console switch 4.

B. Correction of REVISE data. Priority 4.
   1. Correct card(s).
   2. Put corrected deck in card reader.
   3. Put remaining agenda in card reader behind corrected deck.
   4. Turn on console switch 4.
   5. Push START.
   6. Turn off console switch 4.

C. Additions to INPUT data, method 1. Priority 2.
   1. Correct original INPUT data and identification cards.
   2. Insert additional entries, identifications.
   3. Put corrected deck in card reader.
   4. Put remaining agenda in card reader.
   5. Turn on console switch 4.
   6. Push START.
   7. Turn off console switch 4.

D. Additions to INPUT data, method 2. Priority 2.
   1. Insert additional entries, identifications in original input data deck.
   2. Consolidate all revisions:
      a. Order the revise decks from oldest (first) to newest (last).
      b. Remove all REVISE agendum cards except the first one.
      c. Remove all ENDATA and EOF cards except the last one.
   3. Put the INPUT deck, including INPUT. agendum, in the card reader.
   4. Put the consolidated REVISE deck in the card reader.
   5. Put interrupted agenda remaining in card reader.
   6. Turn off console switch 4.
   7. Push START.
   8. Turn on console switch 4.

E. Corrections to input by a REVISE deck. Priority 3.
   1. Prepare a REVISE deck.
      a. REVISE agendum card.
      b. Data indicator card, ROW.ID, COL.ID, MATRIX or FIRST.B as required.
      c. Corrected identification card or element card.
      d. EOF card.
   2. Place REVISE deck in card reader followed by remaining agenda.
   3. Turn on console switch 4.
   4. Push START.
   5. Turn off console switch 4.

F. Continuation if error does not result in incorrect disk storage of data. Priority 6.
   1. Turn on console switch 4.
   2. Push START.
   3. Turn off console switch 4.

G. Corrections to input basis by a REVISE deck. Priority 3.
   1. Prepare a REVISE deck.
      a. REVISE agendum card.
      b. BASIS. indicator card.
      c. VARBLS indicator card.
      d. Corrected basis identification card if the variable is a structural (column) variable; if it is a logical (slack) variable, no card.
      e. SLACKS indicator card.
      f. Corrected basis identification card if the variable is a logical (slack) variable; if it is a structural (column) variable, no card.
      g. EOF card.
   2. Place REVISE deck in card reader followed by remaining agenda.
   3. Turn on console switch 4.
   4. Push START.
   5. Turn off console switch 4.

H. Corrections to agenda deck. Priority 1.
   1. Order and correct the agenda cards.
   2. Place remainder of agenda deck in card reader.
   3. Turn console switch 4 on.
   4. Press START.
   5. Turn console switch 4 off.

I. Inverse destruction. Priority 5.
   1. Prepare a restart agenda.
      a. Monitor cards; cold start ≠ ≠ JOB, ≠ ≠ XEQSLP1620.
      b. INPUT. Dssssssnnnnnn, where sssss is the sector address of the input control file and nnnnnn is the problem name.
      c. ERASEI to erase the inverse.
      d. Followed by the rest of the agenda.
   2. Place restart agenda in the card reader.
   3. Press RESET.
   4. Press LOAD.

J. GETOFF restart procedure after standard interruption. When the 1620-1311 LP interrupt procedure is used (console switch 2 ON), GETOFF punches out the current basis. To restart from the point of interruption:
   1. Prepare a new problem deck.
      a. Remove the original basis, if any.
      b. Insert the new basis just before the ENDATA or EOF card.
   2. Put the complete deck in the card reader.
   3. Press LOAD.

K. Standard Machine Malfunction restart.
   1. Prepare a restart deck.
      a. LP1620 RESTORE card.

Column		
1 - 12	34	00032 00701
13 - 24	36	00032 00702
25 - 32	49	02402 0
33 - 46	1	00100 100 00100

      b. Current agendum card followed by remaining data and agenda.
   2. Press RESET.
   3. Put the restart deck in the card reader.
   4. Press LOAD.

SAMPLE TIMINGS

Problem ID	m	n	Optimality		Machine Model and Size		Comments
			Iter	Time			
SHARE 20	17	25	21	8 min.	1620-1	40K	
SHARE 15	66	72	58	52 min.	1620-1	40K	
SHARE 19	78	87	131	1 hr., 10 min.	1620-2	60K	
SHARE 1	20	31	15	3 min.	1620-2	60K	
SHARE 22	54	44	58	36 min.	1620-2	60K	
SHARE 16	49	62	18	1 hr., 10 min.	1620-1	40K	
SHARE 7	36	78		1 hr., 44 min.	1620-2	60K	Several options and solutions
Sample Problem	9	7		38 min.	1620-1	40K	For entire sequence

MATHEMATICAL DESCRIPTION OF 1620-1311 LP

This section describes the formulae and their application to the routines. The notation used is the program notation or symbology, wherever possible. For proofs, etc., the reader should consult the mathematical references (5, 10, and 11) in the Bibliography. This section assumes a knowledge of vector algebra.

INPUT.

The input routine forms the structural and logical matrix files and classifies the variables, as specified by the user. The disk output of this routine is:

$$aY = b, \quad Y \leq \overline{b}, \quad Y \geq \underline{b}$$

where:

a is the matrix of coefficients of the input structural variables and the generated logical variables.

Y is the vector of variables.

b is the right-hand-side vector (in force at optimization).

$\overline{b}$ are the upper bounds of Y.

$\underline{b}$ are the lower bounds of Y.

Note: The subscript-superscript notation will be used throughout this section.

For example:

a_j : column j of matrix a.

b^i : element (row) i of right-hand-side b.

Y^j : variable j of the (column vector of) variables Y.

a^i_j : coefficient of i^{th} row, j^{th} column.

$\overline{b}^j$, $\underline{b}^j$: elements j of upper, lower bound vectors: $\underline{b}^j \le Y^j \le \overline{b}^j$

Dimensions. - The number of rows is M, the number of structural variables is N, the total number of variables is M + N.

Tolerance and mantissa assignment. - The input routine assigns initial mantissa and tolerances according to the following formulae, unless specified by a BASIS. input control card or an ASSIGN card. All tolerances are expressed as powers of 10: 10^{ϵ}. Define N.D.R.a^i_j as the number of digits to the right of the decimal point of input data element a^i_j.

$$\epsilon_{\text{pivot}} = -1 - \max \left\{ 3, \text{N.D.R.}a^i_j \right\} \quad \begin{matrix} i = 1, \ldots, M \text{ for nonfunctional rows.} \\ j = 1, \ldots, N \end{matrix}$$

$$\epsilon_{\text{functional}} = -1 - \max \left\{ 3, \text{N.D.R.}a^i_j \right\} \quad \begin{matrix} i = 1, \ldots, M \text{ for functional rows.} \\ j = 1, \ldots, N \end{matrix}$$

$$\epsilon_{\text{feasible}} = -1 - \max \left\{ 3, \text{N.D.R.}b^i_j \right\} \quad \begin{matrix} i = 1, \ldots, M \\ \text{for } j \text{ ranging over all right-hand sides.} \end{matrix}$$

$$\epsilon_{\text{error}} = \epsilon_{\text{feasible}}$$

$$\text{Mantissa} = \min \left\{ 18, \max \left\{ 5, 2 + \text{range} - \min \left\{ \epsilon_{\text{pivot}}, \epsilon_{\text{functional}} \right\} \right\} \right\}$$ where range is the difference between largest and smallest exponent of the input data converted to floating-point form.

$$\epsilon_{\text{element}} = -\text{truncated} \left\{ \text{mantissa}/4 \right\} + \min \left\{ \epsilon_{\text{pivot}}, \epsilon_{\text{functional}} \right\} - 2.$$

Mantissas and tolerances are modified during input by:

Mantissa: add quantity ($\epsilon_{\text{error}} - \epsilon_{\text{error max}}$) if positive. $\epsilon_{\text{error max}}$ is the exponent of the maximum error found in a CHECK.

$$\epsilon_{\text{element}} = \epsilon_{\text{element}} - \text{truncated (mantissa } 1^{st} - \text{mantissa old})/2.$$

$$\epsilon_{\text{pivot}}, \epsilon_{\text{functional}}, \epsilon_{\text{feasible}} = \epsilon_{\text{old}} - \text{truncated (mantissa } 1^{st} - \text{mantissa old})/3.$$

Classification of variables. - This is determined according to basis status and activity level.

W: class of variables at lower bound and not in basis.

F: class of variables in the basis.

G: class of variables at upper bound and not in basis.

*: subclass of nonbasis variables, the omitted rows and columns.

Reference to variables of the various classes is by superscript, W^j, F^j, G^j; and the column vectors by small letters, w_j, f_j, g_j when differentiation is required.

Initialization. - The generated logical variables constitute the initial basis, unless a basis is specified. The logical variables are classified as type W, unless a basis is specified. The omitted rows (class * logical variable in basis) and the omitted columns (class * structural variables) are not used in calculation.

INVERSION

The inversion procedure is similar to the revised simplex tableau procedure. The pivot selection is the major difference.

Notation. A time index is required to clarify certain vectors, matrices, etc., within parentheses.

For example:

F(0): the basis at iteration 0; the logical basis.

Right-hand-side initialization.

β: the transformed right-hand side; the current basis activity levels.

π: the basis inverse.

$\eta(1-1)$: the inverse of the logical basis, f(0).

Note: In the program listings, $\eta(1-1)$ is used to reference $\eta(0)$. For consistency, $\eta(1-1)$ is used in this section.

$[f(0)]^{-1} = \eta(1-1)$

Note: $\eta(1-1) = f(0)$.

The first transformation effects a translation of axes to shift the lower bounds to zero.

Define: $X + \underline{b} = Y$

Then: $a(X + \underline{b}) = b, \qquad \underline{b} \le X + \underline{b} \le \overline{b}$

$aX = b - a\underline{b}, \qquad 0 \le X \le \overline{b} - \underline{b}$

Define: $\overline{a} = \overline{b} - \underline{b}$

$F(0) = \beta(0) = \eta(1-1)\ [b - a\underline{b} - g\overline{g}]$

$F(0)$: Initial logical basis.

$\beta(0)$: Logical basis activity.

g : Matrix of vectors of variables at upper bound.

$\overline{g}$: The transformed ($\overline{a}$) upper bounds of these variables.

Logical variable bounds are formed for all range constraints and equations.

$\overline{S}^i = \overset{\bullet}{b}^i - \underset{\bullet}{b}^i$ For range constraints where:

$\overline{S}^i$ is the upper bound of the logical variable.

$\overset{\bullet}{b}^i$ is the upper bound row limit, $\underset{\bullet}{b}^i$ is the lower.

$\overline{S}^i = 0$ for artificial logical constraints generated for equation rows.

Alpha matrix. - A matrix of the basis (type F) structural variables is formed. This matrix is transformed by pivoting once on each column in the available (vacant logical variable for the row) rows until the matrix is completely transformed. Pivot choice is based on minimizing the number of arithmetic operations required in the transformation.

Initialization. - Transform structural variables by logical basis inverse.

$\alpha_j = \eta(1-1)\ a_j\ ,\quad j = 1, \ldots, \gamma$ where a_j's are the vectors of type F (Basis) structural variables.

Define. - Parameters used to select pivot row.

$r^i + 1$ = Number of nonzero entries in unpivoted α^i. Row count.

$c_j + 1$ = Number of nonzero entries in unpivoted α_j. Column count.

Pivot candidate row: The logical variable for the row is:

Type W; nonbasic at lower bound.

Type G; nonbasic at upper bound.

And row has not been pivoted during inversion.

Pivot possible position: $|\alpha_j^i| \geq 10^{\epsilon_{pivot}}$, α_j not pivoted, i pivot candidate row.

Pivot procedure complete when there are no pivot possible positions.

Normal completion if all α_j pivoted and no pivot candidate rows remain.

Pivot Choice. - Minimize transformation operations.

Choose R = pivot row; s = pivot alpha

$$c_s\, r^R = \min \left\{ c_j \min \left\{ r^i \right\} \right\} \text{ for all pivot positions possible.}$$

Pivot procedure. - Pivot selected column and row. Transform remaining alphas and beta.

Define: $I(R, \alpha_s)$ as an identity matrix (dimension M) except that column R of that matrix is α_s.

Define: $\eta(T) = [I(R, \alpha_s)]^{-1}$

Note: $[\eta(T)]_{k \neq R} = I_k$ where I is an identity matrix.

$$[\eta(T)]_R^{i \neq R} = -\alpha_s^i / \alpha_s^R$$

$$[\eta(T)]_R^R = 1/\alpha_s^R$$

Alpha transformation

$$[\alpha(T)]_i = \eta(T)\,[\alpha(T-1)]_i$$

Note: $$[\alpha(T)]_i^{k \neq R} = [\alpha(T-1)]_i^k + \eta_R^k\,[\alpha(T-1)]_i^R$$

$$[\alpha(T)]_i^{k=R} = \eta_R^R\,[\alpha(T-1)]_i^R$$

Beta transformation

$$\beta(T) = \eta(T)\,\beta(T-1)$$

Error conditions. If row candidates and/or alphas remain, but pivoting cannot continue -- no possible pivot candidates remain -- the remaining alphas, if any, are ignored and logical variable restored for any remaining pivot candidates.

The inverse. The product of the etas (η) would form the inverse. The inverse is maintained in the product form.

$$\pi(T) = \eta(T)\,\eta(T-1) \ldots \eta(1)\,\eta(1-1)$$

Inverse row. - During optimization a row of the inverse is required.

$$\pi^i = I^i \pi = I^i\,\eta(T)\,\eta(T-1) \ldots \eta(1)\,\eta(1-1)$$

Notation: Objective function row noted by ϕ

Note: Only the vector $[\eta(T)]_R$ and the pivot row indication R are stored.

DUAL

Row selection. - Choose R, the basis index or position of the variable to leave the basis such that:

$V^R = \min \left[K^i\,(\beta^i, \bar{f}^i - \beta^i)\right]$ for $i = 1, \ldots, m$ excluding * and objective function artificials.

$K^i = 1$ if $\bar{f}^i \neq 0$, where $\bar{f}^i$ is the upper bound of F^i.

$K^i = 100$ if $\bar{f}^i = 0$.

The solution is said to be feasible if $V^R \geq -10^{\epsilon_{feasible}}$.

The variable F^R leaves the basis at its lower bound if $\beta^i < 0$.

The variable F^R leaves the basis at its upper bound if $\beta^i > 0$.

Define: BETA TEST VALUE = $(1/K^R)/V^R$.

Column selection. - Choose s, the index or position of the variable to enter the basis, such that:

$$Z_s = \min \left\{ D_j/C_j \right\} \text{ for } C_j < -10^{\epsilon_{pivot}} \text{ and } X^j \text{ is not an artificial logical variable (generated for an equation row).}$$

Where:

$$D_j = \left\{ \begin{matrix} +1 \text{ if MAXimizing} \\ -1 \text{ if MINimizing} \end{matrix} \right\} \quad \pi^{\phi} \quad \left\{ \begin{matrix} +1 \text{ for type W variables} \\ -1 \text{ for type G variables} \end{matrix} \right\} a_j$$

This formula is used only for the first iteration following inversion. The objective function row is denoted by ϕ.

And:

$$C_j = \left\{ \begin{matrix} -1 \text{ if } \beta^R > 0 \\ +1 \text{ if } \beta^R < 0 \end{matrix} \right\} \quad \pi^R \quad \left\{ \begin{matrix} +1 \text{ for type W variables} \\ -1 \text{ for type G variables} \end{matrix} \right\} a_j$$

D_j is computed during dual by $D_j' = D_j + Z_s C_j$ except for the first iteration following inversion.

A W variable increases from an explicit lower bound, or from the implicit zero lower bound, as it enters the basis.

A G variable decreases from the explicit upper bound as it enters the basis.

The problem is said to be infeasible if no Z_s exists.

Processing error test

$1 - \left| C_s/\alpha_s^R \right| < 10^{\epsilon_{error}}$ Test failure results in increase of mantissa length, change in tolerance and reinversion.

Bound test

$$\text{New BETA TEST VALUE} = \text{old BETA TEST VALUE} - \left\{ \begin{matrix} -1 \text{ for type W} \\ +1 \text{ for type G} \end{matrix} \right\} \bar{a}^s \cdot \alpha_s^R$$

If the new BETA TEST VALUE is negative, the entering variable replaces the leaving variable; otherwise the variable enters at its bound. If the B.T.V. remains negative, the variable would violate its upper or lower bound if it replaced F^R.

PRIMAL

Column selection. Choose s, the index or position of the variable to enter the basis, such that:

$$D_s = \min \left\{ D_j \right\} \text{ and } D_s < -10^{-\epsilon_{\text{functional}}}$$

Where:

$$D_j = \left\{ \begin{matrix} +1 \text{ if MAXimizing} \\ -1 \text{ if MINimizing} \end{matrix} \right\} \pi^{\varphi} \left\{ \begin{matrix} +1 \text{ for W} \\ -1 \text{ for G} \end{matrix} \right\} a_j$$

The solution or basis is said to be optimum if no D_s exists.

A W variable increases from its lower bound as it enters the basis.

A G variable decreases from its upper bound as it enters the basis.

In a degenerate iteration (no change in objective function), a W variable enters at its lower bound, and a G variable enters at its upper bound.

Row selection. - Choose R, the index or position of the variable to leave the basis, such that:

$$V^R = \min \left\{ \begin{matrix} \beta^i / \alpha_s^i \text{ if } \alpha_s^i > 10^{\epsilon_{\text{pivot}}} \\ \left(\beta^i - \bar{f}^i \right) / \alpha_s^i \text{ if } \alpha_s^i < -10^{\epsilon_{\text{pivot}}} \text{ and } \bar{f}^i \text{ exists} \end{matrix} \right\} \begin{matrix} i = 1, \ldots, m \\ \text{excluding * and} \\ \text{objective function} \\ \text{artificials.} \end{matrix}$$

Indexing stops and row i is selected (i=R) if $\bar{f}^i = 0$ and $|\alpha_s^i| > 10^{\epsilon_{\text{pivot}}}$.

The solution is said to be unbounded if V^R is undefined and the variable entering the basis is type W with no upper bound.

The variable F^R leaves the basis at its lower bound if $\alpha_s^i > 0$

The variable F^R leaves the basis at its upper bound if $\alpha_s^i < 0$

Processing error test

$1 - \left| D_s / \alpha_s^{\varphi} \right| < 10^{\epsilon_{\text{error}}}$. Test failure results in increase of mantissa length, change in tolerance and reinversion.

Bound test

If $\bar{a}^s < V^R$ the variable enters at bound.

If V^R is undefined and $\bar{a}^s$ exists, the variable enters at bound.

UPDATE

a_s UPDATE TO CURRENT BASIS

$$\alpha_s = \pi\, a_s$$

$$= [\eta(T\text{-}1)\ [\eta(T\text{-}2) \ldots [\eta(1)\ [\eta(1\text{-}1)\ [a_s]\ ]\ ]\ \ldots]\ ]$$

Define: $[\alpha(0)]_s = [\eta(1\text{-}1)]\ [a_s]$

Where $\eta(1\text{-}1)$ is the inverse of the logical (starting) basis f(0).

Note: $[f(0)]^{-1} = \eta(1\text{-}1) = [f(0)]$

Define: $[\alpha(i)]_s = [\eta(i)]\ [\alpha(i\text{-}1)]_s$, and R(i) is the pivot row for iteration i.

Then: $$[\alpha(i)]_s^{k\neq R(i)} = [\alpha(i\text{-}1)]_s^k + \eta_{R(i)}^k\ [\alpha(i\text{-}1)]_s^{R(i)}$$

$$[\alpha(i)]_s^{R(i)} = \eta_{R(i)}^k\ [\alpha(i\text{-}1)]_s^{R(i)}$$

and

$$\alpha(T\text{-}1) = \alpha_s$$

$\eta(T)$ Formation. - Required for vector types W and G entering the basis at intermediate level (becoming basic or type F), where T is the iteration index.

$$\eta(T) = [I(R, \alpha_s)]^{-1}$$

where $I(R, \alpha_s)$ is an identity matrix (dimension M) except for the $R^{\underline{th}}$ column which is α_s.

Then: $$[\eta(T)]_{k\neq R} = I_k$$

and

$$[\eta(T)]_R^{i\neq R} = -\alpha_s^i/\alpha_s^R$$

and

$$[\eta(T)]_R^R = 1/\alpha_s^R$$

Only the index, R, and the vector $[\eta(T)]_R$ are stored.

Beta update to current basis

$$\beta(T) = \beta(T\text{-}1) - \phi + \theta + \gamma$$

Where

$\beta(T\text{-}1)$ is the vector of basis activity levels at iteration T-1.

$\beta(T)$ is the vector of basis activity levels at iteration T.

$$\phi = \begin{cases} \text{A zero column vector except for } \phi^R, \text{ i.e., } \phi^{i\neq R} = 0 \\ \phi^R = 0 \text{ if } X^s \text{ enters at upper or lower bound.} \\ \phi^R = \beta(T\text{-}1)^R \text{ if the entering variable enters at intermediate level.} \end{cases}$$

$$\theta \quad \begin{cases} -\bar{a}^s \alpha_s \text{ if the variable is } W_s \text{ entering at upper bound.} \\ +\bar{a}^s \alpha_s \text{ if the variable is } G_s \text{ entering at lower bound.} \\ [\beta(T\text{-}1)]^R \eta(T) \text{ if the variable is } W_s \text{ or } G_s \text{ entering at intermediate level and } F_R \text{ is leaving at lower bound.} \\ [\{\beta(T\text{-}1)\}^R - \bar{f}^R] \eta(T) \text{ if the variable is } W_s \text{ or } G_s \text{ entering at intermediate level and } F^R \text{ is leaving at upper bound.} \end{cases}$$

$$\gamma \quad \begin{cases} \text{A zero column vector except for } \gamma^R, \text{ i.e., } \gamma^{i\neq R} = 0 \\ \gamma^R = \bar{a}^s \text{ if the entering variable is } G_s \text{ entering at intermediate level.} \\ \gamma^R = 0 \text{ otherwise.} \end{cases}$$

NEW RIGHT-HAND SIDE

Initialization. Same as inversion except for alpha formation.

Beta transformation by inverse

$$\beta(T) = \pi\beta(0)$$

$$= \eta(T)\,\eta(T\text{-}1) \ldots \eta(1)\,\beta(0)$$

Same as α_s update to current basis except that the vector is $\beta(0)$ and is not transformed by $\eta(1\text{-}1)$.

CHECK.

Row level. The product-sum of the row coefficients multiplied by lower bounds (all variables), basis activity level (type F structural variables), and upper bounds (type G structural variables).

$$L^i = a^i \underline{b} + f^i \beta^* = g^i \bar{g}^*$$

* structural variables only.

Selective output includes product-sums only on the selected structural variables (for each selected row).

Row error. The difference between input value, adjusted by slack level and row level.

$$E^i = B^i - \begin{cases} +1\text{: for row type 0 and +} \\ -1\text{: for row type blank and -} \end{cases} S^i - L^i$$

For selective output this lumps the row error with the product-sum of the nonselected variables.

B^i is the right-hand-side entry.

S^i is the activity level of the logical variable for row i.

Row limit values. - The current right-hand-side coefficients for the row.

COST.RANGE

High-limit variable selection. Choose H, the index or position of the variable which limits the highest cost of basis variable F^R, such that:

$$Z_H = \max \left\{ D_j / C_j \right\} \text{ for } C_j > 10^{\epsilon_{pivot}} \text{ and } X^j \text{ is not an artificial logical variable.}$$

Where:

$$D_j = \left\{ \begin{matrix} +1 \text{ if MAXimizing} \\ -1 \text{ if MINimizing} \end{matrix} \right\} \pi^{\phi} \left\{ \begin{matrix} +1 \text{ for type W variables} \\ -1 \text{ for type G variables} \end{matrix} \right\} a_j$$

The D_j's are not computed during COST.Ranging, but are assumed to be left over from the final primal iteration.

And:

$$C_j = \pi^R \left\{ \begin{matrix} +1 \text{ for type W variables} \\ -1 \text{ for type G variables} \end{matrix} \right\} a_j$$

If Z_H is undefined, there is no high-limit variable.

Lo-limit variable selection. Choose L, the index or position of the variable which limits the lowest cost of basis variable F^R, such that:

$$Z_L \text{ , } \min \left\{ D_j / C_j \right\} \text{ for } C_j < -10^{\epsilon_{pivot}} \text{ and } X^j \text{ is not an artificial logical variable.}$$

See high-limit variable definition for C_j and D_j. If Z is undefined, there is no lo-limit variable.

Cost-range calculation

$$\text{The highest cost} = \left\{ \begin{matrix} -1 \text{ if MAXimizing} \\ +1 \text{ if MINimizing} \end{matrix} \right\} \left[Z_H + \left\{ \begin{matrix} -1 \text{ if MAXimizing} \\ +1 \text{ if MINimizing} \end{matrix} \right\} a_k^{\phi} \right] \text{ if } Z_H \text{ exists}$$

$$\text{The lowest cost} = \left\{ \begin{matrix} -1 \text{ if MAXimizing} \\ +1 \text{ if MINimizing} \end{matrix} \right\} \left[Z_L + \left\{ \begin{matrix} -1 \text{ if MAXimizing} \\ +1 \text{ if MINimizing} \end{matrix} \right\} a_k^{\phi} \right] \text{ if } Z_L \text{ exists}$$

Where a_k^{ϕ} is the input objective row coefficient.

For undefined Z_H, Z_L "infinity" is placed in the data field.

A minus sign follows this word when the sense is negative.

Current cost. Input (or revised) objective row coefficient.

Reduced cost. Product of inverse objective function row by variable, reversed in sign for variables at upper bound, type G.

$$D_j = \begin{Bmatrix} +1 \text{ if MAXimizing} \\ -1 \text{ if MINimizing} \end{Bmatrix} \pi^{\phi} \begin{Bmatrix} +1 \text{ for type W variable} \\ -1 \text{ for type G variable} \end{Bmatrix} a_j$$

Basis value. Difference of current cost and reduced cost.

$$V_j = a_j^{\phi} - \begin{Bmatrix} +1 \text{ if MINimizing} \\ -1 \text{ if MAXimizing} \end{Bmatrix} D_j$$

where a_j^{ϕ} is the objective row coefficient.

Note: Basis value is output only if there is a current cost entry.

Increase B value, decrease B value: Simplex multipliers or marginal values. Entries only for nonzero simplex multipliers

BIBLIOGRAPHY

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

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

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

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

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

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

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

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

9. An Introduction to Linear Programming, IBM General Information Manual (E20-8171).

10. Wagner, Harvey M., "A Practical Guide to the Dual Theorem", Operations Research, Vol. 6, No. 3, May-June, 1958, pp. 364.

11. Markowitz, H.M., "The Elimination Form of the Inverse and its Application to Linear Programming", Management Science, Vol. 3, No. 3, April 1957, pp. 255-269.

12. IBM 1620 Monitor I System Reference Manual (C26-5739).

Printed in U.S.A. H20-0106-0

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601