FORTRAN BASED LINEAR PROGRAMMING
FOR
MICROCOMPUTERS

THESIS

AFIT/GOR/OS/82D-4

THEODORE R. E. FRALEY  DALE A. KEM
MAJOR      USAF      CAPTAIN USA

Approved for public release; distribution unlimited
FORTRAN BASED LINEAR PROGRAMMING

FOR

MICROCOMPUTERS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Theodore R. E. Fraley
Major USAF

and
Dale A. Kem
Captain USA

Graduate Operations Research
December 1982

Approved for public release; distribution unlimited
We would like to thank our thesis advisor, Major Daniel B. Fox, for his guidance and support throughout the AFIT program and specifically during the period of our thesis research. Major Fox and our reader, Major Gerald R. Armstrong, have expended great effort and time to insure that this research was a learning experience and that a beneficial software package was produced. We would like to express our gratitude to Lt/Col Ivy D. Cook for his assistance during the research.

We also wish to thank our families, especially our wives, Gail and Jean, for their support, typing and editing assistance, and, most of all, their patience during this time.
# CONTENTS

Preface .................................................. ii
List of Figures ......................................... iv
List of Tables ........................................... v
Abstract ................................................ vi

I. Introduction ............................................ 1

- Linear Programming ................................. 2
- Computer State-of-the-Art .......................... 3
- Current Microcomputer LP Software Development. 7
- Motivation for Further Research .................. 10

II. Theoretical and Mathematical Background ....... 12

III. Design Considerations .............................. 44

- User Considerations .................................. 44
- Hardware Considerations ............................. 46
- Language Considerations ............................. 48

IV. Implementation ....................................... 52

- Hardware ............................................... 53
- Language ............................................... 54
- User Interface ......................................... 55
- Software Description ................................. 59
  Module 1 - Data Base Entry Module ................. 60
  Module 2 - LP Instructional Module ................. 65
  Module 3 - Problem Solver Module ................. 71
  Module 4 - Sensitivity Analysis Module .......... 73
  Summary ............................................... 77

V. Conclusions and Recommendations ................ 79

Bibliography ............................................. 83

Appendix A: Users Guide ................................ 85

Appendix B: Programmers Guide ....................... 177

Vita ....................................................... 429
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Graphical Solution</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Effect of Single Change to the Original Tableau</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>Column, Constraint Relationship</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>Change to the Objective Function Coefficient</td>
<td>37</td>
</tr>
<tr>
<td>5</td>
<td>Right-Hand-Side Ranging</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>A Two-Dimensional Problem</td>
<td>39</td>
</tr>
<tr>
<td>7</td>
<td>Corner Point at Infinity</td>
<td>41</td>
</tr>
<tr>
<td>8</td>
<td>Developmental Hardware Configuration</td>
<td>54</td>
</tr>
<tr>
<td>9</td>
<td>Minimal Hardware Configuration</td>
<td>78</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Tableau Form of Simplex Algorithm</td>
<td>27</td>
</tr>
<tr>
<td>II</td>
<td>Initial Basic Solution</td>
<td>28</td>
</tr>
<tr>
<td>III</td>
<td>Second Basic Solution</td>
<td>30</td>
</tr>
<tr>
<td>IV</td>
<td>Third Basic Solution</td>
<td>30</td>
</tr>
<tr>
<td>V</td>
<td>A Final Tableau</td>
<td>40</td>
</tr>
<tr>
<td>VI</td>
<td>Modified Problem, Final Tableau</td>
<td>41</td>
</tr>
<tr>
<td>VII</td>
<td>Division By Zero</td>
<td>42</td>
</tr>
</tbody>
</table>
Linear programming is an analytical technique used in decision analysis. This paper describes the development and use of a highly interactive, non-programmer oriented, linear programming software package implemented on a microcomputer. This software, written in FORTRAN and supported by the UCSD Pascal Operating System, has allowed increased portability while providing the capability of solving moderate-sized LP models. Also available are extensive postoptimal sensitivity analysis capabilities.

The modularly implemented package provides interactive, instructional sessions with user input LP models. The user is guided through tableau formulation and pivot element selection to an optimal solution by a series of option displays and user selections. This module also provides instructors the ability to rapidly demonstrate the application of the simplex algorithm.

A separate module provides a more rapid problem solution with minimal interaction. Options allow either primal or dual problem solution with screen-oriented output to either a monitor or printer. The sensitivity analysis capabilities include right-hand-side, cost coefficient, and constraint coefficient ranging. Also provided is the ability to add constraints and variables to the original model.
Managers at all levels of private and public organizations are continuously confronted with the burden of decision making and the subsequent accountability for such decisions. The criticality of these decisions may not be immediately obvious to the manager or to the organization, yet the outcome may contribute to the success or the failure of the organization.

A large fraction of these decisions involve the amount of organizational resources, such as manpower, equipment, or funds, to dedicate to a particular project or operation. Although one could attempt to dedicate the necessary resources required to maximize the output of each operation, one would soon realize a shortage in one or many of these organizational assets. One might then attempt purely subjective evaluations of the worth of various projects and allocate resources based upon this process. However, for high level managers of diverse organizations, this may be beyond the bounds of comprehension due to the magnitude of
activities under their control.

Therefore, managers have sought methods which will allow a systematic and accurate analysis of numerous operations in a timely manner. A decision which is accurate, but late, may be of less value than an inaccurate decision which has been made in sufficient time. This search has lead to the development and implementation of several mathematical programming techniques. One important subset of these mathematical programming techniques is linear programming.

**Linear Programming**

Mathematical programming, which consists of several specific optimization techniques, has been defined as the use of mathematical representations (models) to plan (program) an allocation of scarce resources among competing activities. Linear programming is one such technique commonly used by analysts and is an optimization technique which involves only linear mathematical relationships (Ref 4:35). Although the term "programming" is used, in this context it does not refer to computer programming. The term in this setting refers to the selection of a particular course of action or program and is a synonym for planning.

Although mathematical optimization techniques have been present for many years, the last three decades have shown a great increased use of quantitative tools in aiding managerial decision making (Ref 22:XI). Many techniques
have been developed in this new realm of application, but none so popular as linear programming. George B. Dantzig and his associates first developed and applied this technique in 1947 following a proposal that the interaction of the activities in an organization may be viewed as linear relationships (Ref 10:IX). In conjunction with this development, Dantzig also proposed the simplex algorithm which has been shown to be a systematic procedure for the solution of such linearly defined problems.

The linear programming technique, although fairly recent, was estimated to account for 25 percent of all scientific computations in 1970 (Ref 22:XVI). The extensive use of such a technique coupled with the increased use of the computer for all types of computational procedures has lead to the development of extensive software packages implementing linear programming on mainframe computer systems. These software packages are capable of quickly solving problems consisting of hundreds of variables and constraints. The manager now has the capability of performing complex linear programming computations within a matter of minutes.

**Computer State-of-the-Art**

Managers now have the analytical tools and computational capability to solve linear programming problems, but is the computational power accessible? As mentioned previously, extensive linear programming software
packages have been developed and implemented on large mainframe computer systems. These computer systems have essentially unlimited storage capabilities and very rapid computational rates. These combined capabilities have given rise to the problem solving capabilities previously mentioned. However, availability of these large systems is somewhat limited due to the large acquisition expense and stationary support requirements. Also, access to such systems may be limited to those managers who are operating in the immediate vicinity of such systems. This is particularly true for military leaders who may be operating in remote locations yet still require quantitative decision analysis support.

Recent advances in communications links now allow the use of remote terminals and peripherals which greatly reduce the problem of computer accessibility. However, due to the increased use of computers in all aspects of management, the number of users attempting to access the computer is normally quite large. This aspect may then cause the response time of non-dedicated remote computer systems to be unacceptable in a time critical environment.

The recent explosive development of the microcomputers or "desk-top" computers may offer a solution to many of the problems associated with the large mainframe computer systems. Prior to the late 1970's, microcomputers were little more than toys, characterized by very limited memory
capabilities, difficult input/output procedures, and awkward data storage facilities. From that meager start, the capabilities of microcomputers radically increased. Most business oriented microcomputers have a random access memory (RAM) of 64,000 (64K) bytes (approximately 64,000 characters) with the capability to expand to 256K RAM. The data storage medium has advanced from slow cassette tape to floppy disks to the present hard disks which can store many millions of bytes per disk.

Noteworthy advances have also been achieved in the programming languages available for use with these microcomputers. Until recently, microcomputers were usually limited to the machine specific BASIC language as the only available high-level language. Now, many microcomputers support more universal and powerful languages such as FORTRAN, PASCAL, and APL. It is the availability of these languages, based upon standardized rules, which has allowed increased portability of programs from machine to machine.

The recent microcomputer developments coupled with the even more recent language availability to these machines have surmounted the initial obstacles to the use of microcomputers for application of quantitative analysis techniques. However, the development of software has yet to be considered. The development of software for mainframe computer systems has occurred over several years. Also, software packages are readily available which allow the
non-programming oriented manager access to efficient techniques applying both general and specific problem solving methods. This option is not so readily available for the more recent microcomputers. Although the development of software for the microcomputers is ever increasing, the packages often require the user to be quite knowledgable in programming in order to use the specific decision analysis aids.

Software availability is not the only difficulty that the microcomputer user will encounter. In exchange for the ease of availability, accessibility, and dedicated computational support, the user of "desk-top" computers will find a marked decrease in memory capabilities and computation rates. The limited memory capabilities greatly reduces the problems which may be attempted. The marked decrease in computational speed will considerably increase the length of time required to obtain results.

The above stated problems are not insurmountable, yet are serious limitations imposed by the use of microcomputers. The most serious problem is the critically limited, if existent, availability of user-oriented, portable software for the microcomputers. This problem renders the recent microcomputers virtually useless for the managers who have insufficient background and, possibly even more critical, insufficient time to formulate and implement a decision making algorithm when needed. In order for the
advantages of the microcomputer to be extended to a larger percentage of the decision makers, software development is required of the various mathematical modeling algorithms, and in particular, the intensely used linear programming algorithm. Although the microcomputer cannot replace the large mainframe computer systems, it may prove to be a supplement in areas of moderately sized problems and greatly aid in a more rapid response to less complex problems.

Current Microcomputer LP Software Development

A literature search conducted in June, 1982 revealed only one non-proprietary microcomputer linear programming software package documented. This analysis package, developed by Robert D. Conte (Ref 6), consists of several analysis techniques, including linear programming, implemented on an Apple II microcomputer in its machine specific language Applesoft. This interactive package has been well designed and implemented with true consideration for the non-programmer oriented user. Although the linear programming portion is capable of solving problems consisting of twenty constraints and twenty variables, true sensitivity analysis was not available. However, due to its extensive editing features, one may respecify various parameters and resolve the problem to arrive at equivalent sensitivity analysis results.

Four microcomputer based linear programming software developments were recently discussed and displayed at the
TIMS/ORSA meeting in Detroit, Michigan during April, 1982. The first which will be discussed was developed by Ralph W. Swain (Ref 20). Its primary purpose was that of graphical demonstration of techniques commonly utilized in operations research. Although a great aid in demonstrating the behavior of systems, its use to the manager and analyst is somewhat limited.

Rolf A. Daininger (Ref 7) has developed and implemented an instructional aid which will display the various iteration’s tableaus for a maximum of nine constraints and twenty variables. Implemented on an Apple II microcomputer in Applesoft, it has proven to be a great aid in allowing students to concentrate on the simplex algorithm methodology and solution process rather than the numeric operations involved.

Gary E. Whitehouse and Yassar A. Hosni (Ref 21) presented an extensive software package consisting of forty-two small problem oriented programs. Several of these programs directly or indirectly involved linear programming. Examples are the application of the simplex algorithm to an LP problem and graphical solution of a two variable LP problem. An advantage of these programs is that even though written in BASIC, versions are available for both the Apple II and the TRS-80 microcomputers. Although each version is not portable between these or other systems, a larger potential set of users have access to such software.
The last current development in LP software was presented by Byron Gottfried (Ref 12). This package, as implemented on the Apple II microcomputer, is capable of solving problems of approximately forty-five constraints and ninety variables (after the augmented basis has been implemented). Another version has been implemented on an IBM microcomputer which has larger capabilities and this version includes limited sensitivity analysis. The sensitivity analysis included is right-hand-side and cost-coefficient ranging within the present feasible solution. Both versions were developed in their respective machine specific BASIC language and are currently not portable to other machines or between the two target systems.

The linear programming software presently found to exist for microcomputers has been implemented in the respective machine specific BASIC languages. Although each package individually is of significant value, each has its limitations in both significance and applicability. It would be advantageous to construct a single package which implements many of those already implemented plus expands the capabilities in many areas. Particular emphasis may be desired in the area of instructional aids designed for use by both instructors and students. Although the work of Daininger (Ref 7) has allowed the instructor to more easily demonstrate the computations of the simplex algorithm,
little has been done in the area of software development for independent student use. Such software could allow the student with minimal linear programming background to reinforce the application of a linear programming solution technique to an LP problem. Also, if this implementation was in a high-level language which was more portable, it would enhance such a development even more.

Motivation for Further Research

To insure a tool is utilized to its potential, it must be developed with the user needs as a primary consideration. Also, the availability and accessibility of such a tool must be maximized for users to consider its use beneficial. Linear programming is no exception.

The problem addressed in this research was the development and implementation of a linear programming software package which allows the user extensive problem solving capabilities of small LP problems on a microcomputer system. Although the package was planned for use by analysts requiring responsive dedicated decision analysis support, features may be incorporated which will allow students and instructors of linear programming to be beneficial users. The package was developed with ease of user interface and minimum programming experience as primary considerations as well as the desire for maximum portability between available microcomputer systems. These objectives have lead to a modular package design with the requirement
for user interaction being dependent upon user desires. The aforementioned goals must be balanced in light of the limitations as well as the advantages offered by a dedicated microcomputer system.
II Theoretical and Mathematical Background

Linear programming, as has been mentioned previously, is a very powerful optimization technique commonly used by today's leaders and managers. Although the subject of linear programming is found in numerous text and reference books which a manager may review, each approach the subject in different manners and elaborate to different levels of detail. Some discuss the theoretical development and background, others the methodology, and yet others focus primarily on the application of the optimization algorithms to specific type problems. This wide spectrum of literature may cause an aspiring manager to misinterpret the true power and validity of these techniques if an overview of the subject can not be captured.

The purpose of this chapter is to provide the reader an insight into the theoretical development of linear programming, with emphasis on the simplex algorithm. This theoretical background will be presented in conjunction with the simplex algorithm methodology in hopes of assisting the reader in gaining a more thorough understanding of the simplex algorithm and its application to problem solving.

For our purposes, the LP model to be discussed will be as shown below in EQ(1) through EQ(3). The dimension of $m$ represents the number of functional constraints, excluding nonnegativity constraints, in EQ(2). The dimension of $n$
represents the number of variables in the original problem including the slack variables required to transform the constraints into the equality form as shown in EQ(2) below. Therefore, the LP model is:

maximize \( z = CX \) \hspace{1cm} (1)

Subject to

\[ AX = B \] \hspace{1cm} (2)

\[ X \geq 0 \] \hspace{1cm} (3)

where

- \( z \) = scalar value of objective function
- \( C \) = row vector of dimension \( n \)
- \( X \) = column vector of dimension \( n \)
- \( A \) = \( m \times n \) matrix
- \( B \) = column vector of dimension \( m \)
- \( Q \) = \( n \) dimensional null vector

A few definitions will be presented to provide a basis for further discussion. First, a feasible solution to an LP problem is an \( n \) dimensional vector \( X \) which satisfies EQ(2) and (3) above. Therefore, each element of the vector is nonnegative and provides a solution to EQ(2). A basic solution is also an \( n \) dimensional vector \( X \) which satisfies EQ(2); however, a maximum of \( m \) elements of this vector are nonzero elements. A basic feasible solution is a basic solution which also satisfies EQ(3). Therefore, a basic feasible solution contains a maximum of \( m \) elements (called the basic variables) which are nonnegative with the remaining \( (n-m) \) elements (called nonbasic variables) having a value of zero. A basic feasible solution which contains
fewer than a nonzero elements is called a degenerate solution. An optimal solution is a basic solution which also maximizes the value of \( z \) in EQ(1). If the optimal solution is also feasible, that is, EQ(3) is satisfied, then the solution is an optimal feasible solution. Otherwise, the solution is optimal but infeasible (superoptimal) for the LP problem as stated.

To introduce the simplex method, one may want to first review the geometric considerations of the problem. As the problem is stated, there are \( m \) functional constraints, represented by EQ(2), which may or may not be redundant. Also, \( n \) nonnegativity constraints are imposed by the problem. Considering a two dimensional space \((n=2)\) and three constraints \((m=3)\), one might find a graphical depiction of a problem as shown in Figure 1.

From the graph and the constraints shown below it, it should be recognized that the shaded area is the solution space of this problem (Note: the nonnegativity constraints are enforced in the graphical depiction). This solution space and the solution space for all LP problems forms a convex set, and therefore the convex combination of any two points in the solution set is also in the solution set (Ref 10:50). This solution set is bounded by a finite number of linear constraints which further implies that there are a finite number of intersection points of these constraints. It has been further shown that any point in a non-null
convex set may be represented by a convex combination of the extreme points (annotated by A, B, C, D, and E in Figure 1) of the convex set (Ref 10:29). The above discussion implies that of the infinite possible solutions to an LP problem, all may be represented by a convex combination of a finite number of solution space extreme points.

Assume that a solution set to an LP problem exists. Also assume that this solution set consists of an infinite number of points and that each solution in this set may be
represented by an n dimensional vector $\bar{x}$. It has been proven that a point in n space, which includes our solution set, may be represented by the interaction of m linearly independent vectors, where $m \leq n$ (Ref 19:62-71). As a result, any point represented by m linearly independent n dimensional vectors will contain at most m nonzero elements and at least $(n-m)$ zero elements.

It may be shown that the objective function, Eq(1), assumes its optimal value at an extreme point of the convex set or, if at more than one, the objective value, $z$, is the same for all convex combinations of these extreme points (Ref 10:50-51). Therefore, only the extreme points of the convex set must be investigated in the search for an optimal solution. The number of extreme points, although possibly large, is finite and greatly reduces the number of points which require investigation to determine the optimal solution. If a set of m linearly independent vectors may be found, with the solution vector containing at most m nonnegative elements and at least $(n-m)$ zero elements, the solution corresponds to an extreme point of the convex set (Ref 10:53).

The above implies that only the extreme points generated by m linearly independent vectors must be investigated in the search for an optimal solution. Consider now the A matrix in Eq(2) and envision each column of the matrix as an m dimensional vector $\bar{y}$. Although $m$
linearly independent \( V \) vectors may not be readily identifiable in the problem initially, the matrix may be augmented by a set of \( m \) linearly independent vectors to provide this set of \( m \) linearly independent vectors. With this \( m \) dimensional basis, it is known that, at most, \([n!/m!(n-m)!]\) possible solutions exist and may require investigation since this is the number of combinations of \( m \) vectors from a set of \( n \) vectors (Ref 9:31).

Up to this point, it has been shown that of the infinite number of solutions which may exist, at most \([n!/m!(n-m)!]\) require investigation. But now it must be asked, how are these extreme points determined? Again envision the \( A \) matrix consisting of \( n \) \( m \)-dimensional vectors \( V(1) \) through \( V(n) \). Assume that the first \( m \) vectors are linearly independent and that \( \bar{x} \) is a basic feasible solution. In this form, EQ(2) may be expressed as follows:

\[
\mathbf{x}_b(1)\mathbf{v}(1) + \mathbf{x}_b(2)\mathbf{v}(2) + \ldots + \mathbf{x}_b(n)\mathbf{v}(n) = \mathbf{B} \quad (4)
\]

where

\[
\mathbf{x}_b(i) \text{ are the elements of the basic feasible solution } \bar{x} \\
\mathbf{x}_b(i) \geq 0 \quad i = 1, \ldots, n
\]

It has been assumed that \( V(1) \) through \( V(m) \) are linearly independent. It will be further assumed that a nonnegative combination of these vectors equals the vector \( \mathbf{B} \). Therefore, EQ(4) may be now expressed as shown in EQ(5) below. Note that the elements of \( \bar{x} \) from \( \mathbf{x}_b(m+1) \) through
\( x^n \) are equal to zero due to the linear independence of \( V(1) \) through \( V(m) \).

\[
xb(1) V(1) + xb(2) V(2) + \ldots + xb(m) V(m) = B \quad (5)
\]

It is known that any of the \( n \) vectors may be represented as a linear combination of the basis vectors. Therefore a vector \( V(k) \), where \( k > m \), may be represented as follows:

\[
xk(1) V(1) + xk(2) V(2) + \ldots + xk(m) V(m) = V(k) \quad (6)
\]

where \( xk(i) \) represents the weight of the \( i \)(th) vector in the linear combination forming \( V(k) \) when \( V(k) \) is the selected entering basis vector.

To determine a new solution vector \( x \) to \( EQ(5) \) which includes at most \( (m+1) \) nonzero elements, we may multiply \( EQ(6) \) by some value, say \( T \), then subtract it from \( EQ(5) \) to find:

\[
[xb(1) - T xk(1)] V(1) + [xb(2) - T xk(2)] V(2) + \ldots + [xb(m) - T xk(m)] V(m) = B - T V(k) \quad (7)
\]

or

\[
[xb(1) - T xk(1)] V(1) + [xb(2) - T xk(2)] V(2) + \ldots + [xb(m) - T xk(m)] V(m) = B + T V(k) \quad (8)
\]

(assume one \( xk(i) \geq 0 \) for \( i=1, \ldots m \))

The solution vector \( x \) is \( n \) dimensional, but now contains at most \( (m+1) \) nonzero elements. It must now be
noted that only those vectors with no more than \( m \) nonnegative elements are desired and are possible basic feasible solutions. Therefore, the solution vector for EQ(B) must contain no more than \( m \) nonnegative elements to represent an extreme point of the convex set. The problem at this point is to determine the value of \( T \) which will force one of the elements in the EQ(B) solution vector to zero, thereby forcing one of the previous vectors \([V(1) \text{ through } V(m)]\) out of the basis.

With the above insight, it may be determined that if the multiplier \( T \) is positive, only those values of \( x_k(i) \) which are positive need be checked. If the value of \( x_k(i) \) was nonpositive, the solution element \([xb(i)-T\times x_k(i)]\) would always be positive and would not approach zero. To determine which element of EQ(B)'s solution will be forced to exactly zero, it must also be considered that all other elements which are nonnegative must remain nonnegative in the new solution. Therefore we want to find that element which first reduces to zero. For \( i=1, \ldots, m \), the element which first achieves

\[
x_b(i)-T\times x_k(i) = 0
\]

is the coefficient of the vector which will be forced out of the basis. This may be formulated to be:

\[
T = \min_{i=1}^{m} \left[ \frac{x_b(i)}{x_k(i)} \right]
\]
Using the above value for $T$, one may determine the new basic solution corresponding to another extreme point of the convex set.

One should recognize that if a vector is selected to enter the basis, $[Y(k)]$ in this example and it is found that all $x_k(i)$ are less than zero, this basis will contain $m+1$ elements. Since the new solution cannot be expressed as a basic solution, that is, a nonnegative elements in solution vector, it does not correspond to an extreme point and therefore is not a basic solution. This situation indicates that the problem has no finite maximum solution (unbounded) and the solution process is terminated.

From the previous discussion, one may find all extreme point solutions by enumeration and evaluate each of these solutions in terms of $EQ(1)$ to determine which basic feasible solution produces the maximum objective value $z$. Although the above process could, in theory, be performed, the number of calculations increases exponentially as the number of variables $(n)$ and constraints $(m)$ increase. The discussion of the simplex algorithm will show that once an initial basic feasible solution is found, an optimal solution may then be found, if it exists, in a finite number of steps. The simplex algorithm allows the user to find only those basic feasible solutions which have an objective function of equal or greater value than the present solution. Also, the algorithm identifies for the user when
the optimal solution has been obtained or that a optimal solution does not exist (known as unbounded solution).

To illustrate the methodology of the simplex algorithm in light of the theoretical background, consider the numerical example given earlier. If one would express this problem in the stated form, each constraint would have a slack variable added to form an augmented basis as shown in EQ(12) through EQ(14). The initial basis feasible solution is $x(3)=4$, $x(4)=6$, and $x(5)=8$ since these vectors provide the basis. EQ(15) states the equivalent mathematical relationship of EQ(12) through EQ(14) in the form presented earlier [EQ(5)].

Maximize $z = x(1) + 2x(2) + 0x(3) + 0x(4) + 0x(5) = 0$ \hspace{1cm} (11)

Subject to

$x(1) + x(3) = 4 \hspace{1cm} (12)$
$x(2) + x(4) = 6 \hspace{1cm} (13)$
$x(1) + x(2) + x(5) = 8 \hspace{1cm} (14)$

$x_b(3)x_3 + x_b(4)x_4 + x_b(5)x_5 = b \hspace{1cm} (15)$

Now if each basic variable is expressed in terms of only the non-basic variables, $x(1)$ and $x(2)$, the following is found:

$x(3) = 4 - x(1) \hspace{1cm} (16)$
$x(4) = 6 - x(2) \hspace{1cm} (17)$
$x(5) = 8 - x(1) - x(2) \hspace{1cm} (18)$

The objective function may then be expressed in terms of the
nonbasic variables to arrive at:

\[ z = 0 + x(1) + 2x(2) \]  \hspace{1cm} (19)

At this point, review the previous section's thoughts. A basic feasible solution exists; \( x = (0, 0, 4, 6, 8) \) and the basis is formed by \( n-3 \) linearly independent vectors \([y(3), y(4), y(5)]\). At this point, we would select a vector not in the basis \([y(1)\) or \(y(2)\)] to determine whether this new solution is also a basic feasible solution. Previously, no method has been discussed to select the incoming basis vector. Each nonbasis vector could have been selected to enter; however, random vector selection may cause the value of the objective function to decrease and move away from its optimal value. The simplex algorithm assists in this selection in that it guides the user to select an incoming basis vector which will increase, or at least maintain, the current objective function value. Looking at EQ(19), one will find that the current objective function value is zero since both \( x(1) \) and \( x(2) \) are currently nonbasic variables.

To increase the objective function, either \( y(1) \) or \( y(2) \) may be selected to enter the basis since both have positive coefficients in the objective function. The objective function value, \( z \), will increase as the value of the incoming variable increases since it has been stated that the variables \([x(1), x(2)]\) must be nonnegative. However, if \( y(2) \) is selected, the objective function value will increase.
at a rate or slope of two while \( V(1) \) would only increase the objective value at a rate of one. Therefore, the simplex algorithm will guide the user to select as the entering basis vector, that vector which will cause the most rapid increase in the objective function. One could at this time perform the computations of EQ(6) through EQ(10) to determine the new basis. These calculations will not be performed at this time but will be shown later using the simplex algorithm from beginning to end for the example problem.

Now that the logic for selecting an entering basis vector has been displayed, the theoretical development will be reviewed. Recall that the constraints of the problem were given in EQ(5). The objective function may be then expressed as follows:

\[
xb(1)c(1) + xb(2)c(2) + \ldots + xb(m)c(m) = z \tag{20}
\]

where \( c(i) \) for \( i=1, \ldots, n \) are the cost coefficients of the objective function.

Also recall that \( X \) has been assumed to be a basic feasible solution and that any vector \( V(n) \) may be expressed as a linear combination of the basis vectors, \( V(1) \) through \( V(m) \). At this point, let us define a term \( z(j) \) as

\[
z(j) = \sum_{i=1}^{m} a(i,j)c(i) \quad j=1, \ldots, n \tag{21}
\]

where \( a(i,j) \) are the \( i(\text{th}) \) coefficients of the \( j(\text{th}) \) vector \( V(j) \).
c(i) represents the cost coefficient of the basic variable of row i.

The element z(j) for vector j [V(j)] could be enumerated as:

\[ z(j) = a(1,j) * c(1) + a(2,j) * c(2) + \ldots + a(m,j) * c(m) \] (22)

The z(j) element has been defined by Hillier & Lieberman (Ref 13:88) as the net amount by which the initial coefficients in the objective function have been increased by the simplex method.

For a fixed value of j, if \( z(j) - c(j) < 0 \), a feasible solution exists in which the new value of z is greater than or equal to the current value of z (Ref 10:66-67). The proof of this was shown by multiplying EQ(6) by some value, T in our example, and subtracting this from EQ(5). The results of this are shown in EQ(8). Also as part of this proof, EQ(22) was multiplied by T and subtracted from EQ(20). The results are shown below.

\[
[xb(1) - T*a(1,k)]*c(1) + [xb(2) - T*a(2,k)]*c(2) + \ldots + [xb(m) - T*a(m,k)]*c(m) + T*c(k) = z - T*[z(k) - c(k)] \] (23)

Note that T*c(k) has been added to both sides of EQ(23).

EQ(23) represents the objective function of a feasible solution, assuming the coefficients of V(1)...V(m), V(k) are positive. Also note that if a \([z(k) - c(k)] \) exists which is negative and assuming that T is positive, the right hand side of EQ(23) will increase in value beyond z, the previous
objective function value. This means that a feasible solution exists which possesses a higher objective value and that the simplex procedure requires further iterations to obtain optimality. In order to increase the objective function at the greatest rate, the simplex algorithm directs the user to select as the entering basis vector that vector which has the largest negative \([z(k)-c(k)]\) value.

If a negative \([z(k)-c(k)]\) does not exist for a problem, the current basic feasible solution is optimal. This condition then allows the simplex algorithm to be terminated (Ref 10:67).

For purposes of illustration, assume that a negative \([z(k)-c(k)]\) exists and therefore an entering basis vector (basic variable) may be identified. At this point, one would determine the leaving basis vector by calculating the multiplier \(T\) as in Eq(10). Note that in Eq(8) and Eq(10), \(x_k(i)\) is equivalent to the \(a(i,k)\) in Eq(21), (22), and (23) where \(k\) represents the vector \(\nu(k)\), the selected entering basis vector. Also remember that only those \(a(i,k)\) [or \(x_k(i)\)] coefficients which are positive need to be checked since a basic feasible solution is being sought. Once the value of \(T\) is determined, one would solve Eq(21) and (22) for the new objective value and Eq(8) for the basic variable values. However, the \(a(i,k)\) values [or \(x_k(i)\)] of Eq(8) have not yet been determined, so one further step is required.

Assume that vector \(k [\nu(k)]\) is the entering basis
vector [largest negative $z(j)-c(j)$]. Eq(6) shows that $\nu(k)$ may be expressed as a linear combination of the basis vectors $[\nu(1) \text{ through } \nu(m)]$. Also assume that $\nu(1)$ has been found to the leaving basis vector $[T = \min (xb(i)/a(i,k))]$ where $i$ has been found to be 1). One may then express $\nu(1)$ as $[\text{from Eq}(6)]:$

$$\nu(1) = [1/a(i,k)]\nu(1) - \sum_{i=1}^{m} xk(i)\nu(i) \quad (24)$$

where

$$i=1,2,\ldots, l-1, l+1, \ldots, m$$

Eq(24) may then be substituted into Eq(5) to arrive at Eq(25).

$$\left(\frac{xb(1)}{a(l,k)} - \frac{xb(1)/a(1,k)}{a(1,k)}\right)\nu(1) + \ldots$$

$$+ \left(\frac{xb(1)/a(l,k)}{a(l,k)}\right)\nu(k) + \ldots$$

$$+ \left(\frac{xk(m)}{a(l,k)} - \frac{xb(1)/a(1,k)}{a(1,k)}\right)\nu(m) = B \quad (25)$$

The new solution $x$ is then found to include the basic variables $x(1), x(2), \ldots x(l-1), x(l+1), x(m)$ and $x(k)$.

The above process may be performed, however the time required would be excessive for any significant number of basis changes. The simplex method shortens this process considerably by constructing a "tableau" which contains only the coefficients of the objective function and constraints in a form which greatly simplifies the above manipulations. Although different references form this tableau in slightly different manners, all perform the same function. The form for this review will be as shown in Table 1.
TABLE I
Tableau Form Of Simplex Algorithm

<table>
<thead>
<tr>
<th></th>
<th>z</th>
<th>x(1)</th>
<th>x(2)</th>
<th>x(m)</th>
<th>x(m+1)</th>
<th>x(n)</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj fun</td>
<td>1</td>
<td>0</td>
<td>a(1,1)</td>
<td>a(1,2)</td>
<td>a(1,m)</td>
<td>a(1,m+1)</td>
<td>a(1,n)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>a(2,1)</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>a(3,1)</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>i</td>
<td>0</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>a(m,1)</td>
<td>a(m,2)</td>
<td>a(m,m)</td>
<td>.</td>
<td>.</td>
<td>b(m)</td>
</tr>
</tbody>
</table>

To place the objective function in the proper form for the tableau, it must be in a maximize \( z - \sum c(j)x(j) \) form for the \( c(j) \)'s to be in the \(-c(j)\) form. For our numerical example, this would correspond to \( z = x(1) - 2x(2) = 0 \). Had the problem been a minimization problem, one would simply multiply the entire objective function by \(-1\) and this would then represent an equivalent maximization problem (Ref 10:78). The constraints must be first converted to equalities which may be done by adding a "slack" variable to each constraint (assume that the inequalities are less-than inequalities for the moment). It should be noted that these slack variables form the initial basis of this problem. With the above modifications, the initial tableau would be as shown in Table II.
TABLE II
Initial Basic Solution

<table>
<thead>
<tr>
<th></th>
<th>x(1)</th>
<th>x(2)</th>
<th>x(3)</th>
<th>x(4)</th>
<th>x(5)</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

If the material which has been discussed is now applied, one would first determine the entering basis vector or basic variable. Only two possibilities exist for entering variables and the simplex method directs the selection of that variable with the largest negative \([z(j)-c(j)]\). Variable \(x(2)\) would then be selected for the entering variable. Next, one would determine the value of \(T\), the multiplier which will force one of the present basic variables \([x(3), x(4), \text{ and } x(5)]\) to exactly zero while maintaining the other basic variables at a nonnegative level. The values of \(T\) which are found by EQ(9) are:

\[
\begin{align*}
i=1 & \quad T = 4/0 = \text{undefined} \\
i=2 & \quad T = 6/1 = 6 \text{ (minimum)} \\
i=3 & \quad T = 8/1 = 8
\end{align*}
\]

Therefore, \(T\) would equal 6 in this iteration and identifies
the basic variable of row 2, \( x(4) \), as the leaving basic variable. In the notation of this review, 1 is the leaving basic variable of row 2 while \( k \) is the entering basic variable, column 4. Each element of the tableau may be transformed to that corresponding to the new basis using the following formulas (Note: \( x(i,j) \) represents any element of the tableau):

\[
x'(i,j) = x(i,j) - \left[ x(i,j)/x(1,k) \right] x(i,k) \quad i \neq 1 \quad (26)
\]

\[
x'(i,j) = x(i,j)/x(1,k) \quad i = 1 \quad (27)
\]

The above formulas are simplifications of Eq(25) and are applicable to all rows and columns of the tableau (Ref 10:74).

If one applies the above, one will find a new tableau as is depicted in Table III. One would see that a negative \( [z(j) - c(j)] \) exists \( [x(1)] \) so an optimal solution has not yet been obtained. Therefore one would select \( x(1) \) as the entering basic variable. Performing this iteration, one would find that shown in Table IV.

One would examine this tableau and find that no negative \( [z(j) - c(j)] \) values exist which indicates an optimal solution. The solution vector \( \bar{x} = (2, 6, 2, 0, 0) \) has all nonnegative elements indicating that it is also feasible. One should also check for degeneracy, which means that fewer than \( m \) elements of the basic feasible solution are nonzero. Since the basis dimension \( (m=3) \) equals the number of nonzero
variables, the solution is also nondegenerate. One would note that the same solution was found earlier in the chapter by the graphical method.

If the graph is inspected, it will be found that we initially started at the extreme point labeled E \( [x(1)=x(2)=0] \) in the first tableau. Next, \( x(2) \) entered the basis and we moved to point A \( [x(1)=0, x(2)=6] \). The final
tableau corresponds to point B [x(1)=2, x(2)=6]. Had we not used the selection rule of the largest negative \([z(j)-c(j)]\) for the entering basis vector, we could have selected \(x(1)\) as the first entering basis vector. This pivot would have moved us from point E to point D, and then next to point C, and finally to point B. Although the simplex selection rule does not always cause fewer pivots to be performed, this is an example of it doing so.

One area which has been passed over is that of unboundedness. Problems may arise which have no optimal solution since a basic variable or variables may be increased indefinitely without forcing another vector out of the basis. This occurs when a negative \([z(k)-c(k)]\) exists but all \(a(i,k)<0\) for \(i=1,...,m\). This means that \(T\) may be made arbitrarily large, the basis is \(m+1\) and the objective function increases without bound. If \(\text{EQ}(23)\) is examined with \(T\) being large and \(a(i,k)<0\), it can be seen that this occurrence is possible. In a practical sense, this means that the model has been formulated incorrectly and when this situation occurs, the simplex algorithm is terminated.

The discussion up to this point has assumed an initial basis was present or the problem was stated as \(AX<B\) where slack variables will form the basis. The initial problem may be stated as \(AX=B\) with an initial basis not readily identifiable. A technique which is used in this case is called an "artificial basis" technique. Each constraint has
a unique basis variable added and also each variable is assigned an unspecified large negative number (often called "M") as a cost coefficient. It has been proven that if a feasible solution exists to the original problem, one will also exist for the augmented problem. Also if a feasible solution does not exist for the original problem, the optimal solution to the augmented problem will contain an artificial variable at a positive level (Ref 10:81).

This technique, although powerful, may increase the number of iterations required to obtain optimality. The artificial variables must be driven from the basis prior to determining optimality and therefore should be used only when required. If a basis vector exists in the problem as given, use this as an initial basis vector to minimize the iterations required.

The simplex algorithm is applied as discussed previously to the artificial basis with the exception of selecting the entering variable. Since the \([z(j)-c(j)]\) values may now contain a unspecified large value "M", the selection of the largest negative \([z(j)-c(j)]\) must consider two elements. All \([z(j)-c(j)]\) values which contain a "M" must be examined and the selection differentiator is the numeric element. That \([z(j)-c(j)]\) with an "M" value and the largest negative \([z(j)-c(j)]\) should be selected as the entering basic variable. Once the "M" values has been removed from the objective row of the tableau, the algorithm
proceeds as before.

Sensitivity analysis is based on the relationship between the coefficients of the original columns of the linear programming model and the columns representing the slack and artificial variables.

Considering a problem with \(K\) constraints and \(V\) variables, the final \(K\) columns will initially form an identity matrix. As the simplex algorithm is performed, the identity matrix undergoes a transformation. This matrix is, upon completion of the algorithm, a record of all operations performed on the original equation. It is possible to determine from this record the change to any element of the final tableau which results from one or more changes to the original tableau.

To illustrate these changes, consider an original problem with three variables and four constraints. If a change was made to the original \((3,2)\) element, the changes to the final tableau could be found by pre-multiplying the matrix with the single change by the transformed identity matrix which is commonly called B-inverse.

As can be seen in Figure 2, a change in the third row, second column will produce changes in the entire second column of the final tableau. Also, note that only the values in the third column of B-inverse were pertinent. This is because the third column of the identity matrix or B-inverse is associated with the third row of the original
Tableau. Although not shown in the illustration, the coefficient of the objective function above the third column of the B-inverse matrix, when multiplied by the change to the (3,2) element, will give the change to the coefficient in the objective function above the second column of the original matrix.

When the results of the changes have been determined and added to the respective elements of the final tableau, further manipulations may be necessary. If a change occurred to a column which was in the basis of the final tableau, that column will have to be returned to its final tableau form. This is done by: first, dividing the entire row which had a value of one in the column under investigation by the new value to reestablish the value of one, and second, adding multiples of that row to each of the other rows and the objective function to return all other values in the column to zero. If the value of any
right-hand side or objective function coefficient has become negative, the tableau must be resolved using the LP algorithm.

Determining the range limits of the elements of the original tableau is merely a variation of the method previously described. Rather than solving for the changes which result from a change to an original element, the algorithm individually investigates each element to determine the smallest positive and negative changes which would cause either a multiple optimal (an objective function coefficient not in the basis goes to zero) or a degenerate (right-hand side goes to zero) condition.

$$
\begin{array}{cc}
\Delta C & C \\
X_2 & 0 \Delta W & 0 & 0 & 0 & 0 & W & 0 \\
X_1 & 0 \Delta X & 0 & 0 & 0 & X & 0 \\
X_5 & 0 \Delta Y & 0 & 0 & 0 & Y & 0 \\
X_7 & 0 \Delta Z & 0 & 0 & 0 & Z & 0 \\
\end{array}
$$

← B-inverse →

Figure 3. Column, Constraint Relationship

As shown in Figure 3, a change in the (3,2) element may cause changes to all elements in the second column. This column must now be returned to the final tableau form (assume a 1 was in element (1,2) and the other elements were zero). The first constraint would now be divided by $1 + \Delta W$.
and multiples of the first row would be added to the objective function and to the other rows to return their values in the second column to zero. To be specific, \(-\Delta X\) times the first row would be added to row two, \(-\Delta C\) times the first row would be added to the objective function, etc. Each of these cases can be set up as an equation to determine what value of \(\Delta\) will cause a zero value to be reached in a non-basic column of the objective function or in the right-hand side. Each row and each non-basic column (except artificial variables which are excluded) will produce a \(\Delta\). From these \(\Delta\)'s, the smallest positive and the smallest (absolute) negative represent the bounds on the change to the element.

The specific equation used to determine the maximum positive change is:

\[
\Delta = \min \left[ \min \text{POS} \left( \frac{-C(L)}{(A(ROW,BCOL)SC(L)-A(ROW,L)S C(BCOL))} \right), \min \text{POS} \left( \frac{-B(M)}{(B(M)A(ROW,BCOL)-B(ROW))} \right) \right] A(M,BCOL)) \]

(28)

where

- \(C\) = objective function coefficient in the final tableau
- \(L\) = 1, 2, ..., total variables---excluding artificials and the column with the delta
- \(ROW\) = the row which represents the basis of the column under investigation (has a value of 1 in the final tableau)
\[ A \] = the element coefficient in the final tableau

\[ \text{BCOL} \] = the column in \( B \)-inverse associated with the change being investigated (The third column of \( B \)-inverse if the change was to the \((3,2)\) element.)

\[ B \] = right-hand-side value in the final tableau

\[ M = 1, 2, \ldots K \] — all rows except the row in the basis for the column being investigated

The negative delta is similar except the largest negative (smallest absolute) values are determined.

To illustrate:

\[
\begin{array}{cccc}
\Delta & C & C3 & C \\
1 & X2 & 0 & 1+\Delta W \\
2 & X1 & 1 & \Delta X \\
3 & X5 & 0 & \Delta Y \\
4 & X7 & 0 & \Delta Z \\
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]

Figure 4. Change to the Objective Function Coefficient

For \( L = 3 \) and \( X2 \) is in the basis in ROW 1

\[ \Delta(3,2) = -C(6)/C(6)*A(1,3)-C(3)*A(1,6) \]  \hspace{1cm} (29)

\[ \Delta(3,2) = -C/(C8D-C3&W) \]  \hspace{1cm} (30)

The value of \( \Delta(3,2) \) found in equation 30 will cause the objective function coefficient \( C3 \) (Figure 4) to be driven to zero. All other columns which are not in the basis must also be checked to find the minimum changes. Basic columns do not need to be checked since they have a zero in the
critical element. Therefore, no multiple of the row could change any other row. If the element which is being investigated for range limits is not in a basic column, the only value to be determined is the relationship between the objective function coefficients of the desired column and of the column associated with the constraint. The maximum range limit is determined by \( \Delta = -\frac{C(L)}{C(B) \cdot C(L)} \).

The changes for the right-hand side are checked in a similar manner.

\[
\begin{array}{cccccccc}
\Delta C & C \\
X_2 & 0 & 1+\Delta W & A_{1,3} & A_{1,4} & 0 & W & 0 & B_1 \\
X_1 & 1 & \Delta X & A_{2,3} & A_{2,4} & 0 & X & 0 & B_2 \\
X_5 & 0 & \Delta Y & A_{3,3} & A_{3,4} & 1 & Y & 0 & B_3 \\
X_7 & 0 & \Delta Z & A_{4,3} & A_{4,4} & 0 & Z & 1 & B_4
\end{array}
\]

Figure 5. Right-Hand-Side Ranging

ROW = 1, \( M = 2 \)

\[
\Delta = -\frac{B(2)}{B(2) \cdot A(1,6) - B(1) \cdot A(2,6)} \quad (31)
\]

\[
\Delta = -\frac{B(2)}{B(2) \cdot W - B(1) \cdot X} \quad (32)
\]

Again, each row (\( M=2,3,4 \)) will produce a delta. All of the deltas (positive and negative) are searched to find the delta which first drives an objective function coefficient.
or a right-hand side to zero. This will be the range for the element under consideration.

Sensitivity analysis is subject to ill-conditioning which would not be present if the modified problem was solved from the initial tableau. This condition occurs when one of the two constraints (in a two-dimensional problem) which forms the corner-point solution of a final tableau is modified to cause the two constraints to be parallel. The current corner point is now at infinity. This condition would never occur using the full algorithm since one of these constraints would be outside the convex boundary and, therefore, not involved in any basic solution.

Figure 6 shows a two-dimensional problem. The optimal solution is shown in Table V. A critical value to cause
TABLE V
A Final Tableau

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOJ FUNCTION</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.2000</td>
</tr>
<tr>
<td>CN NAME VAR</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-2.0000</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

RHS

<table>
<thead>
<tr>
<th></th>
<th>CN NAME VAR</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 = 20.0000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4 = 25.0000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2 = 30.0000</td>
<td></td>
</tr>
</tbody>
</table>

ill-conditioning is a change in the (1,1) element of -2.5. This change is shown in the revised problem in Figure 7. As can be seen, two constraints are now parallel, and one of them is not within the convex boundary. This new problem can be easily solved by the full algorithm (Table VI) but requires division by zero when sensitivity analysis is used to find a new solution from the original optimal solution (Table VII).

Ill-conditioned points exist in every tableau. A critical value may exist for each element in an original column which has its variable in the basic solution. Empirical results indicate that these critical values are usually outside of the range limits for individual elements. The only known exception is the unique case where the change makes the
Figure 7. Corner Point at Infinity

TABLE VI

Modified Problem, Final Tableau

<table>
<thead>
<tr>
<th>OBJ FUNCTION</th>
<th>1(1)</th>
<th>1(2)</th>
<th>3(3)</th>
<th>4(4)</th>
<th>5(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>.000000</td>
<td>.000000</td>
<td>.000000</td>
<td>.000000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1.000000</td>
<td>.000000</td>
<td>.000000</td>
<td>1.000000</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>.000000</td>
<td>1.000000</td>
<td>.000000</td>
<td>.000000</td>
</tr>
</tbody>
</table>

RHS

<table>
<thead>
<tr>
<th>OBJ FUNCTION</th>
<th>26.000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 = 50.000000</td>
</tr>
<tr>
<td>2</td>
<td>1 = 40.000000</td>
</tr>
<tr>
<td>3</td>
<td>2 = 30.000000</td>
</tr>
</tbody>
</table>
TABLE VII
Division By Zero

<table>
<thead>
<tr>
<th>OBJ FUNCTION</th>
<th>( (1) )</th>
<th>( (2) )</th>
<th>( (3) )</th>
<th>( (4) )</th>
<th>( (5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN NAME VAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 - 1</td>
<td>0.0</td>
<td>-0.4</td>
<td>0.0</td>
<td>-2.0</td>
</tr>
<tr>
<td>2</td>
<td>6 + 1</td>
<td>0.0</td>
<td>-0.4</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>0 + 0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

constraint parallel and identical. This new problem would be ill-conditioned for all solutions since the constraints would no longer be independent.

Ill-conditioning in sensitivity analysis is an interesting phenomena, but its effects are not major. The ill-conditioning can be avoided if the changes are varied a small amount. The full simplex algorithm can be used if the exact changes must be investigated.

The purpose of this chapter was to provide a brief theoretical review of linear programming in conjunction with the simplex algorithm methodology and postoptimal sensitivity analysis as they were applied in this software package. Those interested in a more thorough discussion of the simplex algorithm are directed to Gass (Ref 10) and also Garfinkel & Nemhauser (Ref 9). The area of postoptimal
sensitivity analysis is covered in depth by Gal (Ref 8) with again a practical view given by Levin & Kirkpatrick (Ref 16). Many other important aspects of linear programming, such as duality theory, have not been presented but are present in numerous references. A quite thorough and practical presentation of duality theory is given by Levin & Kirkpatrick (Ref 16) while a brief, but comprehensive theoretical view is given by Garfinkel & Nemhauser (Ref 9).

Although many areas were not discussed, hopefully the previous discussion has implanted or reinforced the mathematical background of the simplex algorithm. Also, the simplex algorithm has been intended to be shown not as an abstract technique which is blindly applied, but a valuable tool to assist analysts and managers in solving real life problems.
III Design Considerations

The developmental process involved in an effort to produce a well-designed product requires careful consideration of all aspects which may influence the outcome. The objective of this section is to present the major considerations of the software design phase. Chapter IV will then discuss the method of implementation which has evolved from this analysis.

User Considerations

The user must feel that the software is beneficial in terms of the time and effort required to use it in problem solution and analysis. Therefore, several important factors must be considered with the user in mind. One such consideration is that the program should be developed in a logical sequence from model formulation to problem solution to analysis of results. The user should be carefully guided through this sequence, being allowed to correct either incorrect entries or incorrect problem formulation without resorting to complete model reformulation and input. Furthermore, the input of an option selection or the input of a model should occur in a sequence which coincides with the logical progression of problem solution in order to lessen the anticipation and doubts of infrequent users.

Another important area is the ability of the user to
quickly and accurately locate the results of each step in the sequence. The prompts to the user for data or response input should provide meaningful, concise guidance to the user. In some instances, graphically supplemented output may be desirable. This may occur when one value being studied is valid for a range of values for another variable. In this manner, the user is able to visually determine the range, and to a degree, the sensitivity of one variables relationship to another.

Although linear programming and the simplex algorithm are capable of producing the desired solution, alternative methods of applying the simplex algorithm are available which may allow more efficient and timely solutions to an LP model. It may be desirable to provide the user with the capability and option to solve the problem by one of these alternatives. Again, the presentation of these options should be performed as clearly and concisely as possible.

Further software enhancements which may be user desirable include minimal programming and operating system interface. The user should not be required to alter program source code to use the software; however, the programs should be designed and presented in a manner which will not preclude future enhancements or modifications. These considerations suggest that the source code be modularly designed, developed, and documented to reduce the effort required to locate the code of a specific function and then
interpret the source code.

One additional area of concern is program control. Since the user may be inexperienced in either programming or use of the operating system, the program should require minimal and infrequent guidance. This consideration leads to a menu-driven program which displays available program options enroute to problem solution and analysis. Upon input of a desired option by the user, the program ideally will perform all interface with the operating system to pass control to the desired program, unit, or subroutine. Compromises to this ideal environment may be necessary. If so, precise instructions to the user on the required operating interface commands needed to progress through the desired sequence should be appropriately displayed to minimize the required familiarity with the operating system.

**Hardware Considerations**

Microcomputers offer many advantages not available with large, stationary computer systems. These advantages include a substantial decrease in acquisition cost plus virtual elimination of support requirements. Also, the transportability of the "desk top" computers is ever increasing due to recent advances and design considerations. In conjunction with these advantages, one would expect and soon finds areas of diminished capabilities compared to a mainframe system. Two primary areas are the decreased memory capacities and reduced computational rates of the
microcomputers. This exchange of decreased cost and increased transportability for decreased capabilities does impair the size and speed of developed software; therefore, one must consider possible avenues which will counter these decreased capabilities.

In determining the target system for software development, one should consider the availability of the various microcomputers in conjunction with the capabilities of each. Software which is developed on a system with limited availability to the target users will not be used extensively. Even if a readily available system is used for software development, it is necessary to consider the modifications and peripherals of the target system. If these modifications or peripherals are unique to the development system, it limits the use of the software.

Although the available microcomputers vary extensively in their memory capacities and peripherals, a range from 48K to 128K bytes random access memory (RAM) is not unusual (48K is approximately equal to 48,000 characters). Peripherals, such as printers, communication links (modems), and disk drives are quite common, if not necessary, among those who use microcomputers. The dependency of the developed software on these peripherals must be strictly specified, or options must be provided which will allow the user to designate only those peripherals possessed. In this way, many potential users could gain access to the software.
without an added hardware requirement.

Language Considerations

A primary factor in the selection of a language for this research was the portability of the language from one microcomputer system to another. BASIC was once the only high-level language generally available for use with microcomputers. Presently, microcomputer users have access to other high level languages such as Pascal, FORTRAN, and APL. The three languages felt to be most accessible to users were BASIC, Pascal, and FORTRAN and were considered for implementation in this research.

BASIC. As previously noted, that the portability of the language is a primary consideration. From this viewpoint, BASIC does not gain much support since each microcomputer system has modified the BASIC language in order to coincide with the needs of that particular system. This has caused many versions of the BASIC language and results in extremely limited portability.

Execution time is also dependent on the language selection. A program written in BASIC must be interpreted line by line to machine language each time the program is executed. This process causes execution time to be considerably slower than for other languages which are first compiled and then executed. BASIC programs may be compiled; however, the compilers are machine dependent due to language and hardware differences between systems. This compiled
code is then unique to the system and further deters portability.

**Pascal.** Pascal, the most recently developed language of those considered, emerged in the early 1970's and has since proven to be a powerful, high-level language. A significant factor in the growth of this language was the work of Kenneth Bowles who directed the development of UCSD Pascal at the University of California at San Diego (Ref 14:118). UCSD Pascal, originally formulated as a teaching tool, has allowed for larger programs to be implemented on microcomputers and has lead to an increased portability of high-level languages between systems.

The enhancements to Pascal and FORTRAN are the result of the "P-code" and "P-machine" of the UCSD Pascal system. The source code, either Pascal or FORTRAN, is first compiled into P-code and stored as the object code for the P-machine. The P-machine, which interprets the P-code upon execution, emulates instructions to the machine-specific central processing unit. Only the emulator of the UCSD Pascal system must be revised to coincide with each system to allow portability, and this may be performed in a relatively short time. This one-time revision then allows compiled Pascal or FORTRAN programs to be portable between microcomputers. Another advantage of the P-code is that it requires considerably less memory than the equivalent object code or machine language. This allows larger programs to be
resident in memory at any one time. Also, UCSD Pascal has implemented an overlay capability which loads P-code into memory as needed and discards this code upon completion. This capability allows the user to specify, to a great extent, the amount of code in memory during the various phases of program execution (Ref 14:114). The advantages of the UCSD Pascal system are applicable to both Pascal and FORTRAN programs; therefore, little preference is gained for either high-level language.

A disadvantage of Pascal is that it does not possess extensive output formatting capabilities which are often found in other languages. Although not an insurmountable problem, it is a factor when carefully formatted output is desired.

FORTRAN. FORTRAN, the most widely used language within the scientific programming community, was developed in the early 1950's by IBM (Ref 3:1). In the succeeding years, the FORTRAN variations have increased, leading to a need for standardization of such an intensely-used language. Two attempts have been made, with the most recent (1970-1977) specifying a full language and subset language. Attempts to standardize the language have been of assistance to programmers, however, discrepancies between implementations still exist and are a major downfall of the language.

As previously discussed, the portability of FORTRAN has been greatly aided by the development of the UCSD Pascal
operating system. Also, due to FORTRAN's evolution process, formatting features are available which are not present in BASIC and Pascal. A consideration which has not been mentioned is that although increased formatting capabilities do exist, output time is increased when this option is utilized. Dependent upon the type and amount of output desired, this may become a significant factor in program execution time.

The above sections have discussed those features which are desired in the software and also the availability of hardware and languages with respect to microcomputers. The next chapter will discuss the selection of a particular microcomputer system and language. Also discussed are the methods in which those desired features were incorporated into the software within the constraints of the hardware and language selections.
The objective of developing an extensive microcomputer-supported linear programming software package combined with the design considerations discussed in the previous chapter have formed the foundation for this research. Each area discussed in Chapter III was felt to be of significance; however, each must be reviewed in light of the capabilities, advantages, and disadvantages of microcomputers as well as the desires and abilities of the intended users. Due to the requirements placed upon the design by each of these factors, conflicts may arise which will allow less than full incorporation of one or more of those desired areas.

The purpose of this chapter is to present the method of implementation used in the software development of this research. First, the method of incorporation of those significant areas discussed in Chapter III will be discussed as they apply to the software package in general. Next, the method and underlying logic of implementation will be reviewed as it applies to each module of the software package. Then, specific problems or special areas of consideration will be discussed. Chapter V will present findings and recommendations which are felt to be noteworthy to future efforts in similar research.
Hardware

One of the first major decisions of the implementation phase concerned the selection of the microcomputer system to be used in the development. It was felt that the system should be a commonly available system without extensive or unique modifications or peripherals. Also, the memory capacity of the chosen system should be comparable to those most likely accessible by the intended users.

In light of the above considerations in conjunction with the availability of such a system to the developers, an Apple II-plus microcomputer was selected as the developmental system. The Apple II-plus, although not the most advanced microcomputer on the market, has become one of the largest selling systems and is presently selling approximately 20,000 microcomputers per month (Ref 18:19). Other favorable features of the Apple II include high-resolution graphics, memory expansion capabilities, readily available peripherals, and the ability to support the Pascal and FORTRAN languages.

The system which was used for the implementation of this software package is shown in Figure 2. The system, as shown, was felt to be representative in capabilities of those microcomputers available to the intended users.
1. Apple II-plus microcomputer with 48K RAM
2. Two disk drives (5-1/4 inch)
3. 16K memory expansion card (language card)
4. Printer
5. Video display (monitor or TV)

Figure 8. Developmental Hardware Configuration

Language

The three languages which were examined for implementation in this research were BASIC, Pascal, and FORTRAN. FORTRAN, even though it does have limitations and drawbacks, was felt to be the most appropriate language as supported by the Apple II microcomputer.

The BASIC language, either Applesoft or its machine specific counterparts, was found not to be portable to any extent, thereby violating one of the primary language selection criteria. Furthermore, as an interpreted language, the execution times were known to be in excess of compiled languages such as Pascal and FORTRAN.

The selection between Pascal and FORTRAN was much more subjective than the elimination of the BASIC language. In the past, Pascal had offered a greater degree of portability. However, the recent development and use of the UCSD Pascal Software System (Ref 14) has allowed increased portability of both FORTRAN and Pascal. This system allows
both Pascal and FORTRAN to run on most microcomputers thereby virtually eliminating portability problems due to hardware configurations.

A frequently discussed obstacle to FORTRAN is its lack of standardization among the various implementations. Although the magnitude of this problem has been reduced, it still exists for large computer systems as well as microcomputers. It was found that even though a standard Pascal language exists, the microcomputer implementations of this language often do not coincide fully with the standard, allowing for portability problems similar to FORTRAN.

Although Pascal has come to be known as a powerful high-level language, it is relatively new and not as intensely used by the scientific community. It was felt, even at the possible expense of perpetuating an outdated language, that FORTRAN would be an acceptable language by the scientific community due to its familiarity. Also, the well-developed intrinsic output formatting capabilities of FORTRAN, which were extensively used in the research, would allow for a more rapid implementation than those available with Pascal.

User Interface

The requirement for user interaction while employing a microcomputer is inevitable; however, the degree of such interaction is largely programmer controllable. The previous selection of hardware and language has also
delineated the magnitude and frequency of user interaction to some extent, dependent upon program size and complexity. It has been noted that the targeted users should not be required to possess extensive programming or operating system knowledge. This further requires that any interface be preceded by well structured, concise guidance to the inexperienced user. This user interface requires user input of responses or data which may be entered incorrectly causing an execution error. To preclude possible operating system errors due to erroneous input, a system of screening user input has been constructed to prevent such losses. This section describes the manner in which the interface system has been designed within the confines of the hardware and language parameters as well as the design considerations outlined in Chapter III.

Apple FORTRAN, the FORTRAN version implemented on the Apple II, allows the use of a "turnkey" system which automatically begins running a programmer-designated program following prescribed startup procedures (Ref 1:9) (See Appendix A for startup procedures). A turnkey system requires minimal operating system interface with the user for a program to initially gain control and begin to guide the user. Such a system has been included to lessen the interface initially required for software use. Therefore, upon startup, the first program of the package, Module 1, is automatically executed.
The package consists of four separate programs due to the physical size of the LP package and memory capabilities of the selected hardware system. This factor, coupled with the inability to "chain" programs in Apple FORTRAN (in chaining, one program may cause the execution of another program without user intervention), has forced the design to include, to a degree, user interface with the operating system. This interface, which occurs when the user requires a different program to continue the solution process, has been designed such that the user will select from a menu the course of action desired. The program which is currently in control will then terminate and display specific instructions to the user. These instructions will enable the user to execute the desired program of the package.

The previous discussion noted that the user would be presented a list of alternatives. The user would then select one, after which specific operating systems commands would be presented. This method of menu display and user selection input may also be applied within a specific program to cause execution of a particular subroutine or a sequence of code. This method, often designated as "menu-driven", requires minimal user input to cause the desired actions. This method has been implemented, where appropriate, in this software package. User inputs are normally limited to either numeric input (Options 1, 2, or 3), character input (P for printer, S for screen), or Yes/No
inputs (Y for YES, N for NO). User inputs are screened to prevent undesired program termination.

The high-level languages such as FORTRAN do not support intrinsic input error checks which prevent premature program termination and data loss when user inputs are of an improper type or range. Therefore, to assist in prevention of such an event, all user inputs except problem, constraint, and variable names are first screened to insure that they will be acceptable to the program. All user inputs are first placed in character strings which allow any type of numeric or character input. Next, depending on whether the input is a character or a numeric representation, these inputs are inspected character by character.

Character inputs, such as an option selection involving options P, S, or B or a Yes/No response, are checked to insure that one of the possible responses for that particular input has been entered. If so, the option represented by the user input is performed. However, if the input does not coincide with the possible alternatives, a message indicating an invalid input is displayed and the user is directed to reenter the input. This process is repeated indefinitely for any required user input.

Numeric inputs have been separated into real and integer numbers. Inputs which require integers are inspected character by character to insure that all
individual entries are numerics or blanks. If so, the character string representation is converted to its equivalent numeric representation. At this point, the numeric value is checked to ensure that it is within the allowable range for the specific response. If a non-numeric is found or the range is violated, the user is informed of an invalid input and directed to reenter the number. Real numbers are treated similarly to integers except that a decimal point has been included in the set of valid entries. Also, range screening is not performed on real input. It should be noted that real numbers may be input as integers (i.e. if 1 is entered, it will be internally represented by 1.0) and that commas (i.e. 10,000) will not be accepted in real number inputs. Also, a value of zero may be entered by simply pressing RETURN without numeric input.

These steps allow the user to enter an invalid response and recover without data loss; however, an undesired input which is valid to the program will be accepted and the user will be forced to correct this erroneous input. In the case of an erroneous option selection, the user will be forced to complete the process under the option entered.

Software Description

This linear programming software package consists of four distinct FORTRAN programs. There were two primary reasons for the separate programs. First, the memory capacity of the microcomputer would not allow the software
to be developed in fewer than the four which were used. Second, if the software could have been implemented in one program, the compiled code would have been too large to be stored on one disk. Therefore, these programs (annotated as Module 1 through Module 4) have been designed and implemented in a sequence which coincides with the LP model formulation, solution, and analysis sequence used in most analyses.

The four modules serve separate purposes, but are related. Module 1 must be used for problem entry before Modules 2 or 3 may be used. Module 2 (the instructional module) or Module 3 (the problem solver module) must be used to generate a final solution before sensitivity analysis (Module 4) can be performed. The following sections will discuss the purpose of each module and their relationship to each of the other modules of the package. Problems which were encountered and the methods of correction will be discussed.

**Module 1 - Data Base Entry Module.** This module, which is the program automatically executed upon startup through the use of the turnkey system, serves two primary purposes. First, it provides an initial entry point into the software package where the program may begin to guide the user through the sequence of model entry, solution, and sensitivity analysis. This module's second primary function is data base entry. The user is guided through the steps of
selecting the type of model to be entered as well as entering of the various parameters of the LP model desired to be studied.

Module 1, when executed, presents the user with several options related to the possible alternatives available in the complete LP package. The user may elect to enter a model data base; in which case, Module 1 continues to guide the user. If the user elects to solve an LP problem using either Module 2 or 3, Module 1 provides the necessary guidance. The last choice is sensitivity analysis. If this option is selected, Module 1 informs the user of the required data bases and the required steps to implement Module 4.

The above selection allows Module 1 to provide specific instructions to the user on the operating system commands required to enter the desired module. As noted, this is one of the primary functions of this module. It should also be pointed out that, upon the completion of any of the other modules, the user is given instructions to execute Module 1 which will then again present the various options. The user, at that point, may select the desired step of the sequence and continue the analysis.

The second purpose of this module is model entry. If the user elects to enter a data base, Module 1 continues to guide the user through the required steps. This portion of the module consists of three primary sections: data base
Data Base Entry: This section allows the user to enter a model with or without names associated with the various parameters of the model. Also, a model which has previously been stored on a disk may be retrieved for inspection or editing. If the user elects to enter a model, a series of prompts are presented, which requires either an option selection or parameter input. Prior to the entry of the model parameters, instructions are displayed which inform the user of the method and order to be followed in parameter input. During this series of prompts, the user has access to extensive editing functions which allow correction of any previously entered option selection or parameter input.

These editing features, in conjunction with the screening of user inputs allow the inexperienced user to correct invalid responses as well as incorrect input (meaning valid to the program but not the user intended input) without having to resort to complete model reentry to correct either type of error. The editing features, available during data base entry and data base management, constitute the majority of this module in the terms of FORTRAN code; however, this feature was felt to be important in terms of usability of this software package. The number and degree of editing features to be employed was subjective in nature; however, previous work by Conte (Ref 6) aided greatly in the selection of those which would be most
beneficial to the user.

Data Base Management: Upon completion of the initial data base entry or retrieval of a model from a disk, the user is presented several new options. The user may display the model to insure that all parameters have been entered as desired. Also, this option allows the output device to be either a monitor or printer. With this option, those users who possess printers may then receive a hard copy of the input, while those who do not have printing capabilities still have the ability to review input. Should the user find an error in the model, access is provided to the editing features of this model and changes may then be performed. The user may elect to save the model to a disk under a user-specified filename. It should be noted that each model must be saved to disk following data entry and editing to allow further study. This requirement arose from the inability of the package to be implemented as one program. To prevent the user from inadvertently leaving this module without first saving the model to disk, a prompt appears which warns the user that the current model will be lost if not saved.

The ability to read a model from a disk, combined with the editing features, allows a user to use one model as a starting point, perform changes and then solve to determine the effects of those changes. The above procedure would not normally be required due to the extensive sensitivity
analysis features of Module 4; however, the user may wish to see the complete solution process with certain parameter variations applied.

Execution Management: Once the initial model has been saved to disk, the user may elect to enter another model with the data base entry section; however, one would normally elect to solve the model at this point. A menu of options allowing Modules 2, 3 or 4 to be selected is displayed when the user elects to solve the problem. An option of returning to the data base management menu is also provided to allow recovery from an incorrect or undesired input.

The user selects the desired module and is then guided through the required entries. A problem which was encountered during the transition phase was the transmittal of the diskname:filename. To solve to this problem, two data files were created, LP1:LPDATA and LP2:LPDATAW with these files being placed on the disks LP1 and LP2, respectively. These files contain the diskname:filename of the model currently being studied. LP1:LPDATA contains the name of the most recently input or edited model while LP2:LPDATAW contains the diskname:filename of the file which contains the results of the most recently solved problem by Module 2 or 3. LP1:LPDATA is automatically read by Modules 2 and 3 while LP2:LPDATAW is read by Module 4 to determine the file which contains the inputs for that module.
One other area which required careful planning was the arrangement of the various editing subroutines in the overlaid compilation units. Due to the extensive interaction of the various editing functions, attention was required to avoid several overlaid units being resident in memory simultaneously thereby overloading the capacity of the microcomputer. This problem was eliminated by cross-referencing the subroutine interaction and then forming the compilation units so that an overloading of memory would not occur.

Module 1 does not perform or enhance the LP model solution process but provides valuable organization support for the overall package. Organizing the package so that one program provides this support allows more memory to be available for the individual functions of each of the remaining modules whose specific functions will be discussed.

Module 2 - LP Instructional Module. The goal during the development of Module 2 was two-fold. First, the module would allow the user to select a previously entered LP model and apply the simplex algorithm to that model to determine an optimal solution. Second, it was envisioned that this module would be designed so that the user could be assisted in learning the application of the simplex algorithm to an LP model. Although the primary concern of the instructional portion was to assist students in their initial contact with
the simplex algorithm, these features also allow instructors to demonstrate to students the outcome of procedures or selections not complying with the simplex algorithm.

The initial guidance in reference to notation and the sequence of steps representative of simplex algorithm computer implementations was drawn from Gillett (Ref 11:101-105). Although the FORTRAN code presented in Gillett was of assistance in forming the code of this module, extensive modifications and extensions were necessary in order to implement the desired instructional features.

This module and Module 3 are quite similar in that they both allow the solution of an LP model by the simplex algorithm. Due to the desire to provide extensive feedback to the user during the instructional process, this module was much larger than that of Module 3. The size of this module's FORTRAN code required, as did Module 1's, careful consideration of memory capacities in the forming of the compilation units. As is discussed in Appendix B, the manner in which the compilation units are formed may dictate the size of the overall program.

Several steps are required to insure the LP model is in the proper form to apply the simplex algorithm. The standard LP form upon which this and the following modules have been based has been illustrated in Chapter II. The process of modifying the LP model to coincide with the stated form has been labeled as "tableau formulation" and
involves the manipulation of the objective function and resource constraints. Once the tableau formulation is completed, the iterative process of the pivot element selection and moving to an adjacent feasible solution is performed. After moving to each new feasible solution, one must check to ensure that an optimal solution has not been reached prior to continuing the iterative process. Therefore, the application of the simplex algorithm has the following three major steps:

1. Tableau formulation
2. Pivot element selection and determination of new basic feasible solution
3. Check for optimality, unboundedness, etc.

The instructional areas of this module have been divided into three areas coinciding with the above steps. In order to allow the user to concentrate on the specific step or steps desired, options have been provided which allow either: 1.) the user to direct the manner each step is performed or 2.) the program to perform these actions without user interaction.

The three areas in which instructional assistance has been provided have been implemented differently. During tableau formulation, several options which modify the objective function or constraints are presented to the user. The user selects that option which is applicable to the objective function or to the constraint under consideration. At that time, the user is provided feedback as to whether or
not the selection was correct; and, if not, the correct selection is displayed. Although the user's selections may be incorrect, the objective function constraint is modified properly.

The pivot element selection process has been implemented differently than the process described above. The user is presented two methods to select the pivot element: with or without feedback. If the user elects to receive feedback, the user is allowed to correct improper pivot element selections to coincide with the algorithm selection which is also displayed. The user may also elect not to receive feedback on the validity of the selection, and the user-selected pivot element is used for further computations. Under both methods, the user could request that the algorithm divide by zero and cause an execution error. In order to allow the maximum possible freedom of pivot element selection for demonstration purposes yet prevent inadvertent execution errors, a system of checks has been implemented which warns the user that an execution error may occur if the pivot is performed as selected. The user is then allowed to either continue with the previous selection or select a new pivot element.

The step in which the optimality, unboundedness, and infeasibility of the current solution are checked has been implemented similarly to that of tableau formulation. Regardless of user input, the cycle of steps 2 and 3 will
continue until true optimality, unboundedness, or infeasibility exist. During each cycle of the steps, the user must enter responses to questions pertaining to the status of the last tableau and receives feedback on these responses.

Module 2 has implemented, at the expense of less descriptive feedback on user selections, a capability to perform dual pivots. This capability allows the user to become familiar with the dual simplex algorithm while also allowing a more efficient solution technique to be used in certain situations. Again, a system of checks has been incorporated which prevents the user from improperly applying the dual (or regular) pivot method.

Due to memory size limitations, the expansion of any one portion of this module required reduction in another. Considerable effort was required to allow both dual pivots and descriptive and informative feedback to the user. In order to allow the most detailed comments possible, standard formats for comments were used with variables representing the areas applicable to the situation. The comments were more descriptive than would have been possible if separately implemented comments were used for each situation.

Several features have been incorporated into this module and Module 3 which increase the ease of use and decrease the time and peripherals required. First, the user may elect that output be displayed in either scientific
notation (e.g. 12.01E+02) or in the more common representation (e.g. 1201). Although the output format has no influence on the computations performed, the numbers may become too large or too small to be accurately represented in the space allocated in the output without the use of scientific notation. Any numbers larger than 999,999 or less than 0.00001 will be represented by asterisks or 0.00000, respectively, if the scientific notation is not requested.

A second feature allows the user to have the tabular output displayed on the screen or on a printer. This option allows users without printing capability to use the software and allows those with printers to receive hard-copy output for later use and study. The format, which is identical for both output devices, has been constructed in a manner to facilitate viewing on a screen. The tableaus are displayed in sections from left to right with a user-controlled pause between screen displays. This allows the user to study information from that portion of the tableau prior to the next screen being displayed.

The last feature is primarily applicable to those who wish to use this module as a problem solver without interaction. This option allows the user to designate the specific tableaus to be displayed. This option, if minimal output is required, allows a solution to be obtained more rapidly since less output time is required for the
intermediate tableaus to be displayed. The user should be cautioned against changing the programmer-defined defaults for tableau output when using this module for instructional purposes. Should these defaults be changed, the user will not be able to determine the tabular outcome of the previous iteration and, therefore, may not be able to select the proper pivot elements for future iterations.

When an optimal solution has been obtained, the user may save the solution on disk to allow for future sensitivity analysis using Module 4. It should be noted that once a problem has been solved using this module, there is no reason to use Module 3 for the same problem. Module 3 performs the same computations as Module 2 with minimal user interaction.

**Module 3 - Problem Solver Module.** Module 3 is similar in purpose to Module 2. The major distinction is that instructional comments have been deleted allowing the addition of features not possible in Module 2 due to memory size limitations. The primary feature available in Module 3 not previously available is the ability to explicitly solve the dual LP model of an LP problem entered in Module 1. Although user interaction is not required in the transformation process from the primal to the dual problem, the program does inform the user when variables are added to allow for unconstrained variables. One may wish to review duality theory to fully understand this requirement.
Hillier & Lieberman (Ref 13:91-109) offers a basic review of this area. The message which advises the user of added variables denotes the subscript of the original variable and the corresponding added variable so that later tableaus may be correctly interpreted.

This module, as does Module 2, displays the programmer-defined default options upon execution of the module. The options displayed include the type of problem to solve (primal or dual), whether or not to use dual pivots, the output format and destination, and the specific tableaus to be displayed. The selection of these default options was based on the subjective judgement of the frequency of use of the various options. Also, the default options were defined to be consistent between Modules 2 and 3. In this way, a user who has become familiar with the operation of one module may transition smoothly to use of its counterpart.

A problem not encountered in Module 2 implementation arose during the incorporation of the dual problem solving capability. Module 4, the sensitivity analysis module, requires access to the parameters of the problem as entered or, for dual problem solution, as it would have been entered as a primal problem. Since Module 3 performs the transformation of the primal to the dual, the parameters of the dual problem have not previously been saved to disk. This requires that a new data file be created to contain
these parameters. Initially, it was envisioned that the original data file containing the input to Module 1 would be rewritten to contain the dual problem parameters. Two problems were encountered. First, the original data file would be lost, and the user could no longer use this file to edit the model for further problem formulation. Second, and most importantly, due to the possible addition of variables, the new file possibly could not be stored on disk in the space allocated to the original data file. This problem would cause an input/output error and premature program termination. Both of the above problems were overcome by the creation of a new file with a user-specified diskname:filename.

The primary areas noted above were those which differed from that of Module 2 or in which special problems arose. The user may save the results of problem solution to disk for later sensitivity analysis with Module 4. The user is then allowed to retrieve another LP model from disk for solution or to receive instructions on the operating system commands to enter Module 1. From Module 1, the user may elect to perform any of the functions available in the software package.

Module 4 - LP Sensitivity Analysis. Module 4 provides several types of sensitivity analysis for linear programming problems previously solved by Modules 2 or 3. The types are:

1) finding range limits for the right-hand side,
objective function coefficients, or the variable coefficients; 2) solving for a new final tableau after one or more changes to any combination of right-hand side or other coefficients; 3) solving for a new final tableau after a new variable or a new constraint has been added. The sensitivity analysis program is structurally separated into eight parts. A master menu calls one of four sections which allow the user to specify the type of analysis desired and to enter required data. These four sections, in conjunction with three additional sections, perform the analysis and display the results on screen or printer. Each of the seven sub-programs are overlayed on the main program when needed. This allows the total package to greatly exceed the immediate memory of the computer.

The sensitivity analysis package originally included a section which would have presented the results in a high-resolution graphical display; however, the graphics package could not be overlayed. With the graphics held in memory, the remaining space was not sufficient to run any of the other sensitivity analysis sections. Because of this problem, the graphics portion was deleted.

All sections of the sensitivity analysis require data from either Module 2 or 3. This data includes the original tableaus as entered by the user in Module 1 (except for problems solved by the dual method). If the dual of the problem was solved, the sensitivity analysis receives and
displays all data as though the dual of the problem had originally been entered. This means that the right-hand-side values and the objective function coefficients will be switched and all elements of the matrix will be transposed. The sensitivity analysis section also requires the full final tableau from Modules 2 or 3 as well as a number of flags and parameters indicating the solution conditions and method.

Sensitivity analysis is based on relationships between values in the tableau. The data received from Modules 2 or 3 have the ordering of the final K (number of constraints) columns based on the original constraint type (≤, ≥, =). Sensitivity analysis is least complicated when the last K columns are aligned according to their association with the constraints. That is, the first column of the last group is associated with column 1, etc. This reordering is accomplished in Unit 48 immediately after the data is read from the data file. (See Appendix B for specific information on the units and their purposes.)

Unit 41 provides right-hand-side upper and lower bounds for the current optimal solution and displays the value of the basic variables and the objective function (z) at each of these bounds. Unit 42 provides upper and lower bounds for the objective function coefficients and for all original matrix coefficients. Units 41 and 42 require no user input other than selecting the option and the desired display
Unit 43 allows changes to one, any, or all of the original values of problems. After the effects of all changes are summed, the tableau is returned to a basic solution form, and a new optimal solution is obtained using Unit 45. Unit 43 requires user inputs for each desired change.

Unit 44 allows the addition of a new constraint or variable to the original problem. These additions are a special case of the changes investigated in Unit 43. In this case, the original values are assumed to be zero and the changes which are a result of the added constraint or variable are determined. Units 43 and 44 use Unit 47 to check the validity of inputs.

Unit 45 solves the modified problem (if possible) and then displays the new optimal solution. The display routines are generally limited to values between \(10^{-5}\) and \(10^{-5}\). Since the optimal solution must always have a coefficient value of 1 in the element representing the basic variable, the solutions tend to be "sized". Any value outside of the displayed range is suspect and may be caused by ill-conditioning. Such values are usually printed as either zeros or 8's. Ill-conditioning may be the result of the sensitivity analysis techniques and does not necessarily imply that the original problem (as modified) would be ill-conditioned if solved by the other modules.
Several special conditions may arise which will preclude completion of the desired sensitivity analysis. If the total change to any element of the primary matrix sums to approximately -1 for a variable which is in the basis and has a value of one, the new value will approximate zero. Since this value is used as a divisor for the entire row, any non-zero value in the row will approach infinity after division, and the problem will not be solvable. In this case, the user is returned to the main menu.

Whenever the new problem, developed through sensitivity analysis, is either unbounded or infeasible, these conditions are noted, and the solution process is terminated. Other than these noted special cases, the sensitivity analysis should produce the requested output.

Summary. The considerations discussed in Chapter III, combined with the desire to design an accurate and responsive analytical tool, have caused some compromises in each area. The ability to extensively edit or recover from erroneous input has decreased the possible thoroughness of instructional comments. This exchange was felt to be justified since erroneous inputs could require a complete input of data causing the user to be hesitant in future use of the software.

An area not previously discussed is the use of prompts on which disk must be available in a disk drive to insure that an input/output error does not occur. Initially, all
prompts on disk availability assumed a two-disk drive system configuration with one drive of the disk containing the module being used (See Appendix B for disk file structure). It was found that by inserting additional prompts, the LP package could be used with a one-disk drive system. This modification, although it slows the overall solution or analysis process, allows users not possessing two drives to be potential users. This was felt to be an improvement to the package because of an increased potential audience.

The above enhancement, combined with the user being allowed to select an output device, has allowed the minimal system configuration shown in Figure 3 below.

1. Apple II-plus microcomputer with 48K RAM
2. One disk drive (5 1/4 inch)
3. 16K memory expansion card
4. Video display (TV)

Figure 9. Minimal Hardware Configuration
V Conclusions and Recommendations

The major objective of this thesis was the development of an extensive linear programming software package which could be used on a microcomputer system. Several sub-objectives were also defined in the preliminary phases of the research. These sub-objectives were: 1) the design and implementation of the software package would allow maximum portability among microcomputer systems; 2) the software would be user-oriented and designed for use by non-programmer oriented persons; 3) the programs would be able to provide interactive instructional sessions on the simplex algorithm; 4) the software would provide true sensitivity analysis which could be supplemented with a graphical depiction of the parameters and their ranges.

The sub-objectives were often found to be in conflict with each other. The amount of FORTRAN code required to implement all of the desired features could not be implemented in one program due to memory size limitations. Furthermore, the code could not be stored on one disk. To retain many of the desired capabilities, a decision was made to compromise the simplicity of use of the software. The software now resides on two disks in four separate programs. This is somewhat confusing to the new user; but the disadvantages are short-lived while the advantages of retaining the desired features are long-lasting.
The decision to use two disks and four programs also resolved several other problems which were computer-memory size dependent. These retained features include: the ability to run the software on a single-drive system; user prompts abundant with instructions; and the ability to choose the formatting parameters. A desired feature which was not retained was the capability to graphically depict sensitivity range and parameters. The Pascal Operating System, as supported by the Apple II, does not allow the graphics package to be overlaid in memory when using FORTRAN. When programming in Pascal, a different memory allocation scheme is used which permits graphics overlay (Ref 15). Because of this limitation in FORTRAN, graphics and large computational programs could not be combined. A graphics capability would have required a fifth program on a third disk. A compromise had already been made in the size and complexity of the linear programming software package and, therefore, the determination was made to forego the graphical capability.

As a result of using FORTRAN, several other limitations or deficiencies are resident in the programs. There are no provisions in the programs to recover from incorrect file access. If an attempt is made to open an old file where none existed, or to open a new file when an old one of the same name existed, the program will abort. Provisions have been placed in the programs to give the user an opportunity
to correct the filename input, but mistakes are possible which will terminate the program with an execution error. Another deficiency is the lack of character string manipulation capabilities. If concatenation existed in the Apple FORTRAN subset, the appearance and meaning of some prompts and other output could have been improved. These limitations are not present in Pascal. However, implementation of these features in Pascal would require additional source code and could result in memory overload. This would require the elimination of other desired features.

There are several features which could be added to this linear programming package; however, the risk of increased complexity and size may be incurred with these additions. The first feature is the graphical capability previously discussed. The ranges and parameters derived in Module 4 could be written to a disk file. A fifth program could then be executed to display the desired values. This feature could be further enhanced with the capability to print the graphics to hard-copy.

A second area of investigation could be the problem size limitations of the programs. The current system limits the final problem to twenty constraints and sixty variables. This size approaches the system limit as currently coded. By developing a version with reduced features, the maximum size could be enlarged.
Other mathematical programming techniques, such as integer programming and goal programming are also primary candidates for future enhancements. Both techniques could be partially supported by the present modules; however, extensive additions would be required.

This software package has accomplished the objectives, with the exception of the graphics capability. Future research and microcomputer developments may allow this product to be enhanced or variations may be developed for specific purposes. Many of those included features, as well as envisioned modifications to this software, are dependent on the capabilities of the microcomputer system. It must be realized that microcomputers have finite limitations due to memory capabilities and the current language implementations. These factors and their interactions have placed obvious constraints on this linear programming implementation and requires careful research before attempting an effort of comparable magnitude.
Bibliography


APPENDIX A

USERS GUIDE
**CONTENTS**

<table>
<thead>
<tr>
<th></th>
<th>Introduction</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td>87</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Capabilities and Limitations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>II</td>
<td></td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>Capabilities</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>Limitations</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>User Input.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>III</td>
<td></td>
<td>92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>System Startup</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IV</td>
<td></td>
<td>94</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Sample Problem.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td>96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Numerical Example</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module Demonstration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module 1.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module 2.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>117</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module 3.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>151</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module 4.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>163</td>
</tr>
</tbody>
</table>
I Introduction

The primary emphasis in the main body of this thesis has been the design considerations and the method of implementation of this linear programming software package. Specific comments on the use of the package have been limited; therefore, it is the purpose of this appendix to provide the user with specific information and guidance on the use of various programs included in this LP software package. Included in this section are the specific capabilities, limitations, methods of access, and use of each program. Also, suggestions are provided which will reduce the quantity of input required by the user of these programs.

An Apple II microcomputer with 48K RAM is required for the use of this software as implemented. Also required are a 16K memory expansion card, at least one 5 1/4 inch disk drive, and a monitor. Although one disk drive is sufficient, a second drive requires significantly fewer disk changes during problem solution and speeds the process considerably. Also, note that the instructions and suggestions in this appendix apply to the Apple II microcomputer and may not be applicable to other systems.
II Capabilities and Limitations

This section specifically outlines the type of linear programming problems which may be solved and analyzed with this software. Limitations exist and they will be explicitly noted below.

Capabilities

This linear programming software package allows the user to interactively enter LP models with the variables and constraints identified only by numerical identifiers or with names associated with each. These models may be stored to a disk for later review, editing, and solution. Extensive editing features of all entries allow correction during data input or editing at a later time. These editing features also allow the user to input a model, save this model to a disk, then change selected parameters to form new LP models. After the model has been input, it may be displayed for review on the screen or output to a printer. All displays of output, prompts, or comments are screen-oriented with pauses inserted at the end of each screen display to allow inspection of the information prior to proceeding to the next display.

The remaining programs of this package are dedicated to problem solution and sensitivity analysis. Module 2, the instructional module, allows the user to input option selections regarding the application of the simplex
algorithm and receive feedback concerning the validity of the option selection. Areas in which instructional comments and guidance have been provided are tableau formulation, pivot element selection, and identification of optimal, infeasible, or unbounded solutions. Options have been provided which allow the user to specify whether to perform and receive feedback on each of the areas or to allow the program to perform the functions without user input. This permits the program to require varying degrees of user interface and corresponding degrees of feedback and instructional aid.

The instructional module only allows the solution of the primal problem of the LP model; however, it permits this solution by either the regular or the dual simplex method. When the solution process is performed using the dual simplex procedure, instructional comments similar to those available with the regular simplex procedure are presented.

The third module, annotated as the problem solver module, allows the user to solve either the primal or the dual problem of the LP model using the regular or the dual simplex procedure. This expanded capability allows the user to specify the most efficient method of problem solution. Once the type of problem and method of solution have been identified, the program performs all formulation, pivot element selection, and identification of the final solution. Prior to the solution process, the user may specify the
The last module provides the sensitivity analysis capabilities of this package. This module requires that an LP problem have been solved using either problem solving module and the solution parameters saved to disk. The user may then select one of five alternatives offered for sensitivity analysis. Resource constraint right-hand-side ranging within the present optimal solution is provided as well as constraint and cost coefficient ranging. The user is also able to change one or any combination of the original model parameters and proceed to the corresponding new final solution. This solution may be displayed in the same tabular form as discussed earlier. Two more features of the sensitivity analysis module are the ability to add a variable or a constraint to the LP model and the ability to obtain the corresponding solution. This capability allows the user to examine modified examples of the original problem without resorting to model editing and solution with Modules 1 and 2 or Modules 1 and 3.

Limitations
The limitations associated with this package are primarily a result of implementation on a microcomputer rather than a large, stationary system. The principle areas of concern are the size of the LP model and the number of digits in the various parameters of the model.
Due to memory restrictions of the Apple microcomputer, as well as most comparably sized microcomputers, the number of constraints and variables are limited in this software implementation. The maximum number of constraints and variables which the user may input has been set to:

Maximum number of constraints: 20
Maximum number of variables: 20

Note that since an augmented basis is used in problem solution, this translates to a possible twenty constraint, sixty variable problem for solution.

The other area in which limitations exist is the number of significant digits which may be accurately maintained by the microcomputer. A maximum of ten digits may be entered as coefficients and resource limits; however, any number of significant digits larger than six may be subject to round-off during computation and display. The ability to enter ten digits has been allowed since the user may need to enter a negative sign and decimal point in addition to the significant digits. Therefore, the user should be cautious in the use of numbers which have six or more significant digits.
The amount of user input differs among the various modules of this package; however, the amount required is extensive in the data entry and instructional portions of the software. Therefore, minimization of input is necessary whenever possible. Listed below are several suggestions, as well as warnings, regarding user input.

Option Selection: The user will be required to make several option selections requiring one or two digit inputs or single character inputs. An error during this input may have one of two results. If the user inputs a valid entry (one of the possible alternatives) but not the desired option designator, the user will not be able to prevent execution of the selected alternative. If the entry is not a valid alternative, the program will inform the user of an invalid entry and the proper option designator may be entered.

Model Parameters: All numeric parameters (cost and constraint coefficients, constraint right-hand sides) are checked for invalid characters in the input. Any non-numeric input except sign designators and decimal points will cause an invalid entry display followed by a request for the user to reenter the data. Due to this check of input, commas should not be entered (i.e. enter 10000, not
Positive values are assumed unless a negative sign is input. Also, all real numbers which contain no significant digits to the right of the decimal may be entered as integers (i.e. 1.0 may be entered as 1).

Inequality Input: Less-than or equal ($\leq$) must be input as less-than ($<$) inequalities while greater-than or equal ($\geq$) must be entered as greater-than inequalities ($>$).

Yes/No Inputs: All [YES] or [NO] responses may be entered by a [Y] or [N] single entry.

All Inputs: All user inputs must be completed by depressing the [RETURN] key. The computer does not attempt to read input until this action is taken. A zero may be entered as a numeric input by pressing only the [RETURN] key; this allows faster input of a model which has many zero coefficients.

Printer Option: This software package offers the user the capability to have selected output routed to a printer; however, the selection of this option without a printer being available will cause an execution error. Also, if the printer option is selected, the user must insure that the printer has been turned on and is in a printing mode. Otherwise, the system will wait indefinitely for the printer to accept information, and it will "hang" the system.
IV. **System Startup**

The compiled FORTRAN code files which form this LP package have been stored on two 5 1/4 inch floppy disks with volume names LP1 and LP2 (see Appendix B for disk file structure). As was discussed in Chapter IV, a turnkey system has been used which causes Module 1, the data base entry module, to execute automatically when the computer is turned on. However, for this system to function properly, two data files must be placed on the disk LP1 prior to its use. These files, SYSTEM.PASCAL and SYSTEM.MISCINFO, must be transferred from disk APPLE1 (Version 1.0) to LP1 for the turnkey system to operate. For those who are not familiar with the procedures required to transfer files, please refer to the operating system reference manual (Ref 1:156).

Once the transfers have been completed, the following steps should be performed to allow Module 1 to execute.

1. Place disk APPLE1 (Version 1.0) in disk drive #4 (the first disk drive is numbered as #4 with the Pascal Operating System while the second drive, if present, is numbered #5). If two drives are present, also place disk LP2 in drive #5.
2. Turn on the power switch of the Apple-II (#4 drive should activate and run for approximately five seconds).
3. Remove APPLE1 from #4 drive and replace it with LP1.

4. Press the [RESET] key (#4 drive should again run, followed by #5, if a second drive present, and then the #4 drive runs again).

At this point, a title page should appear on the screen and Module 1 has executed.

For those users not familiar with this software, the next section is devoted to an example problem and explanation. The problem demonstrates the use of all modules and may be of assistance in learning the various alternatives available and their method of activation.
This section is dedicated to a step-by-step guide through the major portions of this software package. Users may review this section to learn the specific alternatives available and their sequence of availability. It would not be feasible to demonstrate all of the possible alternatives and their use; however, the sequence and methods felt to be most commonly used will be shown with explanatory comments given for many of the other features available.

**Numerical Example**

The following numerical example will be used for the purposes of this demonstration. The example, although not a large problem, will allow the user to become familiar with the method of data entry and the output formats used in this LP package.

*Problem:* An analyst has been asked to determine the number of helicopters, by type, which would be required to move at least 2000 men and 1200 tons of equipment to a new area of operation. Three types of helicopters are available with the following capabilities:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FUEL COST ($100)</th>
<th>MEN</th>
<th>EQUIP (tons)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE1</td>
<td>30</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>TYPE2</td>
<td>22.5</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>TYPE3</td>
<td>25</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

96
The analyst has been directed to determine the mix of aircraft which minimizes cost; however, the analyst has been restricted to the use of 400 helicopters (Ref 6:75).

The above situation may be formulated as a linear programming problem as follows:

\[ \begin{align*}
  x(1) & = \text{number of TYPE1 aircraft} \\
  x(2) & = \text{number of TYPE2 aircraft} \\
  x(3) & = \text{number of TYPE3 aircraft}
\end{align*} \]

\[
\text{minimize } 30x(1) + 22.5x(2) + 25x(3)
\]

subject to

\[
\begin{align*}
  5x(1) + 8x(2) + 4x(3) & \geq 2000 \\
  6x(1) + 3x(2) + 4x(3) & \geq 1200 \\
  x(1) + x(2) + x(3) & \leq 400
\end{align*}
\]

**Module Demonstration**

In the demonstration which follows, all inputs by the user have been placed in square brackets ([ ]) to distinguish input from screen displays and prompts. All user inputs are completed by depressing the [RETURN] key to allow the program to read this input. Another area which may require clarification is the PAUSE statement which appears after displays not requiring data input. The user must depress the [RETURN] key or the [SPACE] bar whenever a PAUSE statement appears before the program will continue. The input of any other type than the two stated will cause a new PAUSE statement to appear.
Module 1. The first step required in the use of this LP software package is system initialization which is performed as described in Part IV of this appendix, System Startup. Once those procedures have been completed, the following title page will be displayed.

```
****************************
* FORTRAN BASED *
* " " *
* LINEAR PROGRAMMING *
* " " *
* FOR *
* " " *
* MICROCOMPUTERS *
****************************

BY
THEODORE R. E. FRALEY
AND
DALE A. KEM

ARE INTRODUCTORY REMARKS DESIRED?
(Y/N, RETURN) [Y]
```

Should one wish to review the introductory remarks, a [Y] would be entered (remember that the brackets indicate user input followed by a [RETURN] ) and the following comments will be displayed. Note that in the lower left corner of each display the word "PAUSE" appears. This indicates that no further displays or program progress will occur until the [SPACE] bar or [RETURN] key have been depressed.
LINEAR PROGRAMMING SOFTWARE PACKAGE

THIS PACKAGE IS DESIGNED TO ALLOW STUDENTS TO IMPROVE THEIR UNDERSTANDING OF THE SIMPLEX ALGORITHM AND ALSO TO PROVIDE THE MANAGERS AND ANALYSTS WITH A PROBLEM SOLVING TOOL.

THE PACKAGE CONSISTS OF FOUR DISTINCT PROGRAMS (ANNOTATED AS MODULES) WHOSE FUNCTIONS ARE AS FOLLOWS:

MODULE 1: DATA BASE ENTRY
MODULE 2: LP INSTRUCTION
MODULE 3: LP PROBLEM SOLVER
MODULE 4: SENSITIVITY ANALYSIS

ALL LP PROBLEMS MUST BE ENTERED INTO A DATABASE USING MODULE 1. MODULES 2 OR 3 MAY BE USED TO DETERMINE A SOLUTION TO A PROBLEM AND THIS MUST OCCUR PRIOR TO ENTERING MODULE 4.

PAUSE

INSTRUCTIONS ON HOW TO ENTER EACH MODULE WILL BE PRESENTED WHEN APPROPRIATE.

ANSWERS TO SPECIFIC QUESTIONS CONCERNING ANY MODULE WILL BE FOUND IN THE USERS GUIDE (APPENDIX A) OF THE THESIS DOCUMENTATION.

ALL RESPONSES REQUIRE A [RETURN] TO NOTE THE COMPLETION OF INPUT.

ALSO, ALL YES/NO INPUTS MAY BE ENTERED BY [Y] OR [N], RESPECTFULLY.

PAUSE

After the introductory remarks, the first menu displayed is that of module selection. Since a data base has not previously been entered, one would select option [1].
to continue in Module 1 and data base entry.

MODULE SELECTION

THE FOLLOWING OPTIONS ARE AVAILABLE:

1. DATA BASE ENTRY (ENTER DATA BASE OR EDIT CURRENT DATA BASE)

2. LP INSTRUCTIONAL MODULE

3. LP PROBLEM SOLVER MODULE

4. LP SENSITIVITY ANALYSIS MODULE

(NOTE: OPTIONS 2 OR 3 REQUIRE THAT A DATA BASE BE CURRENTLY STORED ON DISK)

(NOTE: OPTION 4 REQUIRES THAT A DATA FILE HAS BEEN SAVED UPON LEAVING THE OPTION 2 OR 3 MODULES ABOVE.)

WHICH OPTION? [1]

The following header confirms entry into the data base entry portion of Module 1.

************************************************************************
*                      *                      *
* DATA                  *
* BASE                  *
* ENTRY                 *
* MODULE                *
* MODULE 1              *
************************************************************************

PAUSE
Following the data base entry header, the user is offered several alternatives of entering a data base. Again, no previous data base exists so either option 1 or 2 must be selected. Option [2] allows the same input as option 1, but allows the constraints and variables to be identified by names as well as subscripts.

DATA BASE ENTRY

TO ENTER LP MODEL DATA BASE
YOU HAVE THE FOLLOWING OPTIONS:

1. CREATE MODEL INTERACTIVELY: SUBSCRIPTS
   (VARIABLES ANNOTATED BY SUBSCRIPTS,
   CONSTRAINTS ANNOTATED BY NUMBER ONLY)

2. CREATE MODEL INTERACTIVELY: NAMED
   VARIABLES AND CONSTRAINTS ARE
   ASSIGNED NAMES)

3. READ FROM DISK
   (PREVIOUSLY CREATED BASE)

4. DISPLAY INTRODUCTORY REMARKS

5. QUIT PROGRAM

WHICH OPTION? [2]

If a data base had been previously entered and saved to disk, one could select option 3. At that time, the following three displays of comments and prompts would appear. If this previous file had been placed on disk LPI (volume name) with a filename of BUSES, one would enter this as shown below to retrieve that model. The next two prompts appear sequentially to inform the user that the designated
disk must be in the #4 disk drive for a one drive system or in either drive for a multiple drive system. A habit of leaving disk LP1 in drive #4 and disk LP2 in drive #5 has been found to be beneficial for a two drive system. In this configuration, only when files are to be written or read from another disk are the two disks not accessible in the disk drives.

```
READ LP MODEL FROM DISK

ENTER THE DISK DRIVE NUMBER AND FILE NAME WHICH HOLDS THE MODEL DESIRED.

ENTER EXACTLY AS FOLLOWS
DISK DRIVE:FILENAME
EG. #4:FILENAME
DISK:FILENAME = [LP1:BUSSES]

INSURE THE DISK CONTAINING THE LP1:MAX
MODEL IS AVAILABLE.
PAUSE

INSURE DISK LP1 IS AVAILABLE.
PAUSE
```

Continuing with the database entry for the example problem, one would enter an identifier for later reference. Also, the type of problem is entered at this time as shown
Next, one identifies the objective name of the problem. This input is not allowed for models with only subscript designators.

At this point, the number of constraints and variables which form the LP model are entered. Nonnegativity constraints are not to be included in the number of constraints entered below.
Since a model with names has been selected, the following three displays are presented allowing the input and correction of the decision variable names. First the names are input and in the next display, a [Y] may be entered to allow correction of the names. An [N] has been entered since corrections were not needed.

VARIABLE NAME INPUT

ENTER VARIABLE NAMES WHICH CORRESPOND TO THE 3 VARIABLES THAT AFFECT COST

NAMES ARE TO BE 6 CHARACTERS OR LESS.

PAUSE

PROBLEM ID: SAMPLE PROBLEM
VARIABLE NAME INPUT

\[
\begin{align*}
X(1) &= \text{[TYPE1]} \\
X(2) &= \text{[TYPE2]} \\
X(3) &= \text{[TYPE3]}
\end{align*}
\]
The same sequence appears next for the input and correction of constraint names. Note that variable and constraint name input is not allowed for models with subscripts only. Although these names are useful for ease of variable identification, input of the data base is slowed considerably by this requirement.

**CONSTRAINT NAME INPUT**

ENTER CONSTRAINT NAMES WHICH CORRESPOND TO THE 3 CONSTRAINTS WHICH AFFECT COST

NAMES ARE TO BE 6 CHARACTERS OR LESS.

PAUSE
Next, the objective function is input following comments regarding the restrictions on numerical input. After completion of input, the option of correcting input is presented. This allows changes of any type to the objective function. Note that the coefficients input below do not include decimals for those without significant digits to the right of the decimal. This allows for more rapid input of the coefficients values.

**OBJECTIVE FUNCTION INPUT**

INPUT THE FUNCTION AS IF IT WERE IN THE FOLLOWING FORM

\[ Z = X(1) + X(2) + X(3) + \text{ETC.} \]

A MAXIMUM OF 10 ENTRIES PER COEFFICIENT INCLUDING DECIMAL AND SIGN ARE ALLOWED.

IF COEFFICIENT IS ZERO, HIT "RETURN" WITHOUT DIGIT ENTRY.

PAUSE

---

**PROBLEM ID: SAMPLE PROBLEM**

OBJECTIVE FUNCTION INPUT

COST MINIMIZATION

\[
\begin{align*}
C(1) &= \text{TYPE1} = [30] \\
C(2) &= \text{TYPE2} = [22.5] \\
C(3) &= \text{TYPE3} = [25]
\end{align*}
\]
No corrections were needed so in the objective function an [N] was entered below.

```
PROBLEM ID: SAMPLE PROBLEM
OBJECTIVE FUNCTION INPUT
COST MINIMIZATION

ARE CORRECTIONS NEEDED? [N]
```

The constraint coefficients are input next in the same type sequence as the objective function. One constraint is entered per display with the option of corrections being presented after the last constraint has been entered. This editing allows the user to change any constraint and any part of the input, including the inequality. Note that the nonnegativity constraints are not entered into the model. Also, note that the constraint coefficients are input as integers since no significant digits exist to the right of the decimal point.
INPUT CONSTRAINT VARIABLE COEFFICIENTS AS IF THE CONSTRAINT WAS IN THE FOLLOWING FORM

\[ x(1) + x(2) + x(3) \leq \text{RHS} \]

THE VARIABLE COEFFICIENTS ARE A MAXIMUM OF 10 CHARACTERS

IF COEFFICIENT IS ZERO, ENTER 0 OR HIT "RETURN" WITHOUT ENTRY.

THE LESS-THAN (\( < \)) REPRESENTS A LESS-THAN OR EQUAL INEQUALITY.

THE GREATER-THAN (\( > \)) REPRESENTS A GREATER-THAN OR EQUAL INEQUALITY.

NEGATIVE RHS IS PERMITTED.

PAUSE

PROBLEM ID: SAMPLE PROBLEM
CONSTRAINT 1 = PERSON

\[
\begin{align*}
x(1) &= \text{TYPE1} = 5 \\
x(2) &= \text{TYPE2} = 8 \\
x(3) &= \text{TYPE3} = 4 \\
\text{INEQUALITY} &= >> \\
\text{RHS} &= 2000
\end{align*}
\]

PROBLEM ID: SAMPLE PROBLEM
CONSTRAINT 2 = EQUIP

\[
\begin{align*}
x(1) &= \text{TYPE1} = 6 \\
x(2) &= \text{TYPE2} = 3 \\
x(3) &= \text{TYPE3} = 6 \\
\text{INEQUALITY} &= >> \\
\text{RHS} &= 1200
\end{align*}
\]
Following the input and correction of the constraints, the data base management menu is displayed. Option 1 allows the user to review input to insure correctness, after which the data base management menu appears again. The user may then select to edit any parameters with option 2. Once the model is corrected, it may be saved to disk by selecting option 3.

DATA BASE MANAGEMENT

THE FOLLOWING OPTIONS ARE AVAILABLE:
1. DISPLAY CURRENT LP MODEL
   (SCREEN OR PRINTER)
2. EDIT CURRENT LP MODEL
   (CHANGE ANY PARAMETER)
3. SAVE CURRENT MODEL TO DISK FILE
   (MAY THEN EDIT TO ANOTHER MODEL)
4. ENTER NEW MODEL
   (CURRENT MODEL LOST IF NOT ON DISK)
5. SOLVE PROBLEM
   (INCLUDES EDUCATIONAL, PROBLEM SOLVER, AND SENSITIVITY ANALYSIS)
6. QUIT: UNSAVED FILES DESTROYED!

WHICH OPTION? [3]
The next four displays pertain to saving the model to a disk. The user first enters the diskname:filename which the current model is to be saved under. Next, the user is given the opportunity to correct this diskname. The user is then directed to insure that the specified disk is in a disk drive. Next, the user is required to input the status of the diskname:filename combination. If one has previously saved a model to the same diskname:filename combination, but answers [N] in the third display, all input data will be lost. This will then require reinitialization of the system and reentry of the last data base.

SAVE LP MODEL TO DISK

ENTER THE DISK DRIVE NUMBER AND FILE NAME WHICH YOU WANT PROBLEM SAMPLE PROBLEM SAVED UNDER.

ENTER EXACTLY AS FOLLOWS DISK DRIVE:FILENAME

EG. #4:FILENAME

THE DRIVE:FILENAME MUST BE 10 CHARACTERS OR LESS

IF THE ABOVE IS ENTERED INCORRECTLY, YOUR MODEL WILL BE LOST!!

DISK:FILENAME = [LP]:SAMPLE

ARE CORRECTIONS NEEDED? [N]
INSURE THE DISK TO CONTAIN THE FILE
LP1: SAMPLE
IS AVAILABLE.

PAUSE

HAS THIS DISK:FILENAME COMBINATION BEEN
USED PREVIOUSLY?
(ARE YOU UPDATING A CURRENTLY EXISTING
FILE?)
(Y/N) [N]

INSURE DISK LP1 IS AVAILABLE.

After the model has been saved, control returns to the
data base management menu as shown below. At that point,
one may select option 2 and the data base editor menu would
be displayed as shown below the data base management menu.
DATA BASE MANAGEMENT

THE FOLLOWING OPTIONS ARE AVAILABLE:
1. DISPLAY CURRENT LP MODEL
   (SCREEN OR PRINTER)
2. EDIT CURRENT LP MODEL
   (CHANGE ANY PARAMETER)
3. SAVE CURRENT MODEL TO DISK FILE
   (MAY THEN EDIT TO ANOTHER MODEL)
4. ENTER NEW MODEL
   (CURRENT MODEL LOST IF NOT ON DISK)
5. SOLVE PROBLEM
   (INCLUDES EDUCATIONAL, PROBLEM SOLVER, AND SENSITIVITY ANALYSIS)
6. QUIT: UNSAVED FILES DESTROYED!

WHICH OPTION? [5]

DATA BASE EDITOR

YOU MAY EDIT THE CURRENT MODEL IN ANY OF
THE FOLLOWING MANNERS:
1. ADD A VARIABLE
2. ADD A CONSTRAINT
3. DELETE A VARIABLE
4. DELETE A CONSTRAINT
5. CHANGE COEFFICIENT BY CONSTRAINT
6. CHANGE COEFFICIENTS BY VARIABLE
7. CHANGE RHS OF CONSTRAINT
8. CHANGE CONSTRAINT INEQUALITY
9. CHANGE OBJECTIVE FUNCTION COST COEFFICIENTS
10. CHANGE MAXIMIZATION/MINIMIZATION CHOICE
11. CHANGE VARIABLE NAMES
12. CHANGE CONSTRAINT NAMES
13. RETURN TO LAST MENU
   (DATA BASE MANAGEMENT)

WHICH OPTION? [13]

As shown, extensive editing features are available and
could also have been used prior to saving the model to disk. These features could also be used to form a new LP model by changing selected parameters of the model in memory. Upon completion of the editing, one may return to the data base management menu by selecting option [13].

Option 4 of the data base management menu above could have been selected if the user wished to enter a new model after saving the first model to disk. The user, if attempting to enter a new model prior to saving the first model to disk, receives a warning that the first model must be saved or will be lost.

Option [5] has been selected in the last data base management menu signifying that the problem is now to be solved. Again, if this option had been selected prior to saving the last entered model to disk, a warning message would have appeared and the user would have been allowed to save the last model before progressing to problem solution.

Following the selection to solve the problem, the following menu, execution management, is displayed. Option [1] has been selected for this demonstration; however, option 2 could also have been selected. Option 3 is not possible at this time since a problem must have previously been solved with Module 2 or Module 3 and the results saved to disk. The user is also permitted to return to the data entry section prior to exiting this module. The user, upon selection of option 1, 2, or 3, is prompted to insure a
required disk is accessible.

EXECUTION MANAGEMENT

THE FOLLOWING OPTIONS ARE AVAILABLE:

1. LP INSTRUCTIONAL MODULE (EACH TABLEAU MAY BE DISPLAYED)
2. PROBLEM SOLVER MODULE (NO USER INTERACTION)
3. SENSITIVITY ANALYSIS MODULE
4. RETURN TO DATA BASE MANAGEMENT MENU
5. QUIT: UNSAVED FILES WILL BE LOST!

WHICH OPTION? [1]

 INSURE DISK LP1 IS AVAILABLE.
PASUE

After the selection of a module, the following display allows the user to specify a data base other than the last entered for solution by the selected module. One may enter [N] in the response below and is then allowed to specify a file as was done earlier for reading a file from disk. As shown, the last file entered is desired so a [Y] was entered. The last display of this module then appears.
Simultaneously with the display below, the control returns to the operating system. The user need only enter [X] (no [RETURN] key should be used after the input of [X]). Next, the user enters [LP1:ED] and Module 2, which contains the instructional code, will be executed. If the user had selected option 2 of the execution management menu, the only difference would have been that [LP2:TAB] would have replaced [LP1:ED] above. One need not memorize this fact since the prompt will reflect the option input by the user.

As a final note, users must insure that the disk containing the desired module is in a disk drive when the above operating system commands are entered. The user may identify the required disk by the LP1 or the LP2 preceding the colon and filename entered to the operating system. This will coincide with one of the two disks provided with this software package.
TO ENTER THE LP INSTRUCTIONAL MODULE:

TYPE

X

LP1:ED
Module 2. The last entries of the Module 1 demonstration were [X] and [LP1:ED]. These are the operating systems commands which direct the execution of the code file ED on disk LP1. Therefore, disk LP1 must be accessible when these commands are entered. Once these steps have been completed, the following header appears which confirms the execution of Module 2, the instructional module.

```
***********************
$  $  
$  LINEAR  $  
$  PROGRAMMING  $  
$  EDUCATIONAL  $  
$  MODULE  $  
$  MODULE 2  $  
***********************
```

Following the header, the default options are displayed as shown below. On the right of the display, the options which have been programmer defined are shown. These options control the extent of user interface required in the instructional areas as well as the method of solution. Options 5 through 7 allow the user to designate those tableaus to be displayed as well as their location and format. Those options desired to be changed may be selected
in any order. The number of alternatives available within the options vary; however, several options have only two possibilities (options 3, 6, and 7). These options require only that they be selected while the other options require further input to cause a change in their value. After the selection and change of any option, the default values are again displayed with the noted changes. When the user is satisfied with the defaults as displayed, option 8 should be selected to allow the program to continue. The major options and their alternatives are shown in the following sequence.

For instance, if the user would like to review the alternatives available under option 1, [1] would be input as shown below.

<table>
<thead>
<tr>
<th>DEFAULT OPTIONS</th>
<th>ENTER OPTION NUMBER TO CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. TABLEAUS FORMATION</td>
<td>USER</td>
</tr>
<tr>
<td>2. PIVOT ELEMENT SELECTION</td>
<td>USER SEL</td>
</tr>
<tr>
<td>3. DUAL PIVOTS</td>
<td>N</td>
</tr>
<tr>
<td>4. INFEASIBLE, UNBOUNDED, OPTIMAL SELECTION IDENTIFICATION</td>
<td>USER</td>
</tr>
<tr>
<td>5. TABLEAUS TO BE DISPLAYED</td>
<td></td>
</tr>
<tr>
<td>INITIAL</td>
<td>Y</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>N = 1</td>
</tr>
<tr>
<td>FINAL</td>
<td>Y</td>
</tr>
<tr>
<td>6. OUTPUT LOCATION</td>
<td>SCREEN</td>
</tr>
<tr>
<td>7. OUTPUT FORMAT</td>
<td>F FORMAT</td>
</tr>
<tr>
<td>8. NO CHANGES</td>
<td></td>
</tr>
</tbody>
</table>

*SEE DOCUMENTATION FOR EXPLANATION*
The display below describes the alternatives available for this area of the instructional aid. Option [1] was selected as shown below which will allow instructional comments to be provided for the user. Note that this was the same alternative as was originally designated as a default.

**EDUCATIONAL MODULE OPTION SELECTION**

IN ORDER TO PLACE THE LP MODEL INTO THE PROPER FORM FOR THE SIMPLEX ALGORITHM (OBJECTIVE FUNCTION CHANGES, ADDITION OF SLACK OR ARTIFICIAL VARIABLES), WHICH METHOD IS DESIRED?

1. USER SELECTS MODIFICATION AND ALGORITHM CHECKS

   OR

2. ALGORITHM PERFORMS MODIFICATIONS. (NO USER INPUT)

   WHICH OPTION? [1]

The default menu then appears and the next option to be changed may be entered. Option [2] was selected as the next to be changed.
The user is now shown the alternatives available for the pivot element selection. Selections 2 and 3 do not provide feedback to the user; however, the use of selection 2 may be beneficial to instructors who wish to demonstrate the outcome of an inappropriate pivot element selection. Option [1] has been selected below.
The default menu would again appear (not shown) and option [4] could be input, resulting in the following display. Again, the user may select the degree of user interaction desired. Option [1] has been selected to allow the demonstration of the instructional comments available. This option requires the user to respond to questions concerning the status of the tableau while option 2 would have allowed the program to perform these actions without user interface.

<table>
<thead>
<tr>
<th>EDUCATIONAL MODULE OPTION SELECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS OPTIMAL, INFEASIBLE, OR UNBOUNDED SOLUTIONS OCCUR, WHICH METHOD WOULD YOU LIKE?</td>
</tr>
<tr>
<td>1. USER ATTEMPTS TO IDENTIFY, ALGORITHM CHECKS.</td>
</tr>
<tr>
<td>2. SYSTEM IDENTIFIES AND REPORTS AS OCCURS.</td>
</tr>
</tbody>
</table>

WHICH OPTION? [1]

The default menu is displayed reflecting any changes which have been requested. Since the user may have no initial feeling for the magnitude of the final solution values, option [7] has been input to allow larger numbers to be displayed.
The option 7 selection changes the default to "E FORMAT" as shown above. Now that all desired changes have been made, option [8] is entered as shown and the program continues.
The next three displays inform the user, and in particular the one-disk-drive system user, that a particular disk must be accessible before continuing.

INSURE DISK LP1 IS AVAILABLE.
PAUSE

INSURE THE DISK CONTAINING THE LP1 SAMPLE MODEL IS AVAILABLE.
PAUSE

INSURE DISK LP1 IS AVAILABLE.
PAUSE
The following display informs the user of requirements regarding the objective function modification. If the user had selected the algorithm to perform the tableau modification, this sequence would not be displayed. The display following the instructions is the objective function as entered in Module 1.

**OBJECTIVE FUNCTION MODIFICATION**

THE OBJECTIVE FUNCTION, AS ENTERED, WILL BE DISPLAYED NEXT. AFTER THE DISPLAY, YOU WILL BE ASKED TO SELECT THE OPTION WHICH WILL TRANSFORM THE OBJECTIVE FUNCTION INTO THE PROPER TABLEAU FORM FOR THE SIMPLEX ALGORITHM.

PAUSE

**PRESENT OBJECTIVE FUNCTION**

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x(1)</td>
<td>x(2)</td>
<td>x(3)</td>
</tr>
</tbody>
</table>

MIN Z = + 3.0000E+01 + 2.2500E+01 + 2.5000E+01

PAUSE

The user has now reviewed the objective function and must select the designator for the option which will place the objective function in the standard LP form defined in Chapter II of the main body. As shown below, option [1] was selected and the feedback concerning this selection is presented. The user may then review the entered and the correct response to determine the reason for an incorrect input.
OBJECTIVE FUNCTION MODIFICATION

TO PLACE THE OBJECTIVE FUNCTION IN THE PROPER FORMAT FOR THE SIMPLEX ALGORITHM WHICH OF THE FOLLOWING SHOULD BE DONE?

1. ADD \(-C(J)\) TO BOTH SIDES OF EQUATION.
2. MULTIPLY EQUATION BY \(-1\) AND THEN ADD \(-C(J)\) TO BOTH SIDES OF EQUATION.
3. NO CHANGES ARE NECESSARY.

WHICH OPTION IS CORRECT? [1]

OPTION #1 IS INCORRECT.

THE PROPER RESPONSE WAS OPTION #2.

Regardless of user input, the objective function is properly modified and displayed as shown below. This allows the user to further review the objective function and its modification.

AFTER THE PROPER MODIFICATION, THE OBJECTIVE FUNCTION FORM IS:

\[
\begin{array}{ccc}
\text{TYPE1} & \text{TYPE2} & \text{TYPE3} \\
X(1) & X(2) & X(3) \\
\text{MAX} (-2) + 3.0000E+01 + 2.2500E+01 + 2.5000E+01 = 0
\end{array}
\]
The constraint modification sequence is similar to that shown above for the objective function. The user is given instructions followed by the display of the constraints as entered. After reviewing these constraints, the options which the user may later select are shown.

**CONTRAINT MODIFICATION**

THE CONSTRAINTS, AS ENTERED, WILL BE DISPLAYED NEXT. AFTER THE DISPLAY, YOU WILL BE SHOWN EACH OF THE CONSTRAINTS INDIVIDUALLY AND ASKED TO SELECT THE OPTION WHICH TRANSFORMS THE CONSTRAINT INTO THE PROPER SIMPLEX ALGORITHM FORM.

PAUSE

**SAMPLE PROBLEM**

**CURRENT CONSTRAINTS**

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>x(1)</td>
<td>x(2)</td>
<td>x(3)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CN NAME</th>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PERSON</td>
<td>5.0000E+00</td>
<td>8.0000E+00</td>
<td>4.0000E+00</td>
<td>2.0000E+03</td>
</tr>
<tr>
<td>2 EQUIP</td>
<td>6.0000E+00</td>
<td>3.0000E+00</td>
<td>6.0000E+00</td>
<td>1.2000E+03</td>
</tr>
<tr>
<td>3 PLANES</td>
<td>1.0000E+00</td>
<td>1.0000E+00</td>
<td>1.5000E+00</td>
<td>4.0000E+02</td>
</tr>
</tbody>
</table>

PAUSE

126
EACH CONSTRAINT WILL BE SEPARATELY DISPLAYED, THEN THE FOLLOWING OPTIONS WILL BE DISPLAYED FOR EACH CONSTRAINT. YOU WILL SELECT THE OPTION WHICH WILL PLACE THE CONSTRAINT IN THE PROPER SIMPLEX ALGORITHM FORM.

1. ADD SLACK VARIABLE ONLY.
2. SUBTRACT SURPLUS VARIABLE, ADD ARTIFICIAL VARIABLE.
3. ADD ARTIFICIAL VARIABLE ONLY.
4. MULTIPLY BY -1, SUBTRACT SURPLUS VARIABLE, ADD ARTIFICIAL VARIABLE.
5. MULTIPLY BY -1, ADD SLACK VARIABLE.
6. MULTIPLY BY -1, ADD ARTIFICIAL VARIABLE.

PAUSE

The constraints are next shown separately. This allows the user to study the constraint individually. When the user is confident that the proper option is known, the [RETURN] key or [SPACE] bar is pressed allowing the display of the options.

```
  TYPE1  TYPE2  TYPE3   
     1   1   1     

  CN1 PERSON  5.0000E+00  8.0000E+00 4.0000E+00  2.0000E+03

PAUSE
```

The user inputs the desired option (option [2] in this case) and receives immediate feedback. The user then may continue on to the remaining constraints for the same sequence of steps.
CONSTRAINT # 1

1. ADD SLACK VARIABLE ONLY.
2. SUBTRACT SURPLUS VARIABLE, ADD ARTIFICIAL VARIABLE.
3. ADD ARTIFICIAL VARIABLE ONLY.
4. MULTIPLY BY -1, SUBTRACT SURPLUS VARIABLE, ADD ARTIFICIAL VARIABLE.
5. MULTIPLY BY -1, ADD SLACK VARIABLE.
6. MULTIPLY BY -1, ADD ARTIFICIAL VARIABLE.

WHICH OPTION? [2]

OPTION #2 IS CORRECT.

PAUSE

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x( 1)</td>
<td>x( 2)</td>
<td>x( 3)</td>
</tr>
<tr>
<td>CN2 EQUIP</td>
<td>6.00000E+00</td>
<td>3.00000E+00</td>
</tr>
</tbody>
</table>

PAUSE

Note that when an incorrect response is entered, the correct response is shown while the corresponding function of each response is still visible. In this manner, the user may review any mistakes made while still being able to review the available options. Also note that the constraints are modified properly regardless of user input.
CONSTRANT # 2

1. ADD SLACK VARIABLE ONLY.
2. SUBTRACT SURPLUS VARIABLE, ADD ARTIFICIAL VARIABLE.
3. ADD ARTIFICIAL VARIABLE ONLY.
4. MULTIPLY BY -1, SUBTRACT SURPLUS VARIABLE, ADD ARTIFICIAL VARIABLE.
5. MULTIPLY BY -1, ADD SLACK VARIABLE.
6. MULTIPLY BY -1, ADD ARTIFICIAL VARIABLE.

WHICH OPTION? [1]
OPTION #1 IS INCORRECT
THE PROPER RESPONSE WAS OPTION #2

The third constraint is now examined in the same manner as the first two.
CONSTRAINT # 3
1. ADD SLACK VARIABLE ONLY.
2. SUBTRACT SURPLUS VARIABLE, ADD ARTIFICAL VARIABLE.
3. ADD ARTIFICAL VARIABLE ONLY.
4. MULTIPLY BY -1, SUBTRACT SURPLUS VARIABLE, ADD ARTIFICAL VARIABLE.
5. MULTIPLY BY -1, ADD SLACK VARIABLE.
6. MULTIPLY BY -1, ADD ARTIFICAL VARIABLE.

WHICH OPTION? [1]

OPTION #1 IS CORRECT.

Next, the user is presented instructions on the next sequence of steps. This is followed by the display of the tableau as it has been modified by the previous two sections, the objective function and constraint modification sections.

THE TABLEAU AS MODIFIED PREVIOUSLY, WILL BE DISPLAYED.

YOU WILL THEN BE ASKED IF THE TABLEAU IS IN THE CORRECT FORM FOR THE SIMPLEX ALGORITHM.

PAUSE
The user is next asked for a response concerning the form of the tableau. This input is followed by immediate feedback concerning the accuracy of this input. Note that this and the subsequent displays concerning the form of the tableau would not have appeared if the user had allowed the program to perform tableau modification without user interface.

**IS THE TABLEAU IN THE PROPER FORM FOR THE INITIAL PIVOT? [Y]**

**YOUR RESPONSE WAS INCORRECT.**

**ARTIFICIAL VARIABLES HAVE BEEN ADDED, YET THE OBJECTIVE FUNCTION HAS NOT BEEN MODIFIED (BIG M) TO REFLECT THIS.**

**PAUSE**
Further instructions are presented below followed by the display of the tableau as modified up to this point.

THE TABLEAU WILL BE DISPLAYED AND YOU WILL BE ASKED TO IDENTIFY THOSE VARIABLES WHICH THE BIG M METHOD IS TO BE APPLIED.

PAUSE

**SAMPLE PROBLEM**
CURRENT LP MODEL: MAXIMIZE COST

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
<th>SURPLS</th>
<th>SURPLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(1)</td>
<td>X(2)</td>
<td>X(3)</td>
<td>X(4)</td>
<td>X(5)</td>
</tr>
</tbody>
</table>
OBJ FUNCTION: -3.0000E+01 -2.2500E+01 -2.5000E+01 0.0000E+00 0.0000E+01

<table>
<thead>
<tr>
<th>CK</th>
<th>NAME</th>
<th>VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PERSON 7</td>
<td>5.0000E+00 8.0000E+00 4.0000E+00 -1.0000E+00 0.0000E+00</td>
</tr>
<tr>
<td>2</td>
<td>EQUIP 8</td>
<td>6.0000E+00 3.0000E+00 6.0000E+00 0.0000E+00 -1.0000E+00</td>
</tr>
<tr>
<td>3</td>
<td>PLANES 6</td>
<td>1.0000E+00 1.0000E+00 1.0000E+00 0.0000E+00 0.0000E+00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SLACK</th>
<th>ARTIF</th>
<th>ARTIF</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(6)</td>
<td>X(7)</td>
<td>X(8)</td>
<td></td>
</tr>
</tbody>
</table>
OBJ FUNCTION: 0.0000E+00 0.0000E+00 0.0000E+00 = 0.0000E+00

<table>
<thead>
<tr>
<th>CK</th>
<th>NAME</th>
<th>VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PERSON 7</td>
<td>0.0000E+00 1.0000E+00 0.0000E+00 = 2.0000E+03</td>
</tr>
<tr>
<td>2</td>
<td>EQUIP 8</td>
<td>0.0000E+00 0.0000E+00 1.0000E+00 = 1.2000E+03</td>
</tr>
<tr>
<td>3</td>
<td>PLANES 6</td>
<td>1.0000E+00 0.0000E+00 0.0000E+00 = 4.0000E+02</td>
</tr>
</tbody>
</table>

PAUSE

The user is now asked to identify those variables which require modification by the Big M method. Note that the variable subscript is entered to designate the selected variable. The input of [6] below refers to variable \( x(6) \) of the previously displayed tableau. Again feedback and instructional comments are provided immediately. The
following display also requires input similar to that discussed above.

WHICH VARIABLES REQUIRE THE USE OF THE BIG M METHOD?
(ENTER SUBSCRIPT VALUES)

FIRST VARIABLE? [6]

YOUR RESPONSE WAS INCORRECT.
THE CORRECT RESPONSE WAS VARIABLE 7
THIS IS THE FIRST ARTIFICIAL VARIABLE AND REQUIRES THE USE OF THE BIG M METHOD.
PAUSE

VARIABLES 7 THRU X(?) REQUIRE THE BIG M METHOD

LAST VARIABLE? [8]

YOUR RESPONSE WAS CORRECT.
THE LAST ARTIFICIAL VARIABLE IS # 8 AND IS THE LAST TO REQUIRE THE USE OF THE BIG M METHOD.
PAUSE

The instructions below are followed by the display of the tableau as modified at this point. This allows the user to review the modifications performed in the previous steps. Once the user has reviewed the tableau, the [RETURN] key or [SPACE] bar is depressed to allow the program to proceed.

THE TABLEAU WILL BE DISPLAYED, THEN YOU WILL BE ASKED IF IT IS IN THE PROPER FORM FOR THE INITIAL PIVOT.
PAUSE
As shown below, the user is asked again whether or not the tableau is ready for the first pivot. The opinion of the user is entered and the corresponding feedback is presented.

IS THE TABLEAU IN THE PROPER FORM FOR THE INITIAL PIVOT? [Y]

YOUR RESPONSE WAS INCORRECT.
THERE IS NO INITIAL BASIC SOLUTION SINCE THE OBJECTIVE FUNCTION COEFFICIENTS OF THE ARTIFICIAL VARIABLES ARE NOT ZERO.

PAUSE

Following the above input and comments, the final modification of the tableau occurs. As shown below, the
next step is the display of the initial tableau. Note that the above sequence of steps from the last default option display to this point would not have been performed if the user had selected the algorithm to perform the tableau modification.

**SAMPLE PROBLEM**
**BASIC SOLUTION 1**

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
<th>SURPLS</th>
<th>SURPLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>x(1)</td>
<td>x(2)</td>
<td>x(3)</td>
<td>x(4)</td>
<td>x(5)</td>
</tr>
<tr>
<td>OBJ FUNCTION</td>
<td>-3.27000E+03</td>
<td>-3.27750E+03</td>
<td>-2.97500E+03</td>
<td>1.00000E+02</td>
</tr>
<tr>
<td>CN NAME VAR</td>
<td>----------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 PERSON</td>
<td>7</td>
<td>5.00000E+00</td>
<td>8.00000E+00</td>
<td>4.00000E+00</td>
</tr>
<tr>
<td>2 EQUIP</td>
<td>8</td>
<td>6.00000E+00</td>
<td>7.00000E+00</td>
<td>6.00000E+00</td>
</tr>
<tr>
<td>3 PLANES</td>
<td>6</td>
<td>1.00000E+06</td>
<td>1.00000E+00</td>
<td>1.00000E+00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SLACK</th>
<th>ARTIF</th>
<th>ARTIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBJ FUNCTION</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>CN NAME VAR</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>1 PERSON</td>
<td>7</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>2 EQUIP</td>
<td>8</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>3 PLANES</td>
<td>6</td>
<td>1.00000E+00</td>
</tr>
</tbody>
</table>

After this tableau display, the user who has elected to identify optimal, unbounded or infeasible solutions will be asked the following four questions. Each question is displayed separately and, as shown, instructional comments accompany the user input. Note that prior to this input, [IT] may be entered allowing the user to reexamine the tableau.
TO REVIEW TABLEAU, ENTER T

WAS THE PREVIOUS TABLEAU OPTIMAL? [N]

YOUR RESPONSE WAS CORRECT
THE LAST TABLEAU WAS NOT OPTIMAL

PAUSE

TO REVIEW TABLEAU, ENTER T

IS THE SOLUTION FEASIBLE? [Y]

YOUR RESPONSE WAS INCORRECT
THE LAST TABLEAU WAS INFEASIBLE

THE SOLUTION IS INFEASIBLE SINCE THE ARTIFICIAL VARIABLE X(8) IS AT A POSITIVE LEVEL.

PAUSE

TO REVIEW TABLEAU, ENTER T

WAS THE PREVIOUS SOLUTION DEGENERATE? [N]

YOUR RESPONSE WAS CORRECT
THE LAST TABLEAU WAS NOT DEGENERATE

PAUSE
TO REVIEW TABLEAU, ENTER T

WAS THE PREVIOUS SOLUTION UNBOUNDED BASED UPON THE NEXT PIVOT COLUMN (ROW) BEING THE COLUMN (ROW) WITH THE LARGEST NEGATIVE Z(J)-C(J) (B(J)) VALUE? [Y]

YOUR RESPONSE WAS INCORRECT
THE LAST TABLEAU WAS BOUNDED

THE CURRENT TABLEAU IS BOUNDED SINCE ALL THE A(I,J) VALUES IN COLUMN 2 ARE NOT NEGATIVE OR ZERO.

PAUSE

The user may elect to have the basic variable values and objective function value displayed or to continue without this display. As shown, the user elected not to display the values by entering a [3].

WOULD YOU LIKE THE BASIC SOLUTION VALUES DISPLAYED?

1. DISPLAY ON SCREEN
2. DISPLAY ON PRINTER
3. DO NOT DISPLAY

WHICH OPTION? [3]

The next four displays request the user to enter the selected pivot column and row. Each input is accompanied by appropriate feedback. As shown, column number 3 was selected initially, but did not coincide with the algorithm.
choice. At that point, the user must select the option representing the pivot column to be used.

**WHICH COLUMN CONTAINS THE CANDIDATE ENTERING VARIABLE?**

COLUMN = [3]

**YOUR SELECTION OF PIVOT COLUMN DOES NOT MATCH THAT OF THE ALGORITHM.**

WHICH SELECTION DO YOU WISH TO USE?

1. YOUR SELECTION COLUMN = 3
   OR
2. ALGORITHM SELECTION COLUMN = 2

WHICH OPTION? [2]

Based upon the above pivot column selection, the ratios for the column are calculated and displayed. The user then enters the number of the row which is felt to be correct for the pivot element. If the user selection had not matched the algorithm selection, the user would have been allowed to change the selection as was shown above for the column selection.
RATIOS FOR COLUMN 2

<table>
<thead>
<tr>
<th>ROW</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.50000E+02</td>
</tr>
<tr>
<td>2</td>
<td>4.00000E+02</td>
</tr>
<tr>
<td>3</td>
<td>4.00000E+02</td>
</tr>
</tbody>
</table>

WHICH ROW CONTAINS THE CANDIDATE LEAVING VARIABLE?
ROW = [1]

YOUR PIVOT ROW SELECTION MATCHES THE ALGORITHM SELECTION.

PAUSE

Once the first pivot has been completed, the resulting tableau is displayed.

SAMPLE PROBLEM
BASIC SOLUTION # 2

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
<th>SURPLS</th>
<th>SURPLS</th>
</tr>
</thead>
</table>

OBJ FUNCTION: -1.22155E+03 0.00000E+00 -1.33625E+03 -1.09687E+02 3.00000E+02
CN NAME VAR: **************************************************

1 PERSON 2 6.25000E-01 1.00000E+00 5.00000E-01 -1.25000E-01 0.00000E+00
2 EQUIP 8 4.12500E+00 0.00000E+00 4.50000E+00 3.75000E-01 -1.00000E+00
3 PLANES 6 3.75000E-01 0.00000E+00 5.00000E-01 1.25000E-01 0.00000E+00

SLACK ARTIF  ARTIF
| [6]  | [7]  | [8] |

OBJ FUNCTION: 0.00000E+00 1.59687E+02 0.00000E+00 = -1.40625E+05
CN NAME VAR: ******************************************

1 PERSON 2 0.00000E+00 1.25000E-01 0.00000E+00 = 2.50000E+02
2 EQUIP 8 0.00000E+00 -3.75000E-01 1.00000E+00 = 4.50000E+02
3 PLANES 6 1.00000E+00 -1.25000E-01 0.00000E+00 = 1.50000E+02

139
After the tableau has been reviewed, the same sequence of questions, displays, and feedback are repeated for the last tableau calculated. This sequence is shown below.

**TO REVIEW TABLEAU, ENTER T**

**WAS THE PREVIOUS TABLEAU OPTIMAL? [N]**

YOUR RESPONSE WAS CORRECT
THE LAST TABLEAU WAS NOT OPTIMAL

PAUSE

**TO REVIEW TABLEAU, ENTER T**

**IS THE SOLUTION FEASIBLE? [Y]**

YOUR RESPONSE WAS INCORRECT
THE LAST TABLEAU WAS INFEASIBLE

THE SOLUTION IS INFEASIBLE SINCE THE ARTIFICIAL VARIABLE X(8) IS AT A POSITIVE LEVEL.

PAUSE

**TO REVIEW TABLEAU, ENTER T**

**WAS THE PREVIOUS SOLUTION DEGENERATE? [Y]**

YOUR RESPONSE WAS INCORRECT
THE LAST TABLEAU WAS NOT DEGENERATE

THE CURRENT TABLEAU IS NOT DEGENERATE SINCE ALL BASIC VALUES ARE AT A NON-ZERO LEVEL.

PAUSE
TO REVIEW TABLEAU, ENTER T

WAS THE PREVIOUS SOLUTION UNBOUNDED
BASED UPON THE NEXT PIVOT COLUMN (ROW)
BEING THE COLUMN (ROW) WITH THE LARGEST
NEGATIVE \( z(j) - c(j)(b(j)) \) VALUE? [N]

YOUR RESPONSE WAS CORRECT
THE LAST TABLEAU WAS BOUNDED

PAUSE

WOULD YOU LIKE THE BASIC SOLUTION VALUES
DISPLAYED?
1. DISPLAY ON SCREEN
2. DISPLAY ON PRINTER
3. DO NOT DISPLAY

WHICH OPTION? [3]

WHICH COLUMN CONTAINS THE CANDIDATE
ENTERING VARIABLE?
COLUMN = [3]

YOUR PIVOT COLUMN SELECTION MATCHES THE
ALGORITHM SELECTION.

PAUSE
The following tableau is the result of the second pivot.

<table>
<thead>
<tr>
<th></th>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
<th>SURPLS</th>
<th>SURPLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ FUNCTION</td>
<td>3.33333E-00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>1.66667E-01</td>
<td>3.05556E+00</td>
</tr>
</tbody>
</table>
| CN NAME VAR | *****************************************************
| 1 PERSON | 2 | 1.66667E-01 | 1.00000E+00 | 0.00000E+00 | -1.66667E-01 | 1.11111E-01 |
| 2 EQUIP | 3 | 9.16667E-01 | 0.00000E+00 | 1.00000E+00 | 9.33333E-02 | -2.22222E-01 |
| 3 PLANES | 6 | -8.33333E-02 | 0.00000E+00 | 0.00000E+00 | 8.33333E-02 | 1.11111E-01 |
| SLACK | ARTIF | ARTIF | RMS | |
| II (6) | X (7) | X (8) | |
| OBJ FUNCTION | 0.00000E+00 | 2.98333E+02 | 2.96944E+02 | =-7.00000E+03 |
| CN NAME VAR | *****************************************************
| 1 PERSON | 2 | 0.00000E+00 | 1.66667E-01 | -1.11111E-01 | = 2.00000E+02 |
| 2 EQUIP | 3 | 0.00000E+00 | 8.33333E-02 | 2.22222E-01 | = 1.00000E+02 |
| 3 PLANES | 6 | 1.00000E+00 | 8.33333E-02 | -1.11111E-01 | = 1.00000E+02 |

The sequence of displays which follows is the same as above with a few exceptions. Since it is found in the next
display that the last tableau was optimal, the user is questioned concerning the existence of multiple optimal solutions in addition to the previous questions.

TO REVIEW TABLEAU, ENTER T

WAS THE PREVIOUS TABLEAU OPTIMAL? [Y]
YOUR RESPONSE WAS CORRECT
THE LAST TABLEAU WAS OPTIMAL
PAUSE

TO REVIEW TABLEAU, ENTER T

IS THE OPTIMAL SOLUTION ALSO FEASIBLE? [Y]
YOUR RESPONSE WAS CORRECT
THE LAST TABLEAU WAS FEASIBLE
PAUSE

TO REVIEW TABLEAU, ENTER T

WAS THE PREVIOUS SOLUTION DEGENERATE?[N]
YOUR RESPONSE WAS CORRECT
THE LAST TABLEAU WAS NOT DEGENERATE
PAUSE
TO REVIEW TABLEAU, ENTER T

ARE THERE MULTIPLE OPTIMAL SOLUTIONS?
[Y]

YOUR RESPONSE WAS INCORRECT
THERE ARE NO MULTIPLE SOLUTIONS.

THIS IS SINCE ALL NON-BASIC VARIABLES
HAVE A VALUE OTHER THAN ZERO IN THE
OBJECTIVE FUNCTION ROW. IF A ZERO VALUE
WAS PRESENT FOR A NON-BASIC VARIABLE,
INCREASING THE VALUE OF THIS VARIABLE
WOULD NOT CHANGE THE Z VALUE.

PAUSE

Since an optimal solution has been obtained, the pivot
element selections are no longer required.

The final tableau display is repeated following the
above questions since an optimal solution has been obtained.
This also occurs when unbounded or infeasible solutions
exist. This is followed by the opportunity to display the
basic values and objective function value. Option [1] has
been selected for screen output as shown below.
### BASIC SOLUTION # 3
#### FINAL TABLEAU - OPTIMAL

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
<th>SURPLS</th>
<th>SURPLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(1)</td>
<td>X(2)</td>
<td>X(3)</td>
<td>X(4)</td>
<td>X(5)</td>
</tr>
<tr>
<td>OBJ FUNCTION</td>
<td>3.33337E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>1.66667E+00</td>
</tr>
<tr>
<td>CN NAME</td>
<td>VAR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 PERSON</td>
<td>2</td>
<td>1.66667E-01</td>
<td>1.00000E+00</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>2 EQUIP</td>
<td>3</td>
<td>9.16667E-01</td>
<td>0.00000E+00</td>
<td>1.00000E+00</td>
</tr>
<tr>
<td>3 PLANES</td>
<td>6</td>
<td>-8.33333E-02</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SLACK</th>
<th>ARTIF</th>
<th>ARTIF</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>I(6)</td>
<td>I(7)</td>
<td>I(8)</td>
<td>RHS</td>
</tr>
<tr>
<td>OBJ FUNCTION</td>
<td>0.00000E+00</td>
<td>2.96944E+02</td>
<td>2.96944E+02</td>
</tr>
<tr>
<td>CN NAME</td>
<td>VAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 PERSON</td>
<td>2</td>
<td>9.00000E+00</td>
<td>1.66667E-01</td>
</tr>
<tr>
<td>2 EQUIP</td>
<td>3</td>
<td>0.00000E+00</td>
<td>-8.33333E-02</td>
</tr>
<tr>
<td>3 PLANES</td>
<td>6</td>
<td>1.00000E+00</td>
<td>-8.33333E-02</td>
</tr>
</tbody>
</table>

**WOULDN'T YOU LIKE THE BASIC SOLUTION VALUES DISPLAYED?**

1. DISPLAY ON SCREEN  
2. DISPLAY ON PRINTER  
3. DO NOT DISPLAY

**WHICH OPTION?** [1]

**SAMPLE PROBLEM**

**BASIC SOLUTION # 3**

- TYPE2 = X(2) = 2.00000E+02  
- TYPE3 = X(3) = 1.00000E+02  
- SLACK = X(6) = 1.00000E+02  

I = -7.00000E+03

145
The user may elect to perform additional pivots at this point. This has been provided to allow the user to recover from improper pivot element selections resulting in an infeasible solution. If this option is selected, the ability to perform dual pivots is automatically provided. As shown, this option was not elected.

WOULD YOU LIKE TO PERFORM FURTHER PIVOTS ON THIS TABLEAU? [N]

The results of the problem must be saved to disk to allow for sensitivity analysis. Since the analyst would like to further study the above solution, a [Y] has been entered.

TO PERFORM SENSITIVITY ANALYSIS ON THIS MODEL, THE INFORMATION OF THE CURRENT TABLEAU MUST BE SAVED TO DISK.

DO YOU WISH TO SAVE THIS FILE TO DISK? [Y]
The sequence shown below requires the input of a diskname:filename of the disk and file in which the results are to be saved. The subsequent displays prompt the user to place the correct disk in a drive.

SAVE LP MODEL TO DISK

ENTER THE DISK DRIVE NUMBER AND FILE NAME YOU WANT THE CURRENT TABLEAU OF SAMPLE PROBLEM SAVED UNDER.

ENTER EXACTLY AS FOLLOWS
DISK DRIVE:FILENAME

EG. #4:FILENAME

THE DRIVE:FILENAME MUST BE 10 CHARACTERS OR LESS

DO NOT USE THE SAME NAME USED WHEN THE ORIGINAL MODEL WAS ENTERED.

DISK:FILENAME = [LP1:SAMCM2]

ARE CORRECTIONS NEEDED? [N]

Note that this prompt is for disk LP2 and not LP1. The users of a one-drive system must remove LP1 and insert LP2 at this time.

INSURE DISK LP2 IS AVAILABLE.

PAUSE

One disk-drive users must reinsert disk LP1.
INSURE THE DISK TO CONTAIN THE FILE
LP1:SAMCM2
IS AVAILABLE.

PAUSE

The user must insure the following question is answered
correctly. If entered as shown below when a file already
exists on LP1 with the name SAMCM2, an output error will
cause the loss of the solution parameters in memory.

HAS THIS DISK:FILENAME COMBINATION BEEN
USED PREVIOUSLY?
(ARE YOU UPDATING A CURRENTLY EXISTING
FILE?)

(Y/N) [N]

The prompt below advises the user that the file of the
original model input into Module 1 must be available at this
time.

INSURE THE DISK CONTAINING THE
LP1:SAMPLE
MODEL IS AVAILABLE.

PAUSE
INSURE THE DISK TO CONTAIN THE FILE
LP1: SAMCM2
IS AVAILABLE.
PAUSE

INSURE DISK LP1 IS AVAILABLE.
PAUSE

If another LP model were available, the user could enter [Y] and would be asked to input the diskname:filename of the model desired. Since no other models are available, [N] was entered.

WOULD YOU LIKE TO STUDY ANOTHER MODEL WHICH HAS BEEN SAVED TO DISK? [N]

INSURE DISK LP1 IS AVAILABLE.
PAUSE

The last display of this module is shown below. This provides the user with the required operating system commands to return to Module 1. The user is cautioned that the period following STARTUP must be entered or Module 1 will not execute.
TO ENTER THE LP DATABASE MODULE:

TYPE
X
LP1:SYSTEM.STARTUP.

The above sequence has given an outline of the use of Module 2. Not all options were employed in the demonstration; however, those able to perform the above steps should not encounter problems in other methods of application.

The sequence of steps normally used in problem solution and analysis would lead the analyst now to the sensitivity module. Since the method of access and use of Module 4, the sensitivity analysis program, is identical following both Modules 2 and 3, this explanation will be presented after the Module 3 demonstration.
Module 3. The method of accessing Module 3 when Module 1 has terminated was briefly noted earlier. It was shown that to enter Module 2, the user would enter [X] and [LP1:ED]. The commands for entering Module 3 are [X] and [LP2:TAB]. Note that no [RETURN] is required following the [X]. Also note that the command [LP2:TAB] communicates that the file TAB on disk LP2 be executed. This requires that disk LP2 be accessible when entering the above commands. After these commands have been entered, the following header will be displayed. This confirms entry into Module 3.

```
*............................*
*   LINEAR                *
*   PROGRAMMING           *
*   PROBLEM               *
*   SOLVER                *
*   MODULE                *
*   MODULE 3              *
*............................*
PAUSE
```

Once either the [RETURN] key or the [SPACE] bar has been depressed, the user is informed to insert disk LP1. One-drive-system users must carefully read these prompts and insure the required disk is available to avoid output errors and data loss.
The user is presented the diskname:filename of the file currently identified as the model to be studied. Should the user not want to study the file shown, an [N] may be entered and the user may then identify the file desired. As shown, the file currently identified is the one desired so a [Y] was entered.

The one-drive system user must now reinsert disk LP2.

The default options are displayed next with the programmer-defined defaults shown on the right. The first option is the only one which is not available in Module 2. The selection of option 1 would change the default to "DUAL"
and the module would then convert the primal LP model into its dual model. This transformation would be performed without user interface; however, the user would be required to input a new diskname:filename which the dual problem formulation would be stored under. Although this option was not selected for the demonstration, it may be useful in reducing the number of iterations required to solve a selected LP problem.

For this demonstration, option [3] has been entered to show that the output format for the printer is identical to that used in Module 2 for screen output.

<table>
<thead>
<tr>
<th>DEFAULT OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTER OPTION NUMBER TO CHANGE</td>
</tr>
<tr>
<td>1. PROBLEM TO SOLVE PRIMAL</td>
</tr>
<tr>
<td>2. SOLVE BY DUAL PIVOTS N</td>
</tr>
<tr>
<td>3. OUTPUT LOCATION SCREEN</td>
</tr>
<tr>
<td>4. OUTPUT FORMAT F FORMAT</td>
</tr>
<tr>
<td>5. TABLEAUS TO BE DISPLAYED</td>
</tr>
<tr>
<td>INITIAL Y</td>
</tr>
<tr>
<td>INTERMEDIATE N = 1</td>
</tr>
<tr>
<td>FINAL Y</td>
</tr>
<tr>
<td>6. NO CHANGES</td>
</tr>
<tr>
<td>* SEE DOCUMENTATION FOR EXPLANATION</td>
</tr>
<tr>
<td>WHICH OPTION (ENTER 1-6) ? [3]</td>
</tr>
</tbody>
</table>

As shown below, the output default value reflects the previous change. One must insure at this time that the
printer is turned on and is in a mode which allows printing. Otherwise, the system will wait indefinitely for the printer to accept information. To insure no confusion exists, the output location refers to the device to which tabular data will be transmitted. This selection has no effect on the location of user prompts and instructions. These will always be displayed on the screen.

The option referring to which tableaus are to be displayed will be demonstrated below. As an initial default, all tableaus are to be displayed; however, since these tableaus were shown in the Module 2 demonstration, only the final tableau will be requested here. To change these defaults, option [5] was entered.

<table>
<thead>
<tr>
<th>DEFAULT OPTIONS</th>
<th>ENTER OPTION NUMBER TO CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. PROBLEM TO SOLVE</td>
<td>PRIMAL</td>
</tr>
<tr>
<td>2. SOLVE BY DUAL PIVOTS</td>
<td>N</td>
</tr>
<tr>
<td>3. OUTPUT LOCATION</td>
<td>PRINTER</td>
</tr>
<tr>
<td>4. OUTPUT FORMAT</td>
<td>F FORMAT</td>
</tr>
<tr>
<td>5. TABLEAUS TO BE DISPLAYED</td>
<td></td>
</tr>
<tr>
<td>INITIAL</td>
<td>Y</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>N = 1</td>
</tr>
<tr>
<td>FINAL</td>
<td>Y</td>
</tr>
<tr>
<td>6. NO CHANGES</td>
<td></td>
</tr>
<tr>
<td>8 SEE DOCUMENTATION FOR EXPLANATION</td>
<td></td>
</tr>
<tr>
<td>WHICH OPTION (ENTER 1-6) ? [5]</td>
<td></td>
</tr>
</tbody>
</table>
The user is asked the sequentially shown questions below. Since only the final tableau is desired, the responses [N], [N], [Y] were entered. If the user had desired to see a selected number of the intermediate tableaus, a [Y] would have been entered for the second response. The user would then be asked to enter a value for the length of cycle between intermediate tableau output. If a [2] were entered, the second, fourth, sixth, etc. intermediate tableaus would be displayed on the selected device.

```
PROBLEM SOLVER OPTION SELECTION
WHICH TABLEAUS WOULD YOU LIKE DISPLAYED?
INITIAL TABLEAU? (Y/N) [N]
INTERMEDIATE TABLEAUS? (Y/N) [N]
FINAL TABLEAU? (Y/N) [Y]
```

As shown below, the option 5 default value reflects the changes to this point. Once the user has made all desired changes, option [6] is entered to continue.
The user is next prompted to insert the disk which contains the original model. Following that action, the user is informed that disk LP2 must be available.

After the \[RETURN\] key or \[SPACE\] bar is depressed in the above prompt, the program begins the formulation and...
iterative solution process. Since the initial and intermediate tableaus were not requested, the next display is the final tableau. This tableau will be displayed on the printer in the format identical to that of Module 2. The tableau is shown below.

<table>
<thead>
<tr>
<th>TYPE1</th>
<th>TYPE2</th>
<th>TYPE3</th>
<th>SURPLS</th>
<th>SURPLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(1)</td>
<td>X(2)</td>
<td>X(3)</td>
<td>X(4)</td>
<td>X(5)</td>
</tr>
<tr>
<td>OBJ FUNCTION</td>
<td>3.33337</td>
<td>.00000</td>
<td>.00000</td>
<td>1.66667</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CN NAME</th>
<th>VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PERSON</td>
<td>2</td>
</tr>
<tr>
<td>2 EQUIP</td>
<td>3</td>
</tr>
<tr>
<td>3 PLANES</td>
<td>6</td>
</tr>
<tr>
<td>OBJ FUNCTION</td>
<td>.16667</td>
</tr>
<tr>
<td>CN NAME</td>
<td>VAR</td>
</tr>
<tr>
<td>1 PERSON</td>
<td>2</td>
</tr>
<tr>
<td>2 EQUIP</td>
<td>3</td>
</tr>
<tr>
<td>3 PLANES</td>
<td>6</td>
</tr>
<tr>
<td>OBJ FUNCTION</td>
<td>.00000</td>
</tr>
</tbody>
</table>

The user is asked whether or not the basic variable values and objective function value are to be displayed. Again, to show the printer output format, a [2] has been entered followed by the output received.
WOULD YOU LIKE THE BASIC SOLUTION VALUES DISPLAYED?

1. DISPLAY ON SCREEN
2. DISPLAY ON PRINTER
3. DO NOT DISPLAY

WHICH OPTION? [2]

SAMPLE PROBLEM
BASIC SOLUTION # 3

\[
\begin{align*}
\text{TYPE2} = & \ x(2) = 200.00000 \\
\text{TYPE3} = & \ x(3) = 100.00000 \\
\text{SLACK} = & \ x(6) = 100.00000 \\
\end{align*}
\]
\[z = -7000.00000\]

The user must next respond if sensitivity analysis will be performed on the solution. If so, a [Y] is entered followed by a request for a diskname:filename to which the solution parameters of Module 3 will be saved. As shown below, [LP1:SAMCM3] has been entered with [N] entered to show no corrections are needed on the filename. The user must be careful not to use a previously used diskname:filename of a file which is still required. If a previously used name has been entered (for example LP1:SAMCM2 from Module 2), this would cause the previous file to be destroyed.
TO PERFORM SENSITIVITY ANALYSIS ON THIS MODEL, THE INFORMATION OF THE CURRENT TABLEAU MUST BE SAVED TO DISK.

DO YOU WANT TO SAVE THIS FILE TO DISK?

[Y]

SAVE LP MODEL TO DISK

ENTER THE DISK DRIVE NUMBER AND FILE NAME YOU WANT THE CURRENT TABLEAU OF SAMPLE PROBLEM SAVED UNDER.

ENTER EXACTLY AS FOLLOWS

DISK DRIVE:FILENAME

E.G. 4:FILENAME

THE DRIVE:FILENAME MUST BE 10 CHARACTERS OR LESS.

DO NOT USE THE SAME NAME USED WHEN THE ORIGINAL MODEL WAS ENTERED.

DISK:FILENAME = [LP1:SMCM3]

ARE CORRECTIONS NEEDED? [N]

The following two messages reference disk availability and should be carefully read, especially for the one-drive-system users.

INSURE DISK LP2: IS AVAILABLE.

PAUSE
INSURE THE DISK TO CONTAIN THE FILE
LP1: SAMCM3
IS AVAILABLE
PAUSE

An [N] has been entered below signifying that the
diskname:filename combination has not been used previously.
If one wishes to overwrite an old file, one may enter the
previously used diskname:filename above and a [Y] below to
accomplish this.

HAS THIS DISK:FILENAME COMBINATION BEEN
USED PREVIOUSLY?
(ARE YOU UPDATING A CURRENTLY EXISTING
FILE?)

(Y/N) [N]

The user is again prompted to insure the availability
of specific disks and files.

INSURE THE DISK CONTAINING THE
LP1: SAMPLE
MODEL IS AVAILABLE.
PAUSE
The user may specify that another model be solved at this time by entering [Y] below. This would then be followed by a diskname:filename input of the desired model. This allows the user to enter several models with Module 1 and then transition to Module 3 and solve all the models without repeated moves between modules. This is the recommended procedure for a multiple problem solving session.

Since another model does not currently exist, [N] has been entered followed by a prompt for disk LP2.

The last inputs required in this module are those
commands which cause the transition to Module 1. The commands \[X\] and \[LP1:SYSTEM.STARTUP.\] are entered with control being returned to Module 1. From that point, instructions on the commands to enter any module may be requested.

TO ENTER THE LP DATABASE MODULE:

\[
\text{TYPE}
\]
\[
X
\]
\[
LP1:SYSTEM.STARTUP.
\]

The next section will discuss the sensitivity analysis module, Module 4, and its method of access and use.
Module 4. This module can only be used after a database has been established using Modules 2 or 3. Upon completion of those modules, you will be directed to type [X] (execute) followed by [LP2:SEN]. When this has been done the following page will appear.

PLEASE SELECT ONE ITEM BY NUMBER

1) RANGE LIMITS----RIGHT-HAND-SIDE AND ASSOCIATED Z VALUES
2) RANGE LIMITS-----A(I,J) & C(J)
3) CHECK OPTIMALITY FOR MULTIPLE A(I,J), B(I), OR C(J) CHANGES
4) ADD A VARIABLE OR A CONSTRAINT
5) EXIT PROGRAM

Four different sensitivity analysis options are available. The first selection does right-hand-side ranging and determines the associated value of z. The second option does constraint coefficient and objective function coefficient ranging. The third selection allows multiple changes to the original problem and finds a new optimal solution if desired. The fourth option finds a new optimal solution after a new constraint or variable has been added.
DO YOU WANT THE OUTPUT TO GO TO:

S)CREEEN
P)RINTER
OR
B)OTH

SELECT S, P, OR B

Output is available on the screen or the printer or both simultaneously if desired. The letter preceding the choice must be entered.

ENSURE DISK LP2: IS AVAILABLE
PAUSE

Disk LP2: contains a file which holds the name of the current data file. This disk must be available or an execution error will occur. This is fatal.
THE CURRENT DATA FILE IS
LP2:SAMPLE

DO YOU WISH TO USE THIS TABLEAU
[Y]

The file name read from LP2: is shown. If a different file name is desired, enter N. The program will then request the new file name. The new file name must then be entered.

ENSURE THE DISK CONTAINING
LP2:SAMPLE
IS AVAILABLE
PAUSE

The disk containing the data file must be present to avoid a fatal execution error.
ENSURE LP2: IS AVAILABLE
PAUSE

The program disk, LP2, must be returned if it had been removed.

********************************************************************
RIGHT HAND SIDE RANGE LIMITS
CONSTRAINT # 1

ORIGINAL RIGHT HAND SIDE = 2000.00000
LOWER BOUND = 800.00000
UPPER BOUND = 3200.00000
********************************************************************

AT THE LOWER BOUND--AT THE UPPER BOUND
X(2) = .00000 X(2) = 400.00000
X(3) = 200.00000 X(3) = .00000
X(8) = 200.00000 X(8) = .00001
Z = -5000.00000 Z = -9000.00000
PAUSE

Output from selection 1 is shown. A similar amount of data is presented for each constraint.
**If option 2 had been selected, the same data input routine would have been encountered. The constraint coefficient ranging is shown.**

<table>
<thead>
<tr>
<th>COEFFICIENT</th>
<th>LOWERLIMIT</th>
<th>UPPERLIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(1,1)</td>
<td>NO LIMIT</td>
<td>7.00000</td>
</tr>
<tr>
<td>A(1,2)</td>
<td>5.00000</td>
<td>NO LIMIT</td>
</tr>
<tr>
<td>A(1,3)</td>
<td>1.33333</td>
<td>8.88884</td>
</tr>
<tr>
<td>A(2,1)</td>
<td>NO LIMIT</td>
<td>7.09093</td>
</tr>
<tr>
<td>A(2,2)</td>
<td>-6.00000</td>
<td>4.80000</td>
</tr>
<tr>
<td>A(2,3)</td>
<td>5.05881</td>
<td>NO LIMIT</td>
</tr>
<tr>
<td>A(3,1)</td>
<td>NO LIMIT</td>
<td>NO LIMIT</td>
</tr>
<tr>
<td>A(3,2)</td>
<td>NO LIMIT</td>
<td>1.50000</td>
</tr>
<tr>
<td>A(3,3)</td>
<td>NO LIMIT</td>
<td>2.00000</td>
</tr>
</tbody>
</table>

**PAUSE**

**The objective function coefficient ranging is presented on a separate page.**
THIS PROGRAM ACCEPTS MULTIPLE CHANGES TO A FINAL TABLEAU AND CHECKS WHETHER OR NOT THE CURRENT SOLUTION IS OPTIMAL FOR THE NEW PARAMETERS

PAUSE

If option 3 had been selected, the caption shown above would appear following the data input routine.

SELECT THE PARAMETERS TO BE CHANGED

1) \( C(J) \)
2) \( A(I,J) \)
3) \( B(I) \)
4) CHANGES COMPLETE
5) RETURN TO MAIN MENU

[2]

Option 3 allows changes to any or all coefficients of the original problem. After each type of change (1, 2, or 3 above) the menu is presented. The choice shown is [2] (changes to the constraint coefficients).
The user must first enter the row to be changed.

The column to be changed is entered next.

The original value is shown, the new value is entered.
When all changes to the constraint coefficients have been completed, a [D] is entered. If you desire to make more changes to these coefficients after other changes have been entered, it is permissible and has no ill effect on the outcome.

SELECT THE PARAMETERS TO BE CHANGED

1) C(J)
2) A(I,J)
3) B(I)
4) CHANGES COMPLETE
5) RETURN TO MAIN MENU

You may select 1, 2, or 3 as many times as desired, including changes to coefficients which have already been changed. On the second change, the original value will be shown. When all desired changes have been entered, select number [4].
The program determines whether or not the changes will cause a basis change.

DO YOU WISH TO SOLVE THIS TABLEAU

SELECT 'Y' OR 'N'

[Y]

If you do not wish to see the new tableau, you may return to the main menu by typing [N].

FINAL TABLEAU - OPTIMAL

PAUSE

This banner announces that a final solution is available and that it is optimal. Other conditions (degenerate) would be shown if they existed.
DO YOU WANT THE OUTPUT IN

1) E FORMAT

OR

2) F FORMAT

Output for the tableaus is available in either E or F format. Enter the number of your choice.

<table>
<thead>
<tr>
<th>OBJ FUNCTION</th>
<th>X(1)</th>
<th>X(2)</th>
<th>X(3)</th>
<th>X(4)</th>
<th>X(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.67862</td>
<td>0.0000</td>
<td>0.0000</td>
<td>2.32143</td>
<td>2.61904</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CN NAME</th>
<th>VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The final tableau is presented. The output is in 80 column format. To see the right 40 columns, type [CONTROL-A].
The 80 column format allows 5 variables to be shown at one time (both sides), The display is continued until all data has been shown.

\[
\begin{array}{ccccc}
X(6) & X(7) & X(8) & \text{RHS} \\
\text{OBJ FUNCTION} & 297.67900 & 297.38100 & .00000 & -7785.71000 \\
\text{CN NAME VAR} & \text{__________________________________________} \\
1 & 2 & .14286 & -.09524 & .00000 = 171.42900 \\
2 & 3 & -.03571 & .19046 & .00000 = 157.14300 \\
3 & 8 & -.10714 & -.09524 & 1.00000 = 71.42860 \\
\end{array}
\]

PAUSE

The final solution is presented separately. Following this display, the program returns to the main menu, just as it did after the results of options 1 and 2 were shown.
THIS SEGMENT ALLOWS YOU TO ADD AN ADDITIONAL CONSTRAINT OR VARIABLE TO AN ALREADY SOLVED LINEAR PROGRAMMING PROBLEM

PAUSE

This caption is shown after data retrieval when option [4] was selected.

DO YOU WISH TO ADD A:

C)ONSTRAINT
OR
V)ARIABLE
SELECT 'C' OR 'V'
[V]

You can enter either one new constraint or one new variable. Select the letter of your choice.

PLEASE ENTER THE COEFFICIENT FOR THE OBJECTIVE FUNCTION

C(4) = [35]

If a problem originally had three variables, the new variable would be shown as number 4. The variables added during the previous solution are moved to the right.
PLEASE ENTER THE COEFFICIENT FOR EACH CONSTRAINT

\[ \begin{align*}
A(1,4) &= 7 \\
A(2,4) &= 5 \\
A(3,4) &= 1
\end{align*} \]

The constraint coefficients for the new variable are entered next.

FINAL TABLEAU - OPTIMAL
MULTIPLE OPTIMAL SOLUTIONS EXIST

PAUSE

If the user requests a full solution (as shown in option 3) the final conditions will be displayed. The full final tableau will be displayed after this statement as it was in option three.
TO ENTER THE LP DATABASE MODULE:

TYPE

X

LP1:SYSTEM.STARTUP.

PAUSE

If option [5] on the main menu is chosen, this instruction is presented. By typing [LP1:SYSTEM.STARTUP.] after the [PAUSE] and [X] (for execution), the program will return to the master menu in Module 1.
APPENDIX B

PROGRAMMERS' GUIDE
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Introduction</td>
<td>179</td>
</tr>
<tr>
<td>II. Microcomputer Dependent Features</td>
<td>180</td>
</tr>
<tr>
<td>III. Disk File Structure</td>
<td>185</td>
</tr>
<tr>
<td>IV. LP Package Structure</td>
<td>189</td>
</tr>
<tr>
<td>Module 1</td>
<td>190</td>
</tr>
<tr>
<td>Module 2</td>
<td>191</td>
</tr>
<tr>
<td>Module 3</td>
<td>192</td>
</tr>
<tr>
<td>Module 4</td>
<td>193</td>
</tr>
<tr>
<td>V. Variable List</td>
<td>194</td>
</tr>
<tr>
<td>Main Variable List</td>
<td>195</td>
</tr>
<tr>
<td>Module 1 Variable List</td>
<td>201</td>
</tr>
<tr>
<td>Module 2 Variable List</td>
<td>203</td>
</tr>
<tr>
<td>Module 3 Variable List</td>
<td>207</td>
</tr>
<tr>
<td>Module 4 Variable List</td>
<td>210</td>
</tr>
<tr>
<td>VI. Program Listings</td>
<td>216</td>
</tr>
<tr>
<td>Module 1</td>
<td>218</td>
</tr>
<tr>
<td>Module 2</td>
<td>279</td>
</tr>
<tr>
<td>Module 3</td>
<td>340</td>
</tr>
<tr>
<td>Module 4</td>
<td>378</td>
</tr>
</tbody>
</table>
I Introduction

The objective of the Programmers' Guide presented in this Appendix is to provide general information and guidance to those programmers and analysts who wish to modify and/or expand the linear programming package developed in this thesis. Information will be presented which will aid in the location of specific code, the interaction of this code with other units of code, and the specific purpose of each block of code. A section of this guide discusses the user-created disk files and the purpose of each file. Another section explains those procedures which are known to be peculiar to the Apple FORTRAN utilized in the LP package implementation. This section will be of specific interest to those who wish to translate all or a portion of this code for use on another computer, either micro or mainframe. The last three sections are devoted to the program code structure, variables, and the text listings as implemented in this thesis.
II Microcomputer Dependent Features

This linear programming package, which consists of four distinct main programs, has been written in Apple FORTRAN as supported by the Apple II and Apple II-plus microcomputers. This FORTRAN version uses the Apple Pascal Operating System which incorporates UCSD Pascal (Ref 1). Although the Apple FORTRAN language was created with the American National Standards Institute (ANSI) FORTRAN 77 subset as its primary reference, certain limitations and extensions do exist. The purpose of this section is to note those areas which do not conform to the ANSI 77 subset of FORTRAN. Those areas which are noted should be carefully examined prior to translation of these programs for implementation on other computer systems. Only the areas not conforming to the ANSI 77 subset need to be examined when translating to other systems which fully support the ANSI 77 language subset.

The first section discusses the areas in which the Apple FORTRAN language does not conform to the ANSI 77 subset. Although the ANSI 77 subset specifies that integer and real data types will require the same amount of memory, Apple FORTRAN does not. Integers require two bytes while reals require four bytes (Ref 1:220). This specification places restrictions on the numerical range of both data types and may be different from that of a system which conforms to the ANSI 77 subset.
Apple FORTRAN also supports some features which are included in the full FORTRAN language but not in the ANSI 77 subset. Several of these features are used in this program.

Subscript expressions: Apple FORTRAN and the full FORTRAN language support array elements as subscript expressions while the ANSI 77 subset does not (Ref 1:220). For example, if \( X(1) = 3 \), Apple FORTRAN allows the \((3,2)\) element of a \( Y \) matrix to be represented as \( Y[X(1),2] \). Conformance to this standard may be accomplished by assigning to a temporary variable the value of the array element and then using this temporary variable as the subscript expression. For the above example, one must state \( Z = X(1) \) and then denote the \((3,2)\) element as \( Y[Z,2] \). This requires the designation of another integer variable, and therefore, more memory will be required.

Limits of a DO statement: Apple FORTRAN, as does the full language, places no restrictions on the integer expressions representing the limits of a DO statement, while the ANSI subset is somewhat restrictive (Ref 1:220). Violation of the ANSI 1977 FORTRAN subset standard may be avoided by designating a temporary variable to represent the limit expression at the cost of memory. An example which Apple FORTRAN allows but the subset does not would be:

\[
\text{DO 300 } i = 1, (A+B)
\]

The equivalent statement for the subset would replace the expression \((A+B)\) by a single variable.
Expressions in the input/output list of a WRITE statement: Again, the subset is the most restrictive and does not allow expressions as elements of a WRITE statement. Apple FORTRAN does support expressions in the I/O list, but the expression must not begin with a left parenthesis (Ref 1:221). This inconvenience may be overcome by using a leading addition operator symbol. This peculiarity may be removed in translation to another system through the use of a temporary variable. This may require a larger memory space. An example of this would be to replace the expression (A+B) in the input/output list with a single variable.

File structures: The file structure of Apple FORTRAN extends beyond the subset. The ANSI subset allows only unformatted, direct access files and formatted, sequential files. Apple FORTRAN supports both formatted and unformatted in either direct access or sequential files (Ref 1:221). Due to this difference, the OPEN and CLOSE statements referring to these files may not conform to the subset language. All data files of this software package are unformatted sequential files and, therefore, do not conform to the ANSI 77 subset. Consequently, the OPEN and CLOSE statements of these files do not conform to the ANSI subset. To translate these programs to a system whose FORTRAN conforms to the ANSI 77 subset, one could add format specifiers for each of the input/output elements of the READ
and WRITE statements and change the OPEN and CLOSE statements accordingly. Specifically, these changes would be required in files which use units 3, 4, or 7 as the input/output units.

CHAR intrinsic function: Apple FORTRAN conforms to the full language but not to the ANSI 77 subset (Ref 1:220-221). The FORTRAN 77 subset does not specify a collating sequence for all possible characters but does specify general guidelines for such a sequence (Ref 3:193). This allows differences to be present among implementations. Apple FORTRAN uses the ASCII (American Standard Code for Information Interchange) in its CHAR intrinsic function implementation. If the new system does not use the ASCII collating sequence, appropriate changes must be made to the present programs prior to translation.

The following features are supported by the Apple FORTRAN language but are not in the subset or the full FORTRAN language (Ref 1:221). Compiler directives, annotated by a "$" in column one, have been used to allow the overlaying of compilation units. Without this feature, each program would have exceeded the memory capabilities of the Apple II-plus microcomputer and prevented the implementation of the software package. This is an area which must be considered very carefully prior to translation attempts. If the new target system is not substantially larger than the Apple II-plus (48K RAM plus a 16K language
card) or does not support some type of overlay operation, translation of this software package may not be practical.

The edit control character "$" is a special Apple FORTRAN feature. This character prevents a line feed following a READ or WRITE statement (Ref 1222). This feature has been used extensively in the WRITE statements which prompt user inputs. This has allowed user input to appear on the same line of the monitor as the prompt and aids greatly in legibility. This feature could be eliminated on translation with careful attention required for the tableau displays.

The previous discussion has noted those areas which are Apple FORTRAN specific. When translating these programs for implementation on another system, one must equally consider the corresponding machine-dependent features of the new target system. These features may coincide with those discussed above and therefore require minimal effort. However, features which are included in the ANSI FORTRAN 77 subset, but are not supported by the new system, must be carefully researched to insure the possibility of a successful translation.
III  Disk File Structure

The object code files, which contain the compiled and linked FORTRAN source code files shown in Part VI of this Appendix, are placed on one of two disks. Each disk also contains a required data file. These two disks have been given the volume names of LP1: and LP2: and contain the following files:

LP1:
- SYSTEM.STARTUP (Module 1)
- ED.CODE (Module 2)
- LPDATA (Data file)

LP2:
- TAB.CODE (Module 3)
- SEN.CODE (Module 4)
- LPDATAW (Data file)

The code files have their corresponding module numbers in parentheses to the right, and the two data files have been annotated for future reference.

The four code files can be placed on a single disk due to their combined size. This factor, combined with the fact that each module is a separate program, required the creation of the two data files named LPDATA and LPDATAW on disks LP1: and LP2:, respectively. Both data files are unformatted, sequential files which contain a character string of maximum length 10. These character strings represent the disk volume number or disk name and filename of a user created data file. These programmer-defined files
contain the volume number or disk name and filename which the user has input as the storage location of either the data file of a model entered or the solution of a linear programming problem.

LPDATA contains the user defined data volume or diskname:filename created by Module 1 when the user either saved a LP model to disk or edited a model currently on disk. The file also is used as an information carrier (transfer file) to either Module 2 or 3, whichever is selected when leaving Module 1. When the user attempts to transfer from Module 1 (data base entry) to either of the problem solver modules (Modules 2 and 3), the user is prompted to input the name of the data file which will be studied in the problem solver module selected. The user inputs a volume or diskname:filename and this is written to LP1:LPDATA. When the user begins either of the problem solver modules, LP1:LPDATA is read. The program then directs the problem solver module to read the designated file contained in LP1:LPDATA.

The same logic is also present in the LP2:LPDATA. This file contains the user-defined volume number or disk name:filename of the data file created by either Modules 2 or 3. The volume number or diskname:filename contained in LP2:LPDATA is the file which contains the results of either problem solver module. When the user begins Module 4, LP2:LPDATA is read to direct the sensitivity analysis
module to read the designated file. **LP2:LPDATAW** may also be changed when transferring directly from Module 1 to Module 4.

The two files discussed above serve to link the various modules together by identifying the location and name of the needed data files. When a problem has been entered using Module 1, the user is prompted to save this data under a user-specified disk volume or disk name and filename. The same sequence also occurs upon completion of the problem solver modules. These files, whose volume number or diskname:filenames are placed in **LPDATA** or **LPDATAW**, may be saved to disk **LP1:**, **LP2:**, or any disk which the user designates in the volume name.

The data files which store the LP models and solutions of the model are unformatted, sequential files. Those files created by Module 1 contain the LP model and are configured in a manner so that Modules 2 and 3 may interpret them. Those data files created by Modules 2 and 3 contain the final results of an LP problem. These files are configured such that Module 4 is capable of interpreting them.

The disk files which have been provided on the two disks must remain as shown. If either **LP1:LPDATA** or **LP2:LPDATAW** is removed, an execution error will occur since the programs will attempt to open those files on their respective disks. Any changing of files must be done in conjunction with corresponding code changes to prevent
execution errors.
IV LP Package Structure

The four main programs, designated as Modules 1 through 4, which form this LP package have each been compiled in separate units called compilation units. After the compilation of each unit in a module, the units were linked into four distinct object code files and stored on disk under a volume:filename. This process of separate compilation permits the use of the OVERLAY procedure which allows a compilation unit to be resident in memory only while in use. When the overlayed unit is no longer required for processing, resident memory is available for use by other overlay units. This OVERLAY procedure allows each program to be much larger than would have been possible if the entire object code of a module were stored in resident memory.

Listed below are the module numbers followed by the volume or diskname:filename where each module is stored on the two disks provided with this LP package. Next, for each module, the compilation unit names are shown. Below each of the compilation unit names are those text files which are present in each compilation unit. The order of listing of the unit and text file names are the same as shown in PART VI. This will aid the programmer in locating the desired text files.
LP1: SYSTEM STARTUP

UNIT 10
PROGRAM DATAB

UNIT 11
SUBROUTINE DATAS
SUBROUTINE DATAN

UNIT 12
SUBROUTINE EDIT
SUBROUTINE VNCH

UNIT 13
SUBROUTINE ADVAR
SUBROUTINE OBJCH

UNIT 14
SUBROUTINE CNVA
SUBROUTINE DELCON
SUBROUTINE DELVAR

UNIT 15
SUBROUTINE ICNRCH
SUBROUTINE ADCON
SUBROUTINE DISPLAY

UNIT 16
SUBROUTINE SAVE
SUBROUTINE INIT
SUBROUTINE DATAD
SUBROUTINE HEADER
SUBROUTINE MODUL(INEW)
SUBROUTINE DBHED
SUBROUTINE INTRO
SUBROUTINE DBE
SUBROUTINE DBM
SUBROUTINE DEM

UNIT 17
SUBROUTINE CHECK(E, INVAL, RNEW)
SUBROUTINE CHECK2(E, D, HVAL, INVAL, INEW)
SUBROUTINE CHECK3(E, INVAL, INEW)
Module 2

UNIT20
PROGRAM EDUC

UNIT21
SUBROUTINE OBMDU
SUBROUTINE OPTION

UNIT22
SUBROUTINE READY
SUBROUTINE CNMDU

UNIT23
SUBROUTINE OPT
SUBROUTINE TCAL

UNIT24
SUBROUTINE PIVOT
SUBROUTINE WORK
SUBROUTINE OVER(RES)

UNIT25
SUBROUTINE HEADER
SUBROUTINE ASKQ(ASK)
SUBROUTINE QUESTN
SUBROUTINE BISM
SUBROUTINE INDEX
SUBROUTINE MODIFA
SUBROUTINE INTRD

UNIT26
SUBROUTINE TDISPL
SUBROUTINE BASDIS

UNIT27
SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
Module 3

LP2: TAB

UNIT30
PROGRAM PROBS

UNIT32
SUBROUTINE OPTN

UNIT33
SUBROUTINE WORK
SUBROUTINE OPTB

UNIT34
SUBROUTINE CONVRT
SUBROUTINE ACNCH
SUBROUTINE INRD
SUBROUTINE NFILE(N)

UNIT35
SUBROUTINE PSHED
SUBROUTINE ASKQ(ASK)
SUBROUTINE BISM
SUBROUTINE INDEX
SUBROUTINE MODIFP
SUBROUTINE MODIFD

UNIT36
SUBROUTINE TDISPL

UNIT37
SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
MODULE 4

LP2:SEN

UNIT40
    PROGRAM MAINSA
    SUBROUTINE SELECT

UNIT41
    SUBROUTINE COMRHS

UNIT42
    SUBROUTINE COEFFR

UNIT43
    SUBROUTINE MULCN

UNIT44
    SUBROUTINE ADDCN

UNIT45
    SUBROUTINE SOLVE
    SUBROUTINE OPTB
    SUBROUTINE WORK
    SUBROUTINE TDISPL

UNIT47
    SUBROUTINE CHECK2(E,D,HVAL,INVAL,INW)
    SUBROUTINE CHECK(E,INVAL,INW)

UNIT48
    SUBROUTINE RETRIV

COMVAR.TEXT
This section discusses all variables contained in this LP software package. It is divided into five subsections. The first subsection discusses those variables which are present in two or more of the four modules and are identically defined. Those variables which are present in two or more, but not all of the modules, have been identified by indicating the modules in which they are used. The next four subsections describe those variables which are specific to just one the four modules. Also listed in the individual module variable listings are those variables which may have different meanings or value ranges in other modules.

A person studying the text files and requiring the meaning of specific variables should first check the respective module variable listing. If the variable is not found there, it will be defined in the main variable listing. Also, if the dimension of an array has been specified by an asterisk (*), the dimension of the array may not be the same in each use. This notation is used only when the array elements are assigned by a data statement each time the subroutine is called.
Main Variable Listing

A(20,60) Real array which contains the models' constraint coefficients, including the surplus, slack, and artificial variable coefficients. (Modules 1, 2, 3, and 4)

ALLOW($) Character array with each element a maximum length of 1. The array contains the integer and symbolic characters which are allowed as user inputs. It is used as a reference to validate user input.

AO(20,20) Real array which contains the original models' constraint coefficients prior to pivot or tableau modification. (Modules 2, 3, and 4)

ARTV(20) Integer array which contains the constraint numbers of those constraints which contain artificial variables.

ASK Integer flag which specifies whether or not another model is to be studied before exiting present module. (Modules 2 and 3 only)

0 = Exit module
1 = Remain in present module

B(20) Real array which contains the original right hand-sides of the constraints. (Modules 1, 2, and 3)

BASIC Integer variable which contains the current iteration number of basic solutions, both feasible and infeasible. (Modules 2 and 3 only)

BM Real variable which contains the value used for M in application of the "Big M" Method. (Modules 2 and 3 only)

C(60) Real array which contains the original objective function coefficients and also the (Z(J)-C(J)) values during subsequent pivots. (Modules 1, 2, and 3)

CB(20) Integer array which contains the variable subscripts of the basic variables. (Modules 2, 3, and 4)
CN(20) Character array with each element a maximum length of 6. The array contains the constraint names assigned by user.

D Integer dummy argument which contains the maximum number of user input characters which will be verified.

E(10) Character array with each element a maximum length of 1. The array is a dummy argument which is used by subroutines which validate user input.

FMT Integer flag which denotes whether output is in E or F format. (Modules 2 and 3 only)
   0 = E format
   1 = F format

FN Character variable with maximum length 10. It contains the disk name:filename of the file currently being studied.

FNO Character variable with maximum length 10. It contains the disk name:filename of the file which has been modified by Module 2 or 3, while the new file (name presently in FN) is being created for further study with Module 4. (Modules 2 and 3 only)

GNES Real variable which contains the largest negative \((Z(J)-C(J))\) during the iterative process of determining the pivot column. (Modules 2, 3, and 4)

HOLD Real variable which contains the tableau element of the pivot column and row currently being modified in the iterative step. (Modules 2, 3, and 4)

HVAL Integer dummy argument which contains the largest integer value allowed as user input.

IBTAB Integer variable which denotes the interval between displayed intermediate basic tableaus. (Modules 2 and 3 only)
   0 = Do not display intermediate basic tableaus
   1 = Display every intermediate basic tableau
   2 = Display every second intermediate basic tableau, etc.
IFLAG(1)-(10) Integer flag. See variable list preceding each module listing for specific meaning in each module.

IFTAB Integer flag which denotes whether or not final tableau is displayed. (Modules 2 and 3 only)
1 = Display final tableau
2 = Do not display final tableau

INDEXE Integer variable which specifies the variable subscripts of the artificial variables.

INDEXG Integer variable which specifies the variable subscripts of the surplus variables.

INDEXL Integer variable which specifies the variable subscripts of the slack variables.

INEQ(20) Integer array which contains the type of inequality or equality of each constraint.
0 = Less-than or equal
1 = Greater-than or equal
2 = Equality

INEW Integer variable which is used as both the actual and dummy arguments of the subroutines which validate user input.

INVAL Integer flag which is used as both the actual and dummy arguments of the subroutines which validate user input.
0 = User input is valid
1 = User input is invalid

ITAB Integer flag which denotes whether or not initial basic tableau is to be displayed.
1 = Display initial basic tableau
2 = Do not display initial basic tableau

K Integer variable which contains the number of constraints in the model.

KFA Integer variable which contains the column number of the first artificial variable. (Modules 2, 3, and 4)
KFS  Integer variable which contains the column number of the first slack variable. (Modules 2 and 3 only)

KFSA  Integer variable which contains the column number of the first surplus variable. (Modules 2 and 3 only)

KFSU  Integer variable which contains the column number of the last surplus variable. (Modules 2 and 3 only)

MN  Character variable of maximum length 3. It contains either "MAX" or "MIN" for maximization or minimization, respectively.

MXMN  Integer flag which denotes whether original problem was maximization or minimization.
   1 = Maximization
   2 = Minimization

NEC  Integer variable which contains the number of equality constraints.

NBC  Integer variable which contains the number of greater-than or equal constraints.

NLC  Integer variable which contains the number of less-than or equal constraints.

OBJN  Character variable of maximum length 10. It contains the name of the objective function.

OPTS  Integer flag which denotes whether or not last basic solution was optimal. (Modules 2, 3, and 4)
   0 = Non-optimal
   1 = Optimal

OUTP  Integer flag which denotes whether output is to be displayed on screen or printer. (Modules 2 and 3 only)
   1 = Display on screen
   2 = Display on printer

P(I0)  Character array with each element a maximum length 1. All user inputs are read as characters. It is also used as actual arguments to subroutine calls which verify user inputs.
PELE  Real variable which contains the coefficient value of the pivot element designated by PK and PR. (Modules 2 and 3 only)

PIN Eq(20) Character array with each element a maximum length 1. It contains the symbolic representation of the equality or inequality for each constraint.

PK  Integer variable which contains the column selected for the current pivot. (Modules 2, 3, and 4)

PN  Character variable of maximum length 20. It contains the problem name supplied by user for current model.

PR  Integer variable which contains the row selected for the current pivot. (Modules 2, 3, and 4)

SPR  Real variable which contains the smallest ratio of the right-hand side/pivot column element for all constraints. (Modules 2, 3 and 4)

SUM  Real variable used as temporary sum of a summation process. (Modules 2 and 3 only)

T  Integer variable which contains the number of 80 column widths required to display tableau.

TIE  Integer flag which denotes a tie exists for entering or leaving variable. (Modules 2 and 3 only)

V  Integer variable which contains the number of variables in the model excluding surplus, slack and artificial variables.

VN(20) Character array with each element a maximum length of 6. It contains the variable names assigned by user.

VT  Integer variable which contains the total number of variables in the model, including surplus, slack and artificials. (Modules 2, 3, and 4)
XB(20)  Real array which contains the constraint right-hand sides of tableau.

Z  Real variable which contains the current objective function value.
Module 1 Variable Listing

CHAK  Integer variable which contains the column number which the user has selected to make coefficient corrections.

CHAN  Character variable of maximum length 6. It contains the new constraint name which user has defined prior to its assignment to CN(I).

CHARO Integer variable which contains the number of the constraint which the user has selected to make corrections.

D  Integer dummy argument which contains the maximum number of user input digits.

DECIMA Integer flag which denotes whether or not a decimal had been found during the process of user input validation.

<table>
<thead>
<tr>
<th>DECIMA</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No decimal processed</td>
</tr>
<tr>
<td>1</td>
<td>Decimal processed</td>
</tr>
</tbody>
</table>

IFLAG(1) Not used

IFLAG(2) Integer flag which denotes whether or not model has been saved to disk since entering of data or change of data.

<table>
<thead>
<tr>
<th>IFLAG(2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not saved to disk</td>
</tr>
<tr>
<td>1</td>
<td>Saved to disk</td>
</tr>
</tbody>
</table>

IFLAG(3) Integer flag which denotes whether or not tableau is in proper form for initial pivot.

<table>
<thead>
<tr>
<th>IFLAG(3)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not in proper form</td>
</tr>
<tr>
<td>1</td>
<td>Form is correct</td>
</tr>
</tbody>
</table>

IFLAG(4) Integer flag which denotes form of objective function.

<table>
<thead>
<tr>
<th>IFLAG(4)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Form is Z=X</td>
</tr>
<tr>
<td>1</td>
<td>Form is Z-X=0</td>
</tr>
</tbody>
</table>

IFLAG(5) Integer flag which denotes whether or not model contains variable, constraint and objective function names.

<table>
<thead>
<tr>
<th>IFLAG(5)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No name, subscripts only</td>
</tr>
<tr>
<td>1</td>
<td>Names and subscripts</td>
</tr>
<tr>
<td>IFLAG(6)</td>
<td>Integer flag which denotes the changes desired in a constraint.</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>0 = Change coefficient, inequality, and right hand side</td>
</tr>
<tr>
<td></td>
<td>1 = Change coefficient only</td>
</tr>
<tr>
<td></td>
<td>2 = Change inequality only</td>
</tr>
<tr>
<td></td>
<td>3 = Change right-hand side only</td>
</tr>
<tr>
<td></td>
<td>4 = Change constraint name only</td>
</tr>
</tbody>
</table>

| IFLAG(7)  | Not used.                                                     |

<table>
<thead>
<tr>
<th>IFLAG(8)</th>
<th>Integer flag which denotes the changes in the objective function.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 = Change cost coefficient and maximization/minimization choice</td>
</tr>
<tr>
<td></td>
<td>1 = Change cost coefficient only</td>
</tr>
<tr>
<td></td>
<td>2 = Change maximization/minimization choice only</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IFLAG(9)</th>
<th>Integer flag which denotes whether to display objective function and constraints or objective function only.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 = Display objective function and constraints</td>
</tr>
<tr>
<td></td>
<td>1 = Display constraints only</td>
</tr>
</tbody>
</table>

| IFLAG(10) | Not used.                                                                                       |

| M         | Real variable which is a multiplier to properly place the decimal in the verified user input.          |

<table>
<thead>
<tr>
<th>NEGAT</th>
<th>Integer flag which denotes whether user input was a positive or negative value.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 = Positive input</td>
</tr>
<tr>
<td></td>
<td>1 = Negative input</td>
</tr>
</tbody>
</table>

| RNEW      | Real actual and dummy argument to SUBROUTINE CHECK2(E, INVAL, RNEW)                                 |
Module 2 Variable Listing

**CD**  
Character variable of maximum length 7. It contains a character string for display noting a correct response in SUBROUTINE OPT.

**CD(20)**  
Real array which contains the absolute value of the constraint's original right-hand side. It is located in SUBROUTINE ASKQ(ASK).

**D(10)**  
Character array with each element a maximum length of 1. Actual argument in subroutine call statements located in SUBROUTINE OPTION. User input is read into this array.

**F**  
Character variable of maximum length 1. It is located in SUBROUTINE OPT for reading user responses.

**IFLAG(1)**  
Used as a dummy storage area to prevent writing over other IFLAG(*) variables. Also an integer variable which denotes whether a basic solution is infeasible due to a negative right-hand side, and if so, which constraint contains the negative right-hand side.

- 0 = Solution not infeasible due to negative right-hand side
- 1 = Solution infeasible due to negative right-hand side in constraint I

**IFLAG(2)**  
Integer flag which denotes whether or not model has been saved to disk since entering of data or change of data.

- 0 = Not saved to disk
- 1 = Saved to disk

**IFLAG(3)**  
Integer flag which denotes that screen is to be cleared after display of tableau.

- 0 = Do not clear screen
- 1 = Clear screen

**IFLAG(4)**  
Integer flag which denotes whether or not multiple optimal solutions exist SUBROUTINE OPT. Also used in SUBROUTINE PIVOT as an integer variable which contains the column selection of the algorithm.

- 0 = No multiple optimal solutions
- 1 = Multiple optimal solutions exist
IFLAG(5)  Integer flag which denotes whether or not model contains variable, constraint, and objective function names.
- 0 = No names, subscripts only
- 1 = Names and subscripts

IFLAG(6)  Integer variable which contains the basic variable subscript of the degenerate variable in SUBROUTINE OPT. Also used in SUBROUTINE PIVOT as an integer variable which contains the row selection of the algorithm.
- 0 = Solution not degenerate
- else = Basic variable subscript which is zero

IFLAG(7)  Integer flag which denotes whether or not current solution is unbounded or bounded
- 0 = Bounded
- 1 = Unbounded

IFLAG(8)  Integer flag utilized in SUBROUTINE OPT to determine if variable is a basic variable.
- 0 = Non basic variable
- 1 = Basic variable

IFLAG(9)  Integer variable which denotes whether to display current constraints only, current LP model without noting basic variables, or LP model with basic variables annotated.
- 0 = Current LP model without annotating basic variables
- 1 = Current constraints only
- 2 = Current LP model with basic variables annotated

IFLAG(10) Integer flag which denotes whether or not further pivots are allowed or desired.
- 0 = Further pivots allowable and/or desired
- 5 = Further pivots not desired and/or allowed

INC  Character variable with maximum length of 9. It contains character string for display noting incorrect response in SUBROUTINE OPT.
INEQ(20)  Integer array which contains values
designating the type of equality or
inequality after constraints with negative
right-hand sides have been multiplied by -1.
  0 = Less-than or equal
  1 = Greater-than or equal
  2 = Equality

INFI  Integer variable which denotes whether or
not a basic solution is infeasible due to an
artificial variable being negative, and if
so, the constraint number of the negative
artificial variable.
  0 = Solution not infeasible due to
      negative artificial variable
  1 = Solution infeasible due to
      negative artificial variable in
      constraint I

L  Integer flag which denotes whether or not
primal pivots are permissible on the current
tableau.
  0 = Primal pivot not permissible
  1 = Primal pivot is permissible

M  Integer flag which denotes whether or not
dual pivots are permissible on the current
tableau.
  0 = Dual pivot not permissible
  1 = Dual pivot is permissible

MOD  Integer flag which denotes whether the user
or the algorithm should modify the initial
tableau into the proper simplex form.
  1 = User modification
  2 = Algorithm modification

NEC  Integer variable which contains the number
of equality constraints after constraints
with negative right-hand sides have been
multiplied by -1.

NGC  Integer variable which contains the number
of greater-than or equal constraints after
constraints with negative right-hand sides
have been multiplied by -1.

NLC  Integer variable which contains the number
of less-than or equal constraints after
constraints with negative right-hand sides
have been multiplied by -1.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNU</td>
<td>Character variable of maximum length 14. It contains a character string for display noting whether solution was not optimal, nondegenerate, or unbounded.</td>
</tr>
<tr>
<td>ODB</td>
<td>Character variable of maximum length 10. It contains a character string for display noting whether solution was optimal, degenerate, or bounded.</td>
</tr>
</tbody>
</table>
| OIU      | Integer flag which denotes whether the user or the algorithm will identify optimal, infeasible, and unbounded solutions.  
1 = User  
2 = Algorithm |
| PES      | Integer flag which denotes the method of pivot element selection.  
1 = User selects, algorithm checks  
2 = User selects, no algorithm check  
3 = Algorithm selects, no user input |
| PKS      | Integer variable which contains the user selected pivot column. |
| PRS      | Integer variable which contains the user selected pivot row. |
| RES      | Integer flag which denotes whether or not the user wishes to perform a pivot with a pivot element that is equal to or approximately zero. |
| RATIO    | Real variable which contains the ratio of the right-hand side of row I/coefficient of row I, column PK element. |
| S        | Integer variable which contains the proper response value to question asked of user. |
### Module 3 Variable Listing

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2(20)</td>
<td>Real array which contains the C(J)'s for the dual problem (the X(I)'s of the primal problem).</td>
</tr>
</tbody>
</table>
| DUAL    | Integer flag which denotes whether or not dual pivots are to be allowed in problem solution.  
  - 1 = Dual pivots are not to be used  
  - 2 = Dual pivots may be used |
| IE      | Integer counter which contains the number of unconstrained variables added to the dual model. |
| IFLAG(1)| Used as a dummy storage area to prevent writing over other IFLAG($) VALUES. |
| IFLAG(2)| Integer flag which denotes whether or not model has been saved to disk since entering of data or change of data.  
  - 0 = Not saved to disk  
  - 1 = Saved to disk |
| IFLAG(3)| Not used. |
| IFLAG(4)| Integer flag which denotes whether or not multiple optimal solutions exist.  
  - 0 = No multiple optimal solutions  
  - 1 = Multiple optimal solutions exist |
| IFLAG(5)| Integer flag which denotes whether or not model contains variable, constraint, and objective function names.  
  - 0 = No names, subscripts on  
  - 1 = Names and subscripts |
| IFLAG(6)| Integer flag which denotes whether the solution is degenerate or nondegenerate.  
  - 0 = Nondegenerate  
  - 1 = Degenerate |
| IFLAG(7)| Integer flag which denotes whether current solution is unbounded or bounded.  
  - 0 = Bounded  
  - 1 = Unbounded |
IFLAG(8) Integer flag utilized in SUBROUTINE OPTB to determine if variable a is basic variable.
   0 = Non basic variable
   1 = Basic variable

IFLAG(9) Integer flag which denotes whether or not the current solution is to be displayed.
   0 = Do not display current solution
   1 = Display current solution

IFLAG(10) Integer flag which denotes whether solution of model was performed by primal pivots or dual pivots.
   0 = Primal pivots
   1 = Dual pivots

IT Integer flag which denotes a constraint is required to be added to insure that an initial pivot may be performed.
   0 = No constraint added
   1 = Constraint added

INFP Integer flag which denotes whether solution is feasible or infeasible due to either a negative right-hand side or an artificial variable at a positive level.
   0 = Feasible
   1 = Infeasible

K2 Integer variable which contains the number of constraints of the dual problem.

N Integer actual and dummy argument which denotes whether variables or constraints have been added to the model.
   1 = Variables
   2 = Constraints

NEC Integer variable which contains the number of equality constraints after constraints with negative right-hand sides have been multiplied by \(-1\).

NGC Integer variable which contains the number of greater-than or equal constraints after constraints with negative right-hand sides have been multiplied by \(-1\).
NLC
Integer variable which contains the number of less-than or equal constraints after constraints with negative right-hand sides have been multiplied by -1.

PROBT
Integer flag which denotes whether problem to be solved is the primal or dual problem of the current model.
1 = Primal
2 = Dual

TN
Character variable of maximum length 11. It contains a string to be displayed annotating whether constraints or variables have been added to the model.

V2
Integer variable which contains the number of variables in the dual problem.

VN2(20)
Character array with each element a maximum length 6. It contains the variable names of the dual problem.
Module 4 Variable Listing

AF(20,60) The matrix of real variables which are the final tableau values of the full matrix. These values are read from the datafile and are then modified by some sections of Module 4 in order to obtain a new final tableau.

ARTVAR An integer variable used to count the number of artificial variables in the problem and to determine which column in B-inverse is associated with the given constraint.

BASIC(20) An integer variable used to indicate whether or not a particular column in the A matrix is in the basis.

\[ 0 = \text{not in basis} \]
\[ 1 = \text{in basis} \]

BO(20) The vector of original values of the right-hand side which were entered in Module 1.

BF(20) The vector of final values for the right-hand side from Module 3 or, after modification, the new final values.

CO(20) The vector of original objective function coefficients from Module 1.

CF(20) The vector of objective function coefficients in the final tableau from Module 3 or, after modification, the new final tableau values.

CKILL1 A real variable used to check whether or not a lower bound will make the sensitivity analysis ill-conditioned.

CKILL2 A real variable used to check whether or not an upper bound will make the sensitivity analysis ill-conditioned.

CLOWER A real variable used to compute the lower bound on each objective function coefficient.

CDL An integer variable used to denote the columns under consideration.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTR</td>
<td>An integer variable used to denote the constraint under consideration.</td>
</tr>
<tr>
<td>CUPPER</td>
<td>A real variable used to compute the upper bound on each objective function coefficient.</td>
</tr>
<tr>
<td>DELADN</td>
<td>A real variable used to compute the minimum negative change to each element of the original A matrix which would cause a multiple optimal solution in the final tableau.</td>
</tr>
<tr>
<td>DELAUP</td>
<td>A real variable used to compute the minimum positive change to each element of the original A matrix which would cause a multiple optimal solution in the final tableau.</td>
</tr>
<tr>
<td>DELTA(20)</td>
<td>A temporary vector of real variables which holds the values of the new column of an added variable while the original values are being multiplied by B-inverse.</td>
</tr>
<tr>
<td>DELTA(20,20)</td>
<td>A temporary matrix of real variables which holds the change (delta) to each of the elements in the A matrix.</td>
</tr>
<tr>
<td>DELTAB(20)</td>
<td>A temporary vector of real variables which holds the change (delta) to each of the elements in the B vector.</td>
</tr>
<tr>
<td>DELTAC(20)</td>
<td>A temporary vector of real variables which holds the change (delta) to each of the elements in the C vector.</td>
</tr>
<tr>
<td>HEAD1</td>
<td>An integer variable which indicates whether or not a heading has been displayed on the screen.</td>
</tr>
<tr>
<td></td>
<td>0 = Heading has not been displayed</td>
</tr>
<tr>
<td></td>
<td>1 = Heading has been displayed</td>
</tr>
<tr>
<td>HEAD2</td>
<td>An integer variable which indicates whether or not a heading has been printed.</td>
</tr>
<tr>
<td></td>
<td>0 = Heading has not been printed</td>
</tr>
<tr>
<td></td>
<td>1 = Heading has been printed</td>
</tr>
<tr>
<td>IFLAG(4)</td>
<td>An integer variable used to indicate a multiple optimal solution.</td>
</tr>
<tr>
<td></td>
<td>0 = Not multiple optimal</td>
</tr>
<tr>
<td></td>
<td>1 = Multiple optimal</td>
</tr>
</tbody>
</table>
IFLAG(5)  An integer variable used to indicate whether or not named resources and variables are used.
          0 = Names not used
          1 = Names used

IFLAG(6)  An integer variable which indicates whether or not the problem is degenerate.
          0 = Not degenerate
          1 = Degenerate

IFLAG(7)  An integer variable which indicates whether or not the problem is unbounded.
          0 = Not unbounded
          1 = Unbounded

IFLAG(8)  An integer variable used as a temporary indicator when checking for multiple optimal solutions.

IFLAG(9)  An integer variable used as an indicator to prevent manipulation of possibly ill-conditioned matrices.

IFLAG(10) An integer variable which shows when dual pivots have been used during the initial tableau solution.
           0 = No dual pivots
           1 = Dual pivots

ILL1     An integer variable which indicates whether or not the lower bound of an element in the A matrix may present an ill-conditioned problem when solved through sensitivity analysis.

ILL2     An integer variable which indicates whether or not the upper bound of an element in the A matrix may present an ill-conditioned problem when solved through sensitivity analysis.

J        An integer variable generally used in DO loops to denote columns I through V or VT.

LWBD(20,20) A real matrix used during right-hand-side ranging to compute the lower bound of a column of the A matrix associated with a particular constraint.
LINES
An integer variable used to count the number of lines which have been displayed on the screen.

LINEP
An integer variable used to count the number of lines which have been printed on a page.

LWBD
A real variable used to compute the lower bound of an element in the A matrix.

NEWA(20,20)
A real matrix used to hold the new (user input) values of the A elements.

NEWB(20)
A real vector used to hold the new values of the right-hand side. The first element of the vector (NEWB(1)) is occasionally used as a temporary holding variable during computations.

NEWCJ(20)
A real vector used to hold the new values of the objective function coefficients.

RMAX(20)
A real vector used to compute the minimum positive resource (right-hand-side) change which would force a change in the basis.

RMIN(20)
A real vector used to compute the minimum negative resource (right-hand-side) change which would force a change in the basis.

ROW
An integer variable generally used in DO loops to vary the constraints from 1 to K while working with a different constraint under the variable name CONSTR.

RSCH(20,20)
A real matrix which holds the ratio between the right-hand-side value and the A element of the column associated with the constraint under consideration. The minimum positive and negative ratios determine the maximum right-hand-side changes allowed for the constraint while maintaining the current basis.

RSLLIM(20)
A real vector which indicates the lower limit for each right-hand-side element.

RSUPLIM
A real vector which indicates the upper limit for each element of the right-hand side.
SELINP(10) A character vector used during keyboard input of numbers. The "characters" are sent to a subroutine to be checked and then returned as numbers if no errors are detected.

SELSUB A character variable used to direct the desired subroutine call.

SELOUT A character variable used to direct output to the screen, printer, or both.

SELSOL A character variable used to indicate whether or not a particular tableau will be solved.

SLACK An integer variable used to count the number of slack variables.

TEMP A real variable used to temporarily store values during computation.

TEMPA A real variable used to temporarily store values during computation.

TEMPA(20,20) A matrix of real variables which is used to hold the columns of the B-inverse matrix while the columns of the matrix are being realigned to the identity matrix order.

TEMPCJ(20) A vector of real variables which holds the values of the objective function coefficients which are above the slack and artificial columns while the columns of the B-inverse matrix are being reordered.

UPBD A real variable used to compute the upperbound of an element in the A matrix.

UPBD(20,20) A real matrix used during right-hand-side ranging to compute the upper bound of a column of the A matrix associated with a particular constraint.

ZLP A real variable which holds the Z lower bound value which will be printed.

ZUP A real variable which holds the Z upper bound value which will be printed.
ZLS A real variable which holds the Z lower bound value which will be displayed on the screen.

ZUS A real variable which holds the Z upper bound value which will be displayed on the screen.
VI Program Listings

The following listings are the text files which have been compiled and linked to form the code files of this LP package. Preceding each program and subroutine listing are comments which may assist a programmer in efforts to modify, expand, or translate the Apple FORTRAN source code.

The comment blocks contain several items of importance in a standard format to assist future programmers. The first item listed in each comment block is the module number which the listed program or subroutine is a part. Immediately following is the compilation unit name which contains this program or subroutine. The next line is only present in the comment block of the first listing of each compilation unit. This line lists those compilation units which contain subroutines called by this compilation unit. Next, the name of the program or subroutine which this comment block precedes is shown. The following section is a brief discussion of the program’s or subroutine’s purpose and any special items of interest. The program or subroutines which call this subroutine are listed following the discussion. Next, a listing of those subroutines or programs which may call this subroutine are shown. The last section of the comment block identifies those variables which either influence execution of the program/subroutine or are changed during execution. The first variables listed
with the heading USED are those which may be utilized, but not changed, during execution. The second heading, MODIFIED, lists those variables which may change in value during the execution of the program or subroutine. Only those variables directly used or modified by the programs or subroutines have been shown. Therefore, if Program A calls Subroutine B which changes the value of variable C, only Subroutine B will list variable C as a modified variable. Also, note that all arrays in which the specific array element or elements used or modified may vary due to problem size are annotated with an asterisk ($\ast$). Numeric subscripts are shown only where a specific element or elements are known to be either used or modified during the execution of the program or subroutine.
PROGRAM DATAB

C USE: MAIN PROGRAM OF MODULE 1 LP PACKAGE. PURPOSE OF MODULE IS THE
C ENTRY OF NEW AND EDITING OF EXISTING LP MODELS IN A FORM
C ACCEPTABLE WITH MODULES 2 AND 3. MODULE 1 CONSISTS OF 8
C SEPARATELY COMPILED UNITS (UNIT10 THRU UNIT17) WITH ALL UNITS
C EXCEPT UNIT10 BEING OVERLAY UNITS.
C PROGRAM DATAB ACTS AS AN OUTER COMMAND LEVEL WHICH SOLICITS
C USER INPUT DESIGNATIONS THE OPTION DESIRED. THIS DESIGN
C ALLOWS OVERLAY UNITS TO BE RELEASED FROM MEMORY PRIOR TO NEW
C UNITS BEING CALLED WHICH WOULD OVERLOAD MEMORY.

C CALLED BY: NONE

C CALLS : SUBROUTINE CHECK2(P,N,N,INVAL,NEW)

C SUBROUTINE DATAD
C SUBROUTINE DATAN
C SUBROUTINE DATA
C SUBROUTINE DBE
C SUBROUTINE DMED
C SUBROUTINE DEM
C SUBROUTINE DISPLAY
C SUBROUTINE EDIT
C SUBROUTINE GENIF
C SUBROUTINE HEADER
C SUBROUTINE INTRO
C SUBROUTINE MODUL(INEN)

C VARIABLES:

C USED: IFLAG(2),INVAL
C MODIFIED: IFLAG(5),IFLAG(9),INEN,P(#)

C $USES UCHECK IN UNIT17.CODE OVERLAY
C $USES USAVE IN UNIT16.CODE OVERLAY
C $USES UCNVRCH IN UNIT15.CODE OVERLAY
C $USES UCNVA IN UNIT14.CODE OVERLAY
C $USES UDVAR IN UNIT13.CODE OVERLAY
C $USES UEDIT IN UNIT12.CODE OVERLAY
C $USES UDATAS IN UNIT11.CODE OVERLAY

PROGRAM DATAB
CHARACTER WM,#M,CM#M,FN#M,NM#M,FN#10,PINE#10,OBJ#10
INTEGER V
COMMON/C14/(60,60),C(60),NATG(20),IFLAG10),,NEC,SEC,MC,S,K,V,
COMMON/C2/(M#M,CM#M,FN#M,NM#M,FN#10,P#10),OBJ
OPEN(1,FILE='CONSOLE:')
OPEN(5,FILE='CONSOLE:')
CALL HEADER
100 WRITE(1,'((/3X,"ARE INTRODUCTORY REMARKS DESIRED":"/13X,"(//
',N, RETURN) '/',9)')
READ(5,'(AI)')P(I)
IF(ICHAR(P(I)) .EQ. 89)THEN
   CALL INTRO
ELSEIF(ICHAR(P(I)) .NE. 78)THEN
   WRITE(1,110)
110   Format(5X,'INVALID ENTRY, PLEASE REENTER')
   GO TO 100
ENDIF
INEW=0
C USER SELECTS DESIRED MODULE
   CALL MODUL(INEW)
   CALL DBMED
C DATA BASE ENTRY OPTIONS DISPLAYED
120 CALL DBE
130 WRITE(I,'(/13X,"WHICH OPTION? ",9)')
READ(5,'(AI)')P(I)
   CALL CHECK2(P,1,5,INVAL,INEW)
IF(INVAL .EQ. 1)THEN
   WRITE(I,110)
   GO TO 120
ENDIF
GO TO(140,140,140,150,160)INEW
C ALL VALUES INITIALIZED TO ZERO
140 CALL INIT
   WRITE(I,'(A)')CHAR(12)
IF(INEW .EQ. 1)THEN
C USER HAS SELECTED TO ENTER MODEL WITH SUBSCRIPTS
   IFLAG(5)=0
   CALL GENIF
   CALL DATAS
   GO TO 200
ELSEIF(INEW .EQ. 2)THEN
C USER HAS SELECTED TO ENTER MODEL WITH NAMES
   IFLAG(5)=1
   CALL GENIF
   CALL DATAN
   GO TO 200
ELSE
C USER HAS SELECTED TO READ MODEL FROM DISK
   CALL DATAD
   GO TO 200
ENDIF
150 CALL INTRO
C USER HAS SELECTED TO REVIEW INTRODUCTORY REMARKS
   GO TO 120
160 STOP
C DATA BASE MANAGEMENT OPTIONS DISPLAYED
200 CALL DBM
210 WRITE(I,'(/13X,"WHICH OPTION? ",9)')
READ(5,'(AI)')P(I)
   CALL CHECK2(F,1,5,INVAL,INEW)
IF(INVAL .EQ. 1)THEN

WRITE(1,110)
GO TO 210
ENDIF
WRITE(1,'(A)')CHAR(12)
GO TO 220,240,250,2700NEW
220
IFLAG(9)=0
C INPUT MODEL IS DISPLAYED
CALL DISPLY
GO TO 200
C CONTROL PASSED TO EDITING SUBROUTINE
230
CALL EDIT
GO TO 200
C INPUT MODEL IS SAVED TO DISK
240
CALL SAVE
GO TO 200
C CHECKS TO INSURE MODEL SAVED TO DISK PRIOR TO TERMINATION
250 IF(IFLAG(12).EQ.0) THEN
WRITE(1,260)
260 FORMAT(1(I,15X,'WARNING!! '/6X,'CURRENT FILE WILL BE LOST!!'/
.15X,'CONTINUE? (Y/N) ',A)
READ(5,'(A)')P(I)
IF(ICHAR(P(I)).EQ.89) THEN
IF(INEW.EQ.4) THEN
C USER HAS CHOSEN TO RETURN TO MANAGEMENT MENU
GO TO 120
ELSE
GO TO 280
ENDIF
ELSEIF(ICHAR(P(I)).EQ.78) THEN
GO TO 200
ELSE
WRITE(1,110)
GO TO 250
ENDIF
ENDIF
IF(INEW.EQ.4) THEN
GO TO 120
ELSE
GO TO 280
ENDIF
270 WRITE(1,260)
READ(5,'(A)')P(I)
IF(ICHAR(P(I)).EQ.89) THEN
STOP
ELSE
GO TO 200
ENDIF
280
C EXECUTION MANAGEMENT OPTIONS DISPLAYED
290 CALL DEM
290 WRITE(1,'(1X,'WHICH OPTION? ',A)')
READ(5,'(A)')P(I)
CALL CHECK2(P,1,5,INVAL,INEW)
IF(INVAL .EQ. I) THEN
    WRITE(1,110)
    GO TO 290
ENDIF
WRITE(1,'(A)')CHAR(12)
GO TO (300,300,300,200,160) IMEM
C USER SELECTS NEXT MODULE DESIRED
300 CALL MODUL(IMEN)
STOP
END
**C**

**MODULE** IUNIT1

**UNIT** USES: UNIT12 THRU UNIT17

**SUBROUTINE** CATAS

**USE:** SOLICITS INPUT OF OBJECTIVE FUNCTION AND CONSTRAINT COEFFICIENTS, CONSTRAINT INEQUALITIES AND RHS'S FOR LP MODELS WHICH VARIABLES ARE DESIGNATED BY SUBSCRIPT ONLY. USED ONLY FOR INPUT OF NEW MODELS.

**CALLED BY:** NONE

**CALLS:**
- SUBROUTINE CHECK(P, INVAL, NEW)
- SUBROUTINE CHKCH
- SUBROUTINE OBJCH

**VARIABLES:**
- USES: INEW, INVAL, K, MM, PN, PNEW, V
- MODIFIED: A(k), b(k), c(k), IFLAG(2), IFLAG(3), IFLAG(4), IFLAG(5), IHEB(k), NEC, NCG, MLC, P(k), PINED(k)

**USES**
- USE CHECK IN UNIT17.CODE Overlay
- USE SAVN IN UNIT16.CODE Overlay
- USE UI7CH IN UNIT15.CODE Overlay
- USE UINVA IN UNIT14.CODE Overlay
- USE JADAR IN UNIT13.CODE Overlay
- USE JEDIT IN UNIT12.CODE Overlay

**SUBROUTINE** DATAS

**CHARACTER** VN46, CN16, FN#20, MM#3, FN#10, PINED#1, OBJN#10
**INTEGER** V

**COMMON:** C1/A(20, 60), E/20, C1/60, JHEB/120, IFLAG(10), NEC, NCG, MLC, K, V, NVMN

**COMMON:** C2/UN(60), CN(70), PN, MM, FN, PINED(20), P(10), OBJS

**WRITE**(1, 100) CHAR(12)

100 FORMAT(A)

WRITE(1, 100) CHAR(12)
WRITE(1, 120) PN, MM

120 FORMAT(5X, 'PROBLEM 13: ', A20, /, 3X, 'OBJECTIVE FUNCTION INPUT'/, 4X, AS, ', IMITATION', '/

**OBJECTIVE COEFFICIENTS INPUT BY USER**
DO 140 J = 1, V

140 FORMAT(5X, 'C1', 12) = ', #)
READ(5, 1341) (P(J), J = 1, 10)
CALL CHECK(P, INVAL, PNEW)
IF(INVAL .EQ. 1) THEN
WRITE(1, 150)
222
150 FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
GO TO 150
ELSE
C(J)=RNEW
ENDIF
160 CONTINUE
WRITE(1, '(/))')
PAUSE
ENDIF
190 WRITE(1,100)CHAR(12)
WRITE(1,120)PN,NN
WRITE(1,180)
180 FORMAT(00i
ratings,7X,'ARE CORRECTIONS NEEDED? ',.)
READ('5,'(A11)')P(1)
IF(ICHAR(P(1)) .LT. 89)THEN
IFLAG(B)=0
C CONTROL PASSED TO OBJ FUNCTION EDITING ROUTINE
CALL OBJCH
GO TO 190
ELSEIF(ICHAR(P(1)) .LE. 76)THEN
WRITE(1,150)
GO TO 170
ENDIF
190 WRITE(1,100)CHAR(12)
WRITE(1,'(12X,'"CONSTRAINT INPUT"'/""INPUT CONSTRAINT VARIABLE CO
EFFICIENTS"'/"AS IF THE CONSTRAINT WAS IN THE"'/""FOLLOWING FORM
"'/""A1X,(1) + (2) + (3) <= (RHS)"'")
WRITE(1,'("THE VARIABLE COEFFICIENTS ARE A MAXIMUM"
"OF 10 CHARACTERS."'/""IF COEFFICIENT IS ZERO, ENTER 0 OR HIT"
"RETURN" WITHOUT ENTRY."'")
WRITE(1,'("THE LESS-THAN () REPRESENTS A LESS-THAN"'/""OR EQUAL
INEQUALITY."'/""THE GREATER-THAN () REPRESENTS A"'/""GREATER-TH
AN OR EQUAL INEQUALITY."'/""NEGATIVE RHS IS PERMITTED.""')
PAUSE
WRITE(1,100)CHAR(12)
C CONSTRAINT COEFFICIENTS INPUT BY USER
DO 270 J=1,K
200 WRITE(1,210)PN,J
210 FORMAT(5X,'PROBLEM ID: ',A20,/13X,'CONSTRAINT ',I2,/
DO 240 J=1,Y
220 WRITE(J,230)J
230 FORMAT(I11,'(',I2,') = ',.)
READ('5','(I4A1)')P(L),L=1:10
CALL CHECK(P,INVAL,RNEW)
IF(INVAL .EQ. 1)THEN
WRITE(1,.150)
GO TO 220
ELSE
A(I,J)=RNEW
ENDIF
240 CONTINUE
C CONSTRAINT INEQUALITY INPUT BY USER
250 WRITE(1,'(6X,"INEQUALITY ",I*)')
READ(5,'(A1)')P(I)
CALL CHECK(P,INVAl,INEM)
IF(INVAL .EQ. 1)THEN
  WRITE(1,150)
  GO TO 250
ELSE
  INE(1)=INEM
ENDIF
C COUNT OF EACH TYPE INEQUALITY PERFORMED
260 IF(INEM .EQ. 0)THEN
  NLC=NLC+1
  PIN(1)='<'
ELSEIF(INEM .EQ. 1)THEN
  NEC=NEC+1
  PIN(1)='>'
ELSE
  NEC=NEC+1
  PIN(1)='
ENDIF
C CONSTRAINT RHS INPUT BY USER
270 WRITE(1,'(ISX,"RHS = ",I*)')
READ(5,'(10AI)')/P(L),L=1,10
CALL CHECK(P,INVAl,RNEW)
IF(INVAL .EQ. 1)THEN
  WRITE(1,150)
  GO TO 270
ELSE
  B(1)=RNEW
ENDIF
WRITE(1,100)CHAR(12)
CONTINUE
280 WRITE(1,100)CHAR(12)
WRITE(1,180)
READ(5,'(A1)')P(I)
IF(ICHAR(P(I)) .EQ. 89)THEN
  CALL INPNCH
  GO TO 290
ELSEIF(ICHAR(P(I)) .EQ. 78)THEN
  CALL ICNPCH
  GO TO 280
ENDIF
IFLAG(2)=0
IFLAG(2)=0
RETURN
END
SUBROUTINE DATAN

USE: SOLICITS INPUT OF OBJECTIVE FUNCTION AND CONSTRAINT

COEFFICIENTS, CONSTRAINT INEQUALITIES AND RHS'S FOR LP MODEL

WHICH VARIABLES ARE DESIGNATED BY NAMED VARIABLES,

CONSTRAINTS, AND OBJECTIVE FUNCTION. USED ONLY FOR INPUT OF

NEW MODELS.

CALLED BY: PROGRAM DATAB

CALLS:
SUBROUTINE CHECK(P,INVAL,PNEW)
SUBROUTINE CHECK3(P,INVAL,PNEW)
SUBROUTINE ICNRC
SUBROUTINE OBJCH
SUBROUTINE VNPC

VARIABLES:

USED: INEW,INVAL,K,MM,OBJN,PN,PNEW,V

MODIFIED: A(1),B(1),C(1),CM(1),IFLAG(2),IFLAG(3),IFLAG(4),

IFLAG(6),IFLAG(8),INEG(1),NEC,NGC,NLC,P(1),PINEQ(1),

VM(1)

SUBROUTINE DATAN

CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,PN1*36JN4*10

INTEGER V

COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,X,V,

.MMN

COMMON/C2/VM*60,CN*20,PN,MM,PN,PINEQ(20),P(10),OBJN

WRITE(1,100)CHAR(12)

100 FORMAT(A)

WRITE(1,'(7X,''VARIABLE NAME INPUT''/1X,''ENTER VARIABLE NAME WHICH CORRESPOND''/3X,''TO THE '''/15X,''VARIABLES THAT AFFECT ''''/5X,''''',/5X,''NAMES ARE TO BE 6 CHARACTERS OR LESS.'',/5X,'')') V,

OBJN

PAUSE

WRITE(1,100)CHAR(12)

WRITE(1,120)FN

120 FORMAT(5X,'PROBLEM ID: ''',A20)

WRITE(1,130)

130 FORMAT(16X,'VARIABLE NAME INPUT'/1)

C VARIABLE NAMES INPUT BY USER

DO 140 J=1,V

WRITE(1,'(15X,''X''/I2,'''') ''''/1) J

READ(5,(*A6)*/VN(1))

140 CONTINUE

150 WRITE(1,190)CHAR(12)

WRITE(1,120)FN

WRITE(1,130)

WRITE(1,160)

160 FORMAT(11X,'ARE CORRECTIONS NEEDED? ''',/1X,9X)

READ(5,(*A1)*/P(1))

IF(ICHAR(P(1)) .EQ. 81) THEN

C CONTROL PASSED TO VARIABLE NAME EDITING ROUTINE
CALL VNICH
ELSEIF(ICHAR(P(I)) .NE. 78) THEN
    WRITE(1,170)
FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
PAUSE
GO TO 150
ENDIF
180 WRITE(1,100)CHAR(12)
WRITE(1,'(7X,9X,CONSTRAN NAME INPUT'//',' ENTER CONSTRAINT NAME WHICH CORRESPOND TO THE ',12',' CONSTRAINTS WHICH AFFECT'//',10X,'NAMES ARE TO BE 6 CHARACTERS OR LESS.',A(/))')X,OBJN
PAUSE
WRITE(1,100)CHAR(12)
WRITE(1,120)PN
WRITE(1,190)
190 FORMAT(9X,'CONSTRAN NAME INPUT')
C CONSTRAINT NAMES INPUT BY USER
DO 210 J=1,K
    WRITE(1,'(9X,'CONTRAN ',12,' = ',8)')
    READ(5,'(A6)')C210
210 CONTINUE
WRITE(1,100)CHAR(12)
WRITE(1,120)PN
WRITE(1,190)
WRITE (1,160)
READ(5,'(A1)')P(K)
IF(ICHAR(P(I)) .EQ. 89) THEN
    C FLAG ALLOWS ONLY CONSTRAINT NAME CHANGE TO BE PERFORMED IN
    C EDITING ROUTINE
    IFLAG(6)=4
    CALL ICNRCH
    IFLAG(6)=0
    ELSEIF(ICHAR(P(I)) .NE. 78) THEN
        WRITE(1,170)
        PAUSE
        GO TO 220
ENDIF
230 WRITE(1,100)CHAR(12)
WRITE(1,'(9X,'OBJECTIVE FUNCTION INPUT'//',' INPUT THE FUNCTION AS IF IT WERE IN THE'//',13X,'FOLLOWING FORM'//',5X,'Z = X11 + (X'//',2) + X15 + ETC.'//',10X,'A MAXIMUM OF 10 ENTRIES PER COEFFICIENT'//',13X)')
WRITE(1,'(1X,'INCLUDING DECIMAL AND SIGN ARE ALLOWED.'//',1X,'IF COEFFICIENT IS ZERO, HIT "RETURN"'/10X,'WITHOUT DIGIT ENTRY.'',1X,4(/))')
PAUSE
WRITE(1,100)CHAR(12)
WRITE(1,120)PN
OBJN
WRITE(1,240)OBJN,MM
240 FORMAT(5X,'PROBLEM ',10X,'OBJECTIVE FUNCTION INPUT'/7X,A10..2X,A3,'IMIZATION'/)
DO 280 J=1,U
C OBJECTIVE COEFFICIENTS INPUT BY USER
226
250 WRITE(1,260)J, VN(J)
260 FORMAT(7X,'(I',12,') = ',A6,' = ',S)
READ(5,'(10D1)')(P(I),I=1,10)
CALL CHECK(P,INVAL,RNEW)
IF(INVAL.EQ.1) THEN
   WRITE(1,170)
   GO TO 250
ELSE
   C(J)=RNEW
ENDIF
280 CONTINUE
PAUSE
C MODEL IS IN I=X FORM
IFLAG(4)=0
290 WRITE(1,100)CHAR(12)
WRITE(1,240)PN,OBJN,NN
WRITE(1,140)
READ(5,'(A1)')(P(I)
IF((CHAR(P(I)) .EQ. 0)' THEN
   IFLAG(B)=0
C CONTROL PASSED TO OBJECTIVE FUNCTION EDITING ROUTINE
   CALL OBJCH
ELSEIF((CHAR(P(I)) .NE. 7)' THEN
   WRITE(I,'(170)')
   PAUSE
   GO TO 290
ENDIF
300 WRITE(1,100)CHAR(12)
WRITE(1,'(12X,"CONTRAINT INPUT"/"INPUT CONSTRAINT VARIABLE COEFFICIENTS"/"AS IF THE CONSTRAINT WAS IN THE FOLLOWING FORM /
"/[X(I) + X(2) + X(3) (=) RHS")"
WRITE(1,'(*/"THE VARIABLE COEFFICIENTS ARE A MAXIMUM")
*/"OF n CHARACTERS"/*"IF COEFFICIENT IS ZERO ENTER 0 OR HIT "/"RETURN" WITHOUT ENTRY."
*/"THE LESS-THAN () REPRESENTS A LESS-THAN OR EQUAL INEQUALITY. */"THE GREATER-THAN () REPRESENTS A GREATER-THAN OR EQUAL INEQUALITY. /*"NEGATIVE RHS IS PERMITTED."*/)
PAUSE
WRITE(1,100)CHAR(12)
C CONSTRAINT COEFFICIENTS INPUT BY USER
310 DO 400 I=1,K
330 DO 350 J=1,Y
340 WRITE(I,140)J,W(J)
350 FORMAT(7X,'(I',12,') = ',A6,' = ',S)
READ(5,'(10D1)')(P(L),L=1,10)
CALL CHECK(P,INVAL,RNEW)
IF(INVAL .EQ. 1) THEN
   WRITE(1,170)
   GO TO 330
ELSE

A(1,j)=RNEW
ENDIF

350 CONTINUE
C CONSTRAINT INEQUALITY INPUT BY USER
360 WRITE(1,'(7X,'"INEQUALITY",5X,i)')
READ(5,'(A1)')P(I)
CALL CHECK3(P,INVAL,INEW)
IF(INVAL .EQ. 1) THEN
   WRITE(1,170)
   GO TO 360
ELSE
   INEW(I)=INEM
ENDIF
C COUNT OF EACH TYPE INEQUALITY PERFORMED
IF(INEM .EQ. 0) THEN
   NLC=NLC + 1
   PINEQ(I)=`
ELSE IF(INEM .EQ. 1) THEN
   NGE=NGE + 1
   PINEQ(I)=`
ELSE
   NEC=NEC + 1
   PINEQ(I)=`
ENDIF
C CONSTRAINT RHS INPUT BY USER
370 WRITE(1,'(7X,'"RHS",12X,"=",5X,i)')
READ(5,'(10X?)(P(L),L=1,10)
CALL CHECK(P,INVAL,RNEW)
IF(INVAL .EQ. 1) THEN
   WRITE(1,170)
   GO TO 370
ELSE
   R(I)=RNEW
ENDIF
WRITE(1,'(100)CHAR(12)
400 CONTINUE
410 WRITE(1,190)CHAR(12)
WRITE(1,160)
READ(5,'(A1)')P(I)
IF(ICHAR(P(I)) .EQ. 89) THEN
C CONTROL PASSED TO CONSTRAINT CHANGE ROUTINE
   CALL ICNRCH
ELSEIF(ICHAR(P(I)) .NE. 79) THEN
   WRITE(1,170)
   PAUSE
   GO TO 410
ENDIF
430 IFLAG(3)=0
IFLAG(2)=0
RETURN
END
C SUBROUTINE EDIT
C USE SOLICITS USER INPUT OF THE TYPE CHANGE REQUIRED TO MODEL,
C SETS FLAGS AND CALLS PROPER SUBROUTINE TO PERFORM IMPUTED
C TYPE CHANGE. FLAGS CAUSE ONLY CHANGES REQUESTED TO BE
C INITIALLY ACCESSIBLE TO USER. USED TO CORRECT MOST RECENT
C MODEL INPUT OR EDIT MODEL READ FROM DISK.
C CALLED BY: PROGRAM DATAB
C CALLS : SUBROUTINE ADCON
C SUBROUTINE ADVAR
C SUBROUTINE CNVAR
C SUBROUTINE DELCON
C SUBROUTINE DELVAR
C SUBROUTINE ICNVAR
C SUBROUTINE OBJCH
C SUBROUTINE VNCH
C VARIABLES:
C USED: INEM, INVAL
C MODIFIED: IFLAG(2), IFLAG(6), IFLAG(8), P(*)
C $USES UCHECK IN UNIT17.CODE OVERLAY
C $USES USAVE IN UNIT16.CODE OVERLAY
C $USES UCNVAR IN UNIT15.CODE OVERLAY
C $USES UADVAR IN UNIT14.CODE OVERLAY
C SUBROUTINE EDIT
CHARACTER VM(60), CN(60), PN(20), HM(10), FN(2), OBJN(10)
INTEGER V
COMMON/A1/A20, P1, P2, I1, I2, I3, I4, INEM, INER, INER2, INER3
COMMON/MA/VM(60), CM(20), PM, MM, FN, PINEM(20), P(10), OBJN
100 WRITE(I, 110) CH1M(12)
110 FORMAT(A)
WRITE(I, '/1X, ''DATA BASE EDITOR''/''YOU MAY EDIT THE CURRENT
MOD. EL IN ANY OF THE FOLLOWING MANNERS: ''1. ADD A VARIABLE''/
''2. ADD A CONSTRAINT''/''3. DELETE A VARIABLE''/''4. DELETE A
CONSTRAINT''/''5. CHANGE COEFFICIENT BY CONSTRAINT''/''6. CHANGE CO
EFFICIENTS BY VARIABLE''/''7. CHANGE RHS OF CONSTRAINT''/''8. CHANGE CONSTRAINT
. INEQUALITY''/''9. CHANGE OBJECTIVE FUNCTION COST''/''10. CHANGE MAXIMIZATION/MINIMIZATION''/)
WRITE(I, '/4X, ''CHOICE''/''11. CHANGE VARIABLE NAMES''/''12. CHANGE
CONSTRAINT NAMES''/''13. RETURN TO LAST MENU''/''14. ''DATA BASE MA
WAGEMENT''/)
C USER INPUTS THE TYPE CHANGE DESIRED IN MODEL
120 WRITE(I, '/1X, ''WHICH OPTION? '', #1)
READ(*,2) (P/I, P/I, P/I)
CALL CHECK2(P, 2, INEM, INEM)
229
IF(INVAL .EQ. 1) THEN
  WRITE(1, '(5X, ' 'INVALID ENTRY, PLEASE REENTER')')
  GO TO 120
ENDIF
GO TO (210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330) INEW
C CONTROL IS PASSED TO APPROPRIATE ROUTINE WITH FLAGS DESIGNATING
C TYPE CHANGE ALLOWED TO BE PERFORMED
210 CALL ADVAR
   GO TO 340
220 CALL ADCON
   GO TO 340
230 CALL DELVAR
   GO TO 340
240 CALL DELCON
   GO TO 340
250 IFLAG(6)=1
   CALL ICMPCH
   GO TO 340
260 CALL CNVA
   GO TO 340
270 IFLAG(6)=3
   CALL ICMPCH
   GO TO 340
280 IFLAG(6)=2
   CALL ICMPCH
   GO TO 340
290 IFLAG(8)=1
   CALL OBJCH
   GO TO 340
300 IFLAG(8)=2
   CALL OBJCH
   GO TO 340
310 CALL VMCH
   GO TO 340
320 IFLAG(6)=4
   CALL ICMPCH
   GO TO 340
330 RETURN
C FLAGS RESET TO DEFAULT VALUES
340 IFLAG(2)=0
   IFLAG(6)=0
   IFLAG(8)=0
   GO TO 100
END
SUBROUTINE VNCH
CHARACTER VN#8, CN#8, PN#20, KM#3, FN#10, PINE#1, PS#1, OBJ#10, RES#6
INTEGER V
COMMON/C1/A(20, 60), B(20), C(60), INEQ (20), IFLAG (20), NEC, HEC, MLC, F, V,
          VNUM
COMMON/C2/VN(60), CN(20), N, H, FN, PINE (20), P(10), OBJ
WRITE(1, 100) CHAR(12)
100 FORMAT(A)
  IF(IFLAG (5).EQ. 0) THEN
    WRITE(1, ' (7(i),16X)''THE MODEL BEING EDITED DOES NOT INCLUDE VARIABLE NAMES, ONLY 
           SUBSCRIPTS.' // '' YOU ARE BEING RETURNED TO THE DATA BASE''/''EDITOR''/))
    PAUSE
    RETURN
  ENDIF
110 WRITE(1, 100) CHAR(12)
WRITE(1, ' (11X,15X) ''DO YOU WANT PRESENT NAMES DISPLAYED? '', '*')
READ(5, ' (A1)') IF(1)
  IF(ICHAR(P(1)).EQ. 89) THEN
    WRITE(1, 100) CHAR(12)
    WRITE(1, ' (11X,15X) ''VARIABLE NAMES''/))
    DO 140 J=1, V
      WRITE(1, 120) J, VN(J)
      120 FORMAT(13X, 'X1', 'X1', 'X1', '') = '', .46
      140 CONTINUE
      PAUSE
  ELSEIF(ICHAR(P(1)).NE. 78) THEN
    WRITE(1, 160)
    160 FORMAT(5X, 'INVALID ENTRY, PLEASE REENTER')
    GO TO 110
  ENDF
180 WRITE(1, 100) CHAR(12)
190 WRITE(1, ' (6X) ''WHICH VARIABLE NAME IS TO BE CHANGED?'', /, 16X, 'USE 
           SUBSCRIPT VALUE.' ')')
  READ(5, ' (A1) ') P(1), P(2)
  CALL CHECK2(P, 2, V, INVAL, INEW)
  IF(INVAL .EQ. 1) THEN
WRITE(1,160)
GO TO 200
ENDIF
WRITE(1,'(A/10X,'PRESENT',7X,'DESIRED'/u,NAME','W',E)
WRITE('i,10X,46,BX,$)')VN(INEM)
READ(5,'(A6))RES
C USER GIVEN OPTION TO DELETE REQUESTED CHANGE
220 WRITE(1,'(A/8I,,'IS CHANGE STILL DESIRED? ''$,#)')
READ(5, '(A1)')P(I)
IF(ICHAR(P(I)) .EQ. 89) THEN
  VN(INEM)=RES
  WRITE(1,'(A/11X,'I CHANGE COMPLETED''))
  PAUSE
ELSEIF(ICHAR(P(I)) .EQ. 78) THEN
  WRITE(1,'(A/11X,'NO CHANGES PERFORMED''))
  PAUSE
ELSE
  WRITE(1,160)
  GO TO 220
ENDIF
230 WRITE(1,100)ICHAR(12)
WRITE(1,'(A11/),IX,'FURTHER VARIABLE NAME CHANGES NEEDED?'/19X, 
$,#)')
READ(5, '(A1)')P(I)
IF(ICHAR(P(I)) .EQ. 89) THEN
  WRITE(1,100)ICHAR(12)
  GO TO 110
ELSEIF(ICHAR(P(I)) .EQ. 78) THEN
  IFLAG(2)=0
  RETURN
ELSE
  WRITE(1,160)
  GO TO 230
ENDIF
END
SUBROUTINE ADVAR

C USE: PERFORMS THE ADDITION OF A VARIABLE TO THE MODEL BY
C SOLICITING USER INPUTS FOR VARIABLE COEFFICIENTS IN ALL
C CONSTRAINTS. MODIFIES NECESSARY VARIABLES TO REFLECT
C ADDITION OF VARIABLE TO MODEL AND REORGANIZES DATA IN ARRAYS.
C CALLED BY: SUBROUTINE EDIT
C CALLS : SUBROUTINE CHECK(P,INVAL,PMNEW)
C SUBROUTINE CHECK2(P,M,INVAL,INNEW)
C SUBROUTINE CINV
C SUBROUTINE OBJCH
C VARIABLES:
C USE: CN(9),NFLG(5),INNEW,IVAL,X,RNEW
C MODIFIED: A(9,9),CN(2),NFLG(2),P(I),V,VM(I)

C USES UCHECK IN UNIT17.COEFILE
C USES UCINV IN UNIT14.COFILE

SUBROUTINE ADVAR
CHARACTER VN66.,CN66.,PN20,MN10,FN10,PINE10,PN100
INTEGER V
COMMON/C1/A(20,60),C(20),C(i60),NFLG(20),IVAL,INNEW,ICNC
COMMON/C2/CN(160),CN(120),PN,MM,FN,PINE(20),P(10),DBJN
100 WRITE(1,110)CHAR(12)
110 FORMAT(12)
C NUMBER OF VARIABLES INCREASED BY 1
V=N+1
WRITE(1,130)PN
:30 FORMAT(5X,'PROBLEM : ','A20')
C DETERMINES IF MODEL CONTAINS NAMES
IF(NFLG(5).EQ.0)THEN
C MODEL DOES CONTAIN NAMES
WRITE(1,'(10x,'"VARIABLE NAME INPUT"'//"ENTER VARIABLE NAME WHICH"
.CORESponds"'//"TO VARIABLE X("',I2,'".""""NAMES ARE TO BE 6"
.CHARACTERS OR LESS."""')")V
WRITE(1,'(10x,"VARIABLE X('',I2,"') = "',A9)')V
READ(5,'(A6)')NV
WRITE(1,110)CHAR(12)
WRITE(1,130)PN
WRITE(1,'(7x,"VARIABLE COEFFICIENT INPUT"'//""VARIABLE X('',1"
.,2,"') = "',A6,'"')')V,VM(V)
C VARIABLE COEFFICIENT INPUT FOR EACH CONSTRAINT
DO 200 I=1,K
140 FORMAT('CONSTRAINT ',I2,' = ',A6)'
200 READ(5,'(10A1)')(P(L),L=1,10)
C CALL CHECK(P,INVAL,NEW)
IF(INVALID .EQ. 1)THEN
WRITE(1,170)
170  FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
   GO TO 140
   ELSE
   A(I,V)=RNEW
   ENDIF
200  CONTINUE
   PAUSE
   ELSE
C MODEL DOES NOT CONTAIN NAMES
   WRITE(1,'(7X,'"VARIABLE COEFFICIENT INPUT"/13X,"VARIABLE X(" ,
   ,I2,'"","/"')')')V
   DO 290 I=1,K
210  WRITE(1,2201)
220  FORMAT(6X,'CONSTRAINT #',I2,' = ',A)
   READ(5,'(10A1)') (PL(I),L=1,10)
   CALL CHECK(P,INVAL,RNEW)
   IF(INVAL.EQ.1) THEN
   WRITE(1,170)
   GO TO 210
   ELSE
   A(I,V)=RNEW
   ENDIF
250  CONTINUE
   PAUSE
   ENDIF
260  WRITE(1,110)CHAR(12)
   WRITE(1,130)PM
C VARIABLE COST COEFFICIENT INPUT BY USER
270  IF(IFLAGS.EQ.1) THEN
   WRITE(1,'(7X,"OBJECTIVE COEFFICIENT INPUT"/13X,"VARIABLE X(" ,
   ,I2,'",")")')V
   WRITE(1,280)V
   ' READ(5,'(10A1)') (PL(I),L=1,10)
   ENDIF
   CALL CHECK(P,INVAL,RNEW)
   IF(INVAL.EQ.1) THEN
   WRITE(1,170)
   GO TO 270
   ENDIF
C(V)=RNEW
290  WRITE(1,110)CHAR(12)
   WRITE(1,360)
300  FORMAT(/11X,'APE CORRECTIONS NEEDED ON THIS VARIABLE?'/19X,A)
   READ(5,'(A1)')PL(I)
   IF(ICHAR(PL(I)).EQ.8) THEN
   WRITE(1,110)CHAR(12)
WRITE(1,'(1x,'CHANGES REQUIRED IN:')//5x,'1. OBJECTIVE COST COEFFICIENT'//5x,'OR'/5x,'2. CONSTRAINT COEFFICIENT')')
320 WRITE(1,'(1x,'WHICH OPTION? ','&$')
READ(5,'(A1)')P(1)
CALL CHECK2(P,1,2,INVAL,INUE)
IF(INVAL.EQ.1)THEN
   WRITE(1,170)
   GO TO 320
ENDIF
IF(INUE.EQ.1)THEN
   USER ELECTS TO CHANGE COST COEFFICIENTS
   CALL OBJCH
   ELSE
   USER ELECTS TO CHANGE CONSTRAINT COEFFICIENT
   CALL CNWVA
   ENDIF
ELSEIF(INCHAR(P(1)) .NE. 78)THEN
   WRITE(1,170)
   GO TO 290
ENDIF
330 WRITE(1,110)INCHAR(12)
340 WRITE(1,'(1x,5x,8x,'ADD ANOTHER VARIABLE? ','&$')
READ(5,'(A1)')P(1)
IF(INCHAR(P(1)) .EQ. 89)THEN
   USER HAS SELECTED TO ADD ANOTHER VARIABLE
   GO TO 100
ELSEIF(INCHAR(P(1)) .NE. 78)THEN
   WRITE(1,170)
   GO TO 330
ENDIF
IF(FLAGS2) .EQ. 0
RETURN
END
SUBROUTINE OBJCH

USE: PERFORMS USER INPUT CHANGES TO OBJECTIVE FUNCTION

COEFFICIENTS AND MAXIMIZATION/MINIMIZATION CHOICE OF THE
CURRENT MODEL. USED IN CORRECTION OF MOST RECENT MODEL INPUT
OF MODEL FROM DISK BEING EDITED.

CALLED BY: SUBROUTINE ADVAR
SUBROUTINE DATAN
SUBROUTINE Datas
SUBROUTINE EDIT

CALLS : SUBROUTINE CHECK(P,INVAL,RNEW)
SUBROUTINE CHECK2(P,N,INVAL,INEN)

VARIABLES:

USED: IFLAG(5),INEN,INVAL,OBJN,PM,RNEW,V,VN(8)
MODIFIED: C(8),IFLAG(2),IFLAI(8),MM,MNN,F(1)

SUBROUTINE OBJCH

CHARACTER VN(8),CM(8),PN(20),MN,S,F(10),OBJN
INTEGER V
COMMON/C1/A(200),B(20),C(20),INEG(20),IFLAG(10),NEC,NEC,HNC,K,N
.MM
COMMON/C2/VM(60),CM(20),PM,MM,F,PINE(20),P(10),OBJN

WRITE (1,110) CHAR(12)
110 FORMAT(A)

IF(FLAG(B) .LT. 1) 120, 200, 400

120 IF(FLAG(S) .EQ. 0) THEN
WRITE (1,130) PM, MM
130 FORMAT(5X,'PROBLEM ID: ',A20,' OBJECTIVE FUNCTION CHANGE'/14X,
,A3,'IMIZATION'/)
ELSE
WRITE (1,140) PM, OBJN
140 FORMAT(5X,'PROBLEM ID: ',A20,'OBJECTIVE FUNCTION CHANGE'/7X,A
,10,2X,3X,'IMIZATION'/)
ENDIF
WRITE(1,1/2X,'WHICH OF THE BELOW REQUIRES CHANGES?'/4X,'1. C
.OEFFICIENTS'/19X,''2. MAXIMIZATION/MINIMIZATION CH
.OICE''/)
150 WRITE(1,1/13X,'WHICH OPTION? '?'.#')
READ(S,'(A)') P(1)
CALL CHECK2(P,1,2,INVAL,INEN)
IF(INVAL ,EQ. 11) THEN
WRITE(1,160)
160 FORMAT(5X,'INVALID ENTRY, PLEASE REENTER')
50 TO 150
ENDIF
WRITE(1,110) CHAR(12)
GOTO(200,400,1NEW)

ACCESS TO CHANGE OBJ FUNCTION COEFFICIENTS ONLY

200 IF(FLAG(S) .EQ. 0) THEN

236
WRITE(1,130)PN,MM
ELSE
WRITE(1,140)PN,OBJN,MM
ENDIF
WRITE(1,'(/3X,5HDISPLAY THE PRESENT COEFFICIENTS? ","))
READ(S,'(A1)')P(1)
IF(ICHAR(P(1)).EQ.89)THEN
WRITE(1,110)CHAR(12)
ELSE IFLAG(S).EQ.0)THEN
WRITE(1,130)PN,NN
ELSE
WRITE(1,140)PN,OBJN,MM
ENDIF
GO TO 210
DO
J=1,
IF(IFLAG(S).EQ.0)THEN
WRITE(1,'(/3X,4HWHICH COEFFICIENT IS TO BE CHANGED? ","/6X,4PLEASE ENTER SUBSCRIPT VALUE."'))
WRITE(S,'(A1)')P(1),P(2)
CALL CHECK2(P,INVAL,NEW)
IF(INVAL.EQ.0)THEN
WRITE(1,160)
GO TO 270
ENDIF
WRITE(1,'(/3X,5HIS CHANGE STILL DESIRED? ","))
READ(5,'(A1)'P(I)
IF(ICHAR(P(I)) .EQ. 89)THEN
C(1NEW)=PNEW
WRITE(1,290)
290 FORMAT(1X,'1 CHANGE COMPLETED')
PAUSE
ELSEIF(ICHAR(P(I)) .EQ. 78)THEN
WRITE(1,300)
300 FORMAT(1X,'NO CHANGES PERFORMED')
PAUSE
ELSE
WRITE(1,190)
GO TO 280
ENDIF
310 WRITE(1,110)CHAR(2)
WRITE(1,'(11/I)''FURTHER COST COEFFICIENT CHANGES? ",")
READ(5,'(A1)'P(I)
IF(ICHAR(P(I)) .EQ. 89)THEN
WRITE(1,110)CHAR(12)
GO TO 250
ELSEIF(ICHAR(P(I)) .NE. 78)THEN
WRITE(1,160)
GO TO 310
ENDIF
330 WRITE(1,110)CHAR(12)
340 WRITE(1,350)
350 FORMAT(11/I)''FURTHER OBJECTIVE FUNCTION CHANGES? ",")
READ(5,'(A1)'P(I)
IF(ICHAR(P(I)) .EQ. 89)THEN
WRITE(1,110)CHAR(12)
GO TO 120
ELSEIF(ICHAR(P(I)) .NE. 78)THEN
WRITE(1,160)
GO TO 330
ENDIF
GO TO 500
C ACCESS TO CHANGE MAX/MIN CHOICE
400 WRITE(1,110)CHAR(12)
IF(IFLAG(5) .EQ. 0)THEN
WRITE(1,150)PN,MM
ELSE
WRITE(1,140)PN,OBJN,MM
ENDIF
WRITE(1,'(/I5,'''PRESENT CHOICE IS ''',A3,'''IMIZATION.'')''MM
WRITE(1,'(/I5,'''WOULD YOU LIKE THIS CHANGED? ''',A3)'
READ(5,'(A1)'P(I)
IF(ICHAR(P(I)) .EQ. 99)THEN
IF(MINN .EQ. 0)THEN
MINN=0
MM='''MIN'
ELSE
MINN=1
ENDIF
ENDIF
ENDIF
ENDIF
239
ENDIF
WRITE(1,290)
ELSEIF(CHAR(P(1)) .EQ. 'B') THEN
  WRITE(1,300)
ELSE
  WRITE(1,160)
  GO TO 420
ENDIF
WRITE(1,'(A/))')
PAUSE
GO TO 330
500  IFLAG(2)=0
      IFLAG(8)=0
RETURN
END
SUBROUTINE SENIF

CHARACTER VNI6, CNI6, PNS20, i0l3, FNti0, PjNEI, PII, OBJN
INTEGER V
COMMON/C1/A(20, 60), B(20), C(60), INEG(20), IFLAS(10), NEC, NLC, V, V.

COMMON/C2/VN(40), CN(20), PN, VM, FN, PINE(20), P(100), OBJN

PROBLEM NAME ENTERED FOR NEW MODEL

WRITE(1, '(/7X,''ENTER A PROBLEM IDENTIFIER''/7X,''(MAXIMUM OF 20 CHARACTERS)''//3X,''PROBLEM ID = '',''N'))
READ(5, '(A20)') P

MAX/MIN CHOICE ENTERED

WRITE(1, '(/40(*''))')
READ(5, '(A1)') P(1)
CALL CHECK2(P, 1, INVAL, INEW)
IF(INVAL .EQ. 2) THEN
WRITE(1, '(/120)')
GO TO 110
ENDIF

MVMN = INEW

CHARACTER STRING ESTABLISHED REPRESENTING MAX/MIN CHOICE

IF(MVMN .EQ. 1) THEN
MM = 'MAX'
ELSE
MM = 'MIN'
ENDIF

IF(IFLAS(5) .EQ. 1) THEN
MODEL INCLUDES NAMES OF OBJ FUNCTION NAME INPUT

WRITE(1, '(/12') CHAR(12)
WRITE(1, '(/17/),1X,''WHAT IS THE NAME OF THE OBJECTIVE YOU WANT TO ''/1X,''(FOR EXAMPLE, COST, MANPOWER, ETC.)''/1X,''(MAXIMUM OF 10 CHARACTERS ALLOWED)''/1X,''OBJECTIVE NAME = '',''N')
READ(5, '(A10)') OBJN
ENDIF

WRITE(1, '(/A)') CHAR(12)
C NUMBER OF CONSTRAINTS ENTERED
WRITE(1,'(////I,'''ENTER NUMBER OF CONSTRAINTS IN PROBLEM''/13x,,"MAXIMUM OF 20")')
30 WRITE(1,'(////8X,"NUMBER OF CONSTRAINTS = '',#')
READ(5,'(2AI)')P(1),P(2)
CALL CHECK2(P,2,20,INVAL,INEW)
IF(INVAL .EQ. 1) THEN
   WRITE(1,120)
   GO TO 130
ENDIF
K=INEW
WRITE(1,'(////40("*")))')
C NUMBER OF VARIABLES ENTERED
WRITE(1,'(////2T,"ENTER NUMBER OF VARIABLES IN PROBLEM''/13x,"MAXIMUM OF 20")')
140 WRITE(1,'(////94,"NUMBER OF VARIABLES = '',#")
READ(5,'(2AI)')P(1),P(2)
CALL CHECK2(P,2,20,INVAL,INEW)
IF(INVAL .EQ. 1) THEN
   WRITE(1,120)
   GO TO 140
ENDIF
V=INEW
WRITE(1,'(A)')CHAR(12)
RETURN
END
SUBROUTINE CNVA

USE: ALLOWS MODEL COEFFICIENTS TO BE CHANGED BY VARIABLE.
SOLICITS INPUT OF VARIABLE COEFFICIENT TO BE CHANGED BY
INPUTING VARIABLE NUMBER. VARIABLE COEFFICIENT OF SPECIFIED
VARIABLE MAY BE CHANGED BY DESIGNATING CONSTRAINT NUMBER AND
THE DESIRED COEFFICIENT.

CALLED BY: SUBROUTINE ADVAR

CALLS : SUBROUTINE CHECK2(P, INVAL, RNEW)
SUBROUTINE CHECK2(P, K, N, INVAL, INEW)

VARIABLES:

USED: A(1,10), CN(10), IFLAG(5), INEW, INVAL, K, RNEW, V, VN(10)

MODIFIED: CN1, CHO, CN(20), P(20)

USES: SUBROUTINE CHECK IN UNIT17.CODE

OVERLAYS SUBROUTINE

CHARACTER VN(10), CN(10), P(20,1), IFLAG(5), IN(10), OB(1)
INTEGER V, CN, P(100), N, I
COMMON/C1A(20,60),B(60),C(60),IN(10),IFLAG(5),NEC, NVC
,VMCN
COMMON/C2/VN(10),CN(10),P(20),IN(10),OB(1)

100 WRITE(1,110)CHAR(12)
110 FORMAT(A)
WRITE(1, '(''WHICH VARIABLE COEFFICIENTS REQUIRE CHANGE?)'')
,75X, 'PLEASE ENTER SUBSCRIPT VALUE. ')
130 WRITE(1, '(''VARIABLE X('') = ''A('')'')
READ(5, )A(I), A(2)
CALL CHECK2(P,2, INVAL, INEW)
IF(INVAL .EQ. 1) THEN
WRITE(1, 140)
140 FORMAT('INVAL ENTRY, PLEASE REENTER')
GO TO 130
ENDIF
150 WRITE(1,110)CHAR(12)
WRITE(1, '(''VARIABLE COEFFICIENT BY CONSTRAINT')''),
,75X,'''
IF(IFLAG(5) .EQ. 1) THEN
WRITE(1, '(''2X,'''VARIABLE COEFFICIENT BY CONSTRAINT')'')
170 DO 160 I=1,K
WRITE(1,110)CHAR(12)
,75X,'''
WRITE(1, '(''2X,'''VARIABLE X('') = ''A('')'')
180 CONTINUE
PAUSE
ELSE
WRITE(1, '(''2X,'''VARIABLE COEFFICIENT BY CONSTRAINT')'')
DO 180 I=1,K
WRITE(1, '(''2X,'''VARIABLE X('') = ''A('')'')
190 CONTINUE
PAUSE
ENDIF
190 WRITE(1,110)CHAR(12)
WRITE(1,'(10(I/),''WHICH CONSTRAINT REQUIRES A CHANGE IN''/9X,'''THE
. IX'',12,''' ) COEFFICIENT?''/4X,'''PLEASE ENTER CONSTRAINT NUMBER.'''
.))CHAR
200 WRITE(1,'(/5,'''CONSTRAINT NUMBER = '',#)')
READ(5,'(2AI)')P(1),P(2)
CALL CHECK2(P,2,K,INVAL,NEW)
IF(INVAL .EQ. 1)THEN
  WRITE(1,140)
  GO TO 200
ENDIF
CHAR=NEW
210 WRITE(1,'(/6X,''PRESENT'',4X,''DESIRED'',7X,'''X''',12,'''),16X,
.'''X'',12,'''))CHAR,CHAR
WRITE(1,'(/3X,1PE12.5X,4B1)')A(NEW),CHAR
READ(5,'(10AI)')(P(L),L=1,10)
CALL CHECK(P,INVAL,NEW)
IF(INVAL .EQ. 1)THEN
  WRITE(1,140)
  GO TO 210
ENDIF
220 WRITE(1,'(/5,'''IS CHANGE STILL DESIRED? '',#)')
READ(5,'(4AI)')P(1)
IF(ICHRAT(P(1)) .EQ. 89)THEN
  A(CHRAT,CHRAT)=NEW
  WRITE(1,'(/11X,''I CHANGE COMPLETED''))
  PAUSE
ELSEIF(ICHRAT(P(1))) .EQ. 78)THEN
  WRITE(1,'(/10X,''NO CHANGES PERFORMED''))
  PAUSE
ELSE
  WRITE(1,140)
  GO TO 220
ENDIF
230 WRITE(1,110)CHAR(12)
WRITE(1,'(10(I/),''FURTHER COEFFICIENT CHANGES OF SAME''/''VARIAB
. E IN DIFFERENT CONSTRAINT?''/19X,#)')
READ(5,'(4AI)')P(1)
IF(ICHRAT(P(1)) .EQ. 89)THEN
  GO TO 190
ELSEIF(ICHRAT(P(1)) .EQ. 78)THEN
  IFLAG(2)=0
  RETURN
ELSE
  WRITE(1,140)
  GO TO 230
ENDIF
240 WRITE(1,110)CHAR(12)
WRITE(1,'(10(I/),''FURTHER COEFFICIENT CHANGES OF DIFFERENT''/''VAR
. IABLE?''/19X,#)')
READ(5,'(A1)'P(1)
IF(I$CHAR(P(1)).EQ. 89)THEN
    GO TO 100
ELSEIF(I$CHAR(P(1)).EQ. 78)THEN
    IFLAG(2)=0
    RETURN
ELSE
    WRITE(1,140)
    GO TO 240
ENDIF
END

SUBROUTINE DELCON

USE: Performs the deletion of a constraint from model by soliciting user input of constraint number. Modifies necessary variables to reflect deletion of constraint and reorganizes data in arrays.

CALLED BY: SUBROUTINE EDIT

CALLS: SUBROUTINE CHECK2(P,N,M,IVAL,INEW)

VARIABLES:

USED: IFLAG(I), INEW, INVAL

MODIFIED: A(I,N), B(I,N), CN(I), IFLAG(I), INED(I), K, NEC, MGC, MLC, P(I)

SUBROUTINE DELCON

CHARACTER VN(6), CN(6), PN(20), M(6), P(10), PINEQ(20), PINEG(10), IFLAG(10), B(20), C(60), INED(10), K, NEC, MGC, MLC, V

COMMON/C1/A(20), B(20), C(60), INE(10), IFLAG(10), NEC, MGC, MLC, K, V

COMMON/C2/VN(60), CN(20), PN(20), M(20), PINEQ(20), PINEQ(10), INED(10), K, NEC, MGC, MLC, V

WRITE(1, 110) CHAR(12)

110 FORMAT(A)

IF(FLAG(5) .EQ. 1) THEN

WRITE(1, ' (11) /','NEED TO SEE CONSTRAINT NAME LIST? ', ')

READ(S, ' (A1) ' ) P(1)

IF(ICHAR(P(1)) .EQ. 89) THEN

WRITE(1, 110) CHAR(12)

WRITE(1, ' (6X, ' CONSTRAINT NAME LISTING ')' )

DO 150 I = 1, K

WRITE(1, ' (7X, ' CONSTRAINT ', I2, ' = ', A6 ) ) I, CN(I)

150 CONTINUE

RETURN

ELSEIF(ICHAR(P(1)) .NE. 78) THEN

WRITE(1, 160)

160 FORMAT(/5X, 'INVALID ENTRY, PLEASE REENTER'

ENDIF

ENDIF

WRITE(1, 110) CHAR(12)

WRITE(1, ' (10) /','WHICH CONSTRAINT DO YOU WISH TO DELETE?' '/5X, ')

READ(S, ' (2A1) ' ) P(1), P(2)

CALL CHECK2(P, 2, K, INED, INEW)

IF(INED .EQ. 1) THEN

WRITE(1, 160)

ENDIF

B(INEW) = 0

DO 220 J = 1, V

A(INEW, J) = 0

220
CONTINUE
C COUNT BY TYPE OF INEQUALITY UPDATED SINCE I LESS CONSTRAINT
IF(INEQ(INEM).EQ.0)THEN
   MLC=MLC - 1
ELSEIF(INEQ(INEM).EQ.1)THEN
   NEC=NEC - 1
ELSE
   NEC=NEC - 1
ENDIF
C IF CONSTRAINT NOT LAST CONSTRAINT, ALL ROWS MOVED UP I
IF(INEM.LT.N)THEN
   DO 300 I=INEM,K-1
      B(I)=B(I+1)
      INEQ(I)=INEQ(I+1)
      PINER(I)=PINER(I+1)
      CN(I)=CN(I+1)
      DO 290 J=1,V
         A(I,J)=A(I+1,J)
   290 CONTINUE
   300 CONTINUE
ENDIF
C NUMBER OF CONSTRAINT DECREASED BY I
Y=K-1
IFLAG(2)=0
RETURN
END
SUBROUTINE DELVAR

USE: PERFORMS THE DELETION OF A VARIABLE FROM MODEL BY SOLICITING USER INPUT OF VARIABLE NUMBER. MODIFIES NECESSARY VARIABLES TO REFLECT VARIABLE DELETION AND REORGANIZES DATA IN ARRAYS.

CALLED BY: SUBROUTINE EDIT

CALLS : SUBROUTINE CHECK2(P,N,NVAL,INEW)

VARIABLES:

USED: IFLAG(5), INVAL, INEW, K
MODIFIED: A(*), A(I), IFLAG(2), P(*), V, VN(*)

SUBROUTINE DELVAR
CHARACTER VN(*), CN(*), PN(*), MN(*), FM(*), PINED(*), IDNV(*), JB(*), ONE(100), X worries, V}
INTEGER V
COMMON/C1/A(20,60), B(20), C(60), INED(20), IFLAG(10), NEC, N6C, NLC, K, V,
COMMON/C2/VN(60), CM(20), PM, MM, FK, PINED(20), P(10), JB
100 WRITE(1,110)CHAR(12)
110 FORMAT(A)
IF(IFLAG(5), .EQ. 1)THEN
MODEL CONTAINS NAMES
120 WRITE(1, '(11(/)','NEED TO SEE VARIABLE NAME LISTING? ', *)')
READ(5, '(A1)') P(1)
IF(ICHAR(P(1)) .EQ. 99)THEN
WRITE(1,110)CHAR(12)
WRITE(1, '9X,'"VARIABLE NAME LISTING"')
DO 150 J=1, V
WRITE(1, 19X,'"VARIABLE X(", I2, ") = ", A6") J, VN(J)
150 CONTINUE
PAUSE
ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
WRITE(1, 160)
160 FORMAT(5X, "INVALID ENTRY, PLEASE REENTER")
GO TO 100
ENDIF
ENDDO
WRITE(1,110)CHAR(12)
WRITE(1, '(11(/)," WHICH VARIABLE DO YOU WISH TO DELETE? "/5X,"PLEASE ENTER SUBSCRIPT VALUE:" ")
220 WRITE(1, '(9X, "DELETE VARIABLE X(?) = ", *)')
READ(5, '(2A1)') P(1), P(2)
CALL CHECK2(P, N, INVAL, INEW)
IF(INVAL .EQ. 1)THEN
WRITE(1,160)
GO TO 220
ENDIF
C COST COEFFICIENT FOR DELETED VARIABLE ZERODED
C(INEW)=0.
C CONSTRAINT COEFFICIENTS FOR DELETED VARIABLE ZERODED
DO 250 I=1,K
247
A(K,INEM)=0.
250 CONTINUE
C IF VARIABLE NOT LAST VARIABLE, ALL COLUMNS MOVED LEFT I
   IF(INEM.LT.9) THEN
      DO 300 J=INEM, V-1
         C(J)=C(J+1)
         IF(IFLAB(5).EQ.1) THEN
            VN(J)=VN(J+1)
         ENDIF
      DO 290 I=1,K
         A(I,J)=A(I,J+1)
      290 CONTINUE
   300 CONTINUE
C NUMBER OF VARIABLES DECREASED BY 1
   V=V-1
320 WRITE(1,110)CHAR(12)
   WRITE(1,'(10(/,5x,"DELETE ANOTHER VARIABLE? ",A))')
   READ(5,' (A1)') P(I)
   IF(ICHAR(P(I)).EQ.99) THEN
      GO TO 100
   ELSEIF(ICHAR(P(I)).NE.78) THEN
      WRITE(1,160)
      GO TO 320
   ENDIF
   IFLAG(2)=0
RETURN
END
MODULE 1 UNIT 15

UNIT USES: UNIT17

SUBROUTINE ICNCH

USE: SOLICITS INPUT OF CONSTRAINT NUMBER AND TYPE OF CHANGE DESIRED IN CONSTRAINT. ALLOWS USER TO CHANGE CONSTRAINT COEFFICIENT, INEQUALITY, RHS OR NAME. PERFORMS DESIRED CHANGE AND MODIFIES THOSE VARIABLES WHICH THE CHANGE NECESSITATES.

CALLED BY: SUBROUTINE ACON
SUBROUTINE DATAN
SUBROUTINE DATAS
SUBROUTINE EDIT

CALLS : SUBROUTINE CHECK(P,INVAL,RHSM)
SUBROUTINE CHECK2(P,M,M,INVAL,INEM)
SUBROUTINE CHECK3(P,INVAL,INEM)
SUBROUTINE DISPLY

VARIABLES:

USED: IFLAG(5),INEM,INVAL,K,RHSM,V
MODIFIED: A(1,1),B(1),CHAK,CHAN,CHRND,CM(1),IFLAG(6),
 IFLAG(9),INEQ(8),NEC,NGC,MLC,P(4),P5,NE6(4)

USES UCHECK IN UNIT17.CODE OVERLAY

SUBROUTINE ICNCH
CHARACTER VNAME,CMAN,PMAN,MMAN,FMN,INEM,PINEQ,INP,OBIN,M,CMAN
INTEGER V,CHAN,CHAR
COMMON/A(20,50),B(20),C(160),INEQ(20),IFLAG(10),NEC,NGC,MLC,K,V,
 .MMN
COMMON/C1,A20,V50),CMN(160),FMN,MMN,PINEM(20),P(10),OB3M

100 WRITE(1,110)CHAR(12)
110 FORMAT(A)
120 FORMAT(11X,'CONSTRAINT CHANGE')
C CHECKS IF ONLY CONSTRAINT NAME IS TO BE CHANGED
IF(IFLAG(6) .EQ. 4) THEN
  GO TO 140
ENDIF
WRITE(1,'(10(2,iX,'''DISPLAY THE PRESENT CONSTRAINTS? ''',4)')
READS,'(A81)')P(I)
IF(ICHAD(P(I)) .EQ. 78) THEN
  GO TO 140
ELSEIF(ICHAD(P(I)) .NE. 89) THEN
  WRITE(1,130)
130 FORMAT(5X,'INVALID ENTRY, PLEASE REENTER')
  GO TO 100
ENDIF
C FLAG ALLOWS ONLY CONSTRAINT TO BE DISPLAYED
IFLAG(9)=1
CALL DISPLY
WRITE(1,110)CHAR(12)
140 WRITE(1,'(14X,',''WHICH CONSTRAINT DO YOU WISH TO CHANGE?''/5X,'')
LEASE ENTER CONSTRAINT NUMBER.

WRITE(1,'(/9X,''CHANGE CONSTRAINT 0'',I)')
READ(5,'(2AI1)')P(I),P(2)
CALL CHECK2(P,1,INVAL,INEW)
IF(INVAL .EQ. 1)THEN
  WRITE(1,130)
  60 TO 150
ENDIF
WRITE(1,110)CHAR(12)
120 FORMAT(1X,'CONRAINTE CHANGE')
WRITE(1,'(/2X,''CHANGE DESIRED IN CONSTRAINT 0'',I2,','' IS: ''/9X,"'1. VARIABLE COEFFICIENT''/BI,''2. INEQUALITY''/BI,''3. RMS''/5X,"'4. NO CHANGES''))CHAR
WRITE(1,'(/13X,''WHICH OPTION? ''$,#)')
READ(5,'(AI)')P(1)
CALL CHECK2(P,1,INVAL,INEW)
IF(INVAL .EQ. 1)THEN
  WRITE(1,130)
ENDIF
GOTO(200,350,500,650,800)(IFLNUM(6)+1)

C ACCESS TO CHANGE ANY PART OF CONSTRAINT ALLOWED
WRITE(1,110)CHAR(12)
WRITE(1,210)
210 FORMAT(1X,''CONRAINTE CHANGE'')
WRITE(1,'(/2X,''CHANGE DESIRED IN CONSTRAINT 0'',I2,','' IS: ''/9X,"'1. VARIABLE COEFFICIENT''/BI,''2. INEQUALITY''/BI,''3. RMS''/5X,"'4. NO CHANGES''))CHAR
WRITE(1,'(/13X,''WHICH OPTION? ''$,#)')
READ(5,'(AI)')P(1)
CALL CHECK2(P,1,INVAL,INEW)
IF(INVAL .EQ. 1)THEN
  WRITE(1,130)
ENDIF
GOTO(200,350,500,650,800)(IFLNUM(6)+1)

C ONLY VARIABLE COEFFICIENTS ALLOWED TO BE CHANGED
WRITE(1,110)CHAR(12)
WRITE(1,'(/5X,**WHICH VARIABLE COEFFICIENT OF''/4X,**CONRAINTE '' ,I2,** REQUIRES CHANGES?''))CHAR
WRITE(1,'(/10X,**VARIABLE X(? = '',#,2(/)))')
READ(5,'(2AI1)')P(I),P(2)
CALL CHECK2(P,2,INVAL,INEW)
IF(INVAL .EQ. 1)THEN
  WRITE(1,130)
ENDIF
GOTO(370,380,390,400)

370 WRITE(1,'(/6X,**PRESENT''/4X,**DESIRED''/7X,**X('') ,I2, ''),16X, ''X('',I2, ''))')CHAR,CHAR
WRITE(1,'(/3X,1PE12.5,9X,0)')A(CHAR,CHAR)
READ(5,'(10AI1)')(PL,L=1,10)
CALL CHECK(P,INVAL,RNEW)
IF(INVAL .EQ. 1)THEN
  WRITE(1,130)
ENDIF
GOTO(370,380,390,400)

380 #WRITE(1,390)
390 FORMAT(/8X,**IS CHANGE STILL DESIRED? '',#)
READ(5,':(AI)')P(1)
IF(ICHAR(P(1)) .EQ. 89)THEN
  A(CHAR,CHAR)=RNEW
  WRITE(1,400)
ENDIF
GOTO(370,380,390,400)

250
400 FORMAT(/'1 CHANGE COMPLETED')
PAUSE
ELSEIF(ICHAR(P(1)) .EQ. 78) THEN
   WRITE(1,410)
410 FORMAT(/'NO CHANGE PERFORMED')
PAUSE
ELSE
   WRITE(1,130)
   GO TO 380
ENDIF
420 WRITE(1,110)CHAR112i
WRITE(1,430)
430 FORMAT(/,'FURTHER COEFFICIENT CHANGES TO THIS')/3X,'CONSTRAINT'
   '.','\(')
READ(5,'(A1)')P(1)
IF(ICHAR(P(1)) .EQ. 89) THEN
   GO TO 200
ELSEIF(ICHAR(P(1)) .EQ. 78) THEN
   WRITE(1,130)
   GO TO 420
ENDIF
GO TO 1200
C ONLY INEQUALITY CHANGES ALLOWED
500 WRITE(1,'(/6X,'CONSTRAINT INEQUALITY CHANGE')')
510 WRITE(1,'(/6X,'PRESENT','14X,'DESIRED')')
WRITE(1,'(9X,A1,20X,\(')')PINEQ(CHARD)
READ(5,'(A1)')P(1)
CALL CHECK3(P, INVAL, INEW)
IF(INVAL .EQ. 1) THEN
   WRITE(1,130)
   GO TO 510
ENDIF
520 WRITE(1,390)
READ(5,'(A1)')P(1)
IF(ICHAR(P(1)) .EQ. 89) THEN
C COUNT BY TYPE OF INEQUALITY UPDATED
   WRITE(1,400)
   IF(INEQ(CHARD) .EQ. 0) THEN
      NLC=NLC-1
   ELSEIF(INEQ(CHARD) .EQ. 1) THEN
      NBC=NBC-1
   ELSE
      NEC=NEC-1
   ENDIF
   INEQ(CHARD)=INEW
   IF(INEQ(CHARD) .EQ. 9) THEN
      NLC=NLC+1
      PINEQ(CHARD)='#'
   ELSEIF(INEQ(CHARD) .EQ. 1) THEN
      NBC=NBC+1
      PINEQ(CHARD)='>'
   ELSE
      END
NEC=NEC+1
FINC(CHAR)="=
ENDIF
ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
    WRITE(1,1410)
ELSE
    WRITE(1,120)
    GO TO 520
ENDIF
PAUSE
GO TO 1200
C ONLY RHS CHANGES ALLOWED
650 WRITE(1,164)"(RHS, CONSTRAINT RHS CHANGE")"
660 WRITE(1,164)"(PRESENT",14X,"DESIRED",7X,B('','I2,''),16X,"B('',I2,'')")CHAR,CHAR
    WRITE1,164)"(IPEI2.5,9X,B(ICHAR)
READS,'(16A1)')(P(L),L=1,10)
CALL CHECK(P,INVAL,RHNEW)
IF(INVAL .EQ. 1)THEN
    WRITE(1,130)
    GO TO 660
ENDIF
680 WRITE(1,1390)
READS,'(A1)'P(1)
IF(ICHAR(P(1)) .EQ. 89)THEN
    B(ICHAR)=RHNEW
    WRITE(1,1400)
ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
    WRITE(1,1410)
ELSE
    WRITE(1,120)
    GO TO 680
ENDIF
PAUSE
GO TO 1200
C ONLY CONSTRAINT NAMES CHANGES ALLOWED
800 WRITE(1,110)CHAR(12)
IF(FLAB(5) .EQ. 0)THEN
C MODEL DOES NOT INCLUDE NAMES
    WRITE1,7("I",16X,"MISTAKE!\"THE MODEL BEING EDITED DOES NOT INCLUDE \"CONSTRAINT NAMES, ONLY SUBSCRIPTS.\"YOU ARE BEING RETURNED TO DATA BASE\"EDITOR\")"
    PAUSE
    RETURN
ENDIF
WRITE1,7("R",CONSTRAINT NAME CHANGE")"
WRITE1,7("PRESENT NAME FOR CONSTRAINT ",12," = ",B')CHAR
.0,CH(ICHAR)
810 WRITE1,7("DESIRED NAME FOR CONSTRAINT ",12," = ",B')CHAR
READS,'(A1)'P(1)
READ5,'(A1)'P(1)
IF(ICHAR(P(1)) .EQ. 89) THEN
    CHAN = CHARD
    WRITE(1,400)
ELSEIF(ICHAR(P(1)) .EQ. 78) THEN
    WRITE(1,410)
ELSE
    WRITE(1,130)
    GO TO 810
ENDIF
PAUSE

1200 WRITE(1,110) CHAR(12)
    IF(FLAG(6) .EQ. 4) THEN
        GO TO 1220
ENDIF

1210 WRITE(1,110) CHAR(12)
    WRITE(1,'(10(') , ''FURTHER CHANGES TO THIS CONSTRAINT? ''')
    READ(5,'(A10)') P(1)
    IF(ICHAR(P(1)) .EQ. 89) THEN
        GO TO 200
    ELSEIF(ICHAR(P(1)) .EQ. 78) THEN
        WRITE(1,130)
        GO TO 1220
    END IF

1220 WRITE(1,110) CHAR(12)
    WRITE(1,'(10(') , ''FURTHER CHANGES TO ANY CONSTRAINT? ''')
    READ(5,'(A10)') P(1)
    IF(ICHAR(P(1)) .EQ. 89) THEN
        GO TO 100
    ELSEIF(ICHAR(P(1)) .EQ. 78) THEN
        WRITE(1,130)
        GO TO 1220
    END IF

    IF(FLAG(2) = 0) THEN
        IF(FLAG(6) = 0) THEN
            RETURN
        END}

END
SUBROUTINE ADCON

USE: SOLICITS INPUT OF COEFFICIENTS, INEQUALITY, RHS, AND NAME OF ADDED CONSTRAINT. MODIFIES NECESSARY VARIABLES TO REFLECT ADDITION OF CONSTRAINT TO MODEL.

CALLED BY: SUBROUTINE EDIT

CALLS : SUBROUTINE CHECKP(INVALID,RHS)

SUBROUTINE CHECK3P(INVALID,NEW)

SUBROUTINE INCRA

VARIABLES:

USED: IFLAG(5), INEN, INVALID, RNEN, V, VM(5)

MODIFIED: A(5,5), B(5), CN(5), IFLAG(2), INEG(5), K, NEC, NMC, NLC, F(5),

SUBROUTINE ADCON

CHARACTER VN#6, CN#6, PN#20, MM#5, FM#10, PINENI, PI1, QRN#10

INTEGER Y

COMMON/C1/A(20,60), B(20), C(60), INEG(20), IFLAG(10), NEC, NMC, NLC, K, V,

. UMN

COMMON/C2/VN(60), CN(20), PNN, FMN, PINEN(20), F(10), ORN

100 WRITE(1,110)CHAR(12)

110 FORMAT(A)

WRITE(1,'(12X,'"CONSTRAINT INPUT""/""INPUT CONSTRAINT VARIABLE COEFFICIENTS""/""AS IF THE CONSTRAINT WAS IN THE""/""FOLLOWING FORM"")

WRITE(1,'(6X,'"X(1) + X(2) + X(3) <= RHS""/""THE VARIABLE COEFFICIENTS ARE A MAXIMUM OF 10 CHARACTERS.""/""IF COEFFICIENT IS ZERO, ENTER 0 OR HIT""/""RETURN WITHOUT ENTERY""/"")

PAUSE

120 WRITE(1,110)CHAR(12)

C NUMBER OF CONSTRAINTS UPDATED

<Y + 1

WRITE(1,130)PM

130 FORMAT(SA,'"PROBLEM ID: ",A20)

IF(FLAG(5) .EQ. 1 THEN

C MODEL INCLUDES NAMES SO CONSTRAINT NAME INPUT BY USER

WRITE(1,'(/9X,'"CONSTRAINT NAME INPUT""/""ENTER CONSTRAINT NAME WHICH CORRESPONDS TO CONSTRAINT 0",I2,""/""NAME ARE TO BE 6 CHARACTERS OR LESS.""'))K

WRITE(1,'(/9X,'"CONSTRAINT ",I2,"" = ",A1)')K

READ(5, '(A6)') CN(K)

WRITE(1,110)CHAR(12)

140 WRITE(1,130)PM

WRITE(1,'("CONSTRAINT VARIABLE COEFFICIENT INPUT")')

WRITE(1,'(/9X,'"CONSTRAINT ",I2,"" = ",A6,"",/"K,CN(K)

C ADDED CONSTRAINT COEFFICIENTS INPUT BY USER

20 200 J=1,V

150 FORMAT(7X,'"X",I2,"" = ",A6,"",/"K,CN(K)

READ(5, '(10A1)')/(PL,L=1,10)
CALL CHECK(P, INVAL, RNEW)
IF(INVAL .EQ. 1) THEN
  WRITE(1, 160)
  FORMAT('5X,"INVALID, ENTRY, PLEASE REENTER")
  GO TO 140
ELSE
  A(K, J) = RNEW
ENDIF
CONTINUE
ELSE
  WRITE(1, 210)
  FORMAT('6X,"CONSTRAINT ",I2")K
  DO 250 J = 1, N
  WRITE(1, 220) J
  220 FORMAT('6X, 12)KZ, READ(5, '10A1') P(L), L = 1, 10)
  CALL CHECK(P, INVAL, RNEW)
  IF(INVAL .EQ. 1) THEN
    WRITE(1, 160)
    GO TO 210
  ELSE
    A(K, J) = RNEW
  ENDIF
CONTINUE
ENDIF
C ADDED CONSTRAINT INEQUALITY INPUT BY USER
IF(FLAG(5) .EQ. 0) THEN
  WRITE(1, '11X,"INEQUALITY ",I2")K
ELSE
  WRITE(1, '11X,"INEQUALITY",12X,0")
ENDIF
READ(5, '10A1') P(I)
CALL CHECK(P, INVAL, INEV)
IF(INVAL .EQ. 1) THEN
  WRITE(1, 150)
  GO TO 260
ELSE
  INEG(K) = INEV
ENDIF
C COUNT BY INEQUALITY TYPE UPDATED
IF(INEV .EQ. 0) THEN
  MLC = MLC + 1
  PINEK(K)="<
ELSEIF(INEV .EQ. 1) THEN
  NEC = NEC + 1
  PINEK(K)="=
ELSE
  NEC = NEC + 1
  PINEK(K)=">
ENDIF
C ADDED CONSTRAINT RHS INPUT BY USER
IF(FLAG(5) .EQ. 0) THEN
  WRITE(1, '18X,"RHS = ",I2")
ELSE
  WRITE(1, '(6X,'"RHS",10X,"=",0)')
ENDIF
READ(5, '(10A1)' ) (P(L),L=1,10)
CALL CHECK(P,INVAL,RNEW)
IF (INVAL .EQ. 1) THEN
  WRITE(1,160)
  GO TO 280
ELSE
  D(X)=RNEW
ENDIF
PAUSE
290 WRITE(1,110)CHAR(12)
WRITE(1, '(11(/,7X,"ARE CORRECTIONS NEEDED? ",0)')
READ(5, '(A1)')F(I)
IF (ICHR(P(I)) .EQ. 69) THEN
  CALL ICNRCII
  GO TO 300
ELSEIF (ICHR(P(I)) .EQ. 78) THEN
  GO TO 300
ELSE
  WRITE(1,160)
  GO TO 290
ENDIF
300 WRITE(1,110)CHAR(12)
310 WRITE(1, '(11(/,8I1,"ADD ANOTHER CONSTRAINT? ",0)')
READ(5, '(A1)')F(I)
IF (ICHR(P(I)) .EQ. 69) THEN
  GO TO 120
ELSEIF (ICHR(P(I)) .EQ. 78) THEN
  IFLAG(2)=0
  RETURN
ELSE
  WRITE(1,160)
  GO TO 300
ENDIF
END
SUBROUTINE DJSPLY
C USE: FORMATS TABLEAU OUTPUT TO BOTH SCREEN AND PRINTER. OUTPUTS
  EITHER THE CONSTRAINTS ONLY OR THE COMPLETE TABLEAU AS INPUT
  BY USER. OPENS AND CLOSES OUTPUT UNIT DESIGNATED BY USER.
C CALLED BY: PROGRAM DATAB
C CALLS : SUBROUTINE CHECK2(P,N,N,INVAL,INEM)
C VARIABLES:
C USED: A(N),B(I),C(I),CN(I),IFLAG(5),INEM,INVAL,K,NH,OBJN,
C PINE0(I),PN,V,VM(N)
C MODIFIED: IFLAG(9),N,P(I),T

SUBROUTINE DJSPLY
CHARACTER VN64,CM64,PN420,MM410,FNM10,PINE01,PN10,OBJN10
INTEGER V,I
COMMON/C1/(A(20,60),B(20),C(I,60),INEM(20),IFLAG10),NHEC,NLC,K,V,
  C(KM4)
COMMON/C2/VN(60),CN(20),PN,MM,FNM10,PINE01,PN10,OBJN
WRITE(1,110)CHAR(12)
110 FORMAT(A)
WRITE(1,'(8U10.60)')"WOUL YOU LIKE DISPLAY ON: '/:5X"1. SCREEN",
  "/(20X,"2. PRINTER")"
WRITE(1,120)CHAR(13)
120 FORMAT(13X,"WHICH OPTION? 1.,2.")
READ(S,'(AI)')P(I)
CALL CHECK2(P,1,2,INVAL,INEM)
IF(INEM.EQ.1)THEN
  WRITE(1,130)
130 FORMAT(110)CHAR(12)
ENDIF
C PROPER FILE FOR SELECTED OUTPUT DEVICE OPENED
IF(INEM.EQ.1)THEN
  OPEN(2,FILE='CONSOLE:')
ELSE
  OPEN(2,FILE='PRINTER:')
ENDIF
WRITE(1,110)CHAR(12)
IF(IFLAG(9).EQ.1)THEN
  C ONLY CONSTRAINTS ARE DISPLAYED
  WRITE(2,220)PN
220 FORMAT(10X,A20/10X,"CURRENT CONSTRAINTS")
ELSE
  C OBJ FUNCTION AND CONSTRAINTS DISPLAYED
  WRITE(2,230)PN
ENDIF
WRITE(2,240)OBJN
240 FORMAT(A10)
ELSE
  WRITE(2,250)
FORMAT( ' ' )
ENDIF
ENDIF
C
NUMBER OF 80 COLUMN DISPLAYS REQUIRED DETERMINED
N = (IV/5) + 1
DO 470 N = 1, N
IF(IFLAG(I5) .EQ. 1) THEN
C
VARIABLE NAMES PRINTED AS COLUMN HEADERS
WRITE(2, '(13X,8)')
GO TO 270
IF(J .GE. VN(I)) THEN
END IF
WRITE(2, (260) ) VN(I)
FORMAT( 4X, 6X, 3X, 8 )
270 CONTINUE
WRITE(2, '  )
ENDIF
WRITE(2, '(13X,8)')
GO TO 290
IF(J .GE. VN(I)) THEN
END IF
WRITE(2, (260) ) VN(I)
FORMAT( 4X, 6X, 3X, 8 )
290 CONTINUE
C
IF LAST 80 COLUMN DISPLAY, DISPLAY RHS
IF(T .EQ. 1 .OR. N .EQ. T) THEN
WRITE(2, 300)
FORMAT( 8X, 1D12.5, 8 )
ELSE
WRITE(2, '('''' ')
ENDIF
IF(IFLAG(9) .EQ. 0) THEN
WRITE(2, '(''OBJ FUNCTION'', 1X,8)')
GO TO 320
IF(J .GE. VN(I)) THEN
END IF
WRITE(2, (310) ) VN(I)
FORMAT( 8X, 1PE12.5, 8 )
310 CONTINUE
IF(T .EQ. 1 .OR. N .EQ. T) THEN
WRITE(2, 330)
FORMAT( 8X, 'I ')
ELSE
WRITE(2, '('''' ')
ENDIF
WRITE(2, '(''CONST NAME'', 2X,67('''', '))
ELSE
WRITE(2, '(''CONST NAME'';')
ENDIF
258
CONSTRAINT NUMBER, NAME, BASIC VARIABLE, COEFFICIENTS, INEQUALITY, AND RHS DISPLAYED

DO 400 L=1,K
  IF(L .GT. K) THEN
    GO TO 400
  ENDIF
  WRITE(2,340) L
  IF(ITAM(L) .EQ. 1) THEN
    WRITE(2,350) CN(L)
  ELSE
    WRITE(2,'(8X,A6,S)') CN(L)
  ENDIF
  IF(IFLAG(5) .EQ. W) THEN
    WRITE(2,350) CW(L)
  ELSE
    WRITE(2,350) CW(L)
  ENDIF
  IF(J .EQ. V) THEN
    GO TO 370
  ENDIF
  WRITE(2,360) A(L,J)
  CONTINUE
  IF(T .EQ. T) THEN
    WRITE(2,380) PINEL(L), VEL(L)
  ELSE
    WRITE(2,'(2(/))')
  ENDIF
  CONTINUE

SPACING APPROPRIATE FOR SELECTED OUTPUT DEVICE IMPLEMENTED

IF(INW .EQ. 1) THEN
  PAUSE
  WRITE(2,110) CHAR(12)
ELSE
  WRITE(2,'(2(/))')
ENDIF

IFLAG(9)=0

OUTPUT DEVICE FILE CLOSED
CLOSE(2)
RETURN
END
C-module 1 unit16
C
C unit uses: unit17
C
C subroutine save
C
C use: solicits volume,filename of disk file to store new or edited
C model. also solicits input which identifies inputted file as
C a new file or an update of file. saves data in disk file
C (fn) and also writes (fn) to diskfile lpi:lpdata for transfer
C to modules 2 and 3.
C called by: program datab
C calls: none
C variables:
C used: (,)(c5,fa(),il5),l8(),c(6),ilas(6),ilas(),ilas(),ilas(),ilas()
C model. also solicits input which identifies inputted file as
C a new file or an update of file. saves
C (fn) and also writes (fn) to diskfile lpi:lpdata for transfer
C to modules 2 and 3.
C called by: program datab
C calls: none
C variables:
C used: (,)(c5,fa(),il5),l8(),c(6),ilas(6),ilas(),ilas(),ilas()

C uses uccheck in unit17.code overlay
C subroutine save
C character vn6,cn6,pn20,nn5,fn10,pineq1,pn1,objm10,res6
C integer v
C common.ci/a20,60),bl20,ci60),ineq20),ilas6,ilas6,ilas6,ilas6,ilas6,ilas6,ilas6
C common.cv/vn60,cn40,pm,nn,fm,pineq20,pn1,objm
100 write1(1,110)char(12)
110 format(6)
C user inputs file name which model is to be saved
write1(1,'/9,1"save lp model to disk"/2x,"enter the disk
C drive number and file"/6x,"name which you want problem"/10x,200,
C "saved under"/10x)/pn
write1(1,'/8,"enter exactly as follows"/10x,"disk drive:filename
C .me"/12x,"eg. #4:filename"/12x,"the drive:filename must be 10 ch
C .acters"/16x,"or less"/16x,"if the above is entered incorrectly
C ,"/7x,"your model will be lost!!")
write1(1,'/7x,"disk:filename = "",91)
read(is,'(a10)'fn
120 write1(1,'/7x,"are corrections needed? ",91)
read(is,'(a1)'f(1)
if(ichar(p(1)) .eq. 89)then
   go to 100
elseif(ichar(p(1)) .ne. 78)then
   write1(1,200)
   pause
   go to 120
endif
write1(1,110)char(12)
C user prompted to insert disk to which file is to be saved
write1(1,'/91,2x,"insure the disk to contain the file"/15x,410
C ./13x,"is available."/611)"fn
pause
write1(1,110)char(12)
WRITE(1, '911,'''HAS THIS DISK:FILENAME COMBINATION BEEN''/12X,'''U
SE, PREVIOUSLY?''/11X(ARE YOU UPDATING A CURRENTLY EXISTING''/17X,
''FILE?''))
150 WRITE(1, '161,'''(Y/N)'') READ(5, '(A1)') P(1)
C PROPER STATUS OF FILE DETERMINED AND OPENED
IF(ICHAR(P(1)) .EQ. 99) THEN
   OPEN(3, FILE=FN, STATUS='OLD', FORM='UNFORMATTED')
ELSE IF(ICHAR(P(1)) .EQ. 78) THEN
   OPEN(3, FILE=FN, STATUS='NEW', FORM='UNFORMATTED')
ELSE
   WRITE(1, 200)
200 FORMAT('5X,'''INVALID ENTRY. PLEASE REENTER'')
GO TO 150
ENDIF
C NO MODEL WRITTEN TO DISK
WRITE (3, P/NM,NH,NK,N,X,HEC,NXC,NLC)
DO 250 I=1,10
   WRITE(3) IFLAG(I)
250 CONTINUE
DO 300 I=1,K
   WRITE(3) INEG(I), PINEQ(I), B(I)
DO 290 J=1,V
   WRITE(3) A(I,J)
290 CONTINUE
300 CONTINUE
DO 350 J=1,V
   WRITE(3) C(J)
350 CONTINUE
IF(IFLAG(5).EQ. 1) THEN
   DO 380 I=1,K
   WRITE(3) CN(I)
380 CONTINUE
DO 400 J=1,V
   WRITE(3) VM(I)
400 CONTINUE
WRITE(3) OBJM
ENDIF
IFLAG(2)=1
CLOSE(3, STATUS='KEEP')
WRITE(1, 101) ICHAR(12)
WRITE(1, '111/111','''INSURE DISK LP1 IS AVAILABLE.'',7(/1))
PAUSE
C NAME OF MODEL LAST SAVED WRITTEN TO TRANSFER FILE
OPEN(3, FILE='LP1\LPDATA', STATUS='OLD', FORM='UNFORMATTED')
WRITE (3, FN)
CLOSE(3, STATUS='KEEP')
RETURN
END
SUBROUTINE INIT

INTEGER V
COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NLC,K,V

INEQ(1)=0
DO 100 J=1,60
   A(I,J)=0.
100   CONTINUE
DO 200 J=1,60
   C(I,J)=0.
200   CONTINUE
DO 300 I=1,10
   IFLAG(I)=0
300   CONTINUE
NEC=0
NGC=0
NLC=0
RETURN
END
C SUBROUTINE DATAD
C USE: SOLICITS VOLUME:FILENAME FROM USER OF FILE TO BE READ FROM
C DISK. PROMPTS USER TO INSERT CORRECT DISK AND READS MODEL
C REQUESTED FROM DISK INTO MEMORY FOR FUTURE EDITING OR
C DISPLAY.
C CALLED BY: PROGRAM DATAB
C CALLS : NONE
C VARIABLES:
C USED: NONE
C MODIFIED: (.)SI,;).IS.NIS1 THRU IFILAO(10),
C VN(60)
SUBROUTINE DATAD
CHARACTER VN$60,CN$60,PN$20,MM$3,FNS$10,?INEQ$1,PSI,OBJN(10)
INTEGER Y
COMMON/C1/A(20,60),B(20),C(6Q),INEQ(20),IFLAG(10),POBN(10),OBJN$10
COMMON/C2/VN$60,CN$20,PN$20,MM$3,FNS$10,PINEQ(20),P(10),OBJN
WRITE 11,110)CHAR412)
110 FORMAT (A)
C USER INPUTS FILE NAME OF MODEL TO BE READ
WRITE1,'(S工商+)''READ LP MODEL FROM DISK='/''ENTER THE DISK DRIVE NUMBER AND FILE NAME WHICH HOLDS THE MODEL DESIRED.''/''ENTER EXACTLY AS FOLLOWS''/''DISK DRIVE:FILENAME''/121,'''E S. #4:FILENAME''))
WRITE1,'(('7X, ''DISK:FILENAME = '''',A)')
READ(5,'(A10)')FN
WRITE(11,'(A10)')CHAR(12)
C USER PROMPTED TO INSERT DISK CONTAINING DESIGNATED FILE
WRITE1,'(('9I5)''INSURE THE DISK CONTAINING THE '''/5X,A10//10X ''MODEL IS AVAILABLE.'''/7(/))''FN
PAUSE
C DESIGNATED FILE OPENED AND READ TO MEMORY
OPEN(3,FILE='''FN,STATUS=OLD',FORM='''UNFORMATTED')
READ(3)PN,MMMN,MM,K,V,NEC,NBG,MLC
DO 180 I=1,10
READ(3)IFLAG(I)
180 CONTINUE
DO 250 I=1,K
READ(3)INEQ(I),PINEQ(I),B(I)
250 CONTINUE
DO 270 J=1,V
READ(3)A(I,J)
270 CONTINUE
IF(IFLAG(5) .EQ. 1) THEN

263
DO 280 I=1,K
    READ(3)CN(I)
    CONTINUE
DO 300 J=1,V
    READ(3)VN(J)
    CONTINUE
READ(3)OB2N
ENDIF
IFLAG(2)=1
CLOSE(3,STATUS='KEEP')
WRITE(1,110)CHAR(12)
WRITE(1,'(111/',11,'INSURE DISK LP1 IS AVAILABLE.'',7(/)
PAUSE
C LAST READ FILE NAME IS WRITTEN TO TRANSFER FILE
OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
WRITE(3,FH)
CLOSE(3,STATUS='KEEP')
RETURN
END
C MODULE 1 UNIT 16
C USE: FORMATS AND DISPLAYS TITLE/AUTHOR PAGE.
C CALLED BY: PROGRAM DATAB
C CALLS: NONE
C VARIABLES: NONE

SUBROUTINE HEADER
WRITE(1,'(A)') CHAR(12)
WRITE(1,'(9X,22("""")/9X,"""",3X,""""FORTAN BASED"""",4X,"""
'/9X,","",20X,"""",4X,""""LINEAR PROGRAMMING"""",9X,"""
'/9X,","",20X,"""",3X,""""MICROCOMPUTERS"""",3X,"""
'/9X,","",22("""")))')
WRITE(1,'(/19X,""BY""/9X,""THEODORE R. E. FRALEY""/18X,""AND"
'/14X,""DALE A. KEM"")')
RETURN
END
C SUBROUTINE MODUL(INEW)

C USE: UPON INITIAL ENTRY INTO MODULE, SOLICITS INPUT AS TO WHICH
C MODULE OF THE SOFTWARE PACKAGE IS DESIRED. PROVIDES
C INSTRUCTIONS AS TO WHAT PRIOR ACTIONS ARE REQUIRED TO ENTER
C EACH MODULE AND THE COMMANDS REQUIRED TO GAIN ACCESS TO THESE
C MODULES. IF USER SELECTS TO ENTER ANOTHER MODULE, MODULE 1
C PROGRAM IS TERMINATED WITH INSTRUCTIONS ON SCREEN SHOWING THE
C COMMANDS REQUIRED TO ENTER SELECTED MODULE. OTHERWISE, USER
C ENTERS NEW MODEL OR EDITS MODEL WITH THIS MODULE.
C CALLED BY: PROGRAM DATAB
C CALLS : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)
C VARIABLES:
C USED: INVAL
C MODIFIED: FN,INEW,P(R)

SUBROUTINE MODUL(INEW)
CHARACTER VN(6),CH(6),PM(20),MN(3),F(10),PINEW(1),P(10),OBJN(10)
COMMON/CV(N(6),CN(20),PN,MM,FN,PINEW(20),P(10),OBJIN

C DETECTS IF MODULE DESIRED ALREADY SELECTED
IF(INEW .NE. 0) THEN
  INEW=INEW+1
  GO TO 200
ENDIF
WRITE(1,110)CHAR(12)
110 FORMAT(A)
C DISPLAYS MODULE OPTIONS
WRITE(1,’(1X,’’MODULE SELECTION’’/’’THE FOLLOWING OPTIONS ARE A
. AVAILABLE’’/’’1. DATA BASE ENTRY (ENTER DATA BASE OR ’’EDIT
. CURRENT DATA BASE’’/’’2. LP INSTRUCTIONAL MODULE’’/’’3. LP PRO
.BLEM SOLVER MODULE’’))’’’’)
WRITE(1,’(1X,’’4. LP SENSITIVITY ANALYSIS MODULE’’/’’(NOTE: OPTIONS
. 2, OR 3 REQUIRE THAT A ’’DATA BASE CURRENTLY STORED ON DISK)
. ’’))’’
WRITE(1,’(1X,’’(NOTE: OPTION 4 REQUIRES THAT A DATA’’/’’FILE HAVE B
.EEN SAVED UPON LEAVING THE’’/’’OPTION 2 OR 3 MODULES ABOVE.’’))’’
120 WRITE(1,130)
130 FORMAT(1X,’WHICH OPTION? ’’A’’)
READ(5,’(A1)’’)P(1)
CALL CHECK2(P,1,4,INVAL,INEW)
IF(INVAL .EQ. 1) THEN
  WRITE(1,140)
140 FORMAT(5X,’INVALID ENTRY, PLEASE REENTER’’)
  GO TO 120
ENDIF
IF(INEW .EQ. 1) THEN
C USER ELECTS TO ENTER MODEL
RETURN
ENDIF
200 WRITE(1,110)CHAR(12)
IF(INEW .EQ. 4) THEN
C USER ELECTS TO PERFORM SENSITIVITY ANALYSIS AND PROMPTED TO
C INSERT DISK LP2 TO WRITE FILE NAME IN TRANSFER FILE
WRITE(1,'(i1,l1,'''INSURE DISK LP2 IS AVAILABLE.'''),7(//
),(//)
PAUSE
C TRANSFER FILE OPENED
OPEN(3,FILE='LP2:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
ELSE
C USER HAS SELECTED OTHER THAN SENSITIVITY ANALYSIS AND PROMPTED
C TO INSERT DISK LP1
WRITE(1,'(i1,l1,'''INSURE DISK LP1 IS AVAILABLE.'''),7(//
),(//)
PAUSE
C TRANSFER FILE OPENED
OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
ENDIF
WRITE(1,110)CHAR(12)
IF(INEW .EQ. 2) THEN
C USEP ELECTS EDUCATIONAL MODULE
WRITE(1,'('BY,'''LP INSTRUCTIONAL MODULE''')
ELSEIF(INEW .EQ. 3) THEN
C USER ELECTS PROBLEM SOLVER MODULE
WRITE(1,'('8X,'''LP PROBLEM SOLVER MODULE''')
ELSE
WRITE(1,'(8X,'''LP SENSITIVITY ANALYSIS MODULE''')
ENDIF
C MODEL FILE NAME IN TRANSFER FILE READ
READ(3)FN
CLOSE(3,STATUS='KEEP')
WRITE(1,'(''TO USE THIS MODULE, A DATA BASE MUST HAVE BEEN PREVIOUSLY CREATED USING THE DATA BASE ENTRY MODULE I AND SAVED TO DISK''):
WRITE(1,'(''THE DATA BASE WHICH IS CURRENTLY IDENTIFIED AS THE PROBLEM TO BE STUDIED IS''/''S''/''S/AIO'')
C USER DETERMINES IF PROPER FILE NAME IN TRANSFER FILE
READ(5,'(A1)')PI
IF(ICHAR(PI) .EQ. 69) THEN
  SO TO 360
ELSEIF(ICHAR(PI) .EQ. 78) THEN
C USER ENTERS DESIRED FILE NAME TO BE ENTERED IN TRANSFER FILE
WRITE(1,'(''PLEASE ENTER THE DISK DRIVE NUMBER AND FILENAME OF THE FILE YOU WISH TO STUDY.'''/''INSURE THIS IS ENTERED EXACTLY AS IT IS''):
WRITE(1,'(''WAS SAVED PREVIOUSLY AND ALSO THAT THE PROPER DISK IS IN THE PROPER DRIVE.''):
WRITE(1,'(''MODEL TO BE STUDIED = ''/''
READ(5,'(A10)')FN
WRITE(1,'(''ARE CORRECTIONS NEEDED? ''/''
READ(5,'(A10)')PI
IF(ICHAR(PI) .EQ. 89) THEN
  270
SD TO 200
ELSEIF (ICHR(P(I)) .NE. 78) THEN
    WRITE(1,140)
    GO TO 250
ENDIF
WRITE(1,110)CHP(I2)
IF(NEW .EQ. 4) THEN
    USER PROMPTED TO INSERT DISK LP2 AND FILE NAME IS WRITTEN
    WRITE(1,'(I10,I4,1X,''INSURE DISK LP2 IS AVAILABLE.'',7,(''(''))')
    PAUSE
    OPEN(3,FILE='LP2:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
ELSE
    MODEL FILE NAME WRITTEN TO TRANSFER FILE FOR OTHER THAN
    SENSITIVITY ANALYSIS
    WRITE(1,'(I10,I4,1X,''INSURE DISK LP1 IS AVAILABLE.'',7,(''(''))')
    PAUSE
    OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
ENDIF
WRITE(3,99)
CLOSE(3,STATUS='KEEP')
ELSE
    WRITE(1,140)
    GO TO 230
ENDIF
WRITE(1,110)CHAR(I2)
WRITE(1,'(I10,I4,1X,''INSURE DISK LP1 IS AVAILABLE.'',7(I))')
PAUSE
300 WRITE(1,110)CHAR(I2)
C INSTRUCTIONS TO ENTER OTHER MODULES DISPLAYED
IF(NEW .EQ. 2) THEN
    WRITE(1,'(B/I,1X,''TO ENTER THE LP INSTRUCTIONAL MODULE:/'',I7X,
    ''TYPE''/I9X,''X''/I7X,''LP1:ED'',3(I))')
    STOP
ELSEIF(NEW .EQ. 3) THEN
    WRITE(1,'(B/I,1X,''TO ENTER THE LP PROBLEM SOLVER MODULE:/'',I7X,
    ''TYPE''/I9X,''X''/I6X,''LP2:TAB'',3(I))'
    STOP
ELSE
    WRITE(1,'(B/I,1X,''TO ENTER THE LP SENSITIVITY ANALYSIS''/I6X,
    ''MODULE''/I7X,''TYPE''/I9X,''X''/I6X,''LP2:SEN'',3(I))')
    STOP
ENDIF
RETURN
END
SUBROUTINE DBHED
WRITE(1,'(A)')CHAR(12)
WRITE(1,'(4X,9X,22(''8''/9X,:''8'',20X,:''8''/9X,:''8'',8X,'''DATA''
,6X,9X/9X,:''8'',20X,:''8''/9X,:''8'',8X,'''BASE'',8X,9X/9X,:''8''
,9X,:''8'',10X,9X,:''ENTRY'',7X,:''8''/9X,:''8'',9X,20X,:''8''),/ 
4X,9X,7X,'''MODULE'',7X,:''8'',2/(9X,:''8'',20X,:''8''),/ 
9X,8X,6X,'''MODULE 1''',6X,9X/9X,:''8'',20X,9X/9X,22(''8''),/ 
3(''i'')')
PAUSE
RETURN
END
SUBROUTINE INTRO
C USE: PROVIDES OVERVIEW OF LP PACKAGE AND PROVIDES BRIEF REFERENCE TO DOCUMENTATION.
C CALLED BY: PROGRAM DATAB
C CALLS : NONE
C VARIABLES: NONE
C
SUBROUTINE INTRO
WRITE(I,'(A)')CHAR(12)
WRITE(I,'(X,'"LINEAR PROGRAMMING SOFTWARE PACKAGE"/*"THIS PACKAGE IS DESIGNED TO ALLOW STUDENTS TO IMPROVE THEIR UNDERSTANDING OF THE SIMPLEX ALGORITHM AND ALSO PROVIDE THE MANAGERS A TOOL FOR ANALYSIS WITH A "PROBLEM SOLVING TOOL.""*')
WRITE(I,'(/,"THE PACKAGE CONSISTS OF FOUR DISTINCT PROGRAMS (ANNOTATED AS MODULES) WHOSE FUNCTIONS ARE AS FOLLOWS:*/"MODULE 1: DATA BASE ENTRY*/"MODULE 2: LP INSTRUCTION*/"MODULE 3: LP PROBLEM SOLVER*/"MODULE 4: SENSITIVITY ANALYSIS"*/')
WRITE(I,'(/,"ALL LP PROBLEMS MUST BE ENTERED INTO A DATABASE USING MODULE 1. MODULES 2 OR 3 MAY BE USED TO DETERMINE A SOLUTION TO A PROBLEM AND THIS MUST OCCUR PRIOR TO ENTERING MODULE 4."*/')
PAUSE
WRITE(I,'(A)')CHAR(12)
WRITE(I,'(/,"INSTRUCTIONS ON HOW TO ENTER EACH MODULE WILL BE PRESENTED WHEN APPROPRIATE. */"ANSWERS TO SPECIFIC QUESTIONS CONCERNING ANY MODULE WILL BE FOUND IN THE USERS GUIDE (APPENDIX A) OF THE THESIS */"DOCUMENTATION."*/')
WRITE(I,'(/,"ALL RESPONSES REQUIRE A [RETURN] TO NOTE */" THE COMPLETION OF INPUT. */"ALSO, ALL YES/NO INPUTS MAY BE ENTERED BY [Y] OR [N], RESPECTFULLY."*/')
PAUSE
RETURN
END
C SUBROUTINE DBE
C USE: DISPLAYS MENU OF DATA BASE ENTRY OPTIONS (NEW MODEL, READ
C EXISTING MODEL, QUIT, INTRODUCTORY REMARKS).
C CALLED BY: PROGRAM DATAB
C CALLS : NONE
C VARIABLES: NONE

SUBROUTINE DBE
WRITE(1,'(A)')CHAR(12)
WRITE(1,'(1X,''DATA BASE ENTRY''/5X,''TO ENTER LP MODEL DATA BAS
,E''/4X,''YOU HAVE THE FOLLOWING OPTIONS:''/1/''1. CREATE MODEL INT
,ERACTIVELY:SUBSCRIPTS''/3X,''(VARIABLES ANNOTATED BY SUBSCRIPTS,''
./3X,''CONSTRAINTS ANNOTATED BY NUMBER ONLY)'')
WRITE(1,'(/'',2. CREATE MODEL INTERACTIVELY:NAMED''/3X,''VARIABLES
, AND CONSTRAINTS ARE''/3X,''ASSIGNED NAMES)''/1/''3. READ FROM DISK
.,''/3X,''(PREVIOUSLY CREATED BASE)''/3X)
WRITE(1,'(/'',4. DISPLAY INTRODUCTORY REMARKS''/5X,''QUIT PROGRAM''
.,''/))
RETURN
END
SUBROUTINE DBM
WRITE(1,'(A)'CHAR(12))
WRITE(1,'(1X,A)'DATA BASE MANAGEMENT''THE FOLLOWING OPTIONS ARE AVAILABLE:"/3X,'(SCREEN OR PRIVATE)''2. EDIT CURRENT LP MODEL''/3X,'(CHANGE ANY PARAMETER)''/3X,'3. SAVE CURRENT MODEL TO DISK FILE''/3X,'(MAY THEN EDIT TO ANOTHER MODEL)')")
WRITE(1,'(1X,A)'CURRENT MODEL LOST IF NOT ON DISK)''/3X,'SOLVE PROBLEM''/3X,'(INCLUDES EDUCATIONAL, PROBLEM SOLVER,'/3X,'AND SENSITIVITY ANALYSIS)''/3X,'QUIT: UNSAVED FILES DESTROYED")")
RETURN
END
SUBROUTINE DEM
WRITE(1,'(A)')CHAR(12)
WRITE(1,'(I10,2X,A)')"EXECUTION MANAGEMENT"/"THE FOLLOWING OPTIONS ARE AVAILABLE:"/"1. LP INSTRUCTIONAL MODULE"/"2. EACH TABLEAU MAY BE DISPLAYED"/"3. PROBLEM SOLVER MODULE"/"NO USER INTERACTION"/"4. SENSITIVITY ANALYSIS MODULE"/"5. RETURN TO DATA BASE MANAGEMENT MENU"/"QUIT: USERSAVED FILES WILL BE LOST!"
RETURN
END
SUBROUTINE CHECK(E,INAL,RENN)

VERIFIES USER INPUT OF REAL NUMBERS WHICH HAVE BEEN READ INTO SINGLE ELEMENT CHARACTER STRINGS. CHECKS ELEMENT BY ELEMENT THAT EACH CHARACTER STRING IS A NUMERIC, A VALID OPERATOR, OR DECIMAL POINT. IF ALL ARE VALID, TRANSFORMS CHARACTER STRING REPRESENTATION INTO NUMERIC REAL. IF INVALID CHARACTER IS FOUND, FLAG SET WHICH CALLING ROUTINE CHECKS TO SIGNAL USER TO REINPUT NUMBER.

CALLED BY: SUBROUTINE ADCON

SUBROUTINE ADVAR

SUBROUTINE CMVA

SUBROUTINE DATAN

SUBROUTINE DATAS

SUBROUTINE ICNCH

SUBROUTINE OBJCH

CALLS: NONE

VARIABLES:

USED: ALLOW(14),E(IN),M

MODIFIED: DECIMA,INVAL,NEGAT,RENN

SUBROUTINE CHECK(E,INVAL,RENN)

CHARACTER ALLOW(14)

DIMENSION E(IN),ALLOW(14)

REAL M

INTEGER DECIMA

DATA ALLOW/'1','2','3','4','5','6','7','8','9','0','+','-','.'/

RENN=0.0

M=-1

INVAL=0

DECIMA=0

NEGAT=0

DO 400 I=1,10

CHECKS FIRST FOR BLANK CHARACTERS

IF (E(I) .EQ. ALLOW(14)) THEN

GO TO 400

ENDIF

CHECKS EACH CHARACTER TO INSURE ACCEPTABILITY

DO 200 J=1,13

IF (E(I) .EQ. ALLOW(J)) THEN

IF (DECIMA .EQ. 1) THEN

GO TO 130

ELSEIF (E(I) .EQ. '"') THEN

NEGAT=1

GO TO 400

ELSEIF (E(I) .EQ. '.') THEN

DECIMA=1

GO TO 400

ENDIF

200 CONTINUE

400 CONTINUE

274
ELSE
  RNEM=10*RNEN + (ICHAR(E(I)) - 48)
  GO TO 400
ENDIF
50
  RNEM=RNEM+(ICHAR(E(I))-48)IN
  M=M+1
  GO TO 400
ELSEIF (J .EQ. 13) THEN
  INVAL=-1
  RNEM=0.0
  RETURN
ENDIF
200  CONTINUE
400  CONTINUE
   IF (NEGAT .EQ. 1) THEN
      RNEM = -1.0  RNEN
   ENDIF
   RETURN
END
SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)

USE: VERIFIES USER INPUT OF INTEGER NUMBERS WHICH HAVE BEEN READ INTO SINGLE ELEMENT CHARACTER STRINGS. CHECKS ELEMENT BY ELEMENT THAT EACH CHARACTER STRING IS NUMERIC OR BLANK. IF ALL ARE VALID, TRANSFORMS CHARACTER STRING REPRESENTATION INTO NUMERIC INTEGER. IF INVALID CHARACTER FOUND, FLAG SET WHICH CALLING ROUTINE CHECKS TO SIGNAL USER TO REINPUT NUMBER.

CALLED BY: PROGRAM DATAB

SUBROUTINES ADVAR, CNVA, DELCON, DELVAR, DISPLY, GENIF, INARCH, MODUL(INEW), OBCH, VNCH

CALLS: NONE

VARIABLES:

USED: ALLOW(I), D, E(I), HVAL

MODIFIED: INEW, INVAL

SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)

CHARACTER ALLOW(1), E(1)

DIMENSION E(10), ALLOW(11)

INTEGER D, HVAL

DATA ALLOW/'1','2','3','4','5','6','7','8','9','0',' '/

INEW=0

INVAL=0

DO 300 J=1,10

C CHECKS FIRST FOR BLANK CHARACTERS

IF(E(I) .EQ. ALLOW(I)) THEN

60 TO 300

ELSEIF(E(I) .EQ. ALLOW(J)) THEN

INEW=INEW+10 + (ICHAR(E(I))-48)

60 TO 300

ELSEIF(J .EQ. 10) THEN

INVAL=1

INEW=0

RETURN

ENDIF

200 CONTINUE

300 CONTINUE

IF(INEW .EQ. 0 .OR. INEW .GT. HVAL) THEN

INVAL=1

INEW=0

276
SUBROUTINE CHECK3(EINVAL,IENS,EIN)
CHARACTER ALLOW(11),E(10)
DIMENSION E(10),ALLOW(3)
DATA ALLOW/"'f','='","
IEN=0
INVAL=0
IF(ICHAR(E(I)) .EQ. 60)THEN
  IEN=0
ELSEIF(ICHAR(E(I)) .EQ. 62)THEN
  'IEN=1
ELSEIF(ICHAR(E(I)) .EQ. 61)THEN
  IEN=2
ELSE
  INVALID=1
ENDIF
RETURN
END
C PROGRAM EDUC
C USE: MAIN PROGRAM OF MODULE 2, LINEAR PROGRAMMING PACKAGE.
C PURPOSE OF MODULE IS TO PROVIDE A TUTORIAL WHICH PROVIDES
C GUIDANCE IN THE SEQUENCE OF STEPS OF TRANSFORMING A GIVEN
C PROBLEM INTO THE TABULAR FORM WHICH THE SIMPLEX METHOD IS
C APPLIED. ALSO THE MATHEMATICAL OPERATIONS ARE EMPHASIZED BY
C SOLICITING RESPONSES FROM THE USER UPON THE VIEWING OF EITHER
C NUMERICAL MANIPULATION OPTIONS OR OBJECTIVE SELECTION OPTIONS.
C USER IS GIVEN IMMEDIATE FEEDBACK ON CORRECTNESS OF OPTION
C SELECTION WITH A BRIEF INSTRUCTIONAL NOTE FOLLOWING INCORRECT
C RESPONSES. MODULE 2 CONSISTS OF 6 SEPARATELY COMPILED UNITS
C (UNIT 20 THRU UNIT 27) WITH ALL UNITS EXCEPT UNIT 20 BEING
C OVERLAY UNITS.
C PROGRAM EDUC ACTS AS A MEMORY RELEASE LOCATION. WHenever THE
C PROGRAM CONTROL RETURNS TO THIS UNIT, ALL OVERLAY UNITS ARE
C RELEASED FROM MEMORY PRIOR TO NEW UNITS BEING SUMMONED.
C CALLED BY: NONE
C CALLS :
C SUBROUTINE ASKQ(ASK)
C SUBROUTINE BGM
C SUBROUTINE CMDU
C SUBROUTINE HEADER
C SUBROUTINE INDEX
C SUBROUTINE INTRO
C SUBROUTINE MODIF
C SUBROUTINE OBMDU
C SUBROUTINE OPT
C SUBROUTINE OPTION
C SUBROUTINE PIVOT
C SUBROUTINE QUESTN
C SUBROUTINE READY
C SUBROUTINE TDISPL
C VARIABLES:
C USED: ASK,BGM,IBTAB,IFLAG(1),IFLAG(7),IFLAG(10),ITAB,KFA,
C MOD,NEC,NMC,OPT,OUT,PN
C MODIFIED: BASIC,C(4),DUAL,IFLAG(2),P(4),PES
C $USES UCHECK2 IN UNIT27.CODE OVERLAY
C $USES UDISPL IN UNIT26.CODE OVERLAY
C $USES UHEADER IN UNIT25.CODE OVERLAY
C $USES UPIVJT IN UNIT24.CODE OVERLAY
C $USES UDPT IN UNIT23.CODE OVERLAY
C $USES UEASY IN UNIT22.CODE OVERLAY
C $USES UMBMCU IN UNIT21.CODE OVERLAY
PROGRAM EDUC
CHARACTER V466,CN46,PW46,MM46,FN46,FINEG46,P46,OB46,W46
INTEGER ARTV,BASIC,PX,PKS,PR,PRS,OPT,VT,CS,PES,OUT,OUTP,
,TIE,FMT,ASK
COMMON/E1/A(20,60),ARTV:20),C(60),1,INRM(20),IFLAG(10),CB(20)
**Routine which allows user to change defaults called**

**Call option**

**Routine which initializes variables and reads model called**

**Call intro**

**D determines whether use or algorithm to place in tabular form**

IF(MOD .EQ. 1) THEN

**User to place in tabular form**

CALL OBMDU

CALL CANDU

ELSE

**Algorithm to place in tabular form**

CALL MODFA

ENDIF

**Routine which adds slack, surplus, and artificial variables called**

CALL INDEX

IF(MOD .EQ. 1) THEN

**Routine which questions user on tableau form called**

CALL READY

ELSE

DO 200 J=KFA,VT

C(i)=SM

200 CONTINUE

ENDIF

**Checks if artificial variables have been added**

IF(NEC+RGC) .NE. 0 THEN

CALL BIGM

ENDIF

**If user has selected screen output and also to select pivot**

**Elements, routine warns user to study tableaus carefully**

IF(PES .NE. 3) AND OUTP .EQ. 1 THEN

**Call questn**

ENDIF

WRITE(1,110)CHAR(12)

IF(PES .EQ. 3) THEN

IF(NEC-RGC) .NE. 0 THEN

WRITE(1,220)

220 FORMAT(11I),8X,'INITIAL BASIC SOLUTION',*/))

ELSE

WRITE(1,240)

240
C BASIC SOLUTION COUNTER INCREMENTED
280 BASIC=BASIC+1
   IF(BASIC .LT. 1) THEN
      WRITE(2,290) PN, BASIC
   END IF
C ROUTINE CALLED WHICH DETERMINES EXISTENCE OF OPTIMALITY
300 CALL OPT
   IF(OPTS .EQ. 1 OR IFLAG(7) .EQ. 1 OR IFLAG(1) .EQ. 0) THEN
      C TABLEAU IS EITHER OPTIMAL, UNBOUNDED, OR INFEASIBLE
320 WRITE(1,110)CHAR(12)
      WRITE(1,'(10I1,)','Would you like to perform further pivots?'
      READ(*)(CHAR(12),I=1,N)
      IF(CHAR(FI) .EQ. 78) THEN
         C ROUTINE ALLOWS EXIT FROM MODULE
         CALL ASK(ASK)
         IF(ASK .EQ. 1) THEN
            GO TO 130
         END IF
      ELSEIF(CHAR(FI) .EQ. 99) THEN
         IF(DUAL .EQ. 1) THEN
            PES=2
         DUAL=2
         WRITE(1,110)CHAR(12)
         WRITE(1,'((,))','The option of performing dual pivots has'
      ENDIF
      ELSEIF(CHAR(FI) .EQ. 89) THEN
         IF(DUAL .EQ. 1) THEN
            PES=2
         DUAL=2
         WRITE(1,110)CHAR(12)
         WRITE(1,'((,))','The option of performing dual pivots has'
      ENDIF
      ELSEIF(CHAR(FI) .EQ. 99) THEN
         IF(DUAL .EQ. 1) THEN
            PES=2
         DUAL=2
         WRITE(1,110)CHAR(12)
         WRITE(1,'((,))','The option of performing dual pivots has'
      ENDIF
      ELSEIF(CHAR(FI) .EQ. 89) THEN
         IF(DUAL .EQ. 1) THEN
            PES=2
90 DUAL=2
         WRITE(1,110)CHAR(12)
         WRITE(1,'((,))','The option of performing dual pivots has'
      ENDIF
"BEEN ACTIVATED AT THIS TIME.",9(1))"
ENDIF
ELSE
   WRITE(1,'(/5x,"INVALID ENTRY, PLEASE REENTER"/))
   PAUSE
   GO TO 320
ENDIF
ENDIF
CALL PIVOT
IF(FLAG(10) .EQ. 5) THEN
   C FLAG INDICATES FURTHER PIVOTS NOT DESIRED OR ALLOWED
   CALL ASKQ(ASK)
   IF(ASK .EQ. 1) THEN
      93 TO 130
   ENDIF
   ENDIF
WRITE(1,110)CHAR(12)
C DETERMINES IF INTERMEDIATE TABLEAU TO BE DISPLAYED
IF(IBTAB .NE. 0) THEN
   IF((FLOAT(BASIC-1)/(FLOAT(IBTAB))) .EQ. FLOAT((BASIC-1)/IBTAB)) THEN
      IF(OUTP .EQ. 1) THEN
         OPEN(2,FILE='CONSOLE;'
      ELSE
         OPEN(2,FILE='PRINTER;'
      ENDIF
      60 TO 280
      ELSE
         BASIC=BASIC+1
      60 TO 300
   ENDIF
   ELSE
      BASIC=BASIC+1
      90 TO 300
   ENDIF
   STOP
END
SUBROUTINE CNODULE

USE: DISPLAYS OBJECTIVE FUNCTION AS INPUT IN MODULE 1, SOLICITS
RESPONSE TO MENU OF OPTIONS OF OPERATIONS WHICH MAY BE
APPLIED TO PLACE OBJECTIVE FUNCTION IN MAXIMIZATION AND Z-X=0
FORM. PROVIDES IMMEDIATE FEEDBACK AND DISPLAYS OBJECTIVE
FUNCTION AFTER PROPER MODIFICATION. ONLY UTILIZED IF USER
ELECTS TO PERFORM OBJECTIVE FUNCTION MODIFICATION.

CALLED BY: PROGRAM EUCD

CALLS : SUBROUTINE CHECK2(P,N,N,INVAL,NEW)

VARIABLES:

USED: IFLAG(5), INW, INVAL, NM, V, VN(1)

MODIFIED: C(N), P(N), T

USES UCHECK2 IN UNIT27. CODE OVERLAY

SUBROUTINE CNODULE

CHARACTER VN(6), PN(20), MN(3), FN(IO), PIN(2), PN(2), VN(10)
INTEGER ARTV, BASIC, PKS, PRS, OPTS, V, VT, CB, PES, DLU, DOUTP,
,TIE, FMT, T
COMMON/EI(20, 60), ARTV(20), C60), I, INER(20), IFLAG(10), CB(20)
, MEC, NFC, MLA, IAE, INDEXE, INDEXG, INDEXL, IB(20)
COMMON/E2/BASIC, K, KFA, KFS, KFLS, OPTS, PKS, PRS, V, VT, MNN
COMMON/E3/ MOD, PES, DLU, DOUTP, TAB, ID, AN, IFLAG, TIE, FMT
COMMON/E4/VN(20), C20), PN, NN, F, PNEB(20), P(10), OBNN

100 WRITE(111, CHAR(12))
110 FORMAT(A)

OBJECTIVE FUNCTION DISPLAYED AS INPUT BY USER
WRITE(1, '('4X, 'OBJECTIVE FUNCTION MODIFICATION', A/')', 'THE OBJECT
IVE FUNCTION, AS ENTERED, WILL', 'BE DISPLAYED NEXT. AFTER THE D
ISPLAY, ', 'YOU WILL BE ASKED TO SELECT THE OPTION'),
WRITE(1, ('WHICH WILL TRANSFORM THE OBJECTIVE', 'FUNCTION INTO T
HE PROPER TABLEAU FORM', 'FOR THE SIMPLEX ALGORITHM'), '/7(//))
PAUSE
WRITE(111, CHAR(12))
WRITE(1, 120)
120 FORMAT(A, 'OBJECTIVE FUNCTION MODIFICATION'/7X, 'PRESENT OBJECTIVE
FUNCTION')

NUMBER OF 80 COLUMN DISPLAYS REQUIRED DETERMINED
T=(V-1)/5+1
DO 300 M=1, T
IF(IFLAG(5), .EQ. 1) THEN
C VARIABLE NAMES PRINTED AS COLUMN HEADERS
WRITE(1, ' ('IOX', ')')
DO 180 J=(MN), 4, MN
IF(J .LT. V) THEN
60 .00 180
ENDIF
WRITE(1, 130) VN(J)
FORMAT(7X, A6, 1X, A)

300 CONTINUE

283
180 CONTINUE
WRITE(1,'(" ")')
ENDIF
WRITE(1,'(1X,5F12)')
DO 250 J=(N5)-4,N5
IF(J.GT. V)THEN
GO TO 250
ENDIF
WRITE(1,200)
200 FORMAT(I4,1X,12,I1),2X,2X
DO 250 Jz(N)-4,N5
IF(J.GT. V)THEN
ENDIF
IF(FMT .EQ. 0)THEN
ELSE
WRITE(I,'(IX,''+',' I3X.F11.4,$)' IC(J))
ENDIF
280 CONTINUE
WRITE(1,'(" ")')
300 CONTINUE
PAUSE
WRITE(1,110)CHAR(12)
WRITE(1,330)
C OPTIONS DISPLAYED WHICH USER SELECTS THE ONE WHICH PROPERLY
C MODIFIES OBJECTIVE FUNCTION
320 FORMAT(4X,'OBJECTIVE FUNCTION MODIFICATION')
WRITE(I,'(//,"PLACE THE OBJECTIVE FUNCTION IN THE"/PROPER F
.OORMAT FOR THE SIMPLEX ALGORITHM "/"/ WHICH OF THE FOLLOWING SHOULD
BE DONE?"/")
WRITE(I,'(//,"1. ADD -C(J) TO BOTH SIDES OF EQUATION."/"2. MULTI
.LY EQUATION BY -1 AND THEN ADD"/"3. NO CHANGES ARE NECESSARY."/"))
330 WRITE(1,340)
340 FORMAT(7X,'WHICH OPTION IS CORRECT? ',I1)
READ(5,'(AI)')IP(1)
CALL CHECK2(P,1,3,INVAL,INMEM)
IF(INVAL .EQ. 1)THEN
WRITE(1,360)
360 FORMAT(5X,'INVALID ENTRY, PLEASE REENTER')
GO TO 330
ENDIF
C OBJ FUNCTION PROPERLY MODIFIED AND USER RECEIVES FEEDBACK.
IF(MINM .EQ. 1) THEN
  DO 370 J=1,10
  C(J) = C(J)
  CONTINUE
  IF(MINM .EQ. 1) THEN
    WRITE(1,380) INEM
  ENDIF
  ELSE
    WRITE(1,400) INEM
  ENDIF
380  FORMAT(10X,'OPTION #1, IS CORRECT.'//,
            PAUSE
                    "GO TO 500")
400  IF(MINM .EQ. 2) THEN
    WRITE(1,380) INEM
    PAUSE
            "GO TO 500" ENDIF
    ELSEIF(MINM .EQ. 2) THEN
      IF(MINM .EQ. 2) THEN
        WRITE(1,380) INEM
        PAUSE
                "GO TO 500"
      ENDIF
    ELSE
      WRITE(1,420) INEM
420  FORMAT(9X,'OPTION #2, IS INCORRECT.'//,
                  THE PROPER RESPONSE WAS OPTION #2.'//,
                                PAUSE
            "GO TO 500") ENDIF
500  WRITE(1,110) CHAR(12)
510  IF(IFLAG(NI).EQ.1) THEN
      WRITE(1,'(101,"")')
      DO 700 M=1,10
      C PROPERLY MODIFIED OBJ FUNCTION DISPLAYED
      WRITE(1,'(2X,"AFTER THE PROPER MODIFICATION, THE")')
          FUNCTION FORM IS:"',")
      T = (V/S)**1
      DO 700 M=1,10
      IF(IFLAG(M).EQ.1) THEN
        WRITE(1,'(101,"")')
        DO 550 J=(M+5)-4,M+5
        IF(J .GT. V) THEN
          WRITE(1,520) VM(J)
      ENDIF
520  FORMAT(7X,6E,11,1)
530  CONTINUE
      WRITE(1,'(" ")') ENDIF
      WRITE(1,'(10X,"")')
      DO 600 J=(M+5)-4,M+5
      IF(J .GT. V) THEN
        WRITE(1,570) J
570  CONTINUE
570 FORMAT(7X,'X(',12,'W',2X,4)
600 CONTINUE
WRITE(1,("''"'))
IF(N .EQ. 1) THEN
  IF(NMIN .EQ. 1) THEN
    WRITE(1,(2X,'MAX I',",",0))
  ELSE
    WRITE(1,(2X,'MAX (-Z)',",",0))
  ENDIF
ELSE
  WRITE(1,(101,V))
ENDIF
90 650 J=MB5-4,MB5
IF(J .GT. V) THEN
  GO TO 650
ENDIF
IF(FMT .EQ. 0) THEN
  WRITE(1,(XI,"*",XI,1PE11.4,"')C(J)
ELSE
  WRITE(1,(XI,"*",XI,F11.4,"')C(J)
ENDIF
656 CONTINUE
IF(N .EQ. 1) THEN
  WRITE(1,("---")
ELSE
  WRITE(1,("---")
ENDIF
700 CONTINUE
PAUSE
RETURN
END
SUBROUTINE OPTION

USE: DISPLAYS DEFAULT OPTION VALUES AND SOLICITS RESPONSE TO
CHANGE THESE DEFAULTS. IF OPTION IS SELECTED TO BE CHANGED,
USER REVIEWS MENU AND SELECTS DESIRED METHOD AND IS RETURNED
TO DEFAULT OPTION DISPLAY. SOME OPTIONS ARE CHANGED UPON
SELECTION DUE TO ONLY TWO METHOD BEING POSSIBLE. OPTIONS ARE
RESET TO PROGRAMMER SPECIFIED DEFAULT UPON EACH CALL TO THIS
SUBROUTINE.

CALLED BY: PROGRAM EDUC

CALLS : SUBROUTINE CHECK2(D.N,M,INVAL,INEN)

VARIABLES:

USED: INEN, INVAL

MODIFIED: D,DUAL,FMT,ITAB,IFTAB,MOD,OIU,OUTP,PES

SUBROUTINE OPTION

CHARACTER D(10) * 1
INTEGER PES,OIU,DUAL,OUTP,TIE,FMT
COMMON/E,I/MOD,PES,OIU,DUAL,OUTP,ITAB,IFTAB,BM,TIE,FMT
MOD=1
PES=1
DUAL=1
OIU=1
ITAB=1
IFTAB=1
OUTP=1
FMT=1

100 WRITE(I,110) CHAR(12)
110 FORMAT(A)
C

DEFAULT OPTIONS DISPLAYED

WRITE(I,'(1X,'"DEFAULT OPTIONS"/5X,"ENTER OPTION NUMBER TO CHAN
GE")')
WRITE(I,'("1.TABLEAU FORMATION",1X,8)')
IF(MOD .EQ. 1) THEN
  WRITE(I,'(6X,"USER")')
ELSE
  WRITE(I,'(1X,"ALGORITHM")')
ENDIF
WRITE(I,'("2.PIVOT ELEMENT SELECTION",5X,8)')
IF(PES .EQ. 1) THEN
  WRITE(I,'(1X,"USER SEL"/6X,"ALGOR CHK")')
ELSEIF(PES .EQ. 2) THEN
  WRITE(I,'(1X,"USER SEL"/5X,"NO CHECK")')
ELSE
  WRITE(I,'(1X,"ALGORITHM"/5X,"SELETS")')
ENDIF
WRITE(I,'("3.DUAL PIVOTS",26X,8)')
IF(DUAL .EQ. 1) THEN
  WRITE(I,'("N")')
ELSE
    WRITE(1,'("y")')
ENDIF
WRITE(1,'("4.INFEASIBLE,UNBOUNDED,OPTIMAL"/4X,"SELECTION IDENTIF-
ICATION",2X,9)')
IF(IQU .EQ. I)THEN
    WRITE(1,'(6X,"USER")')
ELSE
    WRITE(1,'(1X,"ALGORITHM")')
ENDIF
WRITE(1,'("5.TABLEAUS TO BE DISPLAYED"/4X,"INITIAL",19X,9)')
IF(TAB .EQ. 1)THEN
    WRITE(1,'(9X,"Y")')
ELSE
    WRITE(1,'(9X,"N")')
ENDIF
WRITE(1,'("6.OUTPUT LOCATION",13X,9)')
IF(OUTP .EQ. 1)THEN
    WRITE(1,'(4X,"SCREEN")')
ELSE
    WRITE(1,'(4X,"PRINTER")')
ENDIF
WRITE(1,'("7.OUTPUT FORMAT",15X,9)')
IF(FMT .EQ. 1)THEN
    WRITE(1,'(2X,"E FORMAT")')
ELSE
    WRITE(1,'(2X,"F FORMAT")')
ENDIF
WRITE(1,'("8.NO CHANGES"//"SEE DOCUMENTATION FOR EXPLAINATION
ATION"," WHICH OPTION (ENTER 1-8)? ",9)')
READ(S,9101)I
CALL CHECKCD,1,8,INVAL,INEW)
IF(INVAL .EQ. 1)THEN
    WRITE(1,'(5X,"INVALID ENTRY PLEASE REENTER")')
    GO TO 100
ENDIF
SOTC(130,220,270,300,400,500,530,560)INEW
130 WRITE(1,110)ICHAR(12)
C MODIFICATION OPTIONS DISPLAYED
WRITE(1,140)
140 FORMAT(2X,"EDUCATIONAL MODULE OPTION SELECTION")
WRITE(1,'("IN ORDER TO PLACE THE LP MODEL INTO THE"" PROPER FOR-
MATE FOR THE SIMPL TRAL ALGORITHM")" ADDITION OF "SLACK OR ARTIFICIAL VARIABLES), WHICH"" METHOD IS DESI-
RED?"")
VAITE(1,'USER SELECTS MODIFICATION AND ALGORITHM CHECK KS'//IBY,'OR'///2. ALGORITHM PERFORMS MODIFICATIONS.'/'DE (NO USER INPUT)'')

160 WRITE(1,170)
170 FORMAT(13X,'WHICH OPTION? ',$)
READ(5,'(A1)')D(1)
CALL CHECK2(D,1,2,INVAL,INEX)
IF(INVAL .EQ. 1) THEN
  WRITE(1,190)
190 FORMAT(5X,'INVALID ENTRY, PLEASE REENTER')
GO TO 160
ELSE
  MOD=INEX
ENDIF
GO TO 100
220 WRITE(1,110):CHAR(12)
C PIVOT ELEMENT SELECTION OPTIONS DISPLAYED
WRITE(1,140)
WRITE(1,'(13X,''IN SELECTION OF PIVOT ELEMENTS FOR THE''/''SIMPLEX ALG.
. ORITHM, WHICH METHOD WOULD''/''YOU LIKE?'')
WRITE(1,121),''1. USER SELECTS, ALGORITHM CHECKS.'/'2. MAY CH.
. ANSE SELECTION AFTER CHECK)''/''2. USEP SELECTS, NO ALGORITHM CHE.
. CK.'/'3. ALGORITHM SELECTS, NO USER INPUT.'')
240 WRITE(1,170)
READ(5,'(A1)')D(1)
CALL CHECK2(D,1,3,INVAL,INEX)
IF(INVAL .EQ. 1) THEN
  WRITE(1,190)
GO TO 240
ELSE
  PES=INEX
ENDIF
GO TO 100
270 WRITE(1,110):CHAR(12)
C DUAL PIVOT OPTIONS DISPLAYED
280 WRITE(1,140)
WRITE(1,'(13X,''WOULD YOU LIKE TO BE ABLE TO PERFORM''/''DUAL . PIVOTS? ''')
WRITE(1,141),''DUAL PIVOTS ARE ALLOWED ONLY IF USER''/''SELECTS PI.
. VOT ROW AND COLUMN ELEMENTS.'/'(Y/N,RETURN) '',$)
READ(5,'(A1)')D(1)
IF(ICHAR(D(1))) . EQ. 89 THEN
  DUAL=2
ELSEIF(ICHAR(D(1))) . EQ. 78 THEN
  DUAL=1
ELSE
  WRITE(1,180)
  GO TO 270
ENDIF
GO TO 100
310 WRITE(1,110):CHAR(12)
C IDENTIFICATION OF FINAL TABLEAU OPTIONS DISPLAYED

289
WRITE(1,140)
WRITE(1,140)
WRITE(1,'("AS OPTIMAL, INFEASIBLE, OR UNBOUNDED"/"SOLUTIONS OCC
UR, WHICH METHOD WOULD YOU"/"LIKE?")')
WRITE(1,'(2(1),"1. USER ATTEMPTS TO IDENTIFY, ALGORITHM"/3X,"CH
CHECKS"/"2. SYSTEM IDENTIFIES AND REPORTS AS"/3X,"OCCE":")')
WRITE(1,170)
READ(5,'(AI)')(1)
CALL CHECK2(l,2,INVAL,INNEW)
IF(INVAL .EQ. 1) THEN
WRITE(1,180)
GO TO 320
ELSE
OIU=INNEW
ENDIF
GO TO 100
WRITE(1,110)CHAR(12)
TABLEAU DISPLAY OPTIONS SHOWN
WRITE(1,140)
WRITE(1,'("WHICH TABLEAUS WOULD YOU LIKE Displayed?")')
WRITE(1,140)
WRITE(1,'('/5X,"INITIAL TABLEAU? (Y/N) "',%1')
READ(5,'(AI)')(1)
IF(ICHAR(D(1)) .EQ. 69) THEN
ITAB=1
ELSE IF(ICHAR(D(1)) .EQ. 78) THEN
ITAB=2
ELSE
WRITE(1,180)
GO TO 420
ENDIF
WRITE(1,140)
WRITE(1,'('/5X,"INTERMEDIATE TABLEAUS? (Y/N) "',%1')
READ(5,'(AI)')(1)
IF(ICHAR(D(1)) .EQ. 69) THEN
WRITE(1,'("EVERY N(TH) INTERMEDIATE TABLEAU WILL BE"/15X,"DI
PLAYED.")')
ELSE
WRITE(1,180)
GO TO 420
ENDIF
WRITE(1,140)
WRITE(1,'('/4X,"WHAT VALUE DO YOU DESIRE FOR N?"/17X,"N = "',
%,1')
READ(5,'(2AI)')(1),D(2)
CALL CHECK2(0.2,99,INVAL,INNEW)
IF(INVAL .EQ. 1) THEN
WRITE(1,180)
PAUSE
GO TO 450
ENDIF
ITAB=INNEW
ELSE IF(ICHAR(D(1)) .EQ. 78) THEN
ITAB=0
ELSE
WRITE(1,180)
GO TO 440
ENDIF
WRITE(1,140)
WRITE(1,'('/5X,"FINAL TABLEAU? (Y/N) "',%1')
READ(5,'(AI)')(1)
IF(ICHAR(D(I)) .EQ. 89) THEN
   IFTAB=1
ELSEIF(ICHAR(D(I)) .EQ. 78) THEN
   IFTAB=2
ELSE
   WRITE(1,180)
   GO TO 460
ENDIF
GO TO 100
C OUTPUT LOCATION OPTION CHANGED
500 IF(OUTP .EQ. 1) THEN
   OUTP=2
ELSE
   OUTP=1
ENDIF
GO TO 100
C OUTPUT FORMAT CHANGED
530 IF(FMT .EQ. 1) THEN
   FMT=0
ELSE
   FMT=1
ENDIF
GO TO 100
560 RETURN
END
SUBROUTINE READY

C USE: DISPLAYS QUESTIONS CONCERNING WHETHER OR NOT THE TABLEAU
C WHICH IS DISPLAYED IS AN INITIAL BASIC SOLUTION AND READY FOR
C THE INITIAL PIVOT OF THE SIMPLEX ALGORITHM. IF ARTIFICIAL
C VARIABLES HAVE BEEN ADDED, THE "BIG M" METHOD MUST HAVE BEEN
C APPLIED TO THE VARIABLES SPECIFIED BY USER. USER IS GIVEN
C IMMEDIATE FEEDBACK AND INSTRUCTIONAL COMMENTS WITH BOTH
C CORRECT AND INCORRECT RESPONSES. ROUTINE MODIFIES TABLEAU
C CORRECTLY REGARDLESS OF USER INPUT.
C CALLED BY: PROGRAM EDUC
C CALLS : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)
C SUBROUTINE TDISPL
C VARIABLES:
C USED: BM, INEW, INVAL, KFA, NEC, NSC, VT
C MODIFIED: C(1), IFLAG(3), IFLAG(9), P(4)

C    MODULE 2 UNIT22
C    UNIT USES: UNIT26 AND UNIT27
C    SUBROUTINE READY
C USE: DISPLAYS QUESTIONS CONCERNING WHETHER OR NOT THE TABLEAU
C WHICH IS DISPLAYED IS AN INITIAL BASIC SOLUTION AND READY FOR
C THE INITIAL PIVOT OF THE SIMPLEX ALGORITHM. IF ARTIFICIAL
C VARIABLES HAVE BEEN ADDED, THE "BIG M" METHOD MUST HAVE BEEN
C APPLIED TO THE VARIABLES SPECIFIED BY USER. USER IS GIVEN
C IMMEDIATE FEEDBACK AND INSTRUCTIONAL COMMENTS WITH BOTH
C CORRECT AND INCORRECT RESPONSES. ROUTINE MODIFIES TABLEAU
C CORRECTLY REGARDLESS OF USER INPUT.
C CALLED BY: PROGRAM EDUC
C CALLS : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)
C SUBROUTINE TDISPL
C VARIABLES:
C USED: BM, INEW, INVAL, KFA, NEC, NSC, VT
C MODIFIED: C(1), IFLAG(3), IFLAG(9), P(4)

C USES UCHECK2 IN UNIT27.CODE OVERLAY
C USES UDISPL IN UNIT26.CODE OVERLAY

SUBROUTINE READY
CHARACTER VN(6), CN(6), FN(20), MN(10), PN(10), OBJ(10)
INTEGER ARTY, BASIC, Pk, FK, PR, PSL, OPTS, V, VT, CB, PES, OIU, DUAL, OUTP,
.TIE, FKT
COMMON/E1/A(20, 60), ARTV(20), C(60), I, INED(20), IFLAG(10), CB(20),
.NEC, NEC, ASC, IA, INDEI, INDEI, INDEI, IB(20)
COMMON/E2/BASIC, xKFA, KFSA, KFSU, OPTS, PK, FK, PR, XVT, NMN
COMMON/E3/VOC, RES, OIU, DUAL, OUTP, ITAB, IBTAB, BM, TIE, FKT
COMMON/E4/VN(2): N(20), P(10), PM, MM, FN, PINE(20), P(10), OBJ: N

WRITE(1, (110))CHAR(12)
110 FORMAT(A)
WRITE(1, '(A4,1),''THE TABLEAU AS MODIFIED PREVIOUSLY, ''WILL BE D
DISPLAYED, ''YOU WILL THEN BE ASKED IF THE TABLEAU IS ''IN THE
CORRECT FORM FOR THE SIMPLEX ''ALGORITHM. ''/)
PAUSE
WRITE(1, (110))CHAR(12)
C FLAGS ALLOW DISPLAY OF TABLEAU WITHOUT BASIC VARIABLES ANNOTATED
IFLAG(3)=1
IFLAG(9)=2
OPEN(2, FILE='CONSOLE:')
CALL TDISPL
WRITE(1, (110))CHAR(12)
WRITE(1, 'A6,1),''IS THE TABLEAU IN THE PROPER FORM FOR''/101,11
THE INITIAL PIVOT? ''/411
READ(5, 'A11')P(1)
IF((NEC+NSC).EQ.0)THEN
  IF((CHAR(P(1)).EQ. B9)THEN
    WRITE(1, (130))
130 FORMAT(A71, ''YOUR RESPONSE WAS CORRECT. ''/11, ''THE TABLEAU IS IN
THE PROPER FORM. ''1)}
PAUSE
60 TO 300
ELSEIF(ICHAR(P(1)) .EQ. 78) THEN
WRITE(1,150)
150 FORMAT(/, 'YOUR RESPONSE WAS INCORRECT.' )
WRITE(1,160)
160 FORMAT(/, 'ONLY SLACK VARIABLES HAVE BEEN ADDED, 50/2X,' NO FURTHER MODIFICATIONS ARE NEEDED.' /5X, 'YOU PRESENTLY HAVE AN INITIAL BASIC/15X, 'SOLUTION.' /)
PAUSE
60 TO 300
ELSE
WRITE(1,160)
160 FORMAT(/, 'INVALID ENTRY, PLEASE REENTER')
WRITE(1,220)
220 FORMAT(/, 'INTEGIAL VARIABLES HAVE BEEN ADDED, YET THE OBJECTIVE FUNCTION HAS NOT BEEN MODIFIED (BIG M) TO REFLECT THIS.' )
PAUSE
60 TO 300
ELSEIF(ICHAR(P(1)) .EQ. 89) THEN
WRITE(1,150)
150 FORMAT(/, 'YOUR RESPONSE WAS CORRECT.' /5X, 'FURTHER MODIFICATIONS ARE REQUIRED.' /)
PAUSE
60 TO 300
ELSEIF(ICHAR(P(1)) .EQ. 7B) THEN
WRITE(1,220)
220 FORMAT(/, 'INTEGIAL VARIABLES HAVE BEEN ADDED, YET THE OBJECTIVE FUNCTION HAS NOT BEEN MODIFIED (BIG M) TO REFLECT THIS.' )
PAUSE
60 TO 300
ELSE
WRITE(1,150)
150 FORMAT(/, 'INTEGIAL VARIABLES HAVE BEEN ADDED, YET THE OBJECTIVE FUNCTION HAS NOT BEEN MODIFIED (BIG M) TO REFLECT THIS.' )
PAUSE
60 TO 300
ENDIF
300 IF(NEC+NEC) .NE. 0 THEN
WRITE(1,110) CHAR(12)
WRITE(1,120) CHAR(12)
210 FORMAT(/, 'WHICH VARIABLES REQUIRE THE USE OF THE BIG M METHOD?'/, (ENTEF SUBSCRIPT VALUES) )
WRITE(1,'(9X,''FIRST VARIABLE? '',?')
READ(5,'(2A1)'P(1),P(2)
CALL CHECK2(P,2,VT,INVAL,INEW)
IF(INVAL.NE.1)THEN
  WRITE(1,180)
  PAUSE
  GO TO 310
ENDIF
IF(INEW.NE.KFA)THEN
  WRITE(1,340)KFA
  FORMAT(//7X,'''YOUR RESPONSE WAS CORRECT. '/1X,12,''' IS THE FIRST
          ARTIFICIAL VARIABLE AND'/1X,'''REQUIRES THE USE OF THE BIG M.'')
ELSE
  WRITE(1,360)KFA
  FORMAT(//6X,'''YOUR RESPONSE WAS INCORRECT. '/1X,'''THE CORRECT RES
          PONSE WAS VARIABLE ',12,''' THIS IS THE FIRST ARTIFICIAL VARIABLE AND'
          ',12,'''REQUIRES THE USE OF THE BIG M METHOD.'')
ENDIF
PAUSE
380  WRITE(1,110)CHAR(12)
WRITE(1,'(9X,''VARIABLES '',12,''' REQUIRE THE''/13X,'' BIG M METHOD''/1X)'
      KFA
WRITE(1,'(9X,''LAST VARIABLE? '',?')
READ(5,'(2A1)'P(1),P(2)
CALL CHECK2(P,2,VT,INVAL,INEW)
IF(INVAL.NE.1)THEN
  WRITE(1,180)
  PAUSE
  GO TO 380
ENDIF
IF(INEW.NE.VT)THEN
  WRITE(1,400)VT
  FORMAT(//7X,'''YOUR RESPONSE WAS CORRECT. '/1X,'''THE LAST ARTIFICI
          AL VARIABLE IS 0',12,''' AND'/2X,'''IS THE LAST TO REQUIRE THE USE O
          F THE''/14X,'''BIG M METHOD.'')
  PAUSE
ELSE
  WRITE(1,420)VT
  FORMAT(//6X,'''YOUR RESPONSE WAS INCORRECT. '/1X,'''THE LAST ARTIFICI
          AL VARIABLE IS 0',12,''' AND'/2X,'''IS THE LAST TO REQUIRE THE USE O
          F THE''/14X,'''BIG M METHOD.'')
  PAUSE
ENDIF
DO 450 J=KFA,VT
  C(J)=0
  CONTINUE
WRITE(1,110)CHAR(12)
WRITE(1,'(10X),11,'''THE TABLEAU WILL BE DISPLAYED, THEN YOU''/2
     ,11,'''WILL BE ASKED IF IT IS IN THE PROPER''/6X,'''FORM FOR THE INITI
     AL Pivot.'')
PAUSE
C FLAGS ALLOW TABLEAU WITH BASIC VARIABLES ANNOTATED TO BE
C

DISPLAYED
IFLAG(3)=1
IFLAG(9)=0
OPEN(2,FILE='CONSOLE: ')
WRITE(1,110)CHAR(12)
CALL TDISPL
480 WRITE(1,110)CHAR(12)
WRITE(1,'(1X,"IS THE TABLEAU IN THE PROPER FORM FOR"/10X,"THE"
.INITIAL PIVOT? ",")')
READ(5,'(A1)'P(1)
IF(CHAR(P(1)) .EQ. 78) THEN
WRITE(1,'(1X,"YOUR RESPONSE WAS CORRECT")')
WRITE(1,500)
500 FORMAT('THERE IS NO INITIAL BASIC SOLUTION SINCE',"THE OBJECTIVE"
.FUNCTION COEFFICIENTS OF "THE ARTIFICIAL VARIABLES ARE NOT ZERO"
..")')
PAUSE
ELSEIF(CHAR(P(1)) .EQ. 89) THEN
WRITE(1,'(1X,"YOUR RESPONSE WAS INCORRECT")')
WRITE(1,500)
PAUSE
ENDIF
ENDIF
RETURN
END
SUBROUTINE CNMDU

USE: DISPLAYS THE CONSTRAINTS AND A MENU OF OPTIONS WHICH MAY BE APPLIED TO CONSTRAINTS TO PREPARE THE CONSTRAINT FOR THE SIMPLEX ALGORITHM (ADD VARIABLES, MULTIPLY BY -1). SOLICITS USER INPUT AND PRESENTS IMMEDIATE FEEDBACK. IF INCOMPLETE RESPONSE GIVEN, BRIEF INSTRUCTIONAL COMMENTS ARE DISPLAYED. CONSTRAINT IS MODIFIED CORRECTLY REGARDLESS OF USER INPUT. CONSTRAINTS WITH NEGATIVE PHS'S ARE MULTIPLIED BY -1.

CALLED BY: PROGRAM EDUC

CALLS : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)

SUBROUTINE TDISP

VARIABLES:

USED: CN(1),IFLAG(5),INEM.INVAL,1,V,VM(1)

MODIFIED: A(N,1),IFLAG(3),IFLAG(9),INEQ(4),NEC,NLC,PINEQ(4),

C(1),S,T,VT,XB(4)

SUBROUTINE CNMDU

CHARACTER VN6,CN16,FN16,MM16,FN10,PINEG11,PI1,OBJMN10
INTEGER ARTY,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB3,PES,OIU,DUAL,OUTP,
.II.FMIT.T,S

COMMON/E1/A(20,60),ARTV(20),C(60),I,INEQ(20),IFLAG(10),CE(20)
..NEC,NLC,NLC,IA,INDEXE,INDEXL,INDEXAC(20)

COMMON/E2/BASIC,K,KFA,KFS,KFS,UPTS,PK,PK,S,PRS,V,VT,MHM

COMMON/E3/MOE,PES,OIU,DUAL,OUP,ITAB,ISTAB,ITAB,ITAB,INDEXP

COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN

CALLS ONLY CONSTRAINTS TO BE DISPLAYED

IFLAG(9)=1

OPEN1,FILE='CONSOLE';

WRITE(1,110)CHAR(12)

110 FORMAT(A)

WRITE(1,'(6X,'"CONSTRAINT MODIFICATION"',8X,"THE CONSTRAINTS, AS ENTERED, WILL BE"/
"DISPLAYED NEXT. AFTER THE DISPLAY, YOU"/
"WILL BE SHOWN EACH OF THE CONSTRAINTS")')

WRITE(1,'("INDIVIDUALLY AND ASKED TO SELECT THE""OPTION WHICH"
TRANSFORMS THE CONSTRAINT""INTO THE PROPER SIMPLEX ALGORITHM FORM")')

PAUSE

WRITE(1,110)CHAR(12)

SET VARIABLES IN FORM APPROPRIATE FOR TDISP SUBROUTINE

VTW

FLAG INSURES SCREEN IS CLEARED AFTER EACH 60 COLUMN DISPLAY

IFLAG(3)=1

CALL TDISP

WRITE(1,110)CHAR(12)

WRITE(1,'("EACH CONSTRAINT WILL BE SEPARATELY""DISPLAYED, THEN"
THE FOLLOWING OPTIONS""WILL BE DISPLAYED FOR EACH CONSTRAINT.")')

WRITE(1,'("YOU WILL SELECT THE OPTION WHICH WILL", "PLACE THE C"
CONSTRAINT IN THE PROPER""SIMPLEX ALGORITHM FORM")')

296
WRITE(1,150)
150 FORMAT('1. ADD SLACK VARIABLE ONLY.'// '2. SUBTRACT SURPLUS VARIABLE.'// '3. ADD ARTIFICIAL VARIABLE.'// '4. ADD ARTIFICIAL VARIABLE ONLY.'// '')
WRITE(1,160)
160 FORMAT('4. MULTIPLY BY -1, SUBTRACT SURPLUS'/3., 'VARIABLE, ADD ARTIFICIAL VARIABLE.'// '5. MULTIPLY BY -1, ADD SLACK VARIABLE.'// '6. MULTIPLY BY -1, ADD ARTIFICIAL'/3., 'VARIABLE.'// '))
PAUSE
WRITE(1,110)CHAR(12)
T=(I/V/5)+1
DO 900 I=1,T
DO 400 N=1,T
IF(IFLAG(5).EQ.1)THEN
WRITE(I,'(13X,8)')
DO 270 J=(N+5)-4,N+5
IF(J .ST. V)THEN
39 TO 270
ENDIF
WRITE(1,260)VN(J)
FORMAT(6X,6E,1X,4)
270 CONTINUE
WRITE(I,'(13X5)')
ENDIF
WRITE(I,'(13X4)')
DO 290 J=(N+5)-4,N+5
IF(J .ET. V)THEN
60 TO 290
ENDIF
WRITE(1,280)J
FORMAT(6X,'X1',12,'2X,4)
290 CONTINUE
IF(T .EQ. 1 OR N .EQ. T)THEN
WRITE(1,300)
FORMAT(7X,'RHS')
ELSE
WRITE(I,'(13X5)')
ENDIF
WRITE(I,340)I
FORMAT('CNW',12,4)
IF(IFLAG(5).EQ.1)THEN
WRITE(1,350)CN(1)
FORMAT(7X,'A0.1X,4)
ELSE
WRITE(1,'(8)')
ENDIF
370 J=(N+5)-4,N+5
IF(J .ET. V)THEN
60 TO 370
ENDIF
IF(FMT .EQ. 0)THEN
WRITE(1,'(1X,IPE12.5)')A(I,J)
ELSE
  WRITE(1, '(1x,'A(1,1))
ENDIF
370 CONTINUE

IF(T .EQ. 1 .OR. N .EQ. 1) THEN
  IF(WFNT .EQ. 0W) THEN
    ELSE
      WRITE(1, '(1x,'')PINEU(1)
  ENDIF
  ELSE
    WRITE(1, 't''''')
ENDIF

CONTINUE

PAUSE
410 WRITE(1,110)CHAR(12)
WRITE(1, '(1x,'')CONSTRANooth,12,/')I1
WRITE(1,150)
WRITE(1,160)
WRITE(1,420)
420 FORMAT('/13x,''WHICH OPTION? '','$)
READ(S,''AI'')P(1)
CALL CHECK2(P,1,6,NVAL,INEW)
IF(NVAL .EQ. 1) THEN
  WRITE(1,430)
430 FORMAT('/10x,''INVALID ENTRY, PLEASE REENTER'')
  PAUSE
  GO TO 410
ENDIF
C CHECKS FOR NEGATIVE NMS
IF(XB(I) .LT. 0) THEN
  IF(INEW .EQ. 0) THEN
    USER SELECTION CHECKED AND FEEDBACK PROVIDED
    IF(INEW .EQ. 1) THEN
      WRITE(1,450)INEW
      450 FORMAT('/10x,''OPTION 0',11,' IS CORRECT.'/)
      PAUSE
      GO TO 800
    ELSE
      S=1
      WRITE(1,460)INEW,S
      460 FORMAT('/9x,''OPTION 0',11,' IS INCORRECT',/1x,'THE PROPER R')
      RESPONSE WAS OPTION 0',11,)
      PAUSE
      GO TO 800
    ENDIF
  ELSEIF(INEW .EQ. 1) THEN
    IF(INEW .EQ. 2) THEN
      WRITE(1,450)INEW
      PAUSE
      GO TO 800
    ELSE
      ENDIF
  ENDIF
S+2
WRITE(1,460)INEW,S
PAUSE
GO TO 800
ENDIF
ELSEIF(INEW .EQ. 3) THEN
WRITE(1,450)INEW
PAUSE
GO TO 800
ELSE
S=3
WRITE(1,460)INEW,S
PAUSE
GO TO 800
ENDIF
ELSE
IF(INEW(I) .EQ. 0) THEN
IF(INEW .EQ. 4) THEN
WRITE(1,450)INEW
PAUSE
ELSE
S=4
WRITE(1,460)INEW,S
PAUSE
ENDIF
ELSEIF(INEW(I) .EQ. 1) THEN
IF(INEW .EQ. 5) THEN
WRITE(1,450)INEW
PAUSE
ELSE
S=5
WRITE(1,460)INEW,S
PAUSE
ENDIF
ELSEIF(INEW .EQ. 6) THEN
WRITE(1,450)INEW
PAUSE
ELSE
S=6
WRITE(1,460)INEW,S
PAUSE
ENDIF
ENDIF
C CONSTRAINTS WITH NEGATIVE RHS MULTIPLIED BY -1
IB(I)=IB(I)
DO 500 J=1, V
A(I,J)=A(I,J)
500 CONTINUE
C COUNT OF INEQUALITIES BY TYPE CORRECTED DUE TO MUL BY -1
IF(INEW(I) .EQ. 0) THEN
MLC=MLC-1
NGC=NGC-1
ENDIF
IF (IMED(I) = '1') THEN
  INCL = INCL + 1
ELSEIF (IMED(I) .EQ. 1) THEN
  NBG = NBG - 1
  NLC = NLC + 1
  IMED(I) = 0
  PINEQ(I) = '='
ENDIF
800 WRITE(1, 110) CHAR(12)
900 CONTINUE
RETURN
END
SUBROUTINE OPT

USE: DETERMINES THE PIVOT ELEMENT, OPTIMALITY, UNBOUNDEDNESS, AND FEASIBILITY OF THE CURRENT TABLEAU. DEPENDENT ON THE INTERACTION OPTIONS SELECTED BY USER, THE USER WILL BE ASKED QUESTIONS ON THE ABOVE CONDITIONS AND PRESENTED FEEDBACK ACCORDINGLY. ALSO DEPENDENT UPON OPTION SELECTION, THE PIVOT ELEMENT MAY BE THAT SELECTED BY THE ALGORITHM OR AS INPUT BY USER. THIS ROUTINE ALSO DISPLAYS A TABLEAU HEADERS ON THE SELECTED OUTPUT DEVICE FOR DESIGNATED TABLEAUS. USER MAY SELECT PIVOT ELEMENT TO CAUSE SYSTEM OVERFLOW ERROR, BUT USER IS GIVEN OPTION TO ABORT PIVOT PRIOR TO OVERFLOW ERROR.

CALLED BY: PROGRAM EDUC

CALLS : SUBROUTINE BASISIS
SUBROUTINE TCAL
SUBROUTINE TDISPL

VARIABLES:

USES UCHECK2 IN UNIT27.CODE OVERLAY
USES LTDISPL IN UNIT26.COEV OVERLAY

SUBROUTINE OPT

INTEGER ARTV, BASIC, PK, PYS, PR, OPTS, I, J, CB, FES, OUT, Dual, OUTP, .TIE, FMT
COMMON/I/I(2), 60), ARTV(2), CB(60), CF(20), IFLAG(100), CB(20)
..NOC, PCIe, LA, INDEX, INDEX1, INDEX2, INDEX3)
COMMON/EX/BASIC, PK, FES, FES1, FES2, OUT, OPTS, PK, PRS, V, VT, MINN
COMMON/EX/MOD, PK, PK, OUT, ITAB, ITAB, BM, TIE, FMT
IFLAG(1)=0
IFLAG(4)=0
IFLAG(6)=0
INF1=0
SMAP=0.0

CHECKS FOR INFEASIBILITY
DO 100 I=1, K
IF (IB(I) .LT. 0.0) THEN
FLAG DENOTES VARIABLE AT NEGATIVE LEVEL
IFLAG(I)=1
ENDIF
IF (CB(I) .GE. KFA) THEN
DETERMINDS IF VARIABLE IS AN ARTIFICIAL VARIABLE
IF (IB(I) .GE. 0.0) THEN
FLAG DENOTES ARTIFICIAL VARIABLE AT POSITIVE LEVEL
INF1=1
ENDIF
ENDIF

100
CONTINUE

C FINDS THE LARGEST NEGATIVE Z(J)-C(J)
50 200 J=1,VT
IF(C(J) .LT. GNEB)THEN
   TIE=0
   GNEB=C(J)
   PK=J
ELSEIF(C(J) .EQ. GNEB)THEN
   TIE EXISTS FOR ENTERING BASIC VARIABLE
   TIE=1
ENDIF
200 CONTINUE

IF(GNEB .GT. -.0001 .AND. IFLAG(I) .EQ. 0)THEN
   NO NEGATIVE Z(J)-C(J) EXISTS AND RHS'S ARE POSITIVE
   SO OPTIMAL
   OPTS=1
ENDIF
C = 'CORRECT'
INC='INCORRECT'
GEB='OPTIMAL'
NUN='NOT OPTIMAL'

210 IF(DIU .EQ. 1)THEN
   USER HAS SELECTED TO IDENTIFY OPTIMALITY
   WRITE(1,220)CHAR(12)
220 FORMAT(A)
   WRITE(1,510)
   WRITE(1,'(5X,''WAS THE PREVIOUS TABLEAU OPTIMAL? '','I)')
   READ(5,'(A1)')CHAR(F)
   IF(ICHAR(F) .EQ. 84)THEN
      USER ELECTS TO REVIEW TABLEAU
      CALL TCAL
      GO TO 210
   ELSEIF(ICHAR(F) .NE. 89 .AND. ICHAR(F) .NE. 78)THEN
      WRITE(1,230)
      WRITE(1,510)
      WRITE(1,'(5X,''INVALID ENTRY, PLEASE REENTER'','//)')
      GO TO 210
   ENDIF
   USER INPUT CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
   IF(OPTS .EQ. 1)THEN
      IF(ICHAR(F) .EQ. 89)THEN
         WRITE(1,240)CHD,DB
      ELSE
         WRITE(1,510)
ELSEIF(OPTS .EQ. 0) THEN
  IF(ICHAR(F) .NE. 78) THEN
    WRITE(1,240) CD, NNU
  ELSE
    WRITE(1,240) INC, NNU
    WRITE(1,'(1X,2X,"TO BE OPTIMAL, ALL C(J) AND RHS VALUES MUST
      BE ZERO OR POSITIVE. THIS IS CURRENTLY NOT TRUE, THEREFORE THE
      TABLEAU IS NOT OPTIMAL.")')
    ENDIF
    PAUSE
  ENDIF
ELSE
C    ALGORITHM TO IDENTIFY OPTIMAL SOLUTION
    IF(OPTS .EQ. 1) THEN
      IF(INFI .EQ. 0) THEN
        WRITE(1,220) ICHAR(12):
        WRITE(1,'(1X,2X,"THE LAST BASIC SOLUTION WAS OPTIMAL.")')
      ELSEIF(INFI .NE. 0) THEN
        WRITE(1,'(1X,2X,"A FEASIBLE SOLUTION DOES NOT EXIST.")')
        WRITE(1,'(1X,2X,"AN ARTIFICIAL VARIABLE IS AT A POSITIVE LEVEL IN THE
SOL.
UTION.")')
      ELSEIF(IFLA6(I) .NE. 0) THEN
        WRITE(1,'(1X,2X,"THE SOLUTION IS INFEASIBLE SINCE THE"
"
"BASIC VARIABLE X(",I2,"") IS NEGATIVE.")')
      ENDIF
      PAUSE
    ENDIF
ELSEIF(OPTS .EQ. 0) THEN
  WRITE(1,'(1X,"IS THE OPTIMAL SOLUTION ALSO FEASIBLE?"')
  ELSEIF(OPTS .EQ. 1) THEN
    WRITE(1,'(1X,"IS THE SOLUTION FEASIBLE?"')
  ENDIF
ENDIF
READ(5,19) F
IF(ICHAR(F) .EQ. 84) THEN
C    USER ELECTS TO REVIEW TABLEAU
    CALL ICAD
    GO TO 530
ELSEIF(ICHAR(F) .NE. 78 AND ICHAR(F) .NE. 89) THEN
  WRITE(1,230)
  PAUSE
  GO TO 530
ENDIF
C USER PROVIDED FEEDBACK ON RESPONSE

IF(INF1 .EQ. 0 .AND. IFLAG(1) .EQ. 0) THEN
  IF(ICHAR(F) .EQ. 89) THEN
    WRITE(1,240) CB,OB
  ELSEIF(ICHAR(F) .EQ. 78) THEN
    WRITE(1,240) INC,OB
    WRITE(1,'(1X,'"THE CURRENT TABLEAU IS FEASIBLE SINCE"/2X,"'
      .ALL RHS VALUES ARE POSITIVE AND ALSO"/4X,"ANY ARTIFICIAL VARIABLE/
      .S ARE AT A"/14X,"ZERO VALUE."/1X")'
    ) ENDIF
  ELSE
    IF(ICHAR(F) .EQ. 78) THEN
      WRITE(1,240) INC,OB
      IF(INFL .NE. 0) THEN
        WRITE(1,540) CB(INFL)
        FORMAT('"THE SOLUTION IS INFEASIBLE SINCE THE"/"ARTIFICIAL/
          .VAR X*/12," IS AT A"/"POSITIVE LEVEL."')
        ELSE
          WRITE(1,560) CB(IFLAG(1))
        ENDIF
      ELSEIF(ICHAR(F) .EQ. 89) THEN
        WRITE(1,240) CB(INFL)
        WRITE(1,540) CB(INFL)
      ELSE
        WRITE(1,560) CB(IFLAG(1))
      ENDIF
    ENDIF
    ELSEIF(IFLAG .EQ. 0) THEN
      WRITE(1,570) CB(INFL)
      ENDIF
    ELSE
      WRITE(1,240) CB(INFL)
      IF(INFL .NE. 0) THEN
        WRITE(1,540) CB(INFL)
      ELSE
        WRITE(1,560) CB(IFLAG(1))
      ENDIF
      ENDIF
    ENDIF
  ENDIF
  ELSE
    IF(IFLAG .EQ. 0) THEN
      WRITE(1,570) CB(INFL)
    ENDIF
  ENDIF
  GOTO 840
ENDIF
ENDIF

C OPTIMAL AND INFEASIBLE SOLUTIONS NOT CHECKED FOR UNBOUNDEDNESS

GO TO 840
ENDIF
ENDIF
OB='DEGENERATE'
MN='NOT DEGENERATE'
C SOLUTION CHECKED FOR DEGENERACY

DO 520 I=1,K
  IF(B(I) .LT. -0.001 .AND. B(I) .LT. +0.001) THEN
    IFLAG(b)=1
  ENDIF
520 CONTINUE
IF(OIU .EQ. 1) THEN
C USER HAS ELECTED TO IDENTIFY DEGENERATE SOLUTIONS

WRITE(1,220)CHAR(12)
WRITE(1,510)
WRITE(1,.containsKey(""""WAS THE PREVIOUS SOLUTION DEGENERATE? "",4)"")
READ(5,.containsKey(""(A)"")IF
IF(ICHAR(F).EQ.64)THEN
C USER ELECTS TO REVIEW TABLEAU
CALL TCAL
GO TO 430
ELSEIF(ICHAR(F).NE.78.AND.ICHAR(F).NE.69)THEN
WRITE(1,230)
PAUSE
GO TO 430
ENDIF
C USER RESPONSE CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
IF(IFLAG6).EQ.0)THEN
IF(ICHAR(F).EQ.78)THEN
WRITE(1,240)CO,OBK
ELSEIF(ICHAR(F).EQ.99)THEN
WRITE(1,240)IN,OBK
WRITE(1,240)NOU,
ENDIF
PAUSE
ELSEIF(ICHAR(F).EQ.69)THEN
WRITE(1,240)OBK
ELSEIF(ICHAR(F).EQ.78)THEN
WRITE(1,240)NC,OBK
WRITE(1,240)IN,OBK
ENDIF
PAUSE
ENDIF
ENDIF
IF(OPTS.EQ.0)THEN
C NON-OPTIMAL SOLUTION-FIVST ROW DETERMINED
SFR=1.E8
DG 700 I=1.K
IF(A(I,PK).LE.0001)THEN
GO TO 700
ELSEIF(XB(I)/A(I,PK)).GE.SPR)THEN
GO TO 700
ELSE
SPR=(XB(I)/A(I,PK))
PR=1
ENDIF
CONTINUE
IF(SPR.GE.10.E6)THEN
C RATIO IS INFINITE, THEREFORE SOLUTION UNBOUNDED
IFLAG7=1
ENDIF
OBB="BOUNDED"
}

305
IF (IFLAG .EQ. 1) THEN
  C USER HAS ELECTED TO IDENTIFY UNBOUNDEDNESS
  WRITE (1,220) CHAR(12)
  WRITE (1,510)
  WRITE (1,'(A3,''WAS THE PREVIOUS SOLUTION UNBOUNDED''/)
  BASE .O UPON THE NEXT PIVOT COLUMN (ROW)'')/''BEING THE COLUMN (ROW) WITH
  THE LARGEST''/4X,'''NEGATIVE 2(J)-C(J) (B(J)) VALUE? '''
  READ (5,'(I1)''F
  IF (ICHAR(F) .EQ. 84) THEN
    C USER ELECTS TO REVIEW TABLEAU
    CALL TCAL
    GO TO 710
  ELSEIF (ICHAR(F).NE. 78 .AND. ICHAR(F).NE. 99) THEN
    WRITE (1,230)
    PAUSE
    GO TO 710
  ENDIF
  C USER INPUT CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
  IF (IFLAG .EQ. 1) THEN
    IF (ICHAR(F).EQ. 89) THEN
      WRITE (1,240) CO, NNU
      WRITE (1,720) PK
      FORMAT ('3X,'''COLUMN '',1,2,''' COEFFICIENTS ARE ZERO OR''/1,'''WE
      SITIVE, OR THE RATIO OF (B(I))/N(J)/10X, IS EXTREMELY LARGE.'')
    ELSEIF (ICHAR(F).EQ. 78) THEN
      WRITE (1,240) INC, NNU
      WRITE (1,720) PK
    ENDIF
    PAUSE
    GO TO 840
  ELSE
    IF (ICHAR(F).EQ. 78) THEN
      WRITE (1,240) CO, O99
    ELSEIF (ICHAR(F).EQ. 89) THEN
      WRITE (1,240) INC, O99
    ENDIF
    WRITE (1,'(2I1,’”THE CURRENT TABLEAU IS BOUNDED SINCE''/1,'''ALL THE A(I,J)
    VALUES IN COLUMN '',1,2,''' ARE''/9),”NJT NEGATIVE
    OR ZERO,”''/1)'') PK
    ENDIF
  ENDIF
ELSEIF (FLAG(7) .EQ. 1) THEN
  WRITE (1,220) CHAR(12)
  WRITE (1,510)
  WRITE (1,'(A3,''THE LAST BASIC SOLUTION WAS UNBOUNDED.''
 .ENDIF
END IF
C CHECK FOR MULTIPLE OPTIMAL SOLUTIONS
DO 760 J=1,VT
IFLAG(S)=0
DO 750 I=1,K
IF(CB(I) .EQ. J)THEN
IFLAG(S)=1
ENDIF
750 CONTINUE
IF(IFLAG(S) .EQ. 0)THEN
IF(C(J) .LT. .0001 .AND. C(J) .GT. -.0001)THEN
IFLAG(4)=1
ENDIF
ELSEIF
ENDIF
760 CONTINUE
IF(OPTS .EQ. 1)THEN
IF(OIU .EQ. 1)THEN
C USER HAS ELECTED TO IDENTIFY MULTIPLE OPTIMAL SOLUTIONS
770 WRITE(1,220)CHAR(12)
WRITE(1,510)
WRITE(1,510)
READ(S,'(A1)')F
IF(ICHAR(F) .EQ. 84)THEN
C USER ELECTS TO REVIEW TABLEAU
CALL TCL
GO TO 770
ELSEIF(ICHAR(F) .NE. 78 .AND. ICHAR(F) .NE. 89)THEN
WRITE(1,230)
PAUSE
GO TO 770
ENDIF
C USER RESPONSE CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
IF(IFLAG(4) .EQ. 1)THEN
IF(ICHAR(F) .EQ. 89)THEN
WRITE(1,780)CO
780 FORMAT(/A6,'YOUR RESPONSE WAS ',A9,' THERE ARE MULTIPLE SOLUTIONS.')
WRITE(1,800)
800 FORMAT(/A9,'A NON-BASIC VARIABLE HAS A ZERO COEFFICIENT IN THE OBJECTIVE FUNCTION OF ',A9,' THERE ARE NO MULTIPLE SOLUTIONS.')
ELSEIF(ICHAR(F) .EQ. 78)THEN
WRITE(1,780)INC
WRITE(1,800)
ELSEIF(ICHAR(F) .EQ. 78)THEN
WRITE(1,820)CO
820 FORMAT(/A6,'YOUR RESPONSE WAS ',A9/1X,' THERE ARE NO MULTIPLE SOLUTIONS.')
ELSEIF(ICHAR(F) .EQ. 89)THEN
WRITE(1,820)INC
WRITE(1,'()'1X,' THIS IS SINCE ALL NON-BASIC VARIABLES')/1
HAVE A VALUE OF OTHER THAN ZERO IN THE OBJECTIVE FUNCTION. IF A ZERO VALUE WAS PRESENT FOR A NON-BASIC VARIABLE, WRITE(1,'(1X,"INCREASING THE VALUE OF THIS VARIABLE"/5X,"WOULD NOT CHANGE THE Z VALUE."'))

ENDIF

PAUSE

ENDIF

ENDIF

ENDIF

IF(OPTS .EQ. 0) THEN
    990
    GO TO 880
    ENDIF

880 IF(LITAB .EQ. 2) THEN
    C FINAL TABLEAU IS NOT TO BE DISPLAYED
    RETURN

    ELSE
    WRITE(1,220)CHAR(12)
    IF(OUTP .EQ. 1) THEN
        OPEN(2,FILE='CONSOLE:')
    ELSE
        OPEN(2,FILE='PRINTER:')
    ENDIF

    ELSE IF(LITAB .EQ. 2) THEN
    CALL TDISPL
    CALL DASDIS
    RETURN
    END

340 IF(LITAB .EQ. 2) THEN
    C Tableau header printed
    WRITE(2,'(10X,"BASIC SOLUTION","",12)') BASIC
    WRITE(2,'(10X,"FINAL TABLEAU - ",41)')
    IF(INFI .NE. 0 .OR. IFLAG(1) .NE. 0) THEN
        WRITE(2,'("INFEASIBLE")')
    ELSEIF(IFLAG(2) .EQ. 1) THEN
        WRITE(2,'("UNBOUNDED")')
    ELSE
        WRITE(2,'("OPTIMAL")')
    ENDIF

    IFLAG(9)=2
    CALL TDISPL

    ENDIF

880 CALL BASDIS
    RETURN

END
SUBROUTINE TCAL
INTEGER ARIV,CB
COMMON/ELA/A(20,60),ARTV(20),C(60),I,INED(20),IFLAG(10),CB(20)
,NEC,MBC,NCL,IA,INDEXE,INDEXE1,INDEXL,IX(20)
IFLAG(9)=2
IFLAG(3)=1
OPEN(FILE='CONSOLE:')
WRITE(2,'(A)')CHAR(12)
CALL TDISPL
RETURN
END
SUBROUTINE PIVOT

C USE: DEPENDENT UPON USER OPTION SELECTION, EITHER SOLICITS INPUT
C OF USER SELECTED PIVOT COLUMN AND ROW OR PASSES CONTROL TO
C SUBROUTINE WORK USING ALGORITHM SELECTED PIVOT ELEMENT.
C PROVIDES FEEDBACK, IF OPTION SELECTED, AND ALERTS USER OF
C PIVOT ELEMENT SELECTION WITH VALUE OF APPROXIMATELY ZERO.
C WITH THE DUAL PIVOT OPTION, USER MUST SELECT PIVOT ELEMENTS.
C ROUTINE WILL NOT ALLOW FURTHER PIVOTS UPON REACHING BOTH
C PRIMAL AND DUAL OPTIMALITY CONDITIONS.
C CALLED BY: PROGRAM EDUC
C CALLS : SUBROUTINE CHECK2(P,N,M,IVAL,INEW)
C SUBROUTINE OVER(RES)
C SUBROUTINE WORK
C VARIABLES:
C USED: A(8,8),C(8),DUAL,FNT,INEW,IVAL,K,PES,RES,VT,IB(4)
C MODIFIED: IFLAG(4),IFLAG(6),IFLAG(10),L,M,P(8),PK,PKS,PR,RES,
C           RATIO,SPR,TIE

C USES UCHECK2 IN UNIT27.COE OVERLAY
C USES UHEADER IN UNIT25.COE OVERLAY

SUBROUTINE PIVOT

CHARACTER VNI$,CN$,PN$,MN$,FN$,PI$,PINEQ$,P1$,QN$10
INTEGER ARTV,BASIC,PK,PKS,PR,PKS,OPTS,V,VT,CB,PES,DIU,DUAL,OUTP,
       TIE,FNT,RES,ASK
COMMON/E1/A(20,60),ARTV(20),C(60),I,INEQ(20),IFLAG(10),CB(20)
       .NEC,HSC,NLC,IA,INOE,INDCS,INDEXL.IS(20)
COMMON/E2/BASIC,PK,PKS,PR,PKS,OPTS,PES,V,VT,FMIN
COMMON/E3/MODE,PES,DIU,DUAL,OUTP,ITAB,ISTAB,IFTAB,BM,TIE,FNT
COMMON/E4/VN(20),CN(20),PN,MN,FN,PINEQ(20),P(10),OBJN
L=I
N=1
C VARIABLES IPP,IPK CONTAIN ALGORITHM SELECTED PIVOT ELEMENT
IFLAG(4)=PK
IFLAG(6)=PR
100 WRITE(I,'7y'OdAR
WRITE(WIO,I,'7y'DUAL
WRITE(I,'7y')T POPRY EROR DAL /
WRITE(WIO,I,'7y')T POPRY FFFF
WRITE(WIO,I,'7y')ULIO
310

ENDIF
IF(PES .EQ. 3):THEN
C USER HAS ELECTED ALGORITHM TO SELECT PIVOT ELEMENT
CALL WORK
RETURN
ENDIF
IF(L .EQ. 0 .AND. M .EQ. 0):THEN
C NO NEGATIVE RHS OR Z(J)-C(J) ELEMENTS EXIST
WRITE(I,'7y'OdAR
WRITE(I,'7y')YOU ARE UNABLE TO PROPERLY PERFORM DUAL" /" OR
. PRIMAL PIVOTS ON THE CURRENT TABLEAU."
WRITE(I,'7y'OdAR
WRITE(I,'7y')TO OVERRIDE, ENTER DIU

310
AL OR PRIMAL TO CONTINUE. ANY OTHER ENTRY WILL TERMINATE'/.
"PROBLEM"/(18K,"OPTION? ",6)
READ(5,'(A1)')(P(1)
WRITE(1,110)CHAR(12)
IF(ICHARP(P(1)) .EQ. 80)THEN
   GO TO 155
ELSEIF(ICHARP(P(1)) .EQ. 60)THEN
   GO TO 1025
ELSE
   C FLAG DENOTES FURTHER PIVOTS NOT POSSIBLE
   IFLAG(10)=5
   RETURN
ENDIF
ENDIF
IF(DUAL .EQ. 2)THEN
   C USER HAS ELECTED TO ALLOW DUAL PIVOTS
   WRITE(1,'(A1)')P(1)
   CALL CHECK2(P,1,3,INVAL,INEW)
   IF(INVAL .EQ. 1)THEN
      WRITE(1,130)
      FORMAT(5X,'INVALID ENTRY, PLEASE REENTER')
      GO TO 100
   ENDIF
   IF(INEW .EQ. 2)THEN
      C USER HAS ELECTED TO ATTEMPT DUAL PIVOT
      GO TO 1000
   ELSEIF(INEW .EQ. 3)THEN
      C1 FLA(M:95
      RETURN
   ENDIF
ENDIF
ENDIF
140 CONTINUE
IF(L .EQ. 0)THEN
   WRITE(1,110)CHAR(12)
   WRITE(1,'(10F12.1)')"TO PERFORM PRIMAL PIVOTS, AT LEAST ONE"/3X
   "C(J) MUST BE NEGATIVE. THIS IS NOT"/2X,"PRESENT SO A PRIMAL P
   IVOT CANNOT BE"/15X,"DONE."/15X
   PAUSE
   GO TO 100
ENDIF
L=1
DO 150 I=1,K
   IF(10(I) .LT. 0.0)THEN
      10
ENDIF
150 CONTINUE
TO PERFORM PRIMAL PIVOTS, ALL RHS VALUES *MUST BE POSITIVE. THIS IS NOT PRESENT SO "A PRIMAL PIVOT CAN NOT BE DONE."

PAUSE

GO TO 100

ENDIF

150 CONTINUE

WRITE(1,110)CHOR(12)

WRITE(1,1160)

160 FORMAT(9('","WHICH COLUMN CONTAINS THE CANDIDATE"/IIX,"ENTERING

. VARIABLE?"/)

170 WRITE(1,180)

180 FORMAT(IIX,"COLUMN = ",I9)

READ(5,

1(2A1))P(1),P(2)

CALL CHECK2(P,2,VT,IVAL,INEM)

IF(INEM .EQ. 1)THEN

WRITE(1,130)

GO TO 170

ELSE

PKS=INEM

ENDIF

IF(PES .EQ. 2) THEN

C USER HAS ELECTED TO SELECT PIVOT WITHOUT CHECK

FK=PKS

GO TO 300

ENDIF

WRITE(1,1190)

C USER SELECTION CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED

IF(PK .EQ. PKS)THEN

WRITE(1,190)

190 FORMAT(IIX,"YOUR PIVOT COLUMN SELECTION MATCHES THE"/IIX,"A

. ALGORITHM SELECTION"/)

PAUSE

ELSE

IF(TIE .EQ. 1) THEN

C TIE FOR PIVOT COLUMN EXISTS

IF(C(PK)+.0001 .GE. C(PKS) .AND. C(PK)-.0001 .LE. C(PKS))THEN

C CHECKS IF USER SELECTION ONE OF TIES

WRITE(1,190)

FX=PKS

PAUSE

ENDIF

ELSE

WRITE(1,200)

200 FORMAT(50,"YOUR SELECTION OF PIVOT COLUMN DOES NOT"/IIX,"MATCH

. THAT OF THE ALGORITHM"/IIX,"WHICH SELECTION DO YOU WISH TO USE?

"

WRITE(1,220)PKS,FX


. ALGORITHM SELECTION COLUMN = ",I2)
WRITE(1,240)
FORMAT(//13X,'WHICH OPTION? ',A)
READ(5,')A')P
CALL CHE2(P,1,2,INVAL,INNEW)
IF(INVAL .EQ. 1) THEN
  WRITE(1,120)
  GO TO 230
ELSE IF(INNEW .EQ. 1) THEN
  PK=PKS
ENDIF
ENDIF

310 WRITE(1,110)CHAR(12)
C RATIOS FOR PIVOT COLUMN CALCULATED AND DISPLAYED
WRITE(1,320)PK

320 FORMAT(10X,'RATIOS FOR COLUMN ',I2/)
SPR=10.E6
TIE=0
DO 400 J=1,K
  WRITE(1,'("ROW ",I2," = ",F9.9)')
  IF(A(I,PK) .LE. .0001) THEN
    IF(XB(J) .LE. .0001) THEN
      IF(X(I,J) .LE. .0001 .AND. XB(J) .GE. -1.E-6) THEN
        WRITE(1,'("INFINITE")')
      ELSE
        WRITE(1,'("NEGATIVE RATIO")')
      ENDIF
    ELSE
      IF(RATIO+.0001 .GE. 1.E6) THEN
        WRITE(1,'("INFINITE")')
      ELSE
        RATIO=(X(J)/A(I,PK))
        IF(FNT .EQ. 0) THEN
          WRITE(1,'("F12.5")RATIO
        ELSE
          WRITE(1,'("F12.5")RATIO
        ENDIF
      IF:RATIO .LT. SPR THEN
        TIE=0
        SPR=RATIO
        PR=1
      ELSEIF(RATIO+.0001 .GE. SPR .AND. RATIO-.0001 .LE. SPR) THEN
        TIE=1
      ENDIF
      ENDIF
    ELSE
    CONTINUE
C USER SELECTS PIVOT ROW
WRITE(1,410)

410 FORMAT(//4X,'WHICH ROW CONTAINS THE CANDIDATE?/11X,'LEAVING VAR
"TABLE")

313
420 WRITE(1,430)
430 FORMAT(16x,'ROW = ',0)
READS(5,'(2AI)'P(1),P(2)
CALL CHECK2(P,Z,K,INVAL,INEN)
IF(INVAL .EQ. 1) THEN
  WRITE(1,110)
  GO TO 420
ELSE
  PRS=INEN
ENDIF
IF(PEQ.2) THEN
  PRS=PR
  GO TO 700
ENDIF
WRITE(1,110)CHAR(12)
C USER SELECTION CHECKED AND FEEDBACK PROVIDED
IF(PR.EQ.PRS) THEN
  WRITE(1,450)
  FORMAT('YOUR SELECTION ROW MATCHES THE ALGO.
          SELECTION.'))/)
  PAUSE
ELSEIF(TIE.EQ.1) THEN
  IF((IB(PR)/AIPR,PK)+.0001) .GE. (IB(PRS)/A(PRS,PK)) .AND.
  .((IB(PR)/AIPR,PK)-.0001) .LE. (IB(PRS)/A(PRS,PK)) THEN
    WRITE(1,450)
    PR=PR
    PAUSE
    ENDIF
ELSE
  WRITE(1,470)
  FORMAT('YOUR SELECTION OF PIVOT ROW DOES NOT MATCH
          THAT OF THE ALGORITHM.'7/)
  ' WHICH SELECTION DO YOU WISH TO USE?')
  WRITE(1,490)prs,pr
490 FORMAT('1,12//15X,'ROW = ',12//15X,'2. ALGO
          SELECTION ROW = ',12)
500 WRITE(1,240)
READS(5,'(41)'P(1)
CALL CHECK2(P,1,2,INVAL,INEN)
IF(INVAL.EQ.1) THEN
  WRITE(1,130)
  GO TO 500
ELSEIF(INEN.EQ.1) THEN
  PR=PRS
ENDIF
ENDIF
C PIVOT ELEMENT CHECKED TO INSURE NOT ZERO
IF(AIPR,PK) .LT. .0001 .AND. AIPR,PK OR .6. -.0001) THEN
C USER GIVEN OPTION TO CONTINUE WITH ZERO PIVOT ELEMENT
CALL OVER(RES)
IF(RES.EQ.7) THEN
  GO TO 106
ENDIF
ENDIF
ENDIF
ENDIF
CALL WORK
RETURN
C DUAL PIVOT ELEMENT DETERMINED
1000 WRITE(1,1100)CHAR(12)
   30 1010 J=1,VT
   IF(C(J) .LT. 0.0)THEN
      WRITE(1,1100)CHAR(12)
      IF(C(J) .LT. 0.0)THEN
         WRITE(1,1100)CHAR(12)
         TO PERFORM DUAL PIVOTS, ALL C(J) MUST BE
         POSITIVE. THIS IS NOT PRESENT AT THIS TIME SO A DUAL PIVOT
         WILL NOT BE DONE.
      ENDIF
      M=0
      PAUSE
      GO TO 100
   ENDIF
1010 CONTINUE
   M=0
   DO 1020 I=1,K
      IF(I.B .LT. 0.0)THEN
         N=I
      ENDIF
   1020 CONTINUE
   IF(P*EI. .EQ. 0.0)THEN
      WRITE(I,1100)CHAR(12)
      TO PERFORM DUAL PIVOTS, AT LEAST ONE RHS
      MUST BE NEGATIVE. THIS IS NOT PRESENT AT THIS TIME SO A DUAL P
      IVOV WILL NOT BE DONE.
      PAUSE
      GO TO 100
   ENDIF
C USER SELECTS PIVOT ROW
   WRITE(1,1410)
1030 WRITE(1,1430)
   READ5((241),P(1),P(2)
   CALL CHECK2(P,2,K,INVAL,INEW)
   IF(INVAL .EQ. 0.0)THEN
      WRITE(1,1310)
      GO TO 1030
   ELSE
      PRS=INEW
      ENDIF
   IF(P*ES .EQ. 2.0)THEN
      PRS=PRS
      GO TO 1300
   ENDIF
C ALGORITHM SELECTION PIVOT ROW
   BNES=0.0
   DO 1050 I=1,K
      IF(I.B .LT. 0.0)THEN
         GO TO 1050
      ELSEIF(I.B .EQ. BNES)THEN
         TIE=1
      ELSE

TIE=0
OMEG=10(I)
PR=1
ENDIF

1050 CONTINUE
C USER SELECTION CHECKED AND FEEDBACK PROVIDED
WRITE(1,110)CHAR(12)
IF(PR .EQ. PRS)THEN
  WRITE(1,450)
  PAUSE
ELSE
  WRITE(1,470)
END IF
WRITE(1,1490)PPS,PR

1070 WRITE(1,240)
READ(3,'(AI1)')P(I)
CALL CHECK2(P,1,2,INVAL.INEW)
IF(INVAL .EQ, 1)THEN
  WRITE(1,1350)
  GO TO 1070
ELSEIF(INEW .EQ, 1)THEN
  PR=PRS
ENDIF
ENDIF

1300 WRITE(1,110)CHAR(12)
C RATIOS FOR ROW CALCULATED
WRITE(1,1320)PR

1320 FORMAT(11x,'RATIOS FOR ROW ','12x)
PR=10.ED
GO 1400 J=1,VT
DO 1350 I=1,K
  IF(CB(I) .EQ. J)THEN
    GO TO 1400
ENDIF

1350 CONTINUE
WRITE(1,'(7x,"COLUMN ",12x="",$",")')
IF(A(J,J) .GE. -.0001)THEN
  IF(A(J,J) .LE. .0001)THEN
    IF(N(J) .LE. .0001 .AND. C(J) .GE. -.0001)THEN
      WRITE(1,'("0.0")')
    ELSE
      WRITE(1,'("INFINITE")')
    ENDIF
  ELSE
    WRITE(1,'("NEGATIVE INFINITE")')
  ENDIF
ELSE
  RATIO=(C(J)/A(J,J))
  IF(PRJ .EQ. 0)THEN
    WRITE(1,'("PE12.5")')RATIO
  ELSE
    WRITE(1,'("F12.5")')RATIO
  ENDIF
ENDIF
IF(RATIO .GE. SPR) THEN
  PK=J
  SPR=RATIO
ENDIF
ENDIF

1400 CONTINUE
PAUSE
WRITE(1,110)CHAR(12)
C USER SELECTS PIVOT COLUMN
WRITE(1,160)
1420 WRITE(1,180)
READ(5,''201'')P(1),P(2)
CALL CHECK2(P,2,VT,INVAL,INEW)
IF(INVAL .EQ. 1) THEN
  WRITE(1,130)
  GO TO 1420
ELSE
  PKS=INEW
ENDIF
IF(PES .EQ. 2) THEN
  PK=PKS
  GO TO 1700
ENDIF
WRITE(1,110)CHAR(12)
C USER SELECTION CHECKED AND FEEDBACK PROVIDED
IF(PK .EQ. PKS) THEN
  WRITE(1,190)
  PAUSE
ELSE
  WRITE(1,200)
  WRITE(1,220)PKS,PK
1440 WRITE(1,240)
READ(5, ''(A17)'')P(1)
CALL CHECK2(P,1,2,INVAL,INEW)
IF(INVAL .EQ. 1) THEN
  WRITE(1,130)
  GO TO 1440
ELSEIF(INEW .EQ. 1) THEN
  PK=PKS
ENDIF
ENDIF
IF(A(PR,PK) .LT. .0001 .AND. A(PR,PK) .GT. -.0001) THEN
C USER GIVEN OPTION TO CONTINUE WITH PIVOT ELEMENT OF ZERO VALUE
CALL OVER(PES)
IF(PES .EQ. 0) THEN
  GO TO 100
ENDIF
ENDIF
1700 CALL WORK
RETURN
END
SUBROUTINE WORK

USE: PERFORMS SIMPLEX PIVOT USING DESIGNATED PIVOT ELEMENT. NO USER INTERFACE.

CALLED BY: SUBROUTINE PIVOT

CALLS: NONE

VARIABLES:

USED: K,PX,PR,VT

MODIFIED: A(I,I),C(I),CB(*),HOLD,PELE,XB(*,2)

SUBROUTINE WORK
INTEGER CRTV,BASIC,PX,PK,PR,PKS,PTS,VT,VT,PC,PE,CIU,DUAL,UTP
COMMON(E1)(I20,B0),ARTV(20),C160,:2,INED(20),IFLAG(10),CB(20)
..NEC,MNC,MNC,TA,INDEXE,INDEXG,INDEXL,IX(20)
COMMON/ED2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PX,PK,PR,PK,PX,PR,PR,VT,VP
PELE=APR,PK)
DO 200 J=1,VT
A(PX,J)=A(PX,J)/PELE
200 CONTINUE
XB(PX)*=XB(PX)/PELE
CB(PX)=PK
DO 300 I=1,K
IF(I.EQ.PR)THEN
  60 TO 300
ENDIF
HOLD=A(I,PX)
DO 250 J=1,VT
A(I,J)=A(I,J)-HOLD*APR,PX)
250 CONTINUE
XB(I)=XB(I)-HOLD*APR)
300 CONTINUE
HOLD=C(PK)
DO 350 J=1,VT
C(J)=C(J)-HOLD*APR,PX)
350 CONTINUE
Z=Z-HOLD*APR)
RETURN
END
SUBROUTINE OVER(RE3)

C USE: DISPLAYS STATEMENT THAT PIVOT ELEMENT IS APPROXIMATELY ZERO
AND PERFORMANCE OF PIVOT MAY RESULT IN SYSTEM OVERFLOW ERROR.
SOLICITS RESPONSE AS WHETHER TO CONTINUE WITH PIVOT AND SETS
FLAGS TO REFLECT THIS RESPONSE.

CALLED BY: SUBROUTINE PIVOT

CALLS: NONE

VARIABLES: NONE

USED: NONE

MODIFIED: PI().RES

SUBROUTINE OVER(RE3)

CHARACTER PI1

INTEGER RES

RES=0

WRITE(1,110)CHAR(12)

110 FORMAT(A)

WRITE(1,150)

150 FORMAT(A/3X,'THE PIVOT ELEMENT SELECTED IS NOT/2X,'SIGNIFICANT
LY DIFFERENT FROM ZERO./3X,'THIS MAY CAUSE AN OVERFLOW ERROR/!
.7X,'IF THE PIVOT IS PERFORMED.'))

160 WRITE(1,170)

170 FORMAT(/7X,'DO YOU WISH TO CONTINUE? ',A)

READ(5,'(A)')P

IF(ICHAR(P) .EQ. 99) THEN
    RES=1
    RETURN
ELSEIF(ICHAR(P) .NE. 78) THEN
    WRITE(1,'(/5X,'INVALID ENTRY, PLEASE REENTER')')
    GO TO 160
ENDIF
END
C MODULE 2 UNIT25
C UNIT USES: NONE
C
C SUBROUTINE HEADER
C USE: DISPLAYS TITLE PAGE OF MODULE 2, EDUCATIONAL MODULE.
C CALLED BY: PROGRAM EDUC
C CALLS: NONE
C VARIABLES: NONE
C
C SUBROUTINE HEADER
WRITE(1,110)CHAR(12):
110 FORMAT(A)
WRITE(1,'(A)',9X,'("\*\"\",",",9X,20X,",","",",",",",",",",",",",",",",",",",",",","",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",","
C MODULE 2 UNITZ5
C
C SUBROUTINE ASK(ASK)
C USE: ROUTINE PROMPTS USER TO RESPOND WHETHER OR NOT FINAL
C SOLUTION IS TO BE SAVED TO DISK FOR SENSITIVITY ANALYSIS. IF
C USER REQUESTS FILE TO BE SAVED, A VOLUME:FILENAME IS
C REQUESTED. THIS FILE IS OPENED AND FORMATTED SO AS TO BE
C COMPATIBLE WITH MODULE 4 INPUT REQUIREMENTS. ALSO, THIS
C FILENAME IS WRITTEN TO DISK IN THE DATA FILE LP2:LPDATAW
C FOR TRANSITION TO MODULE 4. USER IS NEXT PROMPTED TO
C DETERMINE WHETHER RAW NOT ANOTHER MODEL IS TO BE STUDIED WITH
C THIS MODULE. IF NO OTHER MODEL IS TO BE STUDIED, MODULE 2 IS
C TERMINATED WITH INSTRUCTIONS ON ENTERING MODULE 1.
C CALLED BY: PROGRAM EDUC
C CALLS : NONE
C VARIABLES:
C USED: A, I, B(I), C(I), CB(I), CM(I), CM(I), IFLAG(S), INDEXS,
C INDEXE, INDEXL, XB(I), CH, CH, CH, CH, CH, CH, CH, CH, CH, CH, CH, CH
C MODIFIED: AO, ASK, FN, FNO, IFLAG(I), IFM(I), P(I)

SUBROUTINE ASK(ASK)
CHARACTER VNS16,CNS6,PNS20,MNS3V,FNS10.PNS10.ATV.BAS
INTEGRATION ARTV.BASIC, PK, PKS, PR, PRO, OIFS, V, VT, CB, ASK
COMMON/1A(20,60),ARTV(20),I(60),2,INEQ(20),IFM(10),CM(20)
, IEQ, NIE, IN, INDEXE, INDEXL, XB(I), CH
COMMON/2B/IEQ, KS, IFSA,IFS, IFSA,IFS, IFSA,IFS, IFSA,IFS, IFSA,IFS, IFSA,IFS, IFSA,IFS,
COMMON/E/CH(20),CM(20),PN, M, FN, PINEQ(20), P(10), DBJA
DIMENSION AU(20,20), B(I), X(I), 20
FNO=FN
100 WRITE(1,110)CHAR(12)
110 FORMAT(A)
45K=0
IFLAG(I)=0
WRITE(1,'(B/1)'"TO PERFORM SENSITIVITY ANALYSIS ON THIS"/"MODEL
, THE INFORMATION OF THE CURRENT"/"TABLEAU MUST BE SAVED TO DISK
"/"/1)
130 WRITE(1,('"DO YOU WISH TO SAVE THIS FILE TO DISK? ","'))
READ(5.10)P
IF(ICHAR(P(1)) .EQ. 89 THEN
WRITE(1,'(B/1)'"SAVE LP MODEL TO DISK"/"ENTER THE DISK D
RIVE NUMBER AND FILE"/"NAME YOU WANT THE CURRENT TABLEAU OF"
/"/1,420," SAVED UNDER."/"/1)
WRITE(1,'(B/1)'"ENTER EXACTLY AS FOLLOWS"/"D NAME"/"FILE"/"DRIVE"/"FILE"/"NAME"
/"/1,12X," ES. B NAME"/"THE DRIVE:FILENAME MUST BE 10
 CHARACTERS"/"OR LESS"/"DO NOT USE THE SAME NAME USED
 WHEN THE"/"ORIGINAL MODEL WAS ENTERED."/"/1)
WRITE(1,'(B/1)'"ARE CORRECTIONS NEEDED? ","))
READ(5.10)/FN
150 WRITE(1,'(B/1)'"ARE CORRECTIONS NEEDED? ",="))
READ(5,'(A1)') P(I)
IF(CHAR(P(I)) .EQ. 89) THEN
   GO TO 100
ELSEIF(CHAR(P(I)) .NE. 78) THEN
   WRITE(1,'(1X,''INVALID ENTRY, PLEASE REENTER'')')
   GO TO 150
ENDIF
WRITE(1,110) CHAR(12)
WRITE(1,110) 'INSURE DISK IS AVAILABLE.',7(/)
PAUSE
C TRANSFER FILE OPENED AND FILE NAME WRITTEN
OPEN(3,FILE='LP2:LP001.TXT',STATUS='OLD',FORM='UNFORMATTED')
WRITE(3) FN
C TRANSFER FILE CLOSED
CLOSE(3,STATUS='KEEP')
WRITE(1,110) CHAR(12)
C USER PROMPTED TO INSERT DISK WHICH SOLVED MODEL IS TO BE SAVED
WRITE(1,'(1X,2X,''INSURE THE DISK TO CONTAIN THE FILE'','/15X,''
.A10/13X,''IS AVAILABLE.'',7(/))')
PAUSE
WRITE(1,110) CHAR(12)
C CURRENT STATUS OF FILE INPUT BY USER
WRITE(1,'(1X,''HAS THIS DISK:FILENAME COMBINATION BEEN'','/12X,
.''USED PREVIOUSLY?''/''ARE YOU UPDATING A CURRENTLY EXISTING'''
.17X,''FILE?'')')
200 WRITE(1,'(1X,2X,''(Y/N)'' ,#)')
READ(5,'(A1)') P(I)
C FILE OF STATUS DESIGNATED BY USER OPENED
IF(CHAR(P(I)) .EQ. 89) THEN
   OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
ELSEIF(CHAR(P(I)) .EQ. 78) THEN
   OPEN(3,FILE=FN,STATUS='NEW',FORM='UNFORMATTED')
ELSE
   WRITE(1,210)
ENDIF
WRITE(1,9(/),5X,'(FILE?)')
210 FORMAT(('5X,''INVALID ENTRY, PLEASE REENTER'')')
GO TO 200
C SOLVED MODEL WRITTEN TO DISK
WRITE(3)FN,XMIN,X,Y,IFLAG(5)
WRITE(1,110) CHAR(12)
WRITE(1,'(9(/),5X,''INSURE THE DISK CONTAINING THE'','/15X,A10,
.''/15X,''MODEL IS AVAILABLE.'',7(/))')
PAUSE
C ORIGINAL MODEL FILE OPENED TO READ ORIgINAL PARAMETERS
OPEN(4,FILE=FNOSTATUS='OLD',FORM='UNFORMATTED')
READ(4) FM,XMIN,WY,Y,W,NEC,WEC,WLC
DO 220 I=1,10
   READ(4) IFLAG(I)
220 CONTINUE
DO 240 I=1,K
   READ(4) INEG(I),PINEG(I),B(I)
   322
DO 230 J = 1, V
   READ (4) AO(I, J)

230 CONTINUE
DO 240 J = 1, V
   READ (4) CO(J)

240 CONTINUE
CLOSE (4, STATUS='KEEP')
WRITE (1, 110) CHAR (12)
WRITE (1, ' ") INSURE THE DISK TO CONTAIN THE FILE'//15X, A10
  "/13I, "IS AVAILABLE.'",7(/)') FN
PAUSE
DO 270 I = 1, K
   WRITE (3) NEQ(I), B(I)
DO 260 J = 1, V
   WRITE (3) AO(I, J)

260 CONTINUE
C SOLVED MODEL WITH ORIGINAL PARAMETERS WRITTEN TO FILE
DO 275 J = 1, V
   WRITE (3) CO(J)

275 CONTINUE
IF (FLAG(10) .EQ. 0) THEN
   WRITE (3) FLAG(10), VT
   DO 290 I = 1, K
      WRITE (3) XB(I), CB(I)
   DO 280 J = 1, V
      WRITE (3) VT
   WRITE (3) A(I, J)

280 CONTINUE
DO 300 J = 1, V
   WRITE (3) C(I, J)

300 CONTINUE
WRITE (3) I
IF (FLAG(5) .EQ. 1) THEN
   DO 310 I = 1, K
      WRITE (3) CN(I)

310 CONTINUE
DO 320 J = 1, V
   WRITE (3) VM(J)

320 CONTINUE
WRITE (3) OBJN
ENDIF
CLOSE (3, STATUS='KEEP')
WRITE (1, 110) CHAR (12)
WRITE (1, ' ") INSURE DISK LPI IS AVAILABLE.'",7(/)

PAUSE
ELSEIF (CHAR (P(I)) .NE. 78) THEN
   WRITE (1, 210)
GO TO 130
ENDIF
WRITE(1, '(117/1,1x,"WOULD YOU LIKE TO STUDY ANOTHER MODEL?"/4x," WHICH HAS BEEN SAVED TO DISK? ",9(1)) READ(5, '(A1)')P(1) IF(ICHAR(P(1)) .EQ. 89) THEN
WRITE(1, '(110)CHAR(12) WRITE(1, '(117/1,1x,"ENTER DISK DRIVE NUMBER AND FILENAME"/4x," WHICH THE MODEL IS SAVED UNDER."/)') WRITE(1, '(1/6X,"MODEL TO STUDY = ",9(1)) READ(5, '(A10)')FMN
WRITE(1, '(117/1,1x,"ARE CORRECTIONS NEEDED? ",9(1)) READ(5, '(A1)')P(1) IF(ICHAR(P(1)) .EQ. 89) THEN
GO TO 400 ELSEIF(ICHAR(P(1)) .NE. 78) THEN
WRITE(1, '(1/5X,"INVALID ENTRY, PLEASE REENTER")') GO TO 450 ENDIF
WRITE(1, '(110)CHAR(12) WRITE(1, '(117/1,1x,"INSURE DISK LP1 IS AVAILABLE."",7(1)/) PAUSE C TRANSFER FILE OFENED AND NEW MODEL FILE NAME WRITTEN OPEN(3,FILE="LP1:LPDATA",STATUS="OLD",FORM="UNFORMATTED") WRITE(3)FMN CLOSE(3,STATUS="KEEP") ASK=1 RETURN ELSEIF(ICHAR(P(1)) .NE. 78) THEN
WRITE(1, '(1/5X,"INVALID ENTRY, PLEASE REENTER")') GO TO 390 ENDIF
WRITE(1, '(110)CHAR(12) WRITE(1, '(117/1,1x,"INSURE DISK LP1 IS AVAILABLE."",7(1)/) PAUSE WRITE(1, '(110)CHAR(12) WRITE(1, ' (8(1),1X,"TO ENTER THE LP DATABASE MODULE:"/4X,"TYPE "/19X,"X"/11X,"LP1:SYSTEM.STARTUP. ",5(1)/) STOP RETURN END
SUBROUTINE QUESTN

USE: DISPLAYS INSTRUCTIONS TO USER NOTING THAT TABLEAU SHOULD BE STUDIED TO ALLOW USER TO ANSWER IDENTIFIED QUESTIONS WHICH FOLLOW. ONLY DISPLAYED IF USER IS SELECTING PIVOT ELEMENT AND OUTPUT IS TO SCREEN.

CALLED BY: PROGRAM EDUC

CALLS: NONE

VARIABLES: NONE

SUBROUTINE QUESTN
WRITE(1,10)CHAR(12)
10 FORMAT(A)
WRITE(1,
110 FORMAT(A)
110   WRITE(1,10),(1,'FROM THIS POINT ON, YOU WILL BE ASKED''/9I,''' QUESTIONS CONCERNING:''/''1. PIVOT COLUMN SELECTIONS''/''2. PIVOT ROW SELECTIONS''/''3. DUAL PIVOTS(IF OPTION SELECTED)''/''4. OPTIMALITY, FEASIBILITY, BOUNDEDNESS(IF OPTION SELECTED)''/''BE SURE TO EXAMINE THE SCREEN OUTPUT'' OF THE TABLES CAREFULLY BEFORE'' CONTINUING SO YOU MAY ANSWER QUESTIONS'' AS NOTED ABOVE.''/9I')
PAUSE
RETURN
END
SUBROUTINE EIGH

USE: PERFORMS CALCULATIONS TO ACCUIPE A INITIAL BASIC SOLUTION
WHEN ARTIFICIAL VARIABLES HAVE BEEN ADDED TO MODEL. NO USER INTERFACE.

CALLS: NONE

VARIABLES:
USD: A(*) , ARV(*) , BM , KFA , NGC , V , VT , XB(*)

MODIFIED: C(*) , KFSU , IA , M , SUM , Z

SUBROUTINE BEGIN

INTEGER ARV , I , PFS , KFSU , PPS , KB , PR , PB , CI , DUAL , DOUT ,
TIE , FAT
COMMON/E1/ ( 20 , 80 ) , ARV ( 20 ) , CI ( 80 ) , I , INEQ ( 20 ) , IFLAS ( 10 ) , CB ( 20 )
, NGC , KFA , M , IA , INDECE , INDEX , INDEIL , IB ( 20 )
COMMON/E2/BASIC , K , KFA , KFS , KFSU , KPS , PFS , PR , PPS , V , VT , MIMN
COMMON/E3/ NOD , PES , OIU , DUAL , DOUT , ITAB , IFTAB , BM , TIE , FAT
IA = IA - 1
KFSU = V + NGC
DO 300 J = 1 , KFSU
SUM = 0.0
DO 200 I = 1 , IA
SUM = SUM + ARV ( I , J )
200 CONTINUE
C ( J ) = C ( J ) - SUM / SUM
300 CONTINUE
DO 400 J = KFA , VT
C ( J ) = 0.0
400 CONTINUE
SUM = 0.0
DO 500 I = 1 , IA
SUM = SUM + B ( ARV ( I ) )
500 CONTINUE
Z = 1 - ( SM / SUM )
RETURN
END
SUBROUTINE INDEX

CHARACTER VNS6,CNS6,PN20,MNS3,FN*10,PINEQ4i,PI,OBJNIlO
INTEGER ARTV,BASIC,PK,PKS,PR,OPTS,V,VT,CK
COMMON/E1A(20,60),ARTV(20),C(60),I,INDEG(20),IFLAG(10),CB(20)
   ,NEC,NLC,IA,INDEXE,INDEXL,INDEXLS,J1(20)
   ,COMMON/E2,BASIC,K,KS,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PR,S,V,VT,VMN
COMMON/E4/VN(20),CM(20),PK,MM,FM,PINEQ(C20),F(10),OBJN
IA=1
KFS=K+1
VT=V+(2*NEC)+NLC+NEC
KFS=V+NEC+1
KFSU=KFS+NEC
INDEXE=V+1
INDEXI=V+NEC+1
INDEX=V+NEC+1
DO 230 I=1,K
   IF(INEQ(I) .EQ. 0) THEN
      CB(I)=INDEXI
      A(I,INDEXI)=1.
      INDEXL=INDEXL+1
   ELSE IF(INEQ(I) .EQ. 1) THEN
      CB(I)=INDEXE
      ARTV(IA)=1
      IA=IA+1
      A(I,INDEXE)=1.
      INDEXE=INDEXE+1
      A(I,INDEXE+1)=1.
      INDEXE=INDEXE+1
   ELSE
      ARTV(IA)=1
      IA=IA+1
      CB(I)=INDEXE
   ENDIF
230 CONTINUE

C " Determine the column position of added slack, surplus, and
C   artificial variables. Places the appropriate coefficient in
C   the A(I,J) array for each and identifies the initial basic
C   variable. Changes all inequalities to equalities in PINEQ(I).
C   And names the added variables for named models.
C
C CALLED BY: PROGRAM EDUC
C CALLS : NONE
C VARIABLES: IFLAG, INED, K, NEC, NLC, V
C MODIFIED: A(*,I), ARTV(*), CB(*), CH(*), IA, INDEXE, INDEXL, INDEXLS, KSA,
C   KFS, KFSU, PINEQ (*), VM(*), VT
C
C SUBROUTINE INDEX
C
A(I, INDEXE)=1.
INDEXE=INDEXE+1
ENDIF
PINEG(1)="="
200 CONTINUE
IF:IFLAG(5) .EQ. 1) THEN
  DO 210 J=KFSA,KFS-1
  VN(J)="SURPES"
210 CONTINUE
  DO 220 J=KFS,KFA-1
  VN(J)="SLACK"
220 CONTINUE
  DO 230 J=KFA,YT
  VN(J)="ARTIF"
230 CONTINUE
ENDIF
RETURN
END
SUBROUTINE MODI

C USE: PERFORMS THE SAME FUNCTION AS SUBROUTINE GBNDU AND SUBROUTINE CNNDV EXCEPT NO USER INTERFACE. NOTE THAT CONSTRAINTS WITH NEGATIVE RHS'S ARE MULTIPLIED BY -1.

C CALLED BY: PROGRAM EDUC

C CALLS : NONE

C VARIABLES:

C MODIFIED: A(*)*,C(*)*,INEG(*)*,NGC,MLC,PINEG(*)*,XB(*)

SUBROUTINE MODI

CHARACTER V*86,C*20,P*20.M*20.MH*3,FM*10,PINEG*,P*1,OBJ*10
INTEGER ARTV,BASIC,PK,NL,PR,PRS,DPTS,V,V,T,CB
COMMON/E1/A(20,60),ARTV(20),C(60,2),INEG(20),IFLAB(10),CB(20)
,,NGC,MLC,IA,INDEX,INDEXL,INDEXB,XB(20)
COMMON/E2/BASIC,K,KF,KFS,KFSA,KFST,OPTS,PK,P,S,PR,PRS,V,V,T,MLN
COMMON/E3/VN(20),CM(20),PM,MY,MY,PINEG(20),P(10),OFSW
IF(NING .EQ. 1) THEN
C PROBLEM STATED AS MAXIMIZATION
DO 160 J=1,V
C(J)=-C(J)
160 CONTINUE
ELSE
MM='MIN'
ENDIF
DO 300 I=1,K
IF:XB(I) .LT. 0.0) THEN
XB(I)=0.0
DO 200 J=1,V
A(I,J)=-A(I,J)
200 CONTINUE
ENDIF
C COUNT OF INEQUALITIES BY TYPE UPDATED DUE TO MULT BY -1
IF (INEG .EQ. 0) THEN
NL=NL-1
NGC=NGC+1
INEG(I)=1
PINEG(I)='V'
ELSEIF INEG .EQ. 1 THEN
NGC=NGC-1
MLC=MLC+1
INEG(I)=0
PINEG(I)='V'
ENDIF
ENDIF
300 CONTINUE
RETURN
END
SUBROUTINE INTF

CHARACTER VN#5, CN#5, PN#20, MM#10, FN#10, PINEG#1, P#1, OBJN#10
INTEGER ARTV, BASIC, PK, PR, FRS, OPTS, V, VT, CB, FES, DUAL, OUTP, 
.IF, FMT
COMMON/E1/A(20,60), ARTV(20), C(60), Z, I(NINEG#20), IFLAG#20
.MEC, NNC, NL, IA, INDEE, INDEH, INDEX#20
COMMON/E2/BASIC, K, KFA, KFS, KFSU, OPTS, PK, PR, FES, V, VT, M#NN
COMMON/E3/HOD, PES, DUAL, OUTP, IAS, ITab, ITab#2, TIE, FMT
COMMON/E4/VN#20, CN#20, FN#10, PINEG#20, P#10, OBJN

FORMATA

C *** VARIABLES INITIALIZED
DO 180 I=1,20
  ARTV(I)=0
DO 180 J=1,60
  A(I,J)=0.0
CONTINUE

DO 190 J=1,60
  C(J)=0.0
CONTINUE

DO 200 I=1,10
  IFLAG#(I)=0
CONTINUE

NEC=0
M#C=0
MLC=0
P=0.0
I=0
BASIC=0
OPTS=0
INDEF=0
INDEX#B=0
INDEXT=0
WRITE(1,110)CHAR(12)
WRITE(1,'(11(/),IX,'"INSURE DISK LP1 IS AVAILABLE.",7(/))')
PAUSE
C TRANSFER FILE OPENED AND FILE NAME READ
OPEN(3,FILE="LP1:LPDATA",STATUS='OLD',FORM='UNFORMATTED')
READ(3)FM
CLOSE(3,STATUS='KEEP')
WRITE(1,110)CHAR(12)
WRITE(1,'(9(/),5X,"INSURE THE DISK CONTAINING THE"//15X,A10
//10X,"MODEL IS AVAILABLE.",7(/))')FM
PAUSE
C FILE WHICH CONTAINS MODEL OPENED AND READ FROM DISK
OPEN(3,FILE=FM,STATUS='OLD',FORM='UNFORMATTED')
READ(3)PM,MM,KM,N,KM,NEC,MLC
DO 220 I=1,10
READ(3)IFLAG(I)
220 CONTINUE
DO 300 I=1,K
READ(3)INEG(I),PINEG(I),B(I)
DO 280 J=1,V
READ(3,A(I,J))
280 CONTINUE
300 CONTINUE
DO 320 J=1,V
READ(3)C(I,J)
320 CONTINUE
IF(IFLAG(5).EQ.1)THEN
DO 350 I=1,K
READ(3)CM(I)
350 CONTINUE
DO 360 J=1,V
READ(3)VM(J)
360 CONTINUE
DO 380 J=V+1,20
VM(J)=
380 CONTINUE
READ(3)OBJN
ENDIF
IFLAG(2)=1
CLOSE(3,STATUS='KEEP')
WRITE(1,110)CHAR(12)
WRITE(1,'(11(/),IX,"INSURE DISK LP1 IS AVAILABLE.",7(/))')
PAUSE
C FIND APPROPRIATE VALUE FOR BIG M
CM=0.0
BN=0.0
DO 400 J=1,V
IF(ABS(C(J)) .GT. 2M)THEN
BN=ABS(C(J))
ENDIF
400 CONTINUE
BN=(ANINT(BM))*10.0
IF(BM .LT. 1.0) THEN
  BM=10.0
ENDIF
RETURN
END
SUBROUTINE TDISPL

USE: DISPLAYS EITHER THE CURRENT CONSTRAINTS, COMPLETE LP MODEL,
OR THE CURRENT TABLEAU, DEPENDENT ON FLAGS SET IN CALLING
SUBROUTINE. VARIABLES AND CONSTRAINTS ARE DISPLAYED WITH
NAMES, IF PRESENT, AND THE BASIC VARIABLE OF THE CONSTRAINT
IS IDENTIFIED WHEN DISPLAYING THE CURRENT TABLEAU.
CALculates THE NUMBER OF 80 COLUMN WIDTHS TO DISPLAY COMPLETE
TABLEAU AND PRESENTS ON OUTPUT DEVICE IN SECTIONS. NO USER
INTERFACE.

CALLED BY: PROGRAM EDUC

SUBROUTINE CNNDU
SUBROUTINE OPT
SUBROUTINE READY
SUBROUTINE TCAL

CALLS: NONE

VARIABLES:

USED: A(I,J),C(I),CB(I),CN(I),FMT,IFLAG(3),IFLAG(9),K4n,OD.

MODIFIED: NONE

USES UCHECK2 IN UNIT27.CODE OVERLAY

SUBROUTINE TDISPL
CHARACTER VN(6),CN(6),FM(10),PINEQ(20),OBJN(10)
INTEGER ARTV,BASIC,PK,PKS,PRS.OPTS,V,VT,CB,PES,DIU.DUAL.OUTP.
.TIE,FMT,T
COMMON/E1/A(20,60),ARTV(20),C(60),I,INDE(10),IFLAG10,CB(20)
,.NEC.NSC.NLC.NDE.NDEE,INEE16,INDEX16.IB(20)
COMMON/E2/BASIC,K,KFA,KFS,PKS,PKS,PKS.PRS.PRS.PRS.V,VT,MNM
COMMON/E3/MOB,PE5,DIU.DUAL.OUTP,IFAB.IIFAB,IIFAB,IB,TIE,FMT
COMMON/E4/VN(20),CN(20),PN.MM.FN,PINEQ20,P(10),OBJN

110 FORMAT(A)
IF(IFLAG(9) .EQ. 1)THEN
0 ONLY CONSTRAINTS ARE DISPLAYED
WRITE(2,220)PN
220 FORMAT('Y,A20/10X,'CURRENT CONSTRAINTS'/)
ELSEIF(IFLAG(9) .EQ. 0)THEN
C OBJ FUNCTION AND CONSTRAINTS DISPLAYED
WRITE(2,230)PN,MM
230 FORMAT('Y,A20/7X,'CURRENT LP MODEL: ','N3,'IMIZE ','/
IF(IFLAG(5) .EQ. 1)THEN
WRITE(2,240)OBJN
240 FORMAT(A10)
ELSE
WRITE(2,250)
250 FORMAT(''
ENDIF

C NUMBER OF 80 COLUMN DISPLAYS REQUIRED DETERMINED

333
T=(VT/5)+1
EQ 470 N=1,T
IF(FLAG(5).EQ. 1) THEN
  C
  VARIABLE NAMES PRINTED AS COLUMN HEADERS
  WRITE(2,'(13X,9)')
  DO 270 J=(N5)-4,N5
  IF(J.GT. VT) THEN
    GO TO 270
  ENDIF
  WRITE(2,260) VM(J)
  WRITE(2,'(13X,9)')
  DO 290 J=(N5)-4,N5
  IF(J.GT. VT) THEN
    GO TO 290
  ENDIF
  WRITE(2,280) J
  CONTINUE
  C
  IF LAST 80 COLUMN DISPLAY, DISPLAY RHS
  IF(T.EQ. 1 .OR. N .EQ. 1) THEN
    WRITE(2,300)
  ELSE
    ENDIF
  C
  DISPLAY IS WITH OBJ FUNCTION
  IF(IFLAG(9).EQ. 0 .OR. IFLAG(9).EQ. 2) THEN
    WRITE(2,'("OBJ FUNCTION",1X,9)')
    DO 320 J=(N5)-4,N5
    IF(J.GT. VT) THEN
      GO TO 320
    ENDIF
    IF(FMT.EQ. 0) THEN
      WRITE(2,'("FMT",1X,9)') C(J)
    ELSE
      WRITE(2,'("FMT",1X,9)') C(J)
    ENDIF
  CONTINUE
  IF(T.EQ. 1. OR. N .EQ. 1) THEN
    IF(FMT.EQ. 0) THEN
      WRITE(2,'("=",1PE12.5,1X)')
    ELSE
      WRITE(2,'("=",1PE12.5,1X)')
    ENDIF
  ELSE
    WRITE(2,('""'))
  ENDIF
  ENDIF
ENDIF
IF(IFLAG(9) .EQ. 2) THEN
C BASIC VARIABLES ARE TO BE ANNOTATED
WRITE(2,'('"CN NAME VAR",2X,65('""'))')
ELSE WRITE(2,'('"CN NAME",2X,70('""'))')
ENDIF
C CONSTRAINT NUMBER, NAME, BASIC VARIABLE, COEFFICIENTS,
C INEQUALITY, AND RHS DISPLAYED
50 400 L=1,K
IF (L .GT. K) THEN
   GO TO 400
ENDIF
IF(IFLAG(5) .EQ. 1) THEN
   WRITE(2,'(12,1X,A6,4)')L,CM(L)
ELSE WRITE(2,'(12,7X,4)')L
ENDIF
IF(IFLAG(9) .EQ. 2) THEN
   WRITE(2,'(1X,12,1X,)')CB(L)
ELSE WRITE(2,'(4X,9)')
ENDIF
DO 370 J=(N#5)-4,N#5
IF (J .GT. N) THEN
   GO TO 370
ENDIF
IF (FMT .EQ. 0) THEN
   WRITE (2,'(IPE12.5,1X,)')A(L,J)
ELSE WRITE (2,'(F12.5,1X,)')A(L,J)
ENDIF
370 CONTINUE
IF (T .EQ. 1 .OR. N .EQ. 7) THEN
   IF (MNT .EQ. 0) THEN
      WRITE (2,'(A1,1X,IPE12.5)')PINEQ(L),XB(L)
   ELSE WRITE (2,'(A1,1X,F12.5)')PINEQ(L),XB(L)
   ENDIF
ELSE WRITE (2,'('')')
ENDIF
400 CONTINUE
IF(IFLAG(3) .EQ. 1) THEN
   PAUSE WRITE(2,110)CHAR(2)
   GO TO 470
ENDIF
IF (CUTP .EQ. 1) THEN
   PAUSE WRITE(2,110)CHAR(2)
ELSE WRITE(2,'('')')
ENDIF
470  CONTINUE
471  IF(FLAG(3)=0) THEN
472     IF(FLAG(9)=0) THEN
473        CLOSE(2)
474        RETURN
475     END
476  END
SUBROUTINE BASDIS
C
C USE: DETERMINES WHETHER OR NOT CURRENT TABLEAU WAS REQUESTED TO BE
C DISPLAYED AND IF SO, PROMPTS USER WHETHER OR NOT THE BASIC
C SOLUTION IS DESIRED TO BE DISPLAYED AND ON WHAT DEVICE. IF
C REQUESTED, DISPLAYS BASIC VARIABLES, NAMES AND VALUES OF
C CURRENT SOLUTION AND OR THE OBJECTIVE FUNCTION VALUE.
C CALLED BY: SUBROUTINE OPT
C CALLS : SUBROUTINE CHECK2(P,N,INV,INL,NEW)
C VARIABLES:
C USED: BASIC,C3(10),FNT,IFLAG(5),IFTAB,INV,INV,ITAB,K,OPTS,
C PM,VM(9),VIP(9),Z
C MODIFIED: P(0)
C
SUBROUTINE BASDIS
CHARACTER VM(6),CM(6),PM(20,MM)3,FN(10),PINEO(0),DBJN(10)
INTEGER ARTV,BASIC,PK,FKS,PR,PRS,OPTS,V,VT,CB,PES,0IU,DUAL,OUTP,
,TIE,FNT
COMMON/E1(A(20,60),ARTV(20),CM(60),I,INE(20),IFLAG(CB(20)
..,N,NEC,P,MLC,IA,INDEE,INDEE1,HOCHL,IBM20)
COMMON/E2/BASIC,K,FKA,KFS,FST,OPTS,PK,PKS,PK,PRS,VT,MINN
COMMON/E3/MOD,PES,01U,DUAL,OUTP,ITAB,IT,ITAB,IFTAB,EM, HIP,TIE,FNT
COMMON/E4/VM(20),CM(20),PM,VM,F4,PINEO(20),P(10),DBJN
WRITE(1,110)CHAR(12)
110 FORMAT(4)
C DETERMINES IF USER HAS SELECTED TABLEAU FOR OUTPUT
150 IF(BASIC .EQ. 1 .AND. ITAB .NE. 1) THEN
RETURN
ELSEIF(BASIC .NE. 1 .AND. OPTS .NE. 1 .AND. ITAB .EQ. 0) THEN
RETURN
ELSEIF(OPTS .EQ. 1 .AND. IFTAB .EQ. 2) THEN
RETURN
ENDIF
WRITE(1,110)CHAR(12)
C USER INPUTS SELECTION OF OUTPUT DEVICE, IF ANY
WRITE(1,'(6X,''WOULD YOU LIKE THE BASIC SOLUTION VALUES''/14X,
.'''DISPLAYED?'')
WRITE(1,''(6X,''1. DISPLAY ON SCREEN''/9X,''2. DISPLAY ON PRINTER'
.'''9X,''3. DO NOT DISPLAY'')':
WRITE(1,''(6X,''WHICH OPTION? '','(4)'')
READ(5,A12)P(1)
CALL CHECK2(P,1,3,INV,NEW)
IF(INV .EQ. 1) THEN
WRITE(1,230)
230 FORMAT(5X,''INVALID ENTRY, PLEASE REENTER'')
GO TO 200
ENDIF
IF(INV .EQ. 1) THEN
OPEN(2,FILE='''CONSOLE''))
ELSEIF(INV .EQ. 2) THEN
OPEN(2,FILE="PRINTER")
ELSE
RETURN
ENDIF
WRITE(1,110)CMAR(12)
WRITE(2,'(10X,A20/10X,"BASIC SOLUTION "',12,'/1"/PM,BASIC
IF(IFLAG(5) .EQ. 0) THEN
  DO 250 I=1,K
  IF(FMT .EQ. 0) THEN
    WRITE(2,'(5X,A6,''' = X('''',12,'''') = ''',FPE12.5)')VB(CB(I)),
    .CB(I),IB(I)
  ELSE
    WRITE(2,'(5X,A6,''' = X('''',12,'''') = ''',F12.5)')VB(CB(I)),
    .CB(I),IB(I)
  ENDIF
  CONTINUE
 IF(FMT .EQ. 0) THEN
    WRITE(2,'(18X,"\"',12,'\"")')
  ELSE
    WRITE(2,'(18X,"\"',12,'\"")')
  ENDIF
ELSE
  DO 280 I=1,K
  IF(FMT .EQ. 0) THEN
    WRITE(2,'(10X,"X('''',12,'''') = ''',FPE12.5)')CB(I),IB(I)
  ELSE
    WRITE(2,'(10X,"X('''',12,'''') = ''',F12.5)')CB(I),IB(I)
  ENDIF
  CONTINUE
  IF(FMT .EQ. 0) THEN
    WRITE(2,'(14X,"\"',12,'\"")')
  ELSE
    WRITE(2,'(14X,"\"',12,'\"")')
  ENDIF
ENDIF
IF(INEM .EQ. 2) THEN
  WRITE(2,'("/1")')
ELSE
  PAUSE
ENDIF
RETURN
END
C **********************************************************************
C MODULE 2 UNIT27
C UNIT USES: NONE
C
C SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
C USE: SEE MODULE 1, UNIT17
C CALLED BY: SUBROUTINE BAGDIS
C
C SUBROUTINE CMDU
C SUBROUTINE COMDU
C SUBROUTINE OPTION
C SUBROUTINE PIVOT
C SUBROUTINE READY
C CALLS : NONE
C VARIABLES: SEE MODULE 1, UNIT17
C
C SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
CHARACTER ALLOW(11)
INTEGER D,HVAL
DATA ALLOW/"1","2","3","4","5","6","7","8","9","0","/
INEW=0
INVAL=0
DO 300 I=1,10
DO 200 J=1,10
C CHECKS FIRST FOR BLANK CHARACTERS
IF(E(I) .EQ. ALLOW(I)) THEN
   GO TO 300
ELSEIF(E(I) .EQ. ALLOW(J)) THEN
   INEW=INEW+1+ICHAR(E(I))-46
   GO TO 300
ELSEIF(J .EQ. 10) THEN
   INEW=INEW
   RETURN
ENDIF
200 CONTINUE
300 CONTINUE
IF(INEW .EQ. 0 .OR. INEW .GT. HVAL) THEN
   INVAL=1
   INEW=0
RETURN
ENDIF
RETURN
END
C MODULE 3 UNIT30
C USES: UNIT32 THRU UNIT37
C
C PROGRAM PROBS
C USE: MAIN PROGRAM OF MODULE 3. PURPOSE OF MODULE IS TO PROVIDE A
C MEANS OF SOLVING LINEAR PROGRAMMING PROBLEM IN THE MOST
C EFFICIENT SIMPLEX METHOD. THIS MODULE ALLOWS THE USER TO
C SPECIFY THE PRIMAL OR DUAL PROBLEM, AND FURTHER, PRIMAL OR
C DUAL SIMPLEX APPLICATION TO THE SELECTED PROBLEM. SELECTED
C DISPLAY OF OUTPUT ALLOWS USER TO LIMIT OUTPUT TO THAT
C REQUIRED. MODULE 3 CONSISTS OF 7 SEPARATELY COMPILED UNITS
C (UNIT30, UNIT32 THRU UNIT37) WITH ALL UNITS EXCEPT UNIT0
C BEING OVERLAY UNITS.
C PROGRAM PROBS ACTS AS A MEMORY RELEASE LOCATION. WHENEVER
C THE PROGRAM CONTROL RETURNS TO THIS UNIT, ALL OVERLAY UNITS
C ARE RELEASED FROM MEMORY PRIOR TO NEW UNITS BEING SUMMONED.
C CALLED BY: NONE
C CALLS : SUBROUTINE ACNCH
C SUBROUTINE ASKASK
C SUBROUTINE BIG BGN
C SUBROUTINE CONVRT
C SUBROUTINE INDEX
C SUBROUTINE INKP
C SUBROUTINE Modify
C SUBROUTINE MODIFP
C SUBROUTINE OPTB
C SUBROUTINE OPTN
C SUBROUTINE PSHED
C SUBROUTINE UDOSPL
C SUBROUTINE WORK
C VARIABLES:
C USED: ASK,BM,DUAL,IFLAG(7).IFLAG(9).KFA,MGC,PRCB,VT
C MODIFIED: BASIC,C(*)
C
C $USES UCHECK2 IN UNIT37.CODE OVERLAY
$USES UDOSPL IN UNIT36.CODE OVERLAY
$USES UPSHED IN UNIT35.CODE OVERLAY
$USES UCONVRT IN UNIT34.CODE OVERLAY
$USES UOPTN IN UNIT32.CODE OVERLAY
$USES UOPN IN UNIT31.CODE OVERLAY
PROGRAM PROBS
CHARACTER WM#,CN#,CM#,W#,CN#,CM#,CM#,W#,CN#,CM#,CM#,W#,CN#,CM#,CM#
INTEGER ARTV,BASIC,PK,PR,OPT,V,VT,EB,DUAL,OUTP,FMT,PRB,ASK
COMMON/P1/A(20,60),ARTV(20),C(a0),I,MEG(20),IFLAG(10),CB(20),
,K,F,Y,MM,BASiC,OPTS,PR
COMMON/P3/N,MGC,NLC.IA,INDAT.INDAT.IAND,INDEXL.KFA,KFS,KFU,
.PK,PR
COMMON/P5/DUAL,OUTP,ITSAB,ITSAB,FMT,PRB
COMMON/P4/VM(20),CM(20),PN,MM,F,MN(FM1(20),OR5
OPEN(1,FILE='CONSOLE:1')
OPEN(5,FILE='CONSOLE:1')
WRITE(1,110)CHAR(12)
110 FORMAT(a)
CALL PSHED
C ROUTINE CALLED WHICH ALLOWS USER TO CHANGE DEFAULTS
120 CALL OPTN
BASIC=0
IF(PROBT.EQ.1.AND. DUAL.EQ.1)THEN
C USER HAS ELECTED TO SOLVE PRIMAL PROBLEM WITHOUT DUAL PIVOTS
GO TO 140
ELSEIF(PROBT.EQ.2)THEN
C USER HAS ELECTED TO SOLVE DUAL PROBLEM
CALL CONVRT
ENDIF
IF(DUAL.EQ.2)THEN
C USER HAS ELECTED TO USE DUAL PIVOTS
CALL ACNCH
ENDIF
C ROUTINE CALLED WHICH INITIALIZES VARIABLES AND READS MODEL
140 CALL INRO
IF(DUAL.EQ.1)THEN
C OBJ FUNCTION MODIFIED FOR PRIMAL PROBLEM
CALL MODIFP
ELSE
C OBJ FUNCTION MODIFIED FOR DUAL PROBLEM
CALL MODIFD
ENDIF
C ROUTINE CALLED WHICH ADDS SLACK, SURPLUS, AND ARTIFICIAL VARIABLES
CALL INDEX
DO 160 J=KFA,VT
C(J)=BM
160 CONTINUE
C CHECKS IF ARTIFICIAL VARIABLES HAVE BEEN ADDED
IF((NEC+NGC).NE.0)THEN
   CALL BIGN
ENDIF
170 BASIC=BASIC+1
CALL OPTB
IF(IFLAG(9).EQ.1)THEN
C FLAG INDICATES TABLEAU IS TO BE DISPLAYED
CALL TDISPL
ENDIF
IF((OPTS.EQ.1).OR. (IFLAG(7).EQ.1))THEN
C LAST TABLEAU EITHER OPTIMAL OR UNBOUNDED
CALL ASKQ(ASK)
IF(ASK.EQ.1)THEN
   GO TO 120
ENDIF
ENDIF
CALL WORK
GO TO 170
STOP
END
C MODULE 3 UNIT32
C UNIT USES: UNIT37
C
C SUBROUTINE OPTN
C USE: DISPLAYS DEFAULT OPTION VALUES AND SOLICITS RESPONSE TO
C CHANGE THESE DEFAULTS. IF OPTION IS SELECTED TO BE CHANGED,
C USER REVIEWS MENU AND SELECTS DESIRED METHOD, THEN IS
C RETURNED TO DEFAULT OPTION DISPLAY. SOME OPTIONS ARE CHANGED
C UPON SELECTION DUE TO ONLY TWO METHODS BEING POSSIBLE.
C OPTIONS ARE RESET TO PROGRAMMER SPECIFIED DEFAULT UPON EACH
C CALL TO ROUTINE.
C CALLED BY: PROGRAM PROBS
C CALLS : SUBROUTINE CHECK2(P,N,M,INVAL,NEW)
C VARIABLES:
C USED: INEW, INVAL
C MODIFIED: DUAL,FNT,FN,IBTAB,IFTAB,OUTP,PN,PROBT
C
NUSES UCHECK2 IN UNIT37 CODE OVERLAY
SUBROUTINE OPTN
CHARACTER VNA(6),CMN(6),PNK6,MMN8,FNP10,PINBB1,P(10) '1',OBUN '10
INTEGER ARTV,BASIC,PK,PR,OPTS,N,VI,C9,DUAL,OUTP,FNT,PROBT
COMMON/P3/DUAL,OUTP,ITAB,IFTAB,FNT,PROBT
COMMON/P4/VN(20),CM(20),PNK,MM,FN,PIBEN(20),OB2W
100 WRITE1,110)CHAR(12)
110 FORMAT(A)
WRITE1,110) '111','"INSURE DISK LP1 IS AVAILABLE."',7(',')')
PAUSE
WRITE1,110)CHAR(12)
WRITE1,110)
130 FORMAT(4X,'PROBLEM SOLVER OPTION SELECTION')
C TRANSFER FILE OPENED AND FILE NAME READ
OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
READ(3,IN)
CLOSE(3,STATUS='KEEP')
150 WRITE1,110) '111','"THE PROBLEM CURRENTLY IDENTIFIED AS THE"',7('0X)
WRITE1,110) '111','"PROBLEM TO BE STUDIED IS:"
READ(5,17)
IF(ICHAR(P(1)) .EQ. 7611) THEN
WRITE1,110) '111','"PLEASE ENTER THE DISK DRIVE NUMBER AND"',7('0X)
WRITE1,110) '111','"NAME OF THE FILE YOU WANT TO STUDY."
READ(5,17)
WRITE1,110) '111','"MODEL TO BE STUDIED ="
READ(5,17)
WRITE1,110) '111','"PER DISK IS IN THE PROPER DRIVE."')
WRITE1,110) '111','"WAS SAVED PREVIOUSLY AND ALSO THAT THE"',7('0X)
WRITE1,110) '111','"ARE CORRECTIONS NEEDED?"
READ(5,17)
160 WRITE1,110)
180 FORMAT(7X,'ARE CORRECTIONS NEEDED? ',2X)
READ(5,17)
ELSEIF(CHAR(P(1)) .NE. 78) THEN
  WRITE(1,190)
  FORMAT('SI, 'INVALID ENTRY, PLEASE REENTER')
  PAUSE
  GO TO 160
ENDIF
WRITE (1,110) CHAR(12)
WRITE(1,'(I:/:,IX, 'INSURE DISK LP1 IS AVAILABLE',',7(//))
PAUSE
C TRANSFER FILE REWRITTEN TO INDICATE NEW USER MODEL SELECTION
OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
WRITE(3) EN
CLOSE(3,STATUS='KEEP')
ELSEIF(CHAR(P(1)) .NE. 99) THEN
  WRITE(1,190)
  PAUSE
  GO TO 100
ENDIF
WRITE(1,110) CHAR(12)
WRITE(1,'(I:/:,IX, 'INSURE DISK LP2 IS AVAILABLE',',7(//))
PAUSE
C DEFAULT OPTIONS SET
PROBT=1
DUAL=1
OUTP=1
FMT=1
ITAB=1
ITFAT=1
200 WRITE(1,110) CHAR(12)
C DEFAULT OPTIONS DISPLAYED
WRITE(1,'(12X,'DEFAULT OPTIONS'/5X,'ENTER OPTION NUMBER TO CHANGE
5E'//')
WRITE(1,'(12X,'1. PROBLEM TO SOLVE'/1X,9I1)'
IF(PROBT .EQ. 1) THEN
  WRITE(1,'(12X,'PRIMAL')'
ELSE
  WRITE(1,'(12X,'DUAL')'
ENDIF
WRITE(1,'(12X,'2. SOLVE BY DUAL PIVOTS'/1X,16I1)'
IF(DUAL .EQ. 1) THEN
  WRITE(1,'(12X,'DUAL')'
ELSE
  WRITE(1,'(12X,'PRIMAL')'
ENDIF
WRITE(1,'(12X,'3. OUTPUT LOCATION'/1X,12I1)'
IF(OUTP .EQ. 1) THEN
  WRITE(1,'(12X,'PRINT')'
ELSE
  WRITE(1,'(12X,'SCREEN')'
ENDIF
WRITE(1,'(3X,'DEFAULT OPTIONS'/1X)'
STOP
WRITE(1,'(/,,"4. OUTPUT FORMAT",16X,")')
IF(FMT .EQ. 1) THEN
  WRITE(1,'("F FORMAT")')
ELSE
  WRITE(1,'("E FORMAT")')
ENDIF
WRITE(1,'(/,,"5. TABLEAUS TO BE DISPLAYED",/6X,"INITIAL",26X,\"")
IF(ITAB .EQ. 1) THEN
  WRITE(1,'("Y")')
ELSE
  WRITE(1,'("M")')
ENDIF
WRITE(1,'(/,,"6. NO CHANGES"/"" See documentation for explanation")
WRITE(1,'(/,,"Which option (enter 1-6)? "")
READ(5,(A1)) P(I)
CALL CHECK2(P,1,6,INVAL,INEW)
IF(INVAL .EQ. 1) THEN
  WRITE(1,150)
PAUSE
GO TO 200
ENDIF
GOTO(230,250,290,330,350,460): INEW
C PROBLEM TYPE TO BE SOLVED CHANGED
230 IF(PROBT .EQ. 1) THEN
  PROBT=2
ELSE
  PROBT=1
ENDIF
GO TO 200
C DUAL PIVOT OPTION CHANGED
260 IF(DUAL .EQ. 1) THEN
  DUAL=2
ELSE
  DUAL=1
ENDIF
GO TO 200
C OUTPUT LOCATION CHANGED
290 IF(OUTP .EQ. 1) THEN
  OUTP=2
ELSE
  OUTP=1
ENDIF
GO TO 200

344
C TABLEAU OUTPUT OPTIONS DISPLAYED
350 WRITE(1,110)CHAR(12)
   WRITE(1,130)
   WRITE(1,131)
370 IF(ICHAR(P(1)) .EQ. 89)THEN
   ITAB=1
   ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
   ITAB=2
   ELSE
   WRITE(1,190)
   GO TO 370
   ENDIF
390 WRITE(1,132)
   READ(5,133)P(1)
   IF(ICHAR(P(1)) .EQ. 89)THEN
   WRITE(1,134)
   ENDIF
   IF(ICHAR(P(1)) .EQ. 78)THEN
   WRITE(1,135)
   ENDIF
   ELSEIF(ICHAR(P(1)) .EQ. 89)THEN
   WRITE(1,136)
   ENDIF
   IF(ICHAR(P(1)) .EQ. 78)THEN
   WRITE(1,137)
   ENDIF
400 WRITE(1,138)
   READ(5,139)P(1)
   IF(ICHAR(P(1)) .EQ. 89)THEN
   WRITE(1,140)
   ENDIF
   IF(ICHAR(P(1)) .EQ. 78)THEN
   WRITE(1,141)
   ENDIF
   ELSE
   WRITE(1,142)
   GO TO 400
   ENDIF
410 WRITE(1,143)
   READ(5,144)P(1)
   IF(ICHAR(P(1)) .EQ. 89)THEN
   WRITE(1,145)
   ENDIF
   IF(ICHAR(P(1)) .EQ. 78)THEN
   WRITE(1,146)
   ENDIF
   ELSE
   WRITE(1,147)
   GO TO 410
   ENDIF
430 IF(FMT .EQ. 1)THEN
   FMT=0
   ELSE
   FMT=1
   ENDIF
MODULE 3 UNIT33
UNIT USES: NONE

SUBROUTINE WORK
USE: SEE MODULE 2, UNIT24, SUBROUTINE WORK
CALLED BY: PROGRAM PROBS
CALLS: NONE
VARIABLES: SEE MODULE 2, UNIT24, SUBROUTINE WORK

SUBROUTINE WORK
INTEGER ARTV,BASIC,PK,PR,OPTS,Y,VT,K
COMMON/P1/A:20,6/,ARTV(20),C(60),I,INEQ(20),!FLAG(10),CR(20),
      .XB(20),K,Y,VT,MINN,BASIC,OPTS,BM
COMMON/P2/HEC,HLC,H,INEQE,INDEX6,INDEXL,IFA,KFS,KFSK,IFEU,
      .PK,PR
PELE=A(PR,PK)
50 200 J=1,VT
   A(PR,J)=A(PR,J)/PELE
   CONTINUE
   1B(PR)=B(PR)/PELE
   CB(PR)=PK
   G0 300 I=1,K
      IF(I .EQ. PR)THEN
         250 I=I
      ENDIF
      HOLD=A(I,PK)
      G0 300 J=1,VT
         A(I,J)=A(I,J)-HOLD*A(PR,J)
   CONTINUE
   (PR)=B(PR)-HOLD*B(PR)
   CONTINUE
   HOLD=CB(PR)
   DG 350 J=1,VT
       C(J)=C(J)-HOLD*A(PR,J)
   CONTINUE
   Z=Z-HOLD*B(PR)
   RETURN
END
SUBROUTINE OPTB
CHARACTER (VNS6, CNS6, PNS20, SM3FN$IQ, PINEGII, P(I0)SI, DBU-N6,10)
INTEGER ARIV, BASIC, PK, OPTS, V, VT, CB, DUAL, OUTP, FMT, PROBT
COMMON/A(IA20, 60), ARTV(20), C(160), I, INFD(20), IFLAG(10), CB(20),
. VB(20), K, V, VT, MM, BASIC, OPTS, BM
COMMON/P2/NEC, NLC, IA, INDEXE!, INDEXE!K, KFA, KFS, KFSU, KMK, PK, PR
COMMON/P3/DUAL, OUTP, ITAB, IFTAB, HMT, PROBT
COMMON/PA/VM(20), CN(20), FN.MM, FN.PINE(20), BBJN
IFLAG(4)=0
IFLAG(6)=0
INFP=0
BNEG=0.0
IF (DUAL .EQ. 2) THEN
C USER HAS ELECTED TO PERFORM DUAL PIVOTS
GO TO 300
ENDIF
C FINDS LARGEST Z(J)-C(J)
110 DO 130 J=1, VT
   IF (C(J) .GE. BNEG) THEN
      GO TO 130
   ELSE
      BNEG=C(J)
   ENDIF
   CONTINUE
   IF (BNEG .LT. -0.001) THEN
   C NO NEGATIVE Z(J)-C(J) EXIST! SO OPTIMAL
   OPTS=1
   ENDIF
C CHECKS FOR INFEASIBILITY
DO 150 I=1, Y
   IF (CRI(I) .LT. XFA) THEN
      IF (XBI(I) .LT. 0.0) THEN
         348
IMFP=1
ENDIF
ELSEIF(XB(I) .LE. 0.0)THEN
  GO TO 150
ELSE
  IMFP=1
ENDIF
150 CONTINUE
IF(OPTS .EQ. 0)THEN
  C PREVIOUS SOLUTION NOT OPTIMAL SO FIND PIVOT ROW
  SPR=10.0E8
  DO 190 I=1,K
    IF(A(I,PK) .LE. 0.0001)THEN
      GO TO 190
    ELSEIF((XB(I)/A(I,PK)) .GE. SPR)THEN
      GO TO 190
    ELSE
      SPR=XB(I)/A(I,PK)
      PR=I
    ENDIF
  190 CONTINUE
  C CHECKS FOR UNBOUNDEDNESS
  IF(SPR .GE. 10.0E6)THEN
    IFLAG(7)=1
  ENDIF
ENDIF
GO TO 500
C DUAL PIVOT CALCULATIONS PERFORMED
300 DO 320 J=1,VT
    IF(C(J) .LT. 0.0)THEN
      GO TO 110
    ENDIF
320 CONTINUE
DO 340 I=1,K
    IF(XB(I) .GE. 6MEG)THEN
      GO TO 340
    ELSE
      6MEG=XB(I)
      PR=I
    ENDIF
340 CONTINUE
IF(6MEG .EQ. 0.0)THEN
  OPTS=1
  GO TO 500
ELSE
  SPR=-10.0E8
  DO 370 J=1,VT
    DO 360 I=1,K
      IF(CB(I) .EQ. J)THEN
        GO TO 370
      ENDIF
360 CONTINUE
IF(AIPR,J) .GE. -.0001) THEN
60 TO 370
ELSEIF(C(J)/A(IPP,J)) .LE. SPR) THEN
60 TO 370
ELSE
SPR=C(J)/A(PR,J)
PK=1
ENDIF
370 CONTINUE
ENDIF
IF(SPR .LE. -10.E6) THEN
IFLAB (7,)
-500
MFTS .EQ. W) THEN
I'FINF .EQ. W) THEN
60 TC 600
ENDIF
C CHECKS FOR MULTIPLE OPTIMAL SOLUTIONS
DO 540 J=1,YT
IFLAG(S) : 0
DO 520 =1,K
IF(CBMI .EQ. J) THEN
IFLAG(S) :=1
ENDIF
520 CONTINUE
IF(IFLAG(8) .EQ. 0) THEN
IF(CMJ .LT. -.0001 .AND. CJ .GT. -.0001) THEN
IFLAG(4) =1
ENDIF
ENDIF
540 CONTINUE
ENDIF
IF(IFLAG(7) .EQ. 1) THEN
60 TO 609
ENDIF
C CHECKS FOR DEGENERACY
DC 560 I=1,K
IF(XBI .GE. -.0001 .AND. XB(I) .LE. .0001) THEN
IFLAG(6) =1
ENDIF
560 CONTINUE
C DETERMINES IF PRESENT SOLUTION TO BE DISPLAYED
660 IF(BASIC .EQ. 1 .AND. ITAB .EQ. 1) THEN
60 TO 700
ELSEIF(BASIC .NE. 1) THEN
IF(OPTS .EQ. 0 .AND. ITAB. NE. 0) THEN
IF((FLOAT(BASIC-1)/(FLOAT(IBTAB))) .EQ. FLOAT((BASIC-1)/
.1TAB)) THEN
60 TO 700
ELSE
IFLAG(9) =0
RETURN
350
ENDIF
ELSEIF(OPTS .EQ. 1 .AND. IFLAG .EQ. 1) THEN
  GO TO 700
ELSEIF(IFLAG(7) .EQ. 1 .AND. IFTAB .EQ. 0) THEN
  GO TO 700
ENDIF
ENDIF
IFLAG(9)=0
RETURN
C
PROPER OUTPUT DEVICE OPENED AND TABLEAU HEADER DISPLAYED
700 IF(OUTPF .EQ. 1) THEN
  OPEN(2,FILE='CONSOLE: ')
  WRITE(2,'(A12)')CHAR(12)
ELSE
  OPEN(2,FILE='PRINTER: ')
ENDIF
WRITE(2,'(10X,A20/10X, ''BASIC SOLUTION '',12)')BN,BASIC
IF(OPTS .EQ. 1 .OR. IFLAG(7) .EQ. 1) THEN
  WRITE(2,'(10X, ''FINAL TABLEAU - '' ,12)')
  IF(INF .EQ. 1) THEN
    WRITE(2,'(10X, ''INFEASIBLE'' )')
  ELSEIF(IFLAG(7) .EQ. 1) THEN
    WRITE(2,'( ''UNBOUNDED'' )')
  ELSE
    WRITE(2,'( ''OPTIMAL'' )')
  ENDIF
  IF(IFLAG(6) .EQ. 1) THEN
    WRITE(2,'(12X, ''DEGENERATE'' )')
  ENDIF
ENDIF
ENDIF
IF(OPTS .EQ. 1 .AND. IFLAG(4) .EQ. 1) THEN
  WRITE(2,'(5X, ''MULTIPLE OPTIMAL SOLUTIONS EXIST'' )')
ENDIF
WRITE(2,'(//)
IFLAG(9)=1
RETURN
END
SUBROUTINE CONVRT
CHARACTER VN6, CM6, PM20, MM5, FN10, PINE6I, PL10, OB6N10, VN2*6
INTEGER ARTV, BASIC, PK, PR, OPTS, V, VT, CB, DUAL, OPT, PROBT, V2, K2
COMMON/P1/A, P2/ARTV, C, I, IMAE=I100, IFILAG=I10, CB, VN120, KB
COMMON/F1/NEC, NEC, NLC, IA, INDEXE, INDEXG, V2, X2, X2, C2, S2, Y2, K2
COMMON/B1/V2, CM20, PN, MM, PM, PINE120, OB20
DIMENSION C2=20, VN2=20)

SUBROUTINE CALLED WHICH INITIALIZES VARIABLES AND READS MODEL
CALL INRD
IF(MAXMN .EQ. 1) THEN

C PROBLEM STATED AS A MAXIMIZATION PROBLEM
DO 110 L=1, L
IF(IINEQ=1) .EQ. 1) THEN
DO 100 J=1, V
A(L,J)=C(L,J)
100 CONTINUE
XB(I)=XB(I)
ENDIF
110 CONTINUE ELSE
DO 120 L=1, L
IF(IINEQ=1) .EQ. 0) THEN
DO 120 J=1, V
A(L,J)=C(L,J)
120 CONTINUE
XB(I)=XB(I)
ENDIF
130 CONTINUE ENDIF

C RHS PLACED IN TEMPORARY STORAGE LOCATION
DO 150 L=1, L
C2(I)=IB(I)
150 CONTINUE
DO 160 I=K+1,20
C2(I)=0.0
160 CONTINUE
C PRIMAL C(I)-Z(I) VALUES CONVERTED TO RHS
DO 170 J=1,V
IB(J)=C(J)
170 CONTINUE
C NUMBER OF DUAL VARIABLES INCLUDING UNCONSTRAINED VARIABLES FOUND
IF(NEC.NE.0)THEN
V2=K+NEC
ELSE
V2=K
ENDIF
K2=V
C COEFFICIENT MATRIX ROTATED AND PLACED IN UNUSED ORIGINAL A MATRIX
DO 190 I=1,K
DO 180 J=1,V
A(J,1+20)=A(I,J)
180 CONTINUE
190 CONTINUE
C VARIARLSS ADDED TO ALLOW FOR UNCONSTRAINED VARIABLES
IF(NEC.NE.0)THEN
IE=0
DO 210 I=1,K
IF(INEW(I).EQ.2)THEN
IE=IE+1
WRITE(1,20)CHAR(12)
WRITE(6,20)('VARIABLE ''',I2,''' HAS BEEN ADDED DUE TO ''',I2,''' BEING UNCONSTRAINED IN EIGN.'',7(/I,'K+IE','
PAUSE
DO 200 J=1,K
IF(A(J,1+20).NE.0.0)THEN
A(J,K+IE+20)=-1.0*I(A(J,1+20))
ENDIF
200 CONTINUE
IF(C2(I).NE.0.0)THEN
C2(K+IE)=-1.04*C2(I)
ENDIF
210 CONTINUE
ENDIF
IF(NEC.NE.0)THEN
N=1
ELSE
N=0
ENDIF
NEC=0
NGC=0
COUNT BY INEQUALITY TYPE PERFORMED
DO 220 I=1,K2
   IF(MXMN .EQ. 1) THEN
      INEG(I)=1
      PINEG(I)="\'>"'
      NNC=NNC+1
   ELSE
      INEG(I)=0
      PINEG(I)="<"'
      NLC=NLC+1
   ENDIF
220 CONTINUE
IF(MXMN .EQ. 1) THEN
   MXMN=2
   MN="MIN"'
ELSE
   MXMN=1
   MN="MAX"'
ENDIF
IF(IFLAE(5) .EQ. 1) THEN
   MODEL INCLUDES NAMES SO NAMES ARE CHANGED TO REFLECT DUAL PROB
   OBJN=''
   90 CONTINUE I=1,K
      VN2(I)=CN(I)
230 CONTINUE
   DO 240 J=1,V
      CN(J)=WN(J)
240 CONTINUE
   DO 250 I=K+1,20
      VN2(I)=''
250 CONTINUE
ENDIF
N=1
CALL NFILE(N)
DUAL MODEL WRITTEN TO DISK UNDER USER SPECIFIED NAME
WRITE(SIPM,MXMN,MN,K2,V2,NC,NNC,NLC)
DO 260 I=1,10
   WRITE(S1:IFLAG(I)
260 CONTINUE
   DO 280 J=1,K2
      WRITE(S1:INEG(I),PINEG(I),X(I)
   DO 270 J=1,V2
      WRITE(S1:AI(J),J20)
270 CONTINUE
280 CONTINUE
   DO 290 J=1,V2
      WRITE(S1:C2(J)
290 CONTINUE
IF(IFLAG(5) .EQ. 1) THEN
   DO 300 I=1,K2
      WRITE(S1:CN(I)
300 CONTINUE
   ENDIF
300 CONTINUE
   DO 310 J=1,V2
      WRITE(3),VMZ(J)
310 CONTINUE
   WRITE(3),IJB
ENDIF
CLOSE(3,STATUS='KEEP')
WRITE(1,'(A)')CHAR(12)
WRITE(1,'(114/),II,','INSURE DISK   LP2   IS AVAILABLE.'',?(/):')
PAUSE
RETURN
END
C SUBROUTINE ACNCH
C USE: DEPENDENT UPON USER OPTION SELECTION, EXAMINES PRESENT MODEL
C TO INSURE A NEGATIVE C(I) WILL BE PRESENT. IF NOT PRESENT,
C CONSTRAINT IS ADDED TO INSURE AN INITIAL PRIMAL PIVOT WILL
C BE POSSIBLE. USER IS NOTIFIED OF ADDED CONSTRAINT. ROUTINE
C WRITES DATA FILE WHICH CORRESPONDS TO MODEL WITH ADDED
C CONSTRAINT.
C CALLED BY: PGAMS PGMS
C CALLS : SUBROUTINE INRD
C SUBROUTINE NFILE(N)

C VARIABLES:
C USED: BM, C(i), IFLAG(1) THRU IFLAG(10), MM, MM*1, NEC, NEC, OBJN
C FM, V, VN(N)
C MODIFIED: A(I, N), CN(I), INEQ(I), IT, K, M, HLC, PINEQ(I), YB(I)

C ******************
C SUBROUTINE ACNCH
CHARACTER VN(K, MM), C(N, MM*1, MM*1, MM*10, OBJN*10,
INTERES ARG, BASIC, PK, PK, OPTS, V, VT, CB, DUAL, DUAL, FMT, PROB1
COMMON(P1A, P1B, P1C, ARTV(20), C(60), I, INEQ(20), IFLAG(10), CB(20),
X8(20), K, V, V, MM, BASIC, OPTS, BM
COMMON(P2AE, NEC, NGE, MLC, IA, INDE, INDEXI, INDEXL, KFA, KFS, KFSU, .PK, PR
COMMON/P3/DUAL, DUAL, ITAB, IETAB, IETAB, FMT, PROB1
COMMON/P4/VN(20), CN(20), PM, M, M, FK, PINEQ(20), OBJN
C ROUTINE CALLED WHICH INITIALIZES VARIABLES AND READS MODEL
CALL INRD
IT=0
IF(MM .EQ. 1) THEN
C PROBLEM STATED AS A MAXIMIZATION PROBLEM
DO 160 J=1, V
IF(C(i) .LE. 0.0) THEN
IT=1
ENDIF
160 CONTINUE
ELSE
DO 180 J=1, V
IF(C(i) .LT. 0.0) THEN
IT=1
ENDIF
180 CONTINUE
ENDIF
C DETERMINES IF INITIAL PIVOT POSSIBLE
IF(IT .EQ. 0) THEN
K=0
GO TO 260
ENDIF
C TO INSURE INITIAL PIVOT, CONSTRAINT ADDED
WRITE(1, '(A)') CHAR(12)
K=K+1
INEQ(K)=0
PINEQ(K)='<'
NL(NL)+1
1B(K)=6M
DO 199 J=1,V
   A(K,J)=1.0
190 CONTINUE
WRITE (1,'(A)')CHAR(12)
WRITE (1,'(I)X')2M,'"A CONSTRAINT HAS BEEN ADDED TO THIS"/I3X,""PRO
BLEN TO INSURE AN INITIAL PIVOT MAY"/15X,""BE PERFORMED.""/THE
CONSTRAINT IS THE SUM OF X(1) THRU"''")
WRITE (1,'(5X,'"X(",12,"') IS LESS-THAN OR EQUAL TO"/3X,1PE12.5,2
.',""THE VALUE OF BIG M")"/"(I)')V,8M
PAUSE
IF(IFLAG(5).EQ.1)THEN
   CM(K)="AGRES"
ENDIF
N=2
200 CALL WRITE(N)
C NEW FILE CONTAINING MODEL WITH ADDED CONSTRAINT WRITTEN TO DISK
WRITE (3)PH,MINN,MM,X,V,NEG,NBC,NL
DO 230 I=1,10
   WRITE (3)IFLAG(I)
230 CONTINUE
DO 250 I=1,K
   WRITE (3)INEQ(I),PINEQ(I),XB(I)
DO 240 J=1,V
   WRITE (3)A(I,J)
240 CONTINUE
DO 250 CONTINUE
DO 270 J=1,V
   WRITE (3)C(J)
270 CONTINUE
IF(IFLAG(5).EQ.1)THEN
   DO 290 I=1,K
      WRITE (3)CM(I)
290 CONTINUE
   DO 310 J=1,V
      WRITE (3)VM(J)
310 CONTINUE
   DO 330 J=M+1,20
      VM(J)=0
330 CONTINUE
   WRITE (3)OBJW
ENDIF
CLOSE (3,STATUS='KEEP')
WRITE (1,'(A)')CHAR(12)
WRITE (1,'(11(/,1X,""INSURE DISK LP2 IS AVAILABLE,"",7(/))')
PAUSE
RETURN
END
SUBROUTINE INRD

USE: Initializes variables to IZERD except character variables. Prompts user to insure disk with file to be studied is present and reads file from disk.

CALLED BY: PROGRAll PRODS

SUBROUTINE CONVRT

CALLS: NONE

VARIABLES:

USED: NONE

MODIFIED: A(*,*), ARTV(*), BASIC, EM, C(*), CB(*), CJ, CN(*), FN, IP,

IFLAG(1) thru IFLAG(10), INDEXE, INDEXG, INDEXL, INDEX(1), K,

NEC, NHC, NLC, NDN, OPTS, PINE(1), VM(*), VM(*), XB(*)

SUBROUTINE INRD

CHARACTER VM(*), CN(*), PN(*), MN(*), FN(*), P(10)(*), OBJN(*)

INTEGER ARTV(*), BASIC, PK, PR, OPTS, V, VT, CB, DUAL, OUTP, FM, PROST

COMMON/P/I(120, 80), ARTV(120), C(100), I, IUNE(120), IFLAG(10), CB(20),

XB(20), K, V, VM(*), BASIC, OPTS, BN

COMMON/P2/NHC, NEC, NLC, IP, INDEXE, INDEXG, INDEXL, INDEX(1), XFA, KFS, KFSA, KFS,

.PK, PR

COMMON/P4/VN(20), CN(20), PN(*), MN(*), FILE(*), OBJN(*)

DO 150 I = 1, 20

ARTV(I) = 0

CB(I) = 0

INEQ(I) = 0

XB(I) = 0, 0

DO 130 J = 1, 60

A(I, J) = 0, 0

150 CONTINUE

150 CONTINUE

DO 170 J = 1, 60

C(I) = 0, 0

170 CONTINUE

DO 190 I = 1, 10

IFLAG(I) = 0

190 CONTINUE

NEC = 0

NHC = 0

NLC = 0

I = 0, 0

K = 0

BASIC = 0

OPTS = 0

INDEXE = 0

INDEXG = 0

INDEXL = 0

WRITE(1, *'(A)' 'CHAR(12)

WRITE(1, *'(9I1), 5X,' 'INSURE THE DISK CONTAINING THE''//15X,A10

358
FILE WHICH CONTAINS MODEL OPENED AND READ
OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
READ(3)FM,XXH,WM,K,V,NEC,NLC
DO 210 I=1,10
   READ(3)IFLAG(I)
210 CONTINUE
DO 230 I=1,K
   READ(3)INEG(I),PINEG(I),XN(I)
   DO 220 J=1,V
      READ(3)AM(I,J)
220 CONTINUE
230 CONTINUE
DO 250 J=1,V
   READ(3)CM(J)
250 CONTINUE
IF:IFLAG(5).EQ.1 THEN
   DO 270 I=1,K
      READ(3)CM(I)
270 CONTINUE
   DO 290 J=1,V
      READ(3)VM(J)
290 CONTINUE
   DO 310 J=V+1,20
      VM(J)=
310 CONTINUE
END IF
IFLAG(2)=1
CLOSE(3,STATUS='KEEP')
WRITE(1,'(A)')CHAP(12)
WRITE(1,'(I4/,1X,''INSURE DISK LP2 IS AVAILABLE.'',7(/)))
P A USE
C FIND APPROPRIATE VALUE FOR BIG M
CJ=0.0
BM=0.0
DO 350 J=1,V
   IF(ABS(C(J)).GT. BM) THEN
      BM=ABS(C(J))
   ENDIF
350 CONTINUE
BM=ANINT(BM)*10.0
IF(BM.LT.1.0) THEN
   BM=10.0
ENDIF
RETURN
END
SUBROUTINE NF!LE(N)
C USE: DEPENDENT ON FLAG (K), DISPLAYS BRIEF EXPLANATION OF NEW
C FILE BEING CREATED. SOLICITS VOLUME:FILENAME INPUT BY USER
C FOR CREATION OF NEW FILE AND OPENS THIS FILE ON VOLUME
C SPECIFIED.
C CALLED BY: SUBROUTINE ACNCH
C SUBROUTINE CONVRT
C CALLS : NONE
C VARIABLES:
C USED: N
C MODIFIED: P(11),FN,TN
SUBROUTINE NF!LE(N)
CHARACTER V~.N6P~2,M3FSQPKQ1P1)SBN1,NI
COMMON/P4/VN20),CN20),PN,NN,FN,PIE(12),OBJN
WRITE(1,110)CHAR(12)
110 FORMAT(A)
IF(N .EQ. 1)THEN
TN='VARIABLES'
ELSEIF(N .EQ. 2)THEN
TN='CONSTRAINTS'
ELSE
WRITE(I,'(IX,'A NEW DISK:FILENAME MUST BE CREATED SO''/''THAT T
HE ORIGINAL''/''IS NOT DESTROYED AND MAY BE REUSED.
,$'/'))FN
GO TO 130
ENDIF
WRITE(I,'(IX,'DUE TO THE ADDITION OF ''A11,'', A''/''NEW DISK:FI
LE MUST BE CREATED TO CONTAIN''/''THE NEW MODEL. THIS WILL ALLOW T
HE USER''/''TO REUSE''/''AT A LATER TIME.''$/'))TN,FN
130 WRITE(I,'(IX,'PLEASE ENTER, IN 10 CHARACTERS OR LESS.'/XX,''A DR
VE:FILENAME TO STORE THIS FILE.''$/'))
WRITE(I,'(IX,'NEW MODELS DRIVE:FILENAME = '/XX,$')
READ5,'(AI0)')FN
WRITE(I,'(IX,'ARE CORRECTIONS NEEDED? '/XX,$')
READ5,'(AI)')P(1)
IF(ICHAR(P(1)) .EQ. 89)THEN
GO TO 140
ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
WRITE(I,160)
160 FORMAT(5X,'INVALID ENTRY. PLEASE REENTER'
GO TO 150
ENDIF
WRITE(I'I/''HAS THIS DISK:FILENAME COMBINATION BEEN''/XX,''USED
PREVIOUSLY''? '/XX,''ARE YOU UPDATING A CURRENTLY EXISTING''/XX,''FI
.LE?''$/'))
200 WRITE(I,'161'I/''(Y/N) '/XX,$')
READ5,'(AI)')P(1)
IF(ICHAR(P(1)) .EQ. 89)THEN

OPEN (3, FILE=FN, STATUS='OLD', FORM='UNFORMATTED')
ELSEIF (ICHAR(P(I)) .EQ. 78) THEN
  OPEN (3, FILE=FN, STATUS='NEW', FORM='UNFORMATTED')
ELSE
  WRITE (1, 160)
  GO TO 200
ENDIF
WRITE (1, 110) CHAR(12)
WRITE (1, '19(/),''INSURE THE DISK TO CONTAIN''//15X,A10//10X,
''MODEL IS AVAILABLE.'',/1(/)//FN
PAUSE
RETURN
END
SUBROUTINE PSHED
    WRITE(1, 111), CH(12)
  111 FORMAT(1X, 'PROBLEM', 1X, 'SOLVER', 1X, 'PROGRAM', 1X, 'MODE', 1X, S(12))
    WRITE(1, 112), CH(12)
  112 FORMAT(1X, 'PROGRAM', 1X, 'MODE', 1X, S(12))
    PAUSE
    RETURN
END
SUBROUTINE ASKO(ASK)

CHARACTER VN(4),CH(5),FM(20),MN(3),FN(10),PINEQ(10),P(10)!

INTEGER ART(1),ARTY(20),L(60),1,I,INEQ(20),IFLAG(10),CB(20),
        H(20),N,MIN,BASIC,OPTS,WM
COMMON/P1/NOEC,NEC,IA,INDEXE,INDEXL,KF(6),FSK(6),FSU,
        .F,P
COMMON/P2/NEC,MIN,RIA,IA,INDEXE,INDEXL,KF(6),FSK(6),FSU,
        .F,P
COMMON/P3/DUAL,OUTF,IATB,IBTB,FMT,PPOB
COMMON/P4/VN(20),CN(20),PM,AM,PN,PINEQ(20),OBIN
DIMENSION AD(20,20),B(20,20)

FIN=FN
100 WRITE(1,110)CHAR(12)
110 FORMAT(A)
 IF(IFLAG(10)=DUAL-1)
 WRITE(1,'(B/)','TO PERFORM SENSITIVITY ANALYSIS ON THIS''/''MODEL
 THE INFORMATION OF THE CURRENT''/''TABLEAU MUST BE SAVED TO DISK
 

130 WRITE(1,'("EG YOU WISH TO SAVE THIS FILE TO DISK? ",#)')
 READ(S,'(A1)')P(1)
 IF(IFCHAR(P(1) .EQ. 99)THEN)
 WRITE(1,'(A1)')CHAR(12)
 WRITE(1,'(4/99)','SAVE LP MODEL TO DISK''/''ENTER THE DISK D
 RIVE NUMBER AND FILE''/''NAME YOU WANT THE CURRENT TABLEAU OF''
 THE DISK',
 WRITE(1,'(4/99)','SAVE UNCHANGED''/''ENTER THE DISK DRIVE:FILE
 NAME''/''NAME',
 WRITE(1,'(4/99)','Save exactly as follows''/''THE DRIVE:FILENAME MUST BE 10
 CHARACTERS''/''OR LESS'',''DO NOT USE THE SAME NAME USED
 WHEN THE''/''ORIGINAL MODEL WAS ENTERED'','')
 WRITE(1,'(4/99)','DISK:FILENAME = ",#)')
 READ(S,'(A10)')FN
150 WRITE(1,'(4/99)','ARE CORRECTIONS NEEDED? ",#)')
 READ(S,'(A1)')P(1)
 IF(IFCHAR(P(1) .EQ. 99)THEN)
 GO TO 100
 ELSEIF(IFCHAR(P(1)) .NE. 79)THEN
 WRITE(1,'(4/99)','INVALID ENTRY, PLEASE REENTER'')
 GO TO 150
 ENDF
 WRITE(1,'(11/)','INSURE DISK LP2 IS AVAILABLE. ",(4/99)

363
PAUSE
C TRANSFER FILE OPENED AND FILE NAME WRITTEN
OPEN(3,FILE='LP2:LPDATAW',STATUS='OLD',FORM='UNFORMATTED') WRITE(3)'FN
C TRANSFER FILE CLOSED
CLOSE(3,STATUS='KEEP') WRITE(1,110)CHAR(12)
C USER PROMPTED TO INSERT DISK WHICH SOLVED MODEL IS TO BE SAVED
WRITE(1,'(9,//,2X,'"INSURE THE DISK TO CONTAIN THE FILE"//15X,
.A10//13X,"IS AVAILABLE","7(/)')')FN PAUSE
WRITE(1,110)CHAR(12)
C CURRENT STATUS OF FILE INPUT BY USER
WRITE(1,'(9,//,"HAS THIS DISK:FILENAME COMBINATION BEEN"//12X,
.""USED PREVIOUSLY""/""ARE YOU UPDATING A CURRENTLY EXISTING"/
.17X,""FILE?""/)')')
200 WRITE(1,'(//16X,"(Y/N) "",,1)')
READ(5,'(A1)')P(1)
IF(CHAR(P(1)) .EQ. 89) THEN
OPEN(3,FILE='FN',STATUS='OLD',FORM='UNFORMATTED')
ELSEIF(CHAR(P(1)) .EQ. 79) THEN
OPEN(3,FILE='FN',STATUS='NEW',FORM='UNFORMATTED')
ELSE
WRITE(1,210)
210 FORMAT(//5X,'"INVALID ENTRY, PLEASE REENTER")
60 TO 200
ENDIF
C SOLVED MODEL WRITTEN TO DISK
WRITE(3)PN,NNXN,K,V,IFLAG(5)
WRITE(1,110)CHAR(12)
WRITE(1,'(9,//,5X,"INSURE THE DISK CONTAINING THE"//15X,A10
.//10X,""MODEL IS AVAILABLE","7(/)')')FN
PAUSE
C ORIGINAL MODEL FILE OPENED TO READ ORIGINAL PARAMETERS
OPEN(4,FILE=FN0,STATUS='OLD',FORM='UNFORMATTED')
READ(4)PM,NNXN,MM,K,V,NEC,NGC,NLC
DO 220 I=1,10
READ(4)IFLAG(I)
220 CONTINUE
DO 240 I=1,K
READ(4)PINEB(I),PINEQ(I),B:I
DO 230 J=1,V
READ(4)AR(I,J)
230 CONTINUE
240 CONTINUE
DO 250 J=1,V
READ(4)C3(J)
250 CONTINUE
CLOSE(4,STATUS='KEEP') WRITE(1,110)CHAR(12)
WRITE(1,'(9,//,"INSURE THE DISK TO CONTAIN THE FILE FOR"//15X,
.A10//13X,"IS AVAILABLE","7(/)')')FN
PAUSE
DO 270 I=1,K
WRITE(3)INEB(I),B(I)
DO 260 J=1,V
WRITE(3)ADI(I,J)
260 CONTINUE
270 CONTINUE
C SOLVED MODEL AND ORIGINAL PARAMETERS WRITTEN TO DISK
DO 275 J=1,V
WRITE(3)CO(J)
275 CONTINUE
WRITE(3)IFLAG,VT
DO 290 I=1,K
WRITE(3)XB(I),CB(I)
DO 290 J=1,VT
WRITE(3)A(I,J)
290 CONTINUE
DO 300 J=1,VT
WRITE(3)I
300 CONTINUE
WRITE(3)IFLAG,VT
DO 310 I=1,K
WRITE(3)CN(I)
310 CONTINUE
DO 320 J=1,VT
WRITE(3)VM(J)
320 CONTINUE
WRITE(3)OBJ
ENDIF
CLOSE(3,STATUS="KEEP")
WRITE(1,110)CHAR(12)
WRITE(1,"(1I-./),X,''INSURE DISK LP2 IS AVAILABLE.'',7(I)
,1")
PAUSE
ELSEIF CHAR(P(1)) .NE. 78)THEN
WRITE(1,210)
GO TO 130
ENDIF
390 WRITE(1,110)CHAR(12)
WRITE(1,"(1I-./),X,"WILL YOU LIKE TO STUDY ANOTHER MODEL''/4X,"
.WHICH HAS BEEN SAVED TO DISK? ''/4X,"
READ(5,"(A1)"P(1)
IF(CHAR(P(1)) .EQ. 89)THEN
400 WRITE(1,110)CHAR(12)
WRITE(1,"(9X,2X,"ENTER DISK DRIVE NUMBER AND FILENAME''/4X,"
.WHICH THE MODEL IS SAVED UNDER.'')")
WRITE(1,"(6X,"MODEL TO STUDY = '',$)
READ(5,"(A10)"FN
450 WRITE(1,"(/7X,"ARE CORRECTIONS NEEDED? ''/4X,"
READ(5,"(A1)"P(1)
IF(ICHAR(P(1)) .EQ. 89) THEN
  GO TO 400
ELSEIF(ICHAR(P(1)) .NE. 78) THEN
  WRITE(1, '(/S1,"INVALID ENTRY, PLEASE REENTER")')
  GO TO 450
ENDIF
WRITE(1, 110) ICHAR(12)
WRITE(1, '(I11I.I14,"INSURE DISK LP1 IS AVAILABLE.",7(/))')
PAUSE
TRANSFER FILE OPENED AND NEW MODEL FILE NAME WRITTEN
OPEN(5, FILE='LP1:LPDATA', STATUS='OLD', FORM='UNFORMATTED')
WRITE(5) FM
CLOSE(3, STATUS='KEEP')
ASK=1
RETURN
ELSEIF(ICHAR(P(1)) .NE. 79) THEN
  WRITE(1, '(/S1,"INVALID ENTRY, PLEASE REENTER")')
  GO TO 590
ENDIF
WRITE(1, 110) ICHAR(12)
WRITE(1, '(/I1(/I4X,"INSURE DISK LP2 IS AVAILABLE.",7(/))')
PAUSE
WRITE(1, 110) ICHAR(12)
WRITE(1, '(/B/1,I1,"TO ENTER THE LP DATABASE MODULE:"/17X,"TYPE
  "/194,""X":11X,"LP1:SYSTEM.STARTUP.",3(/))')
STOP
RETURN
END
SUBROUTINE BISM
INTEGER ARAV,BASIC,PY,OPTS,V,VT, CB, DUAL, DUTP, FMT, PROBT
COMMON/P1/A(26,60),ARAV(20),C(60),I,INEQ(26),IFLAG(10),CE(20),
/31(20),K,V,VT,MINN,BASIC,OPTS,BM
COMMON/P2/NEC,NLC,IA,INDE1,E,INDE2,INDE3,IFAKS,KFS,KFSA,KFSU,
,PY,PR
COMMON/P3/DUAL,DUTP,ITAB,IBTAB,IFTAB,FMT,PROBT
IA=IA-1
KFSU=K+M6C
DO 300 J=1,KFSU
   SUM=0.0
   DO 200 I=1,IA
      SUM=SUM+ARTV(I,J)
200   CONTINUE
   C(J)=(ARV(J)-(SUM))/(SUM)
300   CONTINUE
   DO 400 J=KFA,VT
      C(J)=0.0
400   CONTINUE
   SUM=0.0
   DO 500 I=1,IA
      SUM=SUM+81*ARTV(I))
500   CONTINUE
   L=L-(SUM/SUM)
RETURN
END
SUBROUTINE INDEX

CHARACTER VNAB,CNAB,FIA,FIC,FIC10,PIPE01,F(10),Q10,OBJN10
INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,SB
COMMON/R2(A20,60),ARTV(20),C(I60),I,IMEB(20),IFLAG(10),CB(20),
,XB(20),K,V,VT,MN,M,BASIC,OPTS,SM
COMMON/P2/MEC,NLC,INCE,INDEXE,INDEX6,INDEXL,KFA,KFS,KFSU,
,KK,PP
COMMON/P1/IA(20),CN(20),PM,MM,FN,PINE(20),OBJN
IA=1
KFSA=V+1
VT=V+(2*NEC)+NLC+MEC
KFS=V+MEC+1
KFA=KFS+NLC
KFSU=V+MEC
INDEXE=V+1
INDEXL=V+MEC+1
INDEXE=V+MEC+NLC+1
DO 200 I=IA,K
IF(INEB(I).EQ.0)THEN
  C SLACK VARIABLE ADDED TO CONSTRAINT
  CB(I)=INDEXE
  ARTV(IA)=I
  A(I,INDEXE)=1.
  INDEXE=INDEXE+1
ELSEIF(INED(I).EQ.1)THEN
  C SURPLUS AND ARTIFICIAL VARIABLE ADDED TO CONSTRAINT
  CB(I)=INDEXE
  ARTV(IA)=I
  A(I,INDEXE)=1.
  INDEXE=INDEXE+1
  A(I,INDEXE)=-1.
  INDEXE=INDEXE+1
ELSE
  C ARTIFICIAL VARIABLE ADDED TO CONSTRAINT
  ARTV(IA)=I
  IA=IA+1
  CB(I)=INDEXE
  A(I,INDEXE)=1.
  INDEXE=INDEXE+1
ENDIF
200 CONTINUE
IF(IFLAG5).EQ.1)THEN
DO 210 J=KPSA,KPS-1
   VM(J)='SURPLS'
210 CONTINUE
DO 220 J=KFS,KFS-1
   VM(J)='SLACK'
220 CONTINUE
DO 230 J=KFA,KFA-1
   VM(J)='ARTIF'
230 CONTINUE
ENDIF
RETURN
END
SUBROUTINE MODIFP

CHARACTER VNI6,CNS6,PNS2(OMMS3,FNS10,PINEQ10),OPTS,DM
INTEGER ARTV,BASIC,PKIPR,OPTSV,YT,CB
COMMON/P1/IA(20),CNS(200),I,INEQ(20),IFLAG(10),CB(20),
        X(20),V,VT,MXMN,DAS,IC.OPTS,DM
COMMON/P2/NE++,NLC,IA,INDEXE.

IF(Z(I)=0.0)THEN
   NLC=NLC+1
   I=INEQ(I)
ELSE IF(I=.EQ.1)THEN
   NLC=NLC-1
   I=INEQ(I)
END IF
END:
CONTINUE
RETURN
370
SUBROUTINE MODIFO

USE: TRANSFORMS OBJECTIVE FUNCTION FROM MAX OR MIN Z=x FORM TO MAX
2=x Form. Transforms all constraints with greater-than or
equal inequality to less-than or equal inequality.

CALLED BY: PROGRAM PROBS

CALLS : NONE

VARIABLES:

USED: X, MM, N, V

MODIFIED: A(,), C(,), IME(), NBC, NLC, MM, PINEQ(), XB(,)

SUBROUTINE MODIFO
CHARACTER VN#6, CN#6, M#20, MM#3, FM#10, PINEQ#1, P(10)#1, OB#N10
INTEGER ARTV, BASIC, PK, OPTS, V, VT, CB, DUAL, OUTP, FNT, PROBT
COMMON/P1/ARTV(20), C(60), IME(), IFLAG(10), CB(20),
X(20), K, V, VT, MM, BASIC, OPTS, BM
COMMON/P2/NEC, NBC, NLC, IA, INDEXE, INDEXS, INDEXL, KFA, KFSA, KFSU,
PK, PR
COMMON/P3/VM(20), C(20), FM, FN, PINEQ(20), OBJS
IF(MMN .EQ. 1) THEN

PROBLEM STATED AS MAXIMIZATION PROBLEM

DO 120 J=1, V
C(J)=C(J)
120 CONTINUE
ELSE
MM=’MIN’
ENDIF
DO 150 I=1, K
IF(IME() .EQ. 1) THEN
XB()=XB()
DO 140 J=1, V
A(I,J)=A(I,J)
140 CONTINUE
C COUNT OF INEQUALITIES UPDATED DUE TO MULY BY -1
PINEQ()=’
IME()=0
NBC=NBC-1
NLC=NLC+1
ENDIF
150 CONTINUE
RETURN
END

372
SUBROUTINE TDISPL

USE: Determines the number of 80 column display widths required to display a table. Displays table to user selected output device. Also solicits input from user as to displaying the basic variables value after each displayed tableau. If requested, displays the basic values on selected output device with objective function value also displayed.

CALLED BY: PROGRAM PROBS

CALLS: SUBROUTINE CHECK2(P,N,M,INVAL,INew)

VARIABLES:

USED: A(I,I), BASIC(I), CB(I), CM(I), CN(I), FMT, IFLAG5, INEW, INVAL,
K, PINEQ(I), PN, VM(I), V(i), XB(I), Z

MODIFIED: P(I)

*USES UCHECK2 IN UNIT37.CODE OVERLAY

SUBROUTINE TDISPL

CHARACTER VN&6, CM&6, PN&20, MN51, FN&10, PINEQ&1, P&10&1, OBJN&10
INTEGER ARTV, BASIC(P,PK, PR, OPTS, V, VT, CB, DUAL, DTP, OPTS, PROBT)
COMMON/P1/A(20,60), ARTV(20), C(60), Z, INEW(20), IFLAG5(10), CB(20),
.VB(20), X, V, VM, BASIC, OPTS, EM
COMMON/P2/NEC, NBC, NLC, .A, IUNE, INDEX, INDEX, IFA, KFS, KFSI, KFU,
.KF, FR
COMMON/P3/DUAL, DTP, ITAB, ITAB, ITAB, OPTS, PROBT
COMMON/P4/VN(20), VM(20), PN, MM, FM, PINEQ(20), OBJN

110 FORMAT(A)

C NUMBER OF 80 COLUMN DISPLAYS REQUIRED DETERMINED

T=VT/5*T

DO 470 N=1, T

IF(IFLAG5 .EQ. 1) THEN

C VARIABLES NAMES PRINTED AS COLUMN HEADERS

WRITE(2, '('/1X, 'I')')

DO 270 J=(N&5)-4, N&5

IF(J .LT. VT) THEN

GO TO 270

ENDIF

WRITE(2,260) VM(J)

260 FORMAT(5X, A6, 2X, I4)

270 CONTINUE

WRITE(2, '('/3X, '"')')

ENDIF

WRITE(2, '('/4X, 'I')')

DO 290 J=(N&5)-4, N&5

IF(J .LT. VT) THEN

GO TO 290

ENDIF

WRITE(2,280) J

280 FORMAT(5X, '"", ', I2, '"'), 3X, I4)

290 CONTINUE
C IF LAST 80 COLUMN DISPLAY, DISPLAY RHS
IF(T .EQ. 1 .OR. N .EQ. T) THEN
   WRITE(2,300)
300 FORMAT(6X,'RHS')
ELSE
   WRITE(2,'(*** ***)')
ENDIF
WRITE(2,'("OBJ FUNCTION",1X,0)')
DO 320 J=MR5-4,MR5
   IF(J .GT. VI) THEN
      GO TO 320
   ENDIF
   IF(FMT .EQ. 0) THEN
      WRITE(2,'(IPE12.5,1X,0))C(J)
   ELSE
      WRITE(2,'(F12.5,1X,0))C(J)
   ENDIF
320 CONTINUE
IF(T .EQ. 1 .OR. N .EQ. T) THEN
   IF(FMT .EQ. 0) THEN
      WRITE(2,'("CN NAME VAR",2X,6S(" ")))C
   ELSE
      WRITE(2,'("CN NAME VAR",2X,6S(" ")))C
   ENDIF
   ELSE
      WRITE(2,'(*** ***)')
   ENDIF
WRITE(2,'("CONSTRAINT NUMBER, NAME, BASIC VARIABLE, COEFFICIENTS")
C INEQUALITY, AND RHS DISPLAYED')
DO 490 L=1,K
   IF(L .GT. VI) THEN
      GO TO 490
   ENDIF
   IF(FLAG5 .EQ. 1) THEN
      WRITE(2,'(12.1X,A6,0)L,CNL)
   ELSE
      WRITE(2,'(12.7X,0)L')
   ENDIF
   WRITE(2,'(11,12,1X,0)TCB:L)
490 CONTINUE
DO 370 J=MR5-4,MR5
   IF(J .GT. VI) THEN
      GO TO 370
   ENDIF
   IF(FMT .EQ. 0) THEN
      WRITE(2,'(IPE12.5,1X,0)A(L,J)
   ELSE
      WRITE(2,'(F12.5,1X,0)A(L,J)
   ENDIF
370 CONTINUE
IF(T .EQ. 1 .OR. N .EQ. T) THEN
   IF(FMT .EQ. 0) THEN
      374
WRITE(2,'(A1,I1,PE12.5)')PINEQ(L),IB(L)
ELSE
WRITE(2,'(A1,I1,F12.5)')PINEG(L),IB(L)
ENDIF
ELSE
WRITE(2,'(" "")')
ENDIF
400 CONTINUE
IF(OUTP.EQ.1)THEN
PAUSE
WRITE(2,110)CHAR(12)
ELSE
WRITE(2,'(2/(I1)"')
ENDIF
470 CONTINUE
CLOSE(2)
C USER SELECTS DISPLAY LOCATION OF BASIC VARIABLES AND VALUES
490 WRITE(1,110)CHAR(12)
WRITE(1,'(/A/I,"WILL YOU LIKE THE BASIC SOLUTION VALUES"/(A,X,
"DISPLAYED?")')
WRITE(1,'(/9X,"1. DISPLAY ON SCREEN"/9X,"2. DISPLAY ON PRINTER
"/9X,"3. DO NOT DISPLAY")')
500 WRITE(1,'(/A/I,"WHICH OPTION? ",A)')
READ(5,'(A1)')P(1)
CALL CHECKIF,1,3,INVAL,INE)
IF(INVAL .EQ. 1)THEN
WRITE(1,530)
530 FORMAT('X','INVALID ENTRY, PLEASE REENTER')
GO TO 500
ENDIF
IF(INEN .EQ. 1)THEN
OPEN(2,FILE='CONSOLE:')
ELSEIF(INEN .EQ. 2)THEN
OPEN(2,FILE='PRINTER:')
ELSE
RETURN
ENDIF
WRITE(1,110)CHAR(12)
WRITE(2,'(10X,A20/I)BASIC
IF(IFLAG(5) .EQ. 1)THEN
20 550 I=1,K
IF(FMT .EQ. 0)THEN
WRITE2,','(5X,A6,='I"',I2,=",",1PE12.5)';VNCB(!)),
.CE(!),IB(I)
ELSE
WRITE2,','(5X,A6,='I"',I2,=",",F12.5)';VNCB(!)),
.CE(!),IB(I)
ENDIF
550 CONTINUE
IF(FMT .EQ. 0)THEN
WRITE12,'(/18X,"Z=",1PE12.5)"'
WRITE(2,'(1X,','Z= ''\(12.5\))Z
ENDIF
ELSE
DO 580 I=1,K
IF(FMT .EQ. 0) THEN
  WRITE(2,'(1X,','X('','12,'') = ''\(12.5\))Z
  ELSE
  WRITE(2,'(1X,','X('','12,'') = ''\(12.5\))Z
  ENDIF
580 CONTINUE
ENDIF
IF(FMT .EQ. 0) THEN
  WRITE(2,'(14X,','Z = ''\(12.5\))Z
  ELSE
  WRITE(2,'(14X,','Z = ''\(12.5\))Z
  ENDIF
ENDIF
IF(FMT .EQ. 2) THEN
  WRITE(2,'(6(/))')
ELSE
  PAUSE
ENDIF
RETURN
END
C ROUTINE CHECK2(ED,HVAL,INVAL,INEW)
C USE: SEE MODULE 1, UNIT17, SUBROUTINE CHECK2(ED,HVAL,INVAL,INEW)
C CALLED BY: SUBROUTINE OPTN
C CALLED BY: SUBROUTINE TDISPL
C CALLED BY: NONE
C VARIABLES: SEE NODULE 1, UNIT17, SUBROUTINE CHECK2(ED,HVAL,INVAL,INEW)

SUBROUTINE CHECK2(ED,HVAL,INVAL,INEW)
CHARACTER ALLON,ELON
DIMENSION E(10),ALLON(II)
INTEGER D,HVAL
DATA ALLON/'1','2','3','4','5','6','7','8','9','0',' '/
INEW=0
INVAL=0
DO 300 I=1,10
   DO 206 J=1,10
C CHECKS FIRST FOR BLANK CHARACTERS
   IF(E(I) .EQ. ALLOW(I)) THEN
      GO TO 300
   ELSEIF(E(I) .EQ. ALLOW(I)) THEN
      INEW=INEW+1+(ICHAR(E(I))-48)
      GO TO 300
   ELSEIF(J .EQ. 10) THEN
      INVAL=1
      INEW=0
      RETURN
   ENDIF
200 CONTINUE
300 CONTINUE
IF(INEW .EQ. 0 .OR. INEW .GT. HVAL) THEN
   INVAL=1
   INEW=0
   RETURN
ENDIF
RETURN
END

377
MODULE 4 UNIT40

UNIT USES: UNITS 41 THROUGH 45, 47, 48

PROGRAM MAINSA

USES THIS IS THE MAIN PROGRAM IN THE SENSITIVITY ANALYSIS MODULE.

IT IS USED TO CALL OTHER SUBROUTINES IN RESPONSE TO USER

INPUT AND TO CONTROL THE OVERLAY PROCESS WHICH ALLOWS LARGER

PROGRAMS

CALLED BY: NONE

CALLS : SUBROUTINE PETRIV

SUBROUTINE CMHMS

SUBROUTINE COEFFR

SUBROUTINE MULCHS

SUBROUTINE SELECT

SUBROUTINE ADDCON

VARIABLES:

USES : SELSUB

MODIFIES : SELOUT, IFLAG(9), IFLAG(2)

USES CHECK2 IN UNIT47.CODE OVERLAY

USES URETIV IN UNIT46.CODE OVERLAY

USES UCMHMS IN UNIT41.CODE OVERLAY

USES UCOLUMN IN UNIT42.CODE OVERLAY

USES UCOLUMN IN UNIT43.CODE OVERLAY

USES UADDON IN UNIT44.CODE OVERLAY

USES JSOLVE IN UNIT4 CODE OVERLAY

PROGRAM MAINSA

INTEGER K, V, VT, IFLAG, C(I), INDEXC, INDEXE, INDEE, NEG
REAL AO, A|, BX, OF, OF, OF, CF, I, SM
CHARACTER SELSUB, SELOUT, SELSOL, FN*1O
COMMON/P1/OPTSK,F,A,PK,PR
COMMON/GO/SELOUT, FN
COMMON/TN/VT, INDEXC, INDEXE, INDEE, NEG, MLC, NEC, NEG(2O), NHPN, BM
COMMON/THREE/INFP

INCLUDE COMVAR

OPEN(1,FILE="CONSOLE")
OPEN(5,FILE="CONSOLE")

4000 WRITE(1,'(A1,/') CHAR(12)

THE USER SELECTS THE DESIRED TYPE OF SENSITIVITY ANALYSIS.

WRITE(1,'(///,1X,"PLEASE SELECT ONE ITEM BY NUMBER",///
..3X,"1) RANGE LIMITS-----RIGHT-HAND-SIDE",/ 
..3X," AND ASSOCIATED Z VALUES "",/ 
..3X,"2) RANGE LIMITS-----A(I,J) & C(J)",/)
WRITE(1,'(3X,"3) CHECK OPTIMALITY FOR MULTIPLE",/ 
..3X,"4) ADD A VARIABLE OR A CONSTRAINT",/ 

4378
.,3I, 'S') EXIT PROGRAM'""/')
READ(5,'(AI)') SELSEB
IF(SELSUB .EQ. '5') THEN
GOTO 4030
ENDIF
C CHECK FOR VALID INPUT
IF ((SELSUB .NE. '1') .AND. (SELSUB .NE. '2') .AND. (SELSUB .NE. '3') .AND. (SELSUB .NE. '4')) THEN
WRITE(1,'(AI)') CHAR(12)
WRITE(1,'(/,2I, 'INVALID RESPONSE',/))
GOTO 4010
ENDIF
C CLEAR THE SCREEN AND SELECT OUTPUT.
WRITE(1,'(AI)) CHAR(12)
WRITE(1,'(4AI,4X,''DO YOU WANT THE OUTPUT TO GO TO:'''//
.,4AI, ''SCREEN''//,4AI, ''PRINTER''//)
WRITE(1,'(4AI,4X,''OR''//
.,4AI, ''BOTH''//
.,4AI, ''SELECT S, P, OR B''))
READ(5,'(AI)') SELOUT
IF ((SELOUT .NE. 'P') .AND. (SELOUT .NE. 'S') .AND. (SELOUT .NE. 'B')) THEN
WRITE(1,'(AI)) CHAR(12)
WRITE(1,'(/,2I, ''INVALID RESPONSE''))
GOTO 4020
ENDIF
IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B')THEN
OPEN(5,FILE='PRINTER:')
ENDIF
C THE MAIN PART OF THE PROGRAM. SUBROUTINE RETRIV READS ALL
C NECESSARY DATA FROM A DISK FILE AND THEN THE SUBROUTINE IS CALLED
C TO COMPLETE THE SENSITIVITY ANALYSIS. FOLLOWING THE ANALYSIS,
C THE USER IS RETURNED TO THE MAIN MENU.
CALL RETRIV
IF (SELSUB .EQ. '1') THEN
CALL COMHRS
GOTO 4000
ELSEIF (SELSUB .EQ. '2') THEN
CALL COEFFR
GOTO 4000
ELSEIF (SELSUB .EQ. '3') THEN
CALL MLCHG
WRITE(1,'(AI)') CHAR(12)
IF(SELOUT.EQ.'S'.OR.SELOUT.EQ.'B')THEN
IF(IFLAB(9).EQ.1)THEN
WRITE(1,'(''THE ADDED CHANGES HAVE MADE THE PROBLEM''))
WRITE(1,'(''ILL-CONDITIONED, RETURNING TO MAIN MENU''))
PAUSE
ENDIF
ENDIF
IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B')THEN

379
IF(FLAG(9).EQ.1) THEN
    WRITE(6,'(\"THE ADDED CHANGES HAVE MADE THE PROBLEM\")')
    WRITE(6,'(\"ILL-CONDITIONED, RETURNING TO MAIN MENU\")')
    WRITE(6,'(A1)')CHAR(12)
ENDIF
ENDIF
IF(FLAG(9).EQ.1) THEN
    IFLAG(9)=0
    GOTO 4000
ENDIF
IF(FLAG(3).EQ.1) THEN
    GOTO 4000
ENDIF
CALL SELECT
3010 4000
ELSEIF (SELSUB .EQ. \"A\") THEN
    IFLAG(2)=0
    CALL ADDCON
    IF(FLAG(3).EQ.1) THEN
        IF(SELOUT.EQ.\"S\" OR SELOUT.EQ.\"B\") THEN
            WRITE(1,'((S(1),5X,\"******** NOT FEASIBLE ********\")')
            PAUSE
        ENDIF
        IF(SELOUT.EQ.\"P\" OR SELOUT.EQ.\"B\") THEN
            WRITE(6,'((S(1),5X,\"******** NOT FEASIBLE ********\")')
            WRITE(6,'(A1)')CHAR(12)
        ENDIF
    GOTO 4000
ENDIF
WRITE(1,'(A1)')CHAR(12)
IF(FLAG(2).EQ.0) THEN
    CALL SELECT
ENDIF
GOTO 4000
ENDIF
4030  WRITE(1,'(A1)')CHAR(12)
    WRITE(1,'((B/1),1X,\"TO ENTER THE LP DATABASE MODULE: \"/17X,
    \"TYPE: \"/193,\"X"/11X,\"LP1:SYSTEM.STARTUP.\",/11X)
    PAUSE
    STOP
END
C
C MODULE 4 UNIT40
C UNIT USES: NONE
C
C SUBROUTINE SELECT
C USE: THIS SUBROUTINE CALLS SUBROUTINE SOLVE IF THE USER DESIRES A
C NEW FINAL TABLEAU.
C
C CALLED BY: PROGRAM MAINSA
C CALLS : SOLVE
C
C VARIABLES:
C USES : SELSOL
C MODIFIES : NONE
C
C SUBROUTINE SELECT
CHARACTER SELSOL
4090 WRITE(1,'(S(5),5X,"DO YOU WISH TO SOLVE THIS TABLEAU"(1)')
   WRITE(1,'(/,10X,"SELECT "Y" OR "N"")')
   READ(5,'(A)') SELSOL
   WRITE(1,'(A)') CHAR(12)
   IF (SELSOL.EQ. 'Y') THEN
      CALL SOLVE
   ELSEIF(SELSOL.NE. 'N') THEN
      WRITE(1,'(/,10X,"IMPROPER RESPONSE")')
      6070 4090
   ENDIF
RETURN
END
SUBROUTINE COMRHS
INTEGER VT,V,K,J,CDL, IF, INDEXB, INDEXI, INDEXE, INDEXL, N, IN, MAX, MYMIN, BM
REAL RMAX(20), RMIN(20), RSCH(20, 20), LWBD(20, 20), UPBD(20, 20),
     RSLIM(20), RSSLIM(20), ZLP, ZLS, ZUS, ZUP, TEMP, BM, SELOUT,
CHARACTER SELOUT, FN, FT
COMMON/ONE/INDEXB, INDEXI, INDEXE, N, IN, MAX, MYMIN, BM
COMMON/TWO/VT, INDEXL, INDEXE, N, IN, MAX, MYMIN, BM
*INCLUDE CONVAR
C RIGHT-HAND-SIDE RANGING IS DONE FOR EACH CONSTRAINT.
DO 4120 CONSTR = 1, K
C THE COLUMN OF B-INVABSE ASSOCIATED WITH THE CURRENT CONSTRAINT IS
C DETERMINED.
   COL=CONSTR+INDEXL-1
   RMAX(CONSTR) = 1E12
   RMIN(CONSTR) = -1E12
C DETERMINE RESOURCE LIMITS
C THE MINIMUM POSITIVE AND MAXIMUM NEGATIVE VALUES WHICH WILL
C CAUSE A DEGENERATIVE CONDITION ARE DETERMINED.
DO 4140 L = 1, K
   IF (ABS(AF(L, COL)) > 0.0001) THEN
      IF (ABS(BF(L)) > 0.0001) THEN
         RSCH(L, CONSTR) = BF(L)/AF(L, COL)
      ELSE
         RMIN(CONSTR) = RMIN(CONSTR) + RSCH(L, CONSTR)
      ENDIF
   ELSE
      RMAX(CONSTR) = RMAX(CONSTR) + RSCH(L, CONSTR)
   ENDIF
ELSEIF (CD(L) > 0.0) THEN
   RMAX(CONSTR) = 0
   RMIN(CONSTR) = 0
ENDIF
C
ELSEIF(AF(L,COL) .LT. GITHEN ) THEN
  RMAX(CONSTR)=0
ELSE
  RMIN(CONSTR)=0
ENDIF
ENDIF

CONTINUE
RSULIN(CONSTR) = BO(CONSTR) + RMAX(CONSTR)
RSLIM(CONSTR) = BO(CONSTR) + RMIN(CONSTR)

IF THE CONSTRAINT HAS BEEN MULTIPLIED BY MINUS ONE, THE
RESOURCE UPPER LIMIT (RSULIN) AND THE RESOURCE LOWER LIMIT
(RSLIM) AS WELL AS THE ORIGINAL RIGHT-HAND SIDE ARE REVERSED.
IF (NEG(CONSTR) .EQ. 1) THEN
  TEMP=-RSULIN(CONSTR)
  RSULIN(CONSTR)=-RSLIM(CONSTR)
  RSLIM(CONSTR)=TEMP
  BO(CONSTR)=-BO(CONSTR)
ENDIF

THE UPPER BOUND (UPBD) AND LOWER BOUND (LWBD) FOR EACH RHS IN
THE FINAL SOLUTION ARE OBTAINED.

DO 4150 L = 1,K
  UPBD(L,CONSTR) = AF(L,COL) * RMAX(CONSTR) + AF(L)
  LWBD(L,CONSTR) = AF(L,COL) * RMIN(CONSTR) + AF(L)
  IF (NEG(CONSTR) .EQ. 1) THEN
    TEMP=UPBD(L,CONSTR)
    UPBD(L,CONSTR)=LWBD(L,CONSTR)
    LWBD(L,CONSTR)=TEMP
  ENDIF

CONTINUE

705=0
2LS=0
ZUP=0
ZLP=0

THE RESULTS ARE PRINTED FOR EACH BOUND DEPENDING ON THE
CONDITIONS.

DO 4160 L = 1,K
  IF(SELOUT .EQ. 'S' .OR. SELOUT .EQ. 'B') THEN
    IF(L .LE. 1 .OR. L .LE. 8 .OR. L .EQ. 15) THEN
      WRITE(1,'(A1)') CHAR(12)
      WRITE(1,'(/,A40)')
      WRITE(1,'(6X,"RIGHT HAND SIDE RANGE LIMITS")')
      WRITE(1,'(12X,"CONSTRAINT ",L2,/)')CONSTR
      WRITE(1,'("ORIGINAL RIGHT HAND SIDE = ",F12.5)')
      BO(CONSTR)
      IF (ABS(RSLIM(CONSTR)).GT.1E6) THEN
        WRITE(1,'(13X,"LOWER BOUND = NO LIMIT")')
      ELSE
        WRITE(1,'(13X,"LOWER BOUND = ",F12.5)')
        RSLIM(CONSTR)
      ENDIF
    ELSE
      IF (ABS(RSULIN(CONSTR)).GT.1E6) THEN
        WRITE(1,'(13X,"UPPER BOUND = NO LIMIT")')
      ENDIF
ELSE
    WRITE(1,'(1X,'"UPPER BOUND = ",F12.5)')
    RSLIN(CONSTR)
ENDIF
WRITE(1,'(40(" AT THE UPPER "')
WRITE(1,'"BOUND")')
ENDIF
IF(CB(L) .LT.10) THEN
    IF(ABS(UPBD(L,CONST)) .GE.1E6 .AND. ABS(LMBD(L,CONST)) .GE.1E6) THEN
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ELSEIF(ABS(UPBD(L,CONST)) .GE.1E6 .AND. ABS(LMBD(L,CONST)) .GE.1E6) THEN
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ELSE
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ENDIF
ELSE
    WRITE(1,'(1X,"NO LIMIT")')
    IF(CB(L) .LE.V) THEN
        ZLS=ZLS+10EB
        ZUS=ZUS+10EB
    ENDIF
ELSE
    IF(ABS(UPBD(L,CONST)) .GE.1E6 .AND. ABS(LMBD(L,CONST)) .GE.1E6) THEN
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ELSEIF(ABS(UPBD(L,CONST)) .GE.1E6 .AND. ABS(LMBD(L,CONST)) .GE.1E6) THEN
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ELSE
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ENDIF
ELSE
    IF(ABS(UPBD(L,CONST)) .GE.1E6 .AND. ABS(LMBD(L,CONST)) .GE.1E6) THEN
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ELSEIF(ABS(UPBD(L,CONST)) .GE.1E6 .AND. ABS(LMBD(L,CONST)) .GE.1E6) THEN
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ELSE
        WRITE(1,'(1X,"NO LIMIT")')
        IF(CB(L) .LE.V) THEN
            ZLS=ZLS+10EB
            ZUS=ZUS+10EB
        ENDIF
    ENDIF
ENDIF
,12,'') = NO LIMIT')*CB(L),LMBD(L,CONST),
CB(L)
IF(CB(L),LE.0) THEN
ZLS=ZLS+CB(CB(L)),LMBD(L,CONST)
ZUS=ZUS+10EB
ENDIF
ELSEIF(ABS(LMBD(L,CONST)).GE.1E6) THEN
WRITE(1,('''X''',I2,'') = NO LIMIT '',''X','')
IF(CB(L),LE.0) THEN
ZLS=-10EB
ZUS=ZUS+CB(CB(L)),LMBD(L,CONST)
ENDIF
ELSE
WRITE(1,('''X''',I2,'') = '','F12.5'')*CB(L),LMBD(L,CONST),CB(L),
LMBD(L,CONST)
ENDIF
END IF
IF(L.EQ.7.OR.L.EQ.14) THEN
PAUSE
ENDIF
IF(CONSTR.EQ.1.AND. L.EQ.1) THEN
LINES=O
ENDIF
IF(SELOUT.EQ. 'P'.OR. SELOUT.EQ. 'B') THEN
IF(L.EQ.1.) THEN
IF(CONSTR.'NE. 1) THEN
IF(LINES+K ,GT. 44) THEN
WRITE(6,'(AI)') CHAR(12)
LINES=O
ENDIF
ENDIF
WRITE(6,'(I,15X,A10)')FM
WRITE(6,'(I,5F12.5)')
WRITE(6,'(6X,'''RIGHT HAND SIDE RANGE LIMITS''')
WRITE(6,'(12X,'''CONSTRAINT = ''',I2,/)'')CONSTR
WRITE(6,'(12X,'''ORIGINAL RIGHT HAND SIDE = ''',F12.5)')
RSLIM(CONSTR)
IF(ABS(RSLIM(CONSTR)).GE.1E6) THEN
WRITE(6,'(I,13I,'''LOWER BOUND = NO LIMIT''))
ELSE
WRITE(6,'(I,13I,'''LOWER BOUND = ''',F12.5)')
RSLIM(CONSTR)
ENDIF
IF(ABS(RSLIM(CONSTR)).GE.1E6) THEN
WRITE(6,'(I,13I,'''UPPER BOUND = NO LIMIT''))
385
ELSE
WRITE(6, '13X,'"UPPER BOUND = ",F12.5')
ASULIN(CONSTR)
ENDIF
WRITE(6, '50('"S"'),/')
WRITE(6, "12X,'"AT THE LOWER BOUND'",12X,'"AT THE UPPER '" 
"BOUND'",')
LINES=LINES+15
ENDIF
IF(CNIL) .LT.10)THEN
IF(ABS(UPBD(L,CONSTR)) .GE.1E6. AND. ABS(LWBD(L,CONSTR)) .GE.1E6)THEN
WRITE(6, '13X,'"NO LIMIT'",11X,'"X" 
,11,) = ",F12.5,'"X")CB(L),LWBD(L,CONSTR), 
CB(L)
IF(CB(L) .LE. Y)THEN
ZLP=ZUP+10E8
ZUP=ZLP+10E8
ENDIF
ELSEIF(ABS(UPBD(L,CONSTR)) .GE.1E6)THEN
WRITE(6, '13X,'"NO LIMIT'",11X,'"X" 
,11,) = ",F12.5,'"X")CB(L),UPBD(L,CONSTR)
IF(CB(L) .LE. Y)THEN
ZLP=ZUP+10E8
ZUP=ZLP+10E8
ENDIF
ELSEIF(ABS(LWBD(L,CONSTR)) .GE.1E6)THEN
WRITE(6, '13X,'"NO LIMIT'",11X,'"X" 
,11,) = ",F12.5,'"X")CB(L),LWBD(L,CONSTR)
IF(CB(L) .LE. Y)THEN
ZLP=ZLP+10E8
ZUP=ZLP+10E8
ENDIF
ELSE
WRITE(6, '13X,'"NO LIMIT'",11X,'"X" 
,11,) = ",F12.5,'"X")CB(L),UPBD(L,CONSTR)
IF(CB(L) .LE. Y)THEN
ZLP=ZLP+10E8
ZUP=ZLP+10E8
ENDIF
ENDIF
ELSEIF(ABS(UPBD(L,CONSTR)) .GE.1E6. AND. ABS(LWBD(L,CONSTR)) .GE.1E6)THEN
WRITE(6, '13X,'"NO LIMIT'",11X,'"X" 
,11,) = ",F12.5,'"X")CB(L),LWBD(L,CONSTR),CB(L), 
UPBD(L,CONSTR)
ELSE
WRITE(6, '13X,'"NO LIMIT'",11X,'"X" 
,11,) = ",F12.5,'"X")CB(L),UPBD(L,CONSTR)
ENDIF
ELSE
WRITE(6, '13X,'"NO LIMIT'",11X,'"X" 
,11,) = ",F12.5,'"X")CB(L),LWBD(L,CONSTR),CB(L), 
UPBD(L,CONSTR)
ENDIF
END
WRITE(6,('"x','',12,'",'')= ''F12.5,10X','' \"x', \\
,12,'")= NO LIMIT')CB(L),LMDB(L,CONSTR), \\
CB(L) 
     IF(CB(L).LE.Y)THEN 
     ZLP=ZLP+CO(CB(L))XLMDB(L,CONSTR) 
     ZUP=ZUP+10E8 
ENDIF 
ELSEIF(ABS(LMDB(L,CONSTR)) .GE.1E6)THEN 
WRITE(6,('"x','',12,'",'')= ''F12.5,10X','' \"x', \\
,12,'")= ''F12.5')CB(L),LMDB(L,CONSTR),CB(L), \\
LMDB(L,CONSTR) 
     IF(CB(L).LE.Y)THEN 
     ZLP=10E8 
     ZUP=ZUP+CO(CB(L))XLMDB(L,CONSTR) 
ENDIF 
ELSE 
WRITE(6,('"x','',12,'",'')= ''F12.5,10X','' \"x', \\
,12,'")= ''F12.5')CB(L),LMDB(L,CONSTR),CB(L), \\
LMDB(L,CONSTR) 
ENDIF 
ENDIF 
ENDIF 
LINES=LINES+1 
ENDIF 
CONTINUE 
4160 IF(MIN .EQ.2)THEN 
ZLP=-ZLP 
ZUP=-ZUP 
ZLS=-ZLS 
ZUS=-ZUS 
ENDIF 
C THE VALUE OF Z IS PRINTED FOR EACH UPPER AND LOWER LIMIT. 
IF(SLOUT .EQ. "S" OR SLOUT .EQ. "B")THEN 
IF(ABS(ZLS).GE.1E9.AND.ABS(ZUS).GE.1E9)THEN 
WRITE(1,('"z= "NO LIMIT Z="NO LIMIT"')) 
ELSEIF(ABS(ZLS).GE.1E9.AND.ABS(ZUS).LT.1E9)THEN 
WRITE(1,('"z= "NO LIMIT Z="F16.5")ZLS 
ELSEIF(ABS(ZLS).LT.1E9.AND.ABS(ZUS).GE.1E9)THEN 
WRITE(1,('"z= "F16.5" z="NO LIMIT")ZLS 
ELSE 
WRITE(1,('"z= "F16.5" z="F16.5")ZLS,ZUS 
ENDIF 
ENDIF 
IF(SLOUT .EQ. "P" OR SLOUT .EQ. "B")THEN 
IF(ABS(ZLP).GE.1E9.AND.ABS(ZUP).GE.1E9)THEN 
WRITE(6,('"z= "NO LIMIT"',11X, \\
,"z="NO LIMIT"')) 
ELSEIF(ABS(ZLP).GE.1E9.AND.ABS(ZUP).LT.1E9)THEN 
WRITE(6,('"z= "NO LIMIT"',11X,"z="F16.5')) 
ZUP 
ENDIF 
387
ELSEIF (ABS(ZLP).LT.1E9.AND.ABS(ZUP).GE.1E9) THEN
    WRITE(6,'(''Z= '',F16.5,1X,''NO LIMIT''
                    )')ZLP
    ELSE
        WRITE(6,'(''Z= '',F16.5,1X,''Z= '',F16.5)')ZLP,ZUP
    ENDIF
    WRITE(6,'(50(''''))')
    LINES=LINES+2
ENDIF
IF(SELOUT.EQ.'S'.OR.SELOUT.EQ.'B') THEN
    PAUSE
ENDIF
4120 CONTINUE
IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B') THEN
    WRITE(6,'(A1)')CHAP(12)
ENDIF
RETURN
SUBROUTINE COEFFR

INTEGER V,K,J,VT,COL,IFLAG,INEQ,INDEXL,INDEXE,INDEXG,
      INDEXM,INDEXO,INDEXS,INDEXT,INDEXU,INDEXV,
      INDEXW,INDEXX,INDEXY,INDEXZ,
      INDEX1,INDEX2,INDEX3,INDEX4,INDEX5,INDEX6,INDEX7,
      INDEX8,INDEX9,INDEX10,INDEX11,INDEX12,INDEX13,
      INDEX14,INDEX15,INDEX16,INDEX17,INDEX18,INDEX19,
      INDEX20,INDEX21,INDEX22,INDEX23,INDEX24,INDEX25,
      INDEX26,INDEX27,INDEX28,INDEX29,INDEX30,INDEX31,
      INDEX32,INDEX33,INDEX34,INDEX35,INDEX36,INDEX37,
      INDEX38,INDEX39,INDEX40,INDEX41,INDEX42,INDEX43,
      INDEX44,INDEX45,INDEX46,INDEX47,INDEX48,INDEX49,
      INDEX50,INDEX51,INDEX52,INDEX53,INDEX54,INDEX55,
      INDEX56,INDEX57,INDEX58,INDEX59,INDEX60,INDEX61,
      INDEX62,INDEX63,INDEX64,INDEX65,INDEX66,INDEX67,
      INDEX68,INDEX69,INDEX70,INDEX71,INDEX72,INDEX73,
      INDEX74,INDEX75,INDEX76,INDEX77,INDEX78,INDEX79,
      INDEX80,INDEX81,INDEX82,INDEX83,INDEX84,INDEX85,
      INDEX86,INDEX87,INDEX88,INDEX89,INDEX90,INDEX91,
      INDEX92,INDEX93,INDEX94,INDEX95,INDEX96,INDEX97,
      INDEX98,INDEX99,INDEX100,INDEX101,INDEX102,INDEX103,
      INDEX104,INDEX105,INDEX106,INDEX107,INDEX108,INDEX109,
      INDEX110,INDEX111,INDEX112,INDEX113,INDEX114,INDEX115,
      INDEX116,INDEX117,INDEX118,INDEX119,INDEX120,INDEX121,
      INDEX122,INDEX123,INDEX124,INDEX125,INDEX126,INDEX127,
      INDEX128,INDEX129,INDEX130,INDEX131,INDEX132,INDEX133,
      INDEX134,INDEX135,INDEX136,INDEX137,INDEX138,INDEX139,
      INDEX140,INDEX141,INDEX142,INDEX143,INDEX144,INDEX145,
      INDEX146,INDEX147,INDEX148,INDEX149,INDEX150,INDEX151,
      INDEX152,INDEX153,INDEX154,INDEX155,INDEX156,INDEX157,
      INDEX158,INDEX159,INDEX160,INDEX161,INDEX162,INDEX163,
      INDEX164,INDEX165,INDEX166,INDEX167,INDEX168,INDEX169,
      INDEX170,INDEX171,INDEX172,INDEX173,INDEX174,INDEX175,
      INDEX176,INDEX177,INDEX178,INDEX179,INDEX180,INDEX181,
      INDEX182,INDEX183,INDEX184,INDEX185,INDEX186,INDEX187,
      INDEX188,INDEX189,INDEX190,INDEX191,INDEX192,INDEX193,
      INDEX194,INDEX195,INDEX196,INDEX197,INDEX198,INDEX199,
      INDEX200,INDEX201,INDEX202,INDEX203,INDEX204,INDEX205,
      INDEX206,INDEX207,INDEX208,INDEX209,INDEX210,INDEX211,
      INDEX212,INDEX213,INDEX214,INDEX215,INDEX216,INDEX217,
      INDEX218,INDEX219,INDEX220,INDEX221,INDEX222,INDEX223,
      INDEX224,INDEX225,INDEX226,INDEX227,INDEX228,INDEX229,
      INDEX230,INDEX231,INDEX232,INDEX233,INDEX234,INDEX235,
      INDEX236,INDEX237,INDEX238,INDEX239,INDEX240,INDEX241,
      INDEX242,INDEX243,INDEX244,INDEX245,INDEX246,INDEX247,
      INDEX248,INDEX249,INDEX250,INDEX251,INDEX252,INDEX253,
      INDEX254,INDEX255,INDEX256,INDEX257,INDEX258,INDEX259,
      INDEX260,INDEX261,INDEX262,INDEX263,INDEX264,INDEX265,
      INDEX266,INDEX267,INDEX268,INDEX269,INDEX270,INDEX271,
      INDEX272,INDEX273,INDEX274,INDEX275,INDEX276,INDEX277,
      INDEX278,INDEX279,INDEX280,INDEX281,INDEX282,INDEX283,
      INDEX284,INDEX285,INDEX286,INDEX287,INDEX288,INDEX289,
      INDEX290,INDEX291,INDEX292,INDEX293,INDEX294,INDEX295,
      INDEX296,INDEX297,INDEX298,INDEX299,INDEX300,INDEX301,
      INDEX302,INDEX303,INDEX304,INDEX305,INDEX306,INDEX307,
      INDEX308,INDEX309,INDEX310,INDEX311,INDEX312,INDEX313,
      INDEX314,INDEX315,INDEX316,INDEX317,INDEX318,INDEX319,
      INDEX320,INDEX321,INDEX322,INDEX323,INDEX324,INDEX325,
      INDEX326,INDEX327,INDEX328,INDEX329,INDEX330,INDEX331,
      INDEX332,INDEX333,INDEX334,INDEX335,INDEX336,INDEX337,
      INDEX338,INDEX339,INDEX340,INDEX341,INDEX342,INDEX343,
      INDEX344,INDEX345,INDEX346,INDEX347,INDEX348,INDEX349,
      INDEX350,INDEX351,INDEX352,INDEX353,INDEX354,INDEX355,
      INDEX356,INDEX357,INDEX358,INDEX359,INDEX360,INDEX361,
      INDEX362,INDEX363,INDEX364,INDEX365,INDEX366,INDEX367,
      INDEX368,INDEX369,INDEX370,INDEX371,INDEX372,INDEX373,
      INDEX374,INDEX375,INDEX376,INDEX377,INDEX378,INDEX379,
      INDEX380,INDEX381,INDEX382,INDEX383,INDEX384,INDEX385,
      INDEX386,INDEX387,INDEX388,INDEX389,INDEX390,INDEX391,
      INDEX392,INDEX393,INDEX394,INDEX395,INDEX396,INDEX397,
      INDEX398,INDEX399,INDEX400,INDEX401,INDEX402,INDEX403,
      INDEX404,INDEX405,INDEX406,INDEX407,INDEX408,INDEX409,
      INDEX410,INDEX411,INDEX412,INDEX413,INDEX414,INDEX415,
      INDEX416,INDEX417,INDEX418,INDEX419,INDEX420,INDEX421,
C EACH ROW IS CHECKED TO DETERMINE IF IT IS THE ROW WHICH
C HAS THE BASIS VARIABLE OF THE COLUMN UNDER INVESTIGATION.
IF(CB(ROW) .EQ. COL) THEN
   DELADN=-10EB
   DELAUP=10EB
   DO 4245 L=1,VT
   THE ONLY COLUMNS WHICH HAVE AN OBJECTIVE FUNCTION
   COEFFICIENT WHICH COULD BE DRIVEN NEGATIVE ARE THOSE
   WHICH DO NOT HAVE ARTIFICIAL VARIABLES.
   IF(L.LT.INDEXL .OR. INE(变数) .EQ. 0) THEN
   ONLY NON-BASIC COLUMNS HAVE THE POTENTIAL TO ENTER
   TO ENTER THE BASIS.
   IF(BASIC(L) .NE. 1) THEN
   DIVISION BY ZERO IS AVOIDED.
   IF(ABS(AF(ROW,CONSTR+INDEXL-1)).LT.AF(ROW,
         L) .AND.(AF(ROW,CONSTR+INDEXL-1)) .LT. .00001) THEN
   IF(CF(L) .LT. 0) THEN
      TEMP=-9.9EB
   ELSE
      TEMP=9.9EB
   ENDIF
   ELSE
      IF ALL CONDITIONS HAVE BEEN MET, THE VALUE
      OF THE CHANGES TO THE COEFFICIENT IS FOUND
      WHICH WOULD CAUSE AN OBJECTIVE FUNCTION
      COEFFICIENT TO BE DRIVEN TO ZERO.
      TEMP=-CF(L)/(AF(ROW,CONSTR+INDEXL-1)
      CF(L)-AF(ROW,L)*CF(CONSTR+INDEXL-1))
   ENDF
   THE MINIMUM POSITIVE AND MAXIMUM NEGATIVE
   (MINIMUM ABSOLUTE) VALUES ARE RETAINED.
   IF(TEMP .LT. 0) THEN
      DELAUP=MAX1(DELAUP,TEMP)
   ELSE
      DELADN-MIN1(DELADN,TEMP)
   ENDF
   ENDF
   ENDF
   END
4245 CONTINUE
DO 4260 M=1,X
C FOR EACH ROW NOT POSSESSING THE BASIS VARIABLE UNDER
C INVESTIGATION, THE VALUE OF THE CHANGE TO THE
C COEFFICIENT IS FOUND WHICH WOULD CAUSE A RIGHT-HAND-
C SIDE VALUE TO BE DRIVEN TO ZERO.
IF(CB(M) .NE. COL) THEN
   IF(ABS(BF(M)+AF(ROW,CONSTR+INDEXL-1)-BF(ROW))
      AF(M,CONSTR+INDEXL-1)) .LT. .00001) THEN
      IF(BF(M) .LT. 0) THEN
         TEMP=-9.9EB
      ELSE
         TEMP=9.9EB
      ENDF
   ELSE
      ENDF
   END
ELSE IF (CB(N) .GE. INDEXE) THEN
  DELAUP = 0
  DELADN = 0
ELSE
  TEMP = BF(M)/ (BF(N) * AF(N, CONSTR + INDEXL - 1) - 
               BF(ROW) * AF(M, CONSTR + INDEXL - 1))
ENDIF

THE MINIMUM VALUES ARE COMPARED TO THE PREVIOUSLY
FOUND MINIMUM VALUES, AND THE SMALLEST ARE
RETAINED.

IF (TEMP .GT. 0) THEN
  DELAUP = MINI (DELAUP, TEMP)
ELSE
  DELADN = MAXI (DELAUP, TEMP)
ENDIF
ENDIF

C

4260 CONTINUE
C
THE JUST DETERMINED MINIMUM VALUES ARE CHECKED FOR
ILL-CONDITIONING.
C
CKILL1 = + DELADN * AF (ROW, CONSTR + INDEXL - 1)
CKILL2 = - DELAUP * AF (ROW, CONSTR + INDEXL - 1)

IF (ABS (CKILL1) .LT. 1.0 .OR. ABS (CKILL2) .LT. 1.0) THEN
  ILL1 = 1
  ILL2 = 1
ENDIF

CONTINUE

4230 CONTINUE
C
THE UPPER AND LOWER BOUNDS ARE DETERMINED FOR THOSE
COLUMNS WITH VARIABLES NOT IN THE BASIS.
C
UPBD = AO (CONST, COL) + DELAUP
LWBD = AO (CONST, COL) + DELADN
ELSE
THE CONSTRAINT COEFFICIENT RANGES ARE PRINTED ACCORDING TO
C
THE CURRENT CONDITIONS.
C
IF (SELOUT .EQ. 'S' .OR. SELOUT .EQ. 'B') THEN
  IF (HEAD .EQ. 0 .OR. LINES .GE. 10) THEN
    IF (LINES .GE. 18) THEN

391
PAUSE
ENDIF
WRITE1,"'(AI')"\ CHAR(12)
WRITE1,"'(//,A0(("4")))'
WRITE1,"'("COEFFICIENT LOWERLIMIT UPPERLIMIT'')'
LINES=5
HEAD1=1
ENDIF
IF(ABS(LMBD).GE.1E7. AND. ABS(UPBD).GE.1E8) THEN
WRITE1,"'(1X,'''A''',II,''''''',II,'') = '','
'NO LIMIT  NO LIMIT''')CONSTR,COL
ELSEIF(ABS(LMBD).GE.1E7) THEN
WRITE1,"'(1X,'''A''',II,''''''',II,'') = '','
'NO LIMIT'',F15.5')CONSTR,COL,UPBD
ELSEIF(ABS(UPBD).GE.1E8) THEN
WRITE1,"'(1X,'''A''',II,''''''',II,'') = '',F14.5,
'NO LIMIT''')CONSTR,COL,LMBD
ELSE
WRITE1,"'(1X,'''A''',II,''''''',II,'') = '',F14.5,
F15.5')CONSTR,COL,LMBD,UPBD
ENDIF
IF(ILLI.EQ.1) THEN
WRITE1,"'(8X,'''MAY BE ILL-CONDITIONED''/)")
ILLI=0
LINES=LINES+2
ENDIF
LINES=LINES+1
ENDIF
IF(SELOUT.EQ. 'P' . OR. SELOUT.EQ. 'B') THEN
IF(HEAD2.EQ.0 . OR. LINEP.EQ.56) THEN
IF(LINEP.GE.50) THEN
WRITE6,"'(01)''CHAR(12)
ENDIF
WRITE6,"'(//,A5)')'FN
WRITE6,"'(A7(''''))'
WRITE6,"'(5X,'''COEFFICIENT LOWERLIMIT UPPERLIMIT''
'(//'))'
HEAD2=1
LINEP=8
ENDIF
IF(ABS(LMBD).GE.1E7. AND. ABS(UPBD).GE.1E8) THEN
WRITE6,"'(6X,'''A''',II,''''''',II,'') = '','
'NO LIMIT  NO LIMIT''')CONSTR,COL
ELSEIF(ABS(LMBD).GE.1E7) THEN
WRITE6,"'(6X,'''A''',II,''''''',II,'') = '','
'NO LIMIT'',F15.5')CONSTR,COL,UPBD
ELSEIF(ABS(UPBD).GE.1E8) THEN
WRITE6,"'(6X,'''A''',II,''''''',II,'') = '',F14.5,
'NO LIMIT''')CONSTR,COL,LMBD
ELSE
WRITE6,"'(6X,'''A''',II,''''''',II,'') = '',F14.5,
F15.5')')CONSTR,COL,LMBD,UPBD
392
IF (ILL2.EQ.1) THEN
   WRITE(6, '(10X,''MAY BE ILL-CONDITIONED''/))
   ILL2=0
   LINEP=LINEP+2
ENDIF
LINEP=LINEP+1
ENDIF

4220  CONTINUE
4210  CONTINUE

LIKES

IF (SELOUT .EQ. 'S' .OR. SELOUT .EQ. 'B') THEN
   PAUSE
   WRITE(1,'(A1)') CHAR(12)
   WRITE(1,'(A,40(''t''))')
   WRITE(1,'(''COEFFICIENT LOWERLIMIT''
   '' UPPERLIMIT''/)
ENDIF
IF (SELOUT .EQ. 'P' .OR. SELOUT .EQ. 'B') THEN
   IF (LINEP + K .GT. 42) THEN
      WRITE(6, '(A1)') CHAR(12)
      LINEP = 0
   ENDIF
   WRITE(6, '(/''///'',15X,A10,//,47(''t''))') FN
   WRITE6('4X, ''COEFFICIENT LOWERLIMIT''
   '' UPPERLIMIT''/
ENDIF

C OBJECTIVE FUNCTION COEFFICIENT RANGING IS DETERMINED FOR EACH
C COLUMN IN THE ORIGINAL PROBLEM.
DO 4240 COL = 1,7
   CLUMER=10E6
   CLURE=10E6
   IF (BASIC(COL) .EQ. 1) THEN
      DO 4270 CONSTR=1,7
      THE CONSTRAINT IS FOUND WHICH HAS THE VARIABLE IN THE
      BASIS ASSOCIATED WITH THE CURRENT COLUMN.
      IF (B(CO(NSTR)).EQ.0) THEN
         DO 4280 J=1,7
      ALL COLUMNS WITH VARIABLES NOT IN THE BASIS AND NOT
      INCLUDING THE CURRENT COLUMN ARE CHECKED TO FIND THE
      CHANGE (IF ANY) WHICH WOULD DRIVE THE OBJECTIVE
      FUNCTION COEFFICIENT TO ZERO.
      IF (BASIC(J).NE.1) THEN
         IF (ABS(AF(CONSTR)).LT.0.0001) THEN
            TEMP=10E6
            TEMP=10E6
         ELSE
            IF (MINN .EQ. 2) THEN
               TEMP=CF(J)/AF(CONSTR,J)
         C 793
      ENDIF
   IF (BASIC(J).NE.1) THEN
         IF (ABS(AF(CONSTR,J)).LT.0.0001) THEN
            TEMP=10E6
            TEMP=10E6
         ELSE
            IF (MINN .EQ. 2) THEN
               TEMP=CF(J)/AF(CONSTR,J)
ELSE
  TEMP = CF(J)/AF(CONSTR,J)
ENDIF
ENDIF

C
MINIMUM VALUES ARE RETAINED.
IF(TEMP.GT.0)THEN
  CUPPER = MINI(CUPPER, TEMP)
ELSE
  CLOWER = MAXI(CLOWER, TEMP)
ENDIF
ENDIF
ENDIF

4280 CONTINUE
ENDIF

4270 CONTINUE
ELSE
C
VALUES ARE FOUND FOR COLUMNS NOT ASSOCIATED WITH THE BASIS.
IF(MN.LE.2)THEN
  CLOWER = CF(COL)
ELSE
  CUPPER = CF(COL)
ENDIF
ENDIF

C
THE RESULTS ARE PRINTED.
IF(SELOUT.EQ.'S' .OR. SELOUT.EQ. 'B') THEN
  IF(ABS(CLOWER).GE.1E7 .AND. ABS(CUPPER).GE.1E7) THEN
    WRITE(1, '(4X, ''C1(I, I, I) = NO LIMIT'')) COL
  ELSEIF(ABS(CLOWER).GE.1E7) THEN
    WRITE(1, '(4X, ''C1(I, I, I) = NO LIMIT'')) CLOWER, CUPPER
  ELSEIF(ABS(CUPPER).GE.1E7) THEN
    WRITE(1, '(4X, ''C1(I, I, I) = NO LIMIT'')) CUPPER
  ELSE
    WRITE(1, '(4X, ''C1(I, I, I) = '' , F14.5)') C1(I, I, I)
  ENDIF
  LINES = LINES + 1
ENDIF
END IF

IF(LINES.GT.8) THEN
  LINES = 0
  PAUSE 4
  WRITE(1, '(A1)' ) CHAR(12)
  WRITE(1, ( ''COEFFICIENT LOWER LIMIT''
   '' UPPER LIMIT''))
ENDIF

ENDIF

IF(SELOUT.EQ.'P' .OR. SELOUT.EQ. 'B') THEN
  IF(ABS(CLOWER).GE.1E7 .AND. ABS(CUPPER).GE.1E7) THEN
    WRITE(6, '(8X, ''C1(I, I) = NO LIMIT''))
  ELSE
    WRITE(6, '(8X, ''C1(I, I) = '' , F14.5)') C1(I, I)
  ENDIF
ENDIF
ELSEIF(ABS(CLOWER).GE.1E7) THEN
    WRITE(6, '(A1)') = 'NO LIMIT ''F14.5')
    (COL,CLOWER)
ELSEIF(ABS(CUPPER).GE.1E7) THEN
    WRITE(6, '(A1)') = 'NO LIMIT ''F14.5')
    (COL,CUPPER)
ELSE
    WRITE(6, '(A1)') = ''F14.5', (COL.CLOWER,CUPPER)
END IF
ENDIF
ENDIF
CONTINUE

4240 CONTINUE
IF(SELOUT.EQ. 'P' .OR. SELOUT.EQ. 'D') THEN
    WRITE(6, '(A1)') = 'CHAR12'
ENDIF
IF(SELOUT.EQ. 'S' .OR. SELOUT.EQ. 'B') THEN
    PAUSE
ENDIF
RETURN
END
SUBROUTINE MULCM6
USE: THIS SUBROUTINE ACCEPTS MULTIPLE CHANGES TO ANY OR ALL VALUES OF THE ORIGINAL PROBLEM. IT THEN DETERMINES THE TOTAL EFFECT ON THE FINAL TABLEAU.

Called by: PROGRAM MAINSA
Calls: SUBROUTINE CHECK2
SUBROUTINE CHECK
SUBROUTINE COMUL

Variables:

Uses: UCPECK2 in UNIT47.CODE

Variables:

Includes:

If SELOUT .EQ. 'P' .OR. SELOUT .EQ. '3B': THEN
WRITE(6,'(A1)') 'FM
ENDIF

Variables are set to zero.
DO 4390 COL=1,V
DELTAC(COL)=0.0
NEWCJ(COL)=0.0
CONTINUE
DO 4395 CONSTR=1,K
DELTAB(CONSTR)=0.0
NEWB(CONSTR)=0.0
DO 4390 COL=1,V
DELTAC(CONSTR, COL) = 0.0
NEWA(CONSTR, COL) = 0.0

4390 CONTINUE
4395 CONTINUE
4370 WRITE(1, '(A1)') CHAR(12)
C THE USER CAN SELECT ANY OF THE THREE TYPES OF CHANGES. WHEN ALL
C CHANGES HAVE BEEN COMPLETED, THE PROGRAM MOVES TO THE SOLVING
C PHASE.
WRITE(I, '(3X, "SELECT THE PARAMETERS TO BE CHANGED",
   \(6X, "(1)\) C(I) /\",
   \(6X, "(2)\) ALL, J /\",
   \(6X, "(3)\) 3(I) /\",
   \(6X, "(4)\) CHANGES COMPLETE",
   \(6X, "(5)\) RETURN TO MAIN MENU")')
FREAD(5, '(A1)') SELINP(1)
WRITE(I, '(A1)') CHAR(12)
IF (SELINP(1) .EQ. 'I') THEN
    C INPUT OF CHANGES TO THE OBJECTIVE FUNCTION COEFFICIENTS. EACH
    C INPUT COLUMN IS CHECKED FOR VALIDITY AS IS THE NEW VALUE OF THE
    C COEFFICIENT.
4315 WRITE(I, '(5X, 5X, "PLEASE ENTER THE COLUMN",
   \(5X, "TO BE CHANGED",
   \(5X, "PRESS DONE IF COMPLETE")")
READ(5, '(2A1)') SELINP(1), SELINP(2)
IF (SELINP(1) .EQ. 'D') THEN
    GOTO 4370
ENDIF
CALL CHECK2(SELINP, 2, INVAL, COL)
IF (INVAL .EQ. 1) THEN
    WRITE(I, '(A1)') CHAR(12)
    WRITE(I, '(2A1)') SELINP(2), 'INVALID RESPONSE, PLEASE REENTER")")
    GOTO 4315
ENDIF
4302 WRITE(I, '(5X, "THE ORIGINAL VALUE OF C('',12,'') WAS"')
WRITE(1, '(10X,F10.5)') CO(COL)
WRITE(I, '(5X, "PLEASE ENTER NEW VALUE")")
READ(5, '(10A1)') (P(L), L=1,10)
CALL CHECKF(P, INVAL, NEWCJ(COL))
IF (INVAL .EQ. 1) THEN
    WRITE(I, '(A1)') CHAR(12)
    WRITE(I, '(5X, "INVALID RESPONSE, PLEASE REENTER")")
    GOTO 4302
ENDIF
IF (SELOUT. EQ. 'P'. OR. SELOUT. EQ. 'B') THEN
    WRITE(I, '(*') "THE OLD VALUE OF C('',12,'') WAS: ",F10.3,
    \(\*\) "THE NEW VALUE IS: ",F15.8)') CO(COL), CG(COL), NEWCJ(COL)
ENDIF
C THE CHANGE IS DETERMINED.
DELTAC(COL) = NEWCJ(COL) - CO(COL)
WRITE(I, '(A1)') CHAR(12)
GOTO 4315
ELSEIF (SELINP(1) .EQ. '2') THEN
CHANGES TO THE CONSTRAINT COEFFICIENT ARE DETERMINED.

WRITE(1,'(5(1.3x,"PLEASE ENTER THE ROW TO BE CHANGED"/))
WRITE(1,'(5x,"PRESS D.ONE IF COMPLETE"/))
READ(5,'(2AI)') SELINP(1), SELINP(2)
IF (SELINP(1).EQ. 'D') THEN
  SOTO 4370
ENDIF
CALL CHECK2(SELINP,2,K,INVAL,CONSTR)
IF (INVAL.EQ. 1) THEN
  WRITE(1,'(A1)') CHAR(12)
  WRITE(1,'(2(/),5X,"INVALID RESPONSE, PLEASE REENTER"/))
  SOTO 4325
ENDIF
WRITE(1,'(AI)') CHAR(12)
WRITE(1,'(5(1.2x,"PLEASE ENTER THE COLUMN TO BE CHANGED"/))
READ(5,'(2AI)') SELINP(1), SELINP(2)
CALL CHECK2(SELINP,2,V,INVAL,COL)
IF (INVAL.EQ. 1) THEN
  WRITE(1,'(A1)') CHAR(12)
  WRITE(1,'(2(/),5X,"INVALID RESPONSE, PLEASE REENTER"/))
  SOTO 4335
ENDIF
WRITE(:, '3X, "ENTER NEW VALUE (10 CHARACTERS MAX)"/)
READ(5,'(10P:)') (P(L),L=1,10)
CALL CHECKF(F, INVAL, NEWA(CONSTR,COL))
IF (INVAL.EQ. 1) THEN
  WRITE(1,'(A1)') CHAR(12)
  WRITE(1,'(1/,"INVALID RESPONSE, PLEASE REENTER"/))
  SOTO 4312
ENDIF

398
IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B')THEN

IF(COL.EQ.9)THEN
    WRITE(6,'(1X,'"THE OLD VALUE OF A","I","","I","
    "WAS","F14.5","---THE NEW VALUE IS","F15.7")
    CONSTR,COL,AD(CONSTR,COL),NEWA(CONSTR,COL)
    ELSE
    WRITE(6,'(1X,"THE OLD VALUE OF A","I","
    "WAS","F13.5","---THE NEW VALUE IS","F15.7")
    CONSTR,COL,AD(CONSTR,COL),NEWA(CONSTR,COL)
ENDIF
ELSEIF(COL.EQ.9)THEN
    WRITE(6,'(1X,"THE OLD VALUE OF A","I","
    "WAS","R","---THE NEW VALUE IS","F15.7")
    ELSE
    WRITE(6,'(1X,"THE OLD VALUE OF A","I","
    "WAS","F15.7")
ENDIF
ENDIF

IF(NEG(CONSTR).EQ.1)THEN
    AD(CONSTR,COL)=-AD(CONSTR,COL)
    NEWA(CONSTR,COL)=-NEWA(CONSTR,COL)
ENDIF

DELTA(CONSTR,COL)=-DELTA(CONSTR,COL)

WRITE(6,'(A1)') CHAR(12)
6010 4325
ELSEIF(SELINP(1).EQ.'3')THEN
C CHANGES TO THE RIGHT-HAND SIDE ARE FOUND.
4345 WRITE(6,'(5X,"PLEASE ENTER THE ROW TO BE CHANGED",/,
    5X,"PRESS DONE IF COMPLETE")')
READ(5,('2AI') SELINP(1),SELINP(2)
IF(SELINP(1).EQ.'D')THEN
    GOTO 4370
ENDIF
CALL CHECKI(SELINP,2,K,INVAL,CONSTR)
IF(INVAL.EQ.1)THEN
    WRITE(6,'(1X,"A1")') CHAR(12)
    WRITE(6,'(1X,"ENTER NEW VALUE (10 CHARACTERS MAX)")')
    CALL CHECKI(INVAL,NEWB(CONSTR))
    IF(INVAL.EQ.1)THEN
        GOTO 4345
ENDIF
ENDIF
IF(NEG(CONSTR).EQ.1)THEN
    AD(CONSTR)=-AD(CONSTR)
ENDIF

4322 WRITE(6,'(5X,"THE ORIGINAL VALUE OF B","I","
    CONSTR"
    WRITE(6,'(10X,F10.3)') B(CONSTR)
    WRITE(6,'(5X,"ENTER NEW VALUE (10 CHARACTERS MAX)")')
    READ(5,('1D1') P(L),L=1,10)
    CALL CHECKP(INVAL,NEWB(CONSTR))
    IF(INVAL.EQ.1)THEN
        GOTO 4345
ENDIF

399
WRITE(1,'(A1)')CHAR(12)
WRITE(1,'(/,,"INVALID RESPONSE, PLEASE REENTER")')
GOTO 4522
ENDIF
IF(SELOUT.EQ.'F'.OR.SELOUT.EQ.'B') THEN
WRITE(6,'("THE OLD VALUE OF B(1,2) WAS ",FIO.3,
"..THE NEW VALUE IS ",FIO.3)')CONSTR,0DD(CONSTR),
NEWB(CONSTR)
ENDIF
IF(NEG(CONSTR).EQ.1) THEN
DO (CONSTR) u-DU(CONSTR)
NEWB(CONSTR)-NEWB(CONSTR)
ENDIF
DELTAB'CONSTP)=NEWB(CONSTR)-DO(CONSTR)
ELSEIF (SELINP1).EQ.'4') THEN
C WHEN ALL CHANGES HAVE BEEN ENTERED, THE NET EFFECT OF THESE
C CHANGES ON THE FINAL TABLEAU IS DETERMINED. THIS GENERALLY
C CONSISTS OF MATRIX MULTIPLICATION. THE FINAL TABLEAU IS INVERSE
C (PLUS OBJECTIVE FUNCTION COEFFICIENT) IS MULTIPLIED BY THE
C MATRIX CONTAINING THE CUMULATIVE CHANGES.
DO 2000 CONSTR=1,K
z=2*DELTAB(CONSTR)*CF(CONSTR+INDEXL-1)
DO 2010 ROW=1,K
BF(CONSTR)=BF(CONSTR)+DELTAB(ROW)*AF(CONSTR,ROW+INDEXL-1)
2010 CONTINUE
2000 CONTINUE
DO 2030 COL=1,Y
IF(XMNX.EQ.2) THEN
CF(COL)=CF(COL)+DELTAC(COL)
ELSE
CF(COL)=CF(COL)-DELTAC(COL)
ENDIF
DO 2050 CONSTR=1,K
CF(COL)=CF(COL)+DELTAA(CONSTR,COL)+CF(CONSTR+INDEXL-1)
2030 CONTINUE
2050 CONTINUE
DO 2060 CONSTR=1,K
DO 2050 COL=1,Y
DO 2060 I=1,K
AF(CONSTR,COL)=AF(CONSTR,COL)+DELTAA(I,COL)*
AF(CONSTR,I+INDEXL-1)
2060 CONTINUE
2050 CONTINUE
2040 CONTINUE
ELSEIF (SELINP1).EQ.'5') THEN
RETURN
ELSE
WRITE(1,'(A1)')CHAR(12)
WRITE(1,'(/,,"INVALID RESPONSE, PLEASE REENTER")')
GOTO 4570
400
ENDIF
CALL CONHUL
RETURN
END
SUBROUTINE: CONMUL

This subroutine is a continuation of the above subroutine merely separated to allow compilation.

INTEGER CONSTR, COL, IFLAG, VT, ROW, CB
REAL AF, BF, CF, Z, NEWB(20), BM
CHARACTER SELOUT, SELFN
COMMON/ONE/SELOUT, SELFN
COMMON/TWO/VT, INDEP, INDEXE, NSEQ, NLC, NNC, NIEQ(20), MMIN, BM

INCLUDE COMVAR

IFLAG(9)=0
DO 4303 CONSTR=1, K
C THE MODIFIED FINAL TABLEAU IS NOW RETURNED TO THE BASIC SOLUTION FORM.
NEWB(I)=AF(CONSTR,CB(CONSTR))
IF (ABS(NEWB(I)).LT.E-6) THEN
  IFLAG(9)=1
  RETURN
ENDIF
DO 4313 COL=1, VT
C EACH CONSTRAINT IS DIVIDED BY THE VALUE OF THE COEFFICIENT
C IN THE COLUMN ASSOCIATED WITH THE VARIABLE IN THE BASIS.
C THIS WILL RETURN THE COEFFICIENT TO A VALUE OF ONE (DIVIDED
C BY ITSELF).
AF(CONSTR,COL)=AF(CONSTR,COL)/NEWB(I)
CONTINUE
BF(CONSTR)=BF(CONSTR)/NEWB(I)
DO 4323 ROW=1, K
C THIS CONSTRAINT IS THEN ADDED TO EACH OTHER CONSTRAINT AND
C THE OBJECTIVE FUNCTION IN THE REQUIRED MULTIPLES TO DRIVE ALL
C OTHER VALUES IN THE COLUMN TO ZERO.
NEWB(I)=AF(ROW,CB(CONSTR))
IF (POW .NE. CONSTR) THEN
  DO 4333 COL=1, VT
  CONTINUE
  END
  CALL AONE
  CALL AONE
        END
AF(ROW, COL) = AF(ROW, COL) + NEWB(1) # AF(CONSTR, COL)

CONTINUE
BF(ROW) = BF(ROW) + NEWB(1) # BF(CONSTR)
ENDIF

CONTINUE
IF(CP(CONSTR) .LT. INDEXE) THEN
NEWB(1) = CF(CP(CONSTR));
DO 4343 COL = 1, VT
CF(COL) = CF(COL) + NEWB(1) # AF(CONSTR, COL)
CONTINUE
ENDIF

CONTINUE
NEWB(1) = 0.0
DO 4377 CONSTR = 1, K
IF BIG M HAD BEEN SUBTRACTED FROM SOME OBJECTIVE FUNCTION
COEFFICIENTS IN UNIT 48, THEY ARE ADDED BACK.
IF(INEQ(CONSTR) .NE. 0) THEN
CF(CONSTR + INDEXE - 1) = CF(CONSTR + INDEXE - 1) + SM
ENDIF
CONTINUE

CONTINUE
A TEST IS MADE FOR OPTIMALITY OF THE NEW BASIC TABLEAU.
DO 4353 COL = 1, VT
IF(CF(COL) .LT. 0.00001) THEN
NEWB(1) = 100.
ENDIF
CONTINUE

CONTINUE
DO 4363 CONSTR = 1, K
IF(BF(CONSTR) .LT. -0.00001) THEN
NEWB(1) = 100
ENDIF
CONTINUE

IF(NEWB(1) .GT. 99.) THEN
DO 4373 CONSTR = 1, K
IF(CP(CONSTR) .GE. INDEXE) THEN
IF(BF(CONSTR) .GT. 0) THEN
WRITE(1, '(5(/,5X,**********NOT FEASIBLE**********'))
IFLAG(5) = 1
RETURN
ENDIF
ENDIF
CONTINUE
WRITE(1, '(5(/,5X,**********NOT OPTIMAL**********'))
ELSE
WRITE(1, '(5(/,5X,**********STILL OPTIMAL**********'))
ENDIF
PAUSE
RETURN
END
SUBROUTINE ADDCON

USE: THIS SUBROUTINE ALLOWS AN ADDITIONAL CONSTRAINT OR VARIABLE TO BE ADDED TO A PROBLEM WHICH HAS ALREADY BEEN SOLVED BY MODULES 2 OR 3. THE EFFECTS OF THE ADDITION ARE CALculated AND THE NEW TABLEAU IS PUT INTO A BASIC SOLUTION.

CALLED BY: PROGRAM MAINSA
CALLS : SUBROUTINE CHECK2

VARIABLES:

USES : V,K,VT,IFLAG,INEG,MAXMX,CI,NEG,NEC,NLC,NEG,AF,BF,BF,CF,V

MODIFIES : INDEXL,V,VT,K,AF,BF,CF,Z,DELTANLC,NE6(20),NLC,N6C

USES UCHECK2 IN UNIT47.CODE OVERLAY

SUBROUTINE ADDCON
INTEGER V,K,JCDI.,VT,RFLA,INA,NXNN.NEC,N6C,NLC,CONSTR,CB
REAL A,AF,A,F,CF,Z,DELTANLC,NE6(20),BM
CHARACTER SELINP(2),FNIIO,SELOUTSI
COMMON/GNE/SELINP,FN
COMMON/TWO/VT,INDEX6,INDEXE,N6C,NLCNEC,NE6(20),MAXMX,BM

INCLUDE CONVAR

IF (V.EQ.20 .OR. K .EQ. 20 ) THEN
  WRITE(I,'(5(I,5/X,'"THIS PROBLEM IS TOO LARGE FOR"/,4X)
     "ADDIIONS. "')
  RETURN
ENDIF

WRITE(I,'(AI)'1CHAR(12)
WRITE(I,'(/Ii,3X,"THIS SEGMENT ALLOWS YOU TO ADD AN",/3X
     "ADDITIONAL CONSTRAINT OR VARIABLE TO ",")
WRITE(I,'(3X,"AN ALREADY SOLVED LINEAR PROGRAMMING ",/3X
     "PROBLEM ")',/44)

4400 WRITE(I,'(9X,"DO YOU WISH TO ADD A "://,14X)
   "CONSTRAINT"://,9X
   "OR"://,1X
   WRITE(I,'('\"VARIABLE"://,1X
   "SELECT \\
     OR \\
     READ\'S\'(5(\"I\") SELINP(1)
   IF (SELINP(1).EQ. "V") THEN
     IF A VARIABLE IS ADDED, IT IS PLACED JUST AFTER THE LAST VARIABLE IN THE ORIGINAL PROBLEM. ALL VARIABLES TO THE RIGHT ARE MOVED ONE COLUMN TO THE RIGHT. ALL INDICES ARE RESET.
     VT=VT+1
V=V+1
INDEXL=INDEXL+1
INDEXE=INDEXE+1
DO 4420 COL=VT,V+1,-1
   CF(COL)=CF(COL-1)
   DO 4410 CONSTR=1,K
      AF(CONSTR,COL)=AF(CONSTR,COL-1)
   CONTINUE
4420 CONTINUE
DO 4405 CONSTR=1,K
   IF (CB(CONSTR) .GE. V )THEN
      CB(CONSTR)=CB(CONSTR)+1
   ENDIF
4405 CONTINUE
WRITE(6,'(//,4X,0PEASE ENTER THE COEFFICIENT FOR',0X,0THE OBJECTIVE FUNCTION',0X)')
IF(Y.LT.10)THEN
   ELSE
      WRITE(6,11)'
      ENDIF
C THE INPUTS ARE CHECKED FOR VALIDITY.
CALL CHECKCP,INVAL.CD(V))
IF(INVAL .EQ. 1)THEN
   WRITE(6,'(/',5X,'INVALID RESPONSE, PLEASE REENTER')')
   GOTO 4411
ENDIF
WRITE(6,'(/',12)') CHAR(12)
4411 WRITE(6,'(/',4X,'PLEASE ENTER THE COEFFICIENT FOR',0X,0EACH CONSTRAINT',0X)')
DO 4430 CONSTR=1,K
   WRITE(6,'(/,8X,'A(',I2,'',') = ''',0X)') CONSTR,V
   READ(S,'(1041') P(I)L,L=1,10)
   CALL CHECKP(INVAL,AD(INDEKL-1,1)
   IF(INVAL .EQ. 1)THEN
      WRITE(6,11) CHAR(12)
      WRITE(6,'(/,4X,'INVALID RESPONSE, PLEASE REENTER',0X)')
      GOTO 4412
   ENDIF
4430 CONTINUE
CF(V)=-AD(V)
C THE NEW ADDITIONS ARE MULTIPLIED BY B-INVERSE TO GET A
C MODIFIED FINAL TABLEAU.
DO 4495 CONSTR=1,K
   CF(V)=CF(V)+AD(CONSTR,V)*CF(CONSTR,INDEXL-1)
   DO 4495
DELTAA(CONSTRA)=0.0
CONTINUE
DO 4497 CONSTRA=1,K
DO 4499 I=1,K
DELTAA(CONSTRA)=DELTAA(CONSTRA)+AD(I,V)*AF(CONSTRA,
I+INDEX-1)
CONTINUE
4497 CONTINUE
DO 4498 CONSTRA=1,K
AF(CONSTRA,V)=DELTAA(CONSTRA)
CONTINUE
C IF UNIT40 REMOVED THE BIG N VALUE, IT IS ADDED BACK.
DO 4476 CONSTRA=1,K
IF(INEG(K).NE.0)THEN
CF(CONSTRA+INDEX-1)+CF(CONSTRA+INDEX-1)+BM
ENDIF
4476 CONTINUE
ELSEIF (SELPN(I) .EQ. "C") THEN
C IF A CONSTRAINT IS ADDED, IT IS PLACED AT THE BOTTOM.
C INDICES ARE RESET AS REQUIRED. THE COLUMN OR COLUMNS
C REQUIRED FOR THE NEW CONSTRAINT ARE ADDED TO THE RIGHT OF THE
C EXISTING COLUMNS.
K=K+1
WRITE(1,*(A1)) CHAR(12)
4413 WRITE(1,*(//,5X,'#PLEASE ENTER THE NEW CONSTRAINT'))
DO 4440 COL=1,V
WRITE(1,*(//,10X,'A('',',12,'',',',12,'',') = ''4'') K,COL
READ(5,(10A1)) (P(L),L=1,10)
CALL CHECKP(INEL,AFK,CON)
IF(INEL .EQ. 1)THEN
WRITE1,*(A1))CHAR(12)
WRITE(1,*(//,4X,'#INVALID RESPONSE. PLEASE REENTER'))
GOTO 4413
ENDIF
4440 CONTINUE
WRITE1,*(A1,//*')CHAR(12)
4441 WRITE1,*(//,4X,'#IS CONSTRAINT OF THE FORM ''///'')
WRITE1,*(//,10X,'1 LESS THAN''///',10X
.'',2)' GREATER THAN''///',10X
.'',3)' EQUALS''///')
HEADS ((1))/ SELPN(I)
CALL CHECK2(SELPN(I,1,INEL,INEK))
IF(INEL .EQ. 1)THEN
WRITE1,*(A1))CHAR(12)
WRITE1,*(//,4X,'#INVALID RESPONSE, PLEASE REENTER'))
GOTO 4441
ENDIF
INEK(K)=INEK(K)-1
WRITE1,*(A1,//*')CHAR(12)
4444 WRITE1,*(//,4X,'#PLEASE ENTER THE RIGHT HAND SIDE ''///')
READ(5,(10A1)) (P(L),L=1,10)
CALL CHECK(P, INVALID, BF(K))

IF(INVALID .EQ. 1) THEN
  WRITE(1,'(A1)') CHAR(12)
  WRITE(1,'(''/.,4X,''INVALID RESPONSE, PLEASE REENTER'')')
  GOTO 4414
ENDIF

TEMP = 0.0
DO 4439 COL = VT + 1, VT
  AF(K, COL) = 0.0

4439 CONTINUE

DO 4444 CONSTR = 1, K - 1
  TEMP = TEMP + BF(CONSTR) * AF(K, CB(CONSTR))

4444 CONTINUE

IF(INEG(K) .EQ. 0) THEN
  DO 4450 CONSTR = 1, K - 1
    IF(AF(CONSTR, VT) .LE. 0.0) THEN
      CONSTR = CONSTR + 1
      AF(CONSTR, VT) = 0.0
    ELSE
      IF(AF(K, COL) .GT. 0.0) THEN
        AF(K, COL) = AF(K, COL) - 1.0
        CB(K) = VT
      ELSE
        IF(INEG(CONSTR) .EQ. 0) THEN
          CF(CONSTR + INDEXL - 1) = CF(CONSTR + INDEXL - 1) + BM
        ENDIF
      ENDIF
    ENDIF
  END DO 4450
  CONTINUE

4450 CONTINUE

DO 4460 COL = VT + 1, VT - 1
  AF(K, COL) = 0.0

4460 CONTINUE

DO 4477 CONSTR = 1, K
  IF(INEG(CONSTR) .LE. 0) THEN
    CF(CONSTR + INDEXL - 1) = CF(CONSTR + INDEXL - 1) + BM
  ENDIF

4477 CONTINUE

ELSEIF(INEG(K) .EQ. 1) THEN
  IF THE NEW CONSTRAINT WAS A "GREATER THAN", TWO COLUMNS
C ARE ADDED AND THE INDICES ARE ADJUSTED ACCORDINGLY.
VT=VT+2
NSEC=NSEC+1
NEC=NEC+1
DO 4480 COL=VT-1,VT
   CF(COL)=0.0
   DO 4470 CONSTR=1,K-1
      AF(CONSTR,COL)=0.0
4470    CONTINUE
4480    CONTINUE
4481 DO 4481 COL=VT-1,VT-2
   AF(K,CONSTR)=0.0
4481    CONTINUE
AF(K,VT-1)=-1.0
AF(K,VT)=1.0
CB(K)=VT
DO 4490 COL=1,VT
   CF(COL)=CF(COL)-AF(K,COL)*BM
4490    CONTINUE
2=2-BF(K)*BM
CF(VT)=BM
DO 4478 CONSTR=1,K-1
   IF(INEQ(CONSTR).NE.0) THEN
      CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)+BM
   ENDIF
4478    CONTINUE
ELSE IF(INEQ(K).EQ.2) THEN
C IF THE NEW CONSTRAINT IS AN "EQUALS", A SINGLE ARTIFICIAL
C VARIABLE IS Added.
VT=VT+1
NSEC=NSEC+1
DO 4405 COL=VT+1,VT
   AF(K,COL)=0.0
4405    CONTINUE
4415 DO 4415 CONSTR=1,K-1
   AF(CONSTR,VT)=0.0
4415    CONTINUE
AF(K,VT)=1.0
CB(K)=VT
DO 4425 COL=1,VT
   CF(COL)=CF(COL)-AF(K,COL)*BM
4425    CONTINUE
Z=Z-BF(K)*BM
DO 4479 CONSTR=1,K
   IF(INEQ(CONSTR).NE.0) THEN
      CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)+BM
   ENDIF
4479    CONTINUE
ELSE
   WRITE(1,'(A1)') CHAR(12)
   WRITE(1,'(/,/5x,"IMPROPER RESPONSE",/))
   GOTO 4441
408
C COEFFICIENTS IN THE NEW CONSTRAINT WHICH REPRESENT COLUMNS
C WITH VARIABLES IN THE BASIS ARE DRIVEN TO ZERO, AND THE FULL
C CONSTRAINT AND RHS ARE ADJUSTED ACCORDINGLY.
DO 4435 CONSTR=1,K-1
   TEMP=AF(K,CF(CONSTR))
   DO 4445 COL=1,VT
      AF(K,COL)=AF(K,COL)-AF(CONSTR,COL)*TEMP
   CONTINUE
4445 CONTINUE
   BF(K)=BF(K)-BF(CONSTR)*TEMP
4435 CONTINUE
ELSE
   WRITE(1,'(A1)',CHAR(12))
   WRITE(1,'(/,111,,''IMPROPER RESPONSE'',/))
   GOTO 4400
ENDIF
J=0
IFLAB(3)=0
C C IF ALL OBJECTIVE FUNCTION COEFFICIENTS AND ALL RHS ARE
C NON-NEGATIVE AND AN ARTIFICIAL VARIABLE IS IN THE BASIS AT
C A POSITIVE LEVEL, THEN THE PROBLEM IS INFEASIBLE
C DO 4465 CONSTR=1.K
      IF(BF(CONSTR) .LT. -.0001) THEN
         J=1
      ENDIF
4465 CONTINUE
DO 4475 COL=1,VT
      IF(CF(COL) .LT. -.0001) THEN
         J=1
      ENDIF
4475 CONTINUE
DO 4493 CONSTR=1,K
      IF(J.EQ.0.AND.CF(CONSTR) .GE. INDEK) THEN
         IF(BF(CONSTR) .GT. 0.0001) THEN
            IFLAG(3)=1
         ENDIF
      ENDIF
4493 CONTINUE
RETURN
END
SUBROUTINE SOLVE

USE: THIS SUBROUTINE ACCEPTS BASIC TABLEAUS FROM SUBROUTINES MULTCH AND ADDCON AND DIRECTIONS OTHER SUBROUTINES TO DETERMINE THE FINAL SOLUTION AND THEN DISPLAY AS DIRECTED.

CALLED BY: SUBROUTINE SELECT

CALLS: SUBROUTINE OPTB
        SUBROUTINE WORK
        SUBROUTINE TDISPL

VARIABLES:

USES: IFLAG(7), INFP, OPTS

MODIFIES: NONE

*USES UCHECK2 IN UNIT47.CODE OVERLAY

SUBROUTINE SOLVE

INTEGER PK, PR, OPTS, V, VT, CB
REAL A0, AF, B0, BF, CO, CF, Z, BM
CHARACTER SELOUT, FNTO
COMMON/P1/OPTS, KFA, PK, PR
COMMON/ONE/SELOUT, FN
COMMON/TAG/VT, INDEX6, INDEIX, INDEIX, INDEC, NLC, NCE, NCE(20), AMIN, BM
COMMON/THREE/INFP

INCLUDE CONVAR

SUBROUTINE OPTB IS CALLED TO DETERMINE THE CONDITION OF THE TABLEAU. IF THE TABLEAU IS EITHER UNBOUNDED OR INFEASIBLE, THE PROGRAM RETURNS TO THE MAIN MENU. IF IT IS NEITHER OF THESE BUT IS OPTIMAL, THE SUBROUTINE TDISPL IS CALLED TO DISPLAY THE RESULTS. OTHERWISE, SUBROUTINE WORK IS CALLED TO COMPLETE A BASIS CHANGE.

CALL OPTB
IF(OPTS .EQ. 1) OR .IFLAG(7) .EQ. 1) THEN
    IF(IFLAG(7) .EQ. 1) OR .INFP .EQ. 1) THEN
        RETURN
        ENDIF
    ENDIF
IF(OPTS .EQ. 1) THEN
    CALL TDISPL
    RETURN
ELSE
    CALL WORK
    GOTO 4500
ENDIF
RETURN
END
SUBROUTINE OPTB

INTEGER PK, PR, OPTS, V, VT, CB

COMMON/ONE/SELOUT, FNP10
COMMON/P./OPTS, XFA, PK, PR
COMMON/TWO/VT, INDEX, INDEXL, INDEXE, NGC, NLC, NEC, NES(20), MXMN, BM
COMMON/THREE/INFP, OPTS, IFLAG, INFP

INCLUDE COMVAR

IFLAG(4)=0
IFLAG(6)=0
IFLAG(7)=0
IFLAG(8)=0
IFLAG(9)=0
OPTS=0
INFP=0
GMES=0, 0
XFA=V+NGC-NLC+1

100 THE PIVOT COLUMN IS FOUND.
110 J=1, VT

IF (CF(J) .GE. GMES) THEN
   600 GOTO 130
ELSE
   GMES=CF(J)
   PK=J
   ENDIF

130 CONTINUE

C OPTIMALITY IS DETERMINED.
   IF (ABS(GMES) .LT. 0.0001) THEN
      OPTS=1
   ENDIF

C INFEASIBILITY IS DETERMINED.
   30 150 I=1, K
IF (CB(1) .LT. KFA) THEN
  IF (BF(1) .LT. -0.00001) THEN
    INFP=1
    ENDIF
  ELSEIF (BF(1) .LT. -0.00001) THEN
    GOTO 150
  ELSE
    INFP=1
    ENDIF
  CONTINUE
C THE LEAVING BASIC VARIABLE IS FOUND.
  IF (OPTS .EQ. 9) THEN
    SPR=10.0E8
    DO 190 I=1,K
    IF (AF(I,PK) .LE. .0001) THEN
      GOTO 190
    ELSEIF (BF(I)/AF(I,PK) .GE. SPR) THEN
      GOTO 190
    ELSE
      SPR=BF(I)/AF(I,PK)
      PR=:
      ENDIF
  CONTINUE
190  CONTINUE
C THE LEAVING BASIC VARIABLE IS FOUND.
  IF (OPTS .EQ. 9) THEN
    SPR=10.0E8
    DO 190 I=1,K
    IF (AF(I,PK) .LE. .0001) THEN
      GOTO 190
    ELSEIF (BF(I)/AF(I,PK) .GE. SPR) THEN
      GOTO 190
    ELSE
      SPR=BF(I)/AF(I,PK)
      PR=:
      ENDIF
  CONTINUE
ENDIF
GOTO 500
C DUAL PIVOTS ARE USED UNLESS A NEGATIVE OBJECTIVE COEFFICIENT IS FOUND.
300  DO 320 J=1,VT
      IF (CF(J) .LT. -0.00001) THEN
        GOTO 340
      ENDIF
320  CONTINUE
C THE PIVOT ROW IS FOUND.
  DO 340 I=1,K
  IF (BF(I) .GE. GNEB) THEN
    GOTO 340
  ELSE
    GNEB=BF(I)
    PR=I
    ENDIF
340  CONTINUE
  IF (ABS(GNEB) .LT. 0.0001) THEN
    OPTS=1
    GOTO 500
  ELSE
    SPR=10.0E8
    DO 370 J=1,VT
      DO 360 I=1,K
      IF (CB(I) .EQ. J) THEN
        ....
GOTO 370  
ENDIF  
CONTINUE  
IF( (AF(PR,J) .GE. -.0001) THEN  
GOTO 370  
ELSEIF (CF(J)/AF(PR,J) .LE. SPR) THEN  
GOTO 370  
ELSE  
SPR=CF(J)/AF(PR,J)  
PK=J  
ENDIF  
370  CONTINUE  
ENDIF  
IF(SPR .LE. -10.E6) THEN  
IFLAGM=1  
ENDIF  
500  IF(OPTS .EQ. 1) THEN  
IF(INFP .EQ. 1) THEN  
GOTO 600  
ENDIF  
DO 540 J=1,VT  
IFLAGB=0  
DO 520 I=1,K  
IF(CB(I) .EQ. Ji) THEN  
IFLAGB=1  
ENDIF  
520  CONTINUE  
IF( (IFLAGB) .EQ. 0) THEN  
IF(RABS(CF(J)) .LT. .0001 ) THEN  
IFLAG(4)=1  
ENDIF  
ENDIF  
540  CONTINUE  
ENDIF  
IF( (IFLAG(7) .EQ. 1) ) THEN  
GOTO 600  
ENDIF  
DO 560 I=1,K  
IF(ABS(IF(I)) .LE. .0601) THEN  
IFLAG(6)=1  
ENDIF  
560  CONTINUE  
C  THE CONDITION OF THE TABLEAU IS PRINTED.  
600  IF (SELGUT .EQ. 'S' OR SELGUT .EQ. 'B') THEN  
IF (OPTS .EQ. 1 OR IFLAG(7) .EQ. 1) THEN  
WRITE(1,'(10X,,"FINAL TABLEAU = ")')  
WRITE(1,'("INFEASIBLE")')  
ELSEIF (IFLAG(7) .EQ. 1) THEN  
WRITE(1,'("UNBOUNDED")')  
ELSE  
WRITE(1,'("OPTIMAL")')  
ENDIF  
ENDIF
ENDIF
IF(FLAG(6) .EQ. 1) THEN
  WRITE(1,'(26X,"DEGENERATE")')
ENDIF
IF(OPTS .EQ. 1 .AND. FLAG(4) .EQ. 1) THEN
  WRITE(1,'(5X,"MULTIPLE OPTIMAL SOLUTIONS EXIST")')
ENDIF
PAUSE
ENDIF
ENDIF
IF(SELOUT .EQ. 'P' .OR. SELOUT .EQ. 'B') THEN
  IF(OPTS .EQ. 1 .OR. IFLAG(7) .EQ. 1) THEN
    WRITE(6,'(///,10X,"FINAL TABLEAU - ",9)')
    IF(INFP .EQ. 1) THEN
      WRITE(6,"(""INFEASIBLE")")
    ELSEIF(FLAG(7) .EQ. 1) THEN
      WRITE(6,"(""UNBOUNDED")")
    ELSE
      WRITE(6,"(""OPTIMAL")")
    ENDIF
    IF(FLAG(6) .EQ. 1) THEN
      WRITE(6,'(26X,"DEGENERATE")')
    ENDIF
    IF(OPTS .EQ. 1 .AND. FLAG(4) .EQ. 1) THEN
      WRITE(6,'(5X,"MULTIPLE OPTIMAL SOLUTIONS EXIST")')
    ENDIF
    WRITE(6,'(///)')
  ENDIF
ENDIF
IF(FLAG(9)=1
RETURN
END
SUBROUTINE WORK

INTEGER PK, PR, OPTS, V, VT, CF
REAL PELE

COMMON/PTS, V, CF
COMMON/TMB/VT, VINDEX, INDEXL, INDEXE, NG, MLC, MEC, MEG(20), NNN, MM

*INCLUDE CONVAR

THIS SUBROUTINE PERFORMS A BASIS CHANGE WITH A PIVOT ROW AND PIVOT COLUMN DETERMINED BY SUBROUTINE OPTS.

PELE = AF(PR, PK)

DO 200 J = 1, VT
   AF(PR, J) = AF(PR, J) / PELE

200 CONTINUE

BF(PR) = BF(PR) / PELE
CB(PR) = PK

DO 300 I = 1, K
   IF(I .EQ. PR) THEN
      GOTO 300
   ENDIF
   HOLD = AF(I, PK)
   DO 250 J = 1, VT
      AF(I, J) = AF(I, J) - HOLD * AF(PR, J)
   CONTINUE
   BF(I) = BF(I) - HOLD * BF(PR)

250 CONTINUE

BF(PR) = BF(PR)

300 CONTINUE

HOLD = CF(PR)

DO 350 J = 1, VT
   CF(J) = CF(J) - HOLD * AF(PR, J)

350 CONTINUE

RETURN
SUBROUTINE TDISPL
CHARACTER P(10),OBJN#10,SELOUT,FM#10
INTEGER PK,P#OPTS,V,VI,CR,DNL,T,FMT
COMMON/ONE/SELOUT,FM
COMMON/TWO/VT,INDEX,INDEXL,INDEXE,NEC,NLC,NE#(20),MM#NM,BM
COMMON/P1/OPTS,K#A,P,K,R
$INCLUDE COMVAR
THIS SUBROUTINE DISPLAYS THE FINAL TABLEAU IN VARIOUS FORMATS ON
SCREEN, PRINTER, OR BOTH SIMULTANEOUSLY.
WRITE(1,'(A1)') CHAR(12)
100 WRITE(1,'(S/,A1,I1,1,D9.1)')
WRITE(1,'(I10)' ) E FORMAT ''
WRITE(1,'(/,16X)''OR'')
WRITE(1,'(/,10X)''I)
READ(1,280) P(1)
CALL CHECK2(P,I,2,INVAL,FMT)
IF(INVAL.EQ.1) THEN
WRITE(1,'(A1)') CHAR(12)
WRITE(1,'(S/,X)') ''IMPROPER RESPONSE,PLEASE REENTER''
GOTO 100
ENDIF
WRITE(1,'(A1)') CHAR(12)
FORMAT(A)
T=(VT5)*1
IF(SELOUT.EQ.'S' OR SELOUT.EQ.'B') THEN
DO 470 M=1,T
WRITE(1,'(15X,9)')
DO 290 J=(#M#)-4,#M#5
IF(J .LT. VT) THEN
GOTO 290
ENDIF
WRITE(1,280)
280 FORMAT(5X,'X(',I3,')',3X,0)
290 CONTINUE
IF(T .EQ. I OR N .EQ. T) THEN
WRITE(1,300)
   FORMAT(1X,'RHS')
ELSE
   WRITE(1,(''))
ENDIF
WRITE(1,('OBJ FUNCTION'))
DO 320 J=(N5)-4,N5
   IF(J.GT. VT)THEN
      GOTO 320
   ENDIF
   IF(FMT.EQ. 1)THEN
      WRITE(1,('1PE12.5,1X,'))CF(J)
   ELSE
      WRITE(1,('12.5,1X,'))CF(J)
   ENDIF
CONTINUE
320 DO 320 Jz(N15)-4,N15
   IF(J .ST. VT)THEN
      010 320
   IF(J .ED. N) THEN
      010 370
   ELSE
      WRITE(1,400)
   ENDIF
   WRITE(1,('CN NAME VAR'.2X,$5(1X,'')))
DO 400 L=1,K
   WRITE(1,('12,7X,'))L
   WRITE(1,('1I,1I,'))CB(L)
   DO 370 J=(N5)-4,N5
      IF(J.GT. VT)THEN
         GOTO 370
      ENDIF
      IF(FMT.EQ. 1)THEN
         WRITE(1,('1PE12.5,1X,'))AF(L,J)
      ELSE
         WRITE(1,('12.5,1X,'))AF(L,J)
      ENDIF
CONTINUE
370 DO 370 Jz(N15)-4,N15
   IF(T .EQ. 1 OR N .EQ. T) THEN
      010 370
   IF(FMT.EQ. 1) THEN
      WRITE(1,('A1,1I,1PE12.5')$9F(2),BF(L)
   ELSE
      WRITE(1,('A1,1I,F12.5')$9F(2),BF(L)
   ELSE
      WRITE(1,(''))
   ENDIF
   ELSE
      WRITE(1,(''))
   ENDIF
CONTINUE
400 PAUSE
WRITE(1,110)CHAR(12)
CONTINUE
WRITE(1,110)CHAR(12)
DO 580 I=1,K
IF(FMT.EQ.1)THEN
  WRITE(1,"(1X,','I10,'','I2,'') = ',IP12.5')CB(I),BF(I)
ELSE
  WRITE(1,"(1X,','I10,'','I2,'') = ',F12.5')CB(I),BF(I)
ENDIF
CONTINUE
580 CONTINUE
IF(FMT.EQ.1)THEN
  WRITE(1,"(1X,','Z = ',IP12.5')")Z
ELSE
  WRITE(1,"(1X,','Z = ',F12.5')")Z
ENDIF
ENDIF
ENDIF
IF(SELOUT.EQ.'F'.OR.SELOUT.EQ.'B')THEN
  DO 1470 N=1,T
  WRITE(6,'((5X,','I10,''))')N
  DO 1290 Jx(N)=4.N5
  IF(J GT .VT)THEN
    GOTO 1290
  ENDIF
  WRITE(6,1280)
  WRITE(6,'(14X,'A
  1280 FORMAT(5X,'(',2,'')','I2,',')')A
  CONTINUE
  IF(T EQ.1 .OR. N EQ. T)THEN
    WRITE(6,1300)
  ELSE
    WRITE(6,'(''OBJ
  1300 FORMAT(6X,'RHS')
  ELSE
    WRITE(6,'('' ''')
  ENDIF
  WRITE(6,'(''OBJ FUNCTION'',1X,9)')
  DO 1320 J=(N5)-4.N5
  IF(J GT .VT)THEN
    GOTO 1320
  ENDIF
  IF(FMT.EQ.1)THEN
    WRITE(6,'(''= ''',IP12.5,1X)')Z
  ELSE
    WRITE(6,'(''= ''',F12.5,1X)')Z
  ENDIF
  CONTINUE
  IF(T .EQ.1 .OR. N .EQ. T)THEN
    IF(FMT.EQ.1)THEN
      WRITE(6,'(''= ''',IP12.5,1X)')Z
    ELSE
      WRITE(6,'(''= ''',F12.5,1X)')Z
    ENDIF
  ELSE
    WRITE(6,'('' ''')
  ENDIF
  WRITE(6,'(''CH NAME VAR''',2X,55(''''))')
DO 1400 I=1,K
WRITE(6,'((12,1X,9)'')L
WRITE(6,'((11,12.1X,9)'')CB(L)
DO 1370 J=(H5)-4,H5
IF(J .GT. VI)THEN
GOTO 1370
ENDIF
IF(FMT .EQ. 1)THEN
WRITE(6,'(''P12.5,1X,'')AF(L,J)
ELSE
WRITE(6,'(''F12.5,1X,9)'')AF(L,J)
ENDIF
1370 CONTINUE
IF(T .EQ. I .OR. N .EQ. 1)THEN
  P(2)=t'
  IF(FMT .EQ. 1)THEN
    WRITE(6,'((A1,11,1PE12.5)')P(2),BF(L)
  ELSE
    WRITE(6,'((A1,11,F12.5)')P(2),BF(L)
  ENDIF
ELSE
  WRITE(6,'(''**''')
ENDIF
1400 CONTINUE
WRITE(6,'/**/**')
1470 CONTINUE
DO 1580 J=1,K
IF(FMT .EQ. 1)THEN
WRITE(6,'(''I10,1I10,1I12,1I24,1I24,1I24,1I24)'')CB(I),BF(I)
ELSE
WRITE(6,'(''I10,1I12,1I24,1I24,1I24,1I24)'')CB(I),BF(I)
ENDIF
1580 CONTINUE
IF(FMT .EQ. 1)THEN
WRITE(6,'((14X,'')I = '' ,1PE12.5)'')I
ELSE
WRITE(6,'((14X,'')I = '' ,F12.5)'')I
ENDIF
WRITE(6,'((A1)')CHAR(12)
ENDIF
PAUSE
RETURN
END
MODULE 4 UNIT47
UNIT USES: NONE
C
SUBROUTINE CHECK2
USE: THIS SUBROUTINE ACCEPTS KEYBOARD NUMERIC INPUTS AS "CHARACTERS" AND RETURNS THE INTEGER EQUIVALENT IF THE INPUT IS OF THE CORRECT TYPE.
C CALLED BY: SUBROUTINE MULCHG
SUBROUTINE ADDCON
CALLS: NONE
C
VARIABLES:
USES: E,D,HVAL
MODIFIES: INVY4,IN520
C
SUBROUTINE CHECK2(E,D,HVAL,INVY4,IN520)
CHARACTER ALLOW(II) E(D)10
INTEGER D,HVAL
DATA ALLOW/'1','2','3','4','5','6','7','8','9','0',' '/
IN520=0
INVY4=0
DO 4710 I=1,II
C EACH CHARACTER IS CHECKED FOR VALIDITY.
4710 CONTINUE
C
DO 4790 J=1,10
IF(E(I) .EQ. ALLOW(J)) THEN
C SPACES ARE IGNORED.
GOTO 4710
ELSE IF(E(I) .EQ. ALLOW(J)) THEN
IN520=IN520*10+ICHAR(E(I))-48
GOTO 4710
ELSE IF(J .EQ. 10) THEN
C IF THE CHARACTER DOES NOT FIT ONE OF THE ABOVE C DESCRIPTIONS, AN INVALID FLAG IS SET, AND THE PROGRAM C RETURNS TO GET A NEW INPUT.
INVY4=1
IN520=0
RETURN
ENDIF
4790 CONTINUE
4710 CONTINUE
IF(IN520 .EQ. 0 .OR. IN520 .GT. HVAL) THEN
INVY4=1
IN520=0

420
SUBROUTINE CHECK(E, INVAL, RNEW)
CHARACTER ALLOW(14), ALLOW(14)
REAL N
INTEGER DECIMA
DATA ALLOW/'1','2','3','4','5','6','7','8','9','0','+','-'.'/
RENEW=0.0
N=1
INVAL=0
DECIMA=0
NEGAT=0
DO 4740 I=1,10
  IF(E(I) .EQ. ALLOW(14))THEN
    DO 4740
  ENDIF
  DO 4730 J=1,13
  IF(E(I) .EQ. ALLOW(J))THEN
    IF VALID SPECIAL CHARACTERS ARE PRESENT, FLAGS ARE SET
    ACCORDINGLY. IF THE CHARACTER IS A NUMBER, THE NUMBER
    WHICH IS BEING ASSEMBLED IS MULTIPLIED BY 10, AND THE VALUE
    OF THE NEW CHARACTER IS ADDED.
    IF DECIMA .EQ. 1 THEN
      GOTO 4710
    ELSEIF(E(I) .EQ. '+')THEN
      NEGAT=1
      GOTO 4740
    ELSEIF(E(I) .EQ. '-')THEN
      DECIMA=1
      GOTO 4740
    ELSE
      RNEW=RENEW+10+ICAR(E(I))-48
      GOTO 4720
  ENDIF
4710  RNEW=RNEW+(ICHAR(E(I))-48)*N
M=NB.1
GOTO 4720
ELSEIF(J.EQ.13)THEN
  NVALL=1
  RNEW=0.0
  RETURN
ENDIF
GOTO 4730
4720  J=13
4730  CONTINUE
4740  CONTINUE
C IF THE NUMBER IS NEGATIVE, THE VALUE OF THE ASSEMBLED REAL NUMBER
C IS SWITCHED.
IF(NEGAT.EQ.1)THEN
  RNEW=-RNEW
ENDIF
RETURN
END
SUBROUTINE RETRIV

INTEGER I,V,K,J,COL,ROM,IFLAG,INEQ,VT,EB,CONSTR,NEB
.,SLACK,ARVAN
REAL AG,AF,BO,BF,CO,CF,Z,TEMP(120,20),TEMPC(20),BM,
.TEMP(20)

CHARACTER FN#20,FN#10,SELIMP#10#1,SELOUT
COMMON/ONE/SELOUT,FN

COMMON/TWO/VT,INDEX,EINDEX,INDEXE,NGC,NLC,NEG(20),MINN,BM

$ INCLUDE CONVAR

WRITE(1,'(A1)') CHAR(12)
WRITE(1,'(5(/),5X,'ENSURE DISK LP2: IS AVAILABLE '','S(/)')')
PAUSE

THE NAME OF THE FILE WITH THE DATA IS READ.
OPEN(7,FILE='LP2:LPDATAW',STATUS='OLD',FORM='UNFORMATTED') READ(7) FN
WRITE(1,'(A1)') CHAR(12)

THE USER IS ALLOWED TO CHANGE THE DATA FILE NAME IF DESIRED.

WRITE(1,'(5(/),0X,'THE CURRENT DATA FILE IS ''/''/4X,A10,///)FN
WRITE(1,'(5(/),0X,'YOU WISH TO USE THIS TABLEAU'',S(/)')')
READ(5,'(A1)') SELIMP(1)

IF SELIMP(1).EQ. 'N' THEN WRITE(1,'(A1)') CHAR(12)
WRITE(1,'(5(/),2X,'PLEASE ENTER THE NEW VOLUME:FILENAME''/')')
WRITE(1,'(5X,'EB. LP2:TEST1'/'///)')
READ(1,'(A10)') FN

REWRITE 7
WRITE(7,FN

ELSEIF (SELIMP(1).NE. 'Y') THEN WRITE(1,'(A1)') CHAR(12)
WRITE(1,'(5X,'INVALID RESPONSE, TYPE ''Y'' OR ''N'' "
',///)')
GOTO 4800
ENDIF
CLOSE(7,STATUS='KEEP')
WRITE(1,'(A1)') CHAR(12)
WRITE(1,'(5I1,6X,E14.9)') "ENSURE THE DISK ADDRESSED"
WRITE(1,'(3I1,5X,A10)') FN
WRITE(1,'(/,12X,7X,14X)') "IS AVAILABLE"/
PAUSE
C  THE DATA FILE IS READ.
OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
READ(3)PN,HXM,X,V,IFLAG(5)
DO 4810 I=1,K
   READ(3) INEO(I),BO(I)
   DO 4820 J=1,V
      READ(3) AO(I,J)
   4820 CONTINUE
4810 CONTINUE
DO 4830 J=1,V
   READ(3) CO(J)
4830 CONTINUE
READ(3)IFLAG(IO),VT
DO 4840 I=1,K
   READ(3) BT(I),CB(I)
   DO 4850 J=1,VT
      READ(3) AT(I,J)
   4850 CONTINUE
4840 CONTINUE
DO 4860 J=1,VT
   READ(3) CF(J)
4860 CONTINUE
READ(3) Z
CLOSE(3,STATUS='KEEP')
WRITE(1,'(A1)') CHAR(12)
WRITE(1,'(5I1,6X,E14.9)') "ENSURE LP2: IS AVAILABLE"/
PAUSE
NEC=0
NLC=0
NEC=0
DO 4861 CONSTR=1,K
   NES(CONSTR)=0
4861 CONTINUE
C  EACH CONSTRAINT IS ALTERED IF THE PROBLEM WAS A MINIMIZATION AND
C  THE CONSTRAINT WAS A "GREATER THAN" OR IF THE ORIGINAL CONSTRAINT
C  HAD A NEGATIVE RIGHT-HANDED SIDE (NOT MINIMIZATION).
DO 4870 CONSTR=1,K
   IF(IFLAG(IO) .EQ.1) THEN
      IF(INEQ(CONSTR) .EQ.1) THEN
         INEQ(CONSTR) = 0
         NES(CONSTR) = 1
         BD(CONSTR)=BD(CONSTR)
        DO 4805 COL=1,V
            EO(CONSTR,COL)=AO(CONSTR,COL)
        4805 CONTINUE
      ENDIF
      ELSE
        ELSE
IF (BD(CONSTR) .LT. -0.00001 ) THEN
    NEG(CONSTR)=1
    BD(CONSTR) = -BD(CONSTR)
    DO 4200 J=1,V
        A0(CONSTR,J) = -A0(CONSTR,J)
    CONTINUE
4200
    IF (INEQ(CONSTR) .EQ. 0) THEN
        INEQ(CONSTR) = 1
    ELSEIF (INEQ(CONSTR) .GT. 0) THEN
        INEQ(CONSTR) = 0
    ENDIF
ENDIF
ENDIF
C DETERMINE INDICES
IF (INEQ(CONSTR) .EQ. 0) THEN
    NLC = NLC +1
ELSEIF (INEQ(CONSTR) .GT. 0) THEN
    NBC = NBC +1
ENDIF
4870
CONTINUE
INDEX6 = V + 1
INDEXL = INDEX + NBC
INDEXE = INDEXL + NLC
NLC = NLC - NBC - NLC
SLACK = 0
ARTVAR = 0
DO 111 CONSTR=1,K
    TEMP(CONSTR)=0
111 CONTINUE
DO 4899 CONSTR=1,K
    C FIND THE COLUMN ASSOCIATED WITH THE CONSTRAINT
    IF (INEQ(CONSTR) .EQ. 0) THEN
        COL=INDEXL + SLACK
        SLACK = SLACK + 1
    ELSE
        COL=INDEXE + ARTVAR
        ARTVAR = ARTVAR + 1
    ENDIF
    C REARRANGE THE LAST K COLUMNS TO PUT THEN IN THE INVERSE ORDER
    C TEMPORARILY HOLD THE VALUES OF C(J) AND A(i,J) IN TEMP UNTIL
    C THEY ARE SORTED OUT
    DO 4890 ROW = 1,K
        TEMP(ROW,CONSTR)=AF(ROW,COL)
        TEMPC(CONSTR)=CF(COL)
        IF (CD(ROW).EQ.COL) THEN
            TEMP(ROW)=CONSTR+INDEXL-1
        ENDIF
4890 CONTINUE
4899 CONTINUE
C
C PUT THE VALUES WHICH WERE HELD IN TEMP BACK INTO THE CORRECTED
C C(J) AND A(I,J) COLUMNS
C
DO 4890 COL=1,K
   CF(COL+INDEXL-1)=TEMP(COL)
   DC 4897 CONSTR=1,K
   AF(CONSTR,COL+INDEXL-1)=TEMPA(CONSTR,COL)
   IF(TEMP(CONSTR).GT.0) THEN
      CB(CONSTR)=TEMP(CONSTR)
   ENDIF
4897 CONTINUE
4898 CONTINUE
BN=0.0
DO 4866 J=1,V
   IF(ABS(CO(J)) .GT. BN) THEN
      BM=ABS(CO(J))
   ENDIF
4866 CONTINUE
BN=INT(BM)+10
IF(BM .LT. 10) THEN
   BM=10.0
ENDIF
DO 4877 CONSTR=1,K
   IF(INE(CONSTR) .NE. 0) THEN
      CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)-BM
   ENDIF
4877 CONTINUE
RETURN
END
COMMON/VARI/AG(20, 20), AF(20, 60), BD(20), BF(20), CD(20), CF(60), Z
   K, V, IFLAG(10), INEG(20), CB(20)
Theodore R. E. Fraley was born in Greenville, Ohio on 13 April, 1943. After graduation from high school in Atascadero, California, he attended California State Polytechnic College before joining the Air Force in 1964. He received a degree in Aeronautical Engineering from Oklahoma State University in 1968. Following pilot training, he has been involved in tactical flight operations. He entered the Air Force Institute of Technology in June, 1981.

Permanent address: 6843 Santa Lucia
Atascadero, California
93422

Dale A. Kem was born in Huntington, Indiana on 25 February, 1953. He graduated from high school in Huntington, Indiana in 1971 and attended Purdue University from which he received a Bachelor of Chemical Engineering degree in 1975. Following graduation, he received a commission in the U.S. Army. He was assigned to Fort Sill, Oklahoma until 1977. He attended the Defense Language Institute of Monterey, California for one year of language training and subsequently was assigned to TUSLOG, Group 67, Turkey. Upon his return from overseas, he attended the Ordnance Advance Course at Redstone Arsenal, Alabama and in June, 1981 entered the Air Force Institute of Technology.

Permanent address: 9065 S. 900 W 35
Lafontaine, Indiana
46940
# Abstract

Linear programming is an analytical technique used in decision analysis. This paper describes the development and use of a highly interactive, non-programmer oriented, linear programming software package implemented on a microcomputer. This software, written in FORTRAN and supported by the UCSD Pascal Operating System, has allowed increased portability while providing the capability of solving moderate-sized LP models. Also available are extensive postoptimal sensitivity analysis capabilities.
20. The modularly implemented package provides interactive, instructional sessions with user input LP models. The user is guided through tableau formulation and pivot element selection to an optimal solution by a series of option displays and user selections. This module also provides instructors the ability to rapidly demonstrate the application of the simplex algorithm. A separate module provides a more rapid problem solution with minimal interaction. Options allow either primal or dual problem solution with screen-oriented output to either a monitor or printer. The sensitivity analysis capabilities include right-hand-side, cost coefficient, and constraint ranging. Also provided is the ability to add constraints and variables to the original model.