LPVAX - A THREE-DIMENSIONAL LINEAR PROGRAMMING GRAPHICS PACKAGE ON THE INTERGRAPH-VAX SYSTEM

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

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

by
Linda A. Humphreys
November, 1986
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PROBLEM DEFINITION</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>PROBLEM DEVELOPMENT</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>BACKGROUND</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>GENERAL CONCEPTS</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>SPECIFIC PROGRAMMING CONSIDERATIONS</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>SYSTEM HARDWARE DESCRIPTION</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>THE COMPUTER SYSTEM</td>
<td>10</td>
</tr>
<tr>
<td>3.2</td>
<td>THE WORKSTATION</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>The Cursor</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>The Keyboard</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>The Menu Tablet</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>ENVIRONMENTS</td>
<td>20</td>
</tr>
<tr>
<td>3.4</td>
<td>DESIGN FILE</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Design levels</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Design Block</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Update Command</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Dynamic Volume</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Move and Scale</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Hidden Line Removal</td>
<td>31</td>
</tr>
<tr>
<td>3.5</td>
<td>DFPI</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>LPVAX PROGRAM</td>
<td>39</td>
</tr>
<tr>
<td>4.1</td>
<td>LPVAX</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Enter the Equations Option</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Editing the Equations</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Simplex Algorithm</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>Graphical Solution</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Exit</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>LPROG</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>DPLT</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>Constraint Planes</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>FNPT Subroutine</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>ORDPT Subroutine</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Solid Evolution</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>Optimal Point</td>
<td>68</td>
</tr>
<tr>
<td>4.4</td>
<td>OTHER CONSIDERATIONS</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Graphing the Logical Operator</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Drawing the Feasible Region</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Negative Constraint Coefficients</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>CONCLUSION and RECOMMENDATIONS</td>
<td>77</td>
</tr>
<tr>
<td>6</td>
<td>REFERENCES</td>
<td>82</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Figure 1</td>
<td>System Hardware</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Intergraph Workstation</td>
<td>13</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Cursor</td>
<td>15</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Keyboard</td>
<td>17</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Screen Views</td>
<td>25</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Design File Block</td>
<td>26</td>
</tr>
<tr>
<td>Figure 7</td>
<td>UPDATE Command</td>
<td>28</td>
</tr>
<tr>
<td>Figure 8</td>
<td>DYNAMIC VOLUME Command</td>
<td>28</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Clipping Plane</td>
<td>33</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Hidden Line Menu</td>
<td>35</td>
</tr>
<tr>
<td>Figure 11</td>
<td>LPVAX Master Menu</td>
<td>40</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Constraints</td>
<td>44</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Current Problem</td>
<td>46</td>
</tr>
<tr>
<td>Figure 14</td>
<td>LPVAX Edit Menu</td>
<td>46</td>
</tr>
<tr>
<td>Figure 15-A</td>
<td>Constraint Planes</td>
<td>59</td>
</tr>
<tr>
<td>Figure 15-B</td>
<td>Axial Point Connection</td>
<td>59</td>
</tr>
<tr>
<td>Figure 15-C</td>
<td>Two Axial Points and One Constraint</td>
<td>60</td>
</tr>
<tr>
<td>Figure 15-D</td>
<td>Same Two Intersecting Planes</td>
<td>60</td>
</tr>
<tr>
<td>Figure 15-E</td>
<td>Axial Point, Boundary Plane, Shared Plane Connection</td>
<td>61</td>
</tr>
<tr>
<td>Figure 15-F</td>
<td>Final Solution Solid</td>
<td>61</td>
</tr>
<tr>
<td>Figure 16-A</td>
<td>Three Constraint Planes</td>
<td>62</td>
</tr>
<tr>
<td>Figure 16-B</td>
<td>Point Connection with Non-boundary</td>
<td>62</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Connection Matrix</td>
<td>64</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Two Dimensional Graph</td>
<td>69</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Logical Operator Direction</td>
<td>71</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Two Constraint Coefficients are Zero</td>
<td>73</td>
</tr>
<tr>
<td>Figure 21</td>
<td>One Negative Coefficient</td>
<td>75</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Two Negative Coefficients</td>
<td>76</td>
</tr>
</tbody>
</table>
Chapter I

PROBLEM DEFINITION

Linear programming is a subject taught in both the mathematics and engineering disciplines. In industrial engineering, linear programming is a useful tool in solving a wide range of problems such as personnel assignment, production scheduling, transportation problems, and economic analysis. It was suggested that a software package that graphically demonstrated simple linear programming problems would be very beneficial in aiding students learning the fundamental concepts.

Software packages used as teaching aids are not uncommon. The term given to software teaching aids is software infusion (SI). It is believed that utilizing educational software is a useful way to "clarify concepts learned in the class and allow the user to see variations of problems which would otherwise require involved calculations" (Schiffman, 1986). SI is not meant to replace classroom instruction.

If a software package is to clarify concepts in linear programming, a suitable method for solving the linear programming problems must be employed. Looking through almost any linear programming text book, one would see the graphical method of solving LP problems as the first method presented. As stated by James P. Ignizio
(1982) "It should not be inferred that graphical analysis is a practical approach to linear programming. It is, however, a superb teaching tool and visual aid in linear programming." He goes on to say the graphical solution provides students with

1) A concept of processes used for larger problems
2) A clearer visual definition of many of the terms and concepts utilized

As evidence of the importance of graphics in understanding linear programming, refer to the paper written by Dr P.D. Martins (1984). Martins has investigated the advantages of solving multidimensional problems graphically. To solve multidimensional problems in two dimensional space, he chooses two parameters as his window and projects the other constraints through this window. The result is a freeze frame graph with the parameters forming lines through the window. He concludes that a graphical approach offers:

1) A congenial way to aid the solution of small to medium design problems.
2) A better understanding of the design relationships.
3) A means of interpreting the results with little specialized mathematical knowledge for its application.

A three dimensional graphics package would not only have
these benefits, it does not have the freeze frame effect of windowing. The 3-D problem is placed in a 3-D window.

For further clarification of linear programming a second method for the package was needed. The simplex method, which is generally taught after the graphical method, was included. This method utilizes the concepts presented in the graphical method and applies them to larger more complex problems. The concepts of duality and sensitivity are generally presented within the simplex method. Although duality is presented strictly with the simplex method, sensitivity analysis is not. Sensitivity analysis is suggested during the presentation of the graphical method, but due to the recalculations of live analysis and feasible points it seldom gets the attention it deserves.

LPVAX (Linear Program on the VAX), is the 3-dimensional linear programming package developed. It offers flexibility to explore various LP concepts. It will help the students understand both the simplex and graphical methods of solving LP programs. This thesis describes the design and development of the LPVAX program. The benefits and limitations of the Intergraph system used for this package and a brief discussion on alternative methods are also included.
Chapter II
PROBLEM DEVELOPMENT

2.1 BACKGROUND

Very few programs have been written for LP graphics and these were two dimensional (Bevis, 1985). Bevis describes not only his two dimensional graphics program but another LP graphics package written by Bhagavan (1982). These programs, which were written for microcomputers, differ slightly. Bhagavan's program offers an LP tutorial along with the graphics. Bevis's program plots the solution to the problem and also graphs the objective function line which is moveable across the feasible solution space.

A three-dimensional linear programming graphics package would add a concept the other two packages cannot show. That is the concept of constraint and objective equations forming three-dimensional areas instead of lines. The feasible region to these problems are generally solids instead of planes and will be referred to as solution solids. Some texts will show the graphical solutions to three dimensional problems (Blanchard and Fabrycky, 1981; and Kim, 1971), although this is not common. It is very difficult to demonstrate a three dimensional problem on a two dimensional sheet of paper. It is not always possible to show each constraint plane since it may obstruct the solution solid view. There may
also be contours to the solid that are hidden behind the solution solid. This makes a three dimensional graphics package with rotational capabilities very attractive.

2.2 GENERAL CONCERNS

In designing a software package for any problem, human factor considerations must be taken into account. J.D. Foley and A. Van Dam (Foley and Van Dam, 1982) compiled a list of five key human factor considerations which are "rather obvious but often overlooked." Their list is as follows:

1) Provide simple consistent instructions.

2) Do not overload the user with too many different options and styles for communicating with the program.

3) Prompt the novice user at each stage of interaction.

4) Give appropriate feedback to the user.

5) Allow graceful recovery from mistakes.

They also add that the dialogue should be conversational language with natural grammar.

In an article by J.D. Foley and Victor Wallace (1972) the five principles are expanded. A good program should make the user unaware of the program itself and it should also allow flexibility for the individual user. It should minimize user frustration (i.e. allow corrections of mistakes). The system should not induce user panic, a
situation often caused by long delays between input and feedback. The ergonomics of the workstation is also viewed as important; minimal glare, large work area, right and left hand user flexibility, and comfortable chairs. The graphics displays should also be accompanied by a menu. The user should not have to switch back and forth between the menu and the graphics display.

The LPVAX program and the Intergraph-VAX system meet nearly all of these criteria. The LPVAX program prompts the user for each piece of required information with a short statement regarding how it expects the information to be entered. It has only two menus: the main menu and the edit menu accessed through the main menu. The amount of information the user has to supply is kept to a minimum. There are also checks throughout the program that will not allow the user to enter the wrong values such as entering a number where a logical operator belongs or allow alphabetic characters where numeric characters are to be entered. The graphical representation of the problem is placed in an external file called a design file. The user will have almost instantaneous feedback for all responses except when the LPVAX program is accessing the design file, a slight delay will occur here. A message informing the user of this delay will appear on the screen at this point.

The design file was designed to hold up to four problems at one time. The purpose of this was to allow the user to:
1) make slight variations of the same problem and to compare the overall results
2) view the design file for more than one problem without having to exit from the LPVAX program after each problem
3) save critical disc space

The workstation consists of a large work surface, keyboard, and cursor. This station easily accommodates right or left handed users. Both the work surface and chair are adjustable. The graphics design menu is located on the surface of the design table and a table of contents for the design file is listed on the screen. Unfortunately, glare from overhead lights makes certain color monitor lines hard to see.

2.3 SPECIFIC PROGRAMMING CONSIDERATIONS

Based on the five criteria, the requirements for the LPVAX program are as follows:

PROBLEM ENTRY:

1) A logical straightforward means of entering the initial problem
2) The ability to see the entire problem before processing it
3) The ability to make changes in the problem
4) The ability to easily start an entirely new problem
5) Explanatory prompts at each user response
SIMPLEX ALGORITHM CRITERIA:

1) The user should be able to see the initial tableau.
2) The user should have the option of seeing each tableau after each iteration.
3) The final tableau should be displayed.
4) The X, Y, Z variables should be displayed.
5) The optimal value should be displayed.
6) The user should have the option of seeing the solution to the dual.
7) The user should have the option of seeing the solution to the sensitivity analysis.

GRAPHICAL SOLUTION CRITERIA:

1) Should graph the feasible region to a 3-D problem or a 2-D problem
2) Should graph the optimal solution point
3) Should graph each constraint plane and make it distinguishable from the others
4) Should graph the evolution of the solution solid
5) Should graph the objective function plane
6) The objective plane should be moveable through the solution solid
7) Shading should be possible
8) The user should be able to rotate the graphs to see all sides of the solid and planes
9) The user should be able to see variations of the same problem
10) The user should have complete control over the objects being viewed

11) The graphics should be simple to use

SYSTEM CRITERIA:

1) Should be accessible to the user
2) Should be easy to use
3) Should require little prior knowledge
4) Should have a fast processing time
5) Should hold several designs in one file
6) Should be flexible

To explain how these criteria were met, the next two chapters will describe the system and program in detail.
Chapter III
SYSTEM HARDWARE DESCRIPTION

The first step in designing this linear programming package was deciding which computer system to use. All candidates must have graphics capabilities and preferably color graphics capabilities. Possible choices were the HP1000 mini computer, the IBM PC, and the Intergraph. The Intergraph was chosen for its full three-dimensional graphics capabilities, its large selection of built-in graphics functions, and its user flexibility. The Intergraph offers 256 different color shades, 63 design file levels, and is Fortran compatible through the VAX/VMS system.

3.1 THE COMPUTER SYSTEM

There are two separate computer systems involved when using the LPVAX program. These systems are the IGDS (Interactive Graphics Design System) and the VAX-11/750 VMS system. The IDGS controls the design file manipulations and design file storage. Whenever a graphics command is chosen the IGDS references a library of standard design elements. The graphs are stored as files on a magnetic disk. The VAX/VMS system compiles and executes Fortran programs. The DFPI (Design File Processor Input) serves as
a link between these two systems. This enables elements to be placed in the design file from a Fortran program.

Hardware associated with these systems is displayed in Figure 1. There are three disk drives which are addressable by name (QSA0, QSA1, QSA2). The disk controller and disk data scanner works with the information processor to read information from and write to the disks. The information processor, more commonly called the CPU (Central Processing Unit), performs all calculations and interprets the Fortran code. It executes all the support tasks and also houses the ECC/MOS memory. The system console is an I/O unit with an alphanumeric keyboard and printer instead of a screen. The system console may be used for initiating maintenance and programming tasks. The workstation is the I/O unit the LPVAX user will be the most familiar with. The workstation is described in detail in the following section.

3.2 THE WORKSTATION

In the Intergraph room there are two Interact Monocromatic and one Interact Color/Monocromatic work stations (Figure 2). These work stations consist of dual 19 inch raster displays. These displays are high resolution with 1280 by 1024 pixel addressability. The stations offer the latest in ergonomic design. The work surface is a hinged table with a sensitized work surface of 22 by 34 inches. This work table can also pivot at the hinges within a 35 degree range. The screen unit also has pivot capabilities
Figure 1. System Hardware
Figure 2. Intergraph Workstation
within a 30 degree range. The workstation and display together as one unit may be vertically adjusted over a range of 12.5 inches to accommodate users either standing or sitting.

At each workstation there are three additional devices. These devices are:

1) Cursor
2) Keyboard
3) Menu tablet

3.2.1 The Cursor

The cursor (Figure 3) is a twelve button device which is used much like a mouse. This device is used to place interactive graphics commands into the design file. The buttons are defined as follows:

C - Command - This button is for selecting graphics commands from the menu tablet.

D - Data - This button places data points on the screen and selects graphic elements already displayed on the screen.

R - Reset/Reject - This button will reinitialize the starting point in a multiple point entry and will select an identified element.

T - 2-D Tentative - This button is for tentative input. It places a crosshair on the screen where the data point would go if entered.
3-D - This is the 3-D data button. It functions much the same way as the 2-D data button.

3-T - This is the 3-D tentative button. It functions in the same manner as the 2-D tentative button.

3,5 - These buttons may be user defined.

4 - Dynamic rotate

6 - Dynamic Zoom-in

7 - Dynamic Pan

8 - Dynamic Zoom-out

In order to use buttons 4, 6, and 8 the dynamic volume must be activated. To use button 7 the dynamic pan must be activated. Further explanation of the dynamic volume and the dynamic cursor buttons will be given in section 3.4.4.

3.2.2 The Keyboard

The keyboard (Figure 4) is comprised of three keypad sections, the right hand side, the center, and the left hand side. The right hand keypad is the typical numeric keypad found on most keyboards. The Rub Out key is a back space key which erases as it backspaces. The center keyboard is also a typical keyboard with the exception of the function keys in the top row. These function keys may be user defined; however, the LPVAX users need not concern themselves with these keys. The left most keypad may be unfamiliar to the first time LPVAX user. The functions of these keys are as follows:
Figure 4. Keyboard
Right Copy - Copies the image on the right hand screen and prints it out on the V-80 plotter.

Erase - This will erase the screen and begin writing at the top of the same screen. This is not used while inside the design file.

Reset, Exit, Selftest - These keys are not used. They are for hardware diagnostics.

View - This key brings back an image when the screen goes into the save mode. When no input occurs within a two minute period the screen will go blank. Pressing the return key will also bring back the screen.

Left Copy - This key copies the left side of the screen.

Text Size - This key toggles between two different text sizes, large and small. The small text size will place more information on the screen but is harder to read.

Stop Page - This key is used to scroll page material after you have erased at least once. The user will not have to use this key.

Swap - This key allows the user to alternate screens.

3.2.3 The Menu Tablet

The menu tablet (Appendix A) is a sheet of paper located on the table surface of the workstation. This sheet
contains pictorial representations of the various graphic commands. To use the cursor and the menu tablet to invoke the graphic commands the user places the crosshairs of the cursor over the desired command and presses the "C" button. The system will then ask the user to give a data point or select an element. The user then moves the cursor across the tablet until he or she sees the crosshairs on the screen in the desired location or on the desired element. When the crosshairs have been positioned the user pushes the data button.

For exact data placement or element identification, the user may give the X and Y coordinate position, or an X, Y, Z coordinate position for three dimensional files. This is done by typing

\[ XY=200,200 \]

for two dimensional files or

\[ XY=200,200,0 \]

for three dimensional files. The location will be 200 units in the X direction, 200 units in the Y direction, and 0 units in the Z direction.

When selecting an element (a single design object) the user must enter an "Element" command off of the menu tablet and enter the data point or key in the X, Y, Z coordinates. The element being identified will then be highlighted. If more than one element passes through the given data point only one of the elements will be highlighted. If this is the wrong element, then pressing the
"Reset/Reject" button on the cursor will reject the highlighted element and the other element passing through that point will be highlighted. If this is now the correct element then repressing the data button, or keying the coordinate, will execute the chosen "Element" command.

3.3 ENVIRONMENTS

There are three system software environments the user will be using. These environments are UTILITIES, GRAPHICS, and DCL (Data Control Language). Each of these environments has its own set of functions to perform specific tasks. This section describes the purpose of each environment.

The UTILITIES environment allows the user to create a design file. The user would enter CR for create. The system will ask for a file name. The length of the file name must be less than or equal to nine alphanumeric characters. Next the system will ask for 2D/3D/2C/3C. The user must select 3D for a 3-dimensional design file; the "C" stands for cell files which are not used by the user. Finally the system will ask for the number of blocks to be assigned to the design file. Two hundred to three hundred blocks are sufficient for this LP package. This environment also supports other file operations such as file deletion, copying, and renaming of files.

The graphics environment allows the users to access their design file. After moving to the graphics environ-
ment the users have to type in only the name of their design file. After a brief delay the design file will appear on the screen.

The DCL environment must be active in order to run the LPVAX program. To run the program the user must type

RUN LPVAX

Besides executing Fortran programs, the DCL environment supports program editing, compiling, and file operations. For example, to delete the user would type

DELETE fn.EXT

where

fn - is the file name
EXT - is the extension code (DGN for a design file)

Moving from one environment to the next is quite simple. The users have to enter only the name of the environment to which they wish to move. The abbreviations for the environments may also be used. The abbreviations for the Utilities and Graphics environment are UT and GR respectively. For example, if one wanted to go to the Graphics environment from the DCL environment one would type GR after the DCL prompt.

DCL: GR

3.4 DESIGN FILE

A design file is a file which serves as a data base for a graphical image. This file holds all of the param-
eters needed to describe the image such as active level, lines, circles, and colors. It is created in the utilities environment and accessed through the graphics environment. The design file has 63 independent levels, 256 various color shades, and numerous built-in functions. This section will focus on those functions the user will most likely use in viewing his or her design file.

When the design file is created, the system accesses a standard design file known as a seed file and copies its contents into the file being created. The system first checks the users directory for a seed file (called SEEDZ.DGN for 3-dimensional images). If none exists then the system will access the standard system seed file for the creation. A special seed file was designed for this project with the level menu and coordinate axis already in place. The level menu tells the user which levels and in what colors his or her problem planes may be found. Otherwise it would be impossible for the user to keep track of his or her problem. The seed file also keeps redundant elements, such as the level menu and coordinate axis form, from taking up program storage space and processing time.

3.4.1 Design Levels

The design levels may be thought of as clear transparencies or layered design files within the design file. A design element may be placed into different levels or an
entire design may be placed into a single level. As levels are turned "on" the elements are superimposed on top of each other, as one would stack transparencies. Turning off a level would remove it from the user's view. Any combination of levels may be turned on or off. The command to turn on a level is "ON=" followed by the level number(s) and the command to turn off a level is "OF=" followed by the level number(s). Sequential levels may be turned on or off by typing the number of the starting level, then a hyphen, and then the number of the ending level. Nonsequential levels require that the user list all levels separated by commas. For example, if the user wishes to turn off levels 1, 2, 3, 4, 7, and 22 the user would type

\[ \text{OF}=1-4,7,22 \]

One level must always remain on. This level is called the active level. Level 63 was designated as the active level in the SEEDZ file. The active level may be changed by typing the command "LV=" and the new active level number. Level 63 was chosen as the active level because it is not used by the LPVAX program. The coordinate axis was placed in this level so it would not be inadvertently turned off. The level menu was placed in level 62. This menu was not designated as the active level because of its need to be turned off from the coordinate graphics view, otherwise the menu will overlap the design plane making it difficult to see the resulting graphs.
Whenever a level is turned on or off the system will ask the user to select a view. A data point given in the upper right hand corner of either screen will execute that command for that screen. Both screens may be operated on if the data point is given in both screens.

3.4.2 Design Block

The Intergraph has the capability of dividing each screen into quadrants (Figure 5) or views. The SEEDZ file was set up to display quadrant one and quadrant five. Both quadrants are located in the upper right hand corner of the screen; quadrant one on the screen to the right and quadrant five on the screen to the left. When only one quadrant or view is displayed on the screen the view will take up the entire screen. Otherwise, the screen will be divided into four sections.

Lines are drawn and data points given by specifying reference points in the design file. These reference points are called UOR's (units of reference, or sometimes called units of resolution). For three dimensional design files these UOR's form a three dimensional block which is 4,294,967,296 OUR's long, wide, and deep (Figure 6). This 3 dimensional block is known as the design block. Each UOR within and on the surface of the block is addressable by the user. A window is that portion of the block that is displayed on the screen for the user to see. The lower left hand corner of this display window is the window origin.
There is another reference point generally located in the center of the design block known as the global origin. This origin may be user specified. The size of the window is also user specified.

3.4.3 Update Command

There are a few menu tablet commands with which the user must become familiar. First is the UPDATE command (Figure 7). This command refreshes the screen when necessary. The command button entry over the "L" will refresh the left screen, the "R" will refresh the right screen, and "BOTH" will refresh both screens. Refreshing the screen is necessary when images overlap and one or more of those overlapping images is deleted or the level it is in is turned off. The pixels shared by the remaining images and the images being deleted will be turned off. When updating or refreshing the screen these pixels will be turned back on.

3.4.4 Dynamic Volume

The DYNAMIC VOLUME command (Figure 8) allows the design file to function as a three-dimensional image. The computer takes depth into account when calculating position for operations such as rotation. When this command is selected, the user is asked to supply two data points. These data points should be placed in the right hand screen at the upper left hand corner above the coordinate axis and
Figure 7. UPDATE Command

Figure 8. DYNAMIC VOLUME Command
at the lower right hand corner of the screen. The entire design should be contained within these two points. These points set up the window in the design block for which the three-dimensional operations will be used. Once these points are given, the design will automatically move to the center of the screen. The rotate, zoom-in, and the zoom-out buttons can now be used to rotate, zoom-in and zoom-out the design.

To exit from the dynamic mode the dynamic drop is used. This becomes necessary when the hidden line removal option is desired. Hidden line removal will be discussed in section 3.4.6.

3.4.5 Move and Scale

If the user wishes to see how or why the optimal point is located where it is, the objective plane may be viewed to see where it intersects the feasible region or the alternate objective plane may be moved around to see how it approaches the solid. Although the objective plane is mobile, supplying the users with an alternate plane allows them the freedom to move the objective plane without losing the position of the objective plane of the problem.

To move the alternate objective plane the user must first select the move element command. The system will ask the user to select the element. The user then gives a data point anywhere on or inside the plane parameters.
When a point on the plane is located, the plane will become highlighted. The user then moves the cursor to a new location and enters another data point. The plane will move to that new location.

The plane may be scaled up or down as needed. This is done by entering "AS=" and the scale factor you desire. For example, in order to scale a plane to half its current size the user types

\[ \text{AS=} \cdot 0.5 \]

The user then selects the element he or she wishes to scale. A data point on the element will cause the element to become highlighted. If the correct element is highlighted then the user would reenter the point to scale that element. Note that the first data point entry identifies the element and the second data point entry accepts the element.

For better control of the MOVE and SCALE commands, the user may use the above procedure with the exception that the user would enter the coordinate points instead of using the data button. An alternate objective function will be placed in the design files for the purpose of moving. One corner of this objective plane will lie on one of the maximum points on the X, Y, or Z coordinate axis. Therefore, if a maximum objective plane falls on the Y axis the plane could be moved half way towards the origin by using the following sequence.

To MOVE

1) Select the MOVE command
2) Enter \(XY=0,4000,0\)
3) Press the reset button until the objective plane is highlighted.
4) Enter \(XY=0,2000,0\)

The point on the plane at coordinate \(0,4000,0\) will moved to the new coordinate location \(0,2000,0\). The entire plane will retain its orientation at this new point (Figure 9).

To SCALE
1) Enter \(AS=.5\)
2) Enter \(XY=0,2000,0\)
3) Press the reset button until the objective plane is highlighted.
4) Enter \(XY=0,2000,0\)

The first time the coordinate point \(0,2000,0\) was entered was to define a point on the plane to be scaled. The second time the coordinate point was entered is to varify that the highlighted plane is to be scaled (Figure 9).

3.4.6 Hidden Line Removal

Some users may become confused when viewing the wire frame image of the feasible region. It may be easier for these users to view the solution solid (feasible region) as a solid object if the lines that would normally be hidden from view were erased from view. This process of removing unwanted lines from view is known as hidden line removal. The user will only see those surfaces facing him.
When using the Hidden Line option, the user will be exposed to three-dimensional viewing concepts; perspective viewing, eye point, target point, and clipping planes. In Figure 9 the viewing volume is demonstrated.

The clipping planes may be thought of as a truncated pyramid. The front of this pyramid is the near clipping plane. The back portion of this pyramid is the far clipping plane. The solution solid is suspended in the center of these planes. If any object extends beyond these planes it will be truncated in view only, it does not affect the design file. The eyepoint is located at the center of the near clipping plane. The target point is the point on the object that is directly lined up with the eye point. It might be thought of as shining a light on an object. The light source being the eye point and the spot where the light is focused is the target point. The Hidden Line option allows the user to change these parameters.

The perspective angle is the angle of the sides of the pyramid. When the angle is set to zero, the sides of the pyramid become like a box. The solid would be viewed as if it were perpendicular to the user. If this angle were set to 30 degrees, the solid would appear on the screen as if it were rotated 30 degrees. Points and lines further from the near clipping plane would appear smaller and shorter. Changing the perspective angle gives depth to the object.
Figure 9. Clipping Plane
On the Intergraph, the hidden line option is invoked when the HIDDEN LINE command is entered by placing the cursor over the HIDDEN LINE option and pressing the "C" button. The hidden line menu will then appear on the left hand screen (Figure 10). The user will only have to become familiar with a few of the options on the HLINE (Hidden Line) menu. The important options are described below.

SCREEN DISPLAY - This box should have the message CURRENT written in it. This tells the system that the user will be working with the currently displayed design file.

DISPLAY - This box is found in the Output Options block. CURRENT should also be found in this box. This box instructs the system to send the results of the hidden line removal to the screen.

VIEWING PARAMETERS - The current viewing parameters will be displayed on the screen. These parameters may be easily changed to fit the needs of the user. If the user wishes to look right down the center of the solution solid, a data point should be given inside the box marked "EYE POINT". The user would then key in 4000 [return], 4000 [return], and 4000 [return]. This process would then be repeated by selecting the TARGET POINT box; only substituting zeros in place of the 4000's.

The PERSPECTIVE ANGLE option may also be chosen if the solution with three dimensional perspective. The UPDATE
**Figure 10. Hidden Line Menu**

In the Hidden Line Menu, the following parameters and commands are displayed:

- **Viewing Parameters**
  - **Eye** X: 893, Y: -1254, Z: 4214
  - **Target** X: 3120, Y: 973, Z: 1986
  - **Perspective Viewing Angle**: 0
  - **Near Z Clipping Plane**: 0
  - **Far Z Clipping Plane**: 7717.0500
  - **Rectangular Fence**: Inactive
  - **Update Selected View**

- **Output Type**
  - **Output Type**: Smooth Constant
  - **Edges Shading**: Raster
  - **Current Frame Range**: 0 to 0
  - **Number of Cycles**: 1

- **Execution Commands**
  - **Light Source**: Inactive
  - **Execute**

- **Frame Details**
  - **Frame Number**: 0
  - **View**: 1
  - **Processor**: Current
  - **Tolerance**: 0.50

- **Help Status**
  - Click **Point** to select desired option in tutorial view.

- **Visible Surface Screen Display**

This menu is used for hidden line removal in 3D graphics and modeling software, allowing users to view and manipulate 3D models by adjusting various parameters and settings.
SELECTED VIEW option is then chosen to reorient the solid according to the eye and target point positions.

The OUTPUT TYPE block allows the user the opportunity to select the type of shading. There are three types of shading available to the user; smooth, constant, edges. The user selects the type of shading by entering a data point in the appropriate box. The purposes of these shadings are as follows:

SMOOTH SHADING - This shading is useful when shading curved surfaces. The intensity at each pixel is calculated which allows for the contour shading of curves. The solution solids having no curvature would not benefit from this selection.

CONSTANT SHADING - The entire face of the design will have the same intensity of shading. This selection is slightly faster than the smooth shading since no pixel intensity calculations are made.

EDGES (RASTER) SHADING - This type of shading produces an outlined figure with hidden lines removed.

There are two commands which initiate the shading process. These commands are LIGHT SOURCE and EXECUTE. A data point given in one of these boxes will activate that command. The light source command allows the user to enter a data point at the location where the light source is to be placed. The shading routine will begin immediately
following entry of the data. The execute command will invoke the shading routine according to the specified viewing parameters. If the design seems to disappear or becomes exceedingly dim, then the light source is opposite from the side facing the user.

The VIEW block gives the number of the view quadrant to be shaded. This should be set to one if it is not already set to one since the design is in the first quadrant.

The PROCESSOR box allows the user to decide which computer will be used to calculate the hidden line removal. The host computer is the VAX system and the default is to this system. If the GP box is chosen then the graphics processor will calculate the hidden line removal.

The TOLERANCE box is the number of pixels worth of error that can be tolerated when shading. This defaults to .5. However, any value between .5 and 8 may be placed here. This is done by entering a data point in the TOLERANCE box. The higher the tolerance value the faster the shading process.

The STATUS block prompts the user when information is needed. It also informs the user of the current status of the hidden line removal options. For example, The status box will inform the user that the hidden line removal is being calculated before the user may actually see the results on the screen.
The HELP option will allow the user to see the explanation of any of the hidden line commands. To use this option the user would enter a data point in the help block and another data point in the block they wish to see explained.

The EXIT block will allow the user to exit the hidden line removal option. The left screen will redisplay the user's Level menu for the problem. If the user runs into trouble in the HLINE option he can exit the entire design file by entering CTRL Z.

3.5 DFPI

The DFPI (Design File Processor Input) is Intergraph's software package for allowing users to write their own applications programs in Fortran which can access, create, or modify a design file. What this means is that the user may interactively manipulate design files while maintaining control of the application programs to perform other processes. All graphics commands are available to the user's Fortran program through the DFPI. Further information on this topic is presented in the appendices or may be found in Intergraph's IDGS Application Software Interface Document Version 8.6 or 8.8 if the reader desires to know more about the DFPI processing and available commands.
Chapter IV

LPVAX PROGRAM

The LPVAX program is the software package developed to compute the solution to a three variable linear programming problem using the simplex method and graph the results. The LPVAX program is comprised of the LPVAX main program and the subprograms LPROG and DPLT. The function of each of the programs is described below.

4.1 LPVAX

The LPVAX (Linear Program on the VAX) program brings together the two subprograms in a user friendly fashion. This program allows the user to set up the linear program problem, make changes, run the simplex method (LPROG), and graph the results (DPLT). To invoke the LPVAX program the user must go to the DCL environment and type 'RUN LPVAX'. The program will then ask the user to press [return] when ready to begin. The purpose of this statement is to allow the user time to get situated and to swap to a fresh screen if necessary. The next item to appear on the screen will be the master menu (Figure 11). The user enters the number of the selection he or she wishes to use. The options are:

1) ENTER THE EQUATIONS
2) EDIT THE PROBLEM
3) RUN THE SIMPLEX ALGORITHM
**ENTER YOUR SELECTION**

1. ENTER EQUATIONS
2. EDIT EQUATIONS
3. SIMPLEX ALGORITHM
4. GRAPHICAL SOLUTION
5. EXIT

**Figure 11. LPVAX Master Menu**
4.1.1 ENTER THE EQUATIONS OPTION

The first selection "Enter the Equations" is used when the user is creating a new problem. The subroutine ENTP (Enter Points) is called and the arrays A(I,J) (constraint coefficients) and RS(I) (right hand side values) are set to zero to prepare the program for fresh inputs. The first question ENTP asks is "Is this a minimization or a maximization, enter -1 for a min and 1 for max". The user responds accordingly. If any other value is entered the program will not accept the input and will re-ask the question. The values 1 and -1 are used to convert all problems to minimization problems in the simplex algorithm routine to be discussed later.

The second statement that will appear on the screen is "Enter the number of constraint equations. Do not enter more than 6." The user then keys in a number between one and six. If the value is larger than six or less then one the program will re-ask the question. The six constraint limit was placed in the LPVAX package because of the visual difficulty in viewing the graphical solution to more than 6 constraints.

The third required input is the objective function. The program states "Enter your X,Y,Z coefficients for the objective function (return after each value)". The user
then types in the three values in the order requested by the software. Zeros should be placed in the X, Y, or Z locations for variables not being used.

The fourth required piece of information is the set of constraint equations. The program explains how it expects the constraint equations to be entered. The explanation is as follows:

Next you will be entering the coefficients of the constraint equations. Enter the X,Y,Z coefficients [return], the logical operator [return], and then the right hand side of the constraint [return], as prompted. For example, the constraint $X + 2Y + 3X < 20$ is entered as follows:

```
1 [return]
2 [return]
3 [return]
< [return]
20 [return]
```

The program then prompts the user to respond by asking for each input "Enter the coefficients to equation N" where N will actually be displaying the number of the equation to be entered. The program then prompts the user for the X, Y, and Z coefficients, the logical operator, and the right hand side coefficient. After the X, Y, and Z coefficients are entered, the program will ask for the logical operators and the right hand side coefficients with the prompts

"Enter the logical operator"
"Enter the right hand side"

If the user enters an improper operator, something other than <, =, >, the program will re-ask the question. The user then enters the right hand side of the equation. If the user enters a negative value for the right hand side value, the program will inform the user that a negative for the right hand side is not valid and the user will then have to re-enter the constraint. When a constraint has been successfully entered the program will then ask for the next constraint.

Once all of the constraints have been entered, the user then has the option of viewing his equations (Figure 12). The user also has the option to edit the equations that have just been entered. If the user chooses to edit then the program will go directly into the FXPT (Fix Point) subroutine. Otherwise, the user will be returned to the main menu where again the user has the option to edit the program.

When entering or editing the problem the LPVAX program checks for the validity of the entry. When a logical operator is to be entered, only a logical operator is accepted. The same holds true for numerical entries. If a number is to be entered then only a number is to be accepted. This check is accomplished through a subroutine called READER. The reader subroutine will accept a positive or negative first character if a sign is provided. It will also accept one decimal point. Aside from these characters,
\[ \text{MAX} \quad 2.00X \quad 2.00Y \quad 2.00Z \]

\[
\begin{align*}
1.00X + &\quad 2.00Y + &\quad 3.00Z &\leq&\quad 20.00 \\
2.00X + &\quad 1.00Y + &\quad 4.00Z &\geq&\quad 10.00 \\
3.00X + &\quad 2.00Y + &\quad 1.00Z &\geq&\quad 5.00 \\
\end{align*}
\]

0 YOU WISH TO MAKE ANY CHANGES Y OR N

Figure 12. Constraints
any non-numeric input will be rejected and the user will have to re-enter the number. The reader subroutine is set up to read only one number at a time so if the user makes a mistake an entire line of data will not have to be re-entered. It is also just as easy to press a return after each entry as it is to press the space bar so this should not prove to be an inconvenience for the user.

There is an error argument in the Fortran read statement that should make the need for the reader subroutine unnecessary. Unfortunately, the error argument will not always catch a non-numeric character in a numeric entry. In such cases it will assign the value zero to the number and the error condition flag will not be set.

4.1.2 Editing the Constraints

The second option on the main menu is the edit option. This option invokes the FXPT (Fix Point) subroutine. When invoked, this subroutine first displays the current problem (Figure 13). This is then followed by the edit menu (Figure 14). The options in the edit menu are:

1) Change MAX or MIN
2) Change objective function coefficients
3) Add a constraint equation
4) Delete a constraint equation
5) Return to the Main Menu

The "Change MAX or MIN" option asks the user to enter -1 for a minimization problem or 1 for a maximization
**Figure 13. Current Problem**

<table>
<thead>
<tr>
<th>MAX</th>
<th>2.00X +</th>
<th>2.00Y +</th>
<th>2.00Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00X +</td>
<td>2.00Y +</td>
<td>3.00Z &lt;</td>
<td>20.00</td>
</tr>
<tr>
<td>2.00X +</td>
<td>1.00Y +</td>
<td>4.00Z &gt;</td>
<td>10.00</td>
</tr>
<tr>
<td>3.00X +</td>
<td>2.00Y +</td>
<td>1.00Z &gt;</td>
<td>5.00</td>
</tr>
</tbody>
</table>

**MAKE YOUR SELECTION**

1. CHANGE MAX OR MIN
2. CHANGE OBJECTIVE FUNCTION COEFFICIENTS
3. CHANGE AN EXISTING CONSTRAINT EQUATION
4. ADD A CONSTRAINT EQUATION
5. DELETE A CONSTRAINT EQUATION
6. RETURN TO MAIN MENU

**Figure 14. LPVAX Edit Menu**
problem. If anything other than a 1 or a -1 is entered the program will not accept the input and will re-ask the question. Otherwise it will write the new version of the problem and redisplay the edit menu.

When choosing the objective function edit option the program will ask the user to enter the coefficients to the objective function. The user will be prompted for each coefficient. If the user did not really wish to change the objective function, a null line may be entered to return to the main menu instead of the value of the first coefficient. The user is also informed of this option. After the new coefficients are entered the program returns to the edit menu.

If the "Change existing constraint" option is chosen, the program will display the current constraints and number them in the order the user has entered them. The program will ask the user for the row number of the constraint to be changed or to enter a null line if the user wishes to return to the edit menu. If a row is chosen, the row is displayed. At this point the user will be informed that this is the row to be changed and instructed to enter the new constraint. Again the user has the option of entering a null line if the wrong choice was made. A null line entry will bring the user back to the row selection portion of the edit routine. If the user decides to make the necessary change by entering the new constraint the user will be prompted for each entry. Upon completing the new constraint row, the user is
asked if another change is desired. If a change is desired
the user will be brought back to the row selection portion
of the program. Otherwise, the user will be brought back to
the edit menu.

The "Add a constraint" option will allow the
user to add a constraint. The program first check to see
if it is possible to add a constraint (i.e. the user has
less than 6 constraint equations). If it is not possible
to add a constraint the program will inform the user that
the maximum number of constraints has already been entered
and then return the user to the edit menu. If it is
possible to enter a new constraint, the program will
display the instructions on entering the new constraint
and will prompt the user to enter the new row. After this
is entered, the program will then ask the user if another
constraint is to be entered. If yes, the sequence is
started over. If no, the user is returned to the edit
menu. The user also has the option of entering a null
line as the first input instead of the first coefficient
if the user decides not to enter a new constraint. In
which case the program will return the user to the edit
menu.

The "Delete a constraint" option will display the
constraint equations with the row number. The user is
then asked to enter the row number of the constraint to
be deleted or a null line if the user decided to return to
the edit menu without making any changes. If the row number
is entered the row is then displayed and the user is asked if this is the row to be deleted. If the user responds with an "N" for no the program will redisplay the rows and again ask if a row is to be deleted. If the user answers with a yes "Y" the row will then be deleted. The user will then be given the option of deleting another row. Otherwise, the user will be brought back to the edit menu.

The "Return to the main menu" command will allow the user to return to the main menu.

4.1.3 Simplex Algorithm

The LPROG subprogram performs the simplex algorithm on an LP problem and returns the solution to the main program. The arguments passed to the LPROG subprogram are the number of constraint equations, the 1 or -1 if MAX or MIN type problem, logical operators of the problem, the constraint coefficients, the objective function, and the right hand side coefficients. The program returns the location of the optimal solution.

The algorithm used for LPROG is a modification of two separate LP programs. These programs being LINPR, found on the HP1000 system in the Department of Industrial and Systems Engineering at Ohio University, and a program by Lon Poole as referenced in a thesis by Bevis (Poole, 1979; Bevis, 1985). The dual variables and the sensitivity analysis were developed by the author following standard algorithms (Ignizio, 1982).
The solutions to the dual problem and the sensitivity analysis were placed into the program as user options. This allows the student who has not covered these areas to skip the analyses. The solutions would be more confusing than helpful to the beginning LP student.

4.1.4 Graphical Solution

When the user chooses the graphical solution option, the LPVAX calls the DPLT subprogram. The arguments passed to this routine include the constraints and the final solution as determined by the simplex method if one exists. The user is requested to supply several pieces of additional information. First, the program will ask

"ENTER DESIGN FILESPEC - name.DGN"

The program is looking for a keyboard input telling the system the device, the user file directory, the design file name of a design file created prior to starting the LPVAX program, the extension, and the version. However, the device and file directory have defaults to the users directory so these inputs may be ignored. The user will be entering

file.DGN

The system will also default to the most recent version. Next the user is requested to enter the working level of the design file to be started. The user is given four levels to choose from. The program then calculates and graphs the solution solid, each constraint plane, the
evolution of the solution solid, the objective plane, and an alternate objective plane. The program also places a star (or asterisk) at the location of the optimal solution. It is possible to place four problems in one design file.

The graphical solution option should be chosen after the simplex algorithm has been executed. The simplex algorithm determines if there is an actual solution and what that value is. Provisions have been made so that if the simplex algorithm has not been executed or if the solution is infeasible or unbounded only the constraint planes will be graphed. The user will be notified that only the constraint planes have been graphed and why.

4.1.5 EXIT

The exit command allows the user to exit the software package and enter the DCL environment. From here the user can then go to the graphics environment and open the design file to view the problem(s).

4.2 LPROG

When the simplex algorithm is chosen from the main program, the simplex algorithm will be performed on the current problem by the subprogram LPROG (Linear PROGram). The program will rearrange the constraints so that all the less-than logical operators are followed by the equal-to logical operators which are then followed by the greater thanps. This aids the program when adding the
slack, surplus and artificial variables in columns from left to right in that order. If the problem is a maximization type, the program will switch it to be a minimization problem. This allows the program to use the same rule in solving linear programming problems.

There are several advantages to solving only minimization problems. First, the program does not have to check the type and readjust once the artificial variables are eliminated. The program does not need a separate routine when searching the $C_j - Z_j$ row for the most negative (minimization) or the most positive (maximization). The program does not have to readjust the number of columns being processed by eliminating the columns containing the artificial variables. By setting all the problems to be minimization problems, the program continues through the pivoting routine until the solution is found, if one exists. The artificial variables will go to zero and stay zero unless the solution is infeasible.

The process the LPROG program uses in solving the linear programming problems is standard. It searches the $C_j - Z_j$ left to right for the smallest negative number. The column containing this number becomes the pivot column. The $B$ vector column is then searched for the lowest limit (the ratio of the right hand side constraint with its corresponding positive coefficient). In case of ties, the left most or top most value of that tie will be chosen. LPROG checks for infeasible solutions by checking to see if
the artificial variable exists in the final solution. Unbounded solutions are determined when all of the values in the pivot constraint are negative.

The initial tableau is displayed to the user. The program then asks if the user wishes to see the tableaus after each iteration. The user responds with "Y" or "N". The simplex algorithm is then performed. The routine produces one of three possible results, infeasible solution, unbounded solution, or the optimal solution. If the solution is infeasible or unbounded the user will be notified and returned to the main menu. When a feasible solution exists the program displays the final tableau, the X, Y, Z values, the final optimal value and, if one exists, will inform the user that alternate optimal solution(s) exists. The user will then be given the option of seeing the solution to the dual. The option to perform a sensitivity analysis on the problem will then be given. The sensitivity analysis will be performed on the right hand side values and on the variables.

4.3 DPLT

The DPLT (Design PLoTs) subprogram literally plots points in a pre-existing design file. There is room in one design file for four different problems. The design file, which was created in the utility environment, is called up and opened by the DPLT program. The extreme
points are calculated and the lines connecting them are drawn. The graphs that are placed in the design file are:

1) Constraint planes
2) Solution solid
3) Evolution of the solution solid
4) Optimal point
5) Optimal solution plane
6) Varying optimal solution plane

4.3.1 Constraint Planes

The constraint planes are the first objects to be placed in the design file. Since there are three variables, two coefficients at a time are set to zero and the third variable is calculated. This will give the three extreme points on the coordinate axes. The DFPI shape command places the triangular shapes into the design file. If a constraint has less than three variables then a line or point may be plotted. Each plane is placed in a separate design level, always starting at one level greater than the starting level (user specified). Each of these planes is assigned a color so the user will know which constraint(s) is being viewed.

While calculating the points of the solution plane, this program looks for the maximum point value. This point is needed as a scaling factor for all the constraint planes when being graphed. The reason the scaling factor is important is that each coordinate axis is 4000 UOR's in
length and scaling will utilize as much of the 4000 units as possible without allowing any of the planes to extend beyond these limits.

Once the constraint planes have been calculated the points of these planes are entered into the array EXPT (EXtreme PoinT). The subroutine CHKPT (CHeCK PoinT) is used to check the constraint plane points for feasibility. When a feasible point has been determined, it is placed in the array FSPT (FeaSible PoinT). The constraint planes are also entered into the subroutine FNPT (FiNd PoinT) to find all the intersecting points. The subroutine CHK2 (CHeCK 2) then determines which ones of these intersecting points are feasible.

Each plane is graphed and placed in a separate design file level and in a different color. This is to help the user distinguish one constraint or solid from the others. Constraint planes will have lines drawn toward the origin or toward the outer ends of the axes depending on whether it is a less than or a greater than constraint. Equations will not have these extension lines since the surface of the plane is the extent of the solution to that equation. A menu which is found in the design file will help the user find the graph to be viewed.

4.3.3 FNPT Subroutine

The intersection of two planes is determined by
using the Gauss-Jordan reduction method. In solving a linear system the relationship

\[ X_b = [B^{-1}] [b] \]

is used where

- \( X_b \) = the solution to the current basis
- \( B^{-1} \) = the inverse of the original basis
- \( b \) = the original right hand side vector

To solve these equations using the Gauss-Jourdan method, the linear system must be transformed into the augmented matrix

\[ [B \ | \ I \ | \ b] \]

where

- \( B \) = the current basis
- \( I \) = the identity matrix
- \( b \) = the original right hand side vector

Row manipulation is performed yielding the final matrix

\[ [I \ | \ B^{-1} \ | \ X_b] \]

The subroutine FNPT uses this method to find every intersecting point taking two planes at a time. Again the points are placed in the array EXPT. The feasible points are determined in the subroutine CHPT2 and placed in the array FSPT.

4.3.4 ORDPT Subroutine

After all the feasible points are found, they must be connected in such a manner that they outline the solution solid formed from the intersecting planes. An algorithm was
developed to determine the relationship between extreme points and which of these points should be connected. The algorithm was developed by building three dimensional string models and observing the relationships among the planes. Next, a FORTRAN routine was developed to test the extreme points for line connections. This testing is accomplished in the ORDPT (ORDER Point) subroutine. The relationship of each point with every other point is checked and when the connection criteria are met the points are marked for line connection. A line may be drawn between two points if one of the following conditions is met.

1) If both points lie on the same X, Y, or Z axis.

2) If both points lie on different axes but are on the same constraint plane.

3) Both points lie in the same two intersecting planes.

4) If one point is an axial point and the other point is on the boundary plane to that axis and they lie on the same constraint plane.

5) If one point is the intersection of three planes and the other point is the intersection of two of the three planes.

To demonstrate these constraints, assume the two intersecting planes in Figure 15-A represent less than constraints and the darkened spots represent the extreme points. Condition 1 will connect any extreme points which lie on the same axis regardless of the constraint planes.
The origin, when an extreme point, counts as a point an all three axes (Figure 15-B). Condition 2 holds for a segment of a plane that is not intersected by another plane (Figure 15-C). Condition 3 describes the condition of one plane passing through another. A boundary to the feasible solution is formed where the planes intersect (Figure 15-D). Condition 4 allows points to be connected when one extreme point is on one axis and the other extreme point shares a plane with the first point and rest on a boundary plane adjacent to that axis (Figure 15-E). The boundary plane is the xy, xz, and the yz coordinate planes. The result of the first four criteria is demonstrated in Figure 15-F.

It is very difficult to represent condition 5 on a two-dimensional sheet of paper. Figure 16-A shows three constraint planes which intersect at a point not on a boundary plane. Plane one and two represent less than constraints while plane three represents a greater than constraint. Figure 16-B will result from these constraints. The extreme point at the intersection of all three planes will be connected to every point that contains at least two of the planes making up the three plane point. The result is a multifaceted shape.

It would be very easy to draw the line connecting two points as the points are found by using the draw line command. However, this does not allow for hidden line removal in the finished solution solid. Instead, all of the connecting pairs of points must be determined and then a
Figure 15-A. Constraint Planes

Figure 15-B. Axial Point Connection
Figure 15-C. Two Axial Points and One Constraint

Figure 15-D. Same Two Intersecting Planes
Figure 15-E. Axial Point, Boundary Plane, Shared Plane Connection

Figure 15-F. Final Solution Solid
Figure 16-A. Three Constraint Planes

Figure 16-B. Point Connection with Non-boundary Non-axial Point
path covering all possible combinations of connections must be found. This draws the solid as a complete shape rather than a collection of line segments. When a connecting pair of extreme points are found they are placed in a matrix called LNCNT (LiNe CoNnecTion). If, for example, point 2 is connected to point 3 then a one is placed at the locations 2,3 and 3,2 (Figure 17). Once all the connecting pairs are entered into the LNCNT matrix, the points must be ordered so that

1) A line is drawn moving continuously from point to point.
2) All paths to every point are covered.
3) The path ends at the initial point.

Backing over previously drawn lines is allowed in order to meet the above constraints. A second algorithm was developed to accomplish this. The algorithm is as follows:

1) Number each extreme point.
2) Establish an NXN matrix where N is the number of extreme points.
3) Populate the matrix with the number one at every location in the matrix that represents connected points.
4) Establish a second matrix to record every point that is connected as it is connected. This matrix is used for backtracking.
5) Establish a third matrix to keep track of the complete path taken to complete the shape. This
Figure 17. Connection Matrix

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
2 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
3 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
4 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
6 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
7 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
8 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]
matrix is used for drawing the shape.

6) Starting at point one, use the connection matrix to create a path of consecutive points.

7) As points are found, replace the number one in the matrix with the number two.

8) When a point is reached where no new paths away from the point exist and there are still lines to be drawn, backtrack to the first point which still has a non-connected path leading from it.

9) When all connection lines are located, repeat the above process looking for the number two until the first point is reached.

To explain this process refer to Figure 17. Starting with point 1, place it in the one dimensional array DRWLN (DRAW LiNe). The program then searches to find the lowest numbered point that is connected to point 1. When found, a numeral 2 is placed at that location and the reverse. For example, if point 1 is connected to point 2 then positions 1,2 and 2,1 will receive a numeral 2. Point 2 is then entered into the DRWLN array. The second row is then searched for the lowest numbered point to which point 2 is connected, signified by the value 1. When a dead end is encountered (i.e. only twos and zeros in the row) the program checks to see whether all of the lines have been drawn. If not, it will then proceed to retrace its path until it comes to a point from which it is able to draw a new line (see Appendix B for complete box evolution).
When all the lines are drawn, the program will then return to the first point in a similar manner except that it will be looking for the value two instead of one. At each point it will test to see if it is at the first point. If it is then the solid formation is complete. Table 1 shows the final elements of the DRWLN array.

4.3.5 Solid Evolution

One other major aspect of the DPLT subprogram is that it graphs the evolution of the solution solid. This means that the solution solid formed from equations 1 and 2 is graphed. The solution solid to equations 1, 2, and 3 is then drawn. This process continues up to equation 6. Each solid is placed in a separate level of the design plane.

To do this, each solid in the evolution must be kept in its own feasible point array. These arrays will only be concerned with the feasible points associated with the particular constraint equations. This ensures that stray points are not added to the solution regions. For example, a point from constraint four will not be plotted in the graph for constraints one and two even though it may satisfy these two constraints. Whenever a feasible point is found it is placed in all the necessary feasible point arrays. All the arrays use the ORDPT subroutine to graph the solution.
Table 1. Elements of the DRWLN Array

<table>
<thead>
<tr>
<th>Element #</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

} Backtrack

} Backtrack

} Backtrack

} Finished

} Return to Start
4.3.6 Optimal Point

An asterisk is placed on the solution solid at the point of optimality. It is placed by creating a collection of very short lines slightly thicker than the solution lines. Both this and the final solution solid is placed into the starting level specified by the user.

4.4 Other Graphical Considerations

In graphing three-dimensional solution planes, negative value coefficients in the constraints and the objective function require special consideration. Unlike three-dimensional linear programming graphics packages, two-dimensional problems require little graphical compensation. For instance, in a two-dimensional problem only one coefficient may be negative at a time since the variables and rhs coefficient must always be positive. In plotting a constraint like \(-2x + 3y > 6\) two points would be drawn on the graph. A line connecting these points may be extended into the positive feasible region. This extension line serves as a feasible region boundary to the given equation (Figure 18).

Three-dimensional problems are more complicated to graph. The graphics package must allow for one, two, and three variable constraints; must graph two-dimensional problems as well as three-dimensional problems; and must show the direction of the logical operator. A zero rhs coefficient can add to the complexities. This section will
Figure 18. Two Dimensional Graph
deal with the nature of these problems and the resolution of these problems.

Before any graphing can begin, the DPLT program determines the nature of the constraints. It checks the logical operator of each constraint and counts the number of zero coefficients in each equation. It must check for the maximum size of the constraints and scale the constraints to fit the graph.

4.4.1 Graphing the Logical Operator

When the logical operator is either a "greater than" or a "less than" symbol, extension lines are needed to show direction. These extension lines will lie on the coordinate axes extended away from the origin when the constraint is of a "greater than" type and will extend toward the origin when it is of a "less than" type. To graph these directional lines the GREAT subroutine is called or the LESS subroutine is called for the appropriate logical operator. An "equal to" logical needs no directional lines since the feasible region will be the surface of the resulting constraint plane (Figure 19).

4.4.2 Drawing the Feasible Region

If there are no coefficients in a constraint equal to zero then the constraint is drawn as a triangular region on the coordinate axes. This is drawn using the system's shape routine SHDFPI. When one of the coeffi-
Figure 19. Logical Operator Direction
cients in a constraint is zero the subroutine LINES is called. The subroutine will draw a two dimensional triangular region. However, this does not display the entire feasible region. For a three-dimensional graphics problem solution region is a wedge extending infinitely in the coordinate direction where no coefficient was specified. The LINES subroutine will also add extension lines to the graph to demonstrate this feasible region (Figure 18).

When there are two zero coefficients and only one coefficient with a value the POINT subroutine is called. The feasible region in a case such as this is a plane or area bounded by the given constraint coefficient. The other two coordinate variables will be unbounded. These variables will be shown as infinite lines stemming from the boundary point (Figure 20).

4.4.3 Negative Constraint Coefficients

Negative constraint coefficients form an additional problem. There may be one or two constraints with negative values and the resulting regions extend into the negative coordinate regions. Since the feasible region lies solely in the the positive x,y,z quadrant, extension lines must be drawn to show how this region extends into the positive region. The EXT subroutine is used to place these extension lines.

When there is only one negative coefficient, for example -z and positive x and y coefficients, the exten-
Figure 20. Two Constraint Coefficients are Zero
sion is handled differently than when two negative coefficients are present in a constraint. The lines connecting the x and z coefficients and the y and z coefficients are extended in the positive quadrant. The result is a plane slicing through the positive region (Figure 21). The cross hatching in Figure 21 and 22 represents the portion of the plane in the feasible region.

When there are two negative coefficients, for example \(-y\) and \(-z\), the slicing into the positive quadrant is not easy to see. To show how the plane extends into the positive quadrant the xy and the xz lines must be extended. The result is an hourglass shape. The non-negative variable will serve as the focal point to this hourglass (Figure 22).

The graphs of the constraints will show the negative infeasible portion of the triangular region as well as the feasible portion. The reason for including the infeasible region is to show the user how and why the graph is shaped as it is. Graphing the extension lines is an absolute necessity if the user is to see how the final solution to the problem was found.
Figure 21. One Negative Coefficient
Figure 22. Two Negative Coefficients
Chapter V
CONCLUSIONS and RECOMMENDATIONS

In evaluating the overall software package, the program criterion, and program considerations (Section 2.3) should be examined. The resulting graphics must also be examined. Upon completing the project, a check to see if the program addressed the list of considerations created was made prior to the start of coding. From this list, it is believe that the software provides a straightforward means of entering the problem. The user can not enter improper data and the menus are displayed to aid the user in processing the problems. The user may see the problem before processing, make changes, start a new problem, and provides explanatory prompts.

The criteria for the simplex method was likewise met. The user will see the initial tableau, the final tableau, and the x, y, z variables. The user has the option to see each tableau after each iteration, the solution to the dual, and to run a sensitivity analysis.

The graphical solution criteria were also met. The program is designed to run a two-dimensional or a three-dimensional problem. It will graph the optimal solution point with an asterisk at the coordinate location. Each constraint plane is distinguishable from the others by color. The evolution of the feasible region is graphed. It graphs the objective plane which is mobile through the
feasible region. Shading capabilities are supplied to enable the user to remove hidden lines. The user is able to view each graphed object at will; the user is in total control of what is appearing on the screen at any one time. The graphics commands are not very complex to use and the user needs only to learn a few commands to effectively use the system.

Although the software and hardware criteria were met, there are still weaknesses in the overall package and computer system. One weakness is in the ordering of the constraints. The user is free to enter the constraints in any order. The constraints which appear on the screen for viewing or editing and the order in which they appear in the graphics file will be in the same order they were entered. The order which will appear in the solution to the simplex algorithm will be all the less than's, then all the equal to's, followed by the greater than's. This difference should not create a problem for the user as long as the user is aware of it. This method, which does not force the user to input the problem in a particular order, will allow the user to view the evolution of the feasible region in the order the user chooses. If the user feels that the difference in ordering will cause a problem then the user should order the problem before entering it.

A second weakness is the unavailability of the Intergraph system. It is a very expensive system used for research. It may not be readily available for use of the
LPVAX package. The system is very complex and some students would be reluctant to use it. However, other students would benefit greatly by using this system. The LPVAX visually demonstrates linear programming concepts which will aid the student. It would also provide the student with an introduction to the intergraph system which may lead to future research. Considering this advantage and disadvantage, the LPVAX program should be used on a volunteer basis.

There is a condition in which the resulting graphics appear to be missing one or more connection lines. This occurs when graphing the evolution of the solution solid. The evolution may go through phases of infeasibility before reaching the feasible final solution.

In viewing various linear programming problems, it was noticed that it could be very difficult to see how the solution solid was formed by the constraints if the problem was very complex. The more constraints there were the greater the visual problem became. It would be best if the number of constraints were limited to three. If more than three constraints are to be used then only constraints with positive coefficients should be used. Three constraints is sufficient to demonstrate linear programming concepts and create graphs which are not difficult to view and understand.

This package would be very beneficial to students but not necessarily on the same system. It may be downloaded
to a micro vax or converted to a PC. All the routines were made to be modular to ease future enhancements or changes to fit another system.

It is possible to convert this thesis program package to other systems like the HP1000 or a PC. The HP1000 system has a graphics package (Graphics 1000) which can be used for three-dimensional drawings (Grandhee, 1985). In Grandhee's master's thesis he describes his program for three-dimensions. As with the Intergraph system, the Graphics 1000 package is accessed through call statements. However, it is up to the programmer to write the rotation and hidden line routines. Unfortunately, rotation in these routines is not continuous as with the Intergraph. Instead the user would have to enter a new view point and the system will redisplay the object according to that point. Some of the rotational motion effect is lost. There are, however, numerous algorithms for this rotation process and the hidden line removal process.

Microprocessors are also capable of three-dimensional graphics. It is possible to rewrite this package for PC's. As with the HP1000 system, the rotational and hidden line removal will have to be programmed into the package. There are graphics packages on the market which will support programs which would access their graphics routines. The only major changes in this instance would be the call statements that accesses these routines. Using variations of package on another system is highly recommend-
ed since other systems are less expensive and more accessible to the student.
REFERENCES


APPENDIX B

CONNECTION MATRIX
### Initial Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
First Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Second Connection Matrix

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Third Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### Fourth Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Fifth Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Sixth Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Seventh Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
First Backtrack - Matrix Remains Unchanged
### Eighth Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The diagram shows a 3D structure with labeled vertices 1 through 8, and arrows indicating connections between them. The table below represents the connection matrix for these vertices.
**Nineth Connection Matrix**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Second Backtrack - Matrix Remains Unchanged
Tenth Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Eleventh Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Third Backtrack - Matrix Remains Unchanged
Last Connection Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Return to Start
**THIS IS A LINEAR PROGRAMMING PACKAGE**

**FOR A THREE VARIABLE PROBLEM UP TO 6**

**CONSTRAINTS, THE GRAPHICAL SOLUTION**

**TO THE PROBLEM WILL BE PLACED IN A**

**DATA FILE.**

**CREATED BY LINDA A. HUMPHREYS**

**DIMENSION A(7,7),RS(7),OBJ(3),SOLN(16)**

**CHARACTER SIGN(7),CHAR**

**WRITE (6,2070)**

**FORMAT (/',' THIS PROGRAM COMPUTES THE OPTIMAL SOLUTION'**, X2X,' TO A 3 VARIABLE LINEAR PROGRAMMING PROBLEM AND GRAPH**

**X2X,' THE SOLUTION PLANES AND THE RESULTING SOLUTION SOLID**

DO 2080 I=1,7

DO 2091 J=1,7

A(I,J)=0

2081 CONTINUE

RS(I)=0

2080 CONTINUE

WRITE (6,2090)

2090 FORMAT (/'0', ' PRESS [RETURN] WHEN READY TO BEGIN'')

READ (5,2091) CHAR

2091 FORMAT (A)

3000 WRITE (6,3002)

3002 FORMAT (/'", ' LPVAX **************'

WRITE (6,3004)

3004 FORMAT (/'", 'T5,'1', 'T9,'ENTER THE PROBLEM', 'X5', '2', 'T9,'EDIT THE PROBLEM', 'X5', '4', 'T9,'SIMPLEX ALGORITHM', 'X5', '5', 'T9,'EXIT', 'X5'

2082 WRITE (6,3006)

3006 FORMAT (/'", ' ENTER YOUR SELECTION')

CALL READER (C,CFAIL)

INPUT=C

IF (CFAIL.NE.0) GOTO 2082

IF(INPUT.EQ.1) GOTO 3010

IF(INPUT.EQ.2) GOTO 3020

IF(INPUT.EQ.3) GOTO 3030

IF(INPUT.EQ.4) GOTO 3040

IF(INPUT.EQ.5) GOTO 3050

PRINT 2, INPUT, ' IS NOT A VALID SELECTION'

GOTO 3000

3010 CALL ENPT (M,Z,SIGN,A,OBJ,RS,FFLAG)

GOTO 3000

3020 CALL FXPT (M,Z,SIGN,A,OBJ,RS,FFLAG)

GOTO 3000

3030 CALL LPROB (M,Z,SIGN,A,OBJ,RS,SOLN,CS,FFLAG)

GOTO 3000

3040 CALL DPT (M,Z,SIGN,A,OBJ,RS,SOLN,CS,FFLAG)

GOTO 3000

3050 CONTINUE

**SUBROUTINE FOR ENTERING THE INITIAL EQUATIONS**

**SUBROUTINE ENPT (M,Z,SIGN,A,OBJ,RS,FFLAG)**

**CHARACTER AA,SIGN(7)**

**DIMENSION A(7,7),OBJ(3),RS(7)**

**FFLAG=3**

DO 2080 I=1,7

DO 2091 J=1,7

A(I,J)=0

2081 CONTINUE
RS(1) = 0
2080 CONTINUE
2092 WRITE (6, 2003)
2093 FORMAT (/,' IS THIS A MINIMIZATION OR A MAXIMIZATION? ',
X' ENTER -1 FOR MIN AND 1 FOR MAX')
CALL READER (X, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 2002
IF (X .NE. 1 .AND. X .NE.-1) GOTO 2002
2094 WRITE (6, 2006)
2096 FORMAT (/,' ENTER THE NUMBER OF CONSTRAINTS',/
X' DO NOT ENTER MORE THAN 6')
CALL READER (M, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 2004
M = C
IF (M .GT. 6 .OR. M .LT. 1) GOTO 2004
WRITE (6, 2008)
2098 FORMAT (/,' ENTER YOUR X,Y, AND Z COEFFICIENTS FOR THE',/
X' OBJECTIVE FUNCTION (PRESS RETURN AFTER EACH VALUE')
1 PRINT*, ' ENTER THE COEFFICIENT FOR X'
CALL READER (C, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 1
OBJ(1) = C
2 PRINT*, ' ENTER THE COEFFICIENT FOR Y'
CALL READER (C, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 2
OBJ(2) = C
3 PRINT*, ' ENTER THE COEFFICIENT FOR Z'
CALL READER (C, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 3
OBJ(3) = C
WRITE (6, 2011)
2011 FORMAT (/,' NEXT YOU WILL ENTER THE COEFFICIENTS TO THE',/
X' CONSTRAINTS , ENTER THE X,Y,Z COEFFICIENTS',/
X' [RETURN] THEN ENTER THE LOGICAL OPERATOR [RETURN]',/
X' AND THEN THE RIGHT HAND SIDE VALUE [RETURN]',/
X' EXAMPLE: 1 [RETURN]',/
X' 2 [RETURN]',/
X' 3 [RETURN]',/
X' 12 [RETURN]' )
DO 2013 I = 1, M
WRITE (6, 2012) I
2012 FORMAT (/,' ENTER THE COEFFICIENTS TO CONSTRAINT ',I1)
100 PRINT *, ' ENTER THE COEFFICIENT FOR X'
CALL READER (C, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 100
A(I, 1) = C
102 PRINT *, ' ENTER THE COEFFICIENT FOR Y'
CALL READER (C, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 102
A(I, 2) = C
103 PRINT *, ' ENTER THE COEFFICIENT TO Z'
CALL READER (C, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 103
A(I, 3) = C
10 PRINT *, ' ENTER THE LOGICAL OPERATOR'
READ (5, 112) SIGN(I)
112 FORMAT (A1)
IF (SIGN(I) .NE. '<' .AND. SIGN(I) .NE. '=' .AND. SIGN(I) .NE. '>')
X GOTO 10
104 WRITE (6, 113)
113 FORMAT (/,' ENTER THE RIGHT HAND SIDE')
CALL READER (C, CALFAIL)
IF (CALFAIL .NE. 0) GOTO 104
RS(I) = C
IF (RS(I) .GE. 0) GOTO 2013
PRINT*, ' YOU HAVE A NEGATIVE RIGHT HAND SIDE COEFFICIENT'
PRINT *, 'RE-ENTER THE LINE'
GOTO 100
2013 CONTINUE
*
2014 PRINT *, 'DO YOU WISH TO VIEW YOUR PROBLEM, Y OR N'
READ (5,2015) AA
2015 FORMAT (4I)
IF (AA.EQ.'N') GOTO 2024
IF (AA.EQ.'Y') GOTO 2016
GOTO 2014
*
2016 IF (Z.EQ.1) GOTO 3018
WRITE (6,2017) (OBJ(J),J=1,3)
2017 FORMAT (/,'X1:MIN ','F10.2','X1:2X','F10.2','Y1:2X','F10.2')
GOTO 2019
2018 WRITE (6,2018) (OBJ(J),J=1,3)
2019 DO 2022 I=1,M
WRITE (6,2020) (A(I,J),J=1,3),SIGN(I),RS(I)
2020 FORMAT (/,'X1:','F10.2','Z:','F10.2','Z:','A1:','=X10.2')
2022 CONTINUE
2023 WRITE (6,2025)
2025 FORMAT (/,'DO YOU WISH TO MAKE ANY CHANGES Y OR N'
READ (5,2026) AA
2026 FORMAT (4I)
IF (AA.EQ.'N') RETURN
IF (AA.NE.'Y') GOTO 2023
CALL FXPT (M,Z,SIGN,A,OBJ,RS,FFLAG)
2024 CONTINUE
RETURN
*
* THIS SUBROUTINE ALLOWS THE USER TO MAKE CHANGES IN THE PROGRAM *
*
SUBROUTINE FXPT (M,Z,SIGN,A,OBJ,RS,FFLAG)
CHARACTER AA,SIGN(10),AB=3
DIMENSION A(2,7),RS(7),OBJ(3)
2019 IF (Z.EQ.1) AB='MAX'
IF (Z.EQ.-1) AB='MIN'
WRITE (6,2020) AB,(OBJ(J),J=1,3)
2020 FORMAT (/,'X1:','A3:','=F10.2','X1:','=F10.2','Y1','=F10.2','Z:','=F10.2')
DO 2019 I=1,3
KMM=MM+1
DO 2010 J=KMM,7
4(A(J,J),I)=0
RS(JJ)=0
2010 CONTINUE
2019 CONTINUE
DO 2022 I=1,M
WRITE (6,2021) (A(I,J),J=1,3),SIGN(I),RS(I)
2021 FORMAT (/,'X1:','F10.2','Z:','=F10.2','Y1','=F12.2','Z:','=A1:','=X10.2')
2022 CONTINUE
2023 WRITE (6,2023)
2023 FORMAT (/,'EDIT MENU',/)
XTS,1,'CHANGE MAX OR MIN',/ 
XTS,2,'CHANGE OBJECTIVE FUNCTION COEFFICIENTS',/ 
XTS,3,'CHANGE AN EXISTING CONSTRAINT',/ 
XTS,4,'ADD A CONSTRAINT',/ 
XTS,5,'DELETE A CONSTRAINT',/ 
XTS,6,'RETURN TO MAIN MENU',/)
4 WRITE (6,5)
5 FORMAT (/,'ENTER YOUR SELECTION')
CALL READER (C,CFAIL),
INFO=C
IF (CFAIL.NE.0) GOTO 4
0.00

* THIS SECTION ALLOWS MIN OR MAX CHANGES

2010 WRITE (6,3010)
0.00

3010 FORMAT ('/ ', 'ENTER 1 IF YOU WISH TO MAXIMIZE OR A', ' ',  
X', ' -1 IF YOU WISH TO MINIMIZE')

CALL READR (Z, CALFAIL)

IF (CALFAIL, NE, 0) GOTO 2030

IF (Z, NE, 1, AND, Z, NE, 1) GOTO 2030

FIND = 3

GOTO 2019

* THIS SECTION CHANGES THE OBJECTIVE FUNCTION

2040 WRITE (6,2008)

2008 FORMAT ('/ ',' ENTER YOUR X, Y, AND Z COEFFICIENTS FOR THE ' ,  
X', ' OBJECTIVE FUNCTION (PRESS RETURN AFTER EACH VALUE ' ,  
X', ' OR ENTER A NULL LINE TO RETURN TO THE EDIT MENU')

1 PRINT*, ' ENTER THE COEFFICIENT FOR X'

CALL READR (C, CALFAIL)

IF (CALFAIL, EQ, 2) GOTO 2019

IF (CALFAIL, EQ, 1) GOTO 5

OBJ(1) = C

FIND = 3

2 PRINT*, ' ENTER THE COEFFICIENT FOR Y'

CALL READR (C, CALFAIL)

IF (CALFAIL, NE, 0) GOTO 2

OBJ(2) = C

3 PRINT*, ' ENTER THE COEFFICIENT FOR Z'

CALL READR (C, CALFAIL)

IF (CALFAIL, NE, 0) GOTO 3

OBJ(3) = C

GOTO 2019

* THIS SEGMENT ALLOWS ROW CHANGES FOR CONSTRAINTS

2050 DO 2054 I = 1, N

WRITE (6, 2052) I, (A(I,J), J = 1, 3), SIGN(I), RS(I)

X', 'Z', 'A', 'F10.2')

2054 CONTINUE

17 WRITE (6, 251)

251 FORMAT ('/ ', ' ENTER THE NUMBER OF THE ROW YOU WISH TO CHANGE',  
X', ' OR A NULL LINE TO RETURN TO THE EDIT MENU', ' ')  

CALL READR (C, CALFAIL)

IF (CALFAIL, ER, 2) GOTO 2019

IF (CALFAIL, EQ, 1) GOTO 17

I = C

IF (I, GT, M, OR, 1, LE, 0) GOTO 17

WRITE (6, 2056) (A(I,J), J = 1, 3), SIGN(I), RS(I)

X', ' A', 'F10.2',  
X', ' A', 'F10.2')

WRITE (6, 2058)

2055 FORMAT ('/ ', ' THIS IS THE ROW TO BE CHANGED')

WRITE (6, 2057)

2057 FORMAT ('/ ', ' ENTER NEW ROW COEFFICIENTS. REMEMBER JUST', ' ',  
X', ' ENTER THE VALUES AND PRESS RETURN AFTER EACH VALUE', ' ')  

X', ' EXAMPLE: 1 [RETURN] 2 [RETURN]'  

X', ' 3 [RETURN]'  

X', ' 4 [RETURN]'}
100 PRINT 2, 'ENTER THE COEFFICIENT FOR X'
   CALL READER (C, CFAIL)
   IF (CFAIL.EQ.2) GOTO 17
   IF (CFAIL.NE.0) GOTO 100
   A(I,1)=C
   FFLAG=3
20  PRINT 2, 'ENTER THE COEFFICIENT FOR Y'
   CALL READER (C, CFAIL)
   IF (CFAIL.NE.0) GOTO 102
   A(I,2)=C
30  PRINT 2, 'ENTER THE COEFFICIENT TO Z'
   CALL READER (C, CFAIL)
   IF (CFAIL.NE.0) GOTO 103
   A(I,3)=C
40  PRINT 2, 'ENTER THE LOGICAL OPERATOR'
   READ (5,112) SIGN(I)
50  PRINT 2, 'ENTER THE RIGHT HAND SIDE'
   CALL READER (C, CFAIL)
   IF (CFAIL.NE.0) GOTO 104
   RS(I)=C
   IF (RS(I).GE.0) GOTO 200
   PRINT 2, 'YOU HAVE A NEGATIVE RIGHT HAND SIDE COEFFICIENT'
   PRINT 2, 'REENTER THE LINE'
   GOTO 100
200 WRITE (6,20)
   FORMAT ('(',M,1X,'DO YOU WISH TO CHANGE ANOTHER CONSTRAINT'
   X OR N')
   READ (5,22) AA
22 FORMAT (A1)
   IF (AA.EQ.'Y') GOTO 2050
   GOTO 2019
2050 FORMAT (A1)
   IF (M(I).GT.6) GOTO 2055
   WRITE (6,4057)
4057 FORMAT (A1, 'ENTER NEW ROW COEFFICIENTS, REMEMBER JUST'
   X ENTER THE VALUES AND PRESS 'RETURN' AFTER EACH VALUE'
   X EXAMPLE: 1 'RETURN'/'
   X 2 'RETURN'/'
   X 3 'RETURN'/'
   X OR ENTER A NULL LINE TO RETURN TO THE EDIT MENU'
400 PRINT 2, 'ENTER THE COEFFICIENT FOR X'
   CALL READER (C, CFAIL)
   IF (CFAIL.EQ.2) GOTO 2019
   IF (CFAIL.NE.0) GOTO 400
   IF (M(I).GT.6) GOTO 2065
   I=M+1
   A(I,1)=C
   FFLAG=3
402 PRINT 2, 'ENTER THE COEFFICIENT FOR Y'
   CALL READER (C, CFAIL)
   IF (CFAIL.NE.0) GOTO 402
   A(I,2)=C
403 PRINT 2, 'ENTER THE COEFFICIENT TO Z'
   CALL READER (C, CFAIL)
   IF (CFAIL.NE.0) GOTO 403
   A(I,3)=C
READ (5,112) SIGN(I)
412 FORMAT (A1)
   X GOTO 410
404 WRITE (6,413)
413 FORMAT (/,' ENTER THE RIGHT HAND SIDE'
   CALL READR (C,CFAIL)
   IF (CFAIL, .NE., 0) GOTO 404
   RS(I):=C
   IF (RS(I),GE,0) GOTO 4200
   PRINT *, ' YOU HAVE A NEGATIVE RIGHT HAND SIDE COEFFICIENT'
   PRINT *, ' RE-ENTER THE LINE'
   GOTO 400
4200 M=M+1
   WRITE (6,430) DO YOU WISH TO ADD ANOTHER CONSTRAINT?,/
X', Y OR N'
   READ (5,32) AA
32 FORMAT (A1)
   IF (AA, .EQ., 'Y') GOTO 400
   GOTO 19
2045 WRITE (6,2046)
2066 FORMAT (IX,' SORRY YOU ALL READY HAVE THE MAXIMUM',/
X' ALLOWABLE NUMBER OF CONSTRAINTS',/
X' REMEMBER, NO MORE THAN 6 EQUATIONS')
   GOTO 19

* * *
THIS SECTION ALLOWS A CONSTRAINT TO BE DELETED
2070 WRITE (6,2074) ENTER THE ROW YOU WISH TO DELETE?,/
X', ' OR ENTER A NULL LINE TO RETURN TO THE EDIT MENU'
2074 FORMAT (/,' ENTER A NULL LINE TO RETURN TO THE EDIT MENU')
   M = 2070 I=1,M
   WRITE (6,2032) I, (A(I,J), J=1,3), SIGN(I),RS(I)
2078 CONTINUE
   CALL READR (C,CFAIL)
   IF (CFAIL, .EQ., 2) GOTO 2019
   IF (CFAIL, .NE., 0) GOTO 2070
   N=C
   WRITE (6,2056) A(N,1),A(N,2),A(N,3),SIGN(N),RS(N)
   PRINT *, ' IS THIS THE ROW YOU WISH TO DELETE?'
   PRINT *, ' Y OR N'
   READ (5,2080) AA
2080 FORMAT (A1)
   IF (AA, .NE., 'Y') GOTO 2070
   M=M-1
2082 DO 2082 I=N,M
   A(I+1,1)=A(I,I+1)
   A(I+1,2)=A(I,I+1,2)
   A(I+1,3)=A(I,I+1,3)
   RS(I)=RS(I+1)
   SIGN(I)=SIGN(I+1)
2082 CONTINUE
   A(M+1,1)=0
   A(M+1,2)=0
   A(M+1,3)=0
   FFLAG=3
   RS(M)=0
   WRITE (6,40)
40 FORMAT (/,' DO YOU WISH TO DELETE ANOTHER CONSTRAINT? /
X', ' Y OR N'
   READ (5,42) AA
42 FORMAT (A1)
   IF (AA, .EQ., 'Y') GOTO 2070
   GOTO 19
2090 CONTINUE
RETURN
END
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
**READ IN CHARACTERS AND CONVERT THEM TO DECIMAL**
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 

SUBROUTINE READER (C,CALFAIL)
CHARACTER*80 X1
CHARACTER ZERO,ONE
INTEGER X
ZERO='0'
CALFAIL=0
C=0
NEG=0
DEC=0
NUM=0
X=CHAR(ZERO)
READ (3,10) X1
10 FORMAT (ABO)
DO 4 J=1,80
MM=J
ONE=X1(MM:)
4 CONTINUE
CALFAIL=2
RETURN
5 I=MM
JJ=MM
ONE=X1(MM:)
IF (ONE.EQ.,' ') GOTO 50
IF (ONE.EQ.,'-' ) GOTO 11
IF (ONE.EQ.,'+' ) GOTO 12
IF (ONE,GE,'0',OR,ONE,LE,'9') GOTO 20
CALFAIL=1
RETURN
11 NEG=1
12 I=MM+1
20 DO 25 II=I,80
JJ=I
ONE=X1(II:)
IF (ONE.EQ.,'-' ) GOTO 40
IF (ONE.EQ.,'+' ) GOTO 30
IF (ONE,LT,'0',OR,ONE,GT,'9') GOTO 35
NUM=NUM+1
25 CONTINUE
30 G=NUM+1
G=G-1
DO 32 IL=I,G
ONE=X1(IL:)
C=(ICCHAR(ONE) - X)*10**GG + C
G=G-1
32 CONTINUE
IF (NEG.EQ.,1) C=-1*C
RETURN
35 CALFAIL =1
RETURN
40 IF (I.EQ.,JJ) GOTO 50
G=NUM+I-1
GG=NUM-1
DO 42 IL=I,G
ONE=X1(IL:)
C=(ICCHAR(ONE) - X)*10**GG + C
G=G-1
CONTINUE

50 JJ=JJ+1
55 DO 55 KK=JJ,80
ONE=XI(KK)
   IF (ONE.EQ.,') GOTO 60
   IF (ONE.LT.,'0' .OR. ONE.GT.,'9') GOTO 35
   DEC=DEC+1
55 CONTINUE
60 G=DEC+JJ-1
   DO 65 IL=JJ,G
   ONE=XI(IL)
   C=(ICHAR(ONE) - X)*10**G+G
   GO=GO+1
65 CONTINUE
   IF (NEQ,EQ,1) C=CI-1
RETURN
END
**SUBROUTINE LPROG**

**THIS PROGRAM CALCULATES THE**

**SOLUTION TO THE LINEAR**

**PROGRAMMING PROBLEM USING**

**THE SIMPLEX ALGORITHM**

**SUBROUTINE LPROG (N,Z,SIGN,A,OBJ,RS,SOLN,CS,FFLAG)**

* DIMENSION A(7,7),AS(12,36),OBJ(3)
* DIMENSION B(6),RS(7),SOLN(16)
* CHARACTER AA,SIGN(10)

**C**

**INITIALIZE**

**C**

FLAG=0
L=0
E=0
G=0
FLAG=0
M=0
R=0
S=0
N=3
LFLAG=0
ITER=1
DO 12 I=1,12
DO 10 NN=1,36
AS(I,NN)=0
10 CONTINUE
12 CONTINUE
DO 15 I=1,16
SOLN(I)=0
15 CONTINUE
DO 5 I=1,6
R(I)=0
5 CONTINUE
DO 30 I=1,M
IF (SIGN(I),EQ,'<') L=L+1
IF (SIGN(I),EQ,'=') E=E+1
IF (SIGN(I),EQ,'>') G=G+1
30 CONTINUE

**C**

CT=N+M+10
CS=CT+1
CU=N+116
MI=M+1
M2=M+2
I=1

**C**

**SORTING**

**C**

DO 40 ROW=1,M
IF (SIGN(ROW),NE,'<') GO TO 40
AS(1,1)=A(ROW,1)
AS(1,2)=A(ROW,2)
AS(1,3)=A(ROW,3)
AS(1,CS)=RS(ROW)
B(I)=ROW
I=I+1
40 CONTINUE
   DO 42 ROW=i,M
   IF (SIGN(ROW),NE,.'>') GO TO 42
   AS(I,1)=A(ROW,1)
   AS(I,2)=A(ROW,2)
   AS(I,3)=A(ROW,3)
   AS(I,CS)=RS(ROW)
   R(I)=ROW
   I=I+1
42 CONTINUE
   DO 44 ROW=i,M
   IF (SIGN(ROW),NE,.'>') GO TO 44
   AS(I,1)=A(ROW,1)
   AS(I,2)=A(ROW,2)
   AS(I,3)=A(ROW,3)
   AS(I,CS)=RS(ROW)
   R(I)=ROW
   I=I+1
44 CONTINUE
   ADD IN SURPLUS AND ARTIFICIAL VARIABLES
   DO 60 I=I,M
   DO 50 J=1,N
   IF (I.E.LE.L) GO TO 50
   AS(M2,J)=AS(M2,J)-AS(I,J)
50 CONTINUE
   IF (I.GT.L) GO TO 52
   AS(I,N+1)=1
   GO TO 60
52 AS(I,N+I+1)=1
   AS(M2,CS)=AS(M2,CS)-AS(I,CS)
   IF (I.E.LE.L) GO TO 60
   AS(I,N+I+1)=1
60 CONTINUE
   MAKE THE OBJECTIVE FUNCTION A MINIMIZE FUNCTION
   AND SET IT TO AS()
   AS(M1,1)=OBJ(1)*-1#Z
   AS(M1,2)=OBJ(2)*-1#Z
   AS(M1,3)=OBJ(3)*-1#Z
   WRITE (6,162)
162 FORMAT ('/'' INITIAL TABLEAU''/)
   DO 165 I=1,M2
   WRITE (6,167) (AS(I,J),J=1,CS)
165 CONTINUE
   WRITE (6,168)
168 FORMAT ('/''2X, 'DO YOU WISH TO SEE THE TABLEAU FOR',/''2X,'EACH ITERATION. Y OR N''/)
   READ(S,169)AA
169 FORMAT (A1)
   IF (AA.EQ.,'Y') I.FLAG=1
   START OF THE SIMPLEX METHOD
   R=0
   DO 67 J=1,CU
   IF (AS(M2,J).GT.0) GOTO 67
   R=AS(M2,J)
   S=J
67 CONTINUE
   IF (R.ER.0) GOTO 100
Q=1.E138
R=0
DO 82 I=1,M
IF (AS(I,S),LE,0.001) GOTO 82
IF (AS(I,CS)/AS(X,S),GT,0) GOTO 82
Q=AS(I,CS)/AS(I,S)
R=
82 CONTINUE
IF (R,GE,0.5) GOTO 84
PRINT *, 'SOLUTION UNBOUNDED'
FLAG=1
GOTO 130
84 P=AS(R,S)
DO 86 J=1,CS
AS(R,J)=AS(R,J)/P
86 CONTINUE
DO 88 I=1,M
IF (I.EQ.R) GOTO 88
DO J=1,CS
IF (J.EQ.S) GOTO 90
AS(I,J)=ABS(I,J)-ABS(R,J)*ABS(I,S)
IF (ABS(I,J),GT,0.00001) GOTO 90
IF (ABS(I,J),LE,0) GOTO 92
88 CONTINUE
AS(R,S)=0
90 CONTINUE
DO 92 I=1,M
92 CONTINUE
AS(R,S)=1
GOTO 110
100 IF (M2,GE,M1+1) GOTO 120
M2=M2-1
IF (AS(M2+1,CS),GT,0.00001) GOTO 110
PRINT *, 'NO FEASIBLE SOLUTION'
FLAG=2
GOTO 130
110 IF (LFLAG.EQ.0) GOTO 165
WRITE (6,115) ITER
115 FORMAT (2X,'AFTER ITERATION ',I4,/) DO 118 II=1,M2
WRITE (6,167) (AS(II,JJ),JJ=1,CS)
118 CONTINUE
ITER=ITER+1
GOTO 65
* END SIMPLEX METHOD *
120 AS(M1,CS)=AS(M1,CS)+Z
WRITE (6,121)
121 FORMAT (/,' ',FINAL TABLEAU',/)
DO 122 I=1,M1
WRITE (6,124) (AS(I,J),J=1,CS)
124 FORMAT (1X,9(F7.2,2X))
122 CONTINUE
MCS=CS-1
K=1
DO 237 J=1,MCS
ZERO=0
IF (AS(M1,J),NE,0) GOTO 235
DO 234 I=1,M
IF (AS(I,J),EQ,0) ZERO=ZERO+1
234 CONTINUE
IF (ZERO,NE,M) GOTO 236
DO 238 I=1,M
IF (AS(I,J),EQ,1) SOLN(K)=AS(I,CS)
238 CONTINUE
GOTO 235
236 IF (J.LE,3) PRINT *, 'ALTERNATE OPTIMA EXIST'
215 K=K+1
216 CONTINUE
SOLN(CS)=AS(M1,CS)
WRITE (6,240) (SOLN(I),I=1,3)
240 FORMAT (//,lx,'X=',F10.2,/,lx,'Y=',F10.2,/,lx,'Z=',F10.2)
DO 242 I=4,MCS
WRITE (6,241) I-3,SOLN(I)
241 FORMAT (lx,'S',I2,'=',F10.2)
242 CONTINUE
WRITE (6,244) SOLN(CS)
244 FORMAT (//,lx,'THE SOLUTION IS ',F10.2)

** DUAL OPTION **

DO 248 FORMAT (//,2x,'DO YOU WISH TO SEE THE SOLUTION TO X THE DUAL Y OR N?')
READ (5,1248) AA
1248 FORMAT (A1)
IF (AA.EQ.'Y') GOTO 300
DO 250 I=4,MCS
WRITE (6,249) I-3,AS(M1,I)
249 FORMAT (lx,'Y',I2,'=',F10.2)
250 CONTINUE
DO 252 I=1,3
WRITE (6,251) I,AS(M1,I)
251 FORMAT (lx,'S',I2,'=',F10.2)
252 CONTINUE
WRITE (6,302)
302 FORMAT (2x,'DO YOU WISH TO RUN THE SENSITIVITY ANALYSIS? Y OR N')
READ (5,304) AA
304 FORMAT (A1)
IF (AA.EQ.'Y') GOTO 130
CALL SENS (AS,M,CSS,LE,EG,OBJ,SOLN,M1,MCS,RS,Z,R)
130 CONTINUE
RETURN
END
SUBROUTINE SENS

CALL X

THIS SUBROUTINE PERFORMS A SENSITIVITY ANALYSIS ON AN LP PROBLEM.

SUBROUTINE SENS (AS, M, CS, L, E, G, OBJ, SOLN, MI, MCS, RS, Z, B)

DIMENSION OBJ(J), AS(12, 36), SOLN(16), OBJ(9)

DIMENSION R(S, 4), AAS(9, 9), RS(7)

CHARACTER CHAR(3)

A1NCR = 0

DECR = 0

MSG = MCS - 6

DO 3 J = 1, 6

OBJ(J) = 0

3 CONTINUE

DO 5 KK = 1, 3

OBJ(KK) = OBJ(KK) * 1 * Z

DO 6 KK = 1, 9

OBJ(KK) = OBJ(KK)

6 CONTINUE

DO 210 J = 1, MSG

DO 208 I = 1, MI

AAS(I, J) = AS(I, J)

208 CONTINUE

210 CONTINUE

WRITE (6, 8)

8 FORMAT (/ , 2X, 'RANGE IN WHICH BASIS IS UNCHANGED' , // )

WRITE (6, 12)

12 FORMAT (2X, 'VARIABLE', 10X, 'CURRENT', 9X, 'ALLOWABLE', 9X, 'SALLOWABLE')

WRITE (6, 13)

13 FORMAT (22X, 'COEF', 11X, 'DECREASE', 10X, 'INCREASE', '//' )

CHAR(1) = 'X'

CHAR(2) = 'Y'

CHAR(3) = 'Z'

DO 100 I = 1, 3

IF (SOLN(I) .NE. 0) GOTO 50

DELTA = AS(M, I)

IF (Z, EQ, -1) GOTO 15

WRITE (6, 10) CHAR(I), OBJ(I) * 1 * Z, DELTA

10 FORMAT (4X, AL, 5X, E15.2, 10X, 'INFINITY', 3X, E15.2)

GOTO 100

15 WRITE (6, 20) CHAR(I), OBJ(I) * 1 * Z, DELTA

20 FORMAT (4X, AL, 5X, E15.2, 3X, E15.2, 9X, 'INFINITY')

GOTO 100

50 ALES = 1E16

AGRET = 1E16

DO 55 LL = 1, M

IF (AS(LL, I), EQ, 1) IP=LL

55 CONTINUE

DO 90 J = 1, MSG

IF (I, EQ, J) GOTO 90

IF (AAS(IP, J), EQ, 0) GOTO 90

ZI = OBJ(J) - AAS(MI, J)

VAL1 = ZI - OBJ(J) / AAS(IP, J)

ZI = OBJ(J) - VAL1

VAL2 = ZI / AAS(IP, J)

IF (AAS(IP, J), LT, 0) GOTO 60

CREDIT = VAL2 / AAS(IP, J)

IF (CREDIT, LT, AGRET) AGRET = CREDIT

GOTO 90

60 CLES = VAL2 / AAS(IP, J)
IF (CLES.GT.ALES) ALES=CLES
90 CONTINUE
   IF (AGRET.NE.1.E16) AINCR=ABS(OBJ(I)-AGRET)
   IF (ALES.NE.-1.E16) DECR=ABS(ALES-OBJ(I))
   IF (Z.WE.1) GOTO 92
   TEMP=DECR
   DECR=AINCR
   AINCR=TEMP
92 IF (AGRET.EQ.1.E16) GOTO 98
   IF (ALES.EQ.-1.E16) GOTO 95
   WRITE (6,F1) CHAR(I),OBJ(I)*-1*Z,DECR,AINCR
91 FORMAT (4X,A1,5X,E15.2,3X,E15.2,3X,E15.2)
   GOTO 100
95 WRITE (6,96) CHAR(I),OBJ(I)*-1*Z,DECR
96 FORMAT (4X,A1,5X,E15.2,3X,E15.2,9X,'INFINITY')
   GOTO 100
98 WRITE (6,99) CHAR(I),OBJ(I)*-1*Z,AINCR
99 FORMAT (4X,A1,5X,E15.2,10X,'INFINITY',3X,E15.2)
100 CONTINUE
   WRITE (6,200)
200 FORMAT (/4,2X,'RIGHT HAND SIDE ANALYSIS',/)
   WRITE (6,202)
202 FORMAT (2X,'CONSTRAINT',9X,'CURRENT',9X,'ALLOWABLE',9X,'ALLOWABLE')
   WRITE (6,204)
204 FORMAT (24X,'RHS',11X,'DECREASE',10X,'INCREASE',/)
*   K=1
   DO 210 J=4,MSG
      BHIGH=1.E16
   DO 225 I=1,M
      IF (AAS(I,J).EQ.0) GOTO 225
      IF (AAS(I,J).LT.0) GOTO 220
      AL=ABS(I,CS)/ABS(I,J)
      IF (AL.GT.BL) BLOW=ALOW
      GOTO 225
   ALHIGH=ABS(I,CS)/ABS(I,J)
   IF (ALHIGH.LT.BHIGH) BHIGH=ALHIGH
225 CONTINUE
*   BLOW=ABS(BLOW)
   MMSG=MAG
   IF (J=J.GT.L.AND.J.GT.L.E.MMSG.E-3) K=J-34E3
   IF (J.GT.L.E.MMSG.E-3) K=J-3-G
   IF (J.GT.L.AND.J.GT.L.E.E-3) GOTO 222
   LM=B(K)
   TEMP=BLOW
   BLOW=BNHIGH
   BHIGH=TEMP
   IF (BL.GT.EQ.1.E16) GOTO 214
   IF (BHIGH.EQ.-1.E16) GOTO 216
   LM=B(K)
   IF (BHIGH.EQ.1.E16) GOTO 216
   IF (BL.GT.EQ.-1.E16) GOTO 214
   WRITE (6,212) LM,RS(LM),BNHIGH,BLOW
212 FORMAT (5X,13,5X,E15.2,2,3X,E15.2)
   GOTO 230
214 WRITE (6,213) LM,RS(LM),BNHIGH
215 FORMAT (5X,13,5X,E15.2,2,3X,E15.2,9X,'INFINITY')
   GOTO 230
216 WRITE (6,217) LM,RS(LM),BLOW
217 FORMAT (5X,13,5X,E15.2,10X,'INFINITY',3X,E15.2)
230 K=K+1
   CONTINUE
   OBJ(I)=OBJ(I)*-1*Z
   OBJ(2)=OBJ(2)*-1*Z
OBJ(3)=OBJ(3)*-1*Z
RETURN
END
SUBROUTINE DPLT (M,Z,SIGN,A,OBJ,RS,SOLN,CS,FLAG)

DIMENSION ZERO(6),A(7,7),OBJ(3),FT(30)
DIMENSION PLT(7),RS(7),EXPT(100,3),FSPE(100,3)
DIMENSION SMTX(3,7),PK(30),RS2(4),A2(4,4)
DIMENSION SOLN(16),FSPT2(100,3)
DIMENSION FPT(100,3),FPT2(100,3),FPT3(100,3)
DIMENSION FP4A(100,3)
INTEGER*2 INBUF(15),IFILE(7),ICHAR(4),ICHA
INTEGER*4 ISHAPE(12),REGNS0,LINE(6),IORG
INTEGER*4 SHAPE
INTEGER*2 IOG(3),IATLK(4),IRC,ILEY,ISPEC(5)
CHARACTER SIGN(10)
INTEGER R, S, FLAG
DATA IGB/2*0/,ISPEC/5*0/,ICOLOR/0/
DATA SHAPE/4/,IATLK/4*0/,IEX/"EX"/
DATA ICOLOR/0/
DATA REGNS0/6ARREGION/

INITIALIZE

IORG=0
FLAG=0
ICOLOR=0
JKOUNT=1
IKOUNT=1
LCOUNT=1
MKOUNT=1
NKOUNT=1
BMAX=0
R=0
S=0
M3=0
N=3

OPEN A PREVIOUSLY CREATED DESIGN FILE AND ACCESS THE
DFPI PACKAGE

WRITE (6,1)
1 FORMAT ("ENTER DESIGN FILE SPEC IF NAME.DGN",\X2X,'YOU MUST ENTER A FILE NAME, PERIOD, DGN',\X2X,'IF NAME IS DOG',\X2X,'YOU SHOULD ENTER',\X2X,'DUO,DGN',\X2X,'NOW ENTER DESIGN FILESPEC',\X2X)\READ (S,2) ICHA,INBUF
2 FORMAT (0,1542)
PRINT *,"YOUR DESIGN FILE IS BEING OPENED -\XPLEASE WAIT"\CALL LKTC3I (INBUF,IFILE,ICHA,0,IRC)\IF (IRC.NE.0) STOP 'LKTC3I CONVERSION ERROR'\CALL ENDFPI (REGNS0,IFILE,0,0,0,0,0,0,IRC,IE)
IF (IRC.NE.0) GOTO 2005
WRITE (6, 4)
FORMAT (' Initial Complete')
PRINT *, ' Enter the starting level of your design file ' X(1, 15, 30, 0R 45)
READ (5, *) LEV
PRINT *, ' Calculating' TLEV = LEV
DO 15 I = 1, M
PLT(LM) = 0
15 CONTINUE
DO 17 I = 1, 100
DO 16 J = 1, 3
EXPT(I, J) = 0
FSPT(I, J) = 0
FSPT2(I, J) = 0
FPI1(I, J) = 0
FPT2(I, J) = 0
FPT3(I, J) = 0
16 CONTINUE
17 CONTINUE
DO 20 I = 1, 30
PK(I) = 0
PT(I) = 0
20 CONTINUE
DO 22 I = 1, 6
ZERO(I) = 0
22 CONTINUE
**INITIAL SOLUTION PLANES**

JK = 2

DO 210 I = 1, M
DO 211 IJ = 1, 3
211 PT(IJ) = 0
DO 215 J = 1, N
IF (A(I, J), EQ, 0) GOTO 212
PT(J) = RS(I) / A(I, J)
EXPT(JK, J) = PT(J)
212 JK = JK + 1
IF (EXPT(JK - 1, J), EQ, 0) ZERO(I) = ZERO(I) + 1
215 CONTINUE
AMAX = ABS(PT(1))
IF (ABS(PT(2)), GT, AMAX) AMAX = ABS(PT(2))
IF (ABS(PT(3)), GT, AMAX) AMAX = ABS(PT(3))
IF (AMAX, GT, BMAX) BMAX = AMAX
PK(I) = JK - 3
210 CONTINUE
III = 1
FACT = 4000 / BMAX
KJJ = JK - 1
LH = 1
DO 216 K = 2, KJJ, 3
K1 = I
K3 = I + 2
L1 = 1
DO 214 K = K1, K3
LIL = 1
DO 213 J = 1, 3
ISHAPE(LIL) = EXPT(K, J) * FACT + IORG
LIL = LIL + 1
213 CONTINUE
214 CONTINUE
ISHAPE(10) = EXPT(K1, 1) * FACT + IORG
ISHAPE(11) = EXPT(K1, 2) * FACT1* ORG
ISHAPE(12) = EXPT(K1, 3) * FACT1* ORG
ILEV = ILEV + 1
ICOLOR = (COLOR + 1
ISPEC(S) = ICOLOR
NNUL = ZERO(I)
* II = II + 1
IF (SIGN(II).EQ., '<') CALL LESS (ISHAPE, IGG, ILEV, ISPEC, XIRC, IATLK)
IF (IRC.NE.0) GOTO 12
IF (SIGN(II).EQ., '>') CALL GREAT (ISHAPE, IGG, ILEV, ISPEC, IRC, XATLK, ISHAPE, NNUL)
IF (IRC.NE.0) GOTO 12
* IF (NNUL, EQ, 0) CALL SHDFP1 (ISHAPE, IGG, ILEV, ISPEC, ISHAPE, SHAPE, XIRC, IATLK)
IF (IRC.NE.0) GOTO 10
* IF (NNUL, EQ, 1) CALL LINES (ISHAPE, IGG, ILEV, ISPEC, IRC, XATLK, ISHAPE)
IF (IRC.NE.0) GOTO 12
IF (NNUL, GE, 2) CALL POINTS (ISHAPE, IGG, ILEV, ISPEC, IRC, XATLK, ISHAPE)
IF (IRC.NE.0) GOTO 12

II = II + 1
CALL EX1 (ISHAPE, IGG, ILEV, ISPEC, ISHAPE, IRC, IATLK)
IF (IRC.NE.0) GOTO 12
216 CONTINUE
KOUNT = JK
*
IF (IFLAG, EQ, 1) GOTO 2030
IF (IFLAG, EQ, 2) GOTO 2035
IF (IFLAG, EQ, 3) GOTO 2038
*
** FIND THE INTERSECTIONS BETWEEN TWO PLANES
**
MM = M - 1
DO 224 II = 1, MM
IJ = II + 1
DO 220 JJ = II, M
II = II
I2 = JJ
CALL FNDPT (II, XI2, EXPT, KOUNT, RS, A)
PLT(1) = PX(II)
PLT(2) = PK(JJ)
CALL CHCKPT (PLT, EXPT, LCOUNT, A, SIGN, RS, FSPT, M)
CALL CHCK2 (A, EXPT, FSPT, RS, SIGN, JK, KOUNT, LCOUNT, M)
IF (JJ, LT, 2) GOTO 273
NN = 2
CALL CHCKPT(PLT, EXPT, JKOUNT, A, SIGN, RS, FPT1, MN)
CALL CHCK2(A, EXPT, FPT1, RS, SIGN, JK, KOUNT, JKOUNT, MN)
273 IF (JJ, GT, 3) GOTO 274
MN = 3
CALL CHCKPT(PLT, EXPT, IKOUNT, A, SIGN, RS, FPT2, MN)
CALL CHCK2(A, EXPT, FPT2, RS, SIGN, JK, KOUNT, IKOUNT, MN)
274 IF (JJ, GT, 4) GOTO 276
MN = 4
CALL CHCKPT(PLT, EXPT, MKOUNT, A, SIGN, RS, FPT3, MN)
CALL CHCK2(A, EXPT, FPT3, RS, SIGN, JK, KOUNT, MKCOUNT, MN)
276 IF (JJ, GT, 5) GOTO 280
MN = 5
CALL CHCKPT(PLT, EXPT, NKOUNT, A, SIGN, RS, FPT4, MN)
CALL CHCK2(A, EXPT, FPT4, RS, SIGN, JK, KOUNT, NKOUNT, MN)
280 CONTINUE
*
JK = KOUNT
MARK=LCOUNT
FLAG=0
320 CONTINUE
324 CONTINUE
*
FIND THE INTERSECTION OF THREE PLANES
*
MM=M-2
DO 230 II=1,MMM
III=II+i
DO 237 JJ=II,MM
IN=(J+1)
DO 234 KK=IK,M
DO 231 LKJ=I,3
A2(1,LKJ)=A2(II,LKJ)
A2(2,LKJ)=A2(JJ,LKJ)
A2(3,LKJ)=A2(KK,LKJ)
231 CONTINUE
RS2(1)=RS(II)
RS2(2)=RS(JJ)
RS2(3)=RS(KK)
230 IC=1
IR=1
IF (A2(1,1),NE.0) GOTO 232
IR=2
IF (A2(2,1),NE.0) GOTO 236
IR=3
IF (A2(3,1),NE.0) GOTO 236
GOTO 256
232 IC=2
IR=2
IF (A2(2,2),NE.0) GOTO 234
IR=1
IF (A2(1,2),NE.0) GOTO 236
IR=3
IF (A2(1,3),NE.0) GOTO 236
GOTO 256
234 IC=3
IR=3
IF (A2(3,3),NE.0) GOTO 240
IR=1
IF (A2(1,3),NE.0) GOTO 236
IR=2
IF (A2(2,3),NE.0) GOTO 236
GOTO 256
C
236 DO 237 J=1,3
A2(4,J)=A2(IC,J)
A2(IC,J)=A2(IR,J)
A2(IR,J)=A2(4,J)
237 CONTINUE
RS2(4)=RS2(IC)
RS2(IC)=RS2(IR)
RS2(IR)=RS2(4)
GOTO 230
C
240 K=4
DO 244 I=1,3
DO 242 J=1,3
SMTX(I,J)=A2(I,J)
242 CONTINUE
SMTX(I,7)=RS2(I)
SMTX(I,K)=1
K=K+1
244 CONTINUE
C
R=1
S=1
250 F=SMTX(R,S)
    IF (ABS(F),LT,0.00001) GOTO 256
    DO 251 J=1,7
    SMTX(R,J)=SMTX(R,J)/F
251 CONTINUE
    DO 254 I=1,3
    IF (I.EQ.R) GOTO 254
    DO 252 J=1,7
    IF (J.EQ.S) GOTO 252
    SMTX(I,J)=SMTX(I,J)-SMTX(R,J)*SMTX(I,S)
252 CONTINUE
254 CONTINUE
    SMTX(I,S)=0
256 CONTINUE
    R=R+1
    S=S+1
    IF (R.LE.3) GOTO 250
    EXPT(KOUNT,1)=SMTX(1,7)
    EXPT(KOUNT,2)=SMTX(2,7)
    EXPT(KOUNT,3)=SMTX(3,7)
    KOUNT=KOUNT+1
    IF (KK,GT,3) GOTO 290
    MN=3
    CALL sockaddr(A,EXPT,FPT2,RS,SIGN,JK,KOUNT,JKOUNT,NN)
290 IF (KK,GT,4) GOTO 292
    MN=4
    CALL sockaddr(A,EXPT,FPT3,RS,SIGN,JK,KOUNT,KMOUNT,NN)
292 MN=5
    CALL sockaddr(A,EXPT,FPT4,RS,SIGN,JK,KOUNT,KMOUNT,NN)
295 CALL sockaddr(A,EXPT,FSPT,RS,SIGN,JK,KOUNT,LCOUNT,M)
    JK=KOUNT
    MARK=LCOUNT
    FLG=0
256 CONTINUE
257 CONTINUE
258 CONTINUE
    EXPT(KOUNT,1)=0
    EXPT(KOUNT,2)=0
    EXPT(KOUNT,3)=0
    KOUNT=KOUNT+1
    IF (M,LE,2) GOTO 310
777 MN=2
    CALL sockaddr(A,EXPT,FPT1,RS,SIGN,JK,KOUNT,JKOUNT,NN)
    IF (M,LE,3) GOTO 310
777 MN=3
    CALL sockaddr(A,EXPT,FPT2,RS,SIGN,JK,KOUNT,KMOUNT,NN)
    IF (M,LE,4) GOTO 310
777 MN=4
    CALL sockaddr(A,EXPT,FPT3,RS,SIGN,JK,KOUNT,KMOUNT,NN)
    IF (M,LE,5) GOTO 310
777 MN=5
310 CALL sockaddr(A,EXPT,FSPT,RS,SIGN,JK,KOUNT,LCOUNT,M)
* ** GRAPH THE SOLUTION SOLID
           ILEV=LEV
           LC=LCOUNT
           CALL ORDIRT(LCOUNT,FSPT,ILEV,IGB,ISPEC,IRC,IATLK,A,RS,FL
           IF (IRC,NE,0) GOTO 2012
124 888 MMM=M-1
           IF (MMM,LE,0) GOTO 314
           ILEV=LEV+MMM
           CALL ORDIRT(LC,FSPT,ILEV,IGB,ISPEC,IRC,IATLK,A,RS,FACT,M
           IF (IRC,NE,0) GOTO 2012
* ** GRAPH THE OPTIMAL POINT
314 ISPEC(4)=10
ISPEC(5) = 0
ILEV = LEV
DO 320 J = 1, 3
DO 315 I = 1, 3
LINE(I) = SOLN(I) * FACT
LINE(I) = SOLN(I) * FACT
315 CONTINUE
LINE(J) = LINE(J) + 50
LINE(J+3) = LINE(J+3) + 50
CALL LINDFPI(IGG, ILEV, ISPEC, LINE, IRC, IATLK)
IF (IRC .NE. 0) GOTO 2012
320 CONTINUE
ISPEC(4) = 0

*** GRAPH THE EVOLUTION OF THE SOLUTION SOLID ***
ILEV = LEV + 2
IF (JKOUNT, LT, 3) GOTO 322
MN = 3
CALL ORDPT(JKOUNT, FPT1, ILEV, IGG, ISPEC, IRC, IATLK, A, RS, XFACT, MN)
IF (IRC .NE. 0) GOTO 2010
322 ILEV = ILEV + 1
IF (JKOUNT, LT, 3) GOTO 324
MN = 3
CALL ORDPT(JKOUNT, FPT2, ILEV, IGG, ISPEC, IRC, IATLK, A, RS, XFACT, MN)
IF (IRC .NE. 0) GOTO 2010
324 ILEV = ILEV + 1
IF (MKOUNT, LT, 3) GOTO 326
MN = 4
CALL ORDPT(MKOUNT, FPT3, ILEV, IGG, ISPEC, IRC, IATLK, A, RS, XFACT, MN)
IF (IRC .NE. 0) GOTO 2010
326 ILEV = ILEV + 1
IF (MKOUNT, LT, 3) GOTO 330
MN = 5
CALL ORDPT(MKOUNT, FPT4, ILEV, IGG, ISPEC, IRC, IATLK, A, RS, XFACT, MN)
IF (IRC .NE. 0) GOTO 2010
330 CONTINUE

*** GRAPH THE OPTIMAL SOLUTION PLANE ***
ISPEC(4) = 5
ISPEC(5) = ISPEC(5) + 13
ILEV = LEV + 12
DO 272 I = 1, 12
ISHAPE(I) = 0
272 CONTINUE
IF (OBJ(1), ER, 0) GOTO 332
ISHAPE(1) = SOLN(CS)/OBJ(1) * FACT
332 IF (OBJ(2), ER, 0) GOTO 334
ISHAPE(5) = SOLN(CS)/OBJ(2) * FACT
334 IF (OBJ(3), ER, 0) GOTO 336
ISHAPE(9) = SOLN(CS)/OBJ(3) * FACT
336 IF (OBJ(1), ER, 0) GOTO 338
ISHAPE(10) = SOLN(CS)/OBJ(1) * FACT
338 CONTINUE
CALL SHDFPI(IGG, ILEV, ISPEC, ISHAPE, SHAPE, IRC, IATLK)
IF (IRC, NE, 0) GOTO 2010
CALL EXT(IGG, ILEV, ISPEC, ISHAPE, IRC, IATLK)
IF (IRC, NE, 0) GOTO 2012

*** ALTERNATE OPTIMAL SOLUTION ***
ILEV = ILEV + 1
CMAX = ABS(ISHAPE(1))
IF (ABS(ISHAPE(5)).GT.CMAX) CMAX=ABS(ISHAPE(5))
IF (ABS(ISHAPE(9)).GT.CMAX) CMAX=ABS(ISHAPE(9))
BFACT=4000/CMAX
DO 350 I=1,4
   ISHAPE(I)=ISHAPE(I)*BFACT
350 CONTINUE
ISHAPE(10)=ISHAPE(1)
CALL SDFPFI (IGG, ILEV, ISPEC, ISHAPE, SHAPE, IRC, IATLK)
IF (IRC NE 0) GOTO 2010
CALL EXG (IGG, ILEV, ISPEC, ISHAPE, IRC, IATLK)
IF (IRC NE 0) GOTO 2012
ISPEC(4)=0
GOTO 2020

2030 WRITE (6,2032)
2032 FORMAT ('/1X, 'THE SIMPLEX METHOD SHOWS THE SOLUTION',/1X, 'TO YOUR PROBLEM IS UNBOUNDED, ONLY THE CONSTRAINTS', 1X, 'WILL BE BE GRAPHED.' )
GOTO 2020

2035 WRITE (6,2036)
2036 FORMAT ('/1X, 'THE SIMPLEX METHOD SHOWS NO FEASIBLE',/1X, 'SOLUTION, ONLY THE CONSTRAINTS WILL BE GRAPHED.' )
GOTO 2020

2038 WRITE (6,2039)
2039 FORMAT ('/1X, 'PLEASE RUN THE SIMPLEX ALGORITHM',/1X, 'ONLY THE CONSTRAINTS WILL BE SHOWN.' )
GOTO 2020

* CLOSE THE DESIGN FILE *

2005 WRITE (6,2006) IRC
2006 FORMAT ('/ ERROR IN LKTC81 IRC = ',I3)
GOTO 2019

2010 WRITE (6,2011) IRC
2011 FORMAT ('/ ERROR IN SDFPFI IRC = ',I3)
GOTO 2019

2012 WRITE (6,2012) IRC
2013 FORMAT ('/ ERROR IN LDFFP1 IRC = ',I3)
GOTO 2019

2014 WRITE (6,2015) IRC
2015 FORMAT ('/ ERROR IN ELDFFP1 IRC = ',I3)
2019 CONTINUE
   IF (IRC GE 900) CALL ERROR9 (IRC)
   CALL ERROR
2020 CONTINUE
   IARG=1
   IF (IRC NE 0) IARG=0
   CALL DFFPFI (IARG)
   CALL FINDFX
   PRINT 2, 'DFPI COMPLETE'
RETURN
END

************************************************************
***  SUBROUTINES  ***
************************************************************
***  LESS  ***
※ THIS SUBROUTINE PLOTS THE LOGICAL OPERATOR EXTENSION LINES ※
FOR THE CONSTRAINT PLANES ※

SUBROUTINE LESS (ISHAPE, IGG, ILEV, ISPEC, IRC, IATLK)
INTEGER*4 LINE(6), ISHAPE(12)
INTEGER*2 IGG(2), IATLK(4), IRC, ILEV, ISPEC(5)
DO 15 J=1,9,3
DO 10 I=1,6
10 LINE(I)=0
** GREAT  **********
**
** THIS SUBROUTINE PLOTS THE LOGICAL OPERATOR EXTENSION
** LINES FOR THE CONSTRAINT PLANES WITH ' > ' OPERATORS
**
```fortran
SUBROUTINE GREAT (IGG, ILEV, ISPEC, IRC, IATLK, ISHAPE, NNUL)
INTEGER*4 IGG(2), IATLK(4), IRC, ILEV, ISPEC(3)
INTEGER*4 ISHAPE(12), LINE(6)
IF (NNUL .EQ. 3) GOTO 30
IS = 0
DO 25 J = 1, 9, 3
IF (ISHAPE(J), EQ, 0) GOTO 24
DO 10 I = 1, 6
10 LINE(I) = 0
LINE(1) = ISHAPE(J)
LINE(2) = ISHAPE(J + 1)
LINE(3) = ISHAPE(J + 2)
IF (ISHAPE(J), GT, 0) LINE(4) = 4000
IF (ISHAPE(J), LT, 0) LINE(4) = -4000
IF (ISHAPE(J + 1), GT, 0) LINE(5) = 4000
IF (ISHAPE(J + 1), LT, 0) LINE(5) = -4000
IF (ISHAPE(J + 2), GT, 0) LINE(6) = 4000
IF (ISHAPE(J + 2), LT, 0) LINE(6) = -4000
CALL LDNDFPI (IGG, ILEV, ISPEC, LINE, IRC, IATLK)
24 IS = IS + 1
25 CONTINUE
RETURN
30 DO 35 I = 1, 6
35 CONTINUE
DO 40 I = 1, 3
LINE(I + 3) = 4000
CALL LDNDFPI (IGG, ILEV, ISPEC, LINE, IRC, IATLK)
IF (IRC, NE, 0) RETURN
LINE(I + 3) = 0
40 CONTINUE
RETURN
END
```

****** LINES **********
**
** THIS SUBROUTINE DRAWS THE FEASIBLE REGION OF
** A TWO VARIABLE CONSTRAINT
**
```fortran
SUBROUTINE LINES (IGG, ILEV, ISPEC, IRC, IATLK, ISHAPE)
INTEGER*4 IGG(2), IATLK(4), IRC, ILEV, ISPEC(3)
INTEGER*4 DI(6), ISHAPE(12)
I = 1
DO 10 J = 1, 6
10 DLINE(J) = 0
IF (ISHAPE(J), EQ, 0) GOTO 30
DLINE(I) = ISHAPE(J)
I = I + 1
30 IF (ISHAPE(5), EQ, 0) GOTO 35
DLINE(I + 1) = ISHAPE(5)
I = I + 1
35 IF (ISHAPE(9), EQ, 0) GOTO 40
DLINE(I + 2) = ISHAPE(9)
I = I + 1
40 CALL LDNDFPI (IGG, ILEV, ISPEC, DLINE, IRC, IATLK)
```
IF (IRC .NE. 0) RETURN

** EXTENSION LINES **

IF (ISHAPE(I), EQ, 0) PT=1
IF (ISHAPE(5), EQ, 0) PT=2
IF (ISHAPE(9), EQ, 0) PT=3
DO 50 J=1,3
    IF (I.EQ,PT) GOTO 50
    J=I
    IF (I.EQ,2) J=4
    IF (I.EQ,3) J=7
DLINE(1)=ISHAPE(J)
DLINE(2)=ISHAPE(J+1)
DLINE(3)=ISHAPE(J+2)
DLINE(4)=ISHAPE(J)
DLINE(5)=ISHAPE(J+1)
DLINE(6)=ISHAPE(J+2)
DLINE(PT)=4000
CALL LNDPFI (Igg, Ilev, Ispec, Dline, IRC, IatlK)
IF (IRC. NE. 0) RETURN
50 CONTINUE
RETURN
END

** POINTS **********

THIS SUBROUTINE PLACES A POINT FOR A ONE VARIABLE CONSTRAINT AND FOR CONSTRAINTS WITH THE RHS VALUE OF ZERO

SUBROUTINE POINTS (Igg, Ilev, Ispec, IRC, IatlK, Ishape)
INTEGER Igg(2), IatlK(4),IRC, Ilev, Ispec(5)
INTEGER Ishape(12), Line(6)
DO 25 I=1,6
25 Lline(I)=0
Line(1)=ISHAPE(1)
Line(2)=ISHAPE(5)
Line(3)=ISHAPE(9)
Line(4)=ISHAPE(1)
Line(5)=ISHAPE(5)
Line(6)=ISHAPE(9)
DO 50 J=1,3
    IF (Line(J), NE, 0) GOTO 50
    Line(J+3)=4000
CALL LNDPFI (Igg, Ilev, Ispec, Line, IRC, IatlK)
IF (IRC. NE. 0) RETURN
50 CONTINUE
RETURN
END

** EXT **********

THIS SUBROUTINE EXTENDS CONSTRAINT LINES FOR CONSTRAINTS WITH NEGATIVE COEFFICIENTS

SUBROUTINE EXT (Igg, Ilev, Ispec, Ishape, IRC, IatlK)
INTEGER Igg(2), IatlK(4),IRC, Ilev, Ispec(5)
INTEGER Ishape(6), Ishape(12)
NUM=0
DO 10 I=1,9,4
    IF (ISHAPE(I), LT, 0) NUM=NUM+1
10 CONTINUE
IF (NUM. EQ. 0) RETURN
IF (NUM, EQ, 3) RETURN
IF (NUM, EQ, 2) GOTO 30
DO 12 J=1,9,4
    IF (ISHAPE(J), LT, 0) I=J
12 CONTINUE
16    DO 15 J=1,9,4
17      IF (I.EQ.J) GOTO 15
    DO 17 N=1,6
17    LINE(N)=0
      IF (J.EQ.1) LINE(1)=ISHAPE(J)
      IF (J.EQ.5) LINE(2)=ISHAPE(J)
      IF (J.EQ.9) LINE(3)=ISHAPE(J)
      IF (ISHAPE(J).EQ.0) GOTO 15
      LN=ISHAPE(I)*(ISHAPE(J)-4000)/ISHAPE(J)
      IF (I.EQ.1) LINE(4)=ABS(LN)
      IF (I.EQ.5) LINE(5)=ABS(LN)
      IF (J.EQ.1) LINE(6)=4000
      IF (J.EQ.5) LINE(6)=4000
      CALL LNDPFI (IJB, ILEV, ISPEC, LINE, IRC, IATLK)
      IF (IRC. NE. 0) RETURN
15 CONTINUE
    GOTO 40
30    DO 22 I=1,9,4
22      IF (ISHAPE(I).GE.0) J=I
22 CONTINUE
    DO 35 I=1,9,4
      IF (I.EQ.J) GOTO 35
      DO 37 N=1,6
37    LINE(N)=0
      IF (J.EQ.1) LINE(1)=ISHAPE(J)
      IF (J.EQ.5) LINE(2)=ISHAPE(J)
      IF (J.EQ.9) LINE(3)=ISHAPE(J)
      IF (ISHAPE(J).EQ.0) GOTO 25
      LN=ISHAPE(I)*(ISHAPE(J)-4000)/ISHAPE(J)
      IF (I.EQ.1) LINE(4)=ABS(LN)
      IF (I.EQ.5) LINE(5)=ABS(LN)
      IF (J.EQ.1) LINE(6)=4000
      IF (J.EQ.5) LINE(6)=4000
      CALL LNDPFI (IJB, ILEV, ISPEC, LINE, IRC, IATLK)
      IF (IRC. NE. 0) RETURN
35 CONTINUE
40 CONTINUE
RETURN
END

**** FNDPT ****

* THIS SUBROUTINE FINDS THE INTERSECTING POINTS *
* BETWEEN TWO PLANES  *

SUBROUTINE FNDPT (II, I2, EXPT, KOURT, RS, A)
DIMENSION EXPT(100,3), RS(7), A(7,7)
DIMENSION SMTX(3,5)
INTEGER RS, 5, FLAG
DO 412 I=1,3
   DO 410 J=1,5
      SMTX(I,J)=0
410 CONTINUE
412 CONTINUE
   DO 454 II=1,2
      DO 452 JJ=2,3
         NN=II
         DO 420 N=1,2
            IF (II.EQ.JJ) GOTO 452
            SMTX(1,N)=A(II,NN)
            SMTX(2,N)=A(I2,NN)
            NN=JJ
420 CONTINUE
IF (SMTX(1, 1) .NE. 0.0 OR SMTX(2, 2) .NE. 0) GOTO 424
IF (SMTX(1, 2) .NE. 0.0 OR SMTX(2, 1) .NE. 0) GOTO 422
EXPT (KOUNT, 1) = 0
EXPT (KOUNT, 2) = 0
GOTO 450
422
SMTX(1, 1) = 1
SMTX(2, 2) = 1
SMTX(1, 3) = RS(12)
SMTX(2, 5) = RS(11)
K = 1
S = 2
GOTO 425
424
SMTX(1, 3) = 1
SMTX(2, 4) = 1
SMTX(1, 5) = RS(11)
SMTX(2, 5) = RS(12)
R = 1
S = 1
FLAG = 1
425
LPASS - 1
426
P = SMTX(R, S)
IF (ABS(P) .LT. 0.00001) GOTO 451
DO 427 J = 1, 5
SMTX(R, J) = SMTX(R, J) / P
427
CONTINUE
DO 432 I = 1, 2
IF (I .EQ. R) GOTO 432
DO 430 J = 1, 5
IF (J .EQ. S) GOTO 430
SMTX(1, J) = SMTX(1, J) * SMTX(R, J) * SMTX(I, S)
430
CONTINUE
SMTX(I, 5) = 0
432
CONTINUE
R = R + 1
S = J - S
IF (R .LE. 2) GOTO 426
IF (FLAG .NE. 1) GOTO 349
EXPT(KOUNT, II) = SMTX(1, 5)
EXPT(KOUNT, JJ) = SMTX(2, 5)
GOTO 450
348
EXPT(KOUNT, II) = SMTX(2, 5)
EXPT(KOUNT, JJ) = SMTX(1, 5)
450
KOUNT = KOUNT + 1
451
SMTX(1, 3) = 0
SMTX(1, 4) = 0
SMTX(2, 3) = 0
SMTX(2, 4) = 0
452
CONTINUE
454
CONTINUE
RETURN
END

**** CHECKPT ****

THIS SUBROUTINE CHECKS FOR BASIC FEASIBLE POINTS IN THE CONSTRAINT PLANES

SUBROUTINE CHECKPT (PLT, EXPT, LCOUNT, AS, SIGN, RS, FSPT, M)
DIMENSION PLT(7), EXPT(100, 3), RS(7), FSPT(100, 3)
DIMENSION AS(7, 7)
CHARACTER SIGN(7)
S = 1
DO 430 K = 1, 2
KK = PLT(K)
KK2 = KK + 2
DO 620 I = KK, KK2
DO 610 MM = 1, M
430
420
620
610
P 4PT = AS(MM, 1) * EXPT(I, 1) + AS(MM, 2) * EXPT(I, 2) + AS(MM, 3) * EXPT(I, 3) / RS(MM)
IF (SIGN(MM), EQ, '<', AND, PTPT, LE, 0) GOTO 609
IF (SIGN(MM), EQ, '<=', AND, ABS(PTPT), LE, 0.00001) GOTO 609
IF (SIGN(MM), EQ, '<', AND, PTPT, GE, 0) GOTO 609
GOTO 619
609 S = S + 1
610 CONTINUE
IF (EXPT(I, 1), LT, 0, OR, EXPT(I, 2), LT, 0, OR, EXPT(I, 3), LT, 0)
XGOTO 619
FSPT(LCOUNT, 1) = EXPT(I, 1)
FSPT(LCOUNT, 2) = EXPT(I, 2)
FSPT(LCOUNT, 3) = EXPT(I, 3)
LCOUNT = LCOUNT + 1
619 S = 1
620 CONTINUE
630 CONTINUE
RETURN
END

******** CHCK2 ********
**
** THIS SUBROUTINE CHECKS FOR BASIC FEASIBLE POINTS
** IN THE COLLECTION OF INTERSECTING POINTS
**
** SUBROUTINE CHCK2 (AS, EXPT, FSPT, RS, SIGN, JK, KOUNT, LCOUNT, M)
** DIMENSION AS(7, 7), EXPT(100, 3), FSPT(100, 3), RS(7)
** CHARACTER SIGN(7)
** S = 1
** KNT = KOUNT - 1
** DO 650 I = JK, KNT
** DO 640 MM = 1, M
** PTPT = AS(MM, 1) * EXPT(I, 1) + AS(MM, 2) * EXPT(I, 2) + AS(MM, 3) * EXPT(I, 3) / RS(MM)
** IF (SIGN(MM), EQ, '<', AND, PTPT, LE, 0.0001) GOTO 639
** IF (SIGN(MM), EQ, '<=', AND, ABS(PTPT), LE, 0.0001) GOTO 639
** IF (SIGN(MM), EQ, '<', AND, PTPT, GE, -0.0001) GOTO 639
** GOTO 649
** 639 S = S + 1
** 640 CONTINUE
** IF (EXPT(I, 1), LT, 0, OR, EXPT(I, 2), LT, 0, OR, EXPT(I, 3), LT, 0)
** XGOTO 649
** FSPT(LCOUNT, 1) = EXPT(I, 1)
** FSPT(LCOUNT, 2) = EXPT(I, 2)
** FSPT(LCOUNT, 3) = EXPT(I, 3)
** LCOUNT = LCOUNT + 1
** 649 S = 1
** 650 CONTINUE
** RETURN
**

******** ORDPT ********
** DROP IDENTICAL POINTS
** AND GRAPH THE SOLUTION SOLIDS
**
** SUBROUTINE ORDPT (LCOUNT, FSPT, ILEV, IGG, ISPEC, IRC, IATLK, ARS, XFACT, M)
** DIMENSION A(7, 7), RS(7), FSPT(100, 3), FSPT(150, 3)
** DIMENSION MTTX(30, 30), TRACK(100), LINE(6)
** INTEGER*2 IGG(2), IATLK(4), ILEV, IRC, ISPEC(5)
** INTEGER*4 LSHAPE(101), SHAPE
** TORG = 0
** DRVL = 0
** DRWN = 0
** ISPEC(5) = 0
DO 200 I=1,50
DO 199 J=1,3
FSPT(I,J)=0
199 CONTINUE
200 CONTINUE
DO 100 I=1,10
DO 99 J=1,10
MTX(I,J)=0
99 CONTINUE
100 CONTINUE
DO 102 I=1,101
LSHAPE(I)=0
102 CONTINUE
DO 104 I=1,100
TRACK(I)=0
104 CONTINUE
FSPT(I,1)=FSPT(I,1)
FSPT(I,2)=FSPT(I,2)
FSPT(I,3)=FSPT(I,3)
KNT=1
LCOUNT=LCOUNT-1
IF (LCOUNT.LE.1) RETURN
DO 305 J=2,LCOUNT
DO 300 I=1,KNT
IF (FSPT(J,1).NE.FSPT(I,1)) GOTO 300
IF (FSPT(J,2).NE.FSPT(I,2)) GOTO 300
IF (FSPT(J,3).NE.FSPT(I,3)) GOTO 300
GOTO 305
300 CONTINUE
KNT=KNT+1
FSPT(KNT,1)=FSPT(J,1)
FSPT(KNT,2)=FSPT(J,2)
FSPT(KNT,3)=FSPT(J,3)
305 CONTINUE
KNT1=KNT-1
DO 350 KA=1,KNT1
BOUND1=0
Z=0
IF (FSPT1(KA,1).EQ.0) Z=Z+1
IF (FSPT1(KA,2).EQ.0) Z=Z+1
IF (FSPT1(KA,3).EQ.0) Z=Z+1
IF (Z.GE.2) BOUND1=2
KJ=KA+1
DO 345 KB=KJ,KNT
BOUND2=0
Z=0
IF (FSPT1(KB,1).EQ.0) Z=Z+1
IF (FSPT1(KB,2).EQ.0) Z=Z+1
IF (FSPT1(KB,3).EQ.0) Z=Z+1
IF (Z.GE.2) BOUND2=2
IF (BOUND1.NE.2.AND.BOUND2.NE.2) GOTO 320
Y=0
IF (FSPT1(KA,1).EQ.0.AND.FSPT1(KB,1).EQ.0) Y=Y+1
IF (FSPT1(KA,2).EQ.0.AND.FSPT1(KB,2).EQ.0) Y=Y+1
IF (FSPT1(KA,3).EQ.0.AND.FSPT1(KB,3).EQ.0) Y=Y+1
IF (Y.EQ.0) GOTO 320
IF (Y.LE.1) GOTO 318
C*
BRWL=BRWL+1
MTX(KA,KB)=1
MTX(KB,KA)=1
*C
GOTO 345
318 DO 319 I=1,M
COMP1=A(I,1)*FSPT1(KA,1)*A(I,2)*FSPT1(KA,2)*A(I,3)*
XFSPT1(KA,3)-RS(I)
319
IF (ABS(COMP1), GT, 0.0001) GOTO 319
COMP2 = A(I, 1) * FSPT1(KB, 1) + A(I, 2) * FSPT1(KB, 2) + A(I, 3) * 
XSPT1(KB, 3) - RS(1)
IF (ABS(COMP2), GT, 0.0001) GOTO 319

**
DRWL = DRWL + 1
MIX(KA, KB) = 1
MTX(KB, KA) = 1
**
GOTO 319
CONTINUE
DO 321 I = 1, M
COMP2 = A(I, 1) * FSPT1(KB, 1) + A(I, 2) * FSPT1(KB, 2) + A(I, 3) * 
XSPT1(KB, 3) - RS(1)
IF (ABS(COMP2), GT, 0.0001) GOTO 321
COMP1 = A(I, 1) * FSPT1(KA, 1) + A(I, 2) * FSPT1(KA, 2) + A(I, 3) * 
XSPT1(KA, 3) - RS(1)
IF (ABS(COMP1), GT, 0.0001) GOTO 321

**
DRWL = DRWL + 1
MIX(KA, KB) = 1
MTX(KB, KA) = 1
**
GOTO 319
CONTINUE
320 TOT = 0
IF (FSPT1(KA, 1), EQ, 0, AND, FSPT1(KB, 1), EQ, 0) TOT = TOT + 1
IF (FSPT1(KA, 2), EQ, 0, AND, FSPT1(KB, 2), EQ, 0) TOT = TOT + 1
IF (FSPT1(KA, 3), EQ, 0, AND, FSPT1(KB, 3), EQ, 0) TOT = TOT + 1
DO 325 I = 1, M
COMP1 = A(I, 1) * FSPT1(KA, 1) + A(I, 2) * FSPT1(KA, 2) + A(I, 3) * 
XSPT1(KA, 3) - RS(1)
IF (ABS(COMP1), GT, 0.0001) GOTO 325
COMP2 = A(I, 1) * FSPT1(KB, 1) + A(I, 2) * FSPT1(KB, 2) + A(I, 3) * 
XSPT1(KB, 3) - RS(1)
IF (ABS(COMP2), GT, 0.0001) GOTO 325
TOT = TOT + 1
IF (TOT, LT, 2) GOTO 325

C*
DRWL = DRWL + 1
MIX(KA, KB) = 1
MTX(KB, KA) = 1
C
GOTO 345
CONTINUE
325 TOT = Y
DO 330 I = 1, M
COMP2 = A(I, 1) * FSPT1(KB, 1) + A(I, 2) * FSPT1(KB, 2) + A(I, 3) * 
XSPT1(KB, 3) - RS(1)
IF (ABS(COMP2), GT, 0.0001) GOTO 330
COMP1 = A(I, 1) * FSPT1(KA, 1) + A(I, 2) * FSPT1(KA, 2) + A(I, 3) * 
XSPT1(KA, 3) - RS(1)
IF (ABS(COMP1), GT, 0.0001) GOTO 330
TOT = TOT + 1
IF (TOT, LT, 2) GOTO 330

C*
DRWL = DRWL + 1
MIX(KA, KB) = 1
MTX(KB, KA) = 1
C
GOTO 345
CONTINUE
LKA=0
LKB=0
DO 1325 LNM=1,3
IF (FSPT1(KA,LMN),EQ,0) LKA=LKA+1
IF (FSPT1(KB,LMN),EQ,0) LKB=LKB+1
1325 CONTINUE
IF (LKA.EQ.1.AND.LKB.EQ.1) GOTO 1330
GOTO 345
1330 DO 3330 I=1,M
COMPl=A(I,1)*FSPT1(KA,1)+A(I,2)*FSPT1(KA,2)+A(I,3)*
FSPT1(KA,3)-RS(I)
IF (ABS(COMPl).GT.0,0001) GOTO 3330
COMPl=A(I,1)*FSPT1(KB,1)+A(I,2)*FSPT1(KB,2)+A(I,3)*
FSPT1(KB,3)-RS(I)
IF (ABS(COMPl).GT.0,0001) GOTO 2330
* * *
DRWL=DRWL+1
MTX(KA,KB)=1
MTX(KB,KA)=1
* * *
GOTO 345
2330 CONTINUE
DO 3210 I=1,M
COMPl=A(I,1)*FSPT1(KB,1)+A(I,2)*FSPT1(KB,2)+A(I,3)*
FSPT1(KB,3)-RS(I)
IF (ABS(COMPl).GT.0,0001) GOTO 3210
COMPl=A(I,1)*FSPT1(KA,1)+A(I,2)*FSPT1(KA,2)+A(I,3)*
FSPT1(KA,3)-RS(I)
IF (ABS(COMPl).GT.0,0001) GOTO 3210
* * *
DRWL=DRWL+1
MTX(KA,KB)=1
MTX(KB,KA)=1
* * *
GOTO 345
3210 CONTINUE
345 CONTINUE
350 CONTINUE
**
INC=2
I=1
LL=4
LSHAPE(1)=FSPT1(I,1)*FACT
LSHAPE(2)=FSPT1(I,2)*FACT
LSHAPE(3)=FSPT1(I,3)*FACT
KK=1
TRACK(KK)=1
362 DO 360 J=1,KNT
IF (MTX(J,J).EQ.1) GOTO 376
360 CONTINUE
IF (DRWL.EQ.DRWN) GOTO 380
* * *
BACKTRACK
370 KK=KK-1
I=TRACK(KK)
LSHAPE(LL)=FSPT1(I,1)*FACT
LSHAPE(LL+1)=FSPT1(I,2)*FACT
LSHAPE(LL+2)=FSPT1(I,3)*FACT
LL=LL+3
DO 372 J=1,KNT
IF (MTX(I,J),EQ,PT) GOTO 376
372 CONTINUE
GOTO 370
376 KK=KK+1
MTX(I,J)=INC
MTX(J,I)=INC
DRVN=DRVN+1
LSHAPE(LL)=FSPT1(J,1)*FACT
LSHAPE(LL+1)=FSPT1(J,2)*FACT
LSHAPE(LL+2)=FSPT1(J,3)*FACT
LL=LL+1
I=J
TRACK(KK)=J
IF(INC,EQ,1) GOTO 380
GOTO 362
380 (NC=1
PT=2
IF(I,IEQ,1) GOTO 386
DO 382 J=1,KNT
IF(MTX(I,J),EQ,2) GOTO 376
382 CONTINUE
GOTO 370
386 LL=LL-1
SHAPE=LL/3
IF(SHAPE,GE,33) GOTO 440
IF(SHAPE,LT,2) GOTO 390
* IF(SHAPE,LT,2) GOTO 400
388 CALL SHOFFPI (IGG,ILEV,ISPEC,LSHAPE,SHAPE,IRC,IATLK)
* GOTO 420
390 DO 392 I=1,3
LINE(I)=LSHAPE(I)
LINE(I+3)=LSHAPE(I+3)
392 CONTINUE
CALL LNUPF1 (IGG,ILEV,ISPEC,LX,EIRC,IATLK)
IF(IRC,NE,0) RETURN
400 IF(SHAPE,NE,1) RETURN
ISPEC(4)=10
ISPEC(5)=0
DO 394 J=1,3
DO 393 I=1,3
LINE(I)=LSHAPE(I)
LINE(I+3)=LSHAPE(I)
393 CONTINUE
LINE(J)=LINE(J)-50
LINE(I+3)=LINE(I+3)-50
CALL LNUPF1 (IGG,ILEV,ISPEC,LX,EIRC,IATLK)
IF(IRC,NE,0) RETURN
394 CONTINUE
ISPEC(4)=0
GOTO 420
440 IF(LSHAPE(97),NE,FSPT1(1,1)*FACT) GOTO 445
IF(LSHAPE(98),NE,FSPT1(1,2)*FACT) GOTO 445
IF(LSHAPE(99),NE,FSPT1(1,3)*FACT) GOTO 445
GOTO 388
445 WRITE (6,446)
446 FORMAT (/'ERROR, THERE ARE TOO MANY DATA POINTS'/'
  2x,'TO CREATE ONE OF YOUR SOLUTION SOLIDS CORRECTLY'/'
  2X,'PLEASE CHECK YOUR DESIGN FILE')
LSHAPE(97)=FSPT1(1,1)*FACT
LSHAPE(98)=FSPT1(1,2)*FACT
LSHAPE(99)=FSPT1(1,3)*FACT
GOTO 388
420 CONTINUE
RETURN
END
SUBROUTINE ENTP

SET VARIABLES TO ZERO

ENTER -1 FOR MIN 1 FOR MAX

ENTER NUMBER OF CONSTRAINTS

ENTER OBJECTIVE FUNCTION

ENTER THE CONSTRAINTS

 VIEW PROBLEM

 yes

 no

 no

 CALL FXPT

 yes
DISPLAY CONSTRAINTS

CHOOSE CONSTRAINT TO BE EDITED

ENTER NEW CONSTRAINT

NULL LINE ENTERED

EDITED CONSTRAINT COMPLETED

EDIT ANOTHER

A

yes

no

A
ENTER NEW CONSTRAINT

IF NULL

TOTAL CONSTRAINTS > 6

ADD 1 TO THE NUMBER OF CONSTRAINTS

ADD ANOTHER

PRINT "TOO MANY CONSTRAINTS"
JDISPLAY
CONSTRAINTS

ENTER ROW TO BE DELETED

NULL LINE

DISPLAY CHOSEN ROW

CORRECT ROW

ROW IS DELETED

DELETE ANOTHER

A

yes

no

yes

no
GET NEXT CHARACTER RIGHT OF DECIMAL

IS CHARACTER ≠

no

yes

no

yes

FOUND NONDIGIT

RETURN ERROR

FIND TOTAL STRING LENGTH WITHOUT DECIMAL

CONVERT TO DECIMAL EQUIVALENT

MULTIPLY BY -1 IF NEGATIVE

RETURN
SET NEGATIVE FLAG TO 1

GET NEXT CHARACTER

IF

IF \( p \)

IF \( <0 \text{ OR } >9 \)

RETURN ERROR

RETURN

CONVERT STRING TO INTEGER

MULTIPLY BY -1 IF NEGATIVE

END OF STRING

CONVERT STRING TO INTEGER

INTEGER PORTION

no

yes

yes

no

no

yes

no

yes
SUBROUTINE LPROG

INITIALIZE VARIABLES

COUNT THE NUMBER OF \(<, =, >\)

SORT THE CONSTRAINTS BY LOGICAL OPERATOR

ADD SURPLUS AND ARTIFICIAL VARIABLES TO CONSTRAINTS

MAKE OBJECTIVE FUNCTION A MIN FUNCTION

PRINT INITIAL TABLEAU

SEE EACH ITERATION

yes SET FLAG LFLAG=1

no
IF I

J

T

A

I

T

I

1

FIND SMALLEST Cj

IF ALL Cj > 0

yes

no

FIND SMALLEST BASIS

IF COLUMN < 0

yes

no

PIVOT

PRINT NO FEASIBLE SOLUTION

M

RETURN MAIN

UNBOUNDED

UNBOUNDED

RETURN MAIN

PRINT ITERATION

yes

no

ART. VAR. IN BASIS

RETURN MAIN

L

MAIN

IF LFLAG = 1
PRINT FINAL TABLEAU

CHECK Cj-Zj FOR ZEROS

IF Cj-Zj=0

yes
PRINT ALTERNATIVE OPTIMIZATION EXISTS

no
PRINT FINAL SOLUTION

SEE DUAL

yes
PRINT DUAL

no

SENSITIVITY ANALYSIS

yes CALL SENS

no

RETURN MAIN
SUBROUTINE SENS

INITIALIZE VARIABLES

RETURN OBJ. FUNC. TO ITS ORIGINAL MAX OR MIN STATE

DO I=1,3

BASIC VARIABLE

no

FIND MAXIMUM INCREASE

yes

CHECK EACH COLUMN FOR MAX LIMIT

CHECK EACH COLUMN FOR MIN LIMIT

PRINT LIMITS

END DO

| N |
SET UP INVERSE OF CURRENT BASIS ARRAY

SET MIN PARAMETER HIGH

SET MAX PARAMETER LOW

DO N=1,3

COLUMN N OF B INVERSE

FIND NEW LOW RHS

FIND NEW HIGH RHS

PRINT RHS LIMITS

END DO

RETURN
SUBROUTINE DPLT

INITIALIZE VARIABLES

ENTER DESIGN FILE NAME

SET UP DESIGN FILE

ENTER STARTING LEVEL OF DESIGN

INITIALIZE ARRAYS

DO M=1, TOT CONSTR.

CALCULATE CONSTRAINT PARAMETERS

CALCULATE MAX PARAMETER

Determine # parameters per constr. = 0

END DO

Factor constraint parameters

P
DO M = 1, TOT CONSTR.

IF < yes CALL LESS
no

IF > yes CALL GREAT
no

NO CONSTRAINT COEF = 0 yes CALL DRAW SHAPE
no

1 COEFF = 0 yes CALL LINES
no

2 COEFF = 0 yes CALL POINTS
no

CALL EXT

CALL EX

END DO

Q
IF SIMPLEX SOLUTION

yes

CALL FNDPT

CALL CHKPT

CALL CHK2

...,

CALL CHKPT

FOR ALL CONSTRAINTS

FOR CONSTRAINTS 1-5

S
DO

TAKE THREE CONSTRAINTS

SET UP BASES FOR ROW MANIPULATION

PERFORM ROW MANIPULATION

PLACE FINAL b VECTOR COEFF IN EXPT ARRAY

CALL CHK2

END DO

PLACE 0's IN EXPT ARRAY

CALL CHK2
SUBROUTINE LESS

DRAW LINE FROM POINT TO ORIGIN

RETURN

SUBROUTINE GREAT

DRAW LINES FROM CONSTR. POINT TO END OF AXIS

RETURN
SUBROUTINE LINES

FIND CONSTR. ≠ Ø

DRAW LINES BETWEEN NONZERO COEFF.

PROJECT LINES IN DIRECTION OF Ø COEFF

RETURN

SUBROUTINE POINTS

PLACE EXT. LINES FROM Ø COEFF

RETURN
SUBROUTINE EXT

0
\text{COEFF. < 0 OR RHS=0}

\text{RETURN}

2
\text{COEFF. < 0}

\text{RETURN}

\text{DRAW}
-\text{P1 THRU P2}
-\text{P1 THRU P3}

\text{DRAW}
-\text{P1 THRU P2}
-\text{P3 THRU P2}

\text{RETURN}
SUBROUTINE FNDPT

DO CONSTR. COMBINATION

DO COEFF. COMBINATION

SET UP 2X2 BASIS

CALCULATE SOLUTION BY ROW MANIPULATION

PLACE POINT IN EXPT ARRAY

END DO

END DO

RETURN
SUBROUTINE CHKPT

DO
  TOTAL EXPT ARRAY

  POINTS VALID FOR CONSTR.

  POINTS ≠ 0

    yes

    FEASIBLE POINT

    ADD TO FEAS. PT ARRAY

    END DO

RETURN
SUBROUTINE CHK2

DO INTERSECTING POINTS

POINTS VALID FOR CONSTR.  yes  no

POINTS ≠ ∅  yes  no

FEASIBLE POINT

ADD TO FEAS. PT. ARRAY

END DO

RETURN
SUBROUTINE ORDPT

INITIALIZE VARIABLES

REMOVE DUPLICATE POINTS FROM FEAS. PT ARRAY

SET 1ST PT IN FEAS. PT ARRAY AS PT 1

DO PT1 = PT 1, TOTAL PTS - 1

DO PT2 = PT 2, TOTAL PT PTS

DETERMINE IF PT1 IS ON AN AXIS

DETERMINE IF PT2 IS ON AN AXIS

NEITHER PT ON AXIS

ONLY 1 PT ON AXIS

ON DIFFERENT AXIS

ON SAME AXIS ADD PTS TO DRAW LINE MATRIX

END DO

END DO

END

CC

DD

AA

LL
TEST FOR TWO SHARED CONSTRAINTS

2 SHARED CONSTRAINT

yes

ADD TO DRAW LINE MATRIX

no

TEST FOR 1 AXIAL POINT & 1 BOUNDARY PT & SHARE 1 CONSTRAINT

PASS TEST

yes

ADD TO DRAW LINE MATRIX

AA

no

AA
INITIALIZE VARIABLES

SET 1ST PT IN CONNECTION MATRIX

FIND CONNECTION

PLACE IN FINAL SHAPE MATRIX

DO DONE

ALL LINES DRAWN

GO BACK TO PREVIOUS POINT

CONNECTION

STILL DRAWING SHAPE

LL

MM

NN
DRAW SHAPE
OR POINT

RETURN

BACK
AT FIRST
POINT

yes

MORE
THAN 33
POINTS

DRAW SHAPE
OR POINT

RETURN

SET POINT
33 TO
POINT 1

DRAW
SHAPE

RETURN

PRINT
TOO MANY
POINTS

no

yes

no

yes

NN

MM
APPENDIX E

DFPI COMMANDS
The DFPI (Design File Processor Input) is Intergraph's software package for allowing users to write their own applications programs in Fortran which can access, create, or modify a design file. What this means is that the user may interactively manipulate design files while maintaining control of the application programs to perform other processes. All graphics commands are available to the user's Fortran program through the DFPI. Information presented in this chapter may be found in Intergraph's IDGS Application Software Interface Document Version 8.6 or 8.8 if the reader desires to know more about the DFPI processing and available commands.

Opening and Closing Design Files

The DFPI sets up each of the graphics commands as a separate subroutine which may be accessed by call statements in the user's program. The parameters, or arguments, of the DFPI call statements are predefined. The user must make certain that these arguments are defined before invoking the DFPI package. To better clarify this process, the DFPI statements used for the graphics program in this thesis shall be explained next.

To open the design file the user must enter the design file specifications from the keyboard. The first statement in opening a design file is the LKTCSI call statement. Its form is as follows:
CALL LKTCSI (INBUF,IFILE,ICHA,0,IRC)

The LKTCSI routine generates the file specification block. It is needed to convert the file name from ASCII code to RAD50 code (ASCII code has 1 character per 16 bit word and RAD50 code has 3 characters per 16 bit word). The LKTCSI command has five arguments which were called INBUF, IFILE, ICHA, 0, and IRC.

INBUF is the file specifier and is entered via the keyboard in a previous read statement of the DPLT subprogram. It must be entered in the following format.

\[
\text{DEVC:}[\text{UFD}]\text{name.EXT;VER}
\]

where

DEVC = disk drive device

UFD = user file directory

name = the name of the design file

EXT = the extension or type of file such as DGN for a design file

VER = version, defaults to current version

The user running LPVAX would have to type in "QSA2: [51,45]file.DGN". The system will default to the current drive, directory, and version. The user will need only to enter the file name and the extension.

INFILE is a block of 7 bytes which will contain the file specification block. This block is generated by the computer.

The ICHA argument contains the length, in bytes, of the argument INBUF. It is determined through the read
statement. By using the Q parameter in the format statement the number of keystrokes would literally be counted. The read sequence is as follows:

```
READ (ICHA,INBUF)
FORMAT (Q,15A2)
```

The fourth argument in the LKTCSI call statement is given the value of zero. This specifies which side of the command line is to be used in generation of the file specification block. Zero is used for outfile and a one is used for infile.

The IRC argument is used for the error return code. This contains the completion code on the return from the LKTCSI routine. If IRC equals zero no errors were encountered. Otherwise, the error code will be passed back to the calling routine for the programmer's convenience. The error code should always be checked when using any DFPI call statements and the program terminates if any statement produces a non-zero error code.

The next DFPI routine need in opening a design file is the ASC2RD call statement. This statement has four arguments associated with it and is written as:

```
CALL (INBUF,REGN50,ICHA,IRC)
```

The purpose of this statement is to convert a six-character region name written in ASCII to RAD50 code. The user is asked to supply the six characters. The explanation of these arguments follows:
INBUF - is the buffer containing the character string supplied by the user.
REGN50 - is the buffer which contains the converted RAD50 character string.

ICH - contains the number of characters in the character string.
IRC - the error return code for this routine.
The DFPI link requires a region to be set up for processing the DFPI commands. The region name allows the DFPI to address this region.
The next command used in opening a design file is the INDFPI statement. This statement has seven arguments associated with it. It is written in the DPLT subprogram as:

```
CALL INDFPI (REGN50, IFILE, 0,0,0,1,IRC,IEX)
```
The explanation of these arguments follows:

REGN50 - This argument is the six-character region as previously defined.
IFILE - This argument is the file specification block generated by the LKCSI routine.
Argument 3 - This argument will either be a 0, a 1, or a 2. Zero is entered if the user will be accessing only a design file. A one is entered if the user will be accessing both a design file and a cell file, and a two is entered if the user will be accessing only a cell file.
Argument 4 - This is the cell library name. This must be a seven word LKTRAN file specification block. If no cells are to be used then this argument is set to one to specify no cell libraries are to be called.

Argument 5 - A 0 or a 1 is used for this argument. A zero signifies that no swap to the IDGS (Interactive Graphics Design System) format is needed. A 1 signifies that a swap to IDGS is required. Fortran is already in the IDGS format so a zero must be entered here.

Argument 6 - This argument sets the region type. A 0 in this argument produces a noncircular request region while a 1 in this argument produces a circular request region. In the Vax, the request for DFPI services are made in the circular request area. The general size of this area is 256 word pages with 16 pages total. Other sizes are possible. This is the region where communication between the Fortran program and the DFPI routines must pass.

IRC - This is the error return code argument.

IEX - This argument is generally given the value 'EX'. This is used for the terminal number. However, 'EX' may be used in place of the terminal number unless the DFPI is being used with a user command.

A user command is a disk file containing ASCII text records or statements in the following format:
The statement must have the format:

LABEL: Operator, Operand, Operand ....; Comment.

The list of command operators may be found in the Interface Document Version 8.6 or 8.8.

The function of the INDFPI command is to initialize and/or set up the design file and cell library if a cell library is used. This command:

1) Creates, initializes and attaches the user defined region, creates the file builder subprocess and the mailboxes based on the terminal number.
2) Retrieves the design file as specified if argument 3 is zero or one.
3) Retrieves the cell file as specified if argument 3 is zero or one.
4) Determines whether the input will be swapped.
5) Sets region type to be circular or noncircular.
6) Sets the dimension to 2D or 3D.
7) Sets the terminal ID.

The design file or cell library need only be defined once. However, if more than one design file is called by the user's program then one INDFPI statement is used per design file.
If a person wants to open more than one design file in a program, then the GGDFPI command is needed. It is written as:

```
CALL GGDFPI (ARG1, ARG2)
```

**ARG1** - this is a two word array of INTEGER*2 type. The first element will have the value of 3 to update the old design file. The second element will have a zero, to use the next available graphics group number, or $N$, where $N$ is the next available graphics group if known.

**ARG2** - This is the error return code.

This will allow the DFPI to release the old design file and retrieve the new file when the INDFPI is used. The LPVAX program only accesses one design file so this last call statement was not used.

When the program is complete the design file must be closed. The first step in closing the design file is to check the error code. If the design file is being closed because an error in one of the DFPI call statements was encountered then two error routines may be called. These routines are called ERROR9 and ERROR.

The call ERROR9 routine is used when the error code is between 924 and 927. An error code in this range signifies a system error. This routine displays the system error. The ERROR9 statement is written as:

```
CALL ERROR9 (IRC)
```

The other error routine is in the format
CALL ERROR
This statement has no arguments associated with it. It will allow the message describing the error to be displayed. It will also ask you if you wish to see the dump of the request region. This dump is only useful if the user is completely knowledgable about the DFPI and the request region.

There are two statements to close the DFPI routines. These are the DEDFPI and the FINDFPI. The DEDFPI detaches the DFPI for the users program. It is written as:

    CALL DEDFPI (ARG)

The argument is set to zero if the program is terminated under error conditions. This forces the DFPI to detach immediately. When detaching under normal conditions the argument is set to 1. This will allow the DFPI to finish any existing process before terminating.

The final closing routine is the FNODF. This closes the design file. One FNODF statement is needed for each design file opened. This statement is written as:

    CALL FNODF

Design File Manipulations
For the LPVAX program, two basic routines were used. One was the LNDFPI (draw line between two points) and the other was SHDFPI (draw shape). Remember the program is designed for three-dimensional graphics which are reflected in the
commands. Two-dimensional graphics will have slight variations in these commands.

The LNDFPI command has six arguments associated with it. In the program it will be written as:

```
CALL LNDFPI (IGG,ILEV,ISPEC,LINE,IRC,IATLK)
```

The meanings of these arguments are as follows.

**IGG** - This is a two word array defined as INTEGER*2. Both of the two elements were set to zero. However, the options are as follows:

- **WORD 1** is the graphics group code
  - 0 = no operations, continue current association
  - 1 = start graphic group
  - 2 = cancel graphic group

- **WORD 2** is the graphic group number
  - 0 = next available number
  - N = use N

**ILEV** - This is an integer from one to sixty three. This specifies the level in the design file the line is to be placed.

**ISPEC** - This is a five word array containing the line specifications. The parameters are as follows:

- **WORD 1** - Class Number 0-5
- **WORD 2** - Status 0=solid, 1=hole
- **WORD 3** - Style 0-4
- **WORD 4** - Line weight 0-31
LINE - This argument is a one dimensional array containing the X, Y, Z coordinates of the two end points of the line being drawn.

IRC - Error return code

IATLK - The attribute linkage. This is set to zero since there are no attributes assigned to the design file.

The shape SHDFPI routine has seven arguments. When drawing shapes one must keep in mind that the end point must be equal to the starting point in order to close the shape. The advantage of drawing the solution solid as a shape, instead of just connecting lines with the LNDFPI routine, is that a shape has the option of hidden line removal. The SHDFPI is written in the format as follows:

CALL SHDFPI (IGG, ILEV, ISPEC, ISHAPE, SHAPE, IRC, IATLK)

the arguments IGG, ILEV, ISPEC, IRC, and IATLK are identical to the arguments used for the LNDFPI statement. The arguments ISHAPE and SHAPE are described as follows:

ISHAPE - This argument (INTEGER*4) is a variable length one dimensional array. It holds the X, Y, Z coordinates of all points making up the shape. The points must be arranged such that the object can be drawn without "lifting the pen off the paper" so to speak. Doubling back on the same line is allowed to accomplish this. As
stated before, one must end up at the same point where one began.

SHAPE - This is an INTEGER*4 value which gives the number of coordinate sets used to create the shape described in the ISHAPE array.
LPVAX is a linear programming package which solves 3-dimensional linear programming problems and graphs the solutions into a design file. These instructions are in three sections: Intergraph VAX system, Running LPVAX, and Design File Operations. It is recommended that the first time Integraph user carefully read all three sections.

Section 1
Intergraph-VAX System

Logon - To logon to the system press the return key of the keyboard located at the workstation. The message "USERNAME:" should appear. Enter "HUMPHREYS" and press Return. The system should respond with "PASSWORD". Enter the password "LINDA".

WAIT UNTIL YOU GET THE GRAPHICS PROMPT "GRAPHICS:".

Logoff - To logoff type "LO" when you have any of the three environment prompts. These prompts are "DCL: ", "UTILITIES: ", and "GRAPHICS: ".

You will be working in three environments: Utilities, Graphics, and DCL (Data Control Language). You may move from one environment to the next by entering

UT [return] for utilities
GR [return] for graphics
DCL [return] for DCL

UTILITIES

Create a Design File

UTILITIES: This environment will allow you to create a design file. You can accomplish this by the following sequence. System commands are in bold face type and user responses are in normal type

UTILITIES: CR [return]
FILENAME: enter a name using alphanumeric characters with no more than 9 characters [return].
TYPE OF FILE (2D/3D/2C/3C): 3D
(you must enter 3D or program will crash)
BLOCKS REQUIRED: 300
(if there is not enough space to create your design file then try a lower number for the required block space)

Shortcut Method

UTILITIES: CR name 3D 200 [return]

Delete a File

To delete your design file when you are finished go to utilities or DCL and do the following.

UTILITIES: DELETE name.DGN

You must enter the ".DGN" to your file name so the system knows it will be deleting a design file.

The system will ask you if you are sure. Type Y and your file will be deleted.

DO NOT DELETE ANY OTHER FILE !!!!!!!

DCL

DCL: You are now in the VMS VAX system. You must be in the DCL environment to execute the program. To execute type RUN LPVAX.

DCL: RUN LPVAX

SWAPPING — Since there are two screens per workstation, you will have to "SWAP" screens. Whenever one screen is full or you wish to use a fresh screen you may press the SWAP key located on the rightmost keypad on the keyboard. This is particularly helpful when trying to keep menus and equations from being split between screens.

GRAPHICS

GRAPHICS: This environment allows access to your design file. Just type your file name after the graphics prompt and return.

GRAPHICS: file name

The screens will temporarily go blank while the file is being loaded.

To exit from the design file press the control key at the same time you press the Z and then return.
CTRL Z [return]

See Section 3 for design file manipulations

Section Two
RUNNING LPVAX

LPVAX is executed after the RUN LPVAX is typed in the DCL environment. The program will ask you to press any key when ready to begin. This means any alphanumeric key. This will allow you to swap to a new screen if necessary.

Next the master menu will appear. Type in the number corresponding to the function you wish to perform.

1) ENTER EQUATIONS
2) EDIT EQUATIONS
3) SIMPLEX ALGORITHM
4) GRAPHICAL SOLUTION
5) EXIT

ENTER CONSTRAINTS — This routine will allow you to set up new problems. First it will ask you if the problem type is maximization or minimization. Enter 1 if MAX and -1 if MIN.

Next enter the number of constraint equations. Do not enter more than 6.

You will then be asked to enter the objective function. Enter only the coefficients of the objective function pressing return after each value.

Next enter the constraints. Enter the coefficients on the left hand side [return] after each coefficient, enter the logical operator [return], and then enter the right hand side value [return].

You will then be given the option of viewing the problem. Type Y if you wish to do so.

EDIT EQUATIONS — This routine will allow you to

1) Change MAX or MIN problem type
2) Change the objective function
3) Change the constraint equations
4) Add another constraint
5) Delete a constraint equation
6) Return to the main menu
This routine will constantly display the problem and the line you are operating on.

SIMPLEX ALGORITHM - This option performs the simplex algorithm on your problem. It will return to the screen the initial tableau (the problem will be re-ordered according to <, =, and then > equations), the tableau for each iteration (if you desire), and the final tableau. It will also display the optimal value and the optimal point. You will also have the option to display the solution to the dual problem and run a sensitivity analysis on the problem.

GRAPHICAL SOLUTION - This is to be used ONLY if the simplex algorithm has previously been executed and the problem is feasible. Otherwise, the resulting graphs will make little sense. The graphical subprogram will ask you for your design file in the form of

DEV:[UFD]NAME.DGN

DEV - the disk drive device - QSA2
UFD - the user file directory - 51,45
NAME - your file name
DGN - the file type - DGN (design file)

you need only enter

NAME.DGN

The program will then require the starting level for your graphs. You are given 4 levels to choose from (1, 15, 30, and 45). The design file has 63 different levels. These levels have been divided into 4 sections with the starting levels of each section being 1, 15, 30, and 45. This allows four different problems to be placed in the same design file. A directory found in the left hand screen in the design file will list where the planes will be found.

When the graphs have been placed, the program will return you to the main menu. You may wish to enter a new problem or make changes in the present problem and run again.

EXIT - To view your problem, you must exit the LPVAX program. This option return you to the DCL environment and the DCL prompt will appear.

Enter the graphics environment and open your design file to see the results. A bright star will appear
on the level containing your solution solid. This star marks the point of optimality.

Section Three
DESIGN FILE

DCL: GR
GRAPHICS: NAME

The design file contains the graphical results of your problem. When the design file is opened, the left screen will have the level directory describing which figure is on which level and in what color. Based on the starting level given in the program (1, 15, 30, 45). There is also a small isometric representation of your solution solid and the X, Y, Z coordinate axis. The right hand screen displays the large isometric coordinate axis.

LEVELS - There are 63 design file levels. These levels act as superimposed transparencies with part of a design on each transparency. Levels may be turned on and off at will and any number of levels may be turned on or off at one time.

Turning levels on and off - To turn on a level type ON= and the level number. To turn off a level type OF= and the level number. To turn on or off consecutive levels type the first level and the last level with a hyphen between. If you wish to turn on nonconsecutive levels type the level numbers you wish to turn on separated by commas

Example: If you want to turn on levels 1 through 4 and also level 8 you would type

ON=1-4,8

NOTE: When you open your design file all levels will be "ON". Turn off all the levels from 1 to 60 and then turn on the ones you wish to see.

SELECT VIEW: When you turn the levels on and off the system will ask you to select a view. Two views are displayed, view 1 on the right hand screen and view 6 on the left hand screen. To select these views move your cursor across the table top until the cross hairs appear in the upper right hand quadrant of the screen you wish to change. Once the cross hairs are in position push the data button to execute
UPDATE: When you turn a level off portions of the remaining figures may be removed. To return these figures to their natural shape you may have to update the screen. To update move the mouse to the box marked update. Place the mouse crosshairs on the "L" to update the left screen, "R" to update the right screen, or "BOTH" to update both screens. When the mouse is in position, push the command button (marked with a "C") to execute.

UPDATE

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
<th>BOTH</th>
<th>VIEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOP VIEW</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ROTATE: To rotate the large isometric figure place the cursor over the box marked "DYNAMIC VOLUME". The system will then ask for two points. Working with the right screen only, move the cursor until you see the crosshairs on the upper left hand corner of the right screen and push the data button. The system will ask for a second point. Bring the cursor to the lower right hand corner of the screen and push the data button again. If successful you will see the coordinate axis move to the center of the screen. Keeping the cross-hairs in the right screen, press the rotate button to rotate the graphs. Marked with an "R" in Figure 1.

DYNAMIC VOLUME

| DROP | PAN | VOLUME |

ZOOM IN - ZOOM OUT: If you wish to get a closer look at part of the graph you may ZOOM IN on it. The lower left hand corner of the cursor allows you to zoom in. To zoom back out push the button on the lower right hand side of the cursor.

MOVE - To move the objective constraint, turn on the level that contains the alternate constraint. Select the Element MOVE command from the tablet on the table top. Look at the corner of the objective function triangle that has the maximum value; this value has 4000 UOR's (units of reference). You would identify this plane by keying in that maximum coordinate. You will then press the reset button on the cursor until the plane lights up. Once highlighted, key in the new
location. The plane will automatically be repositioned. For example, if the maximum point of this plane lies on the Y axis and you want to move it half way towards the origin you would use the following sequence.

1) Key in \( XY=0,4000,0 \)
2) Press reset until the plane becomes highlighted.
3) Key in \( XY=0,2000,0 \)

**SCALE** - To scale the plane you have just moved you would again key in its coordinate to identify it. You must first enter the scale factor by typing AS= and the factor. For the above move command, you may want to scale the plane by one half since you moved it half way towards the origin. The following sequence will perform this.

1) Key in \( AS=.5 \)
2) Key in \( XY=0,2000,0 \)
3) Hit the reset button until the plane becomes highlighted.
4) Key in \( XY=0,2000,0 \)

**HIDDEN LINE REMOVAL** - If you wish to remove the hidden lines from your solution solid you must turn off the dynamic volume if it is on. Next, using the command button, select the HIDDEN LINE option from the menu tablet. The hidden line menu will appear on the right screen. Entering a data button in any of the boxes in this menu will activate that box command. The explanation of the hidden line menu options are as follows:

Screen Display - The word "CURRENT" should be written in this box. Do not change this.

Display - This is where the results of the hidden line removal will be sent. The box marked display should have the word "CURRENT" written in this box also. Again do not change this selection.

**VIEWING PARAMETERS**

Eye Point - This is where the light source is located relative to your design. To change this, enter a data button here and type in the new values with a return after each value.
Target Point - This is where the light source is pointed. This is changed the same way as the Eye Point parameter.

Perspective Viewing Angle - This gives three dimensional perspective when a value other than zero is entered. The further points from the screen will appear smaller and shorter. You may enter any value between 0 and 179.

Update Selected View - If you change the eye point parameters and/or the target point parameters, this command will reposition the design file accordingly.

OUTPUT TYPE

Smooth Shading - This command allows contour shading for curved surfaces. There are no curved surfaces in the design file so you will not benefit from this command. It will shade your object but at a slightly slower rate than constant shading.

Constant Shading - This option will shade the side of the plane facing the light source.

Edges - The outline of the object will be displayed with the hidden lines removed.

If the object being shaded disappears from view, then the light source is located on the opposite side of the design. Either update the select view or change the viewing parameters.

EXECUTE COMMANDS

Light source - Lets you put a data point on the screen to determine the placement of the light source. The shading will start immediately following the data point placement.

Execute - This will start the shading process based on the eye point and target point coordinates.

Tolerance - This is the amount of pixel error the system will allow. This may be changed to any number between .5 and 8. The higher the number the faster the shading.

Help - This command will give you an explanation of any of the hidden line options. Enter a data point in the help box and another data point in any of the other boxes for the explanation of
that box will appear on the screen.

Status - This box will ask for information or it will inform you of the status of the hidden line removal routine.

Exit - By placing a data point in this box you will exit the hidden line menu and return to the design file.

Note: If you run into trouble in the hidden line routine or anywhere in the design file type CNTL Z to exit the design file.

EXIT: To exit the design file push the control key on the keyboard and at the same time hit the Z button. Then press return. "CTRL Z [return]."

Figure 1. Cursor
GLOSSARY

CLIPPING PLANE - An imaginary plane which sets the depth of the image to be viewed.

CURSOR - A hand held device used for entering commands to the design file.

DESIGN FILE - A computer file where a graphical design is stored.

DYNAMIC ROTATE - A command which sets a graphical image in a continuous rotation.

DYNAMIC PAN - A command which sets a graphical image in a continuous rotation.

DYNAMIC ZOOM-IN - A command which continuously enlarges a graphical image.

DYNAMIC ZOOM-OUT - A command which continuously shrinks a graphical image.

ELEMENT - An image in a design which was constructed as a unit.

EYE POINT - A location outside the design file used to determine the angle at which the design is viewed.

FAR CLIPPING PLANE - A clipping plane which establishes the back depth of the image to be viewed.

FEASIBLE REGION - A set whose elements satisfy all the constraints in a given linear programming problem.

HIDDEN LINES - Lines which would be obstructed from view if the object were a true solid and not a line representation of that solid.

IDGS - Interactive Graphics Design System is the system software which controls the design file manipulations and the design file storage.

LIGHT SOURCE - A location which serves as a source for an imaginary light to be directed onto the image being viewed. It is used for removing hidden lines.

MENU TABLET - A sheet of paper located on the surface of the workstation. This tablet has pictorial representations of the various graphics commands.
NEAR CLIPPING PLANE - A clipping plane which establishes the starting depth of the image to be viewed.

REFRESHING THE SCREEN - This command is used to reset the screen when images are added or removed.

SI - Software Infusion is a term given to the use of software as teaching aids.

TARGET POINT - A point on an image towards which the light source is pointed.

WINDOW - A designated region within a design file which will be displayed.
APPENDIX H

SAMPLE PROBLEM
1. Enter the problem
2. Edit the problem
3. Simplex algorithm
4. Graphical solution
5. Exit

Enter your selection
1
Is this a minimization or a maximization?
Enter -1 for min and 1 for max
1
Enter the number of constraints
Do not enter more than 6
3
Enter your x, y, and z coefficients for the objective function (press return after each value)
Enter the coefficient for x
1
Enter the coefficient for y
2
Enter the coefficient for z
3

Next you will entering the coefficients to the constraints. Enter the x, y, z coefficients [RETURN] then enter the logical operator [RETURN] and then the right hand side value [RETURN]


Enter the coefficients to constraint 1
Enter the coefficient for x
1
Enter the coefficient for y
2
Enter the coefficient to z
3
Enter the logical operator

Enter the right hand side
ENTER THE COEFFICIENTS TO CONSTRAINT 2
ENTER THE COEFFICIENT FOR X
ENTER THE COEFFICIENT FOR Y
ENTER THE COEFFICIENT TO Z
ENTER THE LOGICAL OPERATOR
ENTER THE RIGHT HAND SIDE

ENTER THE COEFFICIENTS TO CONSTRAINT 3
ENTER THE COEFFICIENT FOR X
ENTER THE COEFFICIENT FOR Y
ENTER THE COEFFICIENT TO Z
ENTER THE LOGICAL OPERATOR
ENTER THE RIGHT HAND SIDE

DO YOU WISH TO VIEW YOUR PROBLEM, Y OR N

MAX

1.00X + 2.00Y + 3.00Z

1.00X + 2.00Y + 3.00Z >= 3.00
2.00X + 1.00Y + 4.00Z >= 10.00
3.00X + 2.00Y + 1.00Z >= 5.00

DO YOU WISH TO MAKE ANY CHANGES Y OR N

*************** LPVAX ***************

1. ENTER THE PROBLEM
2. EDIT THE PROBLEM
3. SIMPLEX ALGORITHM
4. GRAPHICAL SOLUTION
5. EXIT

ENTER YOUR SELECTION
### Initial Tableau

<table>
<thead>
<tr>
<th></th>
<th>1.00</th>
<th>2.00</th>
<th>3.00</th>
<th>-1.00</th>
<th>0.00</th>
<th>0.00</th>
<th>1.00</th>
<th>0.00</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.00</td>
<td>2.00</td>
<td>1.00</td>
<td>4.00</td>
<td>0.00</td>
<td>-1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>10.00</td>
<td>-1.00</td>
<td>-2.00</td>
<td>-3.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>5.00</td>
<td>-8.00</td>
<td>-3.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Do you wish to see the tableau for each iteration, y or n?**

### After Iteration 1

<table>
<thead>
<tr>
<th></th>
<th>0.33</th>
<th>0.67</th>
<th>1.00</th>
<th>-0.33</th>
<th>0.00</th>
<th>0.00</th>
<th>0.33</th>
<th>0.00</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>0.67</td>
<td>-1.67</td>
<td>0.00</td>
<td>1.33</td>
<td>-1.00</td>
<td>0.00</td>
<td>-1.33</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>6.00</td>
<td>2.07</td>
<td>1.33</td>
<td>0.00</td>
<td>0.33</td>
<td>0.00</td>
<td>-1.00</td>
<td>-0.33</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>4.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3.00</td>
<td>3.32</td>
<td>0.33</td>
<td>0.00</td>
<td>-1.67</td>
<td>1.00</td>
<td>1.00</td>
<td>2.67</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### After Iteration 2

<table>
<thead>
<tr>
<th></th>
<th>0.00</th>
<th>0.50</th>
<th>1.00</th>
<th>-0.38</th>
<th>0.00</th>
<th>0.13</th>
<th>0.38</th>
<th>0.00</th>
<th>-0.13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>0.00</td>
<td>-2.00</td>
<td>0.00</td>
<td>1.25</td>
<td>-1.00</td>
<td>0.25</td>
<td>-1.25</td>
<td>1.00</td>
<td>-0.25</td>
</tr>
<tr>
<td>5.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.00</td>
<td>0.13</td>
<td>0.00</td>
<td>-0.38</td>
<td>-0.13</td>
<td>0.00</td>
<td>0.38</td>
</tr>
<tr>
<td>1.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3.00</td>
<td>0.00</td>
<td>2.00</td>
<td>0.00</td>
<td>-1.25</td>
<td>1.00</td>
<td>-0.25</td>
<td>2.25</td>
<td>0.00</td>
<td>1.25</td>
</tr>
</tbody>
</table>

### After Iteration 3

<table>
<thead>
<tr>
<th></th>
<th>0.00</th>
<th>-0.10</th>
<th>1.00</th>
<th>0.00</th>
<th>-0.30</th>
<th>0.20</th>
<th>0.00</th>
<th>0.30</th>
<th>-0.20</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.00</td>
<td>0.00</td>
<td>-1.60</td>
<td>0.00</td>
<td>1.00</td>
<td>-0.80</td>
<td>0.20</td>
<td>-1.00</td>
<td>0.80</td>
<td>-0.20</td>
</tr>
<tr>
<td>4.00</td>
<td>1.00</td>
<td>0.70</td>
<td>0.00</td>
<td>0.00</td>
<td>0.10</td>
<td>-0.40</td>
<td>0.00</td>
<td>-0.10</td>
<td>0.40</td>
</tr>
<tr>
<td>1.00</td>
<td>0.00</td>
<td>-1.60</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.80</td>
<td>0.20</td>
<td>0.00</td>
<td>0.80</td>
<td>-0.20</td>
</tr>
<tr>
<td>7.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### After Iteration 4

<table>
<thead>
<tr>
<th></th>
<th>0.00</th>
<th>-0.10</th>
<th>1.00</th>
<th>0.00</th>
<th>-0.30</th>
<th>0.20</th>
<th>0.00</th>
<th>0.30</th>
<th>-0.20</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.00</td>
<td>0.00</td>
<td>-1.60</td>
<td>0.00</td>
<td>1.00</td>
<td>-0.80</td>
<td>0.20</td>
<td>-1.00</td>
<td>0.80</td>
<td>-0.20</td>
</tr>
<tr>
<td>4.00</td>
<td>1.00</td>
<td>0.70</td>
<td>0.00</td>
<td>0.00</td>
<td>0.10</td>
<td>-0.40</td>
<td>0.00</td>
<td>-0.10</td>
<td>0.40</td>
</tr>
</tbody>
</table>
\[
\begin{array}{cccccccc}
1.00 & 0.00 & 1.00 & -0.13 & 0.00 & -0.25 & 0.00 & 0.25 \\
1.00 & 0.00 & 1.00 & 0.25 & 0.00 & 0.07 & 0.29 & 0.07 \\
3.71 & 0.00 & 0.00 & 0.25 & 0.00 & -0.64 & -0.43 & 0.34 \\
12.55 & AFTER ITERATION 5 \\
1.00 & 0.00 & 0.00 & 0.11 & 0.06 & 0.00 & 0.17 & 0.00 \\
1.61 & 0.00 & 0.00 & -0.39 & 1.56 & 1.00 & 0.67 & -1.00 \\
13.11 & 0.00 & 1.00 & -0.22 & 0.39 & 0.00 & 0.17 & 0.00 \\
4.28 & 1.00 & 0.00 & 0.28 & -0.11 & 0.00 & -0.32 & 0.00 \\
4.78 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
20.00 & FINAL TABLEAU \\
1.00 & 0.00 & 0.00 & 0.11 & 0.06 & 0.00 & 0.17 & 0.00 \\
1.61 & 0.00 & 0.00 & -0.39 & 1.56 & 1.00 & 0.67 & -1.00 \\
13.11 & 0.00 & 1.00 & -0.22 & 0.39 & 0.00 & 0.17 & 0.00 \\
4.28 & 1.00 & 0.00 & 0.28 & -0.11 & 0.00 & -0.32 & 0.00 \\
4.78 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
20.00 \\
\end{array}
\]

\[X = \begin{array}{c}
1.61 \\
2.78 \\
4.29 \\
0.00 \\
13.11 \\
0.00 \\
0.00 \\
\end{array}
\]

THE SOLUTION IS 20.00

DO YOU WISH TO SEE THE SOLUTION TO THE DUAL Y OR N

Y

Y

Y

Y

Y

Y

Y

Y

SP 1 = 0.00 
SP 2 = 0.00 
SP 3 = 0.00

DO YOU WISH TO RUN THE SENSITIVITY ANALYSIS

Y

RANGE IN WHICH BASIS IS UNCHANGED
<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>CURRENT COEF</th>
<th>ALLOWABLE DECREASE</th>
<th>ALLOWABLE INCREASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0.10E+01</td>
<td>0.40E-06</td>
<td>INFINITY</td>
</tr>
<tr>
<td>Y</td>
<td>0.10E+01</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>Z</td>
<td>0.30E+01</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
</tbody>
</table>

**RIGHT HAND SIDE ANALYSIS**

<table>
<thead>
<tr>
<th>CONSTRAINT</th>
<th>CURRENT RHS</th>
<th>ALLOWABLE DECREASE</th>
<th>ALLOWABLE INCREASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.12E+02</td>
<td>0.10E+02</td>
<td>0.19E+02</td>
</tr>
<tr>
<td>2</td>
<td>0.20E+02</td>
<td>0.04E+01</td>
<td>0.25E+02</td>
</tr>
<tr>
<td>4</td>
<td>0.10E+02</td>
<td>INFINITY</td>
<td>0.13E+02</td>
</tr>
</tbody>
</table>

**LPVAX**

1. ENTER THE PROBLEM
2. EXIT THE PROBLEM
3. SIMPLEX ALGORITHM
4. GRAPHICAL SOLUTION
5. EXIT

ENTER YOUR SELECTION

5. EXIT

HUMPRHEYS   losted out at 26-JUL-1986 15:44:39.76