Date: June 27, 1975

Project Title: Development of an Interactive (Truly) Conversational Computer Model for Linear Programming

Project No: E-24-642

Principal Investigator: Dr. John J. Jarvis

Sponsor: National Science Foundation

Agreement Period: From June 1, 1975 Until May 31, 1976

Type Agreement: Grant No. H-575-17476

Amount: $17,600 NSF
1,503 GIT
$19,103 Total

Reports Required: (Final (Annual) Technical Report

Sponsor Contact Person(s):

Administrative Matters
Mr. Gaylord L. Ellis
Grants Officer
National Science Foundation
Washington, D.C. 20550
(202) 632-5965

Assigned to: Industrial & Systems Engineering

Copies To:
Principal Investigator
School Director
Dean of the College
Director, Research Administration
Director, Financial Affairs (2)
Security-Reports-Property Office
Patent Coordinator
Library
Rich Electronic Computer Center
Photographic Laboratory
Project File
Other

A-3 (6-71)
GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION

Date: 2/15/80

Project Title: To Conduct a Workshop on, Field Test and Evaluate the "E-Z-LP" Computer Code

Project No: E-24-642

Project Director: Dr. John J. Jarvis

Sponsor: National Science Foundation

Effective Termination Date: 3/31/79

Clearance of Accounting Charges: 3/31/79

Grant/Contract Closeout Actions Remaining:

- Final Invoice and Closing Documents
- Federal Cash Transactions Report
- Final Report of Inventions
- Govt. Property Inventory & Related Certificate
- Classified Material Certificate
- Other

Assigned to: Industrial and Systems Engineering (School/Laboratory)

COPIES TO:

Project Director
Division Chief (EES)
School/Laboratory Director
Dean/Director—EES
Accounting Office
Procurement Office
Security Coordinator (OCA)

Library, Technical Reports Section
EES Information Office
Project File (OCA)
Project Code (GTRI)
Other

A-4 (1/79)
May 8, 1978

Progress Report: NSF Grant Number SED 75-17476

Project Director: John J. Jarvis
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

(1) A draft report describing EZLP and its capabilities has been developed. This draft report is attached.

(2) The draft report has subsequently been disseminated to over 100 universities.

(3) A number of institutions have requested further information on EZLP. Currently, around two dozen universities have EZLP operational on a variety of different computers. These numbers continue to grow as more institutions learn of EZLP.

(4) A few institutions have been selected for followup, and usage statistics are continually being collected on EZLP and other linear programming codes.

(5) EZLP is continually being refined. Feedback from other institutions has been utilized to improve and expand on the capabilities of EZLP and to make it more portable (i.e., transferable to other computer systems).

(6) Presentations have been made at national and international meetings on the development and capability of EZLP.

(7) Several commercial time sharing firms are inquiring of EZLP's availability.
FINAL REPORT  
NSF Grant number SED75-17476

EZLP: An Interactive  
Computer Program for Solving  
Linear Programming Problems

John J. Jarvis, Project Director  
Frank H. Cullen, Research Assistant  
Chris Papaconstadopoulos, Research Assistant

School of Industrial and Systems Engineering  
Georgia Institute of Technology  
Atlanta, Georgia 30332

September, 1976
This report discusses the development of an interactive computer program (EZLP) designed for student-oriented use in solving linear programming problems. The linear programming problem is inputted in the same way as it would be written on a sheet of paper.

The student may select either the (primal) simplex or the dual simplex method; the lower-upper bounded variables procedure with either method; and real (0.5) or rational (1/2) arithmetic for the calculations. EZLP has internal editing capability and is able to read from and write to permanent files.

During execution of EZLP, if the student has difficulty in remembering what to do next, he may utilize a HELP command to obtain general or specific information on EZLP's use.

Since EZLP was developed under a National Science Foundation grant a program listing is freely available (see Appendix B). A listing on cards or tape is available at a nominal charge to cover expenses. EZLP may not be resold.

Finally, a simplified user's manual suitable for handout to students learning EZLP for the first time is contained in Appendix A.
# TABLE OF CONTENTS

## Chapter 1: Introduction

1.1 Background .................................................. 1
1.2 The Current Effort ............................................ 2
1.3 The Project Plan ................................................ 3
1.4 Machine Dependence of the EZLP System ..................... 4
1.5 Availability of the EXLP System ............................... 5
1.6 Future Effort .................................................. 5

## Chapter 2: EZLP - A Narrative Description

2.1 Introduction .................................................. 6
2.2 A General Outline ............................................. 6
2.3 The Driver ..................................................... 6
2.4 The Editor ..................................................... 8
2.5 Syntactical Analysis ........................................... 9
2.6 Optimization and Output ..................................... 10
2.7 Segmentation of EZLP ......................................... 12
2.8 Machine Dependence of EZLP ................................. 13

## Chapter 3: User Instructions

3.1 Introduction .................................................. 16
3.2 Starting EZLP .................................................. 16
3.3 Entering the Model ............................................ 16
3.4 Editing the Model ............................................. 22
3.5 Solving the Model ............................................. 25
3.6 Restarting and Terminating EZLP ............................. 28
3.7 The HELP Command ........................................... 28
3.8 External File-Handling Capabilities ......................... 30

## Chapter 4: Terminal Sessions

4.1 A Simple Example of EZLP ................................... 32
4.2 An Advanced Example of EZLP ................................. 33
4.3 A Transportation Example ................................... 35
4.4 An Integer Programming Example ............................ 37

## Appendix A: A Simplified User's Manual for EZLP

## Appendix B: EZLP Program Listing
Chapter 1: INTRODUCTION

1.1 Background

Linear programming is a mathematical optimization technique utilized intensively in the fields of mathematics, management and engineering—especially within the areas of operations research and management science. The simplex method—the standard technique for solving linear programs—is an iterative method, and, as such, is dependent on the computer to carry out the rote calculations. Computerized simplex codes are available in small problem-solving versions and in large production versions in most university, private and governmental computing centers. Students find that interaction with these computer codes is often very difficult. The result is that small scale usage in the classroom or laboratory or for homework is discouraged. This bad experience carries over and tends to discourage the student's use of linear programming on the job.

In most universities today there are a number of undergraduate and graduate courses utilizing linear programming as a problem-solving tool. Georgia Tech is typical with as many as 6-10 undergraduate courses and 15-20 graduate courses utilizing linear programming. In these courses the student is expected to develop modelling skills with linear programming, appreciate the simplex method (or its variants) as a solution tool and to interpret the output results of the simplex method in the particular problem situation. In addition to course work there are usually a number of students performing research on new techniques which require a check by solving small linear programming problems. Hand calculations are good up to about four constraints and ten variables. Beyond this number the student must go to the computer in order to have the calculations performed, otherwise, negative feedback results. The student
remembers the difficulty with solving the model and not the power of
the modelling technique. When the student gets to the computer he
usually finds a whole new set of difficulties with regard to the fact
that in order to provide the computer with his model he must learn a
computer-oriented (rather than student-oriented) "input language".

A number of institutions including Northwestern University,
Stanford University, the University of Texas, and Georgia Tech have
developed linear programming computer programs which are more directly
student oriented in their design.

In the School of Industrial and Systems Engineering at Georgia Tech
an experimental project has been underway since 1971 to develop a student-
oriented conversational linear programming code (EZLP) for use on small
problems (up to 50 constraints and 100 variables). The innovation is
in the special way that the student interacts with the computer. The
student inputs his linear programming model to the computer in exactly
the same manner that he would write it down on paper. Evolution of this
code has been through class projects. Despite limitations in its initial
design, EZLP was nonetheless widely accepted by the students.

1.2 The Current Effort

The National Science Foundation agreed to support a concerted
effort, during the 1975-1976 academic year, to redesign and develop a
conversational computer code based on the concepts developed at Georgia
Tech. This final report is intended to document that effort.

To accomplish the given task, a Georgia Tech team was organized.
The Georgia Tech team consisted of:
This team had the overall responsibility of designing, developing, coding and documenting the computer system.

An Advisory Committee of eminent researchers and practitioners in the field of linear programming was also organized to provide guidance to the Georgia Tech team. The Advisory Committee consisted of the following individuals:

1. Dr. Claude Cohen  
   Northwestern University
2. Dr. Harvey J. Greenberg  
   Federal Energy Administration
3. Dr. Marvin A. Griffin  
   University of Alabama
4. Dr. Michael E. Thomas  
   University of Florida

This committee had significant impact on the design of the final computer system.

1.3 The Project Plan

During the early stages of the project the Georgia Tech team devoted a great deal of time and effort to the development and documentation of a preliminary design of a computer system for EZLP. This documentation was submitted to the Advisory Committee for their review.

A meeting of the Advisory Committee was conducted on the Georgia Tech campus to discuss the preliminary design. This discussion lead to a much improved design. On advice of the committee the fundamental concept of
complete machine independence was replaced by a moderate level of machine
dependence to permit substantial reduction in program size and complexity.
Also, a number of general features were eliminated from the initial design
while many others were added. The Georgia Tech team was very pleased with
the structure that emerged from that meeting.

The Advisory Committee also suggested, at its meeting, that the Georgia
Tech team retain a certain degree of flexibility in its development of EZLP
so that the Tech team might be able to react in an expeditious manner to
potential difficult situations that might arise during coding.

The remainder of the grant period, following the Advisory Committee
meeting, was spent developing and testing the EZLP code. With a few
changes, the current EZLP system is the one which emerged from the Advisory
Committee meeting.

The remaining chapters discuss the specific structure of EZLP and its
use. Appendix A contains a simplified user's manual which could be passed
out to students learning EZLP for the first time. Appendix B contains the
listing of EZLP.

1.4 Machine Dependence of the EZLP System

The EZLP computer system has been intentionally designed to minimize
the level of machine dependence within the constraint of reasonable pro-
gram size and complexity. EZLP has been coded in FORTRAN and special
forms which may be available only on the CDC Cyber 74 (the machine used)
are avoided. Those places in the code where machine dependence was
unavoidable are few in number and clearly identified.

Section 2.8 of Chapter 2 discusses the required changes to convert
the machine dependent portions of EZLP to other computer systems.
1.5 Availability of the EZLP System

Development of the EZLP computer system was sponsored by a grant from the National Science Foundation. As such the EZLP computer system is freely available to anyone desiring a copy. A computer listing on paper is available at no charge. There is a nominal charge, to cover expenses, for a listing on cards or tape. EZLP may not be resold. For further information write

Dr. William H. Adams, Program Manager
Division of Science Education Development and Research
National Science Foundation
Washington, D. C. 20550

or

Dr. John J. Jarvis, EZLP Project Director
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

1.6 Future Effort

EZLP is an evolving computer system. Many features simply could not be developed under the current contract. Effort continues at Georgia Tech to maintain and improve the EZLP code.
Chapter 2: EZLP - A NARRATIVE DESCRIPTION

2.1 Introduction

This chapter presents the workings of EZLP in a narrative form. Emphasis is given to the identification of the components of EZLP, their attributes, and a discussion of their mutual interaction. Coded in standard FORTRAN, EZLP nonetheless has certain functions which cannot be written in machine-independent code. These are discussed in the last part of this chapter.

2.2 A General Outline

In overview, EZLP consists of three main subprograms tied together by a controlling main program, or "driver." The three main subprograms are:

a. Editor - the ability to create and maintain (update) a current model kept physically in an exterior mass storage device

b. Syntactical Analysis - inputting the model statements from the external file created by the EZLP Editor, performing parsing and syntax-checking operations, and generating a tableau for input to the optimization process

c. Optimization - a solution and output of a model in tableau form by simplex procedures

Figure 2.1 presents a general diagram of the program flow of EZLP which also indicates the interaction of the three main subprograms.

2.3 The Driver

The main program, or driver program, has several functions. Among these are:

a. distribute and pass control among the three main subprograms

b. perform ancillary operations not handled by the subprograms
Subprogram: Editor

EZLP Editor

Model Entry

And "St"
Also "Min"
"Max"

Add" Change"
"Delete"
"List"

Editing Input
Model

"Model"
"Run"
"Save"

Ancillary Operations

Model Entry

Syntactical Analysis

Simplex Optimization Procedures

Tableau and Basis Output

Subprogram: Optimization

Subprogram: Syntactical Analysis

Figure 2.1: EZLP Program Logic Flow
Upon program execution, the driver program assumes control and examines the first input statement. If the keyword (the first word) of the input statement is an editor associated keyword (e.g., AND, MIN, CHANGE), the driver continues to parse the input statement to determine the row name if it is there and to construct the default value if it is not; control is then passed to the editor subprogram. If the keyword of the input statement is associated with an ancillary operation (e.g., RUN, SAVE, or PRINT), the driver continues to parse the input statement to determine what internal program switches are to be set, and in the case of the SAVE statement, perform the requested file operations. If the keyword is USE, to trigger a solution attempt, the driver parses the remainder of the input statement to determine the simplex method to be used, passes control first to the syntactical analysis subprogram, and then to a tableau-building subroutine and the optimization subprogram in succession, if there are no syntactical errors in the model.

In Figure 2.1 the driver program may be considered to be everything not within a subprogram boundary.

2.4 The Editor

The editor subprogram is primarily responsible for model file maintenance and performs four principal functions:

a. Adds a statement to the model file
b. Deletes a statement in the model file
c. Changes a statement in the model file
d. Lists either a single row or the entire contents of the model file

Model File Organization

The model file itself is a simple linked-list structure with chaining used for continuation lines. Two vectors in the common block EDIT, CNTNME
and LINKS, store the alpha name and the relative location of the first record for the particular row of the model on mass storage. This record has a forward link to a continuation record if used. This forward link is zero (0) if there are no further continuations after the current record. The program logic and passed parameters to the mass storage read and write subroutine are such that first-available and other random access techniques can be used on computers of various manufacture.

Subprogram Operation

Upon passage of control and the appropriate parameters, the Editor accesses the model file to perform the desired function, and returns control to the driver which then reads in the next input statement.

Continuations of rows in the model are handled directly by the Editor, which reads in successive input statements and sets up the chaining relationship until the continuation situation no longer exists.

2.5 Syntactical Analysis

The syntactical analysis subprogram is the process wherein the model file contents are read in, parsed, checked for errors in syntax, and translated into a coded form which is subsequently transformed into a tableau structure.

In overview, the syntactical analysis subprogram has two main entities:

a. the model statement parser, which takes lines from the model file and generates a uniform symbol table in which the components of the model line (e.g. coefficient, name or identifier, delimiter) are described and;

b. the analysis routine which uses as input the uniform symbol table, checks for syntax errors, and outputs a quasi-diagonal tableau
which later serves as input to a tableau-building subroutine which adds slack and artificial variables and generates a rectangular tableau structure.

Other output from the syntactical analysis subprogram includes:

a. a vector containing row names as they appear in the tableau

b. a vector containing variable names as they were first mentioned in the model file

c. vectors containing lower and upper bounds for each variable in the same order as (b)

d. A designator of which rows of the tableau are associated with objective functions

e. an indication of which objective function row is to be optimized during phase two of the optimization process

In the instance that there are syntax errors present in the model file, descriptive error messages and pointers are printed in an attempt to pinpoint the exact location of the trouble.

The memory locations used by the syntactical analysis are focused in the common area LEX.

2.6 Optimization and Output

The optimization and output subprogram is the process by which the simplex operations are performed on the rectangular tableau created by the tableau-building subroutine which, in turn, receives its input from the syntactical analysis subprogram. Depending upon the value of the parametric switch METHOD set by the driver program during the parsing of the USE statement, the optimization process selects from four basic algorithms: primal or dual simplex with or without rational arithmetic. The lower-upper bounded simplex approach is used throughout the optimization routines. The different
results obtained by the specification of the UPPER parameter in the USE statement owe to the generation of explicit bound constraints by the syntactical analysis subprogram when UPPER is not specified.

The primal algorithms (real and rational) employ a two-phase method for which the Phase 1 objective function and the starting basis are determined by the tableau-building subroutine. At the successful completion of Phase 1, the Phase 2 objective function (which has been stored as a null constraint in the tableau) is written over the Phase 1 objective function and the process resumes.

The dual algorithms begin with the creation of the Phase 2 objective function from the appropriate null constraint (objective function) in the tableau.

The rational algorithms require the maintenance of both a numerator and a denominator for each tableau entry (and each element of the right-hand side). For example, the numerator for an element in an array may be stored in position I, while the corresponding denominator will be stored in position J+I, where the Jth position of the array held the value of the numerator of highest subscript index.

At the end of each simplex iteration, or both before and after the optimization process has been completed, there might be output specified by the user via a PRINT statement. In this instance, control within the optimization process is passed to a point which prints out the values of the nonzero variables, the current basis and basis activity, or the current tableau and updated objective function. Control is then returned to the appropriate place in the optimization process.

Upon reaching an optimal solution, control within the optimization process passes to a point at which the default output (the optimal primal and dual solutions) is printed out. Even if no other output is specified or requested, the default output is printed.
When the printing default output is completed, control returns to the driver program which then reads in and parses the next input statement.

The memory locations used by the optimization and output subprogram are focused in the COMMON area OPT.

### 2.7 Segmentation of EZLP

In dealing with computers with smaller capacity than large-scale university computing systems, memory economy via code segmentation or overlay is often of considerable interest.

The modular subprogram concept allows some or all of the subprogram code to be overlayable, and hence less wasteful.

Specifically, the three subprograms are mutually overlayable as well as those functions which serve only one of the subprograms. Graphically, we can represent the code components of EZLP as a tree, the root of which is non-overlayable, and the separate branches of which form the overlays. In Figure 2.2, this tree structure is represented in terms of the actual FORTRAN subroutines and function names.

![Figure 2.2: Segmentation Tree-Structure of EZLP](image-url)
2.8 Machine Dependence of EZLP

There are two aspects of the code of EZLP which are likely to cause difficulties when an attempt is made to convert the code to a machine other than the one for which EZLP was written (the CDC Cyber 74). These are

a. Alpha variables - packing and unpacking, and manipulation

b. File manipulations - particularly with respect to error and end-of-file conditions

Alpha Variables

The storing of alphanumerical information in variables has always been a weakness in the design of FORTRAN. Consequently, the use of either integer or real variables to store alphanumerical information is bound to be non-standard and change from machine to machine. For the purpose of program design and implementation, EZLP takes a rather hybrid approach to this problem by using integer variables for the branch of the code tree which deals with syntactical analysis (see Figure 2.2) and real variables for the remainder. For those compilers and computers which require that only one or the other type be used for alpha variables, explicit type statements must be added to those portions of the code affected for the alpha variables.

A basic assumption of the EZLP code is that the alpha variables can hold at least eight (8) characters and are appropriately filled so that a comparison with a Hollerith constant is meaningful. Hexadecimal machines (or character machines) normally have type statements which allow the length of the variable in bytes (characters) to be specified. Where applicable, these constructs should be used.

An often-used subroutine in EZLP for composing a single eight-character alpha variable from eight single-character alpha variables is called PACK.
This subroutine is totally machine-dependent and must be changed for different computers.

**File Manipulation**

EZLP employs a number of files during execution. These files are summarized in Table 2.1. These files, with the possible exception of file 8, are strictly sequential files and require no special programmatic considerations. file 8, the model file, because of the random way in which editing is applied to it, can be effectively used as a random access file. Since random access is normally a machine-dependent function, EZLP separates out the random access read and write functions into separate subroutines MSREAD and MSWRIT. These two subroutines must be re-written for different computers.

<table>
<thead>
<tr>
<th>File Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Input file (from the terminal)</td>
</tr>
<tr>
<td>6</td>
<td>Output file (to the terminal)</td>
</tr>
<tr>
<td>7</td>
<td>Message file (for syntax error messages)</td>
</tr>
<tr>
<td>8</td>
<td>Model file (linked and chained mass storage)</td>
</tr>
<tr>
<td>9</td>
<td>Run file (alternate input file)</td>
</tr>
<tr>
<td>10</td>
<td>Save file (model output file)</td>
</tr>
<tr>
<td>11</td>
<td>Help stored-text file (instructions)</td>
</tr>
</tbody>
</table>

End-of-file checking is also achieved differently with different FORTRAN compilers. On the CYBER 74, the test is the implicit function EOF(u). Occurrences of this function should be replaced by the appropriate READ statement construct for the local FORTRAN compiler. For example, the sequence

```fortran
READ (INFILE, 102) (BUFFER(I), I=1, 80)
IF (EOF(INFILE) .NE.O) GO TO 320
```
could be replaced by something resembling

```
READ (INFILE, 102, END=320) (BUFFER(I), I=1, 80)
```

if such was the appropriate end-of-file construct.
Chapter 3: USER INSTRUCTIONS

3.1 Introduction and Notation

This chapter is designed to describe the operation of EZLP from the user's point-of-view. The material covered here is somewhat more comprehensive than that which would be required for the user with simple problems. Appendix A contains a simplified user's manual.

For clarity of presentation, this chapter uses BNF notation in describing syntactical rules. Briefly, capitalized words indicate keywords, [] indicates an optional clause, \{\} indicates a choice of two or more alternatives, and <> enclose a descriptive term for a syntactical entry.

3.2 Starting EZLP

The exact syntax of executing EZLP will vary from computer to computer and from installation to installation, and so cannot be explicitly stated here. For the sake of clarity, let it be assumed that EZLP has been executed and is awaiting input from the user.

At this point, the user has the option of proceeding to enter the model (3.3) or getting a brief (about two pages) description of how EZLP is operated. This description, in the absence of this chapter or other documentation should supply a minimum of information to the user so that he can use the program. To obtain this brief description, the user must enter the following command:

HELP

This command is discussed further in section 3.7.

3.3 Entering the Model

The first thing the user must do is enter the model. The entry of the model follows certain rules - there are rules concerning the order in which inputted statements must be organized, and there are rules which govern the
way in which different types of statements are made up. Before these rules are discussed, it would be best if some basic ideas were explained: A statement is a single logical input. A constraint is a statement. The objective function is a statement. A delimiter is a single character which serves to terminate one entry and perhaps begin the next. For the entry of the model, the space character (\$) is the usual delimiter. A name provides the convenience of letting the user assign unique identifiers to variables, constraints, and objective functions. The names themselves are composed of from one to eight characters. The first character must be alphabetic (A-Z), while the remaining characters must be alphabetic, numeric (0-9), or numerics separated by commas. X2, PROFIT, ITEM3, and Y2,3 are examples of acceptable names, while ITEM-3 and $VAR are not. A space character cannot be part of a name. EZLP reserves certain names for its internal use. Row names of the form ROW\#n are reserved for each objective function or constraint statement which was unnamed by the user. This generated name can be used for editing purposes. In addition, EZLP uses the names SLK\#n and ART\#n for the slack and artificial variables that it generates. Since the user may not input any name containing the \# character there can never be any confusion between user-generated names and EZLP-generated names. An input-line corresponds to a line of input on the terminal. In batch processing, this is equivalent to a card image.

If desired the user may indicate that heading information is to be printed at the top of the optimal solution output. The user must type in

\[ \text{TITLE } <\text{heading information}> \]

where \(<\text{heading information}>\) appears as a heading at the top of the output.
The actual entry of the model is divided into two basic parts: the objective function and the constraints.

**Entering the Objective Function**

The syntactical rule for the entry of the objective function is:

\[
\{\text{MIN}\} \text{ or } \{\text{MAX}\} \left[ \text{<objective-function-name>} \right]: \text{<arithmetic expression>}
\]

where <objective-function-name> is an optional user-assigned name for the objective function and <arithmetic expression> is a linear combination of variable-names.

Note that the colon (:) is required at all times.

If the objective function statement is too large to be entered on one input line, continuation of the statement is accomplished by placing an ampersand (&), after the last character of the current input line and continuing the statement on the next input line. There are no limits on the number of continuations allowed for a single statement.

Examples of possible objective function statements:

1. **MIN**: \(2 + 3X_2 - 2.3X_4\)
2. **MAX PROFIT**: \(3\text{INCOME} - 4\text{EXPENSE} - .673\text{OVERHD}\)
3. **MIN COST**: \(2.4\text{VAR}_1 - 3.53\text{VAR}_2 + .004\text{VAR}_3 \& -\text{VAR}_4\)

**Alternate Objective Functions**

In some applications of linear programming, it is of interest to consider a number of possible objective functions subject to the same set of constraints. EZLP allows for the entry and subsequent optimization of the entered model in coordination with alternate objective functions.
The syntactical rule for the entry of an alternate objective function is:

\[
\text{ALSO [<objective-function-name>]: <arithmetic expression>}
\]

Notice that the above differs from the ordinary entry of the objective function in that the keyword "ALSO" is required and replaces the keywords "MIN" and "MAX".

Each model must contain exactly one primary objective function (using the keywords "MAX" or "MIN"). However, a model may contain any number of alternate objective functions. EZLP operates with only one objective function at a time.

Specifying an alternate objective function for the purpose of optimization is accomplished just prior to specifying the method of solution. This is discussed in Section 3.5 ii.

**Entering the Constraints**

The constraints associated with normal linear programming model can be broken down into three general classes:

1. Simple arithmetic constraints - those constraints which involve more than one variable and only one relational operator; and
2. Range constraints - those constraints which involve more than one variable and two identical operators; and
3. List constraints - those constraints which involve only one variable. Constraints of this type usually specify upper or lower bounds on a particular variable or a list of variables.
Simple Arithmetic Constraints

The syntactical rule for the entry of a simple arithmetic constraint is:

\[
\begin{align*}
&\text{[ST]} \quad \text{[constraint-name]} : \quad \text{arithmetic expression} \\
&\quad \text{AND} \\
&\quad \text{relational operator} \quad \text{arithmetic expression}
\end{align*}
\]

where <constraint-name> is an optional user-assigned name for the particular constraint. The <arithmetic expression> is a linear combination of variable names as described in the objective function statement above; and <relational operator> takes one of the following forms:

1. \( \geq \) or \( \geq \) (greater than or equal to)
2. \( = \) (equal to)
3. \( \leq \) or \( \leq \) (less than or equal to).

Note that the colon is required at all times.

If any particular constraint statement is too large to be entered on one output line continuation of the statement is accomplished by placing an ampersand (\&) after the last character of the current input line and continuing the statement on the next input line. There are no limits on the number of continuations allowed for a single constraint statement.

The first constraint statement should begin with the keyword "ST", which stands for "subject to". Subsequent constraint statements must begin with the keyword "AND".

Constraint statements which include the optional <constraint-name> are easily referenced for editing by this user-assigned name. Unnamed constraints are assigned a name by EZLP so that the user can also reference and edit these constraints. This is further discussed in section 3.4.

Examples of possible simple arithmetic constraints:
1. AND CNSTR2: 20WIDTH2 <= 15 - LENGTH
2. AND POWER: TOTAL + 3TIMEAVL >= 16.45
3. AND: 4.3x1 + 6x7 = 4.57

Range Constraints

EZLP is designed to accommodate range constraints. The syntactical rule for the entry of range constraints is:

\[
\{ \text{ST}\} \quad \{ \text{AND}\} \quad \{ <\text{constraint-name}> \} : \quad <\text{constant-1}> <\text{inequality-operator-1}> \quad AND \quad <\text{arithmetic expression}> <\text{inequality-operator-2}> <\text{constant-2}>
\]

where the following rules apply:

1. <inequality-operator-1> and <inequality-operator-2> must be exactly alike and must be either >= or <=.
2. If the relational operators are >=, then <constant-1> must be greater than or equal to <constant-2>
3. And, if the relational operators are <=, then <constant-1> must be less than or equal to <constant-2>.

Examples of possible range constraints are:

1. AND: 4 <= 3x1 + 4.2x3 - 4x4 <= 6.2
2. AND WORKERS: 500 >= 6.3FORCE1 - 5.3FORCE2 >= 243
3. AND CONSTRNT: 3.4 <= 7 + VAR3 <= 8.46

List Constraints

The syntactical rule for the entry of a list constraint is:

\[
\text{AND} [ <\text{constraint-name}> ] : \quad \{ \text{ALL} \ [ \text{OTHER} ] \ \text{VARS} \} \quad \{ \text{<variable-list>} \} \quad \{ \text{<relational operator>} <\text{constant}> \} \quad \{ \text{URS} \}
\]
where <constraint-name> and <relational operator> are described above for simple arithmetic constraints; and <variable-list> is a list of one or more variable names, separated by commas (,).

The reserved phrase "ALL VARS" indicates that the bound specified applies to all variables in the model. The reserved phrase "ALL OTHER VARS" indicates that the bound specified applies to all variables which are not included in another list constraint for the same type bound (i.e., upper or lower).

The reserved word "URS" stands for "unrestricted in sign".

The default option for all variables if they do not appear in a list constraint is "URS". EZLP prints a notification of the unrestricted variables.

Examples of possible list constraints:

1. AND: X1, X2, X3 <= 0
2. AND: ALL VARS >= 0
3. AND: OIL <= 375.43
4. AND: ALL OTHER VARS <= 1
5. AND: PROFIT URS
6. AND MYCNSTR: WIDTH, LENGTH >= 0

An example of an entry of a complete model:

MIN: 10.5WIDTH1 + 11.8WIDTH2 - 30LENGTH

ST CNSTR1: 15LENGTH - 2WIDTH1 <= 10
AND CNSTR2: 20WIDTH2 + LENGTH <= 15
AND: ALL VARS >= 0

3.4 Editing the Model

After completion of the model entry, the user has the ability to perform certain editing functions. These functions are:
1. Adding a constraint or objective function
2. Deleting a constraint or objective function
3. Changing a constraint or objective function
4. Listing the model in whole or in part

i) Adding a Constraint or Objective Function

At all times the user has direct access to his model. Thus, addition of a new constraint or objective function to the end of the current model may be accomplished by simply typing the appropriate constraint or objective function statements. EZLP will automatically append the new statement to the previous model.

For example, suppose that the user has attempted to solve his model and this attempt resulted in an unboundedness indication. Having determined that he failed to require nonnegativity he may do so by simply typing

\[
\text{AND: ALL VARS } \geq 0
\]

If one wishes to insert a constraint into the middle of the current model, this may be accomplished by typing

\[
\text{INSERT \{AFTER \{<row-name> \{model entry statement\}\}}
\]

where \(<model entry statement>\) is either a constraint or objective function statement. Examples of the INSERT command are:

1. INSERT AFTER ROW#2 AND: \(X1 + 2X2 \leq 7\)
2. INSERT BEFORE CONST6 AND CONST5: \(7 \leq X1 \leq 9\)
3. INSERT AFTER ROW#3 ALSO: \(3X1 + 2\text{POWER}\)
ii) Deleting a Constraint or Objective Function

The syntactical rule for the deletion of a constraint is:

```
DELETE <row-name>
```

where `<row-name>` is either the user assigned name, or in the absence of this name, the row name assigned by EZLP. The assigned name is simply `ROW#n`, where `n` is the number of the model entry as it was entered. Examples of assigned row-names are `ROW#4`, `ROW#13`, and `ROW#123`.

1. DELETE ROW#13
2. DELETE SURPLS

iii) Changing a Constraint or Objective Function

The syntactical rule for the changing of a constraint or the objective function is:

```
CHANGE <row-name> "<string1><string2>"
```

where `<row-name>` must be identical to some existing row name.

The `CHANGE` command will replace `<string1>` by `<string2>`. The construct "<string1>" will delete `<string1>`. Examples of possible `CHANGE` statements are:

1. CHANGE CONSTR1 "ENG"G"
2. CHANGE SURPLS "3.2"

iv) Listing the Model

The syntactical rule for listing the model on the terminal is:

```
LIST [<row-name>]
```
where the optional <row-name> is included if only a single row is to be printed. If <row-name> is omitted the entire model will be printed. Examples of possible LIST commands are:

1. LIST
2. LIST ROW#5
3. LIST MYCSTR

3.5 Solving the Model

Once the model has been entered and edited to the user's satisfaction, the user takes the following steps:

1. Specifying the Desired Output (this step is optional and, if omitted, results in only the optimal solution being printed).
2. Specifying the Objective Function to be Optimized (this step is optional and is only to be used when selecting an alternate objective function).
3. Specifying the Method of Solution.

i) Specifying the Desired Output

If the user wishes to obtain more output than the optimal solution he may use the PRINT command.

The syntactical rules for the PRINT command are:

\[
\text{PRINT [INITIAL] [FINAL] \{VARS, BASIS, \text{TABLEAU} \},<frequency count>] ALL \{NONE\}}
\]

This statement concerns the output on the terminal of information directly related to the optimization process. The following descriptions apply:
VARS: prints the name and value of each nonzero primal variable; and the name and value of each nonzero dual variable.

BASIS: prints the names of the variables in the basis and the objective function value.

TABLEAU: prints the tableau.

ALL: prints all of the above.

NONE: prints only the optimal solution.

<frequency-count>--the PRINT statement is executed every <frequency-count> iterations. If this optional clause is omitted, the PRINT statement will be executed after every iteration unless the keywords INITIAL and/or FINAL are used. When ALL or NONE are present <frequency-count> is ignored.

INITIAL: prints the requested information only for the initial iteration.

FINAL: prints the requested information only for the final iteration.

(Note that INITIAL and FINAL may be used together.)

Examples of possible PRINT commands are:

1. PRINT VARS 5
2. PRINT ALL
3. PRINT BASIS, VARS, 3
4. PRINT FINAL TABLEAU
5. PRINT INITIAL, FINAL BASIS

ii) Specifying the Objective Function to be Optimized

In the event that the user has entered alternate objective functions, he may wish to specify the name of the objective function to be optimized. If this is not done, EZLP defaults to either the original objective function entered with the model or the last objective function name specified in a prior ALTOBJ statement. The syntactical rule for the specification of the objective function command is
If an ALTOBJ command has been previously given, then to determine the row name of the current objective function being used, the user should type

```
ALTOBJ  {MIN}  <objective-function-name>
```

iii) Specifying the Method of Solution

The specification of the method of solution triggers an attempt by EZLP to solve the entered model and produce output as specified in 3.5.1. The syntactical rule for specifying the method of solution is:

```
USE [RATIONAL] [UPPER] {PRIMAL} {DUAL}
```

Specifying RATIONAL keeps all data in rational form, i.e. "0.5" would become "1/2". Since the RATIONAL option requires storing both a numerator and a denominator matrix, the maximal allowable problem size is cut in half under this option. Further, all coefficients and constants must be entered as integers.

Specifying UPPER before PRIMAL or DUAL will cause the lower-upper bounded primal or dual algorithm to be used.

Examples of possible USE commands are:

1. USE PRIMAL
2. USE UPPER DUAL
3. USE RATIONAL PRIMAL
4. USE RATIONAL UPPER PRIMAL
3.6 Restarting and Terminating EZLP

At the completion of any model solution the user has three options:

1. Editing the current model and re-solving it,
2. Starting fresh with a new model, or
3. Terminating EZLP

The first of these options is accomplished by simply entering the appropriate edit statements (see 3.4) followed by the appropriate USE commands (see 3.5).

The second of these options is accomplished by the RESTART command. The syntax of this command is:

```
RESTART
```

The RESTART command re-initializes all areas. The effect of this command is to clear out the current model and print options.

Terminating EZLP is accomplished by the END command. The syntax of this command is:

```
END
```

3.7 The HELP Command

This command allows the user, after he has started the program, to obtain a short set of instructions on how to solve simple LP problems. The user, at any point in his run, may request additional help in the following manner:

```
HELP [<keyword>]
```
where the optional clause keyword refers to abbreviated instructions concerning a particular area of interest. A list of keywords is:

General Keywords:
- EDIT (describes edit commands)
- EXAMPLES (presents examples of EZLP models)
- FILES (discusses external file handling capabilities)
- KEYWORDS (gives the current list of keywords)
- SOLVING (indicates the methods and options for optimization and discusses output options)
- MODEL (discusses model entry syntax)

Specific Keywords:
- ADD (indicates how to add a constraint or objective)
- ADVANCED (presents an advanced example)
- ALTOBJ (indicates how to specify an alternate objective function for optimization)
- CHANGE (indicates how to change a constraint or objective)
- CONT (discusses the continuation of model statements)
- CONS (describes the options for inputting constraints)
- DELETE (indicates how to delete a constraint or objective)
- LIST (discusses the list options for model display)
- NAMES (provides a definition of acceptable variable names)
- OBJ (discusses objective functions)
- PRINT (discusses the output print options)
- RUN (describes procedure for inputting a model from an external file)
- SAVE (discusses procedure for saving a model)
- TITLE (discusses the options for method of optimization)
- USE (provides a definition of acceptable variable names)
If the keyword is omitted, EZLP will print a short (about two pages) description of how to input and run a model.

3.8 External File-Handling Capabilities

There are certain situations wherein the user wishes to eliminate input effort by storing all or part of his input on a mass storage device. EZLP provides for this capability with two commands: the RUN command and the SAVE command.

i) The RUN Command

At any point in the execution of EZLP, the user may elect to refer EZLP to a mass storage file for subsequent input. This input must be in the form of 80 character source records and must be accessible sequentially by EZLP. The syntax of the RUN command is

\[
\text{RUN <file-name>}
\]

If the RUN command is inputted, EZLP will read the model directly from the working file <file-name>, print a question mark and await the next command.

ii) The SAVE Command

Upon completion of model entry, the user may elect to save the input model in source form in a mass storage file. The syntax of the SAVE command is

\[
\text{SAVE <file-name>}
\]

If the SAVE command is inputted, EZLP will write the current model into the working file <file-name>, print a question mark and await the next command.
Chapter 4: TERMINAL SESSIONS

Introduction

This chapter contains four examples of EZLP used to solve normal linear programming problems. These examples are:

1. Simple example - getting on, solving a simple problem, and getting off.

2. Advanced example - naming rows, editing the model, requesting additional output, and responding to an error in syntax.

3. Transportation problem example - two sources and two sinks to illustrate multiply-indexed variable names and the treatment of primal redundancy.

4. Branch and Bound example - assignment of titles to optimal solution output in addition to using the editing features of EZLP to easily solve a small integer programming problem using the Branch and Bound method.
Session 1: A Simple Example

EZLP - VERSION 9/17/76
TYPE HELP IF YOU HAVE QUESTIONS; OTHERWISE PROCEED
? MAX: 3X1+2X2
? ST: 5X1+3X2 <= 19
? AND: X1-X2 >= 3
? AND: ALL VARS >= 0
? USE RATIONAL PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>X2</td>
</tr>
</tbody>
</table>

SOLUTION

OBJECT MAXIMIZE ROW#1
Z 23/2
ITERATIONS 2

VARIABLE SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>7/2</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>X2</td>
<td>1/2</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

CONSTRAINT SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0</td>
<td>5/8</td>
<td>1/2</td>
</tr>
<tr>
<td>ROW#3</td>
<td>0</td>
<td>-1/8</td>
<td>7/2</td>
</tr>
</tbody>
</table>

? END

.223 CP SECONDS EXECUTION TIME
Session 2: An Advanced Example

EZLP - VERSION 9/17/76
TYPE HELP IF YOU HAVE QUESTIONS, OTHERWISE PROCEED

? MIN: 4.3 A - 5.6 B ++ 7 C
? ST: A + C = B + 7
? AND BOUNDA: A <+ C + 7
? CHANGE BOUNDA "<+<="
? AND BOUNDA: A <= C + 7
? AND BOUNDB: 1 <= B <= 6
? AND: A>= 0
? USE UPPER PRIMAL

MIN: 4.3 A - 5.6 B ++ 7 C

FATAL ERROR # 1 : ARITHMETIC OPERATOR IN WRONG PLACE

** UNRESTRICTED VARIABLES **

C

FATAL ERRORS IN THE MODEL.
PLEASE EDIT THE MODEL
INPUT WILL BE RECOMPILED

? LIST
ROW#1
MIN: 4.3 A - 5.6 B ++ 7 C
ROW#2
ST: A + C = B + 7
BOUNDA
AND BOUNDA: A <= C + 7
BOUNDB
AND BOUNDB: 1 <= B <= 6
ROW#5
AND: A>= 0
? CHANGE ROW#1 "+++
MIN: 4.3 A - 5.6 B + 7 C
? PRINT FINAL ALL
? USE UPPER PRIMAL

** UNRESTRICTED VARIABLES **

C

** VARIABLE LIST **

A   B   C
PHASE 2 ITERATION - 1 CURRENT OBJ VALUE = .3015E+02

PRIMAL NON-ZERO VARIABLES
A BASIC WITH VALUE .7500E+01
B NON-BASIC WITH VALUE .1000E+01
C BASIC WITH VALUE .5000E+00

DUAL NON-ZERO VARIABLES
ROW#2 DUAL = .545E+01
BOUND A DUAL = -.135E+01

THE CURRENT BASIC VARIABLES ARE -
C A

CURRENT TABLEAU -

<table>
<thead>
<tr>
<th>RHS</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>SLK#3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#1</td>
<td>-.302E+02</td>
<td>0.</td>
<td>.500E-01</td>
<td>0.</td>
</tr>
<tr>
<td>C</td>
<td>.500E+00</td>
<td>0.</td>
<td>-.500E+00</td>
<td>.100E+01</td>
</tr>
<tr>
<td>A</td>
<td>.750E+01</td>
<td>.100E+01</td>
<td>-.500E+00</td>
<td>0.</td>
</tr>
</tbody>
</table>

UPDATED OBJECTIVE FUNCTION ROW
0.  -.500E-01  0.  -.135E+01

SOLUTION
----------

OBJECT MINIMIZE ROW#1
Z .3015E+02

ITERATIONS 1

VARIABLE SECTION
-----------------

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>.7500E+01</td>
<td>0.</td>
<td>.1000E+22</td>
<td>0.</td>
</tr>
<tr>
<td>B</td>
<td>.1000E+01</td>
<td>.1000E+01</td>
<td>.6000E+01</td>
<td>.5000E-01</td>
</tr>
<tr>
<td>C</td>
<td>.5000E+00</td>
<td>-.1000E+22</td>
<td>.1000E+22</td>
<td>0.</td>
</tr>
</tbody>
</table>

CONSTRAINT SECTION
------------------

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0.</td>
<td>.5650E+01</td>
<td>.5000E+00</td>
</tr>
<tr>
<td>BOUNDA</td>
<td>0.</td>
<td>-.1350E+01</td>
<td>.7500E+01</td>
</tr>
</tbody>
</table>

? END

.400 CP SECONDS EXECUTION TIME
Session 3: A Transportation Problem Example

EZLP - VERSION 9/17/76
TYPE HELP IF YOU HAVE QUESTIONS, OTHERWISE PROCEED
? TITLE TRANSPORTATION PROBLEM - SOLUTION VIA EZLP.
? MAX TRANSOBJ: 2X1,1 + 3X1,2 + 4X2,1 + 2X2,2
? ST SUPPLY1: X1,1 + X1,2 = 3
? AND SUPPLY2: X2,1 + X2,2 = 4
? AND DEMAND1: X1,1 + X2,1 = 5
? AND DEMAND2: X1,2 + X2,2 = 2
? AND: ALL VARS >= 0
? PRINT BASIS
? USE RATIONAL PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

X1,1 X1,2 X2,1 X2,2

PHASE 1 ITERATION - 1 CURRENT OBJ VALUE = 14
THE CURRENT BASIC VARIABLES ARE -
   ART#2 ART#3 ART#4 ART#5

PHASE 1 ITERATION - 2 CURRENT OBJ VALUE = 8
AT THIS ITERATION, X1,1 ENTERED THE BASIS AND ART#2 LEFT
THE CURRENT BASIC VARIABLES ARE -
   X1,1 ART#3 ART#4 ART#5

PHASE 1 ITERATION - 3 CURRENT OBJ VALUE = 4
AT THIS ITERATION, X2,1 ENTERED THE BASIS AND ART#4 LEFT
THE CURRENT BASIC VARIABLES ARE -
   X1,1 ART#3 X2,1 ART#5

PHASE 1 ITERATION - 4 CURRENT OBJ VALUE = 0
AT THIS ITERATION, X1,2 ENTERED THE BASIS AND ART#3 LEFT
THE CURRENT BASIC VARIABLES ARE -
   X1,1 X1,2 X2,1 ART#5

PHASE 2 ITERATION - 1 CURRENT OBJ VALUE = 24
THE CURRENT BASIC VARIABLES ARE -
   X1,1 X1,2 X2,1 ART#5
TRANSPORTATION PROBLEM - SOLUTION VIA EZLP.

SOLUTION

OBJECT
Z
MAXIMIZE
TRANSOBJ
24

ITERATIONS
1

VARIABLE SECTION

NAME ACTIVITY LEVEL LOWER_BOUND UPPER_BOUND REDUCED COST
X1,1 1 0 POS INF 0
X1,2 2 0 POS INF 0
X2,1 4 0 POS INF 0

CONSTRAINT SECTION

NAME SLACK ACTIVITY DUAL_PRICE RHS_VALUE
SUPPLY1 0 3 1
SUPPLY2 0 5 2
DEMAND1 0 -1 4
DEMAND2 0 0 0

END

.413 CP SECONDS EXECUTION TIME
Session 4: A Branch and Bound Example

If we consider the integer program

\[
\begin{align*}
\text{Max } & \quad X_1 + X_2 \\
\text{ST } & \quad X_1 + 3X_2 \leq 9 \\
& \quad 3X_1 + X_2 \leq 9 \\
& \quad X_1, X_2 \geq 0 \text{ and integer}
\end{align*}
\]

described by the graph

Using EZLP, we can construct the following Branch and Bound tree:

Node 0

- \( Z_0^* = 9/2 \)
- \( X_1^* = X_2^* = 9/4 \)

Node 1

- \( Z_1^* = 13/3 \)
- \( X_1^* = 2 \)
- \( X_2^* = 7/3 \)

- \( X_2 \leq 2 \)

Node 2

- \( Z_2^* = 4 \)
- \( X_1^* = X_2^* = 2 \)

(Incumbent Solution)

Node 4

- \( Z_4^* = 3 \)
- \( X_1^* = 3 \) (fathomed by Node 2)
- \( X_2^* = 0 \)

Node 3

- \( Z_3^* = 3 \)
- \( X_1^* = 0 \) (fathomed by Node 2)
- \( X_2^* = 3 \)
** REGULAR UPPER PRIMAL **

** VARIABLE LIST **

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>9/4</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>X2</td>
<td>9/4</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

** CONSTRAINT SECTION **

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0</td>
<td>1/4</td>
<td>9/4</td>
</tr>
<tr>
<td>ROW#3</td>
<td>0</td>
<td>1/4</td>
<td>9/4</td>
</tr>
</tbody>
</table>

? AND: X1 <= 2

? TITLE NODE 1 - ADDITION OF ONE BOUND ON X1

** REGULAR UPPER PRIMAL **

** VARIABLE LIST **

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**NODE 1 - ADDITION OF ONE Bound ON X1**

**SOLUTION**

- **Objective**: MAXIMIZE ROW#1
- **Z**: \( \frac{13}{3} \)
- **Iterations**: 3

**Variable Section**

<table>
<thead>
<tr>
<th>Name</th>
<th>Activity Level</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Reduced Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>( \frac{2}{3} )</td>
</tr>
<tr>
<td>X2</td>
<td>( \frac{7}{3} )</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>SLK#3</td>
<td>( \frac{2}{3} )</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

**Constraint Section**

<table>
<thead>
<tr>
<th>Name</th>
<th>Slack Activity</th>
<th>Dual Price</th>
<th>RHS Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0</td>
<td>( \frac{1}{3} )</td>
<td>( \frac{7}{3} )</td>
</tr>
<tr>
<td>ROW#3</td>
<td>( \frac{2}{3} )</td>
<td>0</td>
<td>( \frac{2}{3} )</td>
</tr>
</tbody>
</table>

**AND**: X2 <= 2

**Title**: NODE 2 - UPPER BOUNDS ON X1, X2

**Use Rational Upper Primal**

**No Unrestricted Variables in the Model**

**Variable List**

<table>
<thead>
<tr>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
</tr>
<tr>
<td>X2</td>
</tr>
</tbody>
</table>

**NODE 2 - UPPER BOUNDS ON X1, X2**

**SOLUTION**

- **Objective**: MAXIMIZE ROW#1
- **Z**: 4
- **Iterations**: 3

**Variable Section**

<table>
<thead>
<tr>
<th>Name</th>
<th>Activity Level</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Reduced Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>X2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SLK#2</td>
<td>1</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>SLK#3</td>
<td>1</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>
CONSTRAINT SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ROW#3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

? LIST
ROW#1
  MAX: X1+X2
ROW#2
  ST: X1+3X2 <= 9
ROW#3
  AND: 3X1+X2 <= 9
ROW#4
  AND: ALL VARS >= 0
ROW#5
  AND: X1<=2
ROW#6
  AND: X2 <= 2
? CHANGE ROW#6 "<= 2" => 3
  AND: X2 >= 3
? TITLE NODE 3 - LOWER BOUND ON X2, UPPER BOUND ON X1
? USE RATIONAL UPPER PRIMAL

** No unrestricted variables in the model **

** VARIABLE LIST **

| X1  | X2 |

NODE 3 - LOWER BOUND ON X2, UPPER BOUND ON X1

SOLUTION

OBJECT MAXIMIZE ROW#1
Z
ITERATIONS

VARIABLE SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>3</td>
<td>3</td>
<td>POS INF</td>
<td>2</td>
</tr>
<tr>
<td>SLK#3</td>
<td>6</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

CONSTRAINT SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ROW#3</td>
<td>6</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>
** DELETE ROW46
** LIST ROW45
ROW45
AND: X1<=2
** CHANGE ROW45 '<=2''=>3'
AND: X1>=3
** TITLE NODE 4 - LOWER BOUND ON X1 (FINAL NODE)
** USE RATIONAL UPPER PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

X1    X2

NODE 4 - LOWER BOUND ON X1 (FINAL NODE)

SOLUTION

OBJECT MAXIMIZE ROW1

Z

3

ITERATIONS 3

VARIABLE SECTION

NAME ACTIVITY LEVEL LOWER BOUND UPPER BOUND REDUCED COST

X1 3 3 POS INF 2

SLK42 6 0 POS INF 0

CONSTRAINT SECTION

NAME SLACK ACTIVITY DUAL PRICE RHS VALUE

ROW42 6 0 6

ROW#3 0 1 0

? LIST
ROW1 MAX: X1+X2

ROW2 ST: X1+3X2 <= 9

ROW3 AND: 3X1+X2 <= 9

ROW4 AND: ALL VARS >= 0

ROW5 AND: X1>=3

? END

1.152 CP SECONDS EXECUTION TIME
APPENDIX A

A SIMPLIFIED USER'S MANUAL FOR EZLP

The first three pages of this appendix serve as a reasonable handout for most beginning users. The remaining nine pages provide more detailed discussion of the topics covered in the first three pages. By employing the "HELP" command the user can obtain any or all of this information during execution of EZLP.
EZLP: AN INTERACTIVE COMPUTER PROGRAM DESIGNED TO SOLVE
STUDENT-ORIENTED LINEAR PROGRAMMING PROBLEMS.†

1: General

The first word of each line is a keyword, and indicates the function of the input line. The keywords are "MIN" or "MAX" for the objective function, "ST" for the first constraint, followed by "AND" for each additional constraint. Each name (e.g. variable name) consists generally of a combination of up to 8 alphabetic and numeric characters with the first character being an alphabetic.

Model Entry

Model entry statements are composed of a keyword, an optional name, a mandatory colon, and either an objective function or a constraint.

Example of a model entry:

```
MAX: 2PROD1 + 3PROD2 - 4 COST
ST: PROD1 + PROD2 <= 150
AND: COST >= 5
AND: ALL OTHER VARS >= 0
```

Editing

The input model can be edited by "DELETE" and "CHANGE" statements.

Adding a constraint is accomplished by simply typing the constraint.

Examples:

```
AND NEWROW: 5COST-PROD1>=20.23
CHANGE NEWROW "PROD1"PROD2"
DELETE NEWROW
```

†EZLP was developed in the School of Industrial and Systems Engineering, Georgia Tech, Atlanta, Georgia 30332 under NSF grant #SED75-17476. John J. Jarvis, project director; Frank H. Cullen and Chris Papaconstadopoulos, research assistants.
EZLP internally numbers the input model lines and if the optional name before the colon is omitted during model entry, a name of the form "ROW#n" (e.g., ROW#6) is assigned as a default.

The model can be listed either in total or one line at a time as follows:

LIST (THE ENTIRE MODEL)
LIST CONSTR2 (LINE"CONSTR2" ONLY)

Continuation of the input line is achieved by placing an ampersand (&) after a complete name. Example:

AND: HEAT - 2.34WOOD & -6.45POWER>=16.4

Specifying Output

EZLP always outputs the final optimal solution. Other output can be requested after every iteration or after every n iterations. Examples:

PRINT BASIS 5 (The names of the basic variables every fifth iteration)
PRINT TABLEAU, 2 (The tableau at every other iteration)
PRINT VARS (The value of primal and dual variables)
PRINT NONE (Resets print specs to only final solution)
PRINT INITIAL, FINAL TABLEAU
PRINT ALL (Prints everything)

Specifying the Method of Solution

After model entry, editing, and output specification, the use statement triggers the optimization. Examples:

USE PRIMAL
(Ordinary simplex method)
USE RATIONAL PRIMAL
(Ordinary simplex method with rational arithmetic)
USE UPPER DUAL
(Lower-upper bounded dual simplex method)
USE RATIONAL UPPER PRIMAL
(Lower-upper bounded simplex method with rational arithmetic)

Upon completion of the solution attempt, EZLP returns to the editing phase.
Restarting and Stopping EZLP

If another model is to be entered from scratch, type "RESTART", and EZLP will re-initialize the model file area. If EZLP is to be terminated, type "END".

File-Handling Capabilities

EZLP has the ability to use external mass storage files for both input and output. Information on these options can be obtained by consulting Section 5 "FILE-HANDLING".

Simple Example of a Complete EZLP Run

MAX: 2X1+3X2+4.5X3
ST: X1+X2 <=75.3
AND: X2 +X3 <=45
AND: ALL VARS >=0
USE PRIMAL
END

The Help Command

During execution EZLP permits the user to obtain specific information concerning its use. The user may obtain a short introduction to EZLP and its use by typing

HELP

If the user desires he may type "HELP<keyword> ", (e.g., HELP MODEL) where the acceptable keywords are:

<table>
<thead>
<tr>
<th>General keywords</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EDIT</td>
<td>EXAMPLES</td>
<td>FILES</td>
<td>KEYWORDS</td>
</tr>
<tr>
<td>MODEL</td>
<td>SOLVING</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specific keywords</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>ADVANCED</td>
<td>CHANGE</td>
<td>CONT</td>
</tr>
<tr>
<td>CONS</td>
<td>DELETE</td>
<td>OBJ</td>
<td>PRINT</td>
</tr>
<tr>
<td>RUN</td>
<td>SAVE</td>
<td>TITLE</td>
<td>USE</td>
</tr>
<tr>
<td>ALTOBJ</td>
<td>NAMES</td>
<td>LIST</td>
<td></td>
</tr>
</tbody>
</table>
2: The Model

Providing a Title for the Model

EZLP permits the input of a title for the model. This title will appear just before the optimal solution is printed. The title command consists of:

1. The keyword "TITLE"
2. The title text

Example:

TITLE THIS IS MODEL 4

Variable and Constraint Names

In entering the model, the user has the ability of assigning his own names to the variables and the constraints. These names must be from one to eight characters long, begin with an alpha character, and contain no special characters. A variable name may contain a comma provided that the comma separates two numerics. Spaces are not allowed within names. EZLP generates constraint names for unnamed rows and variable names for slack and artificial variables. These take the form:

ROW#n - for generated constraint names
SLK#n - for generated slack variable names
ART#n - for generated artificial variable names

Continuation of Lines

In the event that an input line exceeds 80 characters, continuation is accomplished by placing an ampersand (&) after a complete name and resuming input on the following line.

Objective Functions

EZLP allows for the specification of two kinds of objective functions: primary and alternate. For the simple model there usually will not be an alternate objective function. In essence, this concept allows the user to enter more than one objective function and optimize the various objective functions subject to the same constraints during the
course of the interactive session.

Primary objective function - The entry of the objective function consists of:

1. The keywords "MAX", "MIN"
2. An optional objective function name
3. A mandatory colon
4. And a linear combination of user-defined variable names.

If the optional objective function is omitted, EZLP will generate a name of the form ROW#n for use in editing and output. Examples:

- MIN: 2X1 + 2X2 + 3X3
- MAX MYOBJ: 3.54 COLUMN1-4.543COLUMN2 + 6.25
- MINIMIZE: 5.76 X1,2 - 4.33 X1,3 + 5.45 X2,1 & - 6.43 X2,3

Alternate Objective Functions - Alternate objective functions are entered in the same way as the primary objective function, except that the keyword "MIN" or "MAX" is replaced by the keyword "ALSO". The use of and alternate objective function is discussed in Section 4 "Solving the Model". Examples:

- ALSO: 2X1 - 3X2 - 4X3
- ALSO OBJ2: 3.45 COLUMN1 - 4.511COLUMN2

Constraints

Constraints are divided into three generic types.

1. Simple arithmetic constraints
2. Range constraints
3. List constraints

Each is discussed below.

Simple Arithmetic Constraints

A simple arithmetic constraint consists of

1. The keyword "AND" or "ST"
2. An optional constraint name
3. A mandatory colon
4. A linear combination of user-defined variable names
5. A relational operator (=,<=,>=)
6. A second linear combination of user-defined variable names.
It is not allowable to have the same variable name in both 4 and 6 above.

Examples:

```
AND: X1+X2 =7
AND MYCSTR: 5X1 + 8.3 X2 >= X3 +4
```

Range Constraints

EZLP allows the entry of bounded arithmetic expressions (range constraints). Range constraints consist of the following:

1. The keyword "AND" or "ST"
2. An optional user-defined constraint name
3. A mandatory colon
4. A constant (constant-1)
5. An inequality relational operator (>= or <=)
6. A linear combination of the user-assigned
7. An inequality operator (>= or <=)
8. A constant (constant-2)

The restriction on range constraints are that the two inequality operators must be identical and that the two constants must be consistent. (i.e. if the inequality operators are =, constant-1 must be <= constant-2).

```
AND: 4 <= 3X1+4.3X2<= 6.2
AND BOUND1: 7.2>= 3X2+HEAT - 5POWER>= 5
```

List Constraints

EZLP allows bounds for collective groups (or lists) of user-defined variable names to be specified in list constraints. The variable lists can be either explicit or implicit.

Explicit List Constraints

(These constraints consist of:)

1. The keyword "AND"
2. An optional constraint name
3. A mandatory colon
4. A list of variable names
5. A relational operator
6. A constant
The variable list is simply a list of variable names separated by commas. If the "UPPER" option is not specified in the "USE" statement, those list constraints in which the constant is not 0 are considered as explicit rows in the simplex tableau. Examples:

AND: X1, X2, X5 >=0
AND UBND: HEAT, POWER, LIGHT <=100.3.
AND: MYVAR, YOURVAR URS
(Here URS = unrestricted in sign and takes the place of 5 and 6 above.)

Bounded lists are also permitted, for example

AND: 4 <= X1, X2, X5 <= 7

Implicit List Constraints

(Special abbreviations are available to describe either all variables or variables not appearing in another list constraint. Examples:)

AND: ALL VARS >= 0
AND: ALL VARS <= 1
AND MYBOUND: ALL OTHER VARS >= 3

The phrase "ALL VARS" is a synonym for "all variables", while the phrase "ALL OTHER VARS" refers only to those variables not included in some earlier list constraint.

3: Editing the Model

Adding Rows

New rows may be either appended to the current model file or inserted within the current model file.

a. Appending rows to the end of the model. This is easily accomplished by simply typing in the row exactly as it was done during model entry.
Example:

AND: X1+X2 <= 7
ALSO: 3.5X1+4.6FLOW1

b. Insertion of rows within the model file - Rows may be inserted within the model file by simply typing in:
INSERT AFTER <old-row> <model entry statement>

or

INSERT BEFORE <old-row> <model entry statement>

For example, if the current model file contained:

MIN OBJ: 2X1+3X2
ST CON: X1+X2<=3
AND: ALL VARS >=0

an entry of

INSERT AFTER CON AND CON2: X1+2X2 >=1

would result in a model file of

MIN OBJ: 2X1+3X2
ST CON: X1+X2<=3
AND CON2: X1+2X2 >=1
AND: ALL VARS >=0

Deleting Rows

The deletion of a row is accomplished by entering DELETE <row name>

where <row name> is the user-assigned name, or in the absence of such, takes the form "ROW#n". If a delete is followed by the addition of a row having the same name, the new row is inserted in the model file in the same place. Examples:

DELETE MYCSTRNT
DELETE ROW#7

Changing Rows

To change a character string in a specified row (say ROW#5), the following is entered

CHANGE ROW#5 "<string1>"<string2>"

The double quote (""') is the only allowable delimiter. For example if ROW#5 was originally

AND: 2X1+3X3<=5
an entry of

\[
\text{CHANGE ROW}\#5 \ "3"x2+4"
\]

would result in \(\text{ROW}\#5\) becoming

\[
\text{AND: } 2x_1+x_2+4x_3\leq 5
\]

whereupon an entry of

\[
\text{CHANGE ROW}\#5 \ "3<"\]

would yield \(\text{ROW}\#5\) as

\[
\text{AND: } 2x_1+x_2+4x=5
\]

Changes involving row names and/or continuations are illegal.

Changes of this type should be accomplished by a "DELETE" followed by retyping the entire row.

**Listing the Model**

The list command allows the user to list the current contents of the model file in total or in part. Two options are available:

1. LIST
   
   (lists the whole model)

2. LIST <row name>
   
   (lists only row <row name>)

Examples:

LIST
LIST ROW\#5
LIST CONSTR3

**4: Solving the Model**

**Specifying Alternate Objective Functions**

The user may specify an alternate objective function to be optimized by typing:

\[
\text{ALTOBJ MIN } \text{<row name>}
\]

or

\[
\text{ALTOBJ MAX } \text{<row name>}
\]

where <row name> is the name of an objective function in the model.
Examples:

    ALTOBJ MAX ROW#2
    ALTOBJ MIN OBJ4

This objective function specification remains in force until such time
as a "RESTART" or another "ALTOBJ" command is entered.

The user may identify the current specified objective function
row name by typing:

    ALTOBJ STATUS

Specifying Output

The PRINT command is used to specify the type and frequency of out-
put to be printed during the optimization process. Parameters which
can be used are:

1. **VARS** prints the values of the non-zero primal
   and dual variables in the current tableau.
2. **BASIS** prints the current basic variables and
   denotes entry and exit activities
3. **TABLEAU** prints the current tableau and the updated
   objective function row
4. **ALL** Synonym for "VARS,BASIS,TABLEAU"

Parameters 1-4 can optionally be followed by an integer indicating the
frequency with which the output is to be given. If omitted, the default
value is 1 (output after every iteration). Examples:

    PRINT TABLEAU, BASIS
    PRINT VARS 5
    PRINT ALL 4

Initial and/or final tableau output can also be indicated. In this case,
the integer described does not apply and is ignored if present. Examples:

    PRINT INITIAL TABLEAU
    PRINT FINAL ALL
    PRINT INITIAL, FINAL BASIS TABLEAU
If print specifications are to be altered, the proper action is to enter

```
PRINT NONE
```

which resets print parameters to the default mode.

**Specifying the Method of Solution**

The `USE` command is entered when an attempt to solve the input model is to be made. The following parameters are legal (the order of the parameters is important).

1. `RATIONAL` - (Optional) Rational arithmetic is to be used. Rational arithmetic is 3 to 4 times slower than real arithmetic, and can only solve problems half the size.

2. `UPPER` - (Optional) Lower-upper bounded simplex is to be used. List constraints are implicit in the tableau.

3. `PRIMAL` or `DUAL` - A specification of the type simplex method to be used. "PRIMAL" refers to the simplex method and "DUAL" refers to the dual simplex method.

**Examples:**

```
USE RATIONAL PRIMAL
USE DUAL
USE UPPER DUAL
USE RATIONAL UPPER PRIMAL
```

When possible, "UPPER" should be specified for dual simplex, as this improves the chance of obtaining a starting dual feasible basis.

**5: File-Handling**

**Input From an External File**

The "RUN" command enables the user to use an external file as a store of input statements and commands. Upon entering the command `RUN <file name>`, EZLP opens the local file `<file name>` and references this file for all subsequent input commands, until such time as an "END" statement is encountered or an End-of-file condition on `<file name>` exists.
If an End-of-file is encountered, control returns to the live user. The form of the file should be 80 characters per line.

Output to an External File

The "SAVE" command enables the user to place the current concepts of the model file into a local file. The form of this file is 80 characters per line and editable by the system editor or usable as input to a later execution of EZLP. ENTER:

SAVE <file name>

6: Batch Processing

EZLP can be used in a batch environment in much the same way that it is used interactively. The sole restriction is that the command BATCH must be the first entry in the input deck to EZLP. Omission of this command can result in an infinite print loop at job termination time. Also the command END must be the last entry in the input deck to EZLP.

7: Advanced Example

TITLE ** ADVANCED MODEL **
MAX OBJ: 2INTERST1+3.24INTERST2 5CAPITAL
ALSO OBJ2: 3.4INTERST1+2.54INTERST2-5 CAPITAL
ST CONST: INTERST1-4INTERST2 = 108
AND MNCAP: CAPITAL >= 4.6 INTERST1 + 5.7INTERST2
AND MNCAP: 4 CAPITAL <= 10345.65 - 3.224 INTERST1 & -5.231 INTERST2
AND CAPBND: 800 <= CAPITAL <= 10000
AND INTBND: INTERST1, INTERST2 <=10825
AND NONNEG: INTERST1, INTERST2 >= 0
CHANGE MNCAP "4.6" "4.76"
CHANGE INTBND "825" "825"
PRINT ALL
USE UPPER PRIMAL
DELETE NONNEG
AND NEWCSTR: INTERST2 >= 1.342
AND UBNBD: INTERST1, INTERST2,CAPITAL <= 100000.3
ALT OBJ MAX OBJ2
PRINT FINAL TABLEAU, BASIS, VARS LIST USE UPPER DUAL
END
APPENDIX B

EZLP PROGRAM LISTING
E-Z-LP - INTERACTIVE LINEAR PROGRAMMING

E-Z-LP IS AN INTERACTIVE LINEAR PROGRAMMING PACKAGE DEVELOPED UNDER THE SPONSORSHIP OF THE NATIONAL SCIENCE FOUNDATION.

PARENT INSTALLATION GEORGIA INSTITUTE OF TECHNOLOGY SCHOOL OF IND. & SYS. ENGINEERING ATLANTA, GA. 30332

PROJECT SUPERVISOR DR. JOHN J. JARVIS

PROGRAMMERS FRANK CULLEN
CHRIS PAPACONSTADOPULOS

DATE RELEASED SUMMER 1976

THIS IS THE MAIN (OR DRIVER) PROGRAM OF E-Z-LP. PROGRAM EXECUTION IS BEGUN HERE, AND THE VARIOUS ROUTINES ARE EXECUTED ACCORDING TO KEYWORDS CONTAINED IN THE INPUT STATEMENTS.

PROGRAM EZLP(EZRUN, EZSAVE, INPUT, OUTPUT, EZWORK, EZHELP,
* TAPE5=INPUT,TAPE6=OUTPUT,TAPE7=TAPE8=EZWORK,
* TAPE9=EZRUN,TAPE10=EZSAVE,TAPE11=EZHELP)

*** COMMON AREAS

EDIT COMMON AREA FOR THE EDITOR SUBPROGRAM

OPT COMMON AREA FOR THE OPTIMIZATION AND OUTPUT SUBPROGRAM

KEYS USED ONLY FOR CDC RANDOM ACCESS STORAGE

WORK USED FOR WORK AREAS BY ALL SUBPROGRAMS

LEX COMMON AREA FOR THE SYNTACTICAL ANALYSIS SUBPROGRAM

COMMON /EDIT/ MAXROW,FSTAVL,HIADD,EDTNME,MSFILE,INFILN,
* LINKS

COMMON /OPT/ COEF(5000),RHS(200),BOUND(400),BASIS(200),
* OBJFCN(200),CNTNME(200),VARNME(200),
* PRNTSW(5),ITRLMT,QUARTS,METH0,MHINMAX,Z,OBJROW,
* NULL(200),DIMROW,DIMCOL,TITLE(83)

COMMON /KEYS/ MSICX(500)

COMMON /MCRK/ JUNK(1200)

COMMON /AREA1/ INFPOS,INFNEG,INFZRO

COMMON /LEX/ IWRONG,MSDLK(200),ICNTRL(50),ITRMNL(10),
* IANUM(10),MNFLAG,ALT0BJ

DIMENSION BUFFER(P1),LINKS(200),ECTNME(200),WORK(8)

INTEGER HIADD,FSTAVL,CURADR

REAL INFZRO,INFPOS,INFNEG

LOGICAL BATCH,PRNTSW,MODEL

*** ALPHA VARIABLE NAMES FOR THIS PART OF THE CODE ARE:
** (100) FORMAT STATEMENTS

100 FORMAT(" EZLP - VERSION 9/1/76", " TYPE HELP IF YOU", " HAVE QUESTIONS, OTHERWISE PROCEED")

102 FORMAT(80A1)

104 FORMAT(" UNRECOGNIZED VERB "", A8, " COMMAND IGNORED")

106 FORMAT(" ROW NAME MISSING IN MODEL ENTRY STATEMENT")

108 FORMAT(" CHANGE WHAT?")

110 FORMAT(" DELETE WHAT?")

112 FORMAT(" MISSING COLON AFTER ROW NAME "", A8, " RE-ENTER")

114 FORMAT(" MISSING KEYWORD EXPECTED BEFORE "", A8, " RE-ENTER")

116 FORMAT(" PRINT WHAT?")

118 FORMAT(" PRIMAL OR DUAL SPECIFICATION MISSING IN USE STATEMENT")

120 FORMAT(" "", A8, " INVALID PARAMETER IN PRINT STATEMENT")

122 FORMAT(" RESTART EFFECTED")

124 FORMAT(" ENTER MODEL")

126 FORMAT(" "", A8, " ILLEGAL OR CUT OF ORDER IN USE STATEMENT")

128 FORMAT(" "", A8, " NOT STATUS, MAX, MIN, OR NONE")

130 FORMAT(2A8, 6A1)

132 FORMAT(IX, 64A1)

134 FORMAT(" "", A8, " IS NOT AN ACCEPTABLE HELP KEYWORD")

136 FORMAT(" OBJECTIVE FCN NAME OMITTED IN ALTOBJ STATEMENT")

138 FORMAT(" BEFORE OR AFTER NOT SPECIFIED IN INSERT STATEMENT")

140 FORMAT(" MISSING ROW NAME IN INSERT STATEMENT")

142 FORMAT(" "", A8, " NOT IN TABLE - INSERT IGNORED")

144 FORMAT(" FILE NAME MISSING IN RUN STATEMENT")

146 FORMAT(" FILE NAME MISSING IN SAVE STATEMENT")

148 FORMAT(" NO ALTOBJ IS CURRENTLY ACTIVE")

150 FORMAT(" CURRENT ALTOBJ IS MAX "", A8)

152 FORMAT(" CURRENT ALTOBJ IS MIN "", A8)

** (200) INITIALIZATION

INFZRO=1.0E-8

CONTINUE

DO 210 I=1,80

210 TITLE(I)=" "

WRITE(6,100)

BATCH=.FALSE.

INFILE=5

MSFILE=8

MAXROW=MIAADD=MSAVL=0

CALL OPENMS(MSFILE, MSIDX, 499, 0)

MODEL=.TRUE.

ALTOBJ=" 

MMFLAG=0

PRNTSW(1)=PRNTSW(2)=PRNTSW(3)=.FALSE.

GO TO 300

290 CONTINUE

** (300) READ IN GENERAL STATEMENT

CONTINUE

INSERT=0

READ(INFILE, 102) (BUFFER(I), I=1,80)

IF(EOF(INFILE).NE.0) GO TO 320

IDXBUF=1

CALL GETWRD(BUFFER, IDXBUF, WORD, RESULT, COLON)

IF(RESULT.NE.0) GO TO 300
IF(WORD.EQ."MODEL".OR.WORD.EQ."TITLE") GO TO 400
IF(WORD.EQ."CHANGE".OR.WORD.EQ."DELETE")
* GO TO 500
IF(WOROEQ."LIST") GO TO 650
IF(WOROEQ."ALTOBJ") GO TO 450
IF(WOROEQ."PRINT") GO TO 600
IF(WOROEQ."HELP") GO TO 800
IF(WOROEQ."RUN") GO TO 900
IF(WOROEQ."SAVE") GO TO 1000
IF(WOROEQ."RESTART") GO TO 1050
IF(WOROEQ."BATCH") GO TO 1100
IF(WOROEQ."END") GO TO 1150
IF(WOROEQ."INSERT") GO TO 1400
IF(MODEL) GO TO 500
WRITE(6,104) WORD
GO TO 300
320 IF(INFILE.NE.5) GO TO 340
IF(BATCH) STOP
340 INFILE=5
GO TO 300
C
C*** (400) MODEL STATEMENT
C
400 CONTINUE
DO 410 I=1,6,80
410 TITLE(I-5)=BUFFER(I)
GO TO 300
C
C*** (450) ALT Obj - ALTERNATE OBJECTIVE FUNCTION
C
450 CONTINUE
CALL GETWRO BUFFER,IOBUF,WORD,RESULT,COLON
IF(WORD.EQ."STATUS") GO TO 480
MMFLAG=0
ALT OBJ=" "
IF(RESULT.NE.0.0P.WORD.EQ."NONE") GO TO 300
IF(WOROEQ."MIN".OR.WOROEQ."MINIMIZE") MMFLAG=-1
IF(WOROEQ."MAX".OR.WOROEQ."MAXIMIZE") MMFLAG=1
IF(MMFLAG.NE.0) GC TO 460
WRITE(6,128) WORD
GO TO 300
460 CALL GETPD BUFFER,IOBUF,WORD,RESULT,COLON
IF(RESULT.EQ.0) GO TO 470
WRITE(6,136)
GO TO 300
470 ALT OBJ=WORD
GO TO 300
480 IF(ALT OBJ.NE." "") GO TO 490
WRITE(6,148)
GO TO 300
490 IF(MMFLAG.EQ.-1) WRITE(6,152) ALT OBJ
IF(MMFLAG.EQ.1) WRITE(6,150) ALT OBJ
GO TO 300
C
C*** (500) EDITOR COMMANDS
C
500 CONTINUE
IF(WOROEQ."CHANGE") GO TO 560
IF(WOROEQ."DELETE") GO TO 580
IF(MODEL) GO TO 530
C ADD STATEMENT TO FILE
505 IF(COLON.EQ.1) GO TO 540
CALL GETWRO BUFFER,IOBUF,WORD,RESULT,COLON
IF (RESULT .EQ. 0) GO TO 510
WRITE (6, 106)
GO TO 300

510 IF (COLON .EQ. 1) GO TO 520
WRITE (6, 112) WORD
GO TO 300

520 CONTINUE
CALL EDITOR (BUFFER, WORD, "ADD")
IF (INSERT .NE. 0) GO TO 1470
GO TO 300

530 IF (WORD .EQ. "ST", OR. WORD .EQ. "AND", OR. WORD .EQ. "ALSO"
* OR. WORD .EQ. "MIN", OR. WORD .EQ. "MAX"
* OR. WORD .EQ. "MINIMIZE", OR. WORD .EQ. "MAXIMIZE") GO TO 505
WRITE (6, 114) WORD
GO TO 300

540 ITEM = MAXROW + 1
WORK (1) = "R"
WORK (2) = "0"
WORK (3) = "W"
WORK (4) = "#"
WORK (5) = WORK (6) = WORK (7) = WORK (8) = " "
IMAX = 1
IF (ITEM .GT. 9) IMAX = 2
IF (ITEM .GT. 99) IMAX = 3
IF (ITEM .GT. 999) IMAX = 4
DO 550 I = 1, IMAX
WORK (5 + IMAX - I) = DSPNUM (ITEM - (ITEM / 10) * 10)
550 ITEM = ITEM / 10
CALL PACK (WORK, WORD, 8)
GO TO 520

560 CONTINUE
CHANGE STATEMENT
CALL GETWORD (BUFFER, IDXBUF, WORD, RESULT, COLON)
IF (RESULT .EQ. 0) GO TO 570
WRITE (6, 109)
GO TO 300

570 CALL EDITOR (BUFFER, WORD, "CHANGE")
GO TO 300

580 CONTINUE
DELETE STATEMENT
CALL GETWORD (BUFFER, IDXBUF, WORD, RESULT, COLON)
IF (RESULT .EQ. 0) GO TO 590
WRITE (6, 110)
GO TO 300

590 CALL EDITOR (BUFFER, WORD, "DELETE")
GO TO 300

C
C*** (600) PRINT STATEMENTS

C
600 CONTINUE
CALL GETWORD (BUFFER, IDXBUF, WORD, RESULT, COLON)
IF (RESULT .EQ. 0) GO TO 610
WRITE (6, 116)
GO TO 290

610 CONTINUE
IDXSW = 0
IF (WORD .EQ. "NONE") GO TO 660
IF (WORD .EQ. "ALL") GO TO 670
IF (WORD .EQ. "VARS") IDXSW = 1
IF (WORD .EQ. "BASIS") IDXSW = 2
IF (WORD .EQ. "TABLEAU") IDXSW = 3
IF (WORD .EQ. "INITIAL") IDXSW = 4
IF (WORD .EQ. "FINAL") IDXSW = 5
IF (IDXSW .NE. 0) GO TO 620
WRITE(6,120) WORD
GO TO 290
620 ITEMP=1
PRNTSW(IDXSW)=.TRUE.
IDXBUF=IDXBUF+1
625 IDXBUF=IDXBUF+1
IF(IDXBUF.GT.80) GO TO 640
IF(BUFFER(IDXBUF).EQ."") GO TO 625
IF(NUMGET(BUFFER(IDXBUF)).EQ.-1) GO TO 600
IDXBUF=IDXBUF-1
DO 630 I=1,10
NUM=NUMGET(BUFFER(IDXBUF+I))
IF(NUM.EQ.-1) GO TO 640
630 ITEMP=ITEMP+1
640 PRNTSW(IDXSW)=.TRUE.
ITRLMT=ITEMP
GO TO 290
650 CALL GETwRD(BUFFER,IDXBUF,WORD,RESULT,COLON)
IF(RESULT.NE.0) WORD=" 
IF(WORD.EQ."MODEL") WORD=" 
CALL EDITOR(BUFFER,WORD,"LIST")
GO TO 300
660 PRNTSW(1)=PRNTSW(2)=PRNTSW(3)=PRNTSW(5)=PRNTSW(4)=.FALSE.
GO TO 300
670 PRNTSW(1)=PRNTSW(2)=PRNTSW(3)=.TRUE.
GO TO 620
C
C*** (700) HELP COMMANDS
C
700 CONTINUE
CALL GETwRD(BUFFER,IDXBUF,WORD,RESULT,COLON)
IF(RESULT.NE.0) WORD=" 
REWIND 11
I=0
720 READ(11,130) WORD1,WORD2,(BUFFER(J),J=1,64)
IF(EOF(11).NE.0) GO TO 760
IF(WORD.NE.WORD1.AND.WORD.NE.WORD2) GO TO 720
DO 740 JMAX=1,64
IF(BUFFER(JMAX).NE." ") GO TO 750
740 CONTINUE
750 JMAX=65-JMAX
WRITE(6,132) (BUFFER(J),J=1,JMAX)
I=I+1
GO TO 720
760 IF(I.EQ.0) WRITE(6,134) WORD
GO TO 300
C
C*** (800) RUN COMMANDS
C
800 CONTINUE
CALL GETwRD(BUFFER,IDXBUF,WORD,RESULT,COLON)
IF(RESULT.EQ.0) GO TO 810
WRITE(6,144)
GO TO 300
810 CALL FCHG(9,WORD,RESULT)
INFILE=9
GO TO 300
C
C*** (900) SAVE COMMAND
C
900 CONTINUE
REWIND 10
CALL GETwRD(BUFFER,IDXBUF,WORD,RESULT,COLON)
IF(RESULT.EQ.0) GO TO 910
WRITE(6,146)
GO TO 300
910 CALL FCHG(10,WORD,RESULT)
   IF(MAXROW.LE.0) GO TO 290
   DO 350 I=1,MAXROW
      CURADR=LINKS(I)
   320 IF(CURADR.EQ.0) GO TO 930
      CALL MSREAD(CURADR,BUFFER,RESULT,MSFILE)
      WRITE(19,102) (BUFFER(J),J=1,80)
      CURADR=LINK
      GO TO 920
930 CONTINUE
REWIND 10
GO TO 290

C
C*** (1000) RESTART PROCEDURE
C
1000 CONTINUE
   WRITE(6,122)
   CALL CLCSMS(MSFILE)
   GO TO 200
C
C*** (1100) USE STATEMENT
C
1100 CONTINUE
   CALL GETHD(BUFFER,IDXBUF,WORD,RESULT,Colon)
   IF(RESULT.EQ.0) GO TO 1110
   WRITE(6,116)
   GO TO 290
1110 METHOD=1
   IF(WORD.NE."RATIONAL") GO TO 1120
   METHOD=METHOD+1
   CALL GETHD(BUFFER,IDXBUF,WORD,RESULT,Colon)
   IF(RESULT.NE.0) GO TO 1105
   1120 IF(WORD.NE."UPPER") GO TO 1130
   METHOD=METHOD+4
   CALL GETHD(BUFFER,IDXBUF,WORD,RESULT,Colon)
   IF(RESULT.NE.0) GO TO 1105
   1130 IF(WORD.NE."PRIMAL") GO TO 1140
   METHOD=METHOD+2
   CALL GETHD(BUFFER,IDXBUF,WORD,RESULT,Colon)
   IF(RESULT.EQ.0) GO TO 1150
   GO TO 1140
1140 IF(WORD.NE."DUAL") GO TO 1150
   CALL GETHD(BUFFER,IDXBUF,WORD,RESULT,Colon)
   IF(RESULT.EQ.0) GO TO 1150
   GO TO 1160
1150 WRITE(6,126) WORD
   GO TO 290
1160 CONTINUE
C
C*** CALL SYNT & PARSER
C
   CALL SYNT
   IF(IWRONG.NE.0) GO TO 290
C
C*** CALL TABLEAU BUILDER
C
   CALL BUILD
C
C*** CALL OPTIMIZE & OUTPUT
C
   IF(METHOD.EQ.2.OR.METHOD.EQ.4.OR.METHOD.EQ.6.OR.METHOD.EQ.8)
      CALL SPXRTL
IF(METHOD.EQ.1.0 .OR. METHOD.EQ.3.0 . OR. METHOD.EQ.5.0 . OR. METHOD.EQ.7.0)
*     CALL SPXREL
GO TO 290

C
C*** (1200) BATCH STATEMENT ***
C
1200 CONTINUE
BATCH=.TRUE.
GO TO 300
C
C*** (1300) END COMMAND
C
1300 STOP
C
C*** (1400) INSERT STATEMENT
C
1400 CONTINUE
CALL GETWRD(BUFFER,IDXBUF,WORD,RESULT,COLON)
BAFLAG=0
IF(WORD.EQ."BEFORE") BAFLAG=-1
IF(WORD.EQ."AFTER") BAFLAG=1
IF(BAFLAG.NE.0) GO TO 1410
WRITE(6,138)
GO TO 300
1410 CALL GETWRD(BUFFER,IDXBUF,WORD,RESULT,COLON)
IF(RESULT.EQ.0) GO TO 1420
WRITE(6,140)
GO TO 300
1420 DO 1430 I=1,MAXROW
1430 IF(WORD.EQ.EDTNME(I)) GO TO 1440
WRITE(6,142) WORD
GO TO 300
1440 INSERT=I
IF(BAFLAG.EQ.1) INSERT=INSERT+1
IMAX=81-IDXBUF
DO 1450 I=1,IMAX
1450 BUFFER(I)=BUFFER(1+IDXBUF-1)
IMAX=61-IMAX
DO 1460 I=1,IMAX
1460 BUFFER(81-I)=" 
IDXBUF=1
CALL GETWRD(BUFFER,IDXBUF,WORD,RESULT,COLON)
GO TO 530
1470 CONTINUE
IF(INSERT.EQ.MAXROW) GO TO 300
WORD=EDTNME(MAXROW)
ITEMP=LINKS(MAXROW)
IMAX=MAXROW-INSERT
DO 1480 I=1,IMAX
1480 EDTNME(MAXROW-I+1)=EDTNME(MAXROW-I)
LINKS(MAXROW-I+1)=LINKS(MAXROW-I)
LINKS(INSERT)=ITEMP
EDTNME(INSERT)=WORD
GO TO 300
END
C
C SUBROUTINE GETWRD
C
C THIS SUBROUTINE RETURNS THE NEXT WORD (AFTER THE POINTER
C IDXBUF) IN THE INPUT ARRAY BUFFER. THE FOLLOWING
C PARAMETRIC SWITCHES ARE SET:
C
C 290 300 1200 1300 1400 1410 1420 1430 1440 1450 1460 1470 1480
C
RESULT = 0  SUCCESSFUL COMPLETION
1  THERE IS NO NEXT WORD (THE REST OF BUFFER
IS EMPTY)
COLON = 1  IF THERE IS A COLON FOLLOWING THE NEXT
WORD, AND
0  OTHERWISE
ALPHA VARIABLES IN THIS SUBROUTINE ARE:
BUFFER, WORK, WORD

DIMENSION BUFFER(81), WORK(8)
COLON=0
RESULT=0
IOXTMP=IOXBUF
210 DO 220 IDXBUF=IOXTMP+70
220 IF (BUFFER(IDXBUF).NE." ") GO TO 240
RESULT=1
RETURN
240 IDXBUF=ICXBUF-1
DO 250 I=1,8
250 WORK(I)=" "; DO 270 I=1,8
IF (BUFFER(IDXBUF+I).EQ." ") OR .AND. BUFFER(IDXBUF+I).EQ." ")
*  GO TO 300
IF (BUFFER(IDXBUF+I).EQ." ") GO TO 295
WORK(I)=BUFFER(IDXBUF+I)
CONTINUE
270 DO 280 I=8,80
IF (IDXBUF+I.GE.80) GO TO 290
IF (BUFFER(IDXBUF+I).NE." ") AND .AND. BUFFER(IDXBUF+I).NE." ")
*  GO TO 300
280 CONTINUE
290 RESULT=1
RETURN
295 COLON=1
GO TO 320

PACK THE WORD AND GO
CONTINUE
DO 310 J=1,80
K=IDXBUF+I+J
IF (K.GE.80) GO TO 320
IF (BUFFER(K).NE." ") COLON=1
IF (BUFFER(K).NE." ") GO TO 320
CONTINUE
310 CONTINUE
320 CONTINUE
IDXBUF=IDXBUF+I+1
CALL PACK(WORK,WORD,8)
RETURN
END
FUNCTION CSPNUM(N)
FUNCTION DSPNUM
THIS FUNCTION CONVERTS A NUMERIC DIGIT INTO AN ALPHANUMERIC
AND HENCE DISPLAYABLE IN A1 FORMAT) ONE.
OSPNUM="0"
IF(N.EQ.1) OSPNUM="1"
IF(N.EQ.2) OSPNUM="2"
IF(N.EQ.3) OSPNUM="3"
IF(N.EQ.4) OSPNUM="4"
IF(N.EQ.5) OSPNUM="5"
IF(N.EQ.6) OSPNUM="6"
IF(N.EQ.7) OSPNUM="7"
IF(N.EQ.8) OSPNUM="8"
IF(N.EQ.9) OSPNUM="9"
RETURN
END

FUNCTION NUMGET(X)
C**********************************************************************
C FUNCTION NUMGET
C
C THIS FUNCTION CONVERTS AN ALPHANUMERIC DIGIT INTO A NUMERIC
C INTEGER VALUE. THIS FUNCTION IS USED PRIMARILY IN THE
C DRIVER PROGRAM FOR THE PARSING OF DIGIT PARAMETERS.
C
C**********************************************************************

INTEGER X
NUMGET=1
IF(X.EQ."1") NUMGET=1
IF(X.EQ."2") NUMGET=2
IF(X.EQ."3") NUMGET=3
IF(X.EQ."4") NUMGET=4
IF(X.EQ."5") NUMGET=5
IF(X.EQ."6") NUMGET=6
IF(X.EQ."7") NUMGET=7
IF(X.EQ."8") NUMGET=8
IF(X.EQ."9") NUMGET=9
IF(X.EQ."0") NUMGET=0
RETURN
END

SUBROUTINE PACK(IN,RESULT1,M)
C**********************************************************************
C SUBROUTINE PACK
C
C THIS SUBROUTINE PACK AN 8-ELEMENT ARRAY (IN) CONTAINING ONLY
C SINGLE CHARACTER ALPHA VALUES INTO A 8-CHARACTER ALPHA
C VARIABLE (RESULT1). THIS PARTICULAR SUBROUTINE IS
C MACHINE-DEPENDENT AND MUST BE CHANGED FOR DIFFERENT MACHINES.
C
C**********************************************************************

1000 FORMAT(" IN WENT ",8A1," OUT CAME ",8A)
INTEGER RESULT1,RESULT2
DIMENSION IN(8)
DATA MSK1/770000000000000000008/
DATA MSK2/55000000000000000000B/
RESULT1=0
RESULT2=0
DO 100 I=1,M
IN(I)=AND(IN(I),MSK1)
RESULT1=OR(RESULT1,IN(I))
RESULT1=SHIFT(RESULT1,6)
100 CONTINUE
RESULT1=SHIFT(RESULT1,12)
MARGIN=10-M
DO 200 I=1,MARGIN
RESULT2=OR(RESULT2,MSK2)
RESULT2=SHIFT(RESULT2,6)
CONTINUE
RESULT1=OR(RESULT1,RESULT2)
WRITE(6,1000) (IN(K),K=1,6),RESULT1
RETURN
END

SUBROUTINE EDITOR(BUFFER, ROWNME, ACTION)

C
C SUBROUTINE EDITOR
C
C THIS IS THE MAIN SUBROUTINE OF THE EDITOR SUBPROGRAM.
C
C THE FILE MAINTENANCE OF E-Z-LP IS ACCOMPLISHED IN THIS
C SUBROUTINE AS WELL AS THE FUNCTIONS OF THE "LIST" COMMAND.
C
C THE VALUES OF THE PARAMETERS ARE:
C 1. BUFFER - THE 80-CHARACTER ARRAY READ IN FROM THE
   TERMINAL BY THE DRIVER PROGRAM.
C 2. ROWNME - THE ROW NAME OF THE ROW TO BE ADDED TO THE
   MODEL FILE OR TO WHICH EDITING ACTIONS
   ARE TO BE APPLIED.
C 3. ACTION - AN ALPHA VARIABLE HAVING THE FOLLOWING
   CODES:
   ADD - ADD BUFFER TO THE CURRENT MODEL
   FILE UNDER NAME ROWNME
   CHANGE - CHANGE THE CHARACTER STRING PRESENT
   IN BUFFER IN ROW ROWNME TO ANOTHER
   STRING FOUND IN BUFFER.
   DELETE - DELETE ROW ROWNME FROM THE MODEL
   FILE AND FROM THE POINTER ARRAY.
   LIST - LIST THE ROW ROWNME (INCLUDING
   CONTINUATION LINES) UPON THE TERMINAL
   IF ROWNME IS BLANK, LIST THE ENTIRE
   MODEL FILE.
C
C*****************************************************************************

COMMON /EDIT/ MAXROW, FSTAVL, HIACD, EDTNME, MSFILE, INFILE
   * , LINKS
COMMON /WCRK/ STRNG1(80), STRNG2(80), OUT(81)
INTEGER HIADD, FSTAVL, EDTNME, CURADR
DIMENSION BUFFER(81), LINKS(200), EDTNME(200)

C*** ALPHA VARIABLE NAMES IN THIS SUBROUTINE
C   BUFFER, ROWNME, ACTION, EDTNME, STRNG1, STRNG2, OUT
C
C*****************************************************************************

C*** (100) FORMAT STATEMENTS
C
100 FORMAT(80A1)
102 FORMAT(" ROW NAME ", A8, " ALREADY EXISTS - IGNORED")
104 FORMAT(" ROW NAME ", A8, " NOT IN TABLE - NOT DELETED")
106 FORMAT(" ROW NAME ", A8, " NOT IN TABLE - NOT CHANGED")
108 FORMAT(" LEADING QUOTE MISSING IN CHANGE STATEMENT")
110 FORMAT(" MIDDLE QUOTE MISSING IN CHANGE STATEMENT")
112 FORMAT(" TRAILING QUOTE MISSING IN CHANGE STATEMENT")
114 FORMAT(5X,75A1)
116 FORMAT(" SPECIFIED STRING NOT FOUND IN ", A8)
118 FORMAT(" ROW ", A8, " NOT IN MODEL FILE - LIST IGNORED")
120 FORMAT(" ", A8)
122 FORMAT(" CURRENT MODEL FILE EMPTY")
C
C*****************************************************************************

C*** (100) BRANCH TO PROPER ROUTINE
IF(ACTION.EQ."A" OR ACTION.EQ."ACC") GO TO 300
IF(ACTION.EQ."O" OR ACTION.EQ."CHANGE") GO TO 500
IF(ACTION.EQ."D" OR ACTION.EQ."DELETE") GO TO 400
IF(ACTION.EQ."L" OR ACTION.EQ."LIST") GO TO 700
RETURN

C*** (300) ADD A CONSTRAINT OR OBJECTIVE FUNCTION

300 CONTINUE
IF(MAXROW.EQ.0) GO TO 321
DO 320 I=1,MAXROW
320 IF(EOTNME(I).EQ.ROWNME) GO TO 380
MAXROW=MAXROW+1
EOTROW=MAXROW
EOTNME(MAXROW)=ROWNME
323 CONTINUE
DO 324 I=1,80
324 OUT(I)=BUFFTR(I)
LINKS(EOTROW)=0
LINK=0
325 CONTINUE
DO 330 I=1,80
330 IF(OUT(I).EQ."&") GO TO 350
NEWAOR=FSTAVL
NEWAVL=0
IF(FSTAVL.NE.0)
* CALL MSREAD(FSTAVL,BUFFER,NEWAVL,RESULT,MSFILE)
FSTAVL=NEWAVL
IF(NEWAOR.NE.0) GO TO 340
HIA00=HIA00+1
NEWAOR=HIA00
340 CALL MSWRITE(NEWAOR,OUT,0,RESULT,MSFILE)
IF(LINKS(EOTROW).EQ.0) LINKS(EOTROW)=NEWAOR
RETURN
350 NEWAOR=FSTAVL
NEWAVL=0
IF(FSTAVL.NE.0)
* CALL MSREAD(FSTAVL,BUFFER,NEWAVL,RESULT,MSFILE)
FSTAVL=NEWAVL
IF(NEWAOR.NE.0) GO TO 370
HIA00=HIA00+1
NEWAOR=HIA00
LINK=HIA00+1
370 CALL MSWRITE(NEWAOR,OUT,LINK,RESULT,MSFILE)
IF(LINKS(EOTROW).EQ.0) LINKS(EOTROW)=NEWAOR
READ(INFILE,100) (OUT(I),I=1,80)
GO TO 325
380 IF(LINKS(I).EQ.0) GO TO 390
WRITE(6,102) ROWNME
RETURN
390 EOTROW=I
GO TO 323

C*** (400) DELETE A ROW

400 CONTINUE
IF(MAXROW.EQ.0) GO TO 425
DO 420 I=1,MAXROW
420 IF(EOTNME(I).EQ.ROWNME) GO TO 430
425 WRITE(6,104) ROWNME
RETURN
430 CALL MSREAD(LINKS(I),OUT,LINK,RESULT,MSFILE)
IF(LINK.NE.0) GO TO 450
440 CALL MSREAD(LINK,CUT,LINKCT,RESULT,MSFILE)
IF(LINKCT.EQ.0) GO TO 450
LINK=LINKCT
GO TO 440
450 CALL MSWRIT(LINKS(I),OUT,FSTAVL,RESULT,MSFILE)
FSTAVL=LINK
LINKS(I)=0
RETURN

C
C*** (500) CHANGE A PHRASE
C
500 CONTINUE
IF(MAXROW.EQ.0) GO TO 525
DO 520 I=1,MAXROW
520 IF(EDTNPE(I).EQ.ROWNME) GO TO 530
525 WRITE(6,105) ROWNME
RETURN
530 CONTINUE
DO 540 J=1,80
540 IF(BUFFER(J).EQ.'"') GO TO 550
WRITE(6,108)
RETURN
550 LIM=80-J
DO 560 K=1,LIM
IF(BUFFER(J+K).EQ.'"') GO TO 570
560 STRNG1(K)=BUFFER(K+J)
WRITE(6,110)
RETURN
570 LIM=LIM-K
DO 580 L=1,LIM
IF(BUFFER(J+K+L).EQ.'"') GO TO 590
580 STRNG2(L)=BUFFER(J+K+L)
WRITE(6,112)
RETURN
590 LNGTH1=K-1
LNGTH2=L-1
CURADR=LINKS(I)
600 CALL MSREAD(CURADR,BUFFER,LINK,RESULT,MSFILE)
IF(RESULT.NE.0) RETURN
CALL CHANGE(BUFFER,STRNG1,STRNG2,LNGTH1,LNGTH2,RESULT,OUT)
IF(RESULT.NE.0) GO TO 620
DO 610 KMAX=1,70
610 IF(OUT(71-KMAX).NE."" ) GO TO 615
615 KMAX=71-KMAX
WRITE(6,114) (OUT(K),K=1,KMAX)
CALL MSWRIT(CURADR,OUT,LINK,RESULT,MSFILE)
RETURN
620 CONTINUE
IF(LINK.NE.0) GO TO 630
WRITE(6,116) ROWNME
RETURN
630 CURADR=LINK
GO TO 600

C
C*** (700) LIST STATEMENTS
C
700 CONTINUE
IF(ROWNME.EQ."" ) GO TO 750
DO 720 I=1,MAXROW
720 IF(EDTNME(I).EQ.ROWNME) GO TO 730
WRITE(6,118) ROWNME
RETURN
730 CURADR=LINKS(I)
IF(CURADR.EQ.0) RETURN
WRITE(6,120) ROWNME
SUBROUTINE CHANGE(BUFFER, STRNG1, STRNG2, LNGTH1, LNGTH2,
                RESULT, OUT)

* THIS SUBROUTINE CHANGE A CHARACTER STRING (STRNG1) OF LENGTH
  LNGTH1 TO A CHARACTER STRING (STRNG2) OF LENGTH LNGTH2 IN
  THE 80-CHARACTER ARRAY BUFFER AND STORES THE RESULT IN ARRAY OUT.

  RESULT = 0 SUCCESSFUL CHANGE
  1 STRNG1 NOT FOUND IN BUFFER

  DIMENSION BUFFER(80), STRNG1(80), STRNG2(80), OUT(80)
  INTEGER RESULT

  GO TO 10

  OUT(I) = " "
  RESULT = 0
  I = 1
  IF(LNGTH1 .EQ. 0) GO TO 25
  DO 80 I = 1, LNGTH1
  IF(LNGTH1 + I .GT. 80) GO TO 85
  DO 20 J = 1, LNGTH2

  I + J = 1
  OUT(I + J - 1) = STRNG1(J)
  20 CONTINUE

  25 CONTINUE
  IF(LNGTH2 .EQ. 0) GO TO 35
  DO 30 J = 1, LNGTH2
  IF(J + I .GT. 80) GO TO 90

  30 CONTINUE
  IF(I + LNGTH2 + J .GT. 80) GO TO 90
  OUT(I + LNGTH2 + J - 1) = STRNG2(J)
  35 CONTINUE
  IF(I + LNGTH2 + J .GT. 80) GO TO 90
  OUT(I + LNGTH2 + J .EQ. 0) = BUFFER(I + LNGTH1 + J - 1)

  END
SUBROUTINE MSREAD(NEWADR,BUFFER,LINK,RESULT,MSFILE)
C******************************************************************************
C
C SUBROUTINE MSREAD - MASS STORAGE READ MACRO
C
C THIS SUBROUTINE IS USED TO READ A SINGLE INPUT LINE FROM
C MASS STORAGE DEVICE MSFILE INTO THE 80-CHARACTER ARRAY
C BUFFER AT RELATIVE LOCATION NEWADR. A FORWARD LINK IS ALSO
C WRITTEN ONTO MSFILE AT ADDRESS NEWADR TO INDICATE THE PRESENCE
C OR ABSENCE OF A CONTINUATION. ABSENCE OF SUCH A CONTINUATION
C IS DENOTED BY LINK=0.
C
C THIS SUBROUTINE IS MACHINE-DEPENDENT AND MUST BE CUSTOMIZED
C TO THE PARTICULAR COMPUTER FOR WHICH E-Z-LP IS THE BE ADAPTED.
C******************************************************************************

COMMON /KEYS/ MSICX(500)
DIMENSION BUFFER(81)
INTEGER RESULT
RESULT=0
RESULT=0
CALL READMS(MSFILE,BUFFER,81,NEWADR)
LINK=BUFFER(81)
RETURN
END

SUBROUTINE MSWRIT(NEWADR,BUFFER,LINK,RESULT,MSFILE)
C******************************************************************************
C
C SUBROUTINE MSWRIT - MASS STORAGE WRITE MACRO
C
C THIS SUBROUTINE IS USED TO WRITE INFORMATION FROM
C THE 80-CHARACTER ARRAY BUFFER ONTO A MASS STORAGE
C DEVICE, INSERT A FORWARD LINK FOR CONTINUATION, AND
C RETURN A RESULT FLAG IN CASE OF AN ERROR.
C
C THIS SUBROUTINE IS MACHINE-DEPENDENT AND MUST BE
C CUSTOMIZED TO THE PARTICULAR COMPUTER TO WHICH E-Z-LP IS
C TO BE ADAPTED.
C******************************************************************************

COMMON /KEYS/ MSIDX(500)
DIMENSION BUFFER(81)
INTEGER RESULT
BUFFER(81)=LINK
RESULT=0
CALL WRITMS(MSFILE,BUFFER,81,NEWADR,0)
RETURN
END

SUBROUTINE BUILD
C******************************************************************************
C
C SUBROUTINE BUILD
C
C THIS SUBROUTINE IS THE TABLEAU-BUILDER, AND SERVES AS THE
C MIDDLEMAN BETWEEN THE SYNTACTICAL ANALYSIS SUBPROGRAM AND
C THE OPTIMIZATION AND OUTPUT SUBPROGRAM. THE QUASI-TRIANGULAR
C TABLEAU OUTPUT FROM THE SUBROUTINE SYNT IS EXAMINED HERE, AND
C AFTER THE REQUIRED SLACKS AND ARTIFICIALS ARE DETERMINED, A
C RECTANGULAR TABLEAU SUITABLE FOR USE BY THE SUBROUTINE
C SIMPLX IS GENERATED. ALSO GENERATED ARE VARIABLE NAMES FOR THE
C SLACK AND ARTIFICIAL VARIABLES AND WELL AS BOUNDS.

C******************************************************************************
C
COMMON /OPT/ COEF(5000), RHS(200), BOUND(400), BASIS(200),
* OBJFCN(200), CNTNME(200), WARNME(200),
* PRNISH(5), ITRLFMT, NGAARTS, METHOD, MINMAX, Z, OBJROW,
* NULL(200), DIMROW, DMCOL
COMMON /AREA1/ INFPOS, INFNEG, INFZRO
COMMON /LEX/ IWRONG, BNCSLK(200), ICONTRL(50), ITRMN(10),
* IANUP(10), MMFLAG, ALTCBJ
COMMON /WCRK/ DUALPT(200), SLKPTR(200), ARTCOF(200), SLKCOF(200),
* WORK(8)
DIMENSION TABLEAU(3000)
EQUIVALENCE (COEF(2001), TABLEAU(1))
INTEGER ARTCOF, SLKCOF, DIMCOL, DMCW 9 03JROW, BASIS, DUALPT, SLKPTR
REAL INFPOS, INFNEG, INFZRO
LOGICAL PRNTSW

C**** (200) COUNT NUMBER OF SLACKS AND ARTIFICIALS
C

NOARTS=0
NOSLKS=0
IDOLR="$"
IDX1=0
GO TO 290
I=1, DIMRCW
TEMP=RHS(I)
SLKPTR(I)=0
ARTCOF(I)=0
SLKCOF(I)=0
J=0
210  IDX1=IDX1+1
     J=J+1
     IF(TABLEAU(IDX1) .EQ. ICOLR) GO TO 230
     IF(BOUND(J*2-1) .EQ. INFNEG) GO TO 220
     TEMP=TEMP-BOUND(J*2-1)*TABLEAU(IDX1)
     GO TO 210
220  IF(BOUND(J*2) .EQ. INFPOS) GO TO 210
     TEMP=TEMP-BOUND(J*2)*TABLEAU(IDX1)
     GO TO 210
230  IF(NULL(I) .NE. 0) GO TO 290
     GO TO (290, 240, 250, 260) NULL(I)
240  IF(TEMP .EQ. 0) GO TO 270
     ARTCOF(I) = TEMP/ABS(TEMP)
     NOARTS=NOARTS+1
     GO TO 290
250  SLKCOF(I) = 1
     NOSLKS=NOSLKS+1
     IF(BNCSLK(I) .NE. INFPOS .AND. TEMP.GE. 0) GO TO 270
     IF(TEMP.GE. 0) GO TO 290
255  ARTCOF(I) = -1
     NOARTS=NOARTS+1
     GO TO 290
260  SLKCOF(I) = -1
     NOSLKS=NOSLKS+1
     IF(BNCSLK(I) .NE. INFPOS .AND. TEMP.LE. 0) GO TO 255
     IF(TEMP.LE. 0) GO TO 290
270  ARTCOF(I) = 1
     NOARTS=NOARTS+1
     GO TO 290
290  CONTINUE
IDX2=DIMCOL+NOSLKS
IDX3=IDX2+NOARTS
IDX1=0
C*** (300) PASS TABLEAU FOR USER VARIABLES

C
300  GO 340  I=1,DIMROW
     IDX4=0
310  IDX1=IDX1+1
     IF(TBLEAU(IDX1),E0,IDOLR) GO TO 320
     IDX4=IDX4+1
     COEF((I-1)*IDX3+IDX4)=TELEAU(IDX1)
     GO TO 310
320  IF(IDX4,GT,DIMCCL) GO TO 340
     IDX4=IDX4+1
     DO 330  J=IDX4,IDX2
     330     COEF((I-1)*IDX3+J)=0.0
     CONTINUE

C*** (340) INSERT SLACK VARIABLES INTO TABLEAU

C
340  CONTINUE
     IDX1=IDMCOL
     DO 360  I=1,DIMROW
     IF(SLKCOF(I),E0.0) GO TO 360
     IDX1=IDX1+1
     OBJFCN(IDX1)=0.0
     BOUND(2*IDX1-1)=0.0
     BOUND(2*IDX1)=INFPOS
     DO 350  J=1,DIMROW
     COEF((J-1)*IDX3+IDX1)=0.0
     350     IF(I,NE,J) COEF((J-1)*IDX3+IDX1)=SLKCOF(I)
     BASIS(I)=IDX1
     DUALPT(I)=IDX1*SLKCOF(I)
     SLKPTR(I)=IDX1
     WORK(1)="S"
     WORK(2)="L"
     WORK(3)="K"
     WORK(4)="#"
     WORK(5)=WORK(6)=WORK(7)=WORK(8)=""
     IMAX=1
     ITEMP=I
     IF(ITEMP,GT,9) IMAX=2
     IF(ITEMP,GT,99) IMAX=3
     IF(ITEMP,GT,999) IMAX=4
     DO 355  K=1,IMAX
     WORK(5+IMAX–K)=DSFNUM(ITEMP–(ITEMP/10)*10)
     355     ITEMP=ITEMP/10
     CALL PACK(WORK,VARNME(IDX1),8)
     CONTINUE

C*** (365) INSERT ARTIFICIAL VARIABLES INTO TABLEAU

C
360  CONTINUE
     IDX1=IDX2
     DO 380  I=1,DIMROW
     IF(ARTCOF(I),EQ.0) GO TO 380
     IDX1=IDX1+1
     OBJFCN(IDX1)=1
     BOUND(2*IDX1-1)=INFPOS
     BOUND(2*IDX1)=INFPOS
     DO 370  J=1,DIMROW
     COEF((J-1)*IDX3+IDX1)=0.0
     370     IF(I,EQ,J) COEF((J-1)*IDX3+IDX1)=ARTCOF(I)
     BASIS(I)=IDX1
     DUALPT(I)=IDX1*ARTCOF(I)
     ITEMP=I
     WORK(1)="A"
     WORK(2)="R"
WORK(3) = 'T'
WORK(4) = 'O'
WORK(5) = WORK(6) = WORK(7) = WORK(8) = ' '
IMAX = 1
IF (ITEMP.GT.9) IMAX = 2
IF (ITEMP.GT.99) IMAX = 3
IF (ITEMP.GT.999) IMAX = 4
DO 375 K = 1, IMAX
WORK(5 + IMAX - K) = 16SFNUM(ITEMP - (ITEMP/10)*10)
375 IMTEMP = IMTEMP/10
CALL PACK(WORK, VARNME(ICX1), 8)
380 CONTINUE
C
C ** (400) SET UP TO RETURN
C
400 CONTINUE
  DO 410 J = 1, IMCOL
410 OBJFCN(J) = 0.0
  DIMCOL = IDX3
C
C ** (500) RETURN
C
RETURN
END
SUBROUTINE SYNT

C
C **********************************************************************
C
C SUBROUTINE SYNT
C
C THIS SUBROUTINE PERFORMS THE SYNTACTICAL ANALYSIS
C OF THE INPUT MODEL. IT RETRIEVES THE INFORMATION
C AND STORES IT IN ONE DIMENSIONAL ARRAYS. IT ALSO
C ACTIVATES THE SUBROUTINE PARSER WHICH TRANSFORMES
C THE INPUT IN AN INTERNAL TARGET LANGUAGE MORE
C SUITABLE FOR SYNTACTICAL ANALYSIS. THE SYNT. ANAL. IS
C SYNCHRONIZED SO THAT MOST OF THE ERRORS IN THE
C SAME LINE WILL BE DETECTED.
C IT IS COMPOSED OF TWO BASIC PARTS:
C 1. OBJECTIVE FUNCTION ANALYZER
C 2. CONSTRAINT ANALYZER
C
C THE CONSTRAINT ANALYZER IS COMPOSED OF TWO
C ARITHMETIC EXPRESSION ANALYZERS WHICH ARE THE
C SAME IN PRINCIPLE BUT DESIGNED TO HANDLE
C DIFFERENT TYPES OF CONSTRAINTS.
C
C **********************************************************************

COMMON /EDIT/ MAXROW,FSTAVL,HIACC,EDTNME(200),
*MSFILE,INFILE,LINKS(200)
COMMON /KEYS/ MSIOX(500)
COMMON /OPT/ OPTCOF(5000),RHS(200),TMPBND(400),BASIS(200),
*OBJFCN(200),ICSTNM(200),IVAKNM(200),
*PRNTSW(5),ITRLMT,KOARTS,METHOD,MINMAX,Z,OBJROW,
*ITYPE(200),DIMROW,ICXMAX
COMMON /WCRK/ BUFFER(100),IUST(80),IAROBJ,MSKERR(80),
*MSSKNT(200),MSKERR(80),
*ISUP(200),BOUND(200),
COMMON /LEX/ IWRONG,MCNLSK(200),ICONTRL(50),IBF(80),
*IANUM(10),MMFLAG,AUTOBJ
COMMON /AREA1/ FINF,MINF,FINFZ,BOUND,OBJROW,BASIS,R,AUTOBJ
DIMENSION TBL(3000)
EQUIVALENCE (OPTCOF(200),TBL(1))

** INITIALIZATION 1 **

DATA IBLANK/" /
DATA MSKERR,MSKENT1,MSKENT2/80**"",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","",
DO 10 I=1,3000
10   TBLEAU(I)=0.
    MINMAX=0
    ISTRW=0
    IMPONG=0
    LOSTLM=0
    IFATAL=0
    IROWCT=0
    IROWCT2=0
    ICONT=0
    IEDTRW=0
    IEDTAD=0
    IEDTLK=0
    IUPPER=0
    LOOP=0
    IF(METHOO.GE.5) IUPPER=1

***  MINI-DRIVER  ***

13 IF(IFATAL.EQ.0) GO TO 16
    IMPONG=1
    WRITE(6,4001) (BUFFER(I),I=1,30)
    WRITE(6,4000) (MSKERR(I),I=1,R)
    WRITE(6,4000) (MSKEN1(I),I=1,R)
4001 FORMAT(//,80A1)
    IF(IFATAL.GE.10) WRITE(6,4000) (MSKEN2(I),I=1,R)
    DO 15 I=1,80
        MSKER(I)=IBANK
        MSKEN1(I)=IBANK
        MSKEN2(I)=IBANK
15   CONTINUE

** DISPLAY THE ERROR MESSAGES**

WRITE(7,4999)
4999 FORMAT("END ")
4998 FORMAT(8A10)
REWIND 7
8 READ(7,4998) (IEFER(I),I=1,8)
   IF(IEFER(I),EQ."END ") GO TO 16
   WRITE(6,4998) (IBFER(I),I=1,8)
GOTO 8

** GET NEXT INPUT**

16 IF(IEDTAC.NE.0) GO TO 12
    IEDTRW=IEDTRW+1
    IF(IF(IEDTRW.GT.MAXROW) GO TO 5005
    IEDTAD=LINKS(IEDTRW)
    IF(IEDTAD.EQ.0) GO TO 16
12   CALL MSREAD(IEDTAC,BUFFER,IEDTLK,RESULT,MSFILE)
    IEDTAC=IEDTLK
4000 FORMAT(30A1)
    IF(ICAL.NE.0) GO TO 18
    IROWNM=0
    ICONT2=4
    IROWCT=IROWCT+1
    RHS(IROWCT)=0.

** ACTIVATE THE LEXICAL ANALYZER TO GENERATE THE UNIFORM**
** SYMBOL TABLE ( IUST ) AND TO DETECT THE LOWER LEVEL ERRORS**
CALL PARSER(IBUFFER,ICONTR,IVARNM,IUST,ICSTM,MSKEN,1,
ITRNML,RANIN,MSKERR,MSKEN1,IPNTER,ICTR,INUM,ITRM,ION1,ION2,
2ISTOP,IFATAL,IVROWC,IANUM,IOXMAX,IVARNM,IEOTRM)

REWIND 7
IUST=1
INEXT=1
IF(ICONT.EQ.0) GO TO 19
ICONT=0

** LINK THE EXECUTION TO THE CONTINUATION POINT **

GOTO(233,110,200,93,1233,1110,2200,9219,722,7719773,901,90219,7901,7911,7913) IC

** INITIALIZATION 2 **

BNOLR=RMINF
BNDUPR=PINF
COEF=1.
SIGN=1.
SIGN2=1.
SIGN3=0.
IVARS=0
IRLFL=-1
IRHSL=0
NXTHI1=0
NXTHI2=0
NXTHI3=0
LE=0
IEQ=0
IAESM=0
RHSTMP=0.
ICONT3=0
IDXCF=0
IVARCT=0

** INITIALIZE THE ANALYSIS **

IF(IUST(IUXUST=1).NE.ICTR) GO TO 20
IF(IUST(IUXUST,2).EQ.4) GO TO 490
IF(IUST(IOXUST,2).EQ.2) GO TO 55
IF(IUST(I0XUST,2).EQ.1) GO TO 50
IF(IUST(IOXUST,2).EQ.9) GO TO 90
IF(IUST(IDXUST,2).EQ.3) GO TO 30

IF(IUST(IDXUST,1).EQ.ITRM.AND.IUST(IDXUST,2).EQ.7) GO TO 21
WRONG=1
WRITE(*,4010)
4010 FORMAT(5X,"FATAL ERROR : EACH NEW LINE OF THE MODEL MUST",/,
12X,"BEGIN WITH A CONTROL WORD")
GOTO 492

ISTSW=1
GO TO 13

** ST **

IF(ISTSW,NE.1) GO TO 31
4015 FORMAT(5X,"INFORMATIVE : THE WORD ST SHOULD APPEAR ONLY ONCE","/
11X,"IN EACH MODEL THE SECOND ST IS TAKEN AS AN AND")
31 ISTSW=1
GOTO 490

** MIN **

MINMAX=-1
OBJROW=IVROWC
GOTO 90

C ** MAX **
55 MINMAX=1
OBJROW=IORWCT

C *************************************************** OBJ FUNCTION ANALYZER ***************************************************

90 ICXUST=ICXUST+1
IROWC1=IROWCT+1
93 IF(IUST(IOXUST,1).NE.ION1) GO TO 95
IUST=ICXUST+1
95 IF(IUST(IOXUST,1).EQ.1RM).AND.IUST(IDXUST,2).LT.3) SIGN=0.
IDXUST=IDXUST-1
100 IDCXUST+1
110 IF(IUST(IOXUST,1).EQ.INUM) GO TO 230
120 IF(IUST(IOXUST,1).EQ.ICN2) GO TO 270
200 IF(IUST(IOXUST,1).EQ.1RM) GC TO 280
210 IF(IUST(IOXUST,1).EQ.ISTOP) GO TO 320
220 IF(IUST(IOXUST,1).EQ.ICTR) GO TO 225
225 IFATAL=IFATAL+1
WRITE(7,4035) IFATAL
4035 FORMAT(5X,"FATAL ERROR ",I3," CONTROL WORD IN ARITHMETIC"
1" EXPRESSION")
GOTO 315
230 IF(SIGN.EQ.0.) SIGN=1.
COEF=SIGN*RANUM(IUST(IDXUST,2))
IDXUST=IDXUST+1
ICONT2=1
233 IF(IUST(IOXUST,1).NE.ION2) GO TO 250

C ** ENTER THE COEFFICIENT IN TABLEAU

235 IOXTMP=LSTOLM+IUST(IDXUST,2)
IF(TABLEAU(IOXTMP).NE.0.) GO TO 300
IF(IUST(IDXUST,2).GT.NXTHI3) NXTHI3=IUST(IDXUST,2)
TABLEAU(IOXTMP)=COEF
238 COEF=1.
SIGN=0.
IDXUST=IDXUST+1
ICONT2=3
IF(IUST(IOXUST,1).NE.ISTOP) GO TO 200

C ** GET NEXT INPUT
GOTO 330
250 IF(IUST(IOXUST,1).NE.1RM) GO TO 260
260 IF(IUST(IDXUST,2).NE.7) GO TO 265

C ** GET NEXT INPUT - CONTINUATION
ICONT=1
GOTO 13
265 IF(IUST(IOXUST,1).NE.ISTOP) GO TO 268
IF(NXTHI3.EQ.0) GO TO 320

C ** THIS NUM ACCUMULATES AS RHS FOR THIS LINE

RHS(IROWC1)=RHS(IROWCT)+COEF
GOTO 330
265 RHS(IROWCT)=RHS(IROWCT)+COEF
SIGN=0.
GOTO 200
268 IFATAL=IFATAL+1
WRITE(7,4046) IFATAL
4046 FORMAT(5X,"FATAL ERROR ",I3," + OR - ARE ONLY ALLOWED"
1," FOR THIS LINE")
SIGN=0.
GOTO 315
4045 FORMAT(5X,"FATAL ERROR ",I3," :: ILLEGAL SYNTAX")
270 IF(SIGN.EQ.0.) SIGN=1.
COEF=SIGN
GOTO 235
280 IF(IJUST(IOXUST,2).NE.7) GOTO 281
ICONT=1
GOTO 13
281 IF(IJUST(IOXUST,2).LT.3) GOTO 282
IFATAL=IFATAL+1
WRITE(7,4045) IFATAL
GOTO 315
C ** IDENTIFY THE SIGN
C
282 IF(SIGN.EQ.0.) GO TO 310
IF(IJUST(IOXUST,2).EQ.1) SIGN=1.
IF(IJUST(IOXUST,2).EQ.2) SIGN=-1.
ICONT2=2
IF(SIGN.EQ.0.) GO TO 100
IFATAL=IFATAL+1
WRITE(7,4050) IFATAL
4050 FORMAT(5X,"FATAL ERROR ",I3, ", #WRONG SYMBOL REPLACED A (+)",
1/,23X, "OR A (-)"
GOTO 315
300 IFATAL=IFATAL+1
WRITE(7,4055) IFATAL
4055 FORMAT(5X,"FATAL ERROR ",I3," SAME VARIABLE MORE THAN"
1/",24X,"ONCE IN THE SAME ROW")
CALL ERROR(R,IARR,IANUM,IDXUST,MSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
GOTO 239
310 IFATAL=IFATAL+1
IF(IJUST(IOXUST,2).LT.3) WRITE(7,4060) IFATAL
IF(IJUST(IOXUST,2).GE.3) WRITE(7,4070) IFATAL
315 CALL ERROR(R,IARR,IANUM,IDXUST,MSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
4060 FORMAT(5X,"FATAL ERROR ",I3," ARITHMETIC OPERATOR IN WRONG"
1 " PLACE")
GOTO 100
320 IFATAL=IFATAL+1
WRITE(7,4065) IFATAL
4065 FORMAT(5X,"FATAL ERROR ",I3," :: ILLEGAL SYNTAX AT THE END")
C ** ENTER THE DELIMITER ($) IN TABLEAU
C
330 LSTDLM=LSTDLM+NXTHI3
TBLEAU(LSTDLM)=IDOLR
ICONT=0
C ** GET NEXT INPUT
GOTO 13
C
C *************** CONSTRAINT ANALYZER ***********************
C
490 IDXUST=IDXUST+1
492 ICONTE=3
495 IF(IJUST(IDXUST,1).NE.IDN1) GO TO 540
ICONTE=9
500 IDXUST=IDXUST+1
540 IF(IJUST(IDXUST,1).NE.INUM) GO TO 600
COEF=SIGN*RANUM(IUST(ICXUST,2))
SIGN=1,
ICONT2=10
INEXT=ICXUST+1
IF(IUST(ICXUST,1).AND.ICONT.NE.0) INEXT=1
IF(IUST(INEXT,1).EQ.ISTOP) GO TO 1264
C ** CHECK FOR CONTINUATION
545 IF(IJUST(INEXT,1).NE.ITRM) GO TO 550
IF(IJUST(INEXT,2).NE.7) GO TO 560
ICONT=1
C ** GET NEXT INPUT - CONTINUATION
GOTO 13
C
550 IF(IJUST(INEXT,1).EQ.ION2) GO TO 555
IF(IJUST(INEXT,1).NE.ICTR) GO TO 553
IFATAL=IFATAL+1
WRITE(7,4035) IFATAL
GOTO 554
553 IFATAL=IFATAL+1
WRITE(7,4070) IFATAL
554 CALL ERROR(R,IARR,IANUM,INEXT,MSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
IUST=IJUST+1
INEXT=IUST+1
IF(IJUST(INEXT,1).EQ.ISTOP) GO TO 1264
GOTO 545
4070 FORMAT(5X,"FATAL ERROR ",13," ILLEGAL SYNTAX")
555 IUST=INEXT
SIGN2=1,
GOTO 1235
C
C ** <= **
C
560 IF(IJUST(INEXT,2).NE.3) GO TO 570
LE=1
BNOLR=COEF
LNTYPE=4
GOTO 530
C
C ** >= **
C
570 IF(IJUST(INEXT,2).NE.4) GO TO 573
IGE=1
BNUPR=COEF
LNTYPE=3
GOTO 580
C
C ** = **
C
573 IF(IJUST(INEXT,2).NE.5) GO TO 575
IEQ=1
BNOLR=COEF
BNDEQL=COEF
BNUPR=COEF
LNTYPE=2
GOTO 580
C
575 IF(IJUST(INEXT,2).LT.3) GO TO 578
IFATAL=IFATAL+1
WRITE(7,4075) IFATAL
CALL ERROR(R,IARR,IANUM,INEXT,MSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
4075 FORMAT(5X,"FATAL ERROR ",13," ILLEGAL TERMINAL SYMBOL AFTER" 
1"A NUMERIC ENTRY")
GOTO 500
578 IUST=INEXT
** ENTRY POINT IN ARITHMETIC EXPRESSION ANALYZER #1

GOTO 1263

** CHECK FOR TERMINATION

IF(IUST(INEXT,1).NE.ISTOP) GO TO 582
IF(IUPPER.EQ.1.OR.COEF.EQ.0.) GO TO 581

** GENERATE AN EXPLICIT CONSTRAINT

RHS(IROWCT)=COEF
IDXTMP=LSTDLM+IUST(IDXUST,2)
TBLEAU(IDXTMP)=1.
LSTDLM=LSTDLM+IUST(IDXUST,2)+1
TBLEAU(LSTDLM)=IDOLR
ITYPE(IROWCT)=LTYPE
GOTO 13

** UPDATE THE ARRAY BOUND

BOUND(IUST(IDXUST,2),1)=AMAX1(BOUND(IUST(IDXUST,2),1),BNOLR)
BOUND(IUST(IDXUST,2),2)=AMIN1(BOUND(IUST(IDXUST,2),2),BNoupR)
IROWCT=IROWCT-1
GOTO 13

IF(IUST(INEXT,1).EQ.ITRM) GO TO 585

** THIS IS AN ERROR SITUATION

IF(IUST(INEXT,1).NE.ICTP) GO TO 583
IFATAL=IFATAL+1
WRITE(7,4035) IFATAL
CALL ERROR(R,IARR,IANUM,INEXT,MSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
IOXUST=INEXT
GOTO 2230

584 I Furyatal=IFATAL+1
WRITE(7,4070) IFATAL
ITEM=INEXT
GOTO 5000

585 IF(IUST(INEXT,2).NE.6) GO TO 586

** SO IT IS A "*"

IDXTMP=INEXT
IF(INEXT.NE.1) GO TO 762
TBLEAU(LSTDLM+ISUP1(1))=0.
GOTO 765

** 2270 IS AN ENTRY POINT IN ARITHMETIC EXPRESSION ANALYZER #2
C 586 IF(IUST(INEXT,2), LE, 2, AND, INEXT, EQ, 1) GO TO 2282
IF(IUST(INEXT,2), LE, 2) GO TO 2270
IF(IUST(INEXT,2I), GT, 6) GO TO 588
IDXSUP=1
ISUP1(I)=IUST(IDXUST,2)
IDXUST=INEXT
GOTO 788
588 ICONT=1
IF(IUST(IDXUST,1), NE, IDN2) GO TO 13
TABLEAU(LSTDLM+IUST(IDXUST,2))=1.
IVARCT=IVARCT+1
IDXSUP=1
C C ISUP1 IS A WORKING AREA FOR STORING THE INDECES OF C * 4 * TH VARIABLES IN LIST CONSTRAINTS C
C ISUP1(I)=IUST(IDXUST,2)
GOTO 13
590 IF(IUST(INEXT,1), NE, INUM) GO TO 591
IDXUST=INEXT
GOTO 2230
591 IF(IUST(INEXT,1), NE, ITRM) GO TO 593
IF(IUST(INEXT,2), NE, 7) GO TO 592
ICONT=1
GOTO 13
592 IDXUST=INEXT
SIGN=0.
C ** 2282 IS AN ENTRY POINT IN ARITH. EXP. ANAL #2 C
GOTO 2282
593 IF(IUST(INEXT,1), NE, ICTR) GO TO 594
IFATAL=IFATAL+1
WRITE(7,4035) IFATAL
CALL ERROR(R, IARR, IANUM, IDXUST, MSKERR, MSKEN1, MSKEN2, IPNTER, IFATAL)
C C ** RESYNCHRONIZE THE ANALYZER C
GOTO 580
594 IFATAL=IFATAL+1
WRITE(7,4030) IFATAL
4080 FORMAT(5X,"FATAL ERROR #",I3," : INCOMPLETE INPUT")
ITEM=IDXUST
GOTO 5000
C 600 IF(IUST(IDXUST,1), NE, ITRM) GO TO 700
IF(IUST(IDXUST,2), NE, 7) GO TO 605
ICONT=1
GOTO 13
605 ICONT2=9
C C ** IDENTIFY THE SIGN IF ANY C
IF(SIGN3, NE, 0) GO TO 610
IF(IUST(IDXUST,2), EQ, 1) SIGN3=1.
IF(IUST(IDXUST,2), EQ, 2) SIGN3=-1.
SIGN=SIGN3
IF(SIGN, NE, 0) GO TO 500
IFATAL=IFATAL+1
WRITE(7,4050) IFATAL
CALL ERROR(R, IARR, IANUM, IDXUST, MSKERR, MSKEN1, MSKEN2, IPNTER, IFATAL)
GOTO 500
610 IFATAL=IFATAL+1
WRITE(7,4085) IFATAL
4085 FORMAT(5X,"FATAL ERROR #",I3," : TWO CONSECUTIVE TERMINAL")
1 "SYMBOL-Version"
CALL ERROR(R, IARR, IANUM, IDXUST, MSKERR, MSKEN1, MSKEN2, IPNTER, IFATAL)
GOTO 50
C
700 IF(IUST .EQ. IDNST) GO TO 720
IF(IUST .EQ. IDNST) GO TO 710
IFATAL = IFATAL + 1
WRITE(7, 0) IFATAL
ITEM = IDNST
GOTO 5000
710 IF(IUST(IDXUST, 2) .EQ. 5) GO TO 900
IFATAL = IFATAL + 1
WRITE(7, 4035) IFATAL
CALL ERROR(R, IARR, IANUM, IDXUST, MSKERR, MSKEN1, MSKEN2, IPNTER, IFATAL)
GOTO 500
720 IDXUST = IDXUST + 1
ICONT = 16
722 IF(IUST(IDXUST, 1) .NE. ICTR) GO TO 725
IF(IUST(IDXUST, 2) .EQ. 3) GO TO 882
ITEM = IDXUST
IFATAL = IFATAL + 1
WRITE(7, 4095) IFATAL
4095 FORMAT(5X, "FATAL ERROR ", 13, " ; THE ONLY CONTROL WORD THAT"
1/24X, "WOULD BE ALLOWED AT THIS POINT IS URS")
GOTO 5000
725 IF(IUST(IDXUST, 1) .EQ. ITRM) GO TO 740
C ** IF THE FLOW GOES THROUGH THIS PATH THERE IS AN ERROR
C ** DETECT THE ERROR AND RESYNCHRONIZE THE ANALYZER
IF(IUST(IDXUST, 1) .NE. INUM) GO TO 730
IFATAL = IFATAL + 1
WRITE(7, 4030) IFATAL
CALL ERROR(R, IARR, IANUM, IDXUST, MSKERR, MSKEN1, MSKEN2, IPNTER, IFATAL)
C
C ** 1230 IS AN ENTRY POINT IN ARITH. EXP. ANAL. #1
C
GOTO 1230
730 IFATAL = IFATAL + 1
WRITE(7, 4080) IFATAL
ITEM = IDXUST
GOTO 5000
740 IF(IUST(IDXUST, 2) .GE. 3) GO TO 750
745 IF(IDXUST .GT. 1) GO TO 747
SIGN = 0.
C
C ** ENTRY POINTS IN ARITH. EXP. ANAL. #1
C
GOTO 1200
747 IDXUST = IDXUST - 1
GOTO 1270
750 IF(IUST(IDXUST, 2) .LT. 6) GO TO 745
IF(IUST(IDXUST, 2) .EQ. 6) GO TO 760
IF(IDXUST .EQ. 1) GO TO 755
NXTHI1 = IUST(IDXUST - 1, 2)
TABLEAU(LSTOLM + NXTHI1) = SIGN
IVARCT = IVARCT + 1
IDXSUP = 1
ISUP(1) = NXTHI1
755 ICONT = 1
GOTO 13
760 IF(IDXUST .EQ. 1) GO TO 765
762 LAST = IDXUST - 1
IDXSUP = 1
ISUP(1) = IDXSUP = IUST(LAST, 2)
**THE FOLLOWING SECTION IS THE SYNTACTICAL ANALYSIS OF SINGLE VARIABLE CONSTRAINTS**

765  **ICONT2=17**
    IF(NXTHII .NE. 0) TBLEAU(LSTOLM+NXTHII)=0.
    IF(IUST(IJXUST+1,1).NE.ITRM.OR.IUST(IDXUST+1,2).NE.7) GO TO 770
    ICONT=1
    GOTO 13

770  IJXUST=IDXUST+1

771  IF(IUST(IDXUST,1).NE.ION2) GO TO 775
    IDXSUP=IDXSUP+1
    ISUP1(IDXSUP)=IUST(IDXUST,2)
    IDXUST=IDXUST+1

773  IF(IUST(IDXUST,1).NE.1TRM) GO TO 780
    **ICONT2=18**
    IF(IUST(IDXUST,2).NE.6) GO TO 784
    GOTO 770

775  ITEM=IDXUST-1
    IFATAL=IFATAL+1
    WRITE(7,4100) IFATAL
    4100 FORMAT(5X,"FATAL ERROR ",I3," THE COMMA AT THIS POINT IS AN ILLEGAL ENTRY")
    GOTO 5000

780  IF(IUST(IDXUST,1).EQ.ICTR) GO TO 880
    IF(IUST(IDXUST,1).EQ.ISTOP) GO TO 795
    IFATAL=IFATAL+1
    ITEM=IDXUST
    WRITE(7,4070) IFATAL
    GOTO 5000

784  IF(IUST(IDXUST,2).LT.6.AND.IUST(IOXUST+2).GT.2) GO TO 788
    IF(IUST(IOXUST,2).NE.7) GO TO 785
    **ICONT=1**
    GOTO 13

785  IF(IUST(IDXUST,1).NE.INUM) GO TO 804
    BNDUPR=SIGN*RANUM(IUST(IDXUST,2))-RHSTMP
    GOTO 793

788  INEXT=MAXD(IOXUST,I0XCF)+1
    SIGN=1.
    IF(IUST(INEXT,1).NE.1TRM) GO TO 790
    If(IUST(INEXT,2).EQ.7) GO TO 790
    SIGN=0.

C **IDENTIFY THE SIGN**

    IF(IUST(INEXT,2).EQ.1) SIGN=1.
    IF(IUST(INEXT,2).EQ.2) SIGN=-1.
    INEXT=INEXT+1
    IF(SIGN .NE. 0) GO TO 790
    IFATAL=IFATAL+1
    WRITE(7,4050) IFATAL
    ITEM=INEXT
    GOTO 5000

C **IDENTIFY THE RELATIONAL OPERATOR AND ASSOCIATED BOUND**

790  IF(IUST(IDXUST,2).NE.3) GO TO 791
    IF(IUGE.EQ.1).OR.(IUE.EQ.1)) GO TO 792
    LNTYPE=3
    LE=1
    ICONT2=22

7901  IF(IUST(INEXT,1).NE.INUM) GO TO 804
    BNDUPR=SIGN*KANUM(IUST(INEXT,2))-RHSTMP
    GOTO 793

791  IF(IUST(IDXUST,2).NE.4) GO TO 7912
IF((LE.EQ.1).OR.(IEQ.EQ.1)) GO TO 792
LNTYPE=4
IGE=1
ICONT2=23
7911 IF(IUST(INEXT,1).NE.INUM) GO TO 804
BNOLR=SIGN*RANUN(IUST(INEXT,2))-RHSMP
GOTO 793
7912 IF((LE.EQ.1).OR.(IGE.EQ.1).OR.(IEQ.EQ.1)) GO TO 794
IEQ=1
ICONT2=24
7913 IF(IUST(INEXT,1).NE.INUM) GO TO 804
BNUPK=SIGN*RANUN(IUST(INEXT,2))
BNULR=BNUPK
BNDEOL=BNUPK
GOTO 793
792 IFATAL=IFAT
WRITE(7,41 	 IFATAL
ITEM=IUXU'
GOTO 5000
C ** CHECK THE BOUNDS IF NEEDED
C 793 IF(BNUPK.GE.BNOLR.OR.(IAESW.NE.1.AND.IUPPER.EQ.1)) GO TO 795
IFATAL=IFATAL+1
WRITE(7,4107) IFATAL
4107 FORMAT(5X,"FATAL ERROR ",I3,2X,"THE UPPER BOUND IS LOWER",
1" THAN THE LOWER BOUND")
ITEM=INEXT
GOTO 5000
4105 FORMAT(5X,"FATAL ERROR ",I3,2X," THE TWO RELATIONAL OPERATORS"
1" ARE NOT THE SAME")
794 IFATAL=IFATAL+1
WRITE(7,4110) IFATAL
ITEM=IUXU
4110 FORMAT(5X,"FATAL ERROR ",I3,2X," THE EQUALITY SIGN MAKES THIS
1CONSTRAINT MEANINGLESS")
GOTO 5000
804 IF(IUST(INEXT,1).NE.ITRM) GO TO 810
IF(IUST(INEXT,2).EQ.7) GO TO 805
IF(IUST(INEXT,2).EQ.1) SIGN=1.
IF(IUST(INEXT,2).EQ.2) SIGN=-1.
IF(SIGN.EQ.0) GO TO 810
INEXT=INEXT+1
GOTO 25
805 ICONT=1
GOTO 13
810 IFATAL=IFATAL+1
WRITE(7,4120) IFATAL
4120 FORMAT(5X,"FATAL ERROR ",I3,2X," ILLEGAL SYNTAX TO THE RIGHT OF
1A RELATIONAL OPERATOR")
ITEM=IDXUST
GOTO 5000
C ** STORE THE COEFFICIENTS DEPENDING ON THE SPECIFIED METHOD
C 795 IF(IUARS.EQ.1) GO TO 910
IF((IUPPER.EQ.1.OR.BNOLR.EQ.0.OR.BNUPR.EQ.0).AND
1.IAESW.EQ.0) GO TO 800
IF(IUERS.EQ.1) GO TO 7955
C ** SO WE HAVE A DOUBLE BOUNDED CONSTRAINT
C IF(IUPPER.EQ.1) GO TO 7954
** SINCE \( \text{PER}=0 \) TWO EXPLICIT CONSTRAINTS WILL BE GENERATED

```plaintext
IF(LE .LT. 1) GO TO 7951
ITYPE(IROWCT)=3
RHS(IJWCT)=BNDUPR
IROWC=IROWCT+1
ITYPE(IROWCT)=4
RHS(IROWCT)=BNDLR
ICSTNM(IROWCT)=ICSTNM(IROWCT-1)
GOTO 7952
```

7951

```plaintext
ITYPE(IROWCT)=4
RHS(IROWCT)=BNDLR
IROWCT=IROWCT+1
ITYPE(IROWCT)=3
RHS(IROWCT)=BNDUPR
ICSTNM(IROWCT)=ICSTNM(IROWCT-1)
```

** NOW ENTER THE COEFS OF THE GENERATED CONSTRAINT IN TELEAU **

```plaintext
7952 IDX10=LSTDLM+NXTHI2+1
TBLEAU(IDX10)=IDOLR
DO 7953 I=1,NXTHI2
TBLEAU(IDX10+I)=TELEAU(LSTDLM+I)
7953 CONTINUE
IDX10=IDX10+NXTHI2+1
TBLEAU(IDX10)=IDOLR
LSTDLM=IDX10
GOTO 13
```

** SINCE IUPPER=1 SECOND CONSTRAINT WILL NOT BE GENERATED AND THE TWO BOUNDS WILL BE HANDLED IMPLICITLY BY CONSTRAINTING THE SLACK **

```plaintext
7954 ITYPE(IROWCT)=LNTYPE
RHS(IROWCT)=BNDUPR
BNDSLK(IROWCT)=BNCUPR-BNDLR
LSTDLM=LSTDLM+NXTHI2+1
TBLEAU(LSTDLM)=IDOLR
IDXUST=INEXT+1
IF(IUST(IDXUST,1).EQ.ISTOP) GO TO 13
ITEM=IDXUST
IFATAL=IFATAL+1
WRITE(7,4075) IFATAL
GOTO 5000
```

** ALL THE SINGLE VARIABLE CONSTRAINTS ARE EXPLICIT **

```plaintext
7955 LINAME=ICSTNM(IROWCT)
LNTYPE=ICSTNM(IROWCT)
```

** <= **

```plaintext
IF(LE .GE. 1) GO TO 796
LNTYPE=3
IF(BNDUPR.EQ.PINF) GO TO 7957
BND=BNDUPR
GOTO 797
7957 BND=BNDEL
BNDEL=RMINF
LNTYPE=4
GOTO 797
```

** => **

```plaintext
796 IF(IGL.NE.1) GO TO 7963
7962 LNTYPE=4
IF(BNDLR.EQ.RMINF) GO TO 7961
BND=BNDLR
```

```plaintext
7963 ** SINCE \( \text{PER}=0 \) TWO EXPLICIT CONSTRAINTS WILL BE GENERATED

```
BNDLR=RMINF
GOTO 797
7961 BND=BNDUPR
LNTYPE=3
GOTO 797
7963 IF(IIEQ.NE.1) GO TO 7964
BND=BNDOL
LNTYPE=2
BNDLR=RMINF
GOTO 797
7964 IFATAL=IFATAL+1
WRITE(7,4070) IFATAL
ITEM=IDXUST
GOTO 5000
797 DO 799 I=1,IXSUPT
RHS(IROWCT)=BND
ICTMP=LSTOLM+ISUP1(I)
TABLEAU(ICTMP)=1.
LSTOLM=LSTOLM+ISUP1(I)+1
TABLEAU(LSTOLM)=IDCLR
ITYPE(IROWCT)=LNTYPE
ICSTNM(IROWCT)=LINAME
IROWCT=IROWCT+1
799 CONTINUE
IF(BNDLR.NE.RMINF) GO TO 7962
IROWCT=IROWCT-1
GOTO 13
C
C ** ALL THE SINGLE VARIABLE CONSTRAINTS ARE IMPLICIT
C
800 DO 802 I=1,IXSUPT
BOUND(ISUP1(I),1)=AMAX1(BOUND(ISUP1(I),1),BNDLR)
BOUND(ISUP1(I),2)=AMIN1(BOUND(ISUP1(I),2),BNOLR)
802 CONTINUE
IF(IUPPER.EQ.0.AND.BNOLR.EQ.0.AND.BNDLR.NE.PINF.AND.BNDLR.NE.BNDUPR)
GO TO 885
IF(IUPPER.EQ.0.AND.BNDLR.EQ.0.AND.BNDUPR.NE.BNDLR.AND.BNOLR.NE.RMINF.AND.BNDUPR.
1NE.BNDLR) GO TO 887
ITYPE(IROWCT)=0.
IROWCT=IROWCT-1
GOTO 13
885 BNDLR=RMINF
GOTO 7955
887 BNDUPR=PINF
GOTO 7955
C
880 IF((LE.EQ.1).OR.(IGE.EQ.1).OR.(IEQ.EQ.1)) GO TO 881
IF(IUST(IDXUST,2).EQ.8) GO TO 882
ITEM=IDXUST
IFATAL=IFATAL+1
WRITE(7,4035) IFATAL
GOTO 5000
881 IFATAL=IFATAL+1
ITEM=IDXUST
WRITE(7,4125) IFATAL
4125 FORMAT(5X,"FATAL ERROR #",I3," URS SPECIFICATION OR OTHER
1CONTROL WORD IN A CONSTRAINT",/"WITH A RELATIONAL OPERATOR
2IS ILLEGAL")
GOTO 5000
C ** 882 ** NO ACTION WILL BE TAKEN FOR THE URS WHICH IS DEFAULTED
C
882 GOTO 13
C
900 IDXUST=IDXUST+1
ICONT2=19
IROWICT=IROWICT-1
IOTHER=0
IVARS=0
901 IF(IUST(IDXUST,1).NE.ICTR) GO TO 903
IF(IUST(IDXUST,2).NE.6) GO TO 902
IOTHER=1
IDXUST=IDXUST+1
ICONT2=20
902 IF(IUST(IDXUST,1).NE.ICTR) GO TO 905
IF(IUST(IDXUST,2).NE.7) GO TO 906
IVARS=1
IDXUST=IDXUST+1
ICONT2=21
9021 IF(IUST(IDXUST,1).EQ.ICTR) GO TO 9011
C ** IF THE NEXT ENTRY IS A <= OR A >= OR = GOTO 788
IF(IUST(IDXUST,1).EQ.ITRM.AND.(IUST(IDXUST,2).LT.6.
1AND.IUST(IDXUST,2).GT.2)) GO TO 788
GOTO 908
9011 IF(IUST(IDXUST,2).NE.6) GO TO 907
C ** NO ACTION IS TAKEN FOR THE URS
IDXUST=IDXUST+1
IF(IUST(IDXUST,1).EQ.ISTOP) GO TO 13
IFATAL=IFATAL+1
WRITE(7,4065) IFATAL
GOTO 13
903 IF(IUST(IDXUST,1).NE.ITRM.OR.IUST(IDXUST,2).NE.17) GO TO 904
ICONT=1
GOTO 13
904 ITEM=IDXUST
IFATAL=IFATAL+1
WRITE(7,4130) IFATAL
4130 FORMAT(5X,"FATAL ERROR #",I3," : THE CONTROL WORD ALL MUST BE ",
1"FOLLOWED BY THE WORD VARS")
GOTO 5000
905 IF(IUST(IDXUST,1).NE.ITRM.OR.IUST(IDXUST,2).NE.17) GO TO 906
ICONT=1
GOTO 13
906 ITEM=IDXUST
IFATAL=IFATAL+1
WRITE(7,4135) IFATAL
4135 FORMAT(5X,"FATAL ERROR #",I3," : THE CONTROL WORD OTHER MUST 
1BE FOLLOWED BY THE WORD VARS")
GOTO 5000
907 IF(IUST(IDXUST,1).NE.ITRM) IFATAL=IFATAL+1
WRITE(7,4140) IFATAL
4140 FORMAT(5X,"FATAL ERROR #",I3," : AT THIS POINT THE ONLY CONTROL 
1WORD ALLOWED IS URS")
GOTO 5000
908 IF(IUST(IDXUST,1).NE.ITRM.AND.IUST(IDXUST,2).NE.17) GO TO 909
ICONT=1
GOTO 13
909 ITEM=IDXUST
IFATAL=IFATAL+1
WRITE(7,4145) IFATAL
4145 FORMAT(5X,"FATAL ERROR #",I3," : AT THIS POINT OF THIS CONSTRAINT 
1THIS ENTRY IS SYNTACTICALLY ILLEGAL")
GOTO 5000
C
910 IF(IEQ.EQ.1) GO TO 914
C
C ** IDXBNO WILL BE 1 OR 2
C
C IDXBNO=1+LE
C
C ** LE, IGE ARE THE NEGATION OF EACH OTHER
C ** IE LE=(1,0), IGE=(0,1)
C
913 I=1,IDXMAX
IF(ABS(BOUND(I,1,1).GE.0).AND.IOTHER.
1EQ.1) GO TO 913
VALUE=IGE*AMAX1(BOUND(I,1,1),BNOLR)+
1LE*AMIN1(BOUND(I,2,1),BNUPR)
BOUND(I,ICX3ND)=VALUE
913 CONTINUE
GOTO 13
C
C ** IF A VARIABLE IS SET = TO A VALUE,
C ** THIS VALUE WILL BE STORED AS UPPER AND LOWER BOUND
C
914 I=1,IDXMAX
IF((BOUND(I,1,1).NE.RMINF.OR.BOUND(I,2,1).NE.PINF).
1AND.IOTHER.EQ.1) GO TO 916
BOUND(I,1,1)=AMAX1(BOUND(I,1,1),BNOLR)
BOUND(I,2,1)=AMIN1(BOUND(I,2,1),BNUPR)
916 CONTINUE
GOTO 13
C
C **************************************************************
C
C ** ARITHMETIC EXPRESSION ANALYZER #1 **
C
C 1100 IDXUST=IDXUST+1
C 1110 IF(IUST(IDXUST,1,1).EQ.INUM) GO TO 1230
C 1200 IF(IUST(IDXUST,1,1).EQ.I2N2) GO TO 1270
C 1200 IF(IUST(IDXUST,1,1).EQ.I2N2) GO TO 1280
C 1200 IF(IUST(IDXUST,1,1).EQ.ISTOP) GO TO 1320
C
C IFATAL=IFATAL+1
WRITE(7,4030) IFATAL
CALL ERROR(RT1ARR,IANUM,IDXUST,MSKERR,MSKEN1,MSKEN2,IFNTER,IFATAL)
IF(IUST(IDXUST,1,1).NE.ICTR) GO TO 1110
IFATAL=IFATAL+1
WRITE(7,4035) IFATAL
CALL ERROR(RT1ARR,IANUM,IDXUST,MSKERR,MSKEN1,MSKEN2,IFNTER,IFATAL)
GOTO 1100
C 1230 IF(SIGN.EQ.0.) SIGN=1.
COEF=SIGN*RANUM(IUST(IDXUST,2,1))
SIGN=0.
IDXCF=IDXUST-2
IDXUST=IDXUST+1
ICONT2=5
1233 IF(IUST(IDXUST,1,1).NE.INUM) GO TO 1250
1235 IDXTMP=LSTOLM+IUST(IDXUST,2,1)
IVARCT=IVARCT+1
IF(TBEAU(IDXTMP).NE.0) GO TO 1300
TBEAU(IDXTMP)=SIGN2*COEF
C
C ** NXTHI1 IS THE INDEX OF THE "HIGHEST" VARIABLE IN
C ** THIS EXPRESSION
IF(NXTHI1.LT.IUST(IDXUST,2)) NXTHI1=IUST(IDXUST,2)
1238 COEF=1.
SIGN=0.
IDXUST=IDXUST+1
ICONT=7
IF(IUST(IDXUST,1).NE.ISTOP) GO TO 1200
GOTO 1262
1250 IF(IUST(IDXUST,1).NE.ITRM) GO TO 1260
IF(IUST(IDXUST,2).NE.7) GO TO 1263
ICONT=1
GOTO 13
1260 IF(IUST(IDXUST,1).NE.ISTOP) GO TO 1268
CFTMF=TBLEAU(LSTOLM+NXTHI1)
RHSTMP=—PHS(IPOWCT)
IF(IUPPER.NE.1.OR.IVARCT.GT.1.RHSTMP.NE.1.) GO TO 1261
1258 IOXSUP=1
ISUP1(1)=NXTHI1
TBLEAU(LSTOLM+NXTHI1)=0.
C
C ** RESTORE LOST INFORMATION BY REPRODUCING THE REL. OPERATOR
C
IF(ICONT3.EQ.0) GO TO 1259
DO 1257 I=1,IDXUST
IUST(IDXUST+2-I,1)=IUST(IDXUST+1—I,1)
1257 IUST(IDXUST+2-I,2)=IUST(IDXUST+1—I,2)
IUST(1,1)=ITRM
IUST(1,2)=IROWCT(IROWCT)
IF(IROWCT(IROWCT).EQ.2) IUST(1,2)=5
IDXCF=0
IDXUST=1
GOTO 788
1259 IUST=IDXREL
GOTO 788
C
C ** THIS NUM IS A RHS ENTRY FOR THIS ROW
C
1261 RHS(IROWCT)=RHS(IROWCT)+COEF
IF(IUPPER.EQ.0.AND.RHS(IROWCT).EQ.0.AND.IVARCT.EQ.1.
1.AND.CFTMF.EQ.1) GO TO 1258
C
C ** ENTER THE DELIMITER ($) IN TBLEAU
C
1262 LSTOLM=LSTOLM+NXTHI1+1
TBLEAU(LSTOLM)=IDCLR
ICONT=0
C
C ** CHECK FOR LINE COMPLETENESS
C
IF(IRELFL.NE.-1) GO TO 13
1264 IFATAL=IFATAL+1
WRITE(7,4985) IFATAL
4985 FORMAT(5X,"FATAL ERROR ",13," : INCOMPLETE INPUT , CONTINUATION",
1/23X," OP RELATIONAL OPERATOR IS MISSING ")
C
C ** GET NEXT INPUT
GOTO 13
1263 RHS(IROWCT)=RHS(IROWCT)+IRELFL*COEF
IF(IUST(IDXUST,2).GT.2.AND.IUST(IDXUST,2).LT.6)
GO TO 1500
SIGN=0.
GOTO 1200
1268 IFATAL=IFATAL+1
WRITE(7,4070) IFATAL
SIGN=0.
GOTO 1315
1270 IF(SIGN.EQ.0) SIGN=1.
   COEF=SIGN
   GOTO 1235
1280 IF(IUST(IDXUST,2).EQ.7) GO TO 1330
   IF(SIGN.NE.0) GO TO 1310
   C
   C ** IDENTIFY THIS SYMBOL. IT IS AN ARITHMETIC OR A
   C ** RELATIONAL OPERATOR
   C
   IF(IUST(IDXUST,2).EQ.1) SIGN=1.
   IF(IUST(IDXUST,2).EQ.2) SIGN=-1.
   ICONT2=E
   IF(SIGN.NE.0) GO TO 1100
   IF(IUST(IDXUST,2).GT.2.AND.IUST(IDXUST,2).LT.6) GO TO 1400
   IFATAL=IFATAL+1
   WRITE(7,4050) IFATAL
   GOTO 1315
1300 IFATAL=IFATAL+1
   WRITE(7,4055) IFATAL
   CALL ERROR(R,IANUM,IDXUST,MSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
   GOTO 1233
1310 IFATAL=IFATAL+1
   IF(IUST(IDXUST,2).EQ.3) GO TO 1420
   IF(IUST(IDXUST,2).EQ.3) WRITE(7,4070) IFATAL
   1315 ITEM=IDXUST
   CALL ERROR(R,IANUM,ITEM,MSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
   GOTO 1100
1320 IFATAL=IFATAL+1
   WRITE(7,4065) IFATAL
   GOTO 13
1330 IF(IRELFL.EQ.1) ICONT3=1
   ICONT=1
   GOTO 13
1400 IF(IRELFL.EQ.1) GO TO 1600
1500 ITYPE(IXROWCT)=IUST(IDXUST,2)
   IF(IUST(IDXUST,2).EQ.5) ITYPE(IXROWCT)=2
   IRELFL=1
   SIGN2=-1.
   IOXREL=IDXUST
   SIGN=0.
   ICONT2=E
   GOTO 1100
1600 IFATAL=IFATAL+1
   WRITE(7,4160) IFATAL
   4160 FORMAT(5X,"FATAL ERROR #" ,I3,":	 ONLY ONE RELATIONAL OPERATOR
    1( <=, > = ) IS ALLOWED IN THIS LINE")
   ITEM=IDXUST
   GOTO 5000
   C
   C ************************************************************
   C
   C ** ARITHMETIC EXPRESSION ANALYZER #2 **
   C
2100 IDXUST=IDXUST+1
2110 IF(IUST(IDXUST,1).EQ.INUM) GO TO 2230
   IF(IUST(IDXUST,1).EQ.1N2) GO TO 2270
2200 IF(IUST(IDXUST,1).EQ.ITRM) GO TO 2280
   IF(IUST(IDXUST,1).EQ.ISTOP) GO TO 2320
   IFATAL=IFATAL+1
   WRITE(7,4030) IFATAL
   CALL ERROR(R,IANUM,IDXUST,MSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
   IF(IUST(IDXUST,1).EQ.1CTR) GO TO 2110
IFATAL=IFATAL+1
WRITE(7,4035) IFATAL
CALL ERROR(R,IAARR,IANUM,IDXUST,MSKERR,MSKEN1,MSKEN2,IFNTER,IFATAL)

C ** RESYNCHRONIZE THE ANALYZER
C
GOTO 2100

2230 IF(SIGN.EQ.0.) SIGN=1.
COEF=SIGN*RANUM(IUST(IDXUST,2))
IDXUST=IDXUST+1
ICONT2=12

2233 IF(IUST(IDXUST,1).NE.IDN2) GO TO 2250

2235 IDXTMP=LISTOLM*IDUST(IDXUST,2)
IVARCT=IVARCT+1
IF(TOLEAU(IDXTMP).NE.0.) GO TO 2300
TLEAU(IDXTMP)=COEF
IF(NXTHI2.LT.IUST(IDXUST,2)) NXTHI2=IUST(IDXUST,2)

2238 COEF=1.
SIGN=0.
IDXUST=IDXUST+1
ICONT2=14

2239 IF(IUST(IXUST,1).NE.ISTOP) GO TO 2200

C ** <= **
C
IF(LE.NE.1) GO TO 2241
TLEAU=LISTOLM+NXTHI2
IF(IVARCT.EQ.1.AND.TLEAU.EQ.1.AND.(BNOLR.EQ.RHSTMP.
1OR.IUPPER.EQ.1)) GO TO 2243

C ** INVERT THE TYPE OF THE CONSTRAINT
C
ITYPE(IROWCT)=4
RHS(IROWCT)=BNOLR-RHSTMP
GOTO 2245

2243 TLEAU=LISTOLM+NXTHI2=0.
BOUND(NXTHI2,1)=AMAX1(BOUND(NXTHI2,1),BNOLR-RHSTMP)
IROWCT=IROWCT-1
GOTO 13

C ** >= **

2241 IF(IGE.NE.1) GO TO 2242
ITYPE(IROWCT)=3
RHS(IROWCT)=BNUPR-PHSTMP
GOTO 2245

C ** = **

2242 RHS(IROWCT)=BNDEQL
ITYPE(IROWCT)=2
2245 LISTOLM=LISTOLM+NXTHI2+1
TLEAU=LISTOLM=IDOLR
GOTO 13

2250 RHSTMP=RHSTMP+COEF
SIGN=0.
GOTO 2239

2270 COEF=SIGN
GOTO 2235

2280 IF(IUST(IDXUST,2).EQ.7) GO TO 2330
IF(IUST(IDXUST,2).LE.2.OR.IUST(IDXUST,2).GE.16) GO TO 2281
IAESW=1
IF(IVARCT.GT.1) GO TO 2275
IF(UPPER.NE.1.AND.BNOLR.NE.RHSTMF) GO TO 2275
ISUP1(1)=NXTHI2
IAESW=0
TABLEAU(LSTDLM+NXTHI2)=0.
2275 IF(LE.EQ.1) BNOLR=BNOLR-RHSTMF
IF(IGE.EQ.1) BNCUFR=BNOLR-RHSTMF
C
C ** IAESW=1 IFF AN ARITHMETIC EXPRESSION WAS ANALYZED
C
GOTO 789
C
2281 IF(SIGN.NE.0.) GO TO 2310
2292 IF(IJUST(IDXUST,2).EQ.1) SIGN=1.
IF(IJUST(IDXUST,2).EQ.2) SIGN=-1.
ICONT2=13
IF(SIGN.NE.0.) GO TO 2100
C
IFATAL=IFATAL+1
WRITE(7,4050) IFATAL
CALL ERROR(R,IARR,IANUM,IDUST,MSKERR,MSKEN1,MSKEN2,IFNTER,IFATAL)
GOTO 2100
2300 IFATAL=IFATAL+1
WRITE(7,4055) IFATAL
CALL ERROR(R,IARR,IANUM,IDUST,MSKERR,MSKEN1,MSKEN2,IFNTER,IFATAL)
GOTO 2238
2310 IFATAL=IFATAL+1
IF(IJUST(IDXUST,2).LT.3) WRITE(7,4060) IFATAL
IF(IJUST(IDXUST,2).GE.3) WRITE(7,4070) IFATAL
CALL ERROR(R,IARR,IANUM,IDUST,MSKERR,MSKEN1,MSKEN2,IFNTER,IFATAL)
GOTO 2100
2320 IFATAL=IFATAL+1
WRITE(7,4065) IFATAL
LSTDLM=LSTDLM+NXTHI2+1
TABLEAU(LSTDLM)=IDCLR
GOTO 13

2330 ICONT=1
GOTO 13
C
C **********************************************************
C ** LOCATE ANC POINT THE ERROR **
C
5000 CALL ERROR(R,IARR,IANUM,IDUST,MSKERR,MSKEN1,MSKEN2,IFNTER,IFATAL)
C
C ** CLEAR COEFFICIENTS IN CURRENT LINE
C
DO 5002 I=1,IDXMAX
5002 TABLEAU(LSTDLM+I)=0.
C
C ** GET THE NEXT INPUT
GOTO 13
C
C
C ** IDENTIFY THE UNRESTRICTED VARS AND THE VARS WITH
C ** UPPER BOUND <= LOWER BOUND
C
5005 IDXSUP=0
DO 5010 I=1,IDXMAX
VALUE=BOUND(I,2)-BOUND(I,1)
IF(VALUE.GE.0.) GO TO 5010
IDXSUP=IDXSUP+1
ISUP1(IDXSUP)=I
5010 CONTINUE
IF(IDXSUP.EQ.0) GO TO 5030
IWRONG=1
WRITE(6,4990)
5015 IDIF=IDXSUP
ICXF=1
IDXL=MINO(7,IDIF)
5020 WRITE(6,4990) (IVARNM(ISUP1(I)),I=ICXF,IDXL)
IF(7.0E,10) GO TO 5030
IDXF=IDXL+1
IDIF=IDIF-7
IDXL=ICXF+MINO(7,IDIF)-1
GOTO 5020
5030 IF(LOOP.EQ.1) GO TO 5041
LOOP=1
IDXSUP=0
DO 5040 I=1,IDOXM
IF(BOUND(I,1).NE.RMINF.OR.BOUND(I,2).NE.PINF) GO TO 5040
IDXSUP=IDXSUP+1
ISUP1(IDXSUP)=I
5040 CONTINUE
IF(IDXSUP.NE.0) GO TO 5050
WRITE(6,4995)
C
C  CHECK FOR OBJ FUNCTION AND CGNS IN THE MODEL
C  ALSO CHECK FOR ALTERNATIVE OBJ FUNCTIONS
C
5041 IF(ALTOBJ.EQ." ") GO TO 5044
DO 5042 I=1,IROWCT
IF(ICTSM(I,1).NE.ALTOBJ.OR.ITYPE(I).NE.0) GO TO 5042
MINMAX=MFLAG
OBJROW=I
GOTO 5044
5042 CONTINUE
WRITE(6,4997) ALTOBJ
4997 FORMAT(/,5X,"FATAL ERROR : ALT. OBJ. FCN. NOT FOUND : ",A8)
IWRONG=1
5044 IF(MINMAX.NE.0) GO TO 5043
IWRONG=1
WRITE(6,4989)
4989 FORMAT(/,5X,"FATAL ERROR : NO OBJ FUNCTION IN THE MODEL",/)
C  IROWCT2 IS THE # OF OBJECTIVE FUNCTIONS IN THE MODEL
5043 IF((IROWCT-IROWCT2).GE.1) GO TO 5045
IWRONG=1
WRITE(6,4989)
4989 FORMAT(/,5X,"FATAL ERROR : NO CONSTRAINTS IN THE MODEL",/)
C
C  ** CHECK FOR FATAL ERRORS IN THE MODEL
C
5045 IF(IWRONG.EQ.1) GO TO 6000
C
C  ** CHANGE THE SIGN OF THE RHS OF THE NULL LINES
C
DO 5047 I=1,IROWCT
IF(ITYPE(I).EQ.0) RHS(I)=-RHS(I)
5047 CONTINUE
C
GOTO 7000
C
5050 WRITE(6,4992)
GOTO 5015
4990 FORMAT(/,5X,"** FATAL ERROR : THE FOLLOWING VARS ARE **",
1"ASSIGNED UPPER ",/21X,"BOUND LOWER THAN THE LOWER BOUNDS",/)
4992 FORMAT(/,15X,"** UNRESTRICTED VARIABLES **",/)
4995 FORMAT(/,5X,"** NO UNRESTRICTED VARIABLES IN THE MODEL **",/)
** ERRORS IN THE MODEL - WORKING AREAS WILL BE CLEARED -
** INPUT WILL BE REPARSED AFTER EDITING

6000 DO 6100 I=1,STOLN
6100 TABLEAU(I)=0.
   DO 6200 I=1,IDXMAX
   BOUND(I,1)=RMINF
   BOUND(I,2)=PINF
   IWARNM(I)=0
6200 ISUP1(I)=0.
   DO 6300 I=1,IROWCT
   RHS(I)=0.
   ICSTNM(I)=0
   BNOSLK(I)=PINF
6300 ITYPE(I)=0
C
C *********** RETURN CONTROL TO DRIVER ***********
C
WRITE(6,4900)
4900 FORMAT(/,,5X,"FATAL ERRORS IN THE MODEL.",/,,5X,
1"PLEASE EDIT THE MODEL" ,/,,5X,
2"INPUT WILL BE RECOMPILED")
C
RETURN
C
C *********** OUTPUT ***********
C
7000 WRITE(6,4910)
4910 FORMAT(15X,"** VARIABLE LIST **",//)
4920 FORMAT(7(2X,A8))
   IDIF=IDXMAX
   IDXF=1
   IDXL=MINO(7,IDIF)
7010 WRITE(6,4920) (IVARNM(I),I=IDXF,IDXL)
   IF(7.GE.IDIF) GO TO 7020
   IDXF=IDXL+1
   IDIF=IDIF-7
   IDXL=MINO(7,IDIF)
   IDXL=IDXF+IDXL
GOTO 7010
7020 IF(IWRONG.NE.0) RETURN
   DIMROW=0
   DO 8010 I=1,1000
   DO 7060 I=1,IDXMAX
   TMPBNO(I*2-1)=BOUND(I,1)
7060 TMPBNO(I*2)=BOUND(I,2)
RETURN
END
C  ** SUBROUTINE PARSER TRANSFORMS THE INPUT IN AN INTERNAL TARGET **
C LANGUAGE FOR A MORE EFFICIENT SYNTACTICAL ANALYSIS AND
C INFORMATION RETRIEVAL. IT PERFORMS THE LEXICAL ANALYSIS
C AND INFORMS THE USER ABOUT LOWER LEVEL ERRORS (ILLEGAL
C CHARACTERS ETC.). IN THE PARSER A UNIFORM SYMBOL TABLE
C IS CONSTRUCTED AND THE ASSOCIATED INFORMATION (VARIABLE
C NAMES, NUMERICAL ENTRIES, ETC.) IS STORED IN SUPPORTING
C TABLES. **
C
C  ** SUBROUTINE PARSER TRANSFORMS THE INPUT IN AN INTERNAL TARGET **
C LANGUAGE FOR A MORE EFFICIENT SYNTACTICAL ANALYSIS AND
C INFORMATION RETRIEVAL. IT PERFORMS THE LEXICAL ANALYSIS
C AND INFORMS THE USER ABOUT LOWER LEVEL ERRORS (ILLEGAL
C CHARACTERS ETC.). IN THE PARSER A UNIFORM SYMBOL TABLE
C IS CONSTRUCTED AND THE ASSOCIATED INFORMATION (VARIABLE
C NAMES, NUMERICAL ENTRIES, ETC.) IS STORED IN SUPPORTING
C TABLES. **
SUBROUTINE PARSER (BUFFER,ICONTRL,IVARNM,IUST,ICSTNM,MSKEN2, ITRMN1,RANUM,MSKERR,MSKEN1,IPNTER,ICTR,INUM,ITRM,IDN1,IDN2, 2ISTUP,IPATAL,IPR,IROMCT,IANUM,IDXMAX,IROWNK,IEDTRW)
INTEGER BUFFER,ALFASW
DIMENSION BUFFER(81),ICONTRL(50),IVARNM(100),IUST(80,2), 1ICSTNM(100),ITRM(100),RANUM(80),IPNTER(80),IANUM(10),IWORK2(8)
DIMENSION MSKERR(80),MSKEN1(40),IPNTER(80),MSKEN2(80)
DATA IWORK2/"R","O","W","#",44,0/

** INITIALIZATION **
IDXFNT=0
IDUSTR=0
IPRIOD=0
ICONRL=0
IDXBUF=0
IDXSKP=0
IDXNUM=0
NTFLAG=1

** LEXICAL ANALYZER **
190 ALFASW=0
200 IDEXBUF=IDXBUF+1
       IF(IDXBUF.GT.80) GO TO 338
       IF(BUFFER(IDXBUF).NE." ") GO TO 210
       GOTO 200

** DETECTION OF THE IDENTIFIERS **
210 IDXPNT=IDXPN+1
       IPNTER(IDXPNT)=IDXBUF
220 J=IDXBUF+ALFASW
       IF(J.GT.80) GO TO 270
       IF(ALPHA(BUFFER(J)).NE.i) GO TO 280
       NUMSW=0
230 ITEM=ALFASW+i
       IWORK1(ITEM)=BUFFER(J)
       ALFASW=ALFASW+1
       IF(ALFASW.GE.8) GO TO 250
       GOTO 220
250 IDXSKP=IDXSKP+1
       JSKP=J+IDXSKP
       IF(JSKP.GT.80) GO TO 270
       LOGC=ALPHA(BUFFER(JSKP))
       IF(LOGC.EQ.1) GO TO 250
       NUMRC=NUMGET(BUFFER(JSKP))
       IF(NUMRC.NE.-1) GO TO 250
270 LAST=J+IDXSKP-1
       IDXSKP=0

** THE NEXT "IF" DETERMINES WHETHER OR NOT A **
** "A"," IS PART OF A VARIABLE NAME, A ""," **
** IS CONSIDERED TO BE PART OF A VARIABLE NAME **
** IF IT IS BETWEEN TWO NUMBERS **
NEXT=J+1
       IF(NUMRC.EQ.0.OR.BUFFER(J).NE.ITRM(6).OR. 1NUMGET(BUFFER(NEXT)).EQ.-1.OR.ICOMMA3.EQ.1) GO TO 271
       IF(ICOMMA2.EQ.1) ICOMMA2=1
       IF(ICOMMA.EQ.1) ICOMMA2=1
       ICOMMA=1
NUMSW = 0
GOTO 230

C 271 ICOMMA = 0
ICOMMA2 = 0
ICOMMA3 = 0
NUMSW = 0

C ** PACK THE DETECTED IDENTIFIER **
C
CALL PACK(IWORK1,IWORD,ALFASH)

C ** IDENTIFY THE DETECTED IDENTIFIER **
C
CALL IDENTIFY(BUFFER,ICONTRL,IVARNM,ICSTNM,IANUM,IWORK2,
INDEX,NFLAG,LAST,IFATAL,ICTR,INUM,ITAM,ION1,ION2,
2ION0,IRONWT,ICLASS,IRONNM,IDXMAX,IEOTRM)

IDXBUF = LAST

C ** CLEAR THE BUFFER IWORK1 **
C
GO 275 ICLEAR = 1,8
IWORK1(ICLEAR) = 0
275 CONTINUE
GOTO 320

280 NUMRC = NUMGET(BUFFER(J))
IF(NUMRC .EQ. -1) GO TO 315
NUMSW = 1
IF(ALFASH .NE. 0) GO TO 230
NUMSW = 0
NFLAG = 2
COEF = 0.
ISIGN = J
ISTART = J
JJ = J

290 BUFFER(JJ) = NUMRC
295 JJ = JJ + 1
IF(JJ .GT. 80) GO TO 296
NUMRC = NUMGET(BUFFER(JJ))
IF(NUMRC .NE. -1) GO TO 290

296 IF(IPRIOO .NE. 1) GO TO 298
JQT = JJ - 1
IF(J .NE. JQT) GO TO 297
IPRIOO = 0
IDXBUF = J

2981 IOXPNT = IOXPNT - 1
GOTO 190
297 IPRIOO = 0
IPNTER(IOXPNT) = IPNTER(IOXPNT) + 1
COEF = 0.
ISIGN = 2
ISTART = J + 1

C ** THE NUMERIC ENTRIES OF THIS SUBSTRING ARE FINISHED
C NOW BUILD THE COEFFICIENT
C
298 CALL GETCOF(BUFFER,ISTART,JJ,ISIGN,COEF)

C ** THE NEXT 4 LINES WILL CONVERT THE COEF FIELD IN BUFFER
C ** BACK TO ALPHA ENTRIES
C
IF(JJ .EQ. ISTART) GO TO 302
IEND = JJ - 1
DO 299 I=ISTART,IEND
BUFFER(I)=NUMOSP(BUFFER(I),IANUM)
299 CONTINUE
C
302 IF(BUFFER(JJ) .EQ. "" ) GO TO 300
301 IDXBUF=JJ-1
ICLASS=INUM
IDXNUM=IDXNUM+1
RANUM(IDXNUM)=COEF
INDEX=INDEX+1
GOTO 320
300 IF(ISIGN .EQ. 2) GO TO 310
ISIGN=2
ISTART=JJ+1
GOTO 295
310 IFATAL=IFATAL+1
WRITE(7,20) IFATAL
C ** POINT THE ERROR **
IDXPN=IDXPN+1
IPNTER(IDXPNT)=JJ
CALL ERROR(R,IARR,IANUM,IDXPNT,MSKRR,MSKEN1,MSKEN2,IPNTER,IFATAL)
IDXPN=IDXPN-1
20 FORMAT(5X,"** FATAL ERROR #",I2," : ILLEGAL SYNTAX")
C ** REINITIATE THE CYCLE **
C GOTO 301
315 IF(ALFASH .NE. 0) GO TO 270
C ** THE NEXT ENTRY MAY BE A TERMINAL SYMBOL **
C LAST=J
CALL IDNTRM(BUFFER,ITRMNL,IROWCT,INDEX,ICLASS,LAST,ITRM,ICONT2,
1 ICSTNM)
IF(INDEX .EQ. 0) GO TO 330
NTFLAG=2
C ** UPDATE THE UNIFORM SYMBOL TABLE **
C IDXBUF=LAST
320 IF(IDXUST .EQ. 0) GO TO 321
IF(ICLASS .EQ. ION2 .AND. IUST(IDXUST,1) .EQ. INUM .AND.
1 RANUM(IUST(IDXUST,2)) .EQ. 1.) GO TO 325
321 IDXUST=IDXUST+1
322 IUST(IDXUST,1)=ICLASS
IUST(IDXUST,2)=INDEX
IF(ICONT2 .EQ. 1) GO TO 338
C ** REINITIATE THE CYCLE **
C GOTO 190
C ** SUPRESS THE COEF=1.
C 325 IDXNUM=IDXNUM-1
IDXPN=IDXPN-1
IPNTER(IDXPNT)=IPNTER(IDXPNT+1)
GOTO 322
C 330 IF(BUFFER(J) .NE. "" ) GO TO 335
IPRIO0=1
JJ=J
IF(J .EQ. 80) GO TO 338
GOTO 295
335 IDXBUF=LAST
IF (BUFFER(J).EQ."\"I\"".AND.IUST(IDXUST,1).EQ.ICTR) GO TO 2961
IFATAL=IFATAL+1
IDXPN=IDXPN+1
IPNTER(IDXPN)=J
CALL ERROR(R,IAFR,IANUM,IDXPN,PSKERR,MSKEN1,MSKEN2,IPNTER,IFATAL)
IDXPN=IDXPN-2
WRITE(7,20) IFATAL
GOTO 190
338 IDXUST=IDXUST+1
IUST(IDXUST,1)=ISTOP
C
RETURN
END
C
FUNCTION IALPHA(X)
INTEGER X
IALPHA=0
IF(X.GE."A".AND.X.LE."Z") IALPHA=1
RETURN
END
C
C
C SUBROUTINE IONTRM IDENTIFIES THE TERMINAL SYMBOLS WHICH
C ARE THE ARITHMETIC AND RELATIONAL OPERATORS, COMMA, AND
C CONTINUATION
C
C******************************************************************************
C
SUBROUTINE IONTRM(BUFFER,ITRPNL,IROWCT,INDEX,ICLASS,LAST,ITRM,
1ICONT2,ICSTNM)
INTEGER BUFFER
DIMENSION BUFFER(80),ITRPNL(10),ICSTNM(100)
INDEX=0
I=0
10 I=I+1
IF (BUFFER(LAST).EQ.ITRPNL(I)) GO TO 12
IF (I.LT.7) GOTO 10
RETURN
12 INDEX=I
ICLASS=ITRM
IF (I.EQ.7) ICONT2=1
IF (I.LT.3.OR.I.GT.5) RETURN
IF (I.EQ.5) GO TO 30
15 IF (LAST.EQ.80) GO TO 20
LAST=LAST+1
IF (BUFFER(LAST).EQ."\"\") GO TO 15
IF (BUFFER(LAST).EQ.ITRPNL(5)) RETURN
IF (BUFFER(LAST).EQ.ITRPNL(7)) RETURN
LAST=LAST-1
20 WRITE(7,100) ICSTNM(IROWCT)
100 FORMAT(5X,"INFORmATIVE I = MISSING AFTER < OR > AT LINE ",X,A8,2X,\","<= OR >= IS ASSUMED")
RETURN
30 IF (LAST.EQ.80) RETURN
LAST=LAST+1
IF (BUFFER(LAST).EQ."\"\") GO TO 30
IF (BUFFER(LAST).EQ.ITRPNL(3)) INDEX=3
IF (BUFFER(LAST).EQ.ITRPNL(4)) INDEX=4
IF (INDEX.EQ.5) LAST=LAST-1
RETURN
END
SUBROUTINE IDNTFY (BUFFER, ICONTRL, IVARNM, ICSTNM, IANUM, IWORK2, INDEX, NTFLAG, LAST, IFATAL, ICTR, INUM, ITRM, ION1, ION2, IWORD, IROCN1, ICLASS, IROWNM, IDXMAX, IEDTRM)
INTEGER BUFFER
DIMENSION BUFFER(80), ICONTRL(50), IVARNM(100), ICSTNM(100), IANUM(10), IWORK2(8)
INDEX = 0
GOTO (8, 30), NTFLAG

** IDENTIFY THE CONTROL WORDS

8 I = 0
10 I = I + 1
   IF (I .GT. 11) GO TO 15
   IF (ICONTRL(I) .NE. IWORD) GO TO 10
   INDEX = I
   ICLASS = ICTR
   IF (I .GE. 10) INDEX = 1 + (I - 10)
RETURN

** DETECT THE ROW NAME IF ANY

15 IF (LAST .EQ. 90) GO TO 30
   LAST = LAST + 1
   IF (BUFFER(LAST) .EQ. " ") GO TO 15
   IF (BUFFER(LAST) .EQ. "!") GO TO 28
   IROWNM = 1
   I = 0
20 I = I + 1
   IF (ICSTNM(I) .EQ. 0) GO TO 25
   IF (ICSTNM(I) .NE. IWORD) GO TO 20
   IFATAL = IFATAL + 1
   WRITE (7, 100) IFATAL
100 FORMAT (5X, "SAME LINE NAME < ", 1A8, IX, " USED FOR A SECOND TIME")
   IFATAL = 1
25 ICLASS = IDN1
   INDEX = I
   ICSTNM(I) = IWORD
RETURN

** IDENTIFY THE VARIABLE NAME OR UPDATE THE IVARNM

28 LAST = LAST - 1
   NTFLAG = 2
30 I = 0
35 I = I + 1
   IF (IVARNM(I) .EQ. 0) GO TO 38
   IF (IVARNM(I) .NE. IWORD) GO TO 35
   GOTO 40
38 IDXMAX = I
40 INDEX = I
   ICLASS = IDN2
   IVARNM(I) = IWORD
   IF (IROWNM .EQ. 1) RETURN
** GENERATE A ROW NAME

IF (IEDTRW .GE. 10) GO TO 43
IWORK2(5) = IANUM(IEDTRW)
M = 5
GOTO 60

43 DO 45 I = 1, 5
IDIF = IEDTRW - I * 10
IF (IDIF .LT. 10) GO TO 50

45 CONTINUE

50 IWORK2(5) = IANUM(I)
IF (IDIF .EQ. 0) IDIF = 10
IWORK2(6) = IANUM(IDIF)
M = 6
CALL PACK (IWORK2, IWORD, M)
IWORK2(5) = 0
IWORK2(6) = 0
ICSTNM(IROWCT) = IWORD
RETURN

END

*******************************************************************************

SUBROUTINE GETCOF BUILDS THE NUMERICAL ENTRIES

*******************************************************************************

SUBROUTINE GETCOF (BUFFER, ISTART, JJ, ISIGN, COEF)
INTEGER BUFFER
DIMENSION BUFFER(80)
IEND = JJ - 1
GOTO (50, 100), ISIGN

50 COEF = BUFFER(IENC)
IDIST = IENC - ISTART
IF (IDIST .LE. 0) RETURN

60 I = 1, IDIST
IEND = IEND - 1
COEF = COEF + BUFFER(IENO) * 10 ** I
CONTINUE
RETURN

*******************************************************************************

** BUILD THE FRACTIONAL PART OF THE NUM ENTRY

*******************************************************************************

100 IEXP = 0
IF (IEND .LT. ISTART) RETURN
DO 150 I = ISTART, IEND
IEXP = IEXP + 1
COEFF = COEFF + CONVRT * 10 ** (-IEXP)
CONTINUE
RETURN
END

FUNCTION NUMDSP(X, Y)
INTEGER X, Y
DIMENSION Y(1)
IF (X .NE. 0) GO TO 9
NUMOSP=Y(10)
RETURN
9    DO 10 I=1,9
      IF(X.LT.EQ.I) NUMOSP=Y(I)
10    CONTINUE
    RETURN
END

******************************************************************************

SUBROUTINE ERROR POINTS BY AN UPPER POINTING ARROW
THE ENTRY THAT CAUSE VIOLATION(S) OF THE GRAMMATICAL
RULES.
******************************************************************************

SUBROUTINE ERROR(R,IARR,IANUM,ITEM,MSKERR,MSKEN1,MSKEN2,
1IPNTER,IFATAL)
INTEGER R
DIMENSION MSKERR(80),MSKEN1(80),MSKEN2(80),IANUM(10),
1IPNTER(90)
MSKERR(IPNTER(ITEM))=IARR
R=IPNTER(ITEM)
IF(IFATAL.GE.10) GO TO 10
MSKEN1(IPNTER(ITEM))=NUMOSP(IFATAL,IANUM)
RETURN
10   IFTL=IFATAL/10
     MSKEN1(IPNTER(ITEM))=NUMOSP(IFTL,IANUM)
     IFTL=IFATAL-10*IFTL
     MSKEN2(IPNTER(ITEM))=NUMOSP(IFTL,IANUM)
     RETURN
END
**SUBROUTINE SPXREL - REAL SIMPLEX OPTIMIZATION AND OUTPUT**

**KEY ARRAYS ARE:**
- **COEF:** The rectangular coefficient array maintained in vector form. The coefficient corresponding to the i-th row and j-th column is stored in location (i-1)*DIMCOL+j. Implicitly, the maximum problem size is such that DIMCOL*DIMROW <= 5000.
- **RHS:** The right-hand-side vector (hence DIMROW <= 200).
- **BOUND:** The lower and upper bound vector. The lower bound for variable j would be stored in 2*j-1, and the upper bound for the same variable would be stored in location 2*j.
- **BASIS:** Maintains the indices of the current basic variables. The starting basis is indicated here and must be composed of slacks and artificials.
- **OBJFCN:** The current objective function. In phase 1, this vector contains the phase 1 objective function, while in phase 2 the principal objective function is moved here.
- **CATNME:** An alpha vector of the constraint names. This array is built by the parser and syntactical analyzer.
- **VARNME:** An alpha vector of the variable names. Again built by the syntactical analyzer.
- **PRNTSW:** A logical vector indicating the various print options.
  1. PRINT VARS
  2. BASIS
  3. TABLEAU
  4. INITIAL
  5. FINAL
- **NULL:** A vector indicating which rows of the tableau correspond to objective functions. Set by the syntactical analyzer (0 = OBJ, 2,3,4 = CONSTRAINT).
- **QUALPT:** A vector indicating the location of the dual variables in the tableau and their signs.
- **SLKPTR:** A vector indicating the slack variables corresponding to the each constraint. (0 = for equality constraints)
- **BASIC:** A vector indicating for each variable whether it is currently basic.
- **ORIENT:** This indicates variables of the form xj <= u, but not bounded from below, and vars variables which enter the basis and go negative.
- **VARBND:** Indicates which bound a variable is currently operating from. (1 = LOWER, 2 = UPPER)

**SOFTWARE INFORMED:**
- **The Logic Flow of the Subroutine (with Respect to the Ordering of the Code) Is According to the Primal Simplex Algorithm. The Dual Procedures Unique to the Dual Simplex Method Are Behind the Primal Routines and Enter and Exit the Primal Routines as Needed. This Also Applies to the Rational Simplex Code in the Subroutine "SPXRTL"."
C** (200) SET-UP FOR PHASE I

C
210 PHASE1=.TRUE.
   DO 230 J=1,DIMCOL
      BASIC(J)=0
      ORIENT(J)=1
230 VARBND(J)=1
   MAXCOL=CIHOOL
   RESULT=0
   ITRCNT=0
   ITERN=0
   IF(METH00.GT.4) METHOD=METH00-4
240 CONTINUE

C** (300) ADJUST FOR LOWER AND UPPER BOUNDS

C
310 DO 360 J=1,DIMCOL
   IF(BOUND((J-1)*2+1).EQ.INFNEG) GO TO 340
320 DO 330 I=1,DIMROW
330 RHS(I)=RHS(I)-BOUND((J-1)*2+1)*COEF((I-1)*DIMCOL+J)
   IF(BOUND((J-1)*2+2).NE.INFPOS)
      BOUND((J-1)*2+2)=BOUND((J-1)*2+1)-BOUND((J-1)*2+2)
      GO TO 360
340 IF(BOUND((J-1)*2+2).EQ.INFPOS) GO TO 360
   DO 350 I=1,DIMROW
350 COEF((I-1)*DIMCOL+J)=-COEF((I-1)*DIMCOL+J)
      ORIENT(J)=-1
      BOUND((J-1)*2+1)=-BOUND((J-1)*2+2)
      BOUND((J-1)*2+2)=INFPOS
      GO TO 320
360 CONTINUE
   IF(METH00.EQ.1) GO TO 1100

C** (400) ESTABLISH GIVEN VARIABLES AS BASIC

C
400 CONTINUE
   DO 460 I=1,DIMROW
      IF(NULL(I).EQ.0) GO TO 460
      BASISI=BASIS(I)
      BASIC(BASISI)=I
      PIVOT=COEF((I-1)*DIMCOL+BASISI)
   DO 420 J=1,DIMCOL
420 COEF((I-1)*DIMCOL+J)=COEF((I-1)*DIMCOL+J)/PIVOT
      RHS(I)=RHS(I)/PIVOT
      PIVOT=OBJFCN(BASISI)
   DO 430 J=1,DIMCOL
430 OBJFCN(J)=OBJFCN(J)-COEF((I-1)*DIMCOL+J)*PIVOT
      Z=Z-RHS(I)*PIVOT
   DO 450 K=1,DIMROW
      IF(K.EQ.1) GO TO 450
      PIVOT=COEF((K-1)*DIMCOL+BASISI)
   DO 440 J=1,DIMCOL
440 COEF((K-1)*DIMCOL+J)=COEF((K-1)*DIMCOL+J)
      -COEF((I-1)*DIMCOL+J)*PIVOT
      RHS(K)=RHS(K)-RHS(I)*PIVOT
450 CONTINUE
460 CONTINUE

C** (500) SELECT PROPER SIMPLEX METHOD

C
500 CONTINUE
   IF(PRNTSW(4)) GO TO 2000
510 CONTINUE
IF(METHOC.EQ.1) GO TO 1600  
IF(ABS(Z)+E.E.INFZRO) GO TO 1100

C  1 REAL DUAL
C  2 RATIONAL DUAL
C  3 REAL PRIMAL
C  4 RATIONAL PRIMAL
C  5 REAL UPPER DUAL
C  6 RATIONAL UPPER DUAL
C  7 REAL UPPER PRIMAL
C  8 RATIONAL UPPER PRIMAL

C** (600) LOOK FOR VARIABLE TO ENTER THE BASIS

600 MINVAL=0.0  
   ITERN=ITERN+1  
   ITRCNT=ITRCNT+1  
   IF(ITRCNT.EQ.ITRLPT.AND.(.NOT.(PRNTSW(4).OR.PRNTSW(5))))  
   * GO TO 2000

605 CONTINUE
   REVURS=.FALSE.  
   WARNR=0  
   MAXNTR=MAXCOL  
   IF(MAXCOL.EQ.DIMCOL) MAXNTR=MAXCOL-NOARTS  
   DO 610 J=1,MAXNTR
   IF(BASIC(J).NE.0) GO TO 613
   IF(BOUNOC(J).NE.INFNEG.AND.OBJFCN(J).GE.MINVAL) GO TO 610
   TEMP1=-ABS(OBJFCN(J))  
   IF(TEMP1.GE.MINVAL) GO TO 610  
   MINVAL=TEMP1  
   VARNTR=J  
   IF(TEMP1.LT.O3JECIN(J)) REVURS=.TRUE.

610 CONTINUE
   IF(MINVAL.GE.0.0) GO TO 1000

C** (650) REVERSE ORIENTATION OF URS ENTERING VARIABLE

650 IF(.NOT.REVURS) GO TO 700  
   ORIENT(VARNTR)=-ORIENT(VARNTR)
   DO 660 I=1,DIMROW
   COEF((I-1)*DIMCOL+VARNTR)=-COEF((I-1)*DIMCOL+VARNTR)
   OBJFCN(VARNTR)=-OBJFCN(VARNTR)

C** (700) LOOK FOR BLOCKING VARIABLE

700 MINVAL=INFPOS  
   VARXIT=0  
   ROWXIT=0  
   THETA1=BOUND((VARNTR-1)*2+2)  
   THETA2=INFPOS  
   THETA3=INFPOS  
   GO 750 I=1,DIMROW
   IF(NULL(I).EQ.0) GO TO 750
   BASISI=BASIS(I)
   IF(COEF((I-1)*DIMCOL+VARNTR).EQ.0.0) GO TO 750
   IF(COEF((I-1)*DIMCOL+VARNTR).LT.0.0) GO TO 720
   IF(BASISI.NE.0.AND.BOUND((BASISI-1)*2+2).EQ.INFNEG) GO TO 750
   THETA2=RHS(I)/COEF((I-1)*DIMCOL+VARNTR)
   GO TO 730

720 IF(BASISI.EQ.0) GO TO 750  
   IF(BOUND((BASISI-1)*2+2).EQ.INFPOS) GO TO 750
   THETA2=RHS(I)/BOUND((BASISI-1)*2+2))/COEF((I-1)*DIMCOL+VARNTR)

730 RATIO=AMIN1(THETA1,THETA2,THETA3)
   IF(RATIO.GE.MINVAL) GO TO 750
ROWXIT=I
VARXIT=BASISI
BNOCNT=.FALSE.
IF(THETA3.EQ.RATIO) BNOCNT=.TRUE.
BNOLMT=.FALSE.
IF(THETA1.EQ.RATIO) BNOLMT=.TRUE.
MINVAL=RATIO
750 CONTINUE
IF(MINVAL.NE.INFPCS) GO TO 800
RESULT=2
WRITE(6,114)
IF(PRNTSW(5)) GO TO 2000
RETURN
C
C** (800) UPPER BOUND ENTERING VARIABLE OR BASIC VARIABLE
C
800 IF(.NOT.(BNOLMT.OR.BNOCNT)) GO TO 900
IF(BNOLMT) J=VARXIT
IF(BNOCNT) J=VARXIT
DO 820 K=1,DIMROW
COEF((K-1)*DIMCOL+J)=-COEF((K-1)*DIMCOL+J)
RHS(K)=RHS(K)+COEF((K-1)*DIMCOL+J)*BOUND((J-1)*2+2)
820 CONTINUE
OBJFCN(J)=-OBJFCN(J)
Z=Z+OBJFCN(J)*BOUND((J-1)*2+2)
VARBNO(J)=3-VARBNC(J)
IF(METHCD.EQ.1) GO TO 1610
IF(BNOLMT) GO TO 600
C
C** (900) PIVOT ON ELEMENT (ROWXIT,VARNTR)
C
900 PIVOT=COEF((ROWXIT-1)*DIMCOL+VARNTR)
DO 910 J=1,DIMCOL
910 COEF((ROWXIT-1)*DIMCOL+J)=COEF((ROWXIT-1)*DIMCOL+J)/PIVOT
RHS(ROWXIT)=RHS(ROWXIT)/PIVOT
IF(.EQ.ROWXIT) GO TO 930
PIVOT=COEF((J-1)*DIMCOL+VARNTR)
DO 920 J=1,DIMCOL
COEF((J-1)*DIMCOL+J)=
* COEF((J-1)*DIMCOL+J)-COEF((ROWXIT-1)*DIMCOL+J)*PIVOT
920 IF(ABS(COEF((J-1)*DIMCOL+J)).LE.INFZRO) COEF((J-1)*DIMCOL+J)=0.0
RHS(J)=RHS(J)-RHS(ROWXIT)*PIVOT
IF(ABS(RHS(J)).LE.INFZRO) RHS(J)=0.0
930 CONTINUE
PIVOT=OBJFCN(VARNTR)
DO 940 J=1,DIMCOL
OBJFCN(J)=OBJFCN(J)-PIVOT*COEF((ROWXIT-1)*DIMCOL+J)
940 IF(ABS(OBJFCN(J)).LT.INFZRO) OBJFCN(J)=0.0
Z=RHS(ROWXIT)*PIVOT
BASIS(ROWXIT)=VARNTR
BASIC(VARNTR)=ROWXIT
IF(VARXIT.NE.0) BASIC(VARXIT)=0
IF(METHCD.EQ.1) GO TO 1600
GO TO 600
C
C** (1000) END PHASE I AND BEGIN PHASE II
C
1000 IF(.NOT.PHASE1) GO TO 1200
ITERTN=0
IF(Z.GE.-INFZRO) GO TO 1010
RESULT=1
WRITE(6,116)
IF(PRNTSW(5)) GO TO 2000
RETURN
** (1010) ELIMINATE ARTIFICIAL VARIABLES FROM BASIS

1010 DO 1090 I=1,DIMROW
    IF(NULL(I).EQ.0) GO TO 1090
    IF(BASIS(I).LE.MAXCOL) GO TO 1090
    DO 1020 JTEMP=1,MAXCOL
        J=MAXCOL+1-JTEMP
        IF(BASIC(J).NL.0) GO TO 1020
        IF(COEF((I-1)*DIMCOL+J).NE.0.0) GO TO 1030
    1020 CONTINUE
    IF(PHASE1) GO TO 1090
    WRITE(b,190) VARNPE(BASIS(I))
    RETURN
1030 BASIC(BASIS(I))=0
    BASIS(I)=J
    PIVOT=COEF((I-1)*DIMCOL+J)
    DO 1040 L=1,DIMCOL
        COEF((I-1)*DIMCOL+L)=COEF((I-1)*DIMCOL+L)/PIVOT
    1040 RHS(I)=RHS(I)/PIVOT
    DO 1060 K=1,DIMROW
        IF(I.EQ.K) GO TO 1060
        PIVOT=COEF((K-1)*DIMCOL+J)
        DO 1050 L=1,DIMCOL
            COEF((K-1)*DIMCOL+L)=
                COEF((K-1)*DIMCOL+L)-COEF((I-1)*DIMCOL+L)*PIVOT
            RHS(K)=RHS(K)-RHS(I)*PIVOT
        1050 CONTINUE
    1060 CONTINUE
    PIVOT=OBJFCN(J)
    DO 1070 L=1,DIMCOL
        OBJFCN(L)=OBJFCN(L)-COEF((I-1)*DIMCOL+L)*PIVOT
        Z=Z-RHS(I)*PIVOT
    1070 CONTINUE
    IF(.NOT.PHASE1) GO TO 1310

** (1100) GENERATE PHASE II OBJECTIVE FUNCTION

1100 CONTINUE
    OBJFCN(L)=-DOEFUCBJR0W-11*DIMCOL+J)*MINMAX
    Z=-RHS(OBJROW)*MINMAX
    PHASE1=.FALSE.
    MAXCUL=DIVCOL-NCARTS
    IF(METHOD.EQ.1) GO TO 400
    GO TO 600

** (1200) OPTIMAL SOLUTION

1200 RESULT=3
    IF(PRNTSW(5)) GO TO 2000
    GO TO 3000

** (1300) START OF DUAL SIMPLEX PROCEDURES

1300 CONTINUE

** (1310) ATTEMPT TO FIND A STARTING DUAL FEASIBLE SOLUTION

1310 DO 1330 J=1,MAXCOL
    IF(OBJFCN(J).GE.0.0) GO TO 1330
    IF(BOUNO((J-1)*2+2).E0.INFPOS) GO TO 1340
    I=1,DIMROW
COEF((I-1)*DIMCOL+J) = -COEF((I-1)*DIMCOL+J)
RHS(I) = RHS(I) + COEF((I-1)*DIMCOL+J)*BOUND((J-1)*2+2)
OBJFCN(J) = OBJFCN(J)
Z = Z + OBJFCN(J)*BOUND((J-1)*2+2)
VARBND(J) = 3 - VARBND(J)
WRITE(6,156) VARNME(J)
CONTINUE
GO TO 1400
WRITE(6,158) VARNME(J)
RETURN

C
C** (1400) DETERMINE EXITING VARIABLE (DUAL SIMPLEX)
C
ITERTN=ITERTN+1
ITRCNT=ITRCNT+1
IF(ITRCNT.EQ.ITRLMT) GO TO 2000
CONTINUE
THETA1=0.0
ROWXIT=0
00 CONTINUE
I=1,0IMROW
IF(NULL(I).EQ.0) GO TO 1430
IF(RHS(I).GE.THETA1) GO TO 1430
IF(BOUND((BASIS(I)-1)*2+1).EQ.INFNEG) GO TO 1430
THETA1=RHS(I)
ROWXIT=I
VARNTR=BASIS(I)
CONTINUE
IF(THETA1.EQ.0.0) GO TO 1209
C
C** (1500) DETERMINE ENTERING VARIABLE (DUAL SIMPLEX)
C
THETA1=INFPOS
VARNTR=0
00 CONTINUE
J=1.MAXCOL
IF(COEF((POXIT-1)*DIMCOL+J).GE.0.0) GO TO 1520
RATIO=-OBJFCN(J)/COEF((P0XIT-1)*DIMCOL+J)
IF(RATIO.GE.THETA1) GO TO 1520
THETA1=RATIO
VARNTR=J
CONTINUE
IF(THETA1.LE.INFPCS) GO TO 900
DO 1540 J=1,0IMCOL
IF(NULL(I).EQ.0) GO TO 1540
IF(BOUND((J-1)*2+1).EQ.INFNEG) GO TO 1540
ORIENT(J) = -ORIENT(J)
VARNTR=J
CONTINUE
RESULT=1
IF(PRNTSW(5)) GO TO 2000
RETURN
C
C** (1600) REVISE TABLEAU TO TREAT UPPER-BOUNDS (DUAL SIMPLEX)
C
CONTINUE
DO 1630 L=1,DIMROW
IF(NULL(L).EQ.0) GO TO 1630
BASIS=L
CONTINUE
WRITE(6,114)
RESULT=1
IF(PRNTSW(5)) GO TO 2000
RETURN
J=BASISI
 GO TO 810
 1610 CONTINUE
 IF(VARBN0(J)=0.2) WRITL(6,166) VARNME(J)
 IF(VAROND(J)=0.1) WRIT6(6,167) VARNME(J)
 DO 1615 K=1,DIMCOL
 1615 COEF((L-1)*DIMCOL+K)=-COEF((L-1)*DIMCOL+K)
 RHS(L)=RHS(L)
 1620 CONTINUE
 IF(ITERTN=1.0) GO TO 1010
 GO TO 1400

C C** (2000) PRINT OPTIONS

C 2000 ITRCNT=0
 ZTEMP=Z*MINMAX
 IF(PHASE1) ZTEMP=ABS(ZTEMP)
 PHASE=1
 IF(.NOT.PHASE1) PHASE=2

C C** (2100) PRINT PRIMAL

C 2100 IF(.NOT.PRNTSW(1)) GO TO 2200
 WRITE(6,116) PHASE,ITERTN,ZTEMP
 WRITE(6,172) GO 2130 J=1,MAXCOL
 VARVAL=BOUND((J-1)*2+1)
 IF(VARVAL.EQ.MINNEG) VARVAL=0.
 IF(VARBN0(J)=0.2) VARVAL=VARVAL*BOUND((J-1)*2+2)
 IF(VARBN0(J)=0.1.AND.BASIC(J).NE.0)
 * VARVAL=VARVAL*RHS(BASIC(J))
 IF(VARBN0(J)=0.2.AND.BASIC(J).NE.0)
 * VARVAL=VARVAL*RHS(BASIC(J))
 IF(VARVAL=0.0) GO TO 2130
 VARVAL=VARVAL*ORIENT(J)
 IF(BASIC(J).NE.0) WRITE(6,138) VARNME(J),VARVAL
 IF(BASIC(J).EQ.0) WRITE(6,140) VARNME(J),VARVAL
 2130 CONTINUE
 WRITE(6,174) CNTNME(I),VARVAL
 2150 CONTINUE
 IF(IDX1=0) WRITE(6,180)

C C** (2200) PRINT BASIS

C 2200 IF(.NOT.PRNTSW(2)) GO TO 2300
 IF(.NOT.PRNTSW(1)) WRITE(6,118) PHASE,ITERTN,ZTEMP
 IF(ITERTN.LE.1) GO 2230
 IF(RESULT.NE.0) GO TO 2230
 IF(BNOLMT) GO TO 2220
 WRITE(6,120) VARNME(VARNTR),VARNME(VARXIT)
 GO TO 2230
 2220 IF(VARBNO(VARNTR)=0.1) WRITE(6,122) VARNME(VARNTR)
 IF(VARBNO(VARNTR)=0.2) WRITE(6,124) VARNME(VARNTR)
 2230 WRITE(6,126) J=0
 DO 2250 I=1,DIMROW

IF(NULL(I).EQ.0) GO TO 2250
J=J+1
VARLST(J)=VARNME(BASIS(I))
IF(J.LT.5) GO TO 2250
WRITE(6,128) (VARLST(K),K=1,5)
J=0
2250 CONTINUE
IF(J.NE.0) WRITE(6,128) (VARLST(K),K=1,J)
C
C** (2300) PRINT TABLEAU
C
2300 IF(.NOT.(PRNTSW(3))) GO TO 2400
IF(.NOT.(PRNTSW(2).OR.PRNTSW(1)))
* WRITE(6,118) PHASE,ITERTN,ZTEMP
WRITE(6,130) (VARNME(J),J=1,MACOL)
DO 2320 I=1,DOMROW
BASIS=BASES(I)
TMPNME=CTNME(I)
IF(BASIS.NE.0) TMPNME=VARNME(BASIS)
TMPOR=1
IF(BASIS.NE.0) TMPOR=ORIENT(BASIS)
2320 WRITE(6,132) TMPINFE I RHS(I)*TMPOR,
* (COEF((I-1)*DOMCOL+J)*TMPOR*ORIENT(J),J=1,MACOL)
2340 CONTINUE
WRITE(6,134)
WRITE(6,136) (OBJFCN(J)*ORIENT(J)*MINMAX,J=1,MACOL)
C
C** (2400) TERMINATE PRINT OPTION
C
2400 CONTINUE
IF(ITERTN.EQ.0) GO TO 510
IF(RESULT.NE.0) GO TO 3000
IF(METH00.EQ.1) GC TO 1410
GO TO 605
C
C** (3000) WRITE OUT OPTIMAL SOLUTION
C
3000 CONTINUE
IF(RESULT.NE.3) RETURN
DO 3010 IMAX=1,70
IF(TITLE(71-IMAX).NE."**") GO TO 3020
3010 CONTINUE
3020 IMAX=71-IMAX
WRITE(6,170) (TITLE(I),I=1,IMAX)
WRITE(6,102)
IF(FINMAX.EQ.1) TMPNME="MAXIMIZE"
IF(MINMAX.EQ.-1) TMPNME="MINIMIZ"
WRITE(6,104) TMPNME,CTNME(CBJROW),2*MINMAX,ITERTN
C
C** (3100) VARIABLE SECTION
C
WRITE(6,106)
DO 3150 J=1,MACOL
VARVAL=BOUND((J-1)*2+1)
IF(VARVAL.LE.INFNEG) VARVAL=0
IF(VARNO(J).EQ.2) VARVAL=VARVAL+BOUND((J-1)*2+2)
IF(VARNO(J).EQ.1.AND.BASIC(J).NE.0)
* VARVAL=VARVAL+RHS(BASIC(J))
IF(VARNO(J).EQ.2.AND.BASIC(J).NE.0)
* VARVAL=VARVAL-RHS(BASIC(J))
IF(VARVAL.EQ.0) GO TO 3150
VARROC=OBJFCN(J)*ORIENT(J)
VARVAL=VARVAL*ORIENT(J)
VARLBO=BOUND((J-1)*2+1)
VARUBD = BOUND((J-1)*2+2)
IF(VARUBD .NE. INFPCS .AND. VARLB0 .NE. INFNEG)
*     VARUBD = VARUNC + VARLB0
IF(ORIENT(J) .EQ. 1) GO TO 3120
VARLB0 = BOUND((J-1)*2+2)
VARUBD = BOUND((J-1)*2+1)
3120 CONTINUE
WRITE(6, 108) VARNME(J), VARVAL, VARLB0, VARUBD, VARRDC
3150 CONTINUE
C
C** (3200) CONSTRAINT SECTION
C
WRITE(6, 110)
DO 3250 I = 1, DIMRCH
IF(NULL(I) .EQ. 0) GO TO 3250
BASISI = IABS(DUALPT(I))
VARVAL = OBJFCN(BASISI) * DUALPT(I) * MINMAX/BASISI
SLKACT = 0
IDX1 = SLKPTR(I)
IF(IDX1 .EQ. 0) GO TO 3230
SLKACT = BOUND((ICX1-1)+2*VARBND(ICX1))
IF(BASIC(IDX1) .EQ. 0) GO TO 3230
IF(VARBND(IDX1) .EQ. 1) SLKACT = SLKACT + RHS(BASIC(IDX1))
IF(VARBND(IDX1) .EQ. 2) SLKACT = SLKACT - RHS(BASIC(IDX1))
3230 WRITE(6, 112) CNTNPE(I), SLKACT, VARVAL, RHS(I)
3250 CONTINUE
C
RETURN
C
END
SPXRTL - RATIONAL SIMPLEX OPTIMIZATION AND OUTPUT

This subroutine is virtually identical to "SPXREL" above, except that the arithmetic operations are performed with rational subroutines — that is, both a numerator and a denominator are maintained for some numeric items. Denominator elements are in COEF - the coefficient matrix (starting with BASEC). RHS - the right-hand-side (starting with BASER). OBJFCN - the current objective function (starting with BASED). The denominators are initialized to 1. - necessitating integral input to these key areas.

The rational subroutines associated with this subroutine are:

1. RATFRT - construction of rational output in 9A1 format
2. RATCNG - rational simplex operation A/B = A/B - C/D * E/F
3. RATSUB - rational subtraction A/B = C/D - E/F
4. RATMLT - rational multiplication A/B = C/D * E/F
5. GCD - greatest common divisor of 2 numbers.

For more complete description of the arrays in this subroutine, consult "SPXREL" above.

SUBROUTINE SPXRTL
COMMON /AREA1/ INFPOS,INFNEG,INFZRO
COMMON /OPT/ COEF(5000),RHS(200),BOUND(400),BASIS(200),
* OBJFCN(200),CONTHME(200),VARNAME(200),
* PRNTSW(5),ITERTM,NOARTS,METHD,HINMAX,Z,OBJROW,
* NULL(200),DIMROW,DIMCOL,TITLE(80)
COMMON /WORK/ DUALPT(200),SLKPTR(200),VARLST(6),OUT(9),
* RATK(6,9),VRNTR,VARXIT,
* ROWXIT,BASISI,IMPORT,PHASE,BASEC,BASER,BASEO,
* MINVAL,REVURS,BNDLMT,BNDCNT,BASIC(200),
* ORIENT(200),VARBNG(200)
INTEGER BASIS,VRNTR,VARXIT,VARNTR,ORIENT,ROWXIT,DIMROW,DIMCOL,
* BASIC,BASISI,IMPORT,PHASE,BASEC,BASER,BASEO,
* DUALP,SLKPTR
REAL INFNEG,INFPOS,INFZRO,MINVAL
LOGICAL PHASEI,REVURS,ENCLMT,BNDCNT,PRNTSW

C 4 * FORMAT STATEMENTS

102 FORMAT(/,10X,"SOLUTION",/,,10X,15("-")/,)
104 FORMAT(3X,"OBJECT",11X,A8,2X,A8,
* /,3X,"Z",16X,E15.4,/,3X,"ITERATIONS",7X,14,/) 106 FORMAT(2X,"VARIABLE SECTION",/,,2X,
* 30("-")",/" NAME",6X,"ACTIVITY LEVEL",4X,
* "LOWER BOUND",4X,"UPPER BOUND",
* 3X,"REDUCED COST",/)
110 FORMAT(/,3X,"CONSTRAINT SECTION",/,,3X,34("-")",/" NAME",5X,"SLACK ACTIVITY",10X,
* "DUAL PRICE",6X,"RHS VALUE",/) 112 FORMAT(1X,A3,3E15.4,/) 114 FORMAT(" SOLUTION UNBOUNDED")
116 FORMAT(" NO FEASIBLE SOLUTION")
120 FORMAT(" AT THIS ITERATION, ",A8," ENTERED THE BASIS ",
* " AND ",A8," LEFT")
122 FORMAT(" AT THIS ITERATION, ",A8," WENT TO ITS ",
* " LOWER BOUND")
124 FORMAT(" AT THIS ITERATION, ",A8," WENT TO ITS ",
* ")
UPPER BOUND

THE CURRENT BASIC VARIABLES ARE -

CURRENT TABLEAU = /

Updated Objective Function Row

OBJECT, 11X,A8, 2X,A8,

CURRENT OBJ VALUE = 1X,9A1

PHASE, 12, "ITERATION -", IS,

CURRENT OBJ VALUE = 1X,9A1

PRIMAL NON-ZERO VARIABLES

DUAL NON-ZERO VARIABLES

NON-INTEGRAL BOUND FOR VARIABLE "A8/

RATIONAL SIMPLEX CANNOT BE USED"

NON-INTEGRAL RHS FOR ROW ",A8/

RATIONAL SIMPLEX CANNOT BE USED"

NON-INTEGRAL OBJECTIVE FCN COEFFICIENT FOR",

"RATIONAL SIMPLEX CANNOT BE USED"

NON-INTEGRAL CONSTRAINT MATRIX ENTRY IN ROW ",A8,

"RATIONAL SIMPLEX CANNOT BE USED"

ATTEMPT TO ELIMINATE ARTIFICIAL ",A8," FROM

"STARTING BASIS FAILED",/;" RE-SOLVE USING "

PRIMAL ALGORITHM"

** (200) SET-UP FOR PHASE I

PHASE1=.TRUE.

DO 230 J=1,0IMCOL

BASIC(J)=0

ORIENT(J)=1

VARBND(J)=1

MAXCOL=OIMCOL

RESULT=0

ITRCNT=0

ITERTN=0

Z=2.0

IF(METHOD.GT.4) METHOD=METHOD-4

GO TO 4500

CONTINUE

** (300) ADJUST FOR LOWER AND UPPER BOUNDS

DO 360 J=1,0IMCOL

IF(BOUND((J-1)*2+1).EQ.INFNEG) GO TO 340

RHS(I)=RHS(I)-BOUND((J-1)*2+1)*COEF((I-1)*DIMCOL+J)

BOUND((J-1)*2+2)=BOUND((J-1)*2+2)-BOUND((J-1)*2+1)

GO TO 360

IF(BOUND((J-1)*2+2).EQ.INFPOS) GO TO 360
DO 350 I=1,0INiCW
350    C OEF((I-1)*DIMCCL+J)=-C OEF((I-1)*DIMCOL+J)
    ORIENT(J)=-1
    BOUND((J-1)*2+1)=-BOUND((J-1)*2+2)
    BOUND((J-1)*2+2)=INFPOS
    GO TO 320
360    CONTINUE.
    IF(METH0D.EQ.2) GO TO 5100
C
C** (400) ESTABLISH GIVEN VARIABLES AS BASIC
C
400    CONTINUE
    DO 460 I=1,DIMRCW
    IF(NULL(I).EQ.0) GO TO 460

    BASISI=BASIS(1)
    BASIC(BASISI)=I
    PIVOT=OEF((I-1)*CIMCCL+BASISI)
    DO 420 J=I,DIMCOL

    420       C OEF((I-1)*DIMCCL*J)=COEF((I-1)*DIMCOL+J)/PIVOT
    RHS(I)=RHS(I)/PIVOT
    PIVOT=OBJFCN(BASISI)
    DO 430 J=1,DIMCCL

    430       OBJFCN(J)=OBJFCN(J)-COEF((I-1)*DIMCOL+J)*PIVOT
    00 440 J=1,0IMCCL

    440       C OEF((K-1)*DIMCCL+J)=COEF((K-1)*DIMCOL+J)*PIVOT
    00 450 K=1,0IMROW

    450       IF(K.EQ.I) GO TO 450
    PIVOT=OEF((K-1)*CIMCCL+BASISI)
    DO 440 J=1,DIMCCL

    440       COEF((K-1)*DIMCCL+J)=COEF((K-1)*DIMCOL+J)
    00 450 K=1,0IMROW

    450       PIVOT=COLF((K-1)*CIMCOL+PASISI)
    DO 440 J=1,0IMCCL

    440       COEF((K-1)*DIMCCL+J)=COEF((K-1)*DIMCOL+J)
    -COEF((I-1)*DIMCOL+J)*PIVOT
    RHS(K)=RHS(K)-RHS(I)*PIVOT
    00 450 K=1,0IMROW

    450       CONTINUE
    460    CONTINUE
C
C** (500) SELECT PROPER SIMPLEX METHOD
C
500    CONTINUE
    IF(PRNTSH(4).AND.(METH0D.EQ.2.OR.METH0D.EQ.4)) GO TO 6000
510    CONTINUE
    IF(METH0D.EQ.2) GO TO 5600
    IF(METH0D.EQ.4.AND.ABS(Z).LE.INFZRO) GO TO 5101
    IF(METH0D.EQ.4) GO TO 4600
C
C 1 REAL DUAL
C 2 RATIONAL DUAL
C 3 REAL PRIMAL
C 4 RATIONAL PRIMAL
C 5 REAL UPPER DUAL
C 6 RATIONAL UPPER DUAL
C 7 REAL UPPER PRIMAL
C 8 RATIONAL UPPER PRIMAL
C
C** (4500) INITIALIZATION FOR RATIONAL SIMPLEX
C
4500    CONTINUE
    BASEC=DIMROW*DIMCOL
    BASER=DIMROW
    BASEO=DIMCOL
    DO 4510 I=1,DIMROW
    RHS(BASER+I)=1.
    ITEMP=RHS(I)
    IF(ITEMP.EQ.RHS(I)) GO TO 4510
    WRITE(6,184) CNTNME(I)
    RETURN
4510 CONTINUE
DO 4520 I=1,OIMCOL
OBJFCN(JASEO+I)=1.
ITEMP=OBJFCN(I)
IF(ITEMP.EQ.OBJFCN(I)) GO TO 4520
WRITE(6,186) VARNME(I)
RETURN
4520 CONTINUE
DO 4540 I=1,0IMROW
DO 4530 J=1,OIMCOL
COEF(BASEC+(I-1)*OIMCOL+J)=1.
ITEMR=COEF((I-1)*OIMCOL+J)
IF(ITEMP.EQ.COEF((I-1)*OIMCOL+J)) GO TO 4530
WRITE(6,188) CNMNE(I),VARNME(J)
RETURN
4530 CONTINUE
4540 CONTINUE
ZON=1.
DO 4550 J=1,OIMCOL
IF(BOUND(J*2-1).EC.INFNEG) GO TO 4545
IF(BOUND(J*2-1).NE.INT(BOUND(J*2-1))) GO TO 4560
4545 IF(BOUND(J*2).EQ.INFPOS) GO TO 4550
IF(BOUND(J*2).NE.INT(BOUND(J*2))) GO TO 4560
4550 CONTINUE
GO TO 240
4560 WRITE(6,182) VARNME(J)
RETURN
C** (4600) LOOK FOR VARIABLE TO ENTER THE BASIS
C** 4600 MINVAL=0.0
ITERTN=ITERTN+1
ITRCNT=ITRCNT+1
IF(ITRCNT.EQ.ITRLMT.AND.(.NOT.(FRNTSW(4).OR.PRNTSW(5)))) GO TO 6000
4605 CONTINUE
REVURS=.FALSE.
VARNTR=0
MAXNTR=MAXCOL
IF(MAXCOL.EQ.OIMCOL) MAXNTR=MAXNTR-NOARTS
DO 4610 J=1,MAXNTR
IF(BASIC(J).NE.0) GO TO 4610
IF(BOUND(J*2-1).NE.INFNEG.AND.
. AND. (OBJFCN(J)/OBJFCN(BASEO+J)).GE.MINVAL) GO TO 4610
TEMR1=-ABS(OBJFCN(J)/OBJFCN(BASEO+J))
IF(TEMR1.GE.MINVAL) GO TO 4610
MINVAL=TEMR1
VARNTR=J
IF(TEMR1.LT.OBJFCN(J)/OBJFCN(BASEO+J)) REVURS=.TRUE.
4610 CONTINUE
IF(MINVAL.GE.0.0) GO TO 5009
C** (4650) REVERSE ORIENTATION OF VARS ENTERING VARIABLE
C** 4650 IF(.NOT.REVURS) GO TO 4700
ORIENT(VARNTR)=-ORIENT(VARNTR)
DO 4660 I=1,0IMROW
COEF((I-1)*OIMCOL+VARNTR)=-COEF((I-1)*OIMCOL+VARNTR)
OBJFCN(J)=-OBJFCN(J)
C** (4700) LOOK FOR BLOCKING VARIABLE
C** 4700 MINVAL=INFPOS
VARXIT=0
ROWXIT=0
THETA1=BOUND((VARNTR-1)*2+2)
THETA2=INFPOS
THETA3=INFPOS
DO 4750 I=1,DIMROW
IF(NULL(I).EQ.0) GO TO 4750
BASISI=BASIS(I)
IF(COEF((I-1)*DIMCOL+VARNTR).EQ.0.0) GO TO 4750
IF(COEF((I-1)*DIMCOL+VARNTR).LT.0.0) GO TO 4720
IF(BASISI.NE.0.AND.BOUND((BASISI-1)*2+1).EQ.INFPOS) GO TO 4750
THETA2=RHS(I)/COEF((I-1)*DIMCOL+VARNTR)*COEF(BASEC4-(I-1)*DIMCOL+VARNTR)/RHS(BASER+I)
GO TO 4730
4720 IF(BASISI.EQ.0) GO TO 4750
IF(BOUND((BASEC-1)*2+2).EQ.INFPOS) GO TO 4750
THETA3=(RMS(I)/RHS(3ASER+I)-BOUND((BASEC-1)*2+2))/COEF((I-1)*DIMCOL+VARNTR)*COEF(BASEC4-(I-1)*DIMCOL+VARNTR)
4730 RATIO=MIN1(THETA1,THETA2,THETA3)
IF(RATIO.GE.MINVAL) GO TO 4750
ROWXIT=I
VARXIT=BASISI
BNUINF=.FALSE.
IF(THETA3.EQ.RATIO) BNUINF=.TRUE.
BNCLMT=.FALSE.
IF(THETA1.EQ.RATIO) BNCLMT=.TRUE.
MINVAL=RATIO
4750 CONTINUE
IF(MINVAL.NE.INFPOS) GO TO 4800
RESULT=2
WRITE(6.114)
IF(PRNTSW(5)) GO TO 6000
RETURN
C
C** (4800) UPPER BOUND ENTERING VARIABLE OR BASIC VARIABLE
C
4800 IF(.NOT.(BNOLMT.CR.BNOCNT)) GO TO 4900
IF(BNOCNT) J=VARXIT
IF(BNOLMT) J=VARNTR
4810 DO 4920 I=1,DIMROW
COEF((I-1)*DIMCOL+J)=-COEF((I-1)*DIMCOL+J)
CALL RATCNG(RHS(I),RHS(BASER+I),-COEF((I-1)*DIMCOL+J),
* COEF(BASEC4+(I-1)*DIMCOL+J),BOUND((J-1)*2+2),1.)
4820 CONTINUE
OBJFCN(J)=OBJFCN(J)
CALL RATCNG(z,0,OBJFCN(J),OBJFCN(BASEO+J),
* BOUND(J-1)*2+2),1.)
VARBNO(J)=3-VARBNO(J)
IF(METHO.0.2) GG TO 5610
IF(BNOLMT) GO TO 4600
C
C** (4900) PIVOT ON ELEMENT (ROWXIT,VARNTR)
C
4900 PVTNUM=COEF((ROWXIT-1)*DIMCOL+VARNTR)
PVTDEN=COEF(BASEC4+ROWXIT-1)*DIMCOL+VARNTR)
DO 4910 J=1,DIMROW
CALL RATMLT(COEF((ROWXIT-1)*DIMCOL+J),COEF(BASEC4+
1 (ROWXIT-1)*DIMCOL+J),COEF((ROWXIT-1)*DIMCOL+J),
2 COEF(BASEC4+ROWXIT-1)*DIMCOL+J),PVTDEN,PVTNUM)
4910 CONTINUE
CALL RATMLT(RHS(ROWXIT),RHS(BASER+ROWXIT),
* RHS(ROWXIT),RHS(BASER+ROWXIT),
* PVTDEN,PVTNUM)
4930 CONTINUE
IF(I.EQ.ROWXIT) GO TO 4930
PVTNUM=COEF((I-1)*DIMCOL+VARNTR)
PVTDEN=COEF(BASEC+(I-1)*DIMCOL+VARNTR)
DO 4920 J=1,DIMCOL
CALL RATCNG(COEF((I-1)*DIMCOL+J),COEF(BASEC+(I-1)*DIMCOL+J),COEF(BASEC+(ROWXIT-1)*DIMCOL+J),COEF(BASEC+(ROWXIT-1)*DIMCOL+J),COEF((I-1)*DIMCOL+J),COEF((I-1)*DIMCOL+J))
4920 CONTINUE
IF(ABS(COLF((I-1)*DIMCOL+J)).LE.INFZRO) COEF((I-1)*DIMCOL+J)=0.0
CALL RATCNG(RHS(I),RHS(BASER+II),RHS(ROWXIT),RHS(BASER+II),RHS(ROWXIT),RHS(BASER+II))
IF(AABS(RHS(I)).LE.INFZRO) RHS(I)=0.0
4930 CONTINUE
PVTNUM=OBJFCN(VARNTR)
PVTDEN=OBJFCN(BASEO+VARNTR)
DO 4940 J=1,DIMCOL
CALL RATCNG(OBJFCN(J),OBJFCN(BASEO+J),PVTNUM,PVTDEN,
1 COEF((ROWXIT-1)*DIMCOL+J),COEF(BASEC+(ROWXIT-1)*DIMCOL+J))
4940 CONTINUE
IF(ABS(OBJFCN(J)).LT.INFZRO) OBJFCN(J)=0.0
CALL RATCNG(Z,ZDE,RHS(ROWXIT),RHS(BASER+ROWXIT),PVTNUM,
1 PVTDEN)
BASIS(ROWXIT)=VARNTR
BASIC(VARNTR)=ROWXIT
IF(VAXIT.NE.0) BASIC(VARXIT)=0
IF(METHOD.EQ.2) GO TO 5600
GO TO 4600
C C** (5000) END PHASE I AND BEGIN PHASE II
C 5000 IF(NOT.PHASE1) GC TO 5200
ITERN=0
IF(Z.GE.-INFZRO) GO TO 5010
RESULT=1
WRITE(6,116)
IF(PRNTSW(5)) GO TO 6000
RETURN
C C** (5010) ELIMINATE ARTIFICIAL VARIABLES FROM BASIS
C 5010 DO 5090 I=1,DIMROW
IF(NULL(I).EQ.0) GO TO 5090
IF(BASIS(I).LE.MAXCOL) GO TO 5090
DO 5020 JTEMP=1,MAXCOL
J=MAXCOL+I-JTEMP
IF(BASIC(J).NE.0) GO TO 5020
IF(COEF((I-1)*DIMCOL+J).NE.0.0) GO TO 5030
5020 CONTINUE
IF(PHASE1) GO TO 5090
WRITE(6,190) VARNAME(BASIS(I))
RETURN
5030 BASIC(BASIS(I))=0
BASIS(I)=J
PVTNUM=COEF((I-1)*DIMCOL+J)
PVTDEN=COEF(BASEC+(I-1)*DIMCOL+J)
DO 5040 L=1,DIMCOL
5040 CALL RATMLT(COEF((I-1)*DIMCOL+L),COEF(BASEC+(I-1)*DIMCOL+L),COEF((I-1)*DIMCOL+L),COEF(BASEC+(I-1)*DIMCOL+L),COEF((I-1)*DIMCOL+L),COEF(BASEC+(I-1)*DIMCOL+L),PVTDEN,
1 PVTDEN)
CALL RATMLT(RHS(I),RHS(BASER+I),RHS(I),RHS(BASER+I),RHS(BASER+I),RHS(BASER+I))
DO 5060 K=1,DIMCOL
IF(I.EQ.K) GO TO 5060
PVTNUM=COEF((K-1)*DIMCOL+J)
PVTDEN=COEF(BASEC+(K-1)*DIMCOL+J)
DO 5050 L=1,DIMCOL
! 5050 CALL RATCHG(COEF((K-1)*DIMCOL+L),COEF(BASEC+(K-1)*DIMCOL+L),
C 1 COEF((I-1)*DIMCOL+L),COEF(BASEC+(I-1)*DIMCOL+L),PVTNUM,
C 2 PVTDEN)
! CALL RATCHG(RHS(K),RHS(BASER+K),
RHS(I),RHS(BASER+I),
* PVTNUM,PVTDEN)
5060 CONTINUE
PVTNUM=OBJFCN(J)
PVTDEN=OBJFCN(BASEO+J)
DO 5070 L=1,DIMCOL
! 5070 CALL RATCHG(OBJFCN(L),OBJFCN(BASEO+L),COEF((I-1)*DIMCOL+L),
C 1 COEF(BASEC+(I-1)*DIMCOL+L),PVTNUM,PVTDEN)
! CALL RATCHG(Z,ZDEN,RHS(I),RHS(BASER+I),PVTNUM,PVTDEN)
BASIC(J)=I
5090 CONTINUE
IF(.NOT.PHASE1) GO TO 5310

C
C** (5100) GENERATE PHASE II OBJECTIVE FUNCTION
C
5100 CONTINUE
DO 5110 J=1,DIMCOL
OBJFCN(J)=COEF((OBJROW-1)*DIMCOL+J)*MINMAX
5110 OBJFCN(BASEO+J)=COEF(BASEC+(OBJROW-1)*DIMCOL+J)
Z=RHS(OBJROW)*MINMAX
ZDEN=RHS(BASEO+OBJROW)
PHASE1=.FALSE.
MAXCOL=DIMCOL-NCARTS
IF(METH00.E'2) GO TO 400
GO TO 4500

C
C** (5200) OPTIMAL SOLUTION
C
5200 RESULT=3
IF(PRNTSW(5)) GO TO 6000
GO TO 7000
C
C** (5300) START OF RATIONAL DUAL SIMPLEX PROCEDURES
C
5300 CONTINUE
C
C** (5310) ATTEMPT TO FIND A STARTING DUAL FEASIBLE BASIS
C
5310 DO 5330 J=1,MACCOL
IF(OBJFCN(J).GE.0.0) GO TO 5330
IF(BOUND((J-1)*2+2).EQ.INFPOS) GO TO 5340
DO 5320 I=1,DIMROW
COEF((I-1)*DIMCOL+J)=COEF((I-1)*DIMCOL+J),
* -COEF((I-1)*DIMCOL+J),COEF(BASEC+(I-1)*DIMCOL+J),
* BOUND((J-1)*2+2),1.)
OBJFCN(J)=OBJFCN(J)
CALL RATCHG(Z,ZDEN,-OBJFCN(J),OBJFCN(BASEO+J),
* BOUND((J-1)*2+2),1.)
VARBND(J)=3-VARBND(J)
WRITE(6,166) VARNPE(J)
5330 CONTINUE
GO TO 5400
5340 WRITE(6,166) VARNPE(J)
RETURN
C
C** (5400) DETERMINE EXITING VARIABLE (RAT. DUAL SIMPLEX)
C
5400 ITERN=ITERN+1
ITRCN=ITRCNT+1
IF(ITRCN.EQ.ITFLMT) GO TO 6000

5410 CONTINUE
THETA1=0.0
ROWXIT=0
DO 5430 I=1,JIMROW
   IF(NULL(I).EQ.0) GO TO 5430
   IF(ITHETA1.LT.RHS(I)/RHS(BASER+I)) GO TO 5430
   IF(BOUNDUBASIS(I)-1)*2+1).EQ.INFNEG) GO TO 5430
   THETA1=RHS(I)/RHS(BASER+I)
   ROWXIT=I
5430 CONTINUE
IF(THETA1.LT.0.0) GO TO 5200

C
C** (5500) DETERMINE ENTERING VARIABLE (RAT. DUAL SIMPLEX)
C
5500 CONTINUE
THETA1=INFPOS
VARNTR=0
DO 5520 J=1,MAXCOL
   IF(COLF((ROWXIT-1)*DIMCOL+J).GE.0.0) GO TO 5520
   RATIO=-OBJFCN(J)/OBJFCN(BASEO+J)/COEF((ROWXIT-1)*DIMCOL+J)*COEF(BASEO+(ROWXIT-1)*DIMCOL+J)
   IF(RATIO.GE.THETA1) GO TO 5520
   THETA1=RATIO
   VARNTR=J
5520 CONTINUE
IF(THETA1.LE.INFPCS) GO TO 5900

5540 J=1,DIMCOL
   IF(COLF((ROWXIT-1)*DIMCOL+J).LE.0.0) GO TO 5549
   IF(BOUND((J-1)*2+1).NE.INFNEG) GO TO 5540
   ORIENT(J)=-ORIENT(J)
   VARNTR=J
5530 K=1.DIMROW
   COEF((K-1)*DIMCCL+J)=-COEF((K-1)*DIMCOL+J)
   OBJFCN(J)=-OBJFCN(J)
   GO TO 4900
5540 CONTINUE
WRITE(6,114)
RESULT=1
IF(PRNTSW(5)) GO TO 6000
RETURN
C
C** (5600) REVERSE TABLEAU TO TREAT UPPER BOUNDS (RAT. DUAL)
C
5600 CONTINUE
DO 5620 L=1,JIMROW
   IF(NULL(L).EQ.0) GO TO 5620
   BASISI=BASIS(L)
   RHS(L)/RHS(BASER+L).LE.BOUND(((BASISI-1)*2+2)) GO TO 5620
   J=BASISI
   GO TO 4910
5610 CONTINUE
   IF(VARNME(J).EQ.2) WRITE(6,166) VARNME(J)
   IF(VARNME(J).EQ.1) WRITE(6,167) VARNME(J)
   DO 5615 K=1,DIMCOL
   COEF((L-1)*DIMCOL+K)=-COEF((L-1)*DIMCOL+K)
   RHS(L)=-RHS(L)
5615 CONTINUE
IF(ITERTN.LT.0) GO TO 5010
GO TO 5400
C
C** (6000) PRINT OPTIONS
C

6000  ITRCNT=0
    ZTEMP=Z*MINMAX
    IF(PHASE1) ZTEMP=ABS(ZTEMP)
    PHASE=1
    IF(.NOT.PHASE1) PHASE=2

C** (6100) PRINT PRIMAL

6100  IF(.NOT.PRNTSW(1)) GO TO 6200
    CALL RATFRT(ZTEMP,ZDEN,OUT)
    WRITE(6,156) PHASE,ITERTN,(OUT(K),K=1,9)
    WRITE(6,172)
    DO 6130 J=1,MAXCOL
        TMPOEN=1.
        IF(BASIC(J).NE.0) TMPOEN=RHS(BASEI+BASIC(J))
        VARVAL=BOUND((J-1)*2+1)*TMPOEN
        IF(VARVAL.LE.INFNEG) VARVAL=0.*
        IF(VARBND(J).EQ.2) VARVAL=VARVAL+BOUND((J-1)*2+2)*TMPOEN
    CONTINUE
    WRITE(6,174)
    IDX1=0
    DO 6150 I=1,DIMROW
        IF(NULL(I).EQ.0) GO TO 6150
        BASISI=IABS(DUALPT(I))
        VARVAL=OBJFCN(BASISI)*DUALPT(I)*MINMAX/3ASISI
        IF(VARVAL.EQ.0) GC TO 6150
        IDX1=IDX1+1
        CALL RATFRT(VARVAL,OBJFCN(BASEO+BASISI),OUT)
        WRITE(6,178) CNTNPE(I),(OUT(K),K=1,9)
    CONTINUE
    IF(IDX1.>0) WRITE(6,180)

C** (6200) PRINT BASIS

6200  IF(.NOT.PRNTSW(2)) GO TO 6300
    CALL RATFRT(ZTEMP,ZDEN,OUT)
    IF(.NOT.PRNTSW(1))
        WRITE(6,156) PHASE,ITERTN,(OUT(K),K=1,9)
    IF(RESULT.LT.1) GO TO 6230
    IF(RDCNT.NE.0) GO TO 6230
    IF(BNDLMT) GO TO 6220

6220  IF(VARBC(VARNTR).EQ.1) WRITE(6,122) VARNME(VARNTR),VARNME(VARXII)

6230  WRITE(6,126)
    J=0
    DO 6250 I=1,DIMROW
        IF(NULL(I).EQ.0) GO TO 6250
        J=J+1
        VARLST(J)=VARNME(PASIS(I))
        IF(J.LT.5) GO TO 6250
        WRITE(6,128) (VARLST(K),K=1,5)
        J=0
CONTINUE
IF(J.NE.0) WRITE(6,128) (VARLST(K),K=1,J)
C
C** (6300) PRINT TABLEAU
C
6300 IF(.NOT.PRTSW(1)) GO TO 6400
CALL RATPRT(ZTENP.ZOEN,OUT)
IF(.NOT.(PNTSW(2).OR.FkNTSW(1)))
* WRITE(6,156) PHASE,ITERN,(OUT(K),K=1,9)
WRITE(6,130) (VARNME(J),J=1,MAXCOL)
DO 6350 I=1,INDEX
BASIS=BASIS(I)
TMPNAM=CTNMNE(I)
IF(BASIS.NE.0) TMPNAM=VARNME(BASIS)
TMPORT=1
IF(BASIS.NE.0) TMPOR=ORIENT(BASIS)
IDX1=(MAXCOL-1)/5+1
CALL RATPRT(RHS(I)*TMPOR,RHS(BASE+I),OUT)
DO 6310 K=1,9
6310 RATWRK(1,K)=OUT(K)
DO 6340 IDX2=1,IDX1
IDX3=MAXCOL-(IDX2-1)*5
IF(IDX2.LT.IDX1) IDX3=5
DO 6330 IDX4=1,IDX3
IDX5=5*(IDX2-1)+IDX4
CALL RATPRT(coEF((I-1)*DIMCOL+IDX5)*TMPOR*ORIENT(IDX5),
* OBJFCN(BASEC+(I-1)*DIMCOL+IDX5),OUT)
DO 6320 K=1,9
6320 RATWRK(IDX4+1,K)=OUT(K)
6330 CONTINUE
IDX4=IDX3+1
IF(IDX2.EQ.1)
* WRITE(6,153) TMPNAM,((RATWRK(K,L),L=1,9),K=1,IDX4)
IF(IDX2.GT.1)
* WRITE(6,160) ((RATWRK(K,L),L=1,9),K=1,IDX4)
DO 6335 K=1,9
6335 RATWRK(1,K)=""
6340 CONTINUE
6350 CONTINUE
WRITE(6,134)
DO 6380 IDX2=1,IDX1
IDX3=MAXCOL-(IDX2-1)*5
IF(IDX2.LT.IDX1) IDX3=5
DO 6370 IDX4=1,IDX3
IDX5=5*(IDX2-1)+IDX4
CALL RATPRT(OBJFCN(IDX5)*ORIENT(IDX5)*MINMAX,
* OBJFCN(BASEC+IDX5),OUT)
DO 6360 K=1,9
6360 RATWRK(IDX4,K)=OUT(K)
6370 CONTINUE
WRITE(6,160) ((RATWRK(K,L),L=1,9),K=1,IDX3)
6380 CONTINUE
C
C** (6400) TERMINATE PRINT OPTION
C
6400 CONTINUE
IF(ITERN.EQ.0) GC TO 510
IF(RESULT.NE.0) GO TO 7000
IF(METHOE.EQ.2) GO TO 5410
GO TO 4605
C
C** (7000) WRITE OUT OPTIMAL SOLUTION
C
7000 CONTINUE
IF(RESULT.\text{NE}.3) RETURN

DO 7010 IMAX=1,70
IF(TITLE(71-IMAX).\text{NE}.""") GO TO 7020

7010 CONTINUE

7020 IMAX=71-IMAX
WRITE(6,170) (TITLE(I),I=1,IMAX)
WRITE(6,102)
CALL RAPTPT(Z*MINMAX,ZDEN,OUT)
IF(MINMAX.\text{EQ}1) TMPNAME="MAXIMIZE"
IF(MINMAX.\text{EQ}-1) TMPNAME="MINIMIZE"
WRITE(6,150) TMPNAME,CATNAME(CBJROW),\{OUT(K),K=1,9\}

C

C** (7100) VARIABLE SECTION
C
WRITE(6,106)
DO 7100 J=1,MAXCOL
TMPOEN=1.
IF(BASIC(J).\text{NE}.0) TMPOEN=RHS(BASE+BASE(J))
VARVAL=BOUND((J-1)*2+1)*TMPOEN
IF(VARUS(J).\text{EQ}.2) VARVAL=VARVAL+BOUND((J-1)*2+2)*TMPOEN
IF(VARUS(J).\text{EQ}.1) VARVAL=VARVAL+BOUND((J-1)*2+1)*TMPOEN
VARLBD=-BOUND((J-1)*2+1)
VARUBD=BOUND((J-1)*2+2)
IF(VARUBD.\text{EQ}.1) VARUBD=VARUBD*VARUS(J)
VARLBD=-BOUND((J-1)*2+1)
VARUBD=BOUND((J-1)*2+2)
IF(VARUBD.\text{EQ}.0) GO TO 7150
VARRDC=OBJFCN(J)*ORIENT(J)
VARVAL=VARVAL+BOUND((J-1)*2+1)*TMPOEN
IF(VARUS(J).\text{EQ}.1) VARVAL=VARVAL+BOUND((J-1)*2+2)*TMPOEN
IF(VARUS(J).\text{EQ}.2) VARVAL=VARVAL+BOUND((J-1)*2+1)*TMPOEN
VARLBD=-BOUND((J-1)*2+1)
VARUBD=BOUND((J-1)*2+2)
IF(VARUBD.\text{EQ}.1) VARUBD=VARUBD*VARUS(J)
VARLBD=-BOUND((J-1)*2+1)
VARUBD=BOUND((J-1)*2+2)
IF(VARUBD.\text{EQ}.0) GO TO 7150

7120 CONTINUE

CALL RAPTPT(VARVAL,TMPOEN,OUT)
DO 7110 K=1,9
7110 RAPWRK(1,K)=OUT(K)
DO 7130 K=1,9
7130 RAPWRK(2,K)=OUT(K)

7140 RAPWRK(3,K)=OUT(K)
DO 7150 K=1,9
7150 CONTINUE

C

C** (7200) CONSTRAINT SECTION
C
WRITE(6,110)
DO 7250 I=1,DIMROW
IF(NULL(I).\text{EQ}.0) GO TO 7250
CALL RAPTPT(RHS(I),RHS(BASE+I),OUT)
DO 7220 J=1,9
7220 RAPWRK(1,J)=OUT(J)

7230 RAPWRK(2,J)=OUT(J)
TMPOEN=1.
IDX1 = SLKPTR(I)
SLKACT = 0.
IF (IDX1.EQ.0) GO TO 7240
SLKACT = BOUND(I) * 2 * VARBND(IDX1)
IF (BASIC(IDX1).EQ.0) GO TO 7240
TMPDEN = RHS(3ASIC) * BASIC(IDX1)
SLKACT = SLKACT * TMPDEN
IF (VARBND(IDX1).EQ.1)
* SLKACT = SLKACT + RHS(BASIC(IDX1))
IF (VARBND(IDX1).EQ.2)
* SLKACT = SLKACT - RHS(BASIC(IDX1))
7240 CALL RATPRT(SLKACT, TMPDEN, OUT)
DO 7245 J = 1, 9
7245 RATWRK(1, J) = OUT(J)
WRITE (6, 154) CNTNME(I), (RATWRK(K, J), J = 1, 9), K = 1, 3
7250 CONTINUE
RETURN
END
SUBROUTINE RATPRT(RATNUM, RATDEN, OUT)
C    RATPRT - RATIONAL PRINT ROUTINE
C    THIS SUBROUTINE TAKES A REAL NUMERATOR AND A REAL
C
COMMON /AREA1/ INFPOS, INFNEG, INFZRO
REAL INFPCS, INFNEG, INFZK0
DIMENSION OUT(9)
IF (RATNUM.EQ.0) GO TO 60
IF (RATNUM.EQ.1) GO TO 70
SIGN = 0.
TMPNUM = RATNUM
IF (TMPNUM .LT. 0) SIGN = -1.
TMPNUM = ABS(TMPNUM)
MAX1 = 1
IF (TMPNUM .GE. 10) MAX1 = 2
IF (TMPNUM .GE. 100) MAX1 = 3
IF (TMPNUM .GE. 1000) MAX1 = 4
IF (TMPNUM .GE. 10000) MAX1 = 5
IF (TMPNUM .GE. 100000) MAX1 = 6
IF (TMPNUM .GE. 1000000) MAX1 = 7
IF (SIGN .EQ. -1, I) MAX1 = MAX1 + 1
MAX2 = 1
IF (RATDEN .GE. 10) MAX2 = 2
IF (RATDEN .GE. 100) MAX2 = 3
IF (RATDEN .GE. 1000) MAX2 = 4
IF (RATDEN .GE. 10000) MAX2 = 5
IF (RATDEN .GE. 100000) MAX2 = 6
IF (RATDEN .GE. 1000000) MAX2 = 7
IF (RATDEN .EQ. 1) MAX2 = 0
IF (MAX1 + MAX2 .GT. 8) GO TO 40
DO 10 I = 1, 9
OUT(I) = " 
INC = (9 - (MAX1 + MAX2 + 1)) / 2
ITEMP = TNPNUM
IDX = MAX1 + SIGN
DO 20 I = 1, IDX
OUT(INC + 1 + MAX1 + I) = OSPNUM (ITEMP - (ITEMP / 10) * 10)
ITEMP = ITEMP / 10
IF (SIGN .EQ. -1, I) OUT(INC + 1) = "-
IF (RATDEN .EQ. 1) RETURN
INC = INC + MAX1 + 1
OUT(INC)="/"
ITEMP=RATEN
DO 30 I=1,MAX2
OUT(INC+1+MAX2-I)=DSPNUM(ITEMP=(ITEMP/10)*10)
30
ITEMP=ITEMP/10
RETURN
40 CONTINUE
OUT(I)=OUT(9)=" 
DO 50 I=1,7
50 OUT(I+1)=" 
RETURN
60 OUT(I)=OUT(5)=OUT(9)=" 
OUT(2)="P"
OUT(3)="O"
OUT(4)="S"
OUT(5)="I"
OUT(7)="N"
OUT(8)="F"
RETURN
70 OUT(I)=OUT(5)=OUT(9)=" 
OUT(2)="N"
OUT(3)="E"
OUT(4)="G"
OUT(6)="I"
OUT(7)="N"
OUT(8)="F"
RETURN
RETURN
END
SUBROUTINE RATCNG(A,B,C,D,E,F)
C******************************************************************************
C RATCNG - RATIONAL SIMPLEX OPERATION
C A/B = A/B - C/D * E/F
C******************************************************************************
C A/B = A/B - (C/D)*(E/F)
CALL RATMLT(Q,R,C,D,E,F)
G=A
H=B
CALL RATSUB(A,B,G,H,Q,R)
RETURN
END
SUBROUTINE RATSUB(A,B,C,D,E,F)
C******************************************************************************
C RATSUB - RATIONAL SUBTRACTION OPERATION
C A/E = C/D - E/F
C******************************************************************************
C A/E = C/D - E/F
A=C*D-E*F
B=D*F
G=GCD(A,B)
IF(B.GT.0.) G=-G
A=A/G
B=B/G
RETURN
END
SUBROUTINE RATMLT(A,B,C,D,E,F)
C RATMLT - RATIONAL MULTIPLICATION OPERATION

C
C A/B = C/D * E/F
C
C*********************************************************************
C A/B = (C/D) * (E/F)
A=C*E
B=D*F
G=GCD(A, B)
IF(B.LT.0.) G=-G
A=A/G
B=B/G
RETURN
END

FUNCTION GCD(X, Y)

C*********************************************************************
C
C GCD - GREATEST COMMON DIVISOR
C
*********************************************************************

U=ABS(X)
V=ABS(Y)
IF(V.LT.U) GO TO 1
T=V
V=U
U=T
1 IF(V.EQ.0) GO TO 2
T=INT(U/V)
T=U-V*T
U=V
V=T
GO TO 1
2 GCD=U
IF(GCD.LE.0) GCC=1.
RETURN
END
FINAL REPORT
NSF Grant Number 34075-17476

EZLP: An Interactive Computer Program
for Solving Linear Programming Problems

by

John J. Jarvis, Project Director
T. Ed Dugger, Faculty Associate
Frank H. Cullen, Research Assistant
Chris Papadomitsadopoulos, Research Assistant

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL & SYSTEMS ENGINEERING
ATLANTA, GEORGIA 30332
FINAL REPORT
NSF Grant Number SED75-17476

EZLP: An Interactive Computer Program for Solving Linear Programming Problems

John J. Jarvis, Project Director
V. Ed Unger, Faculty Associate
Frank H. Cullen, Research Assistant
Chris Papaconstadopoulos, Research Assistant

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

September, 1976
ABSTRACT

This report discusses the development of an interactive computer program (EZLP) designed for student-oriented use in solving linear programming problems. The linear programming problem is inputted in the same way as it would be written on a sheet of paper.

The student may select either the (primal) simplex or the dual simplex method; the lower-upper bounded variables procedure with either method; and real (0.5) or rational (1/2) arithmetic for the calculations. EZLP has internal editing capability and is able to read from and write to permanent files.

During execution of EZLP, if the student has difficulty in remembering what to do next, he may utilize a HELP command to obtain general or specific information on EZLP's use.

Since EZLP was developed under a National Science Foundation grant a program listing on computer paper is freely available. A listing on cards or tape is available at a nominal charge to cover expenses. EZLP may not be resold.

Finally, a simplified user's manual suitable for handout to students learning EZLP for the first time is contained in Appendix A.
TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Background ........................................... 1
1.2 Initial Project Effort ................................. 2
1.3 Initial Project Plan ................................. 3
1.4 Follow-on Project Effort ............................ 4
1.5 Machine Dependence of the EZLP System ............. 6
1.6 Availability of the EZLP System .................... 6
1.7 Continuing Effort ................................... 7

Chapter 2: EZLP: A Narrative Description

2.1 Introduction ........................................... 8
2.2 A General Outline .................................... 8
2.3 The Driver ............................................ 8
2.4 The Editor ............................................ 10
2.5 Syntactical Analysis .................................. 11
2.6 Optimization and Output ............................. 12
2.7 Segmentation of EZLP ................................ 14
2.8 Machine Dependence of EZLP ....................... 15

Chapter 3: User Instructions

3.1 Introduction ........................................... 18
3.2 Starting EZLP ......................................... 18
3.3 Entering the Model .................................... 18
3.4 Editing the Model ..................................... 24
3.5 Solving the Model .................................... 27
3.6 Sensitivity Analysis of the Results ................. 30
3.7 Restarting and Terminating EZLP ..................... 31
3.8 The HELP Command ................................... 31
3.9 External File-Handling Capabilities ................. 33

Chapter 4: Terminal Sessions

4.1 Introduction ........................................... 34
4.2 Session 1: A Simple Example of EZLP ............... 35
4.3 Session 2: An Advanced Example of EZLP ............ 36
4.4 Session 3: A Transportation Example ................. 38
4.5 Session 4: An Integer Programming Example .......... 40
4.6 Session 5: A Sensitivity Analysis Example .......... 41

Appendix A: A Simplified User's Manual for EZLP ........... 46
Chapter 1: INTRODUCTION

1.1 Background

Linear programming is a mathematical optimization technique utilized intensively in the fields of mathematics, management and engineering—especially within the areas of operations research and management science. The simplex method—the standard technique for solving linear programs—is an iterative method, and, as such, is dependent on the computer to carry out the rote calculations. Computerized simplex codes are available in small problem-solving versions and in large production versions in most university, private and governmental computing centers. Students find that interaction with these computer codes is often very difficult. The result is that small scale usage in the classroom or laboratory or for homework is discouraged. This bad experience carries over and tends to discourage the student's use of linear programming on the job.

In most universities today there are a number of undergraduate and graduate courses utilizing linear programming as a problem-solving tool. Georgia Tech is typical with as many as 6-10 undergraduate courses and 15-20 graduate courses utilizing linear programming. In these courses the student is expected to develop modelling skills with linear programming, appreciate the simplex method (or its variants) as a solution tool and to interpret the output results of the simplex method in the particular problem situation. In addition to course work there are usually a number of students performing research on new techniques which require a check by solving small linear programming problems. Hand calculations are good up to about four constraints and ten variables. Beyond this number the student must go to the computer in order to have the calculations performed, otherwise, negative feedback results. The student
remembers the difficulty with solving the model and not the power of the modelling technique. When the student gets to the computer he usually finds a whole new set of difficulties with regard to the fact that in order to provide the computer with his model he must learn a computer-oriented (rather than student-oriented) "input language".

A number of institutions including Northwestern University, Stanford University, the University of Texas, and Georgia Tech have developed linear programming computer programs which are more directly student oriented in their design.

In the School of Industrial and Systems Engineering at Georgia Tech an experimental project has been underway since 1971 to develop a student-oriented conversational linear programming code (EZLP) for use on small problems (up to 50 constraints and 100 variables). The innovation is in the special way that the student interacts with the computer. The student inputs his linear programming model to the computer in exactly the same manner that he would write it down on paper. Evolution of this code has been through class projects. Despite limitations in its initial design, EZLP was nonetheless widely accepted by the students.

1.2 Initial Project Effort

The National Science Foundation agreed to support a concerted effort, during the 1975-1976 academic year, to redesign and develop a conversational computer code based on the concepts developed at Georgia Tech.

To accomplish the given task, a Georgia Tech team was organized. The Georgia Tech team consisted of:
Dr. John J. Jarvis, Project Director  
Frank H. Cullen, Research Assistant  
Chris Papaconstadopoulos, Research Assistant  

This team had the overall responsibility of designing, developing, coding and documenting the computer system.

An Advisory Committee of eminent researchers and practitioners in the field of linear programming was also organized to provide guidance to the Georgia Tech team. The Advisory Committee consisted of the following individuals:

1. Dr. Claude Cohen  
Northwestern University

2. Dr. Harvey J. Greenberg  
Federal Energy Administration

3. Dr. Marvin A. Griffin  
University of Alabama

4. Dr. Michael E. Thomas  
University of Florida

This committee had significant impact on the design of the final computer system.

1.3 Initial Project Plan

During the early stages of the project the Georgia Tech team devoted a great deal of time and effort to the development and documentation of a preliminary design of a computer system for EZLP. This documentation was submitted to the Advisory Committee for their review.

A meeting of the Advisory Committee was conducted on the Georgia Tech campus to discuss the preliminary design. This discussion lead to a much improved design. On advice of the committee, the fundamental concept of
complete machine independence was replaced by a moderate level of machine
dependence to permit substantial reduction in program size and complexity.
Also, a number of general features were eliminated from the initial design
while many others were added. The Georgia Tech team was very pleased with
the structure that emerged from that meeting.

The Advisory Committee also suggested, at its meeting, that the Georgia
Tech team retain a certain degree of flexibility in its development of EZLP
so that the Tech team might be able to react in an expeditious manner to
potential difficult situations which could arise during coding.

The remainder of the initial grant period, following the Advisory
Committee meeting, was spent developing and testing the EZLP code. With
a few changes, the initial EZLP system developed was the one which emerged
from the Advisory Committee meeting.

1.4 Follow-on Project Effort

The initial project phase concluded with the development of the basic
EZLP system. The project then proceeded into a second phase consisting of
dissemination, formative evaluation, and modification/improvement.

During the second phase, the dissemination task consisted of mailing
literature on the basic EZLP system to approximately one hundred and fifty (150)
departments of Industrial (and Systems) Engineering, Operations Research,
Management Science, Computer Science and Mathematics throughout the United
States. The literature described the EZLP system, its functions and operation.
Copies of the computer code were offered at nominal or no charge to interested
institutions. The objectives of this offering were to (1) achieve wide dissemina-
tion of the project results, and (2) generate a subset of implementing institu-
tions from which a formative evaluation of the EZLP system could be conducted.
Some two dozen institutions requested the EZLP code for implementation. It was anticipated that a group of five institutions, providing a cross section of EZLP users, would be identified for participation in a seminar on EZLP to be held at Georgia Tech. However, it soon became evident that the critical link would not be understanding the EZLP system, but, instead, would be getting EZLP operational on the myriad of different computers (UNIVAC, Burroughs, Prime, IBM, etc.) involved.

After discussions with NSF, it was decided to carry the EZLP seminar to the users, instead of having the users come to Georgia Tech. A qualified graduate research assistant was sent to several targeted institutions to (1) solve EZLP implementation problems, (2) give a seminar/demonstration of the EZLP system and its capabilities, and (3) establish the necessary mechanism for receiving comments and evaluations of the EZLP system.

Several institutions were targeted on the basis of their interest and enthusiasm and the type of computer system involved. It was hoped that the formative evaluation could include results of experience on as many different computer systems as possible.

The implementation, testing and evaluation process proved invaluable to the continued improvement of the EZLP system. EZLP users evaluations included (1) modifications of the code to provide easier installation of differing computer systems, (2) changes in the materials, handouts, etc. describing EZLP and its uses, (3) expansion and improvement of the EZLP code to include additional features requested by the users, (4) modularization of the EZLP system to overcome the problems associated with core memory requirements for operation of the system, (5) changes in EZLP command structures and output formats to provide more intuitive understanding of the system operation and results, and (6) the alleviation of minor bugs and difficulties associated with the EZLP system.
The EZLP system which resulted from this testing and evaluation process is greatly improved. EZLP is currently operational on approximately two dozen computer installations throughout the United States. This list includes academic institutions, governmental agencies and private firms. EZLP users report great popularity and success of EZLP. The Georgia Tech team is pleased with the resulting system and its acceptance by others. EZLP is becoming known worldwide, and already copies of the code have been sent to several European countries.

The remaining chapters discuss the specific structure of EZLP and its use. Appendix A contains a simplified user's manual which could be passed out to students learning EZLP for the first time.

1.5 Machine Dependence of the EZLP System

The EZLP computer system has been intentionally designed to minimize the level of machine dependence within the constraint of reasonable program size and complexity. EZLP has been coded in FORTRAN and special forms which may be available only on the CDC Cyber 74 (the machine used) are avoided. Those places in the code where machine dependence was unavoidable are few in number and clearly identified.

Section 2.8 of Chapter 2 discusses the required changes to convert the machine dependent portions of EZLP to other computer systems.

1.6 Availability of the EZLP System

Development of the EZLP computer system was sponsored by a grant from the National Science Foundation. As such, the EZLP computer system is freely available to anyone desiring a copy. A computer listing on paper is available at no charge. There is a nominal charge, to cover expenses,
for a listing on cards or tape. EZLP may not be resold. For further infor-

write

Dr. John J. Jarvis, EZLP Project Director
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

1.7 Continuing Effort

EZLP is an evolving computer system. Effort continues at Georgia Tech to maintain and improve the EZLP code. We invite comments and suggestions on improving EZLP.
Chapter 2: EZLP – A NARRATIVE DESCRIPTION

2.1 Introduction

This chapter presents the workings of EZLP in a narrative form. Emphasis is given to the identification of the components of EZLP, their attributes, and a discussion of their mutual interaction. Coded in standard FORTRAN, EZLP nonetheless has certain functions which cannot be written in a machine-independent code. These are discussed in the last part of this chapter.

2.2 A General Outline

In overview, EZLP consists of three main subprograms tied together by a controlling main program, or "driver." The three main subprograms are:

a. Editor - the ability to create and maintain (update) a current model kept physically in an exterior mass storage device
b. Syntactical Analysis - inputting the model statements from the external file created by the EZLP Editor, performing parsing and syntax-checking operations, and generating a tableau for input to the optimization process
c. Optimization - a solution and output of a model in tableau form by simplex procedures

Figure 2.1 presents a general diagram of the program flow of EZLP which also indicates the interaction of the three main subprograms.

2.3 The Driver

The main program, or driver program, has several functions. Among these are:

a. distribute and pass control among the three main subprograms
b. perform ancillary operations not handled by the subprograms
Figure 2.1: EZLP Program Logic Flow
Upon program execution, the driver program assumes control and examines the first input statement. If the keyword (the first word) of the input statement is an editor associated keyword (e.g., AND, MIN, CHANGE), the driver continues to parse the input statement to determine the row name if it is there and to construct the default value if it is not; control is then passed to the editor subprogram. If the keyword of the input statement is associated with an ancillary operation (e.g., RUN, SAVE, or PRINT), the driver continues to parse the input statement to determine what internal program switches are to be set, and in the case of the SAVE statement, perform the requested file operations. If the keyword is USE, to trigger a solution attempt, the driver parses the remainder of the input statement to determine the simplex method to be used, passes control first to the syntactical analysis subprogram, and then to a tableau-building subroutine and the optimization subprogram in succession, if there are no syntactical errors in the model.

In Figure 2.1 the driver program may be considered to be everything not within a subprogram boundary.

2.4 The Editor

The editor subprogram is primarily responsible for model file maintenance and performs four principal functions:

a. Adds a statement to the model file
b. Deletes a statement in the model file
c. Changes a statement in the model file
d. Lists either a single row or the entire contents of the model file

Model File Organization

The model file itself is a simple linked-list structure with chaining used for continuation lines. Two vectors in the COMMON block EDIT, CNTNME
and LINKS, store the alpha name and the relative location of the first record for the particular row of the model on mass storage. This record has a forward link to a continuation record if used. This forward link is zero (0) if there are no further continuations after the current record.

The program logic and passed parameters to the mass storage read and write subroutine are such that first-available and other random access techniques can be used on computers of various manufacture.

Subprogram Operation

Upon passage of control and the appropriate parameters, the Editor accesses the model file to perform the desired function, and returns control to the driver which then reads in the next input statement.

Continuations of rows in the model are handled directly by the Editor, which reads in successive input statements and sets up the chaining relationship until the continuation situation no longer exists.

2.5 Syntactical Analysis

The syntactical analysis subprogram is the process wherein the model file contents are read in, parsed, checked for errors in syntax, and translated into a coded form which is subsequently transformed into a tableau structure.

In overview, the syntactical analysis subprogram has two main entities:

a. the model statement parser, which takes lines from the model file and generates a uniform symbol table in which the components of the model line (e.g. coefficient, name or identifier, delimiter) are described and;

b. the analysis routine which uses as input the uniform symbol table, checks for syntax errors, and outputs a quasi-diagonal tableau.
which later serves as input to a tableau-building subroutine which adds slack and artificial variables and generates a rectangular tableau structure.

Other output from the syntactical analysis subprogram includes:

a. a vector containing row names as they appear in the tableau
b. a vector containing variable names as they were first mentioned in the model file
c. vectors containing lower and upper bounds for each variable in the same order as (b)
d. A designator of which rows of the tableau are associated with objective functions
e. an indication of which objective function row is to be optimized during phase two of the optimization process

In the instance that there are syntax errors present in the model file, descriptive error messages and pointers are printed in an attempt to pinpoint the exact location of the trouble.

The memory locations used by the syntactical analysis are focused in the COMMON area OPT and WORK.

2.6 Optimization and Output

The optimization and output subprogram is the process by which the simplex operations are performed on the rectangular tableau created by the tableau-building subroutine which, in turn, receives its input from the syntactical analysis subprogram. Depending upon the value of the parametric switch METHOD set by the driver program during the parsing of the USE statement, the optimization process selects from four basic algorithms: primal or dual simplex with or without rational arithmetic. The lower-upper bounded simplex approach is used throughout the optimization routines. The different
results obtained by the specification of the UPPER parameter in the USE statement owe to the generation of explicit bound constraints by the syntactical analysis subprogram when UPPER is not specified.

The primal algorithms (real and rational) employ a two-phase method for which the Phase 1 objective function and the starting basis are determined by the tableau-building subroutine. At the successful completion of Phase 1, the Phase 2 objective function (which has been stored as a null constraint in the tableau) is written over the Phase 1 objective function and the process resumes.

The dual algorithms begin with the creation of the Phase 2 objective function from the appropriate null constraint (objective function) in the tableau.

The rational algorithms require the maintenance of both a numerator and a denominator for each tableau entry (and each element of the right-hand side). For example, the numerator for an element in an array may be stored in position I, while the corresponding denominator will be stored in position J+I, where the Jth position of the array held the value of the numerator of highest subscript index.

At the end of each simplex iteration, or both before and after the optimization process has been completed, there might be output specified by the user via a PRINT statement. In this instance, control within the optimization process is passed to a point which prints out the values of the nonzero variables, the current basis and basis activity, or the current tableau and updated objective function. Control is then returned to the appropriate place in the optimization process.

Upon reaching an optimal solution, control within the optimization process passes to a point at which the default output (the optimal primal and dual solutions) is printed out. Even if no other output is specified or requested, the default output is printed.
When the printing default output is completed, control returns to the
driver program which then reads in and parses the next input statement.

The memory locations used by the optimization and output subprogram
are focused in the COMMON area OPT and WORK.

2.7 Segmentation of EZLP

In dealing with computers with smaller capacity than large-scale university
computing systems, memory economy via code segmentation or overlay is often
of considerable interest.

The modular subprogram concept allows some or all of the subprogram code
to be overlayable, and hence less wasteful.

Specifically, the three subprograms are mutually overlayable as well
as those functions which serve only one of the subprograms. Graphically,
we can represent the code components of EZLP as a tree, the root of which
is non-overlayable, and the separate branches of which form the overlays.
In Figure 2.2, this tree structure is represented in terms of the actual
FORTRAN subroutines and function names.

![Segmentation Tree-Structure of EZLP](image)
2.8 Machine Dependence of EZLP

There are two aspects of the code of EZLP which are likely to cause difficulties when an attempt is made to convert the code to a machine other than the one for which EZLP was written (the CDC Cyber 74). These are

a. Alpha variables - packing and unpacking, and manipulation
b. File manipulations - particularly with respect to error and end-of-file conditions

Alpha Variables

The storing of alphanumeric information in variables has always been a weakness in the design of FORTRAN. Consequently, the use of either integer or real variables to store alphanumeric information is bound to be non-standard and change from machine to machine. For the purpose of program design and implementation, EZLP uniformly uses real variables. For those compilers and computers which require that some other type be used for alpha variables, explicit type statements must be added to those portions of the code affected for the alpha variables.

A basic assumption of the EZLP code is that the alpha variables can hold at least eight (8) characters and are appropriately filled so that a comparison with a Hollerith constant is meaningful. Hexadecimal machines (or character machines) normally have type statements which allow the length of the variable in bytes (characters) to be specified. Where applicable, these constructs should be used.

An often-used subroutine in EZLP for composing a single eight-character alpha variable from eight single-character alpha variables is called PACK.
This subroutine is totally machine-dependent and must be changed for different computers.

**File Manipulation**

EZLP employs a number of files during execution. These files are summarized in Table 2.1. These files, with the possible exception of file 8, are strictly sequential files and require no special programmatic considerations. File 8, the model file, because of the random way in which editing is applied to it, can be effectively used as a random access file. Since random access is normally a machine-dependent function, EZLP separates out the random access read and write functions into separate subroutines MSREAD and MSWRIT. These two subroutines must be re-written for different computers.

<table>
<thead>
<tr>
<th>File Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Input file (from the terminal)</td>
</tr>
<tr>
<td>6</td>
<td>Output file (to the terminal)</td>
</tr>
<tr>
<td>7</td>
<td>Message file (for syntax error messages)</td>
</tr>
<tr>
<td>8</td>
<td>Model file (linked and chained mass storage)</td>
</tr>
<tr>
<td>9</td>
<td>Run file (alternate input file)</td>
</tr>
<tr>
<td>10</td>
<td>Save file (model output file)</td>
</tr>
<tr>
<td>11</td>
<td>Help stored-text file (instructions)</td>
</tr>
</tbody>
</table>

End-of-file checking is also achieved differently with different FORTRAN compilers. On the CYBER 74, the test is the implicit function EOF(u). Occurrences of this function should be replaced by the appropriate READ statement construct for the local FORTRAN compiler. For example, the sequence

```fortran
READ (INFILE, 102) (BUFFER(I), I=1, 80)
IF (EOF(INFILE) .NE.0) GO TO 320
```

16
could be replaced by something resembling

```
READ (INFILE, 102, END=320) (BUFFER(I), I=1, 80)
```

if such was the appropriate end-of-file construct.
Chapter 3: USER INSTRUCTIONS

3.1 Introduction and Notation

This chapter is designed to describe the operation of EZLP from the user's point-of-view. The material covered here is somewhat more comprehensive than that which would be required for the user with simple problems. Appendix A contains a simplified user's manual.

For clarity of presentation, this chapter uses BNF notation in describing syntactical rules. Briefly, capitalized words indicate keywords, [] indicates an optional clause, {} indicates a choice of two or more alternatives, and <> enclose a descriptive term for a syntactical entry.

3.2 Starting EZLP

The exact syntax of executing EZLP will vary from computer to computer and from installation to installation, and so cannot be explicitly stated here. For the sake of clarity, let it be assumed that EZLP has been executed and is awaiting input from the user.

At this point, the user has the option of proceeding to enter the model (3.3) or getting a brief (about two pages) description of how EZLP is operated. This description, in the absence of this chapter or other documentation should supply a minimum of information to the user so that he can use the program. To obtain this brief description, the user must enter the following command:

HELP BRIEF

This command is discussed further in section 3.7.

3.3 Entering the Model

The first thing the user must do is enter the model. The entry of the model follows certain rules - there are rules concerning the order in which inputted statements must be organized, and there are rules which govern the
way in which different types of statements are made up. Before these rules are discussed, it would be best if some basic ideas were explained: A statement is a single logical input. A constraint is a statement. The objective function is a statement. A delimiter is a single character which serves to terminate one entry and perhaps begin the next. For the entry of the model, the space character ( ) is the usual delimiter. A name provides the convenience of letting the user assign unique identifiers to variables, constraints, and objective functions. The names themselves are composed of from one to eight characters. The first character must be alphabetic (A-Z), while the remaining characters must be alphabetic, numeric (0-9), or numerics separated by commas. X2, PROFIT, ITEM3, and Y2,3 are examples of acceptable names, while ITEM-3 and $VAR are not. A space character cannot be part of a name. EZLP reserves certain names for its internal use. Row names of the form ROW#n are reserved for each objective function or constraint statement which was unnamed by the user. This generated name can be used for editing purposes. In addition, EZLP uses the names SLK#n and ART#n for the slack and artificial variables that it generates. Since the user may not input any name containing the # character there can never be any confusion between user-generated names and EZLP-generated names. An input-line corresponds to a line of input on the terminal. In batch processing, this is equivalent to a card image.

If desired the user may indicate that heading information is to be printed at the top of the optimal solution output. The user must type in

```
TITLE <heading information>
```

where <heading information> appears as a heading at the top of the output.
The actual entry of the model is divided into two basic parts: the objective function and the constraints.

**Entering the Objective Function**

The syntactical rule for the entry of the objective function is:

\[
\begin{align*}
\{\text{MIN}\} & \text{ or } \{\text{MINIMIZE}\} \quad [\text{objective-function-name}]: \quad \text{<arithmetic expression>} \\
\{\text{MAX}\} & \text{ or } \{\text{MAXIMIZE}\} \quad &
\end{align*}
\]

where \(<\text{objective-function-name}>\) is an optional user-assigned name for the objective function and \(<\text{arithmetic expression}>\) is a linear combination of variable-names.

Note that the colon (:) is required at all times.

If the objective function statement is too large to be entered on one input line, continuation of the statement is accomplished by placing an ampersand (&), after the last character of the current input line and continuing the statement on the next input line. There are no limits on the number of continuations allowed for a single statement.

Examples of possible objective function statements:

1. \text{MIN}: \quad 2 + 3X2 - 2.3X4
2. \text{MAX PROFIT}: \quad 3\text{INCOME} - 4\text{EXPENSE} - .673\text{OVERHD}
3. \text{MIN COST}: \quad 2.4\text{VAR1} - 3.53\text{VAR2} + .004\text{VAR3} \& -\text{VAR4}

**Alternate Objective Functions**

In some applications of linear programming, it is of interest to consider a number of possible objective functions subject to the same set of constraints. EZLP allows for the entry and subsequent optimization of the entered model in coordination with alternate objective functions.
The syntactical rule for the entry of an alternate objective function is:

\[
\text{ALSO [<objective-function-name>] : <arithmetic expression>}
\]

Notice that the above differs from the ordinary entry of the objective function in that the keyword "ALSO" is required and replaces the keywords "MIN" and "MAX".

Each model must contain exactly one primary objective function (using the keywords "MAX" or "MIN"). However, a model may contain any number of alternate objective functions. EZLP operates with only one objective function at a time.

Specifying an alternate objective function for the purpose of optimization is accomplished just prior to specifying the method of solution. This is discussed in Section 3.5 ii.

Entering the Constraints

The constraints associated with normal linear programming model can be broken down into three general classes:

1. Simple arithmetic constraints - those constraints which involve more than one variable and only one relational operator; and
2. Range constraints - those constraints which involve more than one variable and two identical operators; and
3. List constraints - those constraints which involve only one variable. Constraints of this type usually specify upper or lower bounds on a particular variable or a list of variables.
Simple Arithmetic Constraints

The syntactical rule for the entry of a simple arithmetic constraint is:

\[
\{ST\} [\text{constraint-name}]: \text{arithmetic expression} > \text{arithmetic expression} \\
\text{relational operator} \text{arithmetic expression} \\
\]

where \text{constraint-name} is an optional user-assigned name for the particular constraint. The \text{arithmetic expression} is a linear combination of variable names as described in the objective function statement above; and \text{relational operator} takes one of the following forms:

1. \{=> or >=\} (greater than or equal to)
2. = (equal to)
3. \{=< or <=\} (less than or equal to).

Note that the colon is required at all times.

If any particular constraint statement is too large to be entered on one output line continuation of the statement is accomplished by placing an ampersand (&) after the last character of the current input line and continuing the statement on the next input line. There are no limits on the number of continuations allowed for a single constraint statement.

The first constraint statement should begin with the keyword "ST", which stands for "subject to". Subsequent constraint statements must begin with the keyword "AND".

Constraint statements which include the optional \text{constraint-name} are easily referenced for editing by this user-assigned name. Unnamed constraints are assigned a name by EZLP so that the user can also reference and edit these constraints. This is further discussed in section 3.4.

Examples of possible simple arithmetic constraints:
Range Constraints

EZLP is designed to accommodate range constraints. The syntactical rule for the entry of range constraints is:

\[
\begin{align*}
\text{ST} \quad & \quad \{ \text{AND} \} \quad [\text{constraint-name}] : \quad \text{constant-1}<\text{inequality-operator-1}> \quad \text{AND} \quad \text{arithmetic expression}<\text{inequality-operator-2}>\text{constant-2} \\
\end{align*}
\]

where the following rules apply:

1. \text{<inequality-operator-1>} and \text{<inequality-operator-2>} must be exactly alike and must be either \text{>=} or \text{<=}.
2. If the relational operators are \text{>=}, then \text{<constant-1>} must be greater than or equal to \text{<constant-2>}
3. And, if the relational operators are \text{<=}, then \text{<constant-1>} must be less than or equal to \text{<constant-2>}

Examples of possible range constraints are:

1. AND: \quad 4 <= 3x1 + 4.2x3 - 4x4 <= 6.2
2. AND WORKERS: \quad 500 >= 6.3FORCE1 - 5.3FORCE2 >= 243
3. AND CONSTRNT: \quad 3.4 <= 7 + VAR3 =< 8.46

List Constraints

The syntactical rule for the entry of a list constraint is:

\[
\begin{align*}
\text{AND} \quad [\text{constraint-name}] : \quad \{ \text{ALL} [\text{OTHER}] \text{ VARS} \} \{ \text{variable-list} \} \{ \text{relational operator} \text{<constant>} \} \\
\end{align*}
\]
where <constraint-name> and <relational operator> are described above for simple arithmetic constraints; and <variable-list> is a list of one or more variable names, separated by commas (,).

The reserved phrase "ALL VARS" indicates that the bound specified applies to all variables in the model. The reserved phrase "ALL OTHER VARS" indicates that the bound specified applies to all variables which are not included in another list constraint for the same type bound (i.e., upper or lower).

The reserved word "URS" stands for "unrestricted in sign".

The default option for all variables if they do not appear in a list constraint is "URS". EZLP prints a notification of the unrestricted variables.

Examples of possible list constraints:

1. AND: X1, X2, X3 <= 0
2. AND: ALL VARS >= 0
3. AND: OIL <= 375.43
4. AND: ALL OTHER VARS <= 1
5. AND: PROFIT URS
6. AND MYCNSTR: WIDTH, LENGTH >= 0

An example of an entry of a complete model:

MIN: 10.5WIDTH1 + 11.8WIDTH2 - 30LENGTH
ST CNSTR1: 15LENGTH - 2WIDTH1 <= 10
AND CNSTR2: 20WIDTH2 + LENGTH <= 15
AND: ALL VARS >= 0

3.4 Editing the Model

After completion of the model entry, the user has the ability to perform certain editing functions. These functions are:
1. Adding a constraint or objective function
2. Deleting a constraint or objective function
3. Changing a constraint or objective function
4. Listing the model in whole or in part

i) Adding a Constraint or Objective Function

At all times the user has direct access to his model. Thus, addition of a new constraint or objective function to the end of the current model may be accomplished by simply typing the appropriate constraint or objective function statements. EZLP will automatically append the new statement to the previous model.

For example, suppose that the user has attempted to solve his model and this attempt resulted in an unboundedness indication. Having determined that he failed to require nonnegativity he may do so by simply typing

\[
\text{AND:}\quad \text{ALL VARS } \geq 0
\]

If one wishes to insert a constraint into the middle of the current model, this may be accomplished by typing

\[
\text{INSERT}\quad \{\text{AFTER}\quad \langle\text{row-name}\rangle\} \{\text{model entry statement}\}
\]

where \(\langle\text{model entry statement}\rangle\) is either a constraint or objective function statement. Examples of the INSERT command are:

1. \(\text{INSERT AFTER ROW#2 AND: } X_1 + 2X_2 \leq 7\)
2. \(\text{INSERT BEFORE CONST6 AND CONST5: } 7 \leq X_1 \leq 9\)
3. \(\text{INSERT AFTER ROW#3 ALSO: } 3X_1 + 2\text{POWER}\)
ii) Deleting a Constraint or Objective Function

The syntactical rule for the deletion of a constraint is:

```
DELETE <row-name>
```

where `<row-name>` is either the user assigned name, or in the absence of
this name, the row name assigned by EZLP. The assigned name is simply
ROW#n, where n is the number of the model entry as it was entered. Ex-
amples of assigned row-names are ROW#4, ROW#13, and ROW#123.

1. DELETE ROW#13
2. DELETE SURPLS

iii) Changing a Constraint or Objective Function

The syntactical rule for the changing of a constraint or the objective
function is:

```
CHANGE <row-name> "<string1>"<string2>"
```

where `<row-name>` must be identical to some existing row name.

The `CHANGE` command will replace `<string1>` by `<string2>`. The construct
"<string1>"" will delete `<string1>`.

Examples of possible `CHANGE` statements are:

1. CHANGE CONSTR1 "ENG"G"
2. CHANGE SURPLS "3.2"

iv) Listing the Model

The syntactical rule for listing the model on the terminal is:

```
LIST [<row-name>]
```
where the optional <row-name> is included if only a single row is to be printed. If <row-name> is omitted the entire model will be printed. Examples of possible LIST commands are:

1. LIST
2. LIST ROW#5
3. LIST MYCSTR

3.5 Solving the Model

Once the model has been entered and edited to the user’s satisfaction, the user takes the following steps:

1. Specifying the Desired Output (this step is optional and, if omitted, results in only the optimal solution being printed).
2. Specifying the Objective Function to be Optimized (this step is optional and is only to be used when selecting an alternate objective function).
3. Specifying the Method of Solution.

i) Specifying the Desired Output

If the user wishes to obtain more output than the optimal solution he may use the PRINT command.

The syntactical rules for the PRINT command are:

```
PRINT [INITIAL] [FINAL]  

VARS  
BASIS  
TABLEAU  
ALL  
NONE

[,]<frequency count>
```

This statement concerns the output on the terminal of information directly related to the optimization process. The following descriptions apply:
VARS: prints the name and value of each nonzero primal variable;
and the name and value of each nonzero dual variable.

BASIS: prints the names of the variables in the basis and the objective
function value.

TABLEAU: prints the tableau.

ALL: prints all of the above.

NONE: prints only the optimal solution.

<frequency-count>--the PRINT statement is executed every <frequency-count>
iterations. If this optional clause is omitted, the PRINT
statement will be executed after every iteration unless the
keywords INITIAL and/or FINAL are used. When ALL or NONE
are present <frequency-count> is ignored.

INITIAL: prints the requested information only for the initial iteration.

FINAL: prints the requested information only for the final iteration.

(Note that INITIAL and FINAL may be used together.)

Examples of possible PRINT commands are:

1. PRINT VARS 5

2. PRINT ALL

3. PRINT BASIS, VARS,3

4. PRINT FINAL TABLEAU

5. PRINT INITIAL, FINAL BASIS

ii) Specifying the Objective Function to be Optimized

In the event that the user has entered alternate objective functions,
he may wish to specify the name of the objective function to be optimized.
If this is not done, EZLP defaults to either the original objective function
entered with the model or the last objective function name specified in a
prior ALTOBJ statement. The syntactical rule for the specification of the
objective function command is
ALTOBJ \{ MIN \} \{ MAX \} \{objective-function-name\}

If an ALTOBJ command has been previously given, then to determine the row name of the current objective function being used, the user should type

ALTOBJ STATUS

iii) Specifying the Method of Solution

The specification of the method of solution triggers an attempt by EZLP to solve the entered model and produce output as specified in 3.5.i.

The syntactical rule for specifying the method of solution is:

USE \{RATIONAL\} \{UPPER\} \{PRIMAL\} \{DUAL\}

Specifying RATIONAL keeps all data in rational form, i.e. "0.5" would become "1/2". Since the RATIONAL option requires storing both a numerator and a denominator matrix, the maximal allowable problem size is cut in half under this option. Further, all coefficients and constants must be entered as integers.

Specifying UPPER before PRIMAL or DUAL will cause the lower-upper bounded primal or dual algorithm to be used.

Examples of possible USE commands are:

1. USE PRIMAL
2. USE UPPER DUAL
3. USE RATIONAL PRIMAL
4. USE RATIONAL UPPER PRIMAL
3.6 Sensitivity Analysis of the Results

Once an EZLP model has been optimally solved by any of the USE commands, the user has the option of requesting a sensitivity analysis of the cost coefficients or the right-hand-side constants. The syntax of this command is:

\[
\text{RANGE} \begin{cases} \text{RHS} \\ \text{OBJ or OBJFCN} \\ \text{ALL} \end{cases} \begin{cases} \{ < \text{name} > \} \\ \text{ALL} \end{cases}
\]

The effect of the RANGE command is the determination of lower and upper limits on various model parameters which maintain the current (optimal) basis. Examples of possible RANGE commands and their effects are:

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. RANGE RHS ROW#1</td>
<td>Provides lower and upper sensitivity limits for the right-hand-side constant for the ROW#1 constraint.</td>
</tr>
<tr>
<td>2. RANGE OBJ X22</td>
<td>Provides sensitivity information for the cost coefficient for variable X22 in the objective.</td>
</tr>
<tr>
<td>3. RANGE OBJ, ALL</td>
<td>Provides sensitivity information for all objective function coefficients.</td>
</tr>
<tr>
<td>4. RANGE ALL</td>
<td>Provides sensitivity information for all objective function coefficients and all right-hand-side constants.</td>
</tr>
</tbody>
</table>

If a constraint specified in the RANGE command is, itself, a range constraint, then separate sensitivity information will be printed for the lower constant and the upper constant of the constraint.
3.7 Restarting and Terminating EZLP

At the completion of any model solution the user has three options:

1. Editing the current model and re-solving it,
2. Starting fresh with a new model, or
3. Terminating EZLP

The first of these options is accomplished by simply entering the appropriate edit statements (see 3.4) followed by the appropriate USE commands (see 3.5).

The second of these options is accomplished by the RESTART command. The syntax of this command is:

```
RESTART
```

The RESTART command re-initializes all areas. The effect of this command is to clear out the current model and print options.

Terminating EZLP is accomplished by the END command. The syntax of this command is:

```
END
```

3.8 The HELP Command

This command allows the user, after he has started the program, to obtain a short set of instructions on how to solve simple LP problems. The user, at any point in his run, may request additional help in the following manner:

```
HELP [<keyword>]
```
where the optional clause keyword refers to abbreviated instructions concerning a particular area of interest. A list of keywords is:

General Keywords:

- **BRIEF** (briefly describes the general features of EZLP)
- **EDIT** (describes edit commands)
- **EXAMPLES** (presents examples of EZLP models)
- **FILES** (discusses external file handling capabilities)
- **KEYWORDS** (gives the current list of keywords)
- **SOLVING** (indicates the methods and options for optimization and discusses output options)
- **MODEL** (discusses model entry syntax)

Specific Keywords:

- **ADD** (indicates how to add a constraint or objective)
- **ADVANCED** (presents an advanced example)
- **ALTOBJ** (indicates how to specify an alternate objective function for optimization)
- **CHANGE** (indicates how to change a constraint or objective)
- **CONT** (discusses the continuation of model statements)
- **CONS** (describes the options for inputting constraints)
- **DELETE** (indicates how to delete a constraint or objective)
- **LIST** (discusses the list options for model display)
- **NAMES** (provides a definition of acceptable variable names)
- **OBJ** (discusses objective functions)
- **PRINT** (discusses the output print options)
- **RUN** (describes procedure for inputting a model from an external file)
- **SAVE** (discusses procedure for saving a model)
- **TITLE** (discusses the options for method of optimization)
- **USE** (provides a definition of acceptable variable names)
If the keyword is omitted, EZLP will print a short description of how to obtain additional information on the operation of EZLP.

3.9 External File-Handling Capabilities

There are certain situations wherein the user wishes to eliminate input effort by storing all or part of his input on a mass storage device. EZLP provides for this capability with two commands: the RUN command and the SAVE command.

i) The RUN Command

At any point in the execution of EZLP, the user may elect to refer EZLP to a mass storage file for subsequent input. This input must be in the form of 80 character source records and must be accessible sequentially by EZLP. The syntax of the RUN command is

\[
\text{RUN <file-name>}
\]

If the RUN command is inputted, EZLP will read the model directly from the working file <file-name>, print a question mark and await the next command.

ii) The SAVE Command

Upon completion of model entry, the user may elect to save the input model in source form in a mass storage file. The syntax of the SAVE command is

\[
\text{SAVE <file-name>}
\]

If the SAVE command is inputted, EZLP will write the current model into the working file <file-name>, print a question mark and await the next command.
Chapter 4: TERMINAL SESSIONS

4.1 Introduction

This chapter contains several examples of EZLP used to solve normal linear programming problems. These examples include:

1. Simple example - getting on, solving a simple problem, and getting off.

2. Advanced example - naming rows, editing the model, requesting additional output, and responding to an error in syntax.

3. Transportation problem example - two sources and two sinks to illustrate multiply-indexed variable names and the treatment of primal redundancy.

4. Branch and Bound example - assignment of titles to optimal solution output in addition to using the editing features of EZLP to easily solve a small integer programming problem using the Branch and Bound method.

5. Sensitivity Analysis - ranging applied to all constraints and all variables of a simple example.
4.2 Session 1: A Simple Example of EZLP

EZLP - VERSION 9/17/76
TYPE HELP IF YOU HAVE QUESTIONS, OTHERWISE PROCEED
? MAX: 3X1+2X2
? ST: 5X1+3X2 <= 19
? AND: X1-X2 >= 3
? AND: ALL VARS >= 0
? USE RATIONAL PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

X1    X2

SOLUTION

OBJECT
Z
MAXIMIZE

ROW#1
Z
23/2

ITERATIONS
2

VARIABLE SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>7/2</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>X2</td>
<td>1/2</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

CONSTRAINT SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0</td>
<td>5/8</td>
<td>1/2</td>
</tr>
<tr>
<td>ROW#3</td>
<td>0</td>
<td>-1/8</td>
<td>7/2</td>
</tr>
</tbody>
</table>

? END

.223 CP SECONDS EXECUTION TIME
4.3 Session 2: An Advanced Example of EZLP

EZLP - VERSION 9/17/76
TYPE HELP IF YOU HAVE QUESTIONS, OTHERWISE PROCEED

? MIN: 4.3 A - 5.6 B ++ 7 C
? ST: A + C = B + 7
? AND BOUNDA: A <= C + 7
? CHANGE BOUNDA "<+"<="+
  AND BOUNDA: A <= C + 7
? AND BOUNDB: 1 <= B <= 6
? AND: A>= 0
? USE UPPER PRIMAL

MIN: 4.3 A - 5.6 B ++ 7 C

1
FATAL ERROR # 1: ARITHMETIC OPERATOR IN WRONG PLACE

** UNRESTRICTED VARIABLES **

C

FATAL ERRORS IN THE MODEL.
PLEASE EDIT THE MODEL
INPUT WILL BE RECOMPILED

? LIST
ROW#1
 MIN: 4.3 A - 5.6 B ++ 7 C
ROW#2
 ST: A + C = B + 7
BOUNDA
 AND BOUNDA: A <= C + 7
BOUNDB
 AND BOUNDB: 1 <= B <= 6
ROW#5
 AND: A>= 0
? CHANGE ROW#1 "++"+
 MIN: 4.3 A - 5.6 B + 7 C
? PRINT FINAL ALL
? USE UPPER PRIMAL

** UNRESTRICTED VARIABLES **

C

** VARIABLE LIST **

A  B  C
PHASE 2 ITERATION - 1 CURRENT OBJ VALUE = 3015E+02

PRIMAL NON-ZERO VARIABLES
A BASIC WITH VALUE 7500E+01
B NON-BASIC WITH VALUE 1000E+01
C BASIC WITH VALUE 5000E+00

DUAL NON-ZERO VARIABLES
ROW#2 DUAL = 565E+01
BOUND A DUAL = -135E+01

THE CURRENT BASIC VARIABLES ARE -
C A

CURRENT TABLEAU -

RHS A B C SLK#3
ROW#1 -.302E+02 0. .500E-01 0. .135E+01
C .500E+00 0. -.500E+00 .100E+01 -.500E+00
A .750E+01 .100E+01 -.500E+00 0. .500E+00

UPDATED OBJECTIVE FUNCTION ROW
0. -.500E-01 0. -.135E+01

SOLUTION

OBJECT MINIMIZE ROW#1
Z .3015E+02

ITERATIONS 1

VARIABLE SECTION

NAME ACTIVITY LEVEL LOWER BOUND UPPER BOUND REDUCED COST
A .7500E+01 0. .1000E+22 0.
B .1000E+01 .1000E+01 .6000E+01 .5000E-01
C .5000E+00 -.1000E+22 .1000E+22 0.

CONSTRAINT SECTION

NAME SLACK ACTIVITY DUAL PRICE RHS VALUE
ROW#2 0. .5650E+01 .5000E+00
BOUND A 0. -.1350E+01 .7500E+01

? END .400 CP SECONDS EXECUTION TIME
4.4 Session 3: A Transportation Problem Example

EZLP - VERSION 9/17/76
TYPE HELP IF YOU HAVE QUESTIONS, OTHERWISE PROCEED
? TITLE TRANSPORTATION PROBLEM - SOLUTION VIA EZLP.
? MAX TRANSOBJ: 2X1,1 + 3X1,2 + 4X2,1 + 2X2,2
? ST SUPPLY1: X1,1 + X1,2 = 3
? AND SUPPLY2: X2,1 + X2,2 = 4
? AND DEMAND1: X1,1 + X2,1 = 5
? AND DEMAND2: X1,2 + X2,2 = 2
? AND: ALL VARS >= 0
? PRINT BASIS
? USE RATIONAL PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

X1,1   X1,2   X2,1   X2,2

PHASE 1 ITERATION - 1 CURRENT OBJ VALUE = 14
THE CURRENT BASIC VARIABLES ARE -
   ART#2   ART#3   ART#4   ART#5

PHASE 1 ITERATION - 2 CURRENT OBJ VALUE = 8
AT THIS ITERATION, X1,1 ENTERED THE BASIS AND ART#2 LEFT
THE CURRENT BASIC VARIABLES ARE -
   X1,1   ART#3   ART#4   ART#5

PHASE 1 ITERATION - 3 CURRENT OBJ VALUE = 4
AT THIS ITERATION, X2,1 ENTERED THE BASIS AND ART#4 LEFT
THE CURRENT BASIC VARIABLES ARE -
   X1,1   ART#3   X2,1   ART#5

PHASE 1 ITERATION - 4 CURRENT OBJ VALUE = 0
AT THIS ITERATION, X1,2 ENTERED THE BASIS AND ART#3 LEFT
THE CURRENT BASIC VARIABLES ARE -
   X1,1   X1,2   X2,1   ART#5

PHASE 2 ITERATION - 1 CURRENT OBJ VALUE = 24
THE CURRENT BASIC VARIABLES ARE -
   X1,1   X1,2   X2,1   ART#5
TRANSPORTATION PROBLEM - SOLUTION VIA EZLP.

SOLUTION

OBJECT MAXIMIZE TRANSOBJ

Z

ITERATIONS 1

VARIABLE SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1,1</td>
<td>1</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>X1,2</td>
<td>2</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>X2,1</td>
<td>4</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

CONSTRAINT SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPPLY1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SUPPLY2</td>
<td>0</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>DEMAND1</td>
<td>0</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td>DEMAND2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

END

.413 CP SECONDS EXECUTION TIME
4.5 Session 4: A Branch and Bound Example

If we consider the integer program

\[
\begin{align*}
\text{Max} \quad & X_1 + X_2 \\
\text{ST} \quad & X_1 + 3X_2 \leq 9 \\
& 3X_1 + X_2 \leq 9 \\
& X_1, X_2 \geq 0 \text{ and integer}
\end{align*}
\]

described by the graph

Using EZLP, we can construct the following Branch and Bound tree:

- **Node 0**
  \[ Z_0^* = 9/2 \]
  \[ X_1^* = X_2^* = 9/4 \]
  \[ X_1 \leq 2 \]
  \[ X_1 \geq 3 \]

- **Node 1**
  \[ Z_1^* = 13/3 \]
  \[ X_1^* = 2 \]
  \[ X_2^* = 7/3 \]
  \[ X_2 \leq 2 \]
  \[ X_2 \geq 3 \]

- **Node 2**
  \[ Z_2^* = 4 \]
  \[ X_1^* = X_2^* = 2 \]
  (Incumbent Solution)

- **Node 3**
  \[ Z_3^* = 3 \]
  \[ X_1^* = 0 \]
  \[ X_2^* = 3 \]
  (fathomed by Node 2)

- **Node 4**
  \[ Z_4^* = 3 \]
  \[ X_1^* = 3 \]
  (fathomed by Node 2)
4.6 Session 5: A Sensitivity Analysis Example

EZLP - VERSION 9/17/76
TYPE HELP IF YOU HAVE QUESTIONS, OTHERWISE PROCEED
? TITLE NODE 0 - SOLUTION VIA EZLP
? MAX: X1+X2
? ST: X1+3X2 <= 9
? AND: 3X1+X2 <= 9
? AND: ALL VARS >= 0
? USE RATIONAL UPPER PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>9/4</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>X2</td>
<td>9/4</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

** CONSTRAINT SECTION **

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0</td>
<td>1/4</td>
<td>9/4</td>
</tr>
<tr>
<td>ROW#3</td>
<td>0</td>
<td>1/4</td>
<td>9/4</td>
</tr>
</tbody>
</table>

? AND: X1<=2
? TITLE NODE 1 - ADDITION OF ONE BOUND ON X1
? USE RATIONAL UPPER PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NODE 1 — ADDITION OF ONE BOUND ON X1

SOLUTION

OBJECT MAXIMIZE ROW#1
Z 13/3
ITERATIONS 3

VARIABLE SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2/3</td>
</tr>
<tr>
<td>X2</td>
<td>7/3</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>SLK#3</td>
<td>2/3</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

CONSTRAINT SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0</td>
<td>1/3</td>
<td>7/3</td>
</tr>
<tr>
<td>ROW#3</td>
<td>2/3</td>
<td>0</td>
<td>2/3</td>
</tr>
</tbody>
</table>

AND: X2 ≤ 2

TITLE NODE 2 — UPPER BOUNDS ON X1, X2
USE RATIONAL UPPER PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

X1 X2

NODE 2 — UPPER BOUNDS ON X1, X2

SOLUTION

OBJECT MAXIMIZE ROW#1
Z 4
ITERATIONS 3

VARIABLE SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>X2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SLK#2</td>
<td>1</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
<tr>
<td>SLK#3</td>
<td>1</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>
### Constraint Section

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ROW#3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

? LIST

ROW#1
- MAX: X1+X2

ROW#2
- ST: X1+3X2 <= 9

ROW#3
- AND: 3X1+X2 <= 9

ROW#4
- AND: ALL VARS >= 0

ROW#5
- AND: X1<=2

ROW#6
- AND: X2 <= 2

? CHANGE ROW#6 "<= 2" => 3
- AND: X2 >= 3

? TITLE NODE 3 - LOWER BOUND ON X2, UPPER BOUND ON X1

? USE RATIONAL UPPER PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

| X1    | X2    |

NODE 3 - LOWER BOUND ON X2, UPPER BOUND ON X1

### Solution

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>MAXIMIZE ROW#1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>3</td>
</tr>
<tr>
<td>ITERATIONS</td>
<td>2</td>
</tr>
</tbody>
</table>

### Variable Section

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>3</td>
<td>3 POS INF</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>SLK#3</td>
<td>6</td>
<td>0 POS INF</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

### Constraint Section

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ROW#3</td>
<td>6</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

43
DELETE ROW#6
LIST ROW#5
AND: \( X_1 \leq 2 \)
CHANGE ROW#5 "\( X_1 \geq 3 \)"
TITLE NODE 4 - LOWER BOUND ON \( X_1 \) (FINAL NODE)
USE RATIONAL UPPER PRIMAL

** NO UNRESTRICTED VARIABLES IN THE MODEL **

** VARIABLE LIST **

\[
\begin{array}{ll}
& X_1 & X_2 \\
\end{array}
\]

NODE 4 - LOWER BOUND ON \( X_1 \) (FINAL NODE)

SOLUTION

---

OBJECT MAXIMIZE ROW#1

\[
\begin{array}{ll}
Z & 3 \\
\end{array}
\]

ITERATIONS 3

VARIABLE SECTION

---

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>3</td>
<td>3</td>
<td>POS INF</td>
<td>2</td>
</tr>
<tr>
<td>SLK#2</td>
<td>6</td>
<td>0</td>
<td>POS INF</td>
<td>0</td>
</tr>
</tbody>
</table>

CONSTRAINT SECTION

---

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#2</td>
<td>6</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>ROW#3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

LIST

ROW#1
MAX: \( X_1 + X_2 \)
ROW#2
ST: \( X_1 + 3X_2 \leq 9 \)
ROW#3
AND: \( 3X_1 + X_2 \leq 9 \)
ROW#4
AND: ALL vars \( \geq 0 \)
ROW#5
AND: \( X_1 \geq 3 \)

END

1.152 CP SECONDS EXECUTION TIME
SOLUTION

OBJECT

MAXIMIZE ROW#1
Z .2600E+02

ITERATIONS 3

VARIABLE SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY LEVEL</th>
<th>LOWER BOUND</th>
<th>UPPER BOUND</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>.2000E+01</td>
<td>0.</td>
<td>.3000E+01</td>
<td>0.</td>
</tr>
<tr>
<td>X2</td>
<td>.5000E+01</td>
<td>0.</td>
<td>.5000E+01</td>
<td>.1000E+01</td>
</tr>
</tbody>
</table>

CONSTRAINT SECTION

<table>
<thead>
<tr>
<th>NAME</th>
<th>SLACK ACTIVITY</th>
<th>DUAL PRICE</th>
<th>RHS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#4</td>
<td>0.</td>
<td>.3000E+01</td>
<td>.2000E+01</td>
</tr>
</tbody>
</table>

? RANGE ALL

RANGE INFORMATION

<table>
<thead>
<tr>
<th>CONSTRAINT NAME</th>
<th>LOW</th>
<th>CURRENT</th>
<th>HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW#4</td>
<td>.50000E+01</td>
<td>.70000E+01</td>
<td>.80000E+01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VAR TYPE</th>
<th>NAME</th>
<th>LOW</th>
<th>CURRENT</th>
<th>HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC</td>
<td>ROW#4</td>
<td>X1</td>
<td>0.</td>
<td>.30000E+01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X2</td>
<td>.30000E+01</td>
<td>.40000E+01</td>
</tr>
<tr>
<td>NON-BASIC</td>
<td></td>
<td>SLNK#2</td>
<td>-.10000E+21</td>
<td>0.</td>
</tr>
</tbody>
</table>

? END

.320 CP SECONDS EXECUTION TIME
The first three pages of this appendix serve as a reasonable handout for most beginning users. The remaining nine pages provide more detailed discussion of the topics covered in the first three pages. By employing the "HELP" command the user can obtain any or all of this information during execution of EZLP.
EZLP: AN INTERACTIVE COMPUTER PROGRAM DESIGNED TO SOLVE
STUDENT-ORIENTED LINEAR PROGRAMMING PROBLEMS.†

1: General

The first word of each line is a keyword, and indicates the function of the input line. The keywords are "MIN" or "MAX" for the objective function, "ST" for the first constraint, followed by "AND" for each additional constraint. Each name (e.g. variable name) consists generally of a combination of up to 8 alphabetic and numeric characters with the first character being alphabetic.

Model Entry

Model entry statements are composed of a keyword, an optional name, a mandatory colon, and either an objective function or a constraint.

Example of a model entry:

MAX: 2PROD1 + 3PROD2 - 4 COST
ST: PROD1 + PROD2 <= 150
AND: COST >= 5
AND: ALL OTHER VARS >= 0

Editing

The input model can be edited by "DELETE" and "CHANGE" statements. Adding a constraint is accomplished by simply typing the constraint.

Examples:

AND NEWROW: 5COST-PROD1 >= 20.23
CHANGE NEWROW "PROD1" PROD2"
DELETE NEWROW

†EZLP was developed in the School of Industrial and Systems Engineering, Georgia Tech, Atlanta, Georgia 30332 under NSF grant #SED75-17476. John J. Jarvis, project director; Frank H. Cullen and Chris Papaconstadopoulos, research assistants.
EZLP internally numbers the input model lines and if the optional name before the colon is omitted during model entry, a name of the form "ROW#n" (e.g., ROW#6) is assigned as a default.

The model can be listed either in total or one line at a time as follows:

- LIST (THE ENTIRE MODEL)
- LIST CONSTR2 (LINE "CONSTR2" ONLY)

Continuation of the input line is achieved by placing an ampersand (&) after a complete name. Example:

AND: HEAT - 2.34WOOD & -6.45POWER>=16.4

Specifying Output

EZLP always outputs the final optimal solution. Other output can be requested after every iteration or after every n iteration. Examples:

- PRINT BASIS 5 (The names of the basic variables every fifth iteration)
- PRINT TABLEAU, 2 (The tableau at every other iteration)
- PRINT VARS (The value of primal and dual variables)
- PRINT NONE (Resets print specs to only final solution)
- PRINT INITIAL, FINAL TABLEAU
- PRINT ALL (Prints everything)

Specifying the Method of Solution

After model entry, editing, and output specification, the USE statement triggers the optimization. Examples:

- USE PRIMAL (Ordinary simplex method)
- USE RATIONAL PRIMAL (Ordinary simplex method with rational arithmetic)
- USE UPPER DUAL (Lower-upper bounded dual simplex method)
- USE RATIONAL UPPER PRIMAL (Lower-upper bounded simplex method with rational arithmetic)

Upon completion of the solution attempt, EZLP returns to the editing phase.
Sensitivity Analysis

After executing any of the USE commands to obtain an optimal solution, the user may obtain sensitivity information by use of the RANGE command. Application of the range command determines lower and upper limits on the objective coefficients and/or right-hand-side constants which keep the current optimal basis.

Examples:

```
RANGE  RHS  ROW#6
RANGE  OBJ  X54
RANGE  OBJ  ALL
RANGE  ALL
```

Restarting and Stopping EZLP

If another model is to be entered from scratch, type "RESTART", and EZLP will re-initialize the model file area. If EZLP is to be terminated, type "END".

File-Handling Capabilities

EZLP has the ability to use external mass storage files for both input and output. Information on these options can be obtained by consulting Section 5 "FILE-HANDLING".

Simple Example of a Complete EZLP Run

```
MAX:  2X1+3X2+4.5X3
ST:   X1+X2   <=75.3
AND:  X2 +X3   <=45
AND:  ALL VARS >=0
USE PRIMAL
END
```

The Help Command

During execution EZLP permits the user to obtain specific information concerning its use. The user may obtain a short introduction to EZLP and its use by typing

HELP
If the user desires more specific instructions, he may type "HELP<keyword>", (e.g., HELP MODEL) where the acceptable keywords are:

<table>
<thead>
<tr>
<th>General keywords</th>
<th>EXAMPLES</th>
<th>FILES</th>
<th>KEYWORDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDIT</td>
<td>EXAMPLES</td>
<td>FILES</td>
<td>KEYWORDS</td>
</tr>
<tr>
<td>MODEL</td>
<td>SOLVING</td>
<td>BRIEF</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specific keywords</th>
<th>ADVANCED</th>
<th>CHANGE</th>
<th>CONT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>ADVANCED</td>
<td>CHANGE</td>
<td>CONT</td>
</tr>
<tr>
<td>CONS</td>
<td>DELETE</td>
<td>OBJ</td>
<td>PRINT</td>
</tr>
<tr>
<td>RUN</td>
<td>SAVE</td>
<td>TITLE</td>
<td>USE</td>
</tr>
<tr>
<td>ALTOBJ</td>
<td>NAMES</td>
<td>LIST</td>
<td></td>
</tr>
</tbody>
</table>
The Model

Providing a Title for the Model

EZLP permits the input of a title for the model. This title will appear just before the optimal solution is printed. The title command consists of:

1. The keyword "TITLE"
2. The title text

Example:

```
TITLE THIS IS MODEL 4
```

Variable and Constraint Names

In entering the model, the user has the ability of assigning his own names to the variables and the constraints. These names must be from one to eight characters long, begin with an alpha character, and contain no special characters. A variable name may contain a comma provided that the comma separates two numerics. Spaces are not allowed within names. EZLP generates constraint names for unnamed rows and variable names for slack and artificial variables. These take the form:

- `ROW#n` - for generated constraint names
- `SLK#n` - for generated slack variable names
- `ART#n` - for generated artificial variable names

Continuation of Lines

In the event that an input line exceeds 80 characters, continuation is accomplished by placing an ampersand (&) after a complete name and resuming input on the following line.

Objective Functions

EZLP allows for the specification of two kinds of objective functions: primary and alternate. For the simple model there usually will not be an alternate objective function. In essence, this concept allows the user to enter more than one objective function and optimize the various objective functions subject to the same constraints during the
course of the interactive session.

Primary objective function - The entry of the objective function consists of:

1. The keywords "MAX", "MIN"
2. An optional objective function name
3. A mandatory colon
4. And a linear combination of user-defined variable names.

If the optional objective function is omitted, EZLP will generate a name of the form ROW#n for use in editing and output. Examples:

\[
\begin{align*}
\text{MIN:} & \ 2X1 + 2X2 + 3X3 \\
\text{MAX MYOBJ:} & \ 3.54 \ \text{COLUMN1} - 4.543 \ \text{COLUMN2} + 6.25 \\
\text{MINIMIZE:} & \ 5.76 \ X1,2 - 4.33 \ X1,3 + 5.45 \ X2,1 & - 6.43 \ X2,3
\end{align*}
\]

Alternate Objective Functions - Alternate objective functions are entered in the same way as the primary objective function, except that the keyword "MIN" or "MAX" is replaced by the keyword "ALSO". The use of an alternate objective function is discussed in Section 4 "Solving the Model". Examples:

\[
\begin{align*}
\text{ALSO:} & \ 2X1 - 3X2 - 4X3 \\
\text{ALSO OBJ2:} & \ 3.45 \ \text{COLUMN1} - 4.511 \ \text{COLUMN2}
\end{align*}
\]

Constraints

Constraints are divided into three generic types.

1. Simple arithmetic constraints
2. Range constraints
3. List constraints

Each is discussed below.

Simple Arithmetic Constraints

A simple arithmetic constraint consists of

1. The keyword "AND" or "ST"
2. An optional constraint name
3. A mandatory colon
4. A linear combination of user-defined variable names
5. A relational operator (\(=,\leq,\geq\))
6. A second linear combination of user-defined variable names.
It is not allowable to have the same variable name in both 4 and 6 above.

Examples:

AND: \( X_1 + X_2 = 7 \)
AND MYCSTR: \( 5X_1 + 8.3X_2 \geq X_3 + 4 \)

Range Constraints

EZLP allows the entry of bounded arithmetic expressions (range constraints). Range constraints consist of the following:

1. The keyword "AND" or "ST"
2. An optional user-defined constraint name
3. A mandatory colon
4. A constant (constant-1)
5. An inequality relational operator (\( \geq \) or \( \leq \))
6. A linear combination of the user-assigned
7. An inequality operator (\( \geq \) or \( \leq \))
8. A constant (constant-2)

The restriction on range constraints are that the two inequality operators must be identical and that the two constants must be consistent. (i.e. if the inequality operators are \( \leq \), constant-1 must be \( \leq \) constant-2).

AND: \( 4 \leq 3X_1 + 4.3X_2 \leq 6.2 \)
AND BOUND1: \( 7.2 \geq 3X_2 + \text{HEAT} - 5\text{POWER} \geq 5 \)

List Constraints

EZLP allows bounds for collective groups (or lists) of user-defined variable names to be specified in list constraints. The variable lists can be either explicit or implicit.

Explicit List Constraints

(These constraints consist of):

1. The keyword "AND"
2. An optional constraint name
3. A mandatory colon
4. A list of variable names
5. A relational operator
6. A constant
The variable list is simply a list of variable names separated by commas. If the "UPPER" option is not specified in the "USE" statement, those list constraints in which the constant is not 0 are considered as explicit rows in the simplex tableau. Examples:

\[
\text{AND: } X_1, X_2, X_5 \geq 0 \\
\text{AND UBND: } \text{HEAT, POWER, LIGHT} \leq 100.3 \\
\text{AND: MYVAR, YOURVAR URS} \\
\text{(Here URS = unrestricted in sign and takes the place of 5 and 6 above.)}
\]

Bounded lists are also permitted, for example

\[
\text{AND: } 4 \leq X_1, X_2, X_5 \leq 7
\]

**Implicit List Constraints**

Special abbreviations are available to describe either all variables or variables not appearing in another list constraint. Examples:

\[
\text{AND: ALL VARS} \geq 0 \\
\text{AND: ALL VARS} \leq 1 \\
\text{AND MYBOUND: ALL OTHER VARS} \geq 3
\]

The phrase "ALL VARS" is a synonym for "all variables", while the phrase "ALL OTHER VARS" refers only to those variables not included in some earlier list constraint.
3: Editing the Model

Adding Rows

New rows may be either appended to the current model file or inserted within the current model file.

a. Appending rows to the end of the model. This is easily accomplished by simply typing in the row exactly as it was done during model entry.

Example:

\[
\begin{align*}
\text{AND: } & X_1+X_2 \leq 7 \\
\text{ALSO: } & 3.5X_1+4.6\text{FLOW1}
\end{align*}
\]

b. Insertion of rows within the model file - Rows may be inserted within the model file by simply typing in:

\[
\begin{align*}
\text{INSERT AFTER } & \text{<old-row> } <\text{model entry statement}> \\
\text{or} \\
\text{INSERT BEFORE } & \text{<old-row> } <\text{model entry statement}>
\end{align*}
\]

For example, if the current model file contained:

\[
\begin{align*}
\text{MIN OBJ: } & 2X_1+3X_2 \\
\text{ST CON: } & X_1+X_2 \leq 3 \\
\text{AND: } & \text{ALL VARS } \geq 0
\end{align*}
\]

an entry of

\[
\begin{align*}
\text{INSERT AFTER CON AND CON2: } & X_1+2X_2 \geq 1
\end{align*}
\]

would result in a model file of

\[
\begin{align*}
\text{MIN OBJ: } & 2X_1+3X_2 \\
\text{ST CON: } & X_1+X_2 \leq 3 \\
\text{AND CON2: } & X_1+2X_2 \geq 1 \\
\text{AND: } & \text{ALL VARS } \geq 0
\end{align*}
\]

Deleting Rows

The deletion of a row is accomplished by entering \text{DELETE } <\text{row name}> where <row name> is the user-assigned name, or in the absence of such, takes the form "ROW#n". If a delete is followed by the addition of a row having the same name, the new row is inserted in the model file in the same place. Examples:

\[
\begin{align*}
\text{DELETE MYCSTRNT} \\
\text{DELETE ROW#7}
\end{align*}
\]

55
an entry of

\[
\text{CHANGE ROW\#5 } "3\times2+4"
\]

would result in ROW\#5 becoming

\[
\text{AND: } 2X1+X2+4X3\leq 5
\]

whereupon an entry of

\[
\text{CHANGE ROW\#5 } "3<"
\]

would yield ROW\#5 as

\[
\text{AND: } 2X1+X2+4X=5
\]

Changes involving row names and/or continuation characters (&) are illegal. Changes of this type should be accomplished by a "DELETE" followed by retyping the entire row.

Listing the Model

The list command allows the user to list the current contents of the model file in total or in part. Two options are available:

1. LIST (lists the whole model)

2. LIST <row name> (lists only row <row name>)

Examples:

```
LIST
LIST ROW\#5
LIST CONSTR3
```

Changing Rows

To change a character string in a specified row (say ROW\#5), the following is entered

\[
\text{CHANGE ROW\#5 } "<\text{string1}>"<\text{string2}>"
\]

The double quote ("\) is the only allowable delimiter. For example if ROW\#5 was originally

\[
\text{AND: } 2X1+3X\leq 5
\]
4: Solving the Model

Specifying Alternate Objective Functions

The user may specify an alternate objective function to be optimized by typing:

```
ALTOBJ MIN <row name>
```

or

```
ALTOBJ MAX <row name>
```

where <row name> is the name of an objective function in the model.

Examples:

```
ALTOBJ MAX ROW#2
ALTOBJ MIN OBJ4
```

This objective function specification remains in force until such time as a "RESTART" or another "ALTOBJ" command is entered.

The user may identify the current specified alternate objective function row name by typing:

```
ALTOBJ STATUS
```

Specifying Output

The PRINT command is used to specify the type and frequency of output to be printed during the optimization process. Parameters which can be used are:

1. **VARS** prints the values of the non-zero primal and dual variables in the current tableau.
2. **BASIS** prints the current basic variables and denotes entry and exit activities
3. **TABLEAU** prints the current tableau and the updated objective function row
4. **ALL** Synonym for "VARS,BASIS,TABLEAU"

Parameters 1-4 can optionally be followed by an integer indicating the frequency with which the output is to be given. If omitted, the default value is 1 (output after every iteration). Examples:

```
PRINT TABLEAU, BASIS
PRINT VARS 5
PRINT ALL 4
```
Initial and/or final tableau output can also be indicated. In this case, the integer described does not apply and is ignored if present. Examples:

```
PRINT INITIAL TABLEAU
PRINT FINAL ALL
PRINT INITIAL, FINAL BASIS TABLEAU
```

If print specifications are to be altered, the proper action is to enter

```
PRINT NONE
```

which resets print parameters to the default mode.

### Specifying the Method of Solution

The `USE` command is entered when an attempt to solve the input model is to be made. The following parameters are legal (the order of the parameters is important).

1. **RATIONAL** - (Optional) Rational arithmetic is to be used. Rational arithmetic is 3 to 4 times slower than real arithmetic, and can only solve problems half the size.

2. **UPPER** - (Optional) Lower-upper bounded simplex is to be used. List constraints are implicit in the tableau.

3. **PRIMAL** or **DUAL** - A specification of the type simplex method to be used. "PRIMAL" refers to the simplex method and "DUAL" refers to the dual simplex method.

Examples:

```
USE RATIONAL PRIMAL
USE DUAL
USE UPPER DUAL
USE RATIONAL UPPER PRIMAL
```

When possible, "UPPER" should be specified for dual simplex, as this improves the chance of obtaining a starting dual feasible basis.
5: Sensitivity Analysis

EZLP contains capability for sensitivity analysis of an optimal solution of a model. This is accomplished by use of a RANGE command. The format of this command is:

1. The keyword "RANGE".
2. User defined options for ranging: "RHS", "OBJ" or "OBJFCN", "ALL".
3. Optional user defined variable names or row names, or the optional modifier "ALL".

EXAMPLES:

<table>
<thead>
<tr>
<th>Command</th>
<th>Option 1</th>
<th>Option 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANGE</td>
<td>RHS</td>
<td>ROW#4</td>
</tr>
<tr>
<td>RANGE</td>
<td>OBJ</td>
<td></td>
</tr>
<tr>
<td>RANGE</td>
<td>OBJ</td>
<td>X55</td>
</tr>
<tr>
<td>RANGE</td>
<td>OBJ</td>
<td>ALL</td>
</tr>
<tr>
<td>RANGE</td>
<td>ALL</td>
<td></td>
</tr>
</tbody>
</table>
6: File-Handling

Input From an External File

The "RUN" command enables the user to use an external file as a store of input statements and commands. Upon entering the command RUN <file name>, EZLP opens the local file <file name> and references this file for all subsequent input commands, until such time as an "END" statement is encountered or an end-of-file condition on <file name> exists. If an end-of-file is encountered, control returns to the live user. The form of the file should be 80 characters per line.

Output to an External File

The "SAVE" command enables the user to place the current concepts of the model file into a local file. The form of this file is 80 characters per line and editable by the system editor or usable as input to a later execution of EZLP. ENTER:

SAVE <file name>
6: Batch Processing

EZLP can be used in a batch environment in much the same way that it is used interactively. The sole restriction is that the command BATCH must be the first entry in the input deck to EZLP. Omission of this command can result in an infinite print loop at job termination time. Aslo the command END must be the last entry in the input deck to EZLP.
TITLE ** ADVANCED MODEL **
MAX OBJ: 2INTERST1+3.24INTERST2-5CAPITAL
ALSO OBJ2: 3.4INTERST1+2.54INTERST2-5 CAPITAL
ST CONSTR1: INTERST1-4INTERST2 = 108
AND MINCAP: CAPITAL> = 4.6 INTERST1 + 5.7INTERST2
AND MAXCAP: 4 CAPITAL< = 10345.65 - 3.224 INTERST1 &
-5.231 INTERST2
AND CAPBND: 800 <=CAPITAL <= 10000
AND INTBND: INTERST1, INTERST2 <=10825
AND NONNEG: INTERST1, INTERST2 >= 0
CHANGE MINCAP "4.6"4.76"
CHANGE INTBND "825".825"
PRINT ALL
USE UPPER PRIMAL
DELETE NONNEG
AND NEWCSTR: INTERST2 >= 1.342
AND UBND: INTERST1, INTERST2,CAPITAL <= 100000.3
ALTOBJ MAX OBJ2
PRINT FINAL TABLEAU, BASIS, VARS
LIST
USE UPPER DUAL
END