AN USER FRIENDLY FRONTEND FOR A MPS PC PROGRAM

by

Jong-I Perng

B.S., National Taiwan University, 1960
M.S., University of New Hampshire, 1963

A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

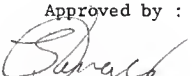
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

Approved by :



Major Professor

ACKNOWLEDGEMENT

"The best way to get something done is to begin"

Many people I would like to express my sincere thanks, for without their help and encouragement this report would not have been possible. I am especially indebted to my major advisor, Dr. Elizabeth A. Unger. Her great patience, firm guidance, kind understanding, and continued encouragement throughout my long extended years of study program, have given me the inspiration to finish this report. Dr. Unger's valuable suggestions and corrections on preliminary drafts have made this report readable and understandable.

A sincere appreciation is also extended to Dr. Orlan Buller, who has given freely his time for numerous consultations in preparation of this report. My work has been benefited from his helpful suggestions, his expertise and extended knowledge in linear programming. I also want to thank him for serving on my graduate committee.

Special thanks are also expressed to Dr. David Gustafson for serving on my graduate committee, and for his wise comments and attention to my work. His software engineering course has helped me in designing the computer program used in this report.

Thanks are due to the staff member of the Department of Computer Science for handling the paper works and getting me straight on the official procedures required by the graduate school. A sincere appreciation is also extended to the Department of Agricultural Economics for letting me use the computer facilities in preparing the draft and final copy of this report.

Above all, I would like to thank my family for their love, support, and understanding over a great many years of my study. My husband, Shian-Koong, has been the one who encourages me to start the master program and has supported me all the way until the program's finish. My son, William, who has been so sweet, considerate, and showing his willingness to take care himself so his mom could concentrate on her graduate work.

Last and foremost, I would like to dedicate my report to my mother, Mrs. May-Zuen Pan, whose life encountered so many hardships and she faced them with grace, gratitude, and working-harder attitude, which gave me the faith and all my endeavors.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW.

One of the problems in agricultural economic analysis is the optimal allocation of limited resources to a number of competing production activities in a range of possible alternative plans for a given objective. It is a problem in linear economics [DORM 58]; the best solution must be obtained subject to a simple mathematical rule, namely, the total amount of any resource used in these different production alternatives must not exceed the total amount available.

It is also a problem of decision making, the best solution must be chosen among all the possible competing activities in a way that minimizes cost or maximizes profit. Examples include (a) determining an income maximizing combination of products that should be produced by a farm of which a number of resources, such as labor, land, and capital, are limited, and are to be combined to produce one or more crops or livestock, and (b) selecting the cost minimizing irrigation scheme that should be adopted by a farm operates under situations where the natural gas price is expected to increase and the underground water table of wells on the farm is declining which will further restrict the supply of irrigation water.

Given the objective and a number of potentialities of production alternatives to choose from, there are numerous solutions that can be

obtained, but with the limited resources the process of reaching for one optimal solution is very complex. Thus, a systematic process is required; more specifically, a mathematical programming technique is needed for modeling physical situations that can be represented in linear form to find the best feasible solution to meet the given objective. Before linear programming (simply referred to as LP) was made available to agricultural economists, methods, such as calculus, and differential calculus, had been applied to the solution of this type of problems. However, these methods have been found to be less accurate and provide little assistance in solving these programming problems [HADL 62]. New methods of analysis have been developed, the most important ones are game theory, input-output analysis, and linear programming. They originated separately and gradually emerged together [DORF 58].

1.1.1 History of Linear Programming.

Game theory was developed by John von Neumann during the late 20's to the early 40's[NEUM 28,44]. His linear model of an expanding economy [NEUM 45] is an application of mathematics in economics. It is identical to linear programming; the analysis of linear inequalities. Leontief developed input-output models of the economy in 1936 and a full exposition in 1941 [LEON 41,51]. His work dealt with determining how much a change in the output of the intermediate goods would have to be in order to meet the changes in the output of any final product. The input-output model does not seek optimization; instead it solves the system of simultaneous linear equations. Input-output analysis may be considered

as a special case of linear programming. And the linear programming may be thought of as a generalization of Leontief's input-output models; it maximizes or minimizes some linear objective functions with resource constraints. Linear programming was developed by George B. Dantzig during World War II. He was a member of the U.S. Air Force group under the direction of Marshall K. Wood worked on an allocation problem, SCOOP (Scientific Computation Of Optimal Program) for the U.S. Air Force. Dantzig formulated the linear programming problem and developed the simplex method (see 1.1.3 The Simplex Algorithm) for planning and solving the allocation problem in 1947. His methods was made available to the general public in 1951 [DANT 51]. Since then, rapid progress has been made in the areas of theory and applications of linear programming [DANT 63]. Along with the development of electronic digital computers, linear programming has become the most important analysis tool for programming problems in the linear economics (or linear programming problems, see Appendix D, The LP Model Library).

1.1.2 Basic Concepts of Linear Programming.

A typical linear programming (LP) problem in agricultural economic analysis has three quantitative components: an objective, production alternatives, and resources constraints. It is a quantitative analysis in which all the relations are (1) linear additive, (2) divisible, (3) proportional, and (4) non-negative [Thom 71]. Thus, an LP model is a set of mathematical expressions that quantitatively represents an LP problem; it consists a linear objective function and a system of linear inequalities (represent a set of constraints), they may be expressed in

"Canonical Form" (see Appendix E, Mathematical Forms of LP Problems) [Bart76] as follows:

$$\text{Maximize} \quad Z = c_1X_1 + c_2X_2 + \ldots + c_nXn$$

$$\text{Subject to:} \quad a_{11}X_1 + a_{12}X_2 + \ldots + a_{1n}X_n \leq b_1$$
$$a_{21}X_1 + a_{22}X_2 + \ldots + a_{2n}X_n \leq b_2$$
$$\vdots$$
$$a_{i1}X_1 + \ldots + a_{ij}X_j + \ldots + a_{in}X_n \leq b_i$$
$$\vdots$$
$$a_{m1}X_1 + a_{m2}X_2 + \ldots + a_{mn}X_n \leq b_m$$

and $X_j \geq 0$, $i=1,2,\ldots,m$, $j=1,2,\ldots,n$.

The X_j's represent the levels of the possible alternative activity variables (the decision variables). The b_j's, c_j's and a_{ij}'s are constants and may be positive, negative or zero. The a_{ij}'s (the coefficients) are the amount of ith resources required by one unit of jth activity, X_j. Z is the value of the objective function that is to be maximized or minimized. The c_j's represent the amount change as a result of selecting different combination of X_j in order to improve the overall effectiveness of the model.

A canonical form can be converted into a "Standard Form" by adding a disposal variable (represents the unused resource) to each of the constraint inequalities. As a result, the system contains a set of equalities plus an objective function (see Appendix E, Mathematical Forms of LP Problems]

A system of equalities forms a convex feasible region with finite many corner points [Hill74]. The feasible region plus a linear objec-

tive function, the linear programming problem should have an optimum solution at a corner point, or if there is no better adjacent corner point to be found.

Having these properties, a linear programming problem can be solved by a mathematical procedure that will systematically and iteratively select plans which increase profit or reduce costs until an optimum feasible solution is reached. The method developed by Dantzig [Dant 51] for setting up an initial plan and systematically replacing it with a better plan is called the simplex algorithm.

1.1.3 The Simplex Algorithm.

The computations in the simplex algorithm are based on relatively simple matrix algebraic manipulations; addition, subtraction, multiplication, and division to solve the problem. The set of inequalities are rewritten into a set of equalities by introducing one disposal variable (to represent unused resource) to each constraint function in the system of inequalities. Adding the disposal variables ensures that thee number of equations is always less than the number of variables, which guarantees a solution. The algorithm requires all the basic data to be converted into the relevant quantitative data. They include: (a) supplies of limited resources, (b) input-output coefficients which is the unit of resource required by a unit of decision variable, and (c) product and resources prices [HEAD 60]. In the following, we present an example to demonstrate how a LP model is transformed from a system of inequalities to a simplex form.

1.1.4 An Example of Linear Programming Problem.

A farm has 12 acres of land, 48 hours of labor, and $360.00 of capital available to produce corn, soybean, and oats[Bene 73]. The problem is what crops to raise to maximize net return over the variable costs. Information needed to solve this problem are:

(1) Units that the production activities are defined in are acres.

(2) Input-output coefficients; units of resources required by one acre of crop produced:

Corn: 1 acre of land, 6 hours of labor, and $36 of capital.

Soybean: 1 acre of land, 6 hours of labor, and $24 of capital.

Oats: 1 acre of land, 2 hours of labor, and $18 of capital.

(3) Net returns are the values of gross sales over the variable cost to produce one acre of crop:

Corn: $40 per acre,

Soybean: $30 per acre,

Oats: $20 per acre.

Let: X_c = Units of corn produced

X_s = units of soybean produced

X_o = units of oats produced

Z = the return over variable cost

The objective is to maximize the net return, Z. The system of inequalities are stated as follows:

$$\text{Max.} \quad 40X_C + 30X_S + 20X_O = Z \qquad [1]$$

$$1X_C + 1X_S + 1X_O <= 12 \text{ acres of land} \qquad [2]$$

$$6X_C + 6X_S + 2X_O \leq 48 \text{ hours of labor} \quad [3]$$

$$36X_C + 24X_S + 18X_O \leq \$360 \text{ of capital} \quad [4]$$

Let: X_A = the units of unused land,

X_L = the units of unused labor, and

X_M = the units of unused capital

to represent the disposal activities and each of these disposal variables is added to its corresponding inequality, we arrive at the following a system of equalities:

$$40X_C + 30X_S + 20X_O = Z \quad [5]$$

$$1X_C + 1X_S + 1X_O + 1X_A + 0X_L + 0X_M = 12 \quad [6]$$

$$6X_C + 6X_S + 2X_O + 0X_A + 1X_L + 0X_M = 48 \quad [7]$$

$$36X_C + 24X_S + 18X_O + 0X_A + 0X_L + 1X_M = 360 \quad [8]$$

To apply the simplex algorithm, the system of equations are setup in table 1 as the initial plan for the linear programming computation.

Table 1: Simplex Tableau

Type resource	Resource rhs*	Real activities corn	soybean	oats	Disposal activities land	labor	capital
land	12	1	1	1	1	0	0
labor	48	6	6	2	0	1	0
capital	360	36	24	18	0	0	1
OBJ(net return)		40	30	20			

* rhs stands for Right-Hand-Side values of the system equations, i.e., the resource levels.

In this initial plan all the resources are in disposal, no production is taking place. It is a feasible plan but not an optimal solution. The simplex algorithm is based upon the fact that if there are m constraints (rows) in the matrix and these are linearly independent, then there is a set of m columns which are also linearly independent. Hence, any right-hand-side (RHS) can be expressed in terms of these m columns (called a basis). The simplex method uses these basic solutions, begins a series of modifications; a routine of substituting of one column (the activity or the decision variable) in the basis with one column not in the basis. The incoming and outgoing activities are determined by the outcome of computing and comparing the relative values of the program in terms of the objective function before and after the new unit of an activity is brought into the new plan in each iteration. The process is iterated until an optimum plan is reached that meets all of the criteria described in 1.1.3 of this report, including the requirement that all the column values be non-negative.

1.1.5 Mathematical Programming System.

Even with this small linear programming problem, the routine is long and cumbersome. The simplex algorithm routine is iterative and it can best be calculated using computers. Computerization began in the early 50's, its application was limited because the capacity of computer hardware was not adequate for handling large data information and processing the complicated algorithm. This problem has been solved after the larger and faster IBM computers were introduced. A well known

computer program is the Mathematical Programming System (MPS). It was first developed by the CEIR Inc. in Washington D.C., the package is called LP/90 for use with IBM computers 7090 console, on-line card equipment, and tapes. Its operating system is EDPM [CEIR 52]. Its data form and control language are similar to what it is in MPSX/370 today, except it requires user to supply the slack columns for the disposal variables. IBM released the first edition MPS/360 in 1967 [IBM 67] for use with IBM 360; it is a refinement of LP/90. It is written in the assembly language of OS/360. Its fourth edition was released in the early 70's for use with IBM 370. IBM also released two newer packages, MPSX/370 [IBM 78] and MPSX/370 Primer [IBM 79]. There are many commercial linear programming packages available in additional to IBM MPS packages, such as MPS III, developed by the Ketron, Inc. [KETR 84]. Its first release was as early as 1969, and IMSL released by the IMSL Inc. in 1984. Because IBM MPS is free, many universities and other organizations have continued to use MPS.

An MPS is a system of computer programs that solves a LP model. An MPS includes mathematical algorithms(the simplex method), database management procedures, and a host control language. It has been used widely not only for limited resource allocation problem in agriculture but also in industries, operation planning, engineering design and transportation problems.

For more than three decades (late 50's to the present), MPS has been a system that operates on IBM main frame computers. MPS is highly

technical and problematical oriented; an application that has been left to computer professionals. MPS runs on a main frame computer and it has no direct user interface, the philosophy of MPS designers was not to create an environment to put the user first and technology second. Users must enter their data in a pre-defined way so the mapping of users' inputs onto MPS database can be performed. Preparing computer inputs for MPS is a time-consuming task. Researchers usually do not concern themselves with the MPS data input procedures while designing, developing, and constructing an LP model. They normally represent their data in a matrix form similar to the simplex initial plan tableau shown in Table 1 of this report. Data preparation then becomes the task of a computer or a trained person to map this original data form onto a MPS entering form (let's call it a standard form). And usually there may be a third person, a computer key stroke operator, who inputs data into computer. A data preparation process, involving mapping, restatement of information, and transcription of data, requires extreme patience and carefulness in order to maintain data integrity and consistency. Editing, tracing information flows, and detecting errors become major tasks.

The introduction of the personal computer (PC), on-line computer use, and interactive PC software packages have broaden the spectrum of computer users. Using the computer and applications of computer programs are no longer the activities for specialists. Users' perceptions and expectations about computers have changed. Users judge the quality of an interface by the amount of support and the degree of

friendliness they receive from the interchange. The definition of "user-friendliness" is elusive but it becomes an important concept in a human-computer interface design.

1.2 THE PHILOSOPHY OF "USER-FRIENDLINESS"

The computer industry has discovered that users are human. The perception of "user-friendliness" is affected by human behavior and the "mechanics" of the human body and mind [Abbo83]. Many guidelines have been proposed [Shne79, Gain81, Abbo83, Crof84, Lieb84, McCr84, Riss84, Norm84, Lust85] to reflect human factors in user-computer interface designs, so the system can be friendlier to its user. However, there are no universal tenets to guide the design of the user-friendly interface, because different users will have different desideratum for "user-friendliness". Rissland suggests that knowledge about the user, user's tasks, tools available, domain, how to interact, and evaluation of effectiveness are all relevant to a good interface design [Riss84].

Thus for this report we will focus on the uniqueness of MPS PC users, their expectations and perceptions about user-friendliness while interfacing with the computer in application of a MPS program.

In the following seven categories we explain what "user friendliness" is for the MPS PC user:

1.2.1 User' desideratum.

MPS for main frame has been used for decades (1950's to the

present). There are many users of it and they are accustomed to its formats, its input form, and its output interpretations. To user who is accustomed to using MPS main frame, he/she expects the MPS program for PC to keep all the formats and interpretation exactly same as they are in the main frame LP program.

1.2.2 User's Task.

When a MPS main frame user applies MPS to solve a linear programming problem, such as the example described on page 3, he/she generally put all information together in a matrix form as shown in Table 1 on page 7 except the section of disposal activities because a MPS/360 or MPSX/370 program can generate them. In most research problems the matrix is large in size and sparse in non-zero elements, usually about 10% or less. A 75 X 75 matrix is considered as a small problem. Therefore the physical layout of a large matrix (say 255 X 2255) would be spread out more to the right than to the downward direction of a work sheet; its horizontal width may be 30 times more than its vertical length. The MPS algorithm requires all the basic data (cost, prices, coefficients, and values of resources) to be arranged in a particular sequential way. The standard form to prepare data for input is to sequentially restate non-zero elements of the matrix in column order. Using the example problem, the non-zero elements are organized in the following order and format:

```
corn    C         40
corn    land       1
corn    labor      6
corn    capital   36
soybean C         30
soybean land       1
```

soybean	labor	6
soybean	capital	24
oats	C	20
oats	land	1
oats	labor	2
oats	capital	18
rhs	land	12
rhs	labor	48
rhs	capital	360

The main frame users usually record these information on paper first, and then punch them on cards, or enter them into a LP file from keyboard. For MPS PC users, a direct transcription from the prepared matrix on paper to computer from keyboard is desired. Entering information in the matrix form to computer is a time-consuming and confusing task. MPS PC users are no longer just the computer professionals who are trained to cope with this type of task, so it is important to the MPS PC users to have a user interface that can ease the process for entering data into the computer, and can help them to minimize input errors.

1.2.3 Accuracy.

The accuracy of the solution of an LP problem is very sensitive to the coefficients that are entered by the users. A pre-execution data inspection for data integrity is essential to obtaining a degree of accuracy for the particular LP solution. In order to accomplish this, users should have direct control over data entry to the database with facilities for error checking, data flow tracing, and data searching and editing. If errors could not be detected and hence corrected before the program is executed, waste of a run would result.

In this situation we define friendliness to a MPS PC user is that the user should be able to work with a visual display of data stored by the system, and be able to select a small segments of inputs from a large file for inspection and operate on them.

1.2.4 Easy to use and low learning overhead.

For MPS PC users the computer is merely a tool to help them solve LP problems in which they have already invested enormous time and effort. Users attempt to accomplish their tasks as effortlessly as possible.

1.2.5 Robust.

There are three components that make up an interactive application program: data handling, processing, and dialogue management [Spra 80, Gain 87]. Many special programs are needed to carry out these functions. An interactive system should be designed for imbedded applications. Namely it should be robust enough that these special purpose programs can be automatically added to these functions, at least to the frequently used functions [McCr 84].

1.2.6 Safe exploratory environment.

Users not only face a machine but also an environment in which to use the machine and the application programs [Carr 81]. Since MPS PC users are subject to the time pressure in their research work, most likely they will take the "learn by doing' approach in respect to application programs. Therefore the environment should be safe for user

to experiment, users should not stumble over an unintentional mistake and there should not be any dangerous, or irreversible actions in which the user's environment and system are destroyed [McCr84].

1.2.7 Consistency.

As it has been mentioned above, MPS PC users most likely will take the "learn-by-doing" approach. They are what education psychologists call "rote learners" who are only interested in getting the right answers in respect to using application programs [Maye 81]. In learning to use the program, the human brain must use short-term memory to store all these how-to's and what-will-be's. However short-term memory is limited in capacity. Psychologists believe that the number of things which can be stored simultaneously in short-term memory is seven [Abbo 83]. It would be very difficult for users having learned one way to react in one stage of the interface and have to learn a new way to react in another similar stage of the interface throughout the application. The inconsistency may well drive the novice as well as the experienced user mad. A friendly interface design should have consistency throughout a system, across systems, and throughout time. Especially the last one; an unexpectedly long delay will cause panic and confusion.

1.3 INTERFACE MECHANISM.

There are many techniques that can be used to facilitate user/system interface. The most adapted ones are: frames, windows, prompting, command, menu, icons, and help facilities. An application

program may adapt one of these techniques, or two or more of them in their designs. The effectiveness in terms of user-friendliness is not a question of which technique should be used but rather a question of whether this technique or techniques used in the design can meet the users' desiderata about this application program.

1.4 REVIEW THE EXISTINC MPS PROCRAMS FOR PC.

There are many MPS programs for PCs that are currently available [Pfei83, Tate85, ACME85, ESP86, Scic86, Murt87, Mulv87, GAMS88]. We will evaluate three packages on their effectiveness based on our knowledge about the MPS PC users, users' tasks, LP problem domain and modalities.

1.4.1 Linear Programming Software by Brian Tate, dba Vintage Resources, Ltd.. Released in 1985 by Vintage Resources. Ltd..

The maximum size of matrix this program will handle is 75 X 75. The data creation program is not integrated with the program. The user is required to handle data in a separate manner, and thus it does not offer any help in editing, tracing, or correcting input errors, nor it is designed to facilitate modification which is the most frequently used function in solving the LP problems.

The system accepts a simplex tableau (see Table 1) as its input data. User may use a spreadsheet or a word processor software to create the matrix which must be set up with a certain format that is acceptable to the program. The created matrix file should be saved as a text file

on diskette in ASCII format with a filename prefix followed by a three letter PRN suffix.

The advantage of using Brain Tate's program is that the system accepts user's form (matrix form), and there is no restatement of information involved. However, entering a 75 X 75 matrix with 10% or less non-zero elements across a 80 X 25 computer screen is very hard in terms of locating the right coordinate for the non-zero entry. In the application of MPS main frame program, only the coordinates that have non-zero elements are required to be specified; the program will generate the zero elements automatically. The trade-off is that it requires users to map the matrix into a sequence of input statements as shown in the section 1.2.2 of this report.

As far as the interfacing mechanism is concerned, the system prompts the user with 3 choices, and offers very limited help and feedback to the user. Its response time is not consistent, and there are unexpectedly long delays. Its output is not the familiar MPS main frame form. The results are very hard for user to interpret.

1.4.2 MPS-PC, version 2.1 by George H. Pfeiffer, Department of Agricultural Economics, University of Nebraska-Lincoln. Released in 1983 by Research Corporation/Research Software.

The MPS-PC application program is designed for small size LP problems. It can accommodate a matrix of up to 50 constraint rows and 70 column activities.

The system contains two interactive programs, MPS-DATA and MPS-ALG, and two utility programs, MPS=123 and PC-X. The MPS-DATA is a data handling program which has the following application function:

(1). creates and saves an LP problem file,

(2). prints a file,

(3). modifies an existing file, and

(4). creates change rows or columns for post optimality procedures.

The MPS-ALG is a computing process program and solves linear programming problems using a revised simplex algorithm. These two programs interface each other through the system's menu selection mechanism. The MPS-DATA is invoked in the MPS-ALG, thus implementation of the above 4 functions are built with-in the program.

Users interact with the system by making selections from the displayed menus and typing answers to the prompted questions. Once a selection is made, users are led to answer a series of questions that are prompted to them; users can not navigate freely without answering them. Users must either answer the questions or choose to return to menu by pressing the F1 key and the return key.

There are two ways to enter data: using the MPS-DATA's file creation function, or using a separate system, the MPS-123's file conversion program to transform a file that has been created with the 1-2-3 spreadsheet to a file that is acceptable to MPS-ALG.

The file creation function is listed in the MPS-DATA menu and guides users through the input process. First, users are asked to key in row names (the constraint variables of a LP problem), and then the coordinates and the non-zero coefficients in these coordinates in column orders.

MPS-PC may be ideal for a very small LP problem (say a 20X20 matrix), because the system guides you through and the input is in column order as in the main frame MPS. But it is impractical if the LP problem is large, because users may very easily stumble over their own mistakes in the following respects:

(a). user can not see the data that they have just entered beyond the limits of the screen,

(b). if the user finds that he/she has just entered a wrong name or a wrong value, he/she can not back up to correct it; the user either must choose to end the entering process and lose all the input data that has just been entered, or proceed with the faulty file, and

(c). a serious problem arises when the following situation is encountered:

suppose there is a undetected typing error in a row name, for example a lamd instead of a land, when user follows the system's guide proceeds to the coordinate section and enters,

corn land 1 (see example on page 13, corn is the column name and land is the row name.)

the system will response,

row name not found in data.

Since the user can not see what is being typed in, there is no way for the user to find the "right" mistake. There is no other choice but to end the entering process and lose all the data that have been entered.

The MPS-123 program converts a matrix that are being created using Lotus 1-2-3 spreadsheet to be used by the MPS-PC. The required format for creating the file is very rigid. An incorrectly formatted file will be rejected by the MPS-123; it will not indicate what or where is the error occurred. User has to leave the MPS-123 system and go back to the Lotus and try again.

1.4.3 Linear Programming for the IBM PC, Eastern Software Products, Inc., released in May, 1986.

The ESP program allows users to solve LP problems with up to 255 constraints and 2255 variables, and up to 510 constraints and 2510 variables if the problem is sparse (10% or less non-zero elements). The system includes display editor, simplex algorithm, file management, and report generator.

Input process is handled by its display editor. In the case of a new LP problem, the display editor generates an equation-like tableau with default constraint names and variable names for the LP problem. The user may enter the matrix coefficients onto the corresponding

coordinates using the computer key board. In the case of an exiting problem file, the display editor will display the retrieved LP problem file on screen for the user to operate on.

Solution procedures are done by its simplex algorithm. The starting point of the algorithm can be an initial basis consisting of singleton and artificial variables, or can be a stored basis from previous solution.

File management does saving, retrieving, file listing, and file deleting. The report generator lets the user select among six different tables showing the problem information and the solution results.

All these functions can be accomplished within a single environment; user does not have to leave the system when switching from one function to another.

The interactive mechanism is menu driven and there is a prompting system. After the user has made a selection from the listed menu, the user is prompted to answer a series of questions to carry out the selected function.

The advantages for using ESP are:
(a).It supports reasonably large sized LP problems, which makes it possible for the MPS main frame users to adapt this system to solve

their LP problems on PC.

(b).It provides a total environment for users to accomplish various functions without having to leave the system.

(c).It lets users enter their LP problems as they are formulated without having to restate information. Its display editor gives users visual display of data stored by the system and allows users to operate on it.

(d).Its output interpretations are familiar formats to the users who are accustomed to the main frame MPS formats.

The problems of using ESP are:

(a) Its data form is confusing to the end user of LP models. The end user of this program finds the input lines complex and cumbersome (see the Section 2.3.5 of this report).

(b) Entering a large matrix (say 255 X 2255) onto an 80 columns by 25 lines computer screen is a difficulty task. Although the display editor automatically generates default names of constraints and variables which makes the locating of a coordinate a little easier, the system does not support functions for locating a particular cell of the matrix, nor does it support browsing and searching through the matrix.

Most end users are comfortable with ordinary data tables as a way of writing down the LP program. Entering data into computer from the prepared table in column order is more natural. ESP accepts previously-written sequential files. However it must be separately created outside of the ESP system. The format used to create the file is in row order rather than in column order. A prepared 510 X 2510 matrix sheet

may be over 200 feet wide (from the left to the right) and 10 feet long (from the top to the bottom). Layouting the sheet in front of a computer and trying to enter non-zero elements in row order is physically difficult. Entering data in row order requires user to look up the non-zero elements and locate their coordinates by moving his/her eyes along the 156 foot sheet, which can lead to mental disorientation, short-memory failure, and information corruption. On the other hand, entering data in column order, the user can use a single width sheet (8.5 inch wide consisting 10 columns and 510 rows) at a time. This is physically manageable for user in a fixed position.

(c) The interface system is 3-levels deep hierarchically structured menu. The system provides a Function Key line (line 23 on the computer screen) indicating menu selections for current level, but it does not provide information about what the current level is, and the path that leads to this level. Users may lose sense of position unless he/she recognizes the level by memorizing the menu structure. After a user has selected a function (by pressing one of the Function Keys), then the system prompting mechanism leads the user down to a lower level; user has to answer a series questions that are prompted to him/her. Once a choice is made intentionally or unintentionally, it is a "point-of-no--return" until the function is completely carried out. If the choice happens to be a mistake, then this mistake will destroy the LP problem that has just been set up and has not been saved before the mistake is made.

1.5 THE PROBLEM.

MPS PC users are accustomed to the standard simplex table form in preparing a LP model to be solved by a MPS program on the main frame, and they have very large LP problem to solve. They prefer an input format that can be entered in column order without mapping or restatements from their prepared matrix tables. Data handling is the most frequently used function among the three components of an interactive MPS PC program. Data handling includes input, editing, and modifying. Since most MPS PC users are subject to time pressure in research work, even though they are expert in their particular disciplines, mapping and restatement is often confusing. Therefore a direct transcription from a prepared matrix table into computer is preferable. The process for entering data should be as simple as writing them down on a piece of paper with a pencil.

Among the three MPS PC programs that have been reviewed in this report, the ESP program comes closest to the MPS PC users's tasks and their LP problem domains and modalities. In respect to data format specification, the three programs are different from each other and none of them having the same data format that is specified in the main frame MPS.

Our objective is to build a user control frontend to a MPS program to handle input process that will have the following characteristics:

(1). Make the transcribing from a prepared matrix into the computer in column order as easy, and as friendly as possible.

(2). The data information is arranged in the most basic form, i.e., the matrix table form so that each data item is only to be entered once.

(3). Automatically convert the user entered matrix into a LP data file in the data format required by the main frame MPS or in the format required by a MPS PC program. The generated LP data file is going to be stored in a floppy disk and is ready to be read into a MPS program.

The following chapters will describe the design and implementation of this user-friendly frontend (UFF) system for a personal computer. Chapter 2 gives the details of the system design and requirements, the data structure and the file structure of the program, and a brief description the algorithm. The implementation of the User-Friendly-Frontend and testing the system with a LP model that studies the impact of energy price on an average Northwest Kansas Farm are described in Chapter 3. Chapter 4 examines the results and discusses the limitations and the potential of user-friendly frontend development. Appendix A contains the user's manual. Appendix B contains the UFF program listing, Appendix C gives an UFF sample print out of the LP programming problem that uses the UFF program to enter its data into computer, Appendix D lists LP models in various research areas, and Appendix E presents four mathematical forms of LP problems. The appendices are provided for users who intend to use the program and may need to modify the program to fit their special situations.

CHAPTER 2

USER FRIENDLY FRONTEND SYSTEM

2.1 Objective

In designing the user friendly frontend program for a MPS package the objective is twofold; to assist the user to input LP data into computer in a friendly environment, and to create a LP data file that is acceptable to the linear programming software package. The following sections will present the design and the implementation algorithm of this program.

2.2 THE OVERALL SYSTEM DESIGN

There are four important steps in developing a system; preliminary design, detail design, coding, and testing. Design is a process driven by information gathered from requirements analysis. The designer should translate the requirement specification into a representation of software and should apply various techniques and principles to determine a device, a system, a process, and a language to use in the software representation [Pres82]. An extensive analysis of the requirements for the User Friendly Frontend program is given in Chapter one of this report. Seven principles that define a user friendliness environment is also discussed in detail in the previous chapter. Based on those principles and requirements, an overall user friendly frontend system is developed and is displayed in the following page where Figure 1 shows the architecture of the overall system.

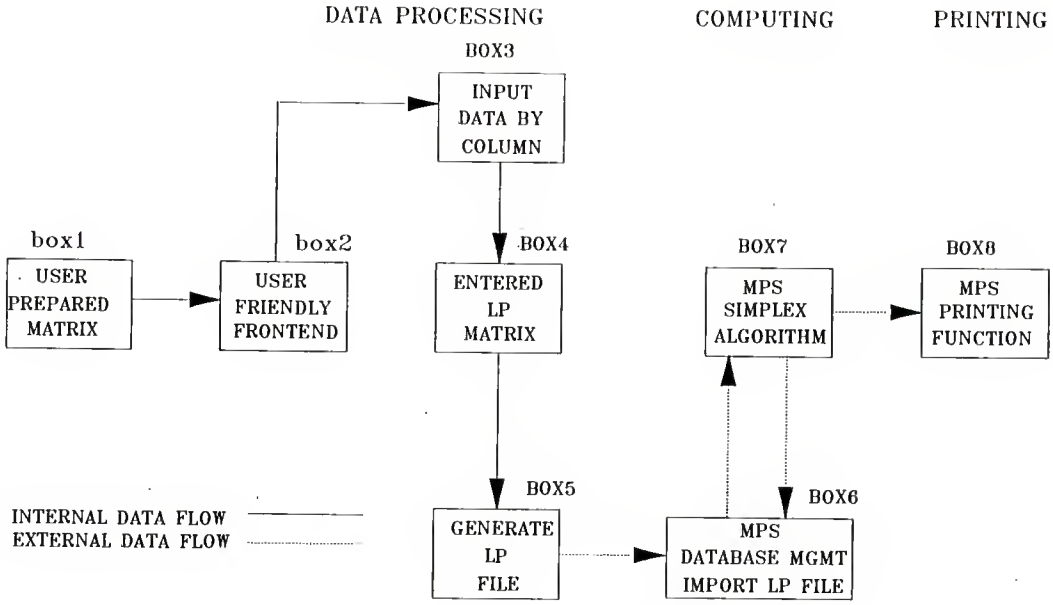


Figure 1 : Overall System Architecture.

Each of the boxes in Figure 1 has the following meaning:

Box 1 -Prepared matrix is the user's LP problem formulated into a tableau (see Table 1) that is ready for data entry processing.

Box 2- User friendly frontend (UFF) is a program that provides assistance to the user to enter data from the prepared matrix into a computer worksheet and supports data display and data editing. The system also converts the worksheet into data file to be used by a MPS program to solve the LP problem with the simplex algorithm.

Box 3- Input data in column order is part of the data handling process provided by the user friendly frontend program.

Box 4- Data display in matrix form is also a part of the data handling process controlled by the UFF to allow the user to view the entered data worksheet.

Box 5- Data information converting from matrix to a LP data file is another data handling process generated by the UFF program.

Box 6- Computation will be done by a MPS program. The MPS program imports the LP data file generated by the UFF system and applies the simplex algorithm to carry out the computation to reach an optimal solution for the given LP problem.

Box 7- Output is generated by the MPS program to print out the results for the given LP problem.

2.3 USER-FRIENDLY FRONTEND PROGRAM.

In this section we will describe the detail design for the system which includes hardware and software requirements, the format of screen

display, the menu structure, the data and file structures, and algorithm of the UFF program.

2.3.1 Hardware and Software Requirements

The application of the UFF program requires an IBM or IBM-compatible personal computer(either the XT or PC/AT model) with a hard disk, 640K of memory, must run under MS-DOS 3.0 or higher, and a mathematics coprocesser. The UFF requires the version 2.0 of LOTUS 1-2-3 program, and floppy disks for data storage. The user of this program needs to use the LOTUS 1-2-3, but does not have to be familiar with the LOTUS 1-2-3 package. The UFF program provides the user with a menu driven interface to let the user to select functions from the menu to carry out a task.

2.3.2 Display Screen Format

When the user first starts the UFF program, a portion of the 1-2-3 worksheet (user interface region) made up of horizontal rows (numbered 1498-2011), vertical columns (lettered A-Z, then AA-AZ, then BA-BZ, and so on to GV), and cells (the unit of worksheet that can store data. Each cell has an unique address that is the coordinate of column letter and row number) is displayed on the computer screen. Figure 2 shows how the 1-2-3 worksheet is partitioned into regions by the UFF program.

Only the user interface region is accessible to the user. The UFF program code region is protected to avoid unintentional alteration of the program code. The internal file working space is for the storage of

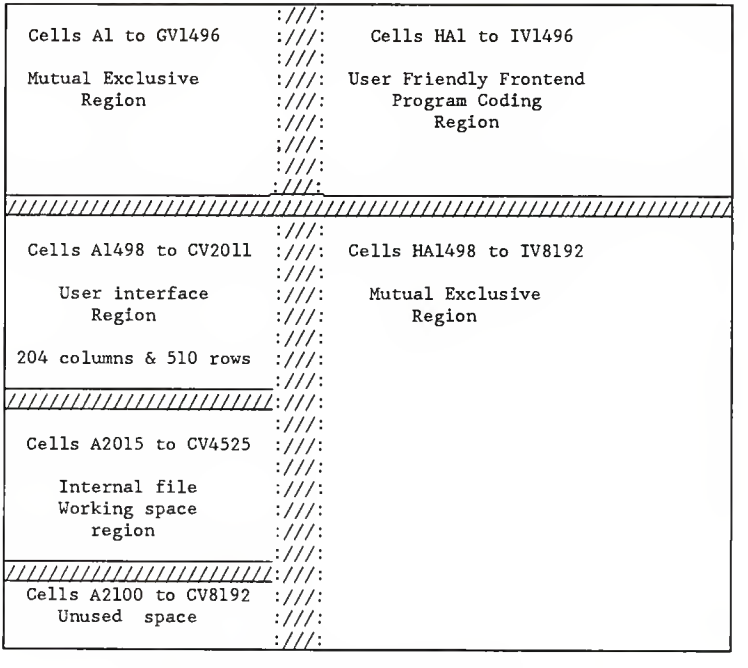


Figure 2. Worksheet Partitioning

the transposed ESP data form before it is saved on files.

The screen display format for the user interface region is shown in Figure 3 as follows:

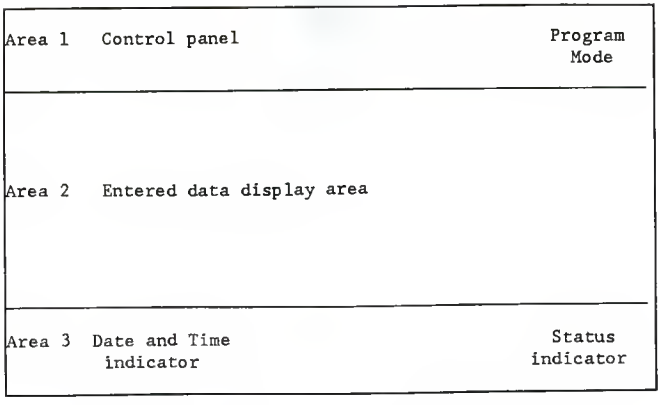


Figure 3. Screen Display - User Interface Region.

Area 1 is the control panel. It contains three lines of information about the current cell(indicated by a rectangular highlight called cell pointer):

The first line contains cell address, cell contents, column width, format and the current mode or state of the program (see Appendix C, Table C.1).

The second line displays the Main menu. It also displays a prompt or request for information and the user's response to the prompt.

The third line displays the submenu or the description the Main menu item currently highlighted.

Area 2 displays a window of any page of a LP problem data information that either generated by the UFF program or entered by the user. The window holds up to 8 columns and 20 rows of data and each page holds

maximum of 204 columns and 510 rows of data. The size of window is limited by the size of computer screen and the size of the page is determined by two factors; one the Lotus 1-2-3 worksheet, and the other is the file structure of the UFF program which will be described in the section 2-2-5 of this report.

Area 3 displays the current date and time, and the current status.

The screen display format will be kept the same throughout the entire process. So the user will not encounter any unfamiliar settings during any stage of the interface.

2.3.3 Menu Structure

The menu organization resembles a tree structure with the main menu located at the root and the submenus located at various branches of the tree. User may traverse freely between the main menu and the sub-menu as well as between any two submenus. The system allows the user to bring up the next menu or to return to the previous menu one level at a time. The user selects a menu item by entering the highlighted item or type in the first letter of any item on the menu. Once the menu selection is made, either the program will display a submenu for further selection or the program will execute the selected task.

The Main menu consists three functions; data processing, file managing, and printing. Figure 4 on page 37 displays the menu structure of the UFF program.

The data processing function helps the user to input data, to edit data, and to retrieve data. There are three basic data groups in a LP model; rows, columns, and rhs. To input them the user must first enter all the row names and their types before he/she can enter information into the column, and the rhs sectors. Because the row names are required for the program to generate a matrix tableau for user to enter data into the columns, and the rhs sector. The program will prompt the request to direct the user to proceed to the input process. The user will be allowed to add, delete, update, and browse the data stored in each cell of the worksheet by executing the proper menu items.

The file managing function helps the user to save and retrieve files. Each data file holds a page of data. The user may save and retrieve files by page. The UFF program will remind the user with a prompt to save the file if the number of columns reaches 200 during data input process. But the user may save the data any time before column 200 is reached by entering a file command from the file managing menu and type in the prompted information. Besides the page files, there are other three files; one is the row file that stores all the row names row types, one is the RHS file stores the rhs values, and the text file which is generated automatically by the UFF program in the format of equations that is acceptable a MPS PC program or by the MPS mainframe program.

The printing function helps the user to print a hard copy of data from the file specified by the user.

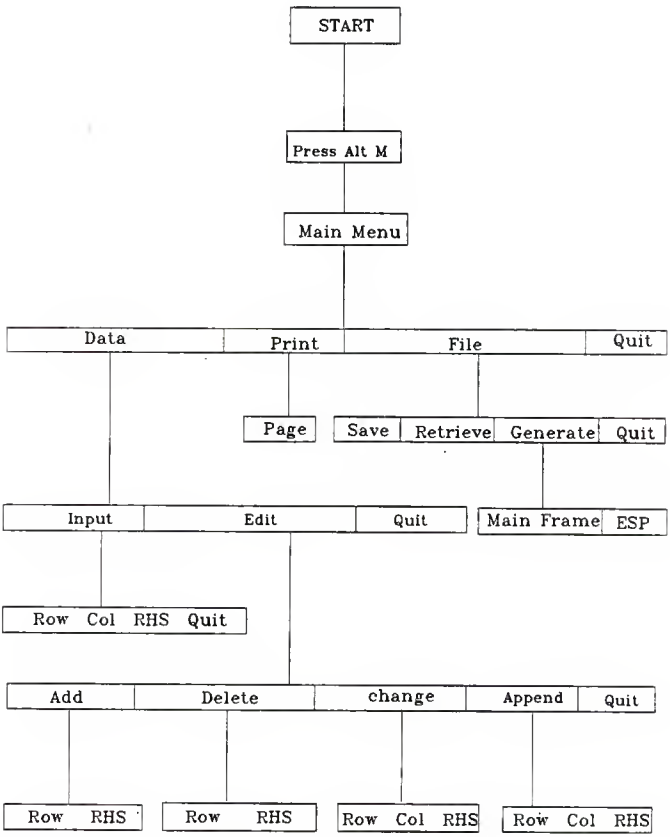


Figure 4 : Menu Structure

2.3.4 Data Structure

There are three basic types of data groups in a LP model; the rows, the columns, and the rhs's. The rows group includes the name of each row and its type. The row name is the variable name of each constraint in the LP model, such as land, labor, and capital. The row type is the type of constraint; none, less than equal to, equal to, and greater than equal to. The column group includes the column name that is the variable name of each production or other activities and the unit of resources required by each production activities as shown in the example problem in section 1.1.4 of this report. The rhs group includes the name of the resources and the level of these resources, such as, 12 acres of land, $360 of capital and 48 hours of labor, available at the initial plan of the LP model.

These data will be stored into the computer just as they are in the user prepared matrix sheet. However, because the number of columns of an 1-2-3 worksheet available for data storage are limited to 204 in the UFF program, the UFF program breaks up the user prepared matrix into pages, each with maximum of 510 rows and 204 columns. The first three columns of each page store the row numbers, the row type, and the row names, the 4th column stores the RHS values, and the rest of the 200 columns of the worksheet store the column names and the number of resources required by each unit of activity specified by each column name. The first two rows of column 5 to column 204 on each page are reserved for the column name and a line. The units of resources (or the coefficients) required by that activity named by the column name are

entered into each row of that column.

Data entered by a PC user from the prepared table into computer is guided by this friendly UFF program. The UFF program first will asks the user to enter the total number of rows of the LP problem. Then the UFF program will generate headings, that include the row numbers, the row type, and the row name, to make row input easier. The UFF interface Program will prompt the user with request for row information and store the user's input into the proper rows. Then the interface program will ask the user to enter the number of columns of the LP model and directs the user to enter data into rows in a column by column fashion.

Because the display screen of a computer is 20 rows by 8 columns (each column has a width of 9) in size, so the UFF program divides the page into windows, and each window is 20 x 8. The UFF will remind the user to visually review the data after a window is filled up. And the user may use the edit menu to correct the typing errors before leaving the current window and starting the next window. When each page is filled up, the interface program will prompt the user to save that page. The user is allowed to browse the page window by window by executing the browsing submenu.

The data structure is developed under two conditions; one is to create a friendly environment for user to input data, the other is to work within the limitation of the hardware as well as the software that are available to the UFF program. The next section will describe the

file structure that is designed to accommodate this data structure.

2.3.5 File Structure

There are three types of files: The first data file is the row file which contains the row names and row types entered by the user. The row file will be retrieved by the UFF program whenever there is a new page is needed. The row data information will be put into the first three columns of each new page. The second data file is the page files sequentially store the user entered column information, each page file holds up to 510 rows and 207 columns (the 1st page also includes the RHS column). One may visualize that each page is equivalent to a segment of the user prepared matrix sheet that having 510 rows and 2510 columns. Putting all the pages together will form the entire LP problem matrix. The user is allowed to view and edit the row file and the page files with the built in functions of the UFF program. The third data file is the LP data file which is generated by the UFF program. For example, the LP file will have the following data format for the ESP software package:

Line 0 $Z_{0j}, Y_{0.j}, X_{0j}$ the objective function,

Line n $RHS_{nj}, Y_{nj}, X_{nj}, C_{nj}$ the constraint functions,

where:

the X_{nj}'s are the column variables (the activity names),

the Y_j's are the row variables (the constraint names),

the C_{nj}'s are the non-zero coefficients at the nth row and the jth column,

Z is the initial value in the objective function,

RHS_j's are the values of the constraint level,

and j=1,2,...,m where m=1,2,...,2255, and

n=1,2,...,k where k=1,2,...,255.

The ESP data file format is accessible and acceptable by the ESP program. ESP reads this file into its system as an input file.

Using the example listed in Table 1 of this report, the input lines that appear on the sequential input file will be stated as:

line 0 Z,corn,40,Z,soybean,30,Z,oats,20

line 1 land,corn,1,land,soybean,1,land,oats,1

line 2 labor,corn,6,labor,soybean,6,labor,oats,2

line 4 capital,corn,36,capital,soybean,24,capital,oats,18

Without the UFF program, a user has to transfer the information from the prepared matrix and restate them according to the above data format. The user has to enter n+3 lines and repeatedly type the row variable names (n + 3) x j times, type the column variable names n x j times, and enter as many as (n +3) x j coefficients. For a 255 X 2255 size matrix, to enter 258 lines, type 581,790 many times of row names, 57,375 many times of column names, plus 581,790 of coefficients, could be a tremendous task, even the non-zero elements is 10 percent or less.

The UFF program will generate the non-zero coefficients and their

correspondent row and column names from the user entered page files according to the data format described above. This will not only save the data processing time and effort, it will also reduce the typing errors.

The UFF program generates the MPS main frame file according to the data format required by the main frame MPS (see example on page 12).

2.4 Transpose from the Matrix Format to the MPS Program Format.

After the user has entered a page of data into the computer, the UFF will prompt the user to check the entered data and make corrections if there are errors detected. The UFF user will have a choice to save the page data according to a MPS PC format or according to the MPS main frame format. The transposed statements will be appended to the MPS data file as each page being entered and saved by the user. The user will be informed while the transposing is in progress and after it is completed.

2.5 Description of Algorithm.

The UFF program consists four modules; the main program module, the row entry program module, the column entry program module, and the rhs entry program module. Each module has many subroutines which includes instructions and menus, argument passing between modules is done through menus. A menu can be viewed as the node of the branch which belongs to the UFF program tree with the main menu as the root; a number of lines of instructions and subroutine are attached to each

branch at the node to carry out the tasks listed in each menu.

In Chapter 3 detailed discussions are given on how the four modules are linked together, what are the major functions in each of the four modules, and what are the grammar structures for the programming language. The last two steps of a program design, coding and testing, are also extensively addressed.

CHAPTER 3 IMPLEMENTATION

3.1 Introduction.

This UFF program is implemented on a Zenith 386 personal computer, an IBM compatible, with 640 K-bytes of main memory, 32 K-bytes of display memory, and 4 K-bytes extra memory. A 40 MB hard disk drive, one double side, double density, 320 K-bytes, 5.25 inch disk drive, a color/graphics video display adapter, Intel 80386 main processor, and Intel 80287 co-processor, running MS-DOS 3.21 operating system. Its screen display is 80 characters x 25 lines and is a color monitor. The software packages selected for use are Lotus 1-2-3 release 2.0.

3.2 Programming Language.

The UFF program is written with the macro of Lotus 1-2-3 version 2.0 released by the Lotus Development Corporation, in 1985 [Lotu 85]. The Lotus 1-2-3 consists three programs; the Lotus worksheet, graphics, and database. Only the Lotus 1-2-3 worksheet program is used in implementing the UFF program. The Lotus worksheet program is accessed through the Lotus 1-2-3 Access System. The Lotus 1-2-3 worksheet program allows the user to create program with its macro language. The Lotus 1-2-3 macro is a quite limited language because of the restricted grammar. A macro program is written with a set of instructions made up of a sequence of keystrokes and commands that are pre-defined in the Lotus 1-2-3 macro lagnuage. The macro has 25 keystrokes and it also has total of 44 commands. The 44 commands includes 6 macro commands for controlling the worksheet screen, 10 interactive commands allowing the

end user to enter information as well as preventing the end user from interfering with the program, 11 commands for controlling program flow, 6 commands to manipulate data, and 9 commands to facilitate file management. It also supplies many built-in functions which includes 17 mathematical functions, 7 logical functions, 11 special functions, 18 string functions, and 7 statistical functions. The keystrokes are listed in Table 2. The commands and their descriptions are listed in Table 3 - Table 7, and the built-in functions that are used in the UFF program are listed in Table 8 as follows.

Table 2 : Macro Keystrokes.

Macro Key	Description
~	RETURN (referred to as tilde)
{down}	DOWN
{up}	UP
{left}	LEFT
{right}	RIGHT
{home}	HOME
{end}	END
{pgup}	PAGE UP
{pgdn}	PAGE DOWN
{bigleft}	BIG LEFT (move left one screen)
{bigright}	BIG RIGHT (move right one screen)
{edit}	EDIT
{name}	NAME
{abs}	ABS
{goto}	GOTO
{window}	WINDOW
{query}	QUERY
{table}	TABLE
{calc}	CALC
{graph}	GRAPH
{escape} or {esc}	ESCAPE
{backspace} or {bs}	BACKSPACE
{delete} or {del}	DELETE (use only EDIT mode)
{~}	to have tilde appear as ~
{{} and {}}	to have braces appear { and }

Table 3 : 1-2-3 Macro Commands - Control the Screen.

Keyword	Functioning
{BEEP}	Sounds the computer's bell or tone.
{INDICATE}	Changes the indicator in the upper right corner of the screen.
{PANELOFF}	Suppresses redrawing the control panel during macro execution.
{PANELON}	Restores control panel redrawing, undoing {PANELOFF}.
{WINDOWSOFF}	Suppresses redrawing the display screen during macro execution.
{WINDOWSON}	Restores standard screen redrawing, undoing {WINDOWSOFF}.

Table 4 : 1-2-3 Macro Commands - Control Program Flow.

Keyword	Functioning
{BRANCH}	Continues executing macro instructions located in a different cell.
{DEFINE}	Declaring an argument type in a subroutine call.
{DISPATCH}	Branches to a destination specified in a location cell.
{FOR}	Repeatedly executes the macro subroutine that begins at a particular location.
{FORBREAK}	Cancel execution of current {FOR} loop.
{IF}	Conditionally executes the command that follows the {IF} command.
{ONERROR}	Continues execution at a specified location if an error occurs.
{QUIT}	Terminates macro execution, returning control to the keyboard.
{RESTART}	Clears the subroutine stack.
{RETURN}	This is a subroutine return. Continues macro execution just after the location of the last {routine-name} or {MENUCALL} command.
{routine-name}	Calls a subroutine.

Table 5 : 1-2-3 Macro Commands - Manipulating data.

Keyword	Functioning
{BLANK}	Erases the contents of a specified cell or range.
{CONTENTS}	Places the contents of one cell in another cell as a label.
{LET}	Stores a label or number in a specified cell.
{PUT}	Stores a label or number in one cell of a specified range.
{RECALC}	Recalculates the formulas in a specified range, row by row.
{REGALCCOL}	Recalculates the formulas in a specified range, column by column.

Table 6 : 1-2-3 Macro Commands - Keyboard Interaction.

Keyword	Functioning
{?}	Ask for keyboard input, macro execution is temporarily halted.
{BREAKOFF}	Disables the BREAK key during macro execution.
{BREAKON}	Restores the BREAK key, undoing {BREAKOFF}.
{GET}	Stores a single character the user types in a specified cell.
{GETLABEL}	Prompts the user to type, and stores the characters as a label in a specified cell.
{GETNUMBER}	Prompts the user to type, and stores the characters as a number in a specified cell.
{LOOK}	Check to see if the user has typed a character.
{MENUBRANGH}	Sets up a customized menu with user-defined choices.
{MENUCALL}	Calls a subroutine in a customized menu.
{WAIT}	Suspends macro execution until a specified time.

Table 7 : 1-2-3 Macro Commands - Working With Files.

Keyword	Functioning
{CLOSE}	Closes a file that has been opened with the {OPEN} command.
{FILESIZE}	Determines the number of bytes in a currently open file.
{GETPOS}	Determines the current position of the file pointer in an open file.
{OPEN}	Opens a specified file for reading, writing, or both.
{READ}	Reads characters from a file into specified cell.
{READLN}	Copies a line of characters from the currently open file into a specified location.
{SETPOS}	Sets a new position for the file pointer in the currently open file.
{WRITE}	Copies characters into an open file.
{WRITELN}	Adds a carriage-return line-feed sequence to a string of characters and writes the string to a file.

Table 8 : 1-2-3 Macro Functions Used in UFF System.

Function	Formats, arguments, and Results
@INT	@INT(x) returns the integer part of x.
@MOD	@MOD(x,y) returns the remainder of x/y.
@@	@@(cell address) returns the value in the cell referenced by the cell address[1].
@CELL	@CELL(attribute,range) returns a particular piece of information, called an attribute[2], about a given cell.
@@CELLPOINTER	@CELLPOINTER(attribute) returns information, called attribute, about the current cell.
@STRING	@STRING(x,n) converts numeric value x to a string, with n places to the right of the decimal point.

1 A cell address can contain a cell name, a cell address preceded by a label prefix, or a string-valued formula.

2. An attribute can be an address, a row, a column, the contents, a type, a prefix(',",^), a protect status, a width, and a format. For detail please see the Reference Manual of Lotus [Lotus 85].

3.3 Macro grammars.

There are two types of macro grammars in Lotus macro language.
(1) Macro basic grammar: the basic grammar is a sequence of keystrokes to automate the menu selections and responses to prompt. The keystrokes are entered into each cell of the worksheet with the following format:

'/first letter of the menu items,{keystrokes},keyboard entries~

Where the " ' " is a label prefix sign to indicate that the statement is a label entry, the "/" is the menu key which will execute the selected menu items from the menu. The first letter of the menu items, the {keystrokes}, and the keyboard entries can be in any order depending upon the task required. And the "~" (the tilde) keystroke causes the instruction statement to be entered.

For example to erase a range of cell contents, say, the range is from cell a1498 to cell gc2200, the instruction statement will be:

'/REa1498..gc2200~

(2) Advance macro grammar: the grammatical structure has the following formats:

{KEYWORD}

{KEYWORD arg1, arg2,...,argn}

Where keyword can be any of the commands listed in Table 3. Some of the keywords do not need argument, some of the commands may have one or more than one arguments. It also can process more than one type of arguments.

The macro supports four argument types; number, string, location, and condition, where

Number is any single number(real, integer, and scientific notation) or expression (cell address, single-cell range name, and formula) that results in a numeric value. Where a range name is a named cell address, or a named range of cell addresses.

String is any sequence of characters but not formula.

Location is any range of one or more cells.

Condition is any logical expression.

The LOTUS 1-2-3 also has a menu format that allow a customized menu construction in the macro program as follows:

menu	menu item 1	menu item 2	Quit
name	menu description	menu description ...	Brief statement
	{BRANCH loc}	{BRANCH loc} ...	{Keyword}

Where menu name is a 1-10 character range name which is called from a {MENUBRANCH menuname} statement somewhere in the program. The menu items, such as Add, Edit, and Delete are given a brief description to let the user know the function of this particular menu item. {BRANCH loc} is a flow control statement that pass the macro execution to a subroutine at the specified location (a cell address or a range name) to

perform the task of the menu item.

The instruction statements in the macro program must have the correct grammatical structure. The macro language allows certain degrees of modular programming with its branch and menubranch commands, subroutine calls, and arguments passing modes.

The Lotus worksheet program can execute user created macro by invoking the name of the macro (a back slash and a letter); by pressing the MACRO(Alt) key and the letter name simultaneously. Once the macro is invoked, the user created program becomes active and the Lotus 1-2-3's menu and commands are submerged. Lotus 1-2-3 reads the instructions of the macro program and executes them accordingly to carry out the required tasks.

3.4 Coding

The UFF is an Lotus 1-2-3 macro program, named \M, has four program modules and each module consists instruction statements, subroutines, menus, and named range cells (see Figure 5). The UFF system contains over 700 lines of source code (see Appendix B, Program Listing). The source code mixes statements written in basic grammar, in advanced grammar, and in customized menu format. The program is a top down design. The modules are organized resembling a tree structure with the main program module at the root of the tree, and the other modules are at the nodes of the tree branches. Figure 5 in the following page shows the program flow chart.

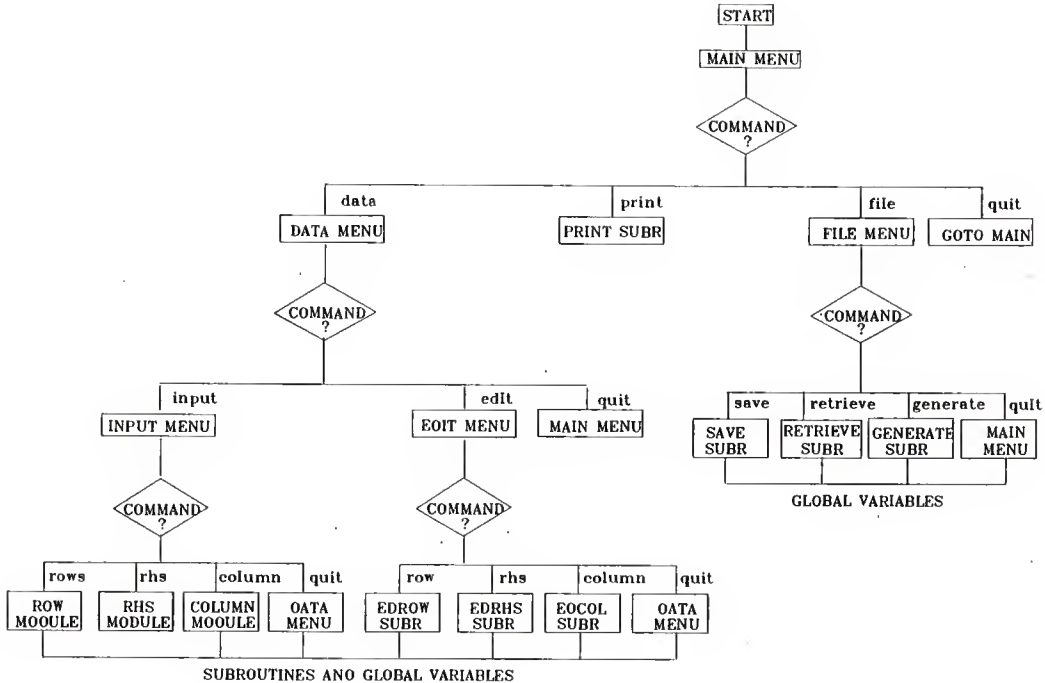


Figure 5 : Main Module Flow Chart

The first line of the main module consists three commands written in advanced macro grammar to perform three tasks; to set the cell pointer at the upper left corner of the user interface region, to start the macro execution with main menu, and to instruct the UFF system that in case an error occurs the execution is branched to a fix subroutine to tell the user to fix the error and continue the macro execution after the error is fixed by the user. The rest of the coding in the main module are customized menus which consist menubranch to other menus, menucalls for submenus, and subroutine calls. The main menu is at the first level of the tree. The data menu and file menu are at the second level. The input menu and the edit menu are at the third level. The row module, the column module, and the rhs module are attached at the bottom level of the tree structure.

Once the macro \M is invoked, the main module displays the current cell pointer on the first line, the main menu on the second line, and the menu item description of the highlighted menu item(bold letters) on the third line of the control panel on the screen display of the user interface region of the worksheet as in Figure 6 in the following page.

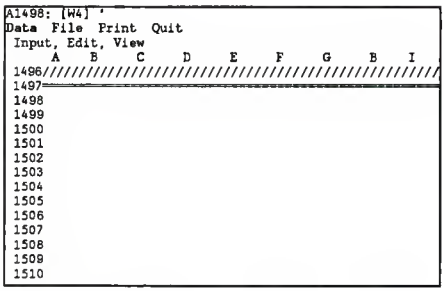


FIGURE 6: SCREEN DISPLAY - WHEN MAIN MODULE IS INVOKED.

Figure 7 in the following depicts the screen display when the row module is invoked and the user responses to the prompt: "Enter Total Number of Rows" by entering from the keyboard a number, say "10". And Figure 8 in the following page describes the flow charts of the row module.

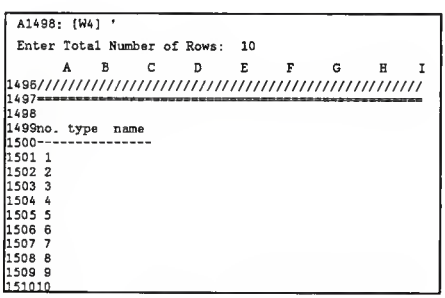


FIGURE 7: SCREEN DISPLAY - WHEN ROW MODULE IS INVOKED.

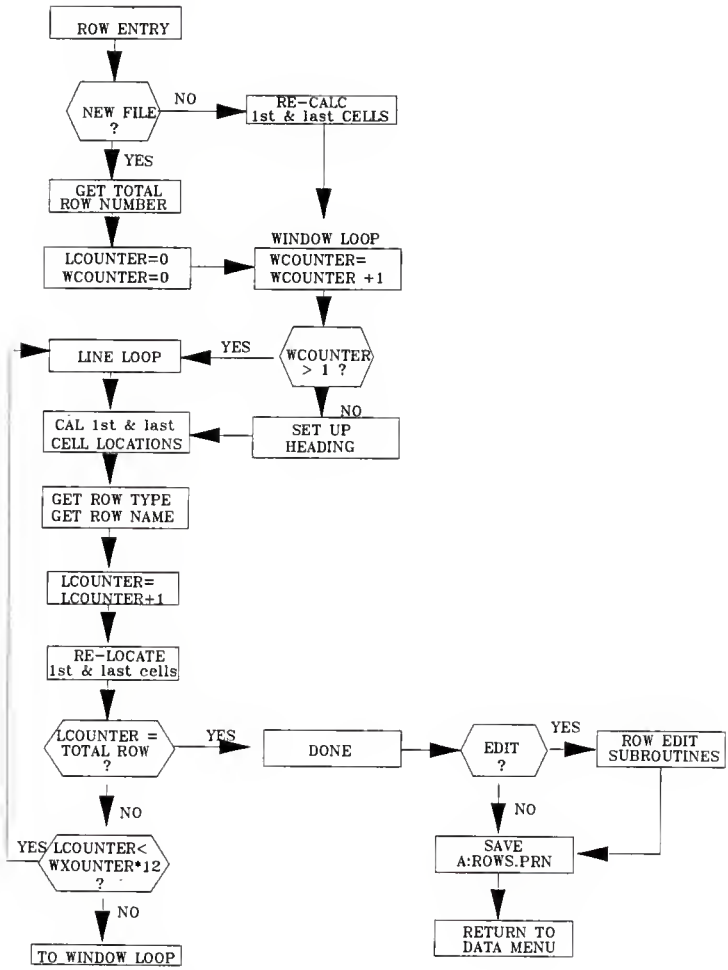


Figure 8: Row Module Flow Chart

Once the execution is passed over to the row module from the main module, the computer prompted sentences will appear at the control panel asking the user to enter a number representing the total row number of the LP matrix. Then the system will compute all parameters based on the information entered by the user. Values, such as the number of lines for each display window, the line counters, and the last cell pointers are computed. Next the system will generate three row headings; no., type, and name, and also draw dash lines below each of the headings. generate the numbers for the rows that entered by the user below the heading. The UFF has 10 subroutines to perform these functions: automatically moves the cell pointer to the next cell after the user entered value is stored in the current cell, does error checking, page checking, row file saving tasks. It also make submenu calls and subroutine calls to carry out the user's commands: edit, insert, delete, append, and browse.

After the row entry is completed, the user should select data-input-rhs command sequence to start rhs entry. The execution is then passed to the rhs module by the main module. Figure 9 presents the screen display when the rhs module is invoked.

As the macro execution is passed over to the RHS module by the main module, the control panel of the screen display will prompt the sentence "Enter RHS value in column D" and put it on the second line waiting for the user to enter rhs data into the current cell. The module has 5

subroutines for these functions: generating the heading, dash line drawing, automating the cell pointer movements, allowing error checking, editing, and file saving. Figure 10 depicts the flow chart of the RHS module.

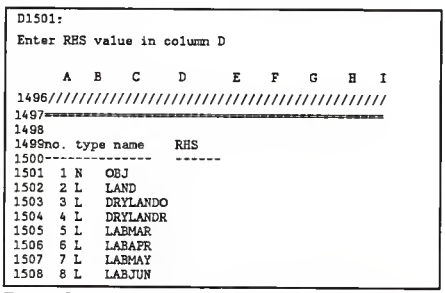


FIGURE 9: SCREEN DISPLAY - WHEN RHS MODULE IS INVOKED.

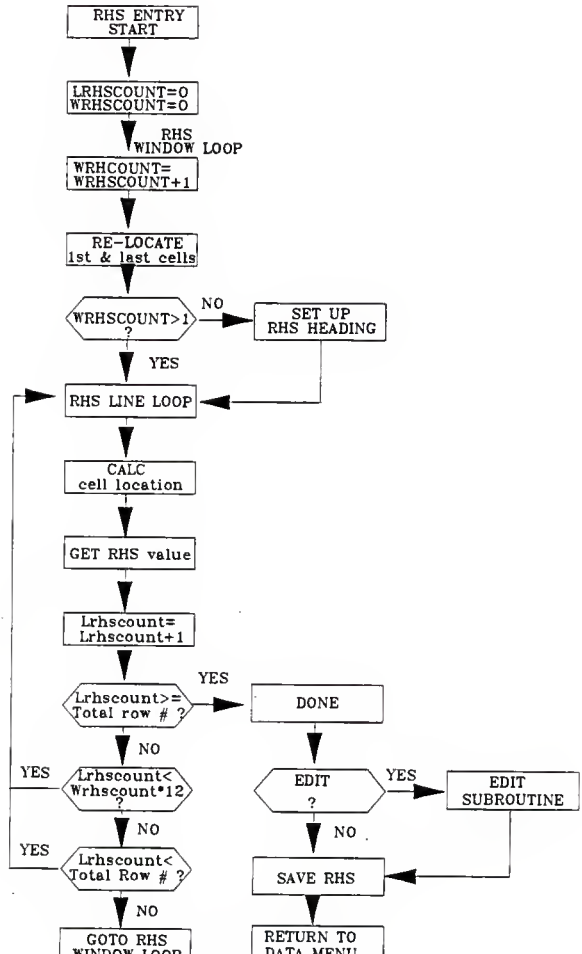


Figure 10 : RHS Module Flow Chart

When the RHS column entry is completed, user may select the Data-Input-column command sequence to enter the variable names and the coefficients into computer. When the commands are given, the column module continues the macro execution. The screen display will prompt the user with "Enter number of column:". The UFF system will take the user's response from the keyboard, say "5", to determine the total number of pages (each page is 200 columns, thus 5/200 is one page) for the entire matrix. A page number will be indicated at cell A1498 and the column numbers in the column heading area with dashed lines will be also generated by the system. The module consists 6 subroutines to determine page numbers, to retrieve row and rhs files, to allow error checking, to automate the cell pointer movements, and to save files by page. Figure 11 shows the screen display after the user gives the column input commands and types in "5" to response the prompt "Enter total number of columns:".

```
E1499:
Enter column name(1..8)
      A  B  C      D      E      F      G      H      I
1496//////////////////////////////////////////////////////////
1497==========================================================
1498 page 1                   1      2      3      4      5
1499no. type name    RHS
1500------------    -----    ------ ------ ------ ------ -----
1501  1 N   OBJ       0
1502  2 L   LAND     1695
1503  3 L   DRYLANDO 213
1504  4 L   DRYLANDR 564
1505  5 L   LABMAR   186
1506  6 L   LABAPR   192
1507  7 L   LABMAY   202
1508  8 L   LABJUN   259
1509  9 L   LABJUL   259
1510 10 L   LABAUG   259
```

FIGURE 11: SCREEN DISPLAY - WHEN COLUMN MODULE IS INVOKED.

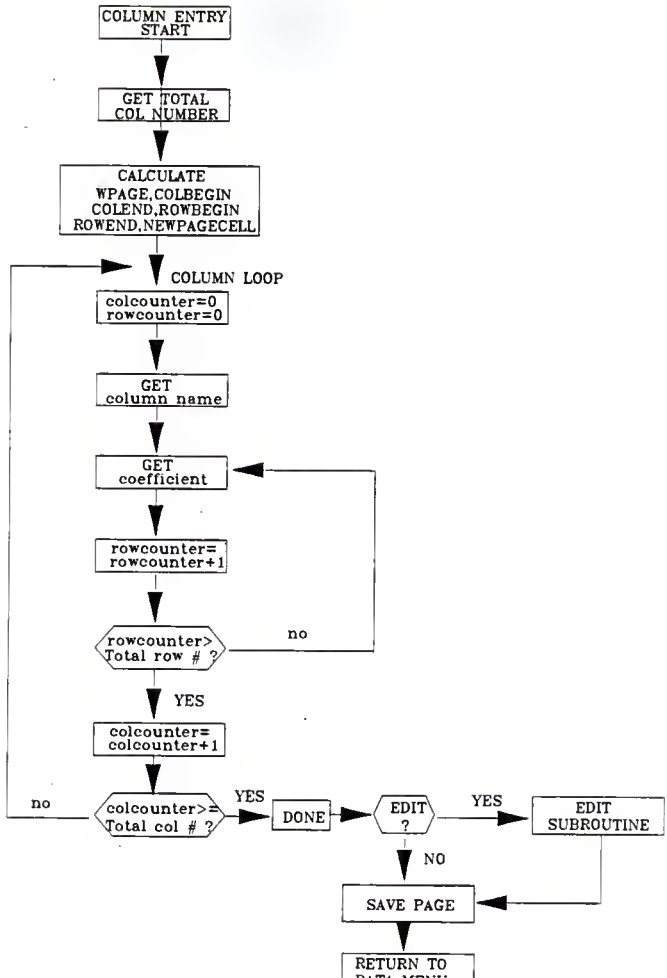


Figure 12 : Column Module Flow Chart

Figure 12 in the above page displays the column module flow chart. The other functions; printing, file saving, file retrieving, and the file transposing for export to a MPS PC or to the MPS main frame are done by subroutine calls from the main module's menus.

The program coding is typed in the cells of the Lotus 1-2-3 worksheet, which occupies part of the worksheet (see Figure 2 - worksheet partition) saved under the file name UFF.WK1 on a floppy disk. Since the user interface region, the internal file working region, and the program source code share the same worksheet, the UFF program does not allow insertion or deletion of columns by the user to avoid any unintentional alteration of the source code. This is because the 1-2-3 worksheet program automatically recalculates the cell addresses for the entire worksheet whenever there is a column that are going to be deleted or inserted. The macro program is designed for the user to append the columns at the last page of the LP matrix, if there is any column to be added to the data file.

3.5 Using the UFF System.

The UFF is a menu driven and user interface program with many of the features that will meet the criteria of an User-friendly program to a MPS PC user. The user of this UFF system need not to be an skillful user of the LOTUS 1-2-3 package in order to use the UFF system. The UFF user needs only to familiarize himself/ herself with the basic keystrokes and a few keywords listed in Appendix A, the user's manual for the system. The user gives a command by pressing the highlighted

menu item or by typing the first letter of the menu item. There is no computer jargon; everyday English is used to prompt the UFF user with request for information to perform the task selected by the user from the menu or submenus. The user responds to the prompt by typing in the information requested. The information requested is either a variable name, or a numerical value, or simply a character, such as Y for yes, N for no, and S for saving files. If the user enters data with the wrong data type, an error message will be indicated at the control panel asking the user to re-enter the data. The UFF user must complete the data entry in the row section first, then the RHS section, and the column entry comes last. If the Coefficient is a zero, a zero must be entered.

3.6 A LP Model - The Impact of Energy Price on a NW Kansas Farm.

The LP problem model selected to test this UFF system is "The Impact of Energy Price on a Northwest Kansas Farm". It is developed by Orlan Buller[1] and the author[2] of this report. It is an economic model to study how to maximize net returns with a given energy price for an representative irrigation farm in Northwestern Kansas.

The basic information needed to build this model (simply called energy model) are:

---

[1] Orlan Buller, Professor and Agricultural Economist, Department of Agricultural Economics, Kansas State University, Manhattan, Kansas.

[2] Jong-I Perng, Research Assistant to Dr. Orlan Buller, Department of Agricultural Economics, Kansas State University, Manhattan, Kansas.

The basic information needed to build this model (simply called energy model) are:

(1) The constraint variables which include land, labor, fertilizer, capital, machinery and water.

(2) The crops that are suitable to be produced in Northwestern Kansas, such as, wheat, corn, grain sorghum, and alfalfa.

(3) Irrigation systems, and pumping activities.

(4) Coefficients that represent the requirements for land, labor, fertilizer, water and other resources by each crop.

(5) Production cost, energy price, labor costs, fertilizer price and sale price of each crop.

With this basic information the model can be computed by a MPS PC program and obtain an optimal solution. It will be a over simplified solution which does not resolve the real farm situation, other factors will be considered in the discussion that follows. One of the advantages for using a LP model to conduct an economic analysis is that the LP model can represent most if not all of the possible alternatives in terms of resources, crop type, production strategy, and other factors that a farm can have. The solution of a LP model can be significant at the most detailed level that applies to a real farm situation.

Thus any basic information can be geometrically expanded into hundreds of alternatives even for a very simple analysis. In Northwestern Kansas, a typical farm has two types of land resources, owned land and rented land, that are available to the farm. To

consider this land ownership in the energy model, the size of the LP model is doubled. There are two types of irrigation systems that are commonly practiced in the region; flood irrigation and center pivot irrigation system. To model these two irrigation systems in addition to the land ownership, the LP model size is quadruple over the size of the basic model. There are also many possible irrigation regimes, i.e., the proper timing and quantity of water application for each irrigated crop, depends on the crop's plant growth stages. In this energy model, there are 10 alternative irrigation regimes for irrigated corn, 9 possible irrigation regimes for irrigated grain sorghum, 7 potential irrigation regimes for irrigated wheat, and one for alfalfa. And there are additional choices for wheat and grain sorghum crops, dryland or irrigated. With these expansions the original LP model has a magnitude of [2x2x(10+9+7+1)+(2x2)] (land ownership x irrigation system x crops x irrigation regimes+ land ownership x dryland crops) times as many as the basic model.

This model is for studying the impact of energy price on irrigation, thus irrigation activities carry more weight in the model. The irrigation activities include: (1) pumping water from wells, (2) water delivered to crops from pumps, (3) water stored in soil from pumps, (4) rainfall water stored in soil, and (5) water supply to crops from soil. The timing of irrigation will affect the crop yields which in turn will affect the profit of the farm, each of these irrigation activities are divided into 14 time periods; each time period identifies a critical water demand stage during a crop's development from planting to

harvesting. To link the crop activities to irrigation activities, the energy model is expanded to [2 x 2 x 4 x (10+9+7+1) x 5 x 14 + (2x2)] (land ownership x irrigation systems x crop types x irrigation regimes x irrigation activities x crop growing stages + landownership x dryland crops) times as many as the basic model. In addition to the irrigation activities, there are 10 labor hiring, 3 fertilizer purchasing, and 4 crop selling activities plus the transfer activities that serve as linkage in the model.

The energy model mushrooms to a 393 rows by 1069 variables LP matrix with 1.28 density (i.e., there are 5380 non-zero elements in the matrix) that has sufficient details to describe an average irrigation farm situation in Northwestern Kansas for economic analysis on energy price-irrigation-crop relationship to estimate the most profitable water use at a given energy price level. More details can be added, such as, well yields (gallon of water produced by a well per minute), different land rental arrangements between farmer and land owner, and participation or not in government agricultural program. Each of these details may further enlarge the size of the model and complicate its structure.

The Kansas LP model is not a complex model, its matrix size is considered to be medium sized. There are many models that are more complex in structure and much larger in matrix size among the list in The LP Model Library [see Appendix D]. One of the characteristics of a LP model is "large". Thus the point needs to be stressed here again is

that to help the user to input a large data set into computer, a front-end program with capability to minimize data entry error and mainatain data integrity are more important than a front-end program with promise of high input speed. Actually, data incorruptibility means speed, because an error-free data set will save a great deal of a researcher's time in detecting, tracing, and eliminating the input errors.

CHAPTER 4

SUMMARY

Linear Programming (LP) is a very important mathematical procedure to solve a large mathematical programming problem. With the development of the simplex algorithm and electronic computers in the 1950's and 1960's, and with the fast growing usage of PC's in the 1970's and 1980's, a new class, learn-by-doing computer users has emerged. Motivated by the fact that entering a large LP data set into a computer is a time consuming and confusing task for most of the linear programming application packages, the User-Friendly-Frontend program is developed to improve the data preparation procedure.

4.1 Limitations.

The prepared LP problem matrix is a large two dimensional table. To design a program with a user friendly environment that will help the user to transfer the information from paper into computer without any restatement or transformation of data, a spread sheet software is a logical choice. Among all the spread sheet packages, the Lotus 1-2-3 is chosen because many of its user interface features are included in the seven principles that define a user friendly environment for the PC users. On the other hand the Lotus 1-2-3 worksheet macro programming language serves the purpose reasonably well even though not without limitations.

(1) One of the limitations is that the Lotus 1-2-3 worksheet

limits the number of columns to 256. The column limitation causes fragmentation because a large LP matrix is usually over 1000 columns, the UFF program must be designed such that the matrix is stored into many page-files rather than one complete file. The user has to mark them on his/her matrix sheet in order to have information to retrieve the desired page for reviewing. The fragmentation problem and the amount of memory that is available to the UFF program limits the number of variables in each ESP equation to be generated at a time.

(2) The 1-2-3 worksheet program has a feature that it will automatically renumber the columns and redraw the entire worksheet whenever there is an insertion or deletion of a column is carried out by the user in the user interface region. These will affect the index and alter the formulas that are used in the instruction statement of the UFF program coding that shares the same worksheet with the user interface region. The UFF is designed to avoid to use this feature, which prohibits the user to insert or delete columns. If an insertion or a deletion is needed, the user may append the columns at the very last page of the file for additional columns, and erase the contents of that column and leave the column blank for the column that is going to be deleted.

(3) Because of the column limitations of the Lotus 1-2-3 worksheet, the UFF program does not include database query function to let the user find a particular record for reviewing, instead a browsing

menu is provided to facilitating a visual searching for the desired items.

4.2 Future Improvement.

Future direction for transporting the format of an input file to a format that is acceptable by a MPS software package should not be limited to only one PC software and should not be limited to just PC software. The file export function needs to be extended to a menu format which has different transpose subroutines, so that the user may have a choice as to which MPS ,PCs or main frame, the input file is to be exported. The expansion could be accomplished by building a large database in the input data format for exporting file selection. The format database feature will greatly broaden the potential usage of the UFF system.

In terms of user interface, the UFF system could build a large database of menus to speed up the response to selections. If the future release of Lotus 3.0 shall increase its worksheet size columnwise, a database query function should be added to speed up record searching. The UFF also could improve its interface design based on the concept of frame. The notion of frame meant a "structured screenful" [Robe81]. A "structured screenful", according to Robertson, is that everything the user could see on the terminal screen at one time. With the advent of hardware, operating system, and high-resolution screens, implementations of UFF could have several frames to be displayed simultaneously on the screen to list all the menus and options.

For the next 10 years, if the new DOS[3] idea becomes reality which will let the PC user to run multiple applications at the same time and allow data to be move between those applications, the UFF may move its data to any MPS PC programs without having to leave the system, and may not even have to change the format of the input data.

4.3 Conclusions.

The project has demonstrated that the user-friendly-frontend system is a useful and effective program to help lessen the burden of the data preparation process for people who uses MPS to solve a large LP problem. The UFF is particularly useful in helping the main frame MPS users, using a PC, to process the LP data set. However, because of time, manpower, and the inherent limitations of Lotus 1-2-3 worksheet and its macro language, the UFF may have room for refinements, written in a highly structured programming manner to improve its efficiency. Most of all the end users' experience with this system and their feedback will help the designer reveals more new areas for future improvement.

---

[3] William H. Gates III and the Operating System division of his Microsoft Corp have been engaged in developing an operating system that will take the advantages of the Intel 80386 microprocessor to allow multitasking applications for several years since early 1980s.

BIBLIOCRAPHY

[Abbo 83] Abbott, Joe, "Presentation of computer I/O for people", Nation Computing Centre, England, 1983.

[ACME 85] Acme Widget Company, "Linear Optimizer, 2.1", The Acme Computer Company, 1985.

[Ball 80] Ball, E. and Hayes, P. "Representation of task-specific knowledge in a gracefully interacting user interface". The Proceedings AAAI-80, Stanford University, August, 1980.

[Bart 76] Barton, David C., and Cheryl Parks Francis, "A User's Cuide to MPS, A Linear Programming System from IBM", Department of Agricultural Economics, New York State College of Agriculture and Life Sciences, Cornell University, Ithaca, New York 14853.

[Benb 81] Benbasat, I., Dexter, A.S. & Masulis, P.S., "An experimental study of the human/computer interface." Communications of the ACM. 24(11), 752-762.

[Benb 84] Benbasat, Izak, and Wand Yair, "A structured approach to designing human-computer dialogues", Int. J. Man-Machine Studies, Vol. 21, 105-126, 1984.

[Bene 73] Beneke, R. R. and Ronald Winterboer, "Linear programming applications to Agriculture", The Iowa State University Press, Ames, 1973.

[Carr 82] Carroll, J.M., "The adventure of getting to know a computer", IEEE Computer, 49-58, November, 1982.

[CEIR 52] CEIR (1952), "A comprehensive Computing System for Linear Programming and Related Work on the IBM 7097 EDPM", Usage Manual, Washington D.C..

[Crof 84] Croft, W. Bruce, "The role of context and adaptation in user interfaces", Int. J. Man-Machine Studies, Vol.21, 283-292, 1984.

[Dant 51] Dantzig, Ceorge B., "Maximization of a Linear Function of Variables Subject to Linear Inequalities," in T.C. Koopmans(ed.), Adtivity Analysis of Production and Allocation. pp. 339-347, John Wiley & Sons, Inc., New York, 1951.

[Dant 63] Dantzig, Ceorge B., "Linear Programming and Extensions", Princeton University Press, Princeton N. J., 1963.

[Dorm 58] Dorfman, Robert, Paul A. Samuelson, and Robert M. Solow, "Linear Programming and Economic Analysis", McRaw-Hill Book Company, Inc. 1958.

[GAMS 88] GAMS, Anthony Brooke, David Kendrick, Alexander Meeraus,"GAMS, A User's Guide", The Scientific Press, 1988.

[ESP 86] Eastern Software Products, Inc., "Linear Programming of the IBM PC, version 5.12", 1986.

[Gain 81] Gains, B. R., "The technology of interaction-dialogue programming rules", Int. J. Man-Machione Studies, Vol.14, no.1, 130-150, 1981.

[Hadl 62] Hadley, G., " Linear Programming", Addison-Wesley publishing Company, Inc. 1962.

[Haye 80] Hayes, P., "Computers with natural communication Skill.", Computer Science Research Review 1979-80, Carnegie-Mellon University , Pittsburgh, Pennylvanis.

[Head 60] Heady, Earl O., and Wilfred Candler, "Linear Programming Methods," The Iowa State University Press, Ames, Iowa, 1960.

[Hill 74] Hillier, Frederick S., and Cerald J. Lieberman, "Operations Research", San Francisco, Holden-Day, 1974.

[IBM 67] IBM "Mathematical Programming System/360 (360A-CO-14x) Linear and Separable Programming - User's Manual", First Edition, 1967, IBM, White Plains, N.Y..

[IBM 78] IBM, MPSX/370 " Introduction to the Extended Control Language" Document Number SH19-1147-1, 1978,IBM, Endicot, New York.

[IBM 79] IBM, "MPSX/370 Primer, Document Number CH19-1091-1, 1979,IBM, Edicot, New York.

[IMSL 84] IMSL, "LP/PROTRAN, - A Problem Solving System for Linear Programming", 1984,IMSL,Inc., Houston, Texas.

[KETR 84] Ketron, "MPS III, Mathematical Programming System", User' Manual, Ketron, Inc. 1984, Arlinton, Virginia.

[Leon 41] Leontief, W. W., "The Structure of American Economy, 1919-1920", Harvard University Press, Cambridge, Mass, 1941. Second edition, Oxford University Press, New York, 1951.

[Lieb 84] Lieberman, H., "Seeing what your programs are doing", Int. J. Man-Machine Studies, Vol21, 311-331, 1984.

[Lotu 85] Lotus, "1-2-3 Reference Manual, Release 2", Lotus Development Corporation, 55 Cambridge Parkway, Cambridge, MA. 1985.

[Lust 85] Lustman, F., Mercier, P., & Gratton, L., "A dialog-based architecture for interactive information systems", Database ACM SIG Business Data Processing, Vol.16, no.3, 18-24, Spring 1985.

[Maye 81] Mayer, R.E., "The psychology of how novices learn computer programming", Computer Surveys, 13, 121-141, March, 1981.

[McCr 84] McCracken, D.L. & Akscyn, R.M., "Experience with the ZOG human-computer interface system", Int. J. Man-Machine Studies, Vol.21, 293-310, 1984.

[Mega 84] Megaw, E.D., edited, Contemporary Ergonomics 1984, "Man-computer dialogues for fixed-sequence data entry", by Hallam, J., & Stammers, R.B., Proceedings of the Ergonomics Society's Conference, April, 1984.

[Mulv 87] Mulvey, John M., and Stavros A. Zenios, "GENOS 1.0, A Generalized Network Optimization System", Report 87-12-03, Decision Science Department. The Wharton School, University of Pennsylvania, Philadelphia.

[Murt 87] Murtagh, Bruce A., and Michael A. Saunders, "MINOS 5.1 User's Guide", Report SOL 83-20R, December 1983, revised January 1987, Stanford University.

[Neum 28] Neumann, John Von, "Zur Theorie Der Gesellschaftsspiele," Mathematische Annale, 100:295-320, 1928.

[Neum 44] Neumann, John Von and Oskar Morgenstern, " Theory of Games and Princeton University Press, Princeton, N.J.,1944. Third edition, 1953.

[Nenu 45] Neumann, John Von, "A Model of General Economic Equilibriums", Rev. of Econ. Studies, 13(1), 1-9 (1945-46).

[Norm 83a] Norman, D.A., "Design principles for human-computer interfaces", Proceedings of the CHI 1983 Conference on Human Factors in Computing Systems, Boston, 1983.

[Norm 83b] Norman, D.A., "Design rules based on analyses of human error", Communications of the ACM, 26(4),245-258,1983.

[Norm 84] Norman, D.A., "Stages and levels in human-machine interaction", Int. J. Man-Machine Studies, Vol.21, 365-375, 1984.

[Pfei 83] Pfeiffer, G.H., "MPS-PC version 2.1", Research Corporation Research Software. 1983.

[Pres 82] Pressman, Roger S. " Software Engineering : A Practitioner's Approach", McGraw-Hill, Inc. 1982.

[Riss 84] Rissland, E.L., "Ingredients of intelligent user interfaces", Int. J. Man-Machine Studies, Vol.21, 377-388, 1984.

[Robe 81] Robertson, C., McCracken, D., & Newell, A. "Experimental Evaluation of the ZOG Frame Editor. Proceedings of the 7th Canadian Man-Computer Communications Conference, Waterloo, Ontaio, June, 1981, pp. 115-123.

[Scic 86] Scicon Ltd, "Scicon V/M User Cuide ver 1.40", Milton Keynes, England.

[Schn 83] Schneider, M. & Thomas, J.C., "Introduction:the humanization of computer interfaces", Communications of the ACM, 26(4), 252-253, 1983.

[Shne 79] Shneiderman, B., "Human factors experiments in Designing interactive systems", IEEE Computer, 9-19, December, 1979.

[Spra 80] Sprague, R.H., "A framework for the development of decision support systems", Management Information Systems quarterly, 4(4), 1-26, 1980.

[Tate 85] Tate, Brain, "Linear programming software", DBA Vintage Resources, Ltd., 1985.

[Thom 71] Thompson, Gerald E., "Linear Programming," New York, Macmillan, 1971.

APPENDIX A

THE USER-FRIENDLY-FRONTEND SYSTEM USER'S MANUAL

Table of Contents

LIST OF FIGURES

LIST OF TABLES

A.1 Introduction

USER-FRIENDLY-FRONTEND (or UFF) is a program that operates on the IBM or IBM compatible Personal Computer (either the XT or PC/AT model) running under MS-DOS 3.0 or higher. You should be familiar with DOS in order to use UFF.

UFF is not a stand-alone program; it is written in the proprietary Lotus 1-2-3 worksheet macro language to program the commands, keystrokes, and menus of the Lotus 1-2-3 to provide a friendly environment in which ESP users may input data. It automates the input procedures with many of its built-in functions. It saves the user-entered data to files and then converts the files to ESP files, and it print files on printer.

A.2 System Requirement.

The following hardware and software are required to use UFF:

(1) An IBM or IBM Personal Computer (XT or PC/AT model) a hard disk at least 640k of memory and must be run under MS-DOS 3.0 or higher. A Math coprocessor is recommended, it will reduce the processor times required.

(3) At least one double-sided double-density floppy disk drive.

(4) The IBM Personal Computer MS-DOS Operating System.

(5) An 80-column by 25-lines display monitor.

(6) Lotus 1-2-3 version 2.0 software loaded to the hard disk under the directory Lotus.

(7) Some double-side double-density floppy disks, formatted ready for file storage.

(8) A printer.

(9) The UFF (name AUTO123.WK1) program loaded to hard disk in the Lotus directory.

(10) Make sure your default drive is the hard disk, namely, the C drive.

A.3 Getting Started.

To start the system, user should follow these steps:

(1) Mark prepared LP matrix sheets every 200 columns; mark the first 200 columns as "page 1", mark the second 200 columns as "page 2", and so on.

(2) At DOS prompt, type UFF, the DOS will automatically load the Lotus 1-2-3 worksheet program, and automatically retrieve the UFF program called AUTO123.WK1.

(3) A blank worksheet (see Figure A.1) will appear on the screen, then user should press the "Alt" key and the letter "M" simultaneously to begin the macro program named M.

(8) You should now see on the screen display the UFF main menu as in Figure A.2.

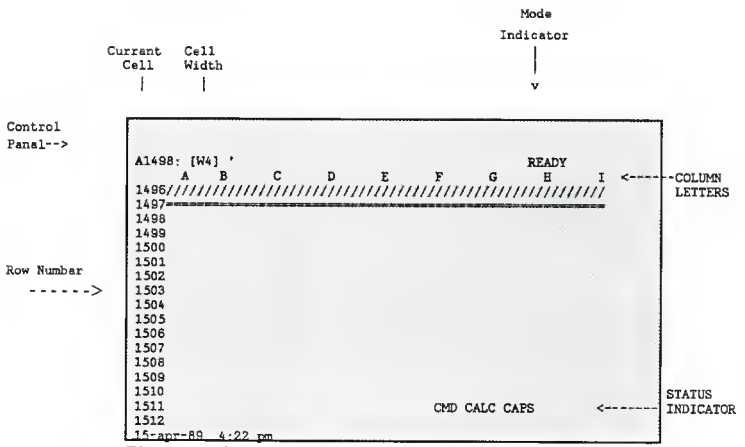


Figure A.1 : UFF User Worksheet Display.

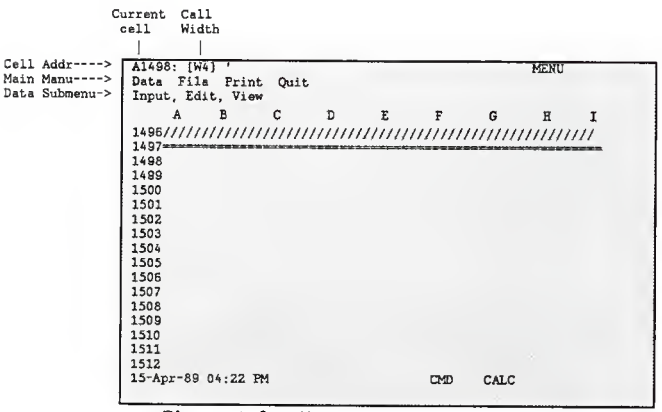


Figure A.2 : UFF Main Menu Display.

A.4 Know Your Worksheet.

To use the UFF program one needs not to be a skillful Lotus 1-2-3 user, but it will be helpful if you are acquainted with the following identifiers and their meanings [Lotu 85]:

Worksheet is a grid made up of horizontal rows(by row numbers) and vertical columns (by column letters). Each intersection of a row and a column forms a cell, in which one can store data.

Cell Address, each cell has a unique address that consists of its column letter and row number. For example, A1498 shown on the upper left corner of Figure 1 and Figure 2.

Cell Width is the column width measured by number of character space.

Cell Pointer is a rectangular highlight that appears on one cell in the worksheet (here can not depicted in Figure 1 or Figure 2).

Current Cell is the highlighted cell, entry will be stored in this current cell.

Row Number is the row identifier for each cell's horizontal position in the worksheet.

Column Letter is the column identifier for each cell's vertical position in the worksheet.

Control Panel is information panel. It contains three lines.

First Line provides information about the current cell including cell address, cell width and contents of the cell.

Second Line displays menu items and one of the items is highlighted, a mode indicator (see Table A.1) on the upper right corner of the worksheet (see Figure 1 and Figure 2). It also displays a program prompt tells what is the next step you should

take, or a request for information that the UFF program needs to carry out a selected command.

Third Line Displays either a submenu of the highlighted menu item or an one-line description of the highlighted menu item.

Mode Indicator tells the UFF program's current mode of operation. Table A.1 lists the mode indicators used in the program:

Table A.1 : UFF Program Mode Indicators.

Indicator	Meaning
READY	Ready for command or cell entry.
WAIT	Program is executing the instructions, calculating the index and cannot process commands.
MENU	Menu items are displayed in the control panel waiting for user's selection.
ERROR	Waiting for respond to the error.
EDIT	Edit a cell entry.
VALUE	Entering a number.
POINT	Pointing to a cell.

Status Indicators they appear on the bottom right corner of the worksheet (see Figure 1 or Figure 2). It tells the state of the UFF executing process is in and the key conditions.

Table A.2 : UFF Program Status Indicators.

Indicator	Meaning	User's Response
MD	UFF is pausing during an execution	None
CALC	UFF recalculates all its formulas	None
CAPS	The CAPS LOCK key is on	Must be kept on
NUM	The NUM LOCK key is on	Your choice
OVR	The INSERT key is on	Must be kept off
SCROLL	The SCROLL LOCK key is on	Must be kept off
END	The END key is on	Must be kept off

A.5 Basic Keystrokes.

The UFF program is a menu driven and user interactive program. There are only few basic keystrokes needed to use the program. Where a "keystroke" refers to a particular key on the computer keyboard that a specific function will be performed when it is stroked by the user. The basic keystrokes and their functions are listed in Table A.3:

Table A.3 : UFF Program Keystrokes.

Keystroke	Function
Alt and M	Pressed simultaneously to start the UFF program.
ENTER	Enters the user typed value or name into the current cell, or enters the menu item selection on the menu.
<-	Left arrow key to move the cell pointer to left to let the user to choose a menu item from the displayed menu item on the second line of the control panel of the worksheet.
->	Right arrow key to move the cell pointer to the right to let the user to select a menu item from the menu.
BACKSPACE	To back up one space.
ALPHABETS	Use as the typewriter keys to type row, rhs, and column names or row types.
NUMERICS	Use as the typewriter keys to type numbers.
S	To respond a prompt for saving a file.
P	To respond a prompt for printing a file.
Y	To confirm a prompt.
N	To deny a prompt.

A.6 Use the UFF Program Menu.

The menu is hierarchically structured and is four levels deep. To select the menu item, one can either highlight the cell which contains the menu item (use the arrow key to move the cell pointer) or type the first letter of the menu item and enter it.

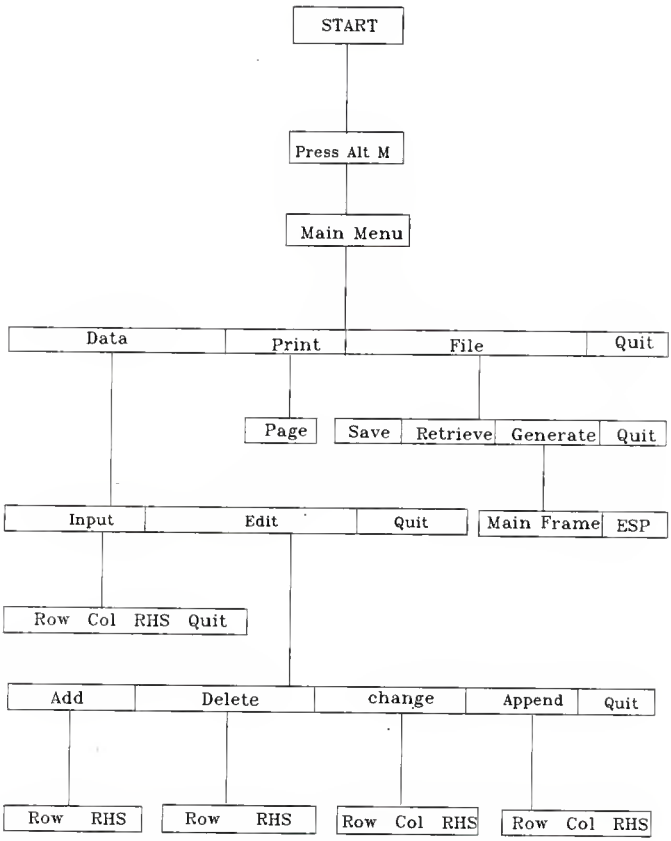


Figure A.3 : UFF Menu Structure

After the selected menu itemis entered, the UFF will bring up the submenu. If that item has a submenu, make a further selection etc. and finally the menu item that will perform the wanted task is encountered. For example to input row data, one needs to use the sequence: "DATA(enter)INPUT(enter)ROW(enter)" from main menu, data submenu, and input submenu respectively. For some menu items the system will display a one-line message requesting for related information needed to carry out the menu item's task selected. A response to the prompt is either to type in the information or to press the keystroke as instructed by the system.

The system allows traversal freely from a menu to its submenu by entering the menu item or from a submenu to its main menu by highlighting the "Quit" or type the letter "Q.

A.7 Input LP Data - A Tutorial.

After the main menu is displayed on the control panel of the worksheet and the mode indicator shows "READY"(see Figure A.2), begin to input the LP data into computer. First input the row data, followed by the rhs data, the column data is the last to be entered. The following is a step by step tutorial designed to illustrate how to input data:

Step 1 - Select "Data" from the Main menu :

Data File Print Quit

Step 2 - Select "Input" from the Data submenu:

```
Input  Edit  View  Quit
```

Step 3 - Select "Row" from the Input submenu:

```
Row  Col  V-RHS  Quit
```

Step 4 - To respond to the following prompt, type the total number of rows in your prepared LP matrix table:

```
Enter Total number of rows :
```

Step 5 - After entering the number of rows the UFF will generate the heading and the following requests will be displayed on the second line of the control panel of the worksheet one after another. It will guide your row data input and automatically move the cell pointer to the next cell after the user's information is entered and is stored into the current cell.

```
Enter row type (N,L,G,E) in col. B
```

```
Enter row name(1..8) in col. C
```

This step will be repeated until the total number of rows are reached. For every 20 rows, the program will stop to remind you to check your entries:

```
Do you want to make changes in this page? (Y,N)
```

Your response may be either Y or N. If it is Y, the program will ask you for the information about the cell address:

```
Enter cell address to be changed
```

The cell pointer will be moved to the cell according to the cell address that is entered and ask for the correct data:

```
Enter the correct value :
```

If the answer is N, the program will resume the row entry step until the last row is filled. A request to save the row file by using the keystroke "S" will be made.

```
Row entry completed, enter S to save:
```

After pressing the key S, the row entries will be saved under the name A:ROWS.PRN.

Step 6 - Go back to the Input submenu to select "V-RHS" :

```
Row  Col  V-RHS  Quit
```

Step 7 - Then follow the prompt and respond to the request displayed on the second line of the control panel:

```
Enter RHS value in col D :
```

The UFF program will automatically move the cell pointer down to the next cell after the information is stored into the current cell. This step will be repeated until the row number reaches the total number of rows,

the user will be reminded to save the RHS values :

```
RHS entry completed, press S to save:
```

Step 8 - Go back to the input submenu to choose "Col" from the menu items:

```
Row   RHS   Col   Quit
```

The program will ask to have the total number of columns entered:

```
Enter Total number of columns:
```

After entering the number of columns, the program will generate the column headings and prompt the entering of a column name followed by a coefficient value.

```
Enter column name (1..8) :
```

```
Enter value of coefficient
```

The program will automate the cell pointer movements to put your entry into the right column and row. For every 200 columns, it will prompt a line to remind you to check your entered values and allow you to edit them if your response to the request is Y.

```
Enter cell address to be changed:
```

```
Enter the correct value :
```

If the answer is N, the program will remind the user to save the page under the file name A:PAGEX.PRN, where **X** is the page number calculated by the UFF program according to the total number of columns which have entered; every 200 columns is a page.

APPENDIX B

THE UFF PROGRAM LISTING

\m	{ONERROR fix,fix}{GOTO}a1498~{menubranch mainmenu}			
mainmenu	Data Input,Edit,Quit {menubranch datamenu} {branch \m}	File Save, Retrieve, generate {menubranch filemenu} {branch \m}	Print print a page {print_sub} {branch \m}	Quit End-Friendly-Frontend {quit}
datamenu	Input Row, Column, V-RHS {menubranch inptmenu} {menubranch mainmenu}	Edit Row, Column, V-RHS {menubranch editmenu} {menubranch mainmenu}	Quit Return to mainmenu {menubranch mainmenu}	
inptmenu	Row Enter row type and name {rowentry} {menubranch datamenu}	Column Enter column name and values {colentry} {menubranch datamenu}	V-RHS Enter RHS name and value {rhsentry} {menubranch datamenu}	Quit Return to mainmenu {menubranch mainmenu}
editmenu	Row edit row data {edrow_set} {menubranch datamenu}	Column edit column data {edcol_set} {menubranch datamenu}	V-RHS edit RHS data {edrhs_set} {menubranch datamenu}	Quit Return to mainmenu {menubranch mainmenu}
filemenu	Save save this page {save_sub} {menubranch mainmenu}	Retrieve retrieve a page {retr_sub} {menubranch mainmenu}	Generate ESP, Mainframe {menubranch genmenu} {menubranch mainmenu}	Quit Return to mainmenu {menubranch mainmenu}
genmenu	ESP Generate ESP file {esp_sub} {menubranch mainmenu}	Mainframe Generate mainframe file {main_sub} {menubranch mainmenu}		

```
edrow_set     {GOTO}A1498~
              {menucall edrows}
              {menubranch editmenu}
              {return}


edcol_set     {GOTO}A1498~
              {menucall edcols}
              {menubranch editmenu}
              {return}


edrhs_set     {GOTO}A1498~
              {menucall edrhs}
              {menubranch editmenu}
              {return}

rowentry      {GETLABEL Is this a new file (Y,N)?,key20}
              {IF key20="Y"}{BRANCH new_file}
old_file      {OPEN "a:rows.PRN",m}
              {GOTO}A1499~{READLN A1499}
              {paneloff}{windowson}
              {BRANCH recount}
new_file      {paneloff}{windowsoff}
              /REa1498..gv2210~
              {goto}a1500~
              \~~/Ca1500~b1500..c1500~
              {goto}a1500~
              /wcs4~
              {goto}b1500~
              /wcs4~
```

```
                {goto}c1500~
                {windowson}
re_entry        {getnumber "Enter total number of rows:  ",trow}
                {windowsoff}{paneloff}
                /dfa1501..a1756~1~1~trow~
                {let rowind,@int(trow/20)+1}
                {let lcounter,0}
                {let wcounter,0}
                {LET ccounter,1501}
                {windowson}
windloop        {LET wcounter,wcounter+1}
                {if wcounter>1}{branch lineloop}
                {begin}
                                no.
                {right}
                               type
                {right}
                  name
                {down}{down}{left}
lineloop        {RESTART}{LET ccell,+"b"&@string(ccounter,0)}
                {goto}{ccell}~
                {recalc next1}
next1           {getlabel Enter row type(N,L,G,E) in col. B   ,$G$1499}
                {PANELOFF}{right}
                {recalc next2}{PANELON}
next2           {getlabel Enter row name(1..8) in col. C   ,$G$1499}
                {down}{left}
                {LET lcounter,lcounter+1}
                {LET ccounter,ccounter+1}
                {LET pcell,+"c"&@STRING(lcounter+1500,0)}
                {LET ccell,+"b"&@string(ccounter,0)}
                {LET acell,+"a"&@string(ccounter,0)}
continue        {IF lcounter>=trow}{BEEP 1}{branch ckdone}
```

```
                {IF lcounter<wcounter*100}{branch lineloop}
                {IF lcounter=wcounter*10+10}{ckpage}
next3           {if lcounter<trow}{branch lineloop}
                {if wcounter=rowind}{BEEP 2}{branch ckdone}
                {branch windloop}

ckpage          {BEEP}{getlabel Do you want to make changes in this pages?(Y,N) ,key}
                {if key="Y"}{menucall edrows}{ckpage}
                {if key="N"}{return}

ckdone          {getlabel Do you want to make final changes to row entry(Y,N)? ,key}
                {IF key="N"}{BEEP 1}{branch tosave}
                {IF key="Y"}{menucall edrows}{ckdone}

tosave          {GETLABEL "Row entry completed, enter S to save(a:rows.prn): ",key}
                {IF key=""}{branch tosave}
                {IF key="S"}{SAVE}

save            /PFrows{windowsoff}{paneloff}~RRa1500..{pcell}~AGQ
                {BEEP}
                {menubranch datamenu}

edrows          Edit_row                    insert_row                  Delete_row                  browse
                Correct a row entry         add a new row               delete an existing row      move page up,down,left, or right
                {corr_sub}                  {ins_sub}                   {del_sub}                   {menubranch browsemenu}
                {menubranch edrows}         {menubranch edrows}         {menubranch edrows}         {menubranch edrows}

caller          {RETURN}

corr_sub        {panelon}
                {getlabel "Enter cell address to be changed: ",var1}
                {goto}{var1}~
                {getlabel "Enter the correct input: ",temp1}
```

```
            /C{paneloff}temp1~~
            {RETURN}
ins_sub     {panelon}{getnumber "Enter row # to be inserted: ",var2)
            {LET var3,+"b"&@string(var2,0)}
            /wir{var3}~
            {LET trow,trow+1}{LET rowind,@int(trow/19)+1}
            /dfa1501..a1756~1~1~trow~
            {goto}{var3}~
            {getlabel "Enter new row type: ",temp2)
            /C{paneloff}temp2~~
            {right}
            {getlabel "Enter new row name: ",temp3)
            /C{paneloff}temp3~~
            {LET lcounter,lcounter+1}
            {let ccounter,ccounter+1}
            {LET ccell,+"b"&@string(ccounter,0)}
            {LET pcell,+"c"&@STRING(lcounter+1500,0)}
            {LET acell,+"a"&@string(ccounter,0)}
            {return}

Del_sub     {panelon}{GETNUMBER "Enter row # to be deleted: ",var4)
            {LET var5,+"a"&@STRING(var4,0)}
            /wdr{var5}~~
            {LET trow,trow-1}{LET rowind,@int(trow/19)+1}
            /dfa1501..a1756~1~1~trow~
            {LET lcounter,lcounter-1}
            {let ccounter,ccounter-1}
            {LET ccell,+"b"&@string(ccounter,0)}
            {LET pcell,+"c"&@STRING(lcounter+1500,0)}
            {LET acell,+"a"&@string(ccounter,0)}
            {return}

begin       {if wcounter=1}{page1}
```

```
                {return}
page1           {goto}a1499~
                {return}

msg             {beep 2}
                {GETLABEL "Row entry completed, press ESC/RETURN, use Edit to make changes ",key2}
                {IF key2=""}{menubranch datamenu}
                {MENUBRANCH datamenu}

windows         {GOTO}a1501~
                /WWH
                {window}
                {pgdn}{goto}{acell}~
                {return}

browsemenu      up                          down                        right                       left
                move a window up            move a window down          move a window right         move a window left
                {windowup}                  {windowdown}                {windowright}               {windowleft}
                {menubranch browsemenu}     {menubranch browsemenu}     {menubranch browsemenu}     {menubranch browsemenu}

windowup        {pgup}
                {return}

windowdown      {pgdn}
                {return}

windowright     {bigright}
                {return}

windowleft      {bigleft}
                {return}

recount         {LET lcounter,lcounter-1}
```

```
                {LET ccounter,ccounter-1}
                {LET pcell,+"c"&@STRING(lcounter+1500,0)}
                {LET ccell,+"b"&@string(ccounter,0)}
                {LET acell,+"a"&@string(ccounter,0)}
                {BRANCH continue}

key20           Y
trow                                        4
rowind                                      1
lcounter                                    4
wcounter                                    1
ccounter                                 1505
counter                                     4

pcell           c1504
                {return}
ccell           b1505
                {return}
acell           a1505
                {return}
key             S
                {return}
key2
                {return}
temp1           LAND
                {return}
temp2           L
                {return}
temp3           LD
                {return}
var1            C1502
                {return}
var2                                     1502
```

```
                {return}
var3            b1502
                {return}
var4                                        1503
                {return}
var5            a1503
                {return}

colentry        {GOTO}d1496~{row_freez}{window}{GOTO}e1499~
                {LET wpcounter,0}
                {LET colcounter,0}
                {LET wbegin,1498}
                {LET pagecounter,1}
                {GOTO}a1498~
                {GETNUMBER "Enter total number of columns : ",tcol}
                {IF tcol>2510}{LET maxcol,200}{branch compute}
                {LET maxcol,tcol}
compute         {LET wpage,@INT(tcol/maxcol)+1}
wpageloop       {LET wbegin,1498}/REe1498..gv2011~
                {LET colbegin,wpcounter*maxcol+1}
                {LET colmod,@MOD(tcol,maxcol)}
                {LET colend,colbegin+maxcol-1}
                {LET rowbegin,wbegin+3}
                {LET rowend,rowbegin+trow-1}
                {LET newpagecell,+"a"&@STRING(wbegin,0)}
                {GOTO}{newpagecell}~
                page
                {windowson}{panelon}
                {LET pagenum,pagecounter}
                {LET b1498,pagenum}{window}
                {LET newcolcell,+"e"&@STRING(wbegin,0)}
                {GOTO}{newcolcell}~
                /DF{setrange}~colbegin~1~colend~
```

```
                {LET head,rowbegin-2}
                {LET colhead,+"d"&@STRING(head,0)}
                {LET lastcell,colhead}
                {GOTO}{colhead}~
columnloop      {RESTART}{LET rowcounter,0}
                {GOTO}{lastcell}~
                {right}
                {LET nextcol,@CELLPOINTER("address")}
headloop        {IF @CELLPOINTER("row")=head}{branch header}
                {upone}{RESTART}{headloop}
header          {RECALC next4}
next4           {GETLABEL Enter column name(1..8) ,$G$1499}
                {down}
drawline          ------
nextrow         {LET rowcounter,rowcounter+1}
                {DOWN}
                {RECALC next5}
next5           {GETNUMBER Enter coeffient value  ,$G$1499}
                ~
                {LET lastcell,@CELLPOINTER("address")}
                {INDICATE WAIT}
                {IF rowcounter<trow}{branch nextrow}~
                {IF rowcounter=10*pagecounter}{BEEP 1}{ckcol}
next6           {LET colcounter,colcounter+1}
                {IF colcounter>=tcol}{BEEP}{branch donepage}
                {IF colcounter<maxcol}{columnloop}
                {LET pagecounter,pagecounter+1}
                {IF pagecounter<wpage}{branch wpageloop}
                {RETURN}

row_freez       /WTB

donepage        {GETLABEL Final changes to column entries in this page(Y,N)?,key3}
```

```
                {IF key3="N"}{BEEP 1}{branch save_sub}
                {menucall edcols}{donepage}

edcols          Edit                                Browse                              Quit
                namecol, coefficient                move page up,down,left,right        done column editing
                {menubranch corr#_sub}              {menubranch browsemenu}             {branch caller}
                {menubranch edcols}                 {menubranch edcols}

corr#_sub       namecol                             coefficient                         Quit
                change a colname                    change a coefficient                Quit corrections
                {colnm_sub}                         {coef_sub}                          {menubranch edcols}
                {menubranch corr#_sub}              {menubranch corr#_sub}

colnm_sub       {panelon}{GETLABEL "Enter cell address to be changed: ",var50}
                {GOTO}{var50}~
                {GETLABEL "Enter correct column name: ",temp50}
                /C{paneloff}temp50~~{return}

coef_sub        {panelon}{getlabel "Enter cell address to be changed: ",var6}
                {GOTO}{var6}~
                {getnumber "enter the correct value: ",temp6}
                /RV{paneloff}temp6~~{return}

upone           {up}

ckcol           {GETLABEL Do you want to make changes in this column ?(Y,N) ,key3}
                {IF key3="N"}{branch next6}
                {IF key3="Y"}{menucall edcols}{ckcol}
```

```
save_sub    {GETLABEL "Save this page , enter S to save: ",key)
            {IF key=""}{branch save_sub}
            {IF key="S"}{WINDOWSOFF}{PANELOFF}{defpage}

defpage     {IF pagenum=1}{branch spage1}{return}
            {IF pagenum=2}{branch spage2}{return}
            {IF pagenum=3}{branch spage3}{return}
            {IF pagenum=4}{branch spage4}{return}
            {IF pagenum=5}{branch spage5}{return}
            {IF pagenum=6}{branch spage6}{return}
            {IF pagenum=7}{branch spage7}{return}
            {IF pagenum=8}{branch spage8}{return}
            {IF pagenum=9}{branch spage9}{return}
            {IF pagenum=10}{branch spage10}{return}
            {IF pagenum=11}{branch spage11}{return}
            {IF pagenum=12}{branch spage12}{return}

spage1      {LET filename,"lppage1"}
            {branch savepage}

spage2      {LET filename,"lppage2"}
            {branch savepage}

spage3      {LET filename,"lppage3"}
            {branch savepage}

spage4      {LET filename,"lppage4"}
            {branch savepage}

spage5      {LET filename,"lppage5"}
            {branch savepage}

spage6      {LET filename,"lppage6"}
            {branch savepage}
```

```
spage7      {LET filename,"lppage7"}
            {branch savepage}

spage8      {LET filename,"lppage8"}
            {branch savepage}

spage9      {LET filename,"lppage9"}
            {branch savepage}

spage10     {LET filename,"lppage10"}
            {branch savepage}

spage11     {LET filename,"lppage11"}
            {branch savepage}

spage12     {LET filename,"lppage12"}
            {branch savepage}


savepage    /PF{filename}~RRa1498..{lastcell}~
            {BEEP}{RESTART}
            {menubranch mainmenu}

pagecounter                         1
tcol                                3
wpage                               2
wpcounter                           0
colcounter                          3
colbegin                            1
colend                              3
```

```
colmod                                  0
wbegin                                  1498
setrange        e1498..gc1498
rowbegin                                1501
rowend                                  1504
rowcounter                              4
nextcol         $G$1504
colhead         d1499
head                                    1499
newpagecell     a1498
                {return}
var6            G1502
temp6                                   1.00
key3            N
lastcell        $G$1504
pagenum         1
                {return}
newcolcell      e1498
                {return}
maxcol                                  3
                {return}

filename        lppage12
                {return}
var50           G1499
                {return}
temp50          OATS
                {return}

rhsentry        {windowsoff}{paneloff}
                {LET lrhscount,0}
                {LET rhscount,0}
                {LET wrhscount,0}
```

```
                {LET crhscount,1501}
wrhsloop        {LET wrhscount,wrhscount+1}
                {IF wrhscount>1}{branch lrhsloop}
                {windowson}{goto}d1499~
                {rhs1}~
                {down}
                  -------
                {down}
lrhsloop        {LET crhscell,+"d"&@string(crhscount,0)}
                {GOTO}{crhscell}~
                {windowson}{panelon}
                {RECALC getrhs}
getrhs          {GETNUMBER Enter RHS value  ,$G$1499}
                {DOWN}
                {LET lrhscount,lrhscount+1}
                {LET rhscount,rhscount+1}
                {LET crhscount,crhscount+1}
                {IF lrhscount>=trow}{branch ckrhsdone}
                {IF lrhscount<wrhscount*trow}{branch lrhsloop}
                {BEEP1}{CKRHSWIN}
                lrhscount<trow}{branch lrhsloop}
                {IF wrhscount=rowind}{BEEP 1}{ckrhsdone}
                {branch wrhsloop}

ckrhswin        {BEEP}{GETLABEL Do you want to make changes in RHS (Y,N),key4}
                {IF key4="Y"}{menucall edrhs}{ckrhswin}
                {IF key4="N"}{return}

ckrhsdone       {BEEP}{GETLABEL Do you want make final changes in RHS ? (Y,N),key5}
                {IF key5="N"}{branch rhssave_sub}
                {IF key5="Y"}{menucall edrhs}{ckrhsdone}
```

```
rhssave_sub   {BEEP}{GETLABEL  "Press S to save RHS column  :",key6}
              {IF key6=""}{branch rhssave_sub}
              {IF key6="S"}{save_rhs}

save_rhs      {paneloff}/PFrhs~RRC1501..D1601~AGQ
              {BEEP 1}
              {RETURN}

edrhs         Edit_RHS                          Quit
              correct a RHS entry               Quit RHS editing
              {corr_rhs}                        {branch caller}
              {menubranch edrhs}

corr_rhs      {panelon}
              {GETLABEL Enter cell address to be changed  ,rhsaddr}
              {goto}{rhsaddr}~
              /RFF2~{rhsaddr}~
              {getnumber "Enter the correct RHS value: ",newrhs}
              /C{paneloff}newrhs~~
              {RETURN}
wrhscount                                     1
crhscount                                  1505
rhsvalue                                 300.00
rhsaddr       D1502
newrhs                                    12.00
crhscell      d1504
rhscount                                      4
lrhscount                                     4
key4
key5          N
key6          S
rhs1          RHS1
```

```
retr_sub    {panelon}{windowson}
            {GETNUMBER Enter page number to be retrieved   ,pageno}
            {IF pageno<=15}{findpage}
            {BEEP 1}{outrange_msg}

findpage    {IF pageno=1}{branch rpage1}{return}
            {IF pageno=2}{branch rpage2}{return}
            {IF pageno=3}{branch rpage3}{return}
            {IF pageno=4}{branch rpage4}{return}
            {IF pageno=5}{branch rpage5}{return}
            {IF pageno=6}{branch rpage6}{return}
            {IF pageno=7}{branch rpage7}{return}
            {IF pageno=8}{branch rpage8}{return}
            {IF pageno=9}{branch rpage9}{return}
            {IF pageno=10}{branch rpage10}{return}
            {IF pageno=11}{branch rpage11}{return}
            {IF pageno=12}{branch rpage12}{return}

rpage1      {LET filename,"lppage1"}
            {branch retrpage}

rpage2      {LET filename,"lppage2"}
            {branch retrpage}

rpage3      {LET filename,"lppage3"}
            {branch retrpage}

rpage4      {LET filename,"lppage4"}
            {branch retrpage}

rpage5      {LET filename,"lppage5"}
            {branch retrpage}
```

```
rpage6          {LET filename,"lppage6"}
                {branch retrpage}

rpage7          {LET filename,"lppage7"}
                {branch retrpage}

rpage8          {LET filename,"lppage8"}
                {branch retrpage}

rpage9          {LET filename,"lppage9"}
                {branch retrpage}

rpage10         {LET filename,"lppage10"}
                {branch retrpage}

rpage11         {LET filename,"lppage11"}
                {branch retrpage}

rpage12         {LET filename,"lppage12"}
                {branch retrpage}


retrpage        {panelon}{windowson}{GOTO}A1495~
                /FIT{filename}~
                {BEEP}{RESTART}
                {menubranch mainmenu}

wait_msg


outrange_msg    {GETLABEL Out of page number's range - press retrun and try again ,key8}
                {IF key8=""}{paneloff}{windowsoff}{branch retr_sub}
```

```
pageno                          1
key8


esp_sub     {WINDOWSON}{PANELON}{de_freez}{RESTART}
            /REa2015..ca2215~{GOTO}C2015~
            {LET rowx,1}{LET rowy,1}
row_loop    {LET colx,0}{LET coly,2}{restart}
            {LET rowx,rowx+1}
            {LET rownum,2013+rowx}
            {LET rowaddr,+"C"&@STRING(rownum,0)}
            {IF rowx=2}{branch obj_loop}
            {IF rowx>2}{LET coly,2}
            {IF rowx>=trow+2}{BRANCH espfile}
rtrn_loop   {LET row_nm,@INDEX(B1499..gc1600,rowy,rowx):string}
            {LET coladdr,rowaddr}
col_loop    {IF coly>tcol+2}{BRANCH row_loop}
            {LET col_nm,@INDEX(B1499..gc1600,coly,colx):string}
coef_loop   {LET coefvalue,@INDEX(B1499..gc1600,coly,rowx):string}
            {GOTO}testvalue~{coefvalue}~{IF testvalue<>0.00}{BRANCH towrite}
            {LET coly,coly+1}{branch col_loop}
towrite     {GOTO}{coladdr}~{INDICATE WAIT}{PANELOFF}
            {row_nm}~
            {right}
            {comma}~
            {right}
            {col_nm}~
            {right}
            {comma}~
            {right}
            {coefvalue}~
            {LET coly,coly+1}
            {IF coly>tcol+2}{BRANCH row_loop}
```

```
                {right}{comma}~{right}
                {LET coladdr,@cellpointer("address")}
                {IF coly=tcol+2}{BRANCH slack_loop}
                {branch col_loop}
obj_loop        {LET coly,3}
                {branch rtrn_loop}
slack_loop      {LET type,@INDEX(B1499..gc1600,0,rowx)}
                {GOTO}typetest~{type}~
                {IF typetest="L"}{LET slackvalue,1:string}
                {IF typetest="G"}{LET slackvalue,-1:string}
                {IF typetest="E"}{LET slackvalue,0.00:string}
                {IF typetest="N"}{LET slackvalue,0.00:string}
                {GOTO}{coladdr}~
                {row_nm}~
                {right}
                {comma}~
                {right}
                {row_nm}~
                {right}
                {comma}~
                {right}
                {slackvalue}~
                {BRANCH row_loop}
espfile         {GOTO}C2015~{INDICATE}{WINDOWSON}{PANELON}
                {LET lastentry,coladdr}
                {BEEP}{GETLABEL "ESP file is generated, enter S to save ",keyesp}
                {IF keyesp=""}{BRANCH espfile}
                /RVc2015..{lastentry}~c2015..{lastentry}~
                /PFesp~RRc2015..{lastentry}~AGQ
                {BEEP}{menubranch mainmenu}
main_sub        {WINDOWSON}{PANELON}{DE_FREEZ}{RESTART}
                {LET rowend,+"C"&@string(trow+2016,0)}
                /REa2015..g2215~{GOTO}b2014~{DOWN}
```

```
            {headrow}~
            {DOWN}
            /Cb1501..c1601~C2016..{rowend}~
            {GOTO}{rowend}~
            {left}{headcol}~
            {LET it_end,@cellpointer("row")}
            {LET colx,0}{LET coly,1}
col_data    {LET coly,coly+1}{IF coly>tcol+1}{BRANCH rhs_data}
            {LET col_nm,@INDEX{c1499..gc2010,coly,colx):string}
            {LET rowx,2}{LET rowy,0}{RESTART}
row_data    {LET rownum,it_end+1}
            {LET rowaddr,+"C"&@string{rownum,0)}
            {LET row_nm,@INDEX{c1499..GC2010,rowy,rowx):string}
            {LET coefvalue,@INDEX{c1499..gc2010,coly,rowx):string}
            {GOTO}testvalue~{coefvalue}~
            {IF testvalue<>0.00}{BRANCH write}
            {LET rowx,rowx+1}{If rowx>trow+2}{branch col_data}
            {branch row_data}
write       {GOTO}{rowaddr}~
            {INDICATE WAIT}{PANELOFF}
            {col_nm}~
            {right}
            {row_nm}~
            {right}
            {coefvalue}~
            {LET it_end,@cellpointer("row")}
            {LET rowx,rowx+1}
            {BRANCH row_data}
rhs_data    {GOTO}{rowaddr}~{LEFT}
            {headrhs}~{LET it_end,@cellpointer("row")}
            {LET rowx,2}{LET rowy,0}
            {LET rhs_nm,@INDEX{c1499..gc2010,1,0):string}
loop        {LET rownum,it_end+1}
```

```
                {LET rowaddr,+"C"&@STRING(rownum,0)}
                {LET row_nm,@INDEX(c1499..gc2010,rowy,rowx):string}
                {LET coefvalue,@INDEX(c1499..gc2010,1,rowx):string}
                {GOTO}testvalue~{coefvalue}~
                {IF testvalue<>0.00}{BRANCH rhs_write}
                {LET rowx,rowx+1}{IF rowx>trow+1}{BRANCH mpsfile}{BRANCH loop}
rhs_write       {GOTO}{rowaddr}~
                {rhs_nm}~
                {right}
                {row_nm}~
                {right}
                {coefvalue}~
                {let it_end,@cellpointer("row")}
                {LET rowx,rowx+1}
                {IF rowx>trow+1}{branch mpsfile}
                {BRANCH loop}

mpsfile         {GOTO}A2015~{INDICATE}{WINDOWSON}{PANELON}
                {LET mainlast,+"E"&@string(it_end,0)}
                {GETLABEL "MPS file generated, enter S to save  ",keymps}
                {IF keymps=""}{BRANCH mpsfile}
                {IF keymps="S"}{branch savemps}
savemps         /RVb2015..{mainlast}~b2015..{mainlast}~
                /PFmps~RRb2015..{mainlast}~AGQ
                {menubranch mainmenu}

keymps          S
                {return}
mainlast        E2036
                {return}
headrow         ROWS
                {return}
headcol         COLUMNS
```

```
                (return)
headrhs         RHS
                (return)
rowend          C2020
                (return)
it_end                                  2036
                (return)
rhs_nm          @INDEX(c1499..gc2010,1,0)
                (return)
de_freez        /WTC
                (return)
colx                                       0
coly                                       2
rowx                                       6
rowy                                       1
rownum                                  2019
                (RETURN)
rowaddr         C2019
                (RETURN)
coladdr         $U$2018
                (RETURN)
comma           ,
                (RETURN)
row_nm          @INDEX(B1499..gc1600,rowy,rowx)
                (RETURN)
col_nm          @INDEX(B1499..gc1600,coly,colx)
                (RETURN)
coefvalue       @INDEX(B1499..gc1600,coly,rowx)
                (RETURN)
testvalue                                  24
                (return)
type            L
                (RETURN)
```

```
typetest    L
            {RETURN}
slackvalue  1
            {RETURN}
keyesp      S
            {return}
lastentry   $U$2018
            {return}

fix         File does not exist
            {BEEP}-Press Return, back to main menu{?}{ESC}
            {menubranch mainmenu}
            {return}
print_sub   {GETLABEL "enter pagenum to print: ",pagenum}
            {IF key=""}{branch print_sub}
            {INDICATE WAIT}{WINDOWSOFF}{PANELOFF}{locpage}
            {return}
locpage     {IF pagenum=1}{branch locpage1}{return}
            {IF pagenum=2}{branch locpage2}{return}
            {IF pagenum=3}{branch locpage3}{return}
            {IF pagenum=4}{branch locpage4}{return}
            {IF pagenum=5}{branch locpage5}{return}
            {IF pagenum=6}{branch locpage6}{return}
            {IF pagenum=7}{branch locpage7}{return}
            {IF pagenum=8}{branch locpage8}{return}
            {IF pagenum=9}{branch locpage9}{return}
            {IF pagenum=10}{branch locpage10}{return}
            {IF pagenum=11}{branch locpage11}{return}
            {IF pagenum=12}{branch locpage12}{return}

locpage1    {LET filename,"lppage1"}
            {branch prtpage}
```

```
locpage2      {LET filename,"lppage2"}
              {branch prtpage}

locpage3      {LET filename,"lppage3"}
              {branch prtpage}

locpage4      {LET filename,"lppage4"}
              {branch prtpage}

locpage5      {LET filename,"lppage5"}
              {branch prtpage}

locpage6      {LET filename,"lppage6"}
              {branch prtpage}

locpage7      {LET filename,"lppage7"}
              {branch prtpage}

locpage8      {LET filename,"lppage8"}
              {branch prtpage}

locpage9      {LET filename,"lppage9"}
              {branch prtpage}

locpage10     {LET filename,"lppage10"}
              {branch prtpage}

locpage11     {LET filename,"lppage11"}
              {branch prtpage}

locpage       {LET filename,"lppage12"}
              {branch prtpage}

prtpage       {panelon}{windowson}{GOTO}A1495~
              /FIT{filename}~
              {BEEP}{RESTART}
              /PPRa1498..m1600-AG
              {menubranch mainmenu}
```

APPENDIX C

SAMPLE OUTPUT LISTING OF THE UFF PROGRAM

Table of Contents

C.1 : User Entered LP Problem in Matrix Form
(Sample Problem)

pagepage no.	no.	type	name	RHS1	AC1	1 AC13	2 AC14	3 AC15	4 AC25	5 AC135	6
1	1	N	RETRUN	0.00	-105.39	-105.39	-105.39	-105.39	-105.39	-105.39	
2	2	L	LAND	1695.00	1.00	1.00	1.00	1.00	1.00	1.00	
3	3	L	DRYLANDD	213.00	0.00	0.00	0.00	0.00	0.00	0.00	
4	4	L	DRYLANDR	564.00	0.00	0.00	0.00	0.00	0.00	0.00	
5	5	L	LAND-FLD	262.00	1.00	1.00	1.00	1.00	1.00	1.00	
6	6	L	LAND-FLR	266.00	0.00	0.00	0.00	0.00	0.00	0.00	
7	7	L	LAND-CPD	130.00	0.00	0.00	0.00	0.00	0.00	0.00	
8	8	L	LAND-CPR	260.00	0.00	0.00	0.00	0.00	0.00	0.00	
9	9	L	LABMAR	186.00	0.40	0.40	0.40	0.40	0.40	0.40	
10	10	L	LABAPR	192.00	0.54	0.54	0.54	0.54	0.42	0.54	
11	11	L	LANMAY	202.00	0.23	0.23	0.23	0.23	0.23	0.23	
12	12	L	LABJUN	259.00	0.21	0.28	0.28	0.28	0.37	0.42	
13	13	L	LABJUL	259.00	0.00	0.35	0.35	0.35	0.35	0.35	
14	14	L	LABAUG	259.00	0.00	0.00	0.00	0.00	0.00	0.00	
15	15	L	LABSEP	255.00	0.00	0.00	0.00	0.00	0.00	0.00	
16	16	L	LABOCT	250.00	0.42	0.42	0.42	0.42	0.42	0.42	
17	17	L	LABNOV	225.00	0.52	0.52	0.52	0.52	0.52	0.52	
18	18	L	LABDEC	177.00	0.15	0.15	0.15	0.15	0.15	0.15	
19	19	L	PUMPLTDF	1159.00	0.00	0.00	0.00	0.00	0.00	0.00	
20	20	L	PUMPLTDC	941.00	0.00	0.00	0.00	0.00	0.00	0.00	

C.2 : UFF Generated ESP Data Statements
(Sample Problem)

```
 RETRUN , AC1 , -105.39 , RETRUN ,   AC13 , -105.39 , RETRUN ,  RETRUN ,   0.00
   LAND ,RHS1 , 1695.00 ,   LAND ,   AC1 ,   1.00 ,   LAND ,    AC13 ,   1.00 ,   LAND ,   LAND ,   1.00
DRYLANDO,RNS1 ,  213.00 ,
DRYLANDR,RHS1 ,  564.00 ,
LAND-FLO,RNS1 ,  262.00 ,LAND-FLO,   AC1 ,   1.00 ,LAND-FLO,   AC13 ,   1.00 ,LAND-FLO ,LAND-FLO ,   1.00
LAND-FLR,RHS1 ,  266.00 ,
LAND-CPO,RHS1 ,  130.00 ,
LAND-CPR,RHS1 ,  260.00 ,
 LABMAR ,RNS1 ,  186.00 , LABMAR ,   AC1 ,   0.40 , LABMAR ,   AC13 ,   0.40 ,  LABMAR ,  LABMAR ,   1.00
 LABAPR ,RNS1 ,  192.00 , LABAPR ,   AC1 ,   0.54 , LABAPR ,   AC13 ,   0.54 ,  LABAPR ,  LABAPR ,   1.00
 LANMAY ,RHS1 ,  202.00 , LANMAY ,   AC1 ,   0.23 , LANMAY ,   AC13 ,   0.23 ,  LANMAY ,  LANMAY ,   1.00
 LABJUN ,RNS1 ,  259.00 , LABJUN ,   AC1 ,   0.21 , LABJUN ,   AC13 ,   0.28 ,  LABJUN ,  LABJUN ,   1.00
 LABJUL ,RHS1 ,  259.00 , LABJUL ,  AC13 ,   0.35 , LABJUL ,  LABJUL ,   1.00
 LABAUG ,RHS1 ,  259.00 ,
 LABSEP ,RHS1 ,  255.00 ,
 LABOCT ,RHS1 ,  250.00 , LABOCT ,   AC1 ,   0.42 , LABOCT ,   AC13 ,   0.42 ,  LABOCT ,  LABOCT ,   1.00
 LABNOV ,RNS1 ,  225.00 , LABNOV ,   AC1 ,   0.52 , LABNOV ,   AC13 ,   0.52 ,  LABNOV ,  LABNOV ,   1.00
 LABDEC ,RHS1 ,  177.00 , LABDEC ,   AC1 ,   0.15 , LABDEC ,   AC13 ,   0.15 ,  LABDEC ,  LABDEC ,   1.00
PUMPLTDF,RNS1 , 1159.00 ,
PUMPLTDC,RHS1 ,  941.00 ,
```

C.3 : UFF Generated MPS Main Frame Data State
(Sample Problem)

```
ROWS
    N          RETRUN
    L          LAND
    L          DRYLANDD
    L          DRYLANDR
    L          LAND-FLD
    L          LAND-FLR
    L          LAND-CPO
    L          LAND-CPR
    L          LABMAR
    L          LABAPR
    L          LANMAY
    L          LABJUN
    L          LABJUL
    L          LABAUG
    L          LABSEP
    L          LABDCT
    L          LABNDV
    L          LABDEC
    L          PUMPLTDF
    L          PUMPLTDC
COLUMNS
    AC1        RETRUN      -105.3900
    AC1        LAND           1.0000
    AC1        LAND-FLD       1.0000
    AC1        LABMAR         0.4000
    AC1        LABAPR         0.5400
    AC1        LANMAY         0.2300
    AC1        LABJUN         0.2100
    AC1        LABOCT         0.4200
    AC1        LABNDV         0.52D0
    AC1        LABDEC         0.1500
    AC13       RETRUN      -105.3900
    AC13       LAND           1.000D
    AC13       LAND-FLD       1.0000
    AC13       LABMAR         0.4000
    AC13       LABAPR         0.5400
    AC13       LANMAY         0.2300
    AC13       LABJUN         0.28DD
    AC13       LABJUL         D.3500
    AC13       LABDCT         0.4200
    AC13       LABNOV         0.5200
    AC13       LABDEC         0.1500
```

(C.3 Continued)

AC14	RETRUN	-105.3900
AC14	LAND	1.0000
AC14	LAND-FLO	1.0000
AC14	LABMAR	0.4000
AC14	LABAPR	0.5400
AC14	LANMAY	0.2300
AC14	LABJUN	0.2800
AC14	LABJUL	0.3500
AC14	LABOCT	0.4200
AC14	LABNOV	0.5200
AC14	LABDEC	0.1500
AC15	RETRUN	-105.3900
AC15	LAND	1.0000
AC15	LAND-FLO	1.0000
AC15	LABMAR	0.4000
AC15	LABAPR	0.5400
AC15	LANMAY	0.2300
AC15	LABJUN	0.2800
AC15	LABJUL	0.3500
AC15	LABOCT	0.4200
AC15	LABNOV	0.5200
AC15	LABDEC	0.1500
AC25	RETRUN	-105.3900
AC25	LAND	1.0000
AC25	LAND-FLO	1.0000
AC25	LABMAR	0.4000
AC25	LABAPR	0.4200
AC25	LANMAY	0.2300
AC25	LABJUN	0.3700
AC25	LABJUL	0.3500
AC25	LABOCT	0.4200
AC25	LABNOV	0.5200
AC25	LABDEC	0.1500
AC135	RETRUN	-105.3900
AC135	LAND	1.0000
AC135	LAND-FLO	1.0000
AC135	LABMAR	0.4000
AC135	LABAPR	0.5400
AC135	LANMAY	0.2300
AC135	LABJUN	0.4200
AC135	LABJUL	0.3500
AC135	LABOCT	0.4200
AC135	LABNOV	0.5200
AC135	LABDEC	0.1500

APPENDIX D

THE LP MODEL LIBRARY

Table of Contents

D.1 List of Selected LP Models.

This section contains a list of LP models selected from the model library which are compiled by GAMS [GAMS 88]. The list shows the LP model names, number of lines, and references by area of applications.

AREA OF APPLICATION		NMUBER OF LINES	REFERENCE
Agricultural Economics			
Farm Credit and Income Distribution model		488	HUSA77
Pakistan Punjab Livestock Model		176	WB 77A
Organic fertilizer Use in Intensive Farming	398	WIEN85	
Egypt Agricultural Model		900	KUTC80
Turkey Agicultural Model		1132	LESI82
North-East Brazil Regional Agricultural Model		1049	KUTC81
Agricultural Farm Level Model of NE Brazil	310	KUTC81	
Indus Agricultural Model		1062	DUL084
Simple Farm Level Model		130	KUTC88
Economic Development			
Optimal Patterns of Growth and Aid		127	CHEN79B
DINAMICO, A Dymamic Multisectoral, Multiskill		849	MANN73
Energy Economics			
Investment Planning in the Oil-Petro Indst		707	MELT82
Single-Region Contingency Planning Model		199	MANN82
Turkey Power Planning Model		215	TURV77
Tobora Rural Development-Fuelwood Production		160	WB 77B
Strategic Petroleum Reserve		67	TEIS81
Management/Operations Research			
A Transportation Problem		66	DANT63
Aircraft Allocation Under Uncertain Demand		107	DANT63
APEX-Production Schduling Model		139	CDC 80
Enlementary Production and Inventory Model		57	FOUR83

D.2 List of References.

The following is a list of references cited by the CAMS LP MODEL LIBRARY.

[CDC 80] Control Data Corporation (1980), "Apex-III Reference Manual Vers. 1.2, PN 76070000, Minneapolis.

[CHEN79B] Chenery, H. B., and A. MacEwan (1979), "Optimal Pattern of Crowth and AID," in H. B. Chenery (ed.), Structural Change and Development Policy", Oxford University Press, New York and Oxford.

[DANT63] Dantzig, C.B., (1963), "Linear Programming and Extensions", Princeton University Press, Princeton, New Jersey.

[FOUR83] Fourer, R. (1983), "modeling Languages Versus Matrix Cenerators for Linear Programming", ACM Transaction of Mathematical Softwaree, Volume 9, Number 2.

[HUSA77] Husain, T., and R. Inman (1977), "A Model For Estimating the Effects of Credit Pricing on Farm Level Employment and Income Distribution", World Bank Staff Working Paper Number 261, The World Bank, Washington, D.C..

[KUTC80] Kutcher, G. (1980), "The Agro-economic Model", Technical Report Number 16, Master Plan for Water and Resources Development, UNDP-EGY/73/024, Cairo.

[KUTC81] Kutcher, F., and P. Scandizzo (1981), "The Agricultural Economy of Northeast Brazil", The Johns Hopkins University Press, Baltimore and London.

[KUTC88] Kutcher, C., A. Meeraus, and G.T. O'Mara (1988), "Modeling for Agricultural Policy and Project Analysis", The World bank, Washington, D.C.

[LESI82] Le-Si, V., P. Scandizzo, and H. Kasnagoklu, "Turkey Agricultural Sector Model", ACREP Working Paper Number 67, The World Bank, Washington, D.C.

[MANN73] Manne, A. S. (1973), "Dinamico, A Multsector, Multiskill Model", in L. M. Coreux and A. S. Manne (ed.s), Multilevel Planning: Case Stuidies in Mexico, North-Holland, Amsterdam.

[MANN82] Manne, A. S., C. R. Nelson, K. C> So, and J. P. Weyant (1982), "CPM: A Continency Planning Model of the International Oil Market", International Energy Report, Stanford University, Stanford.

[TEIS81] Teisberg, T. J. (1986), "A Dynamic Programming Model of the U. S. Strategic Petroleum Reserve", Bell Journal of Economics, Autumn 1981.

[TURV77] Turvey, R., and D. Anderson (1977), "Electricity Economics: Essays and Case Studies, Chapter 8", Johns Hopkins University Press, Baltimore and London.

[WB 77A] World Bank (1977), "Parkistan Punjab Livestock Project: Staff Project Report", Report Number 1193-PAK, Annexes I and II, Washington, D.C.

[WB 77B] World Bank (1977), "Tanzania: Appraisal of the Tabora Rural Development Project", Report Number 1360-TA, Annex 7, Washington, D>C>

[WIEN85] Wiens, T. B. (1985), "The Economics of High-Yield Agriculture in China:Triple-Cropping at the Baimano People's Commune", The World Bank, Washington, D.C.

APPENDIX E

MATHEMATICAL FORMS OF LINEAR PROGRAMMINC MODELS

Table of Contents

E.1 General Form of the LP Problem.

Expanded Notation

Maximize (or Minimize):

$$Z = c_1X_1 + c_2X_2 + \ldots + c_nX_n$$

Subject to:

$$a_{11}X_1 + a_{12}X_2 + \ldots + a_{1n}X_n \quad (\leq, =, \geq)\ b_1$$

$$a_{21}X_1 + a_{22}X_2 + \ldots + a_{2n}X_n \quad (\leq, =, \geq)\ b_2$$

$$\vdots$$

$$a_{m1}X_1 + a_{m2}X_2 + \ldots + a_{mn}X_n \quad (\leq, =, \geq)\ b_m$$

$$X_1 \geq 0,\ X_2 \geq 0,\ \ldots,\ X_n \geq 0$$

where c_j, b_i and a_{ij} (i=1, 2, ..., m; j=1, 2, ..., n) are constants and X_j are decision variables. The c_j, b_i and a_{ij} constants may be positive, negative or zero. A negative b_i is uncommon.

Summation Notation

Maximize (or Minimize):

$$Z = \sum_{j=1}^{n} c_j X_j$$

Subject to:

$$\sum_{j=1}^{n} a_{ij}X_j \quad (\leq, =, \geq)\ b_i,\ i=1, 2, \ldots, m$$

$$X_j \geq 0,\ j=1, 2, \ldots, n$$

E.2 Canonical Form of the LP Problem.

Summation Notation

Maximize:

$$Z = \sum_{j=1}^{n} c_j X_j$$

Subject to:

$$\sum_{j=1}^{n} a_{ij} X_j \leq b_i, \quad i=1, 2, \ldots, m$$

$$X_j \geq 0, \quad j=1, 2, \ldots, n$$

The characteristics that distinguish the "Canonical Form" from the "General Form" are:

1. All constraints are of the "$\leq$" type.
2. The objective function is of the maximization type.

E.3 Standard Form of the LP Problem.

Summation Notation

Maximize (or Minimize):

$$Z = \sum_{j=1}^{n} c_j X_j$$

Subject to:

m_1 "$\leq$" constraints $\sum_{j=1}^{n} a_{ij}X_j + S_i = b_i$, $i=1,2, \ldots, m_1$

m_2 "$\geq$" constraints $\sum_{j=1}^{n} a_{ij}X_j - S_i = b_i$, $i=m_1 + 1,..,m_1 + m_2$

m_3 "=" constraints $\sum_{j=1}^{n} a_{ij}X_j = b_i$, $i=m_1 + m_2 + 1,.., m$

$X_j \geq 0$, $j=1,2,\ldots,n$.

$b_i \geq 0$, $i=1,2,\ldots,m$.

where $m_1 + m_2 + m_3 = m$.

S_i is unrestricted, $i=1,2,\ldots, m_1 + m_2$.

The characteristics that distinguish the "Standard Form" from the "General Form" are:

1. All the original constraints are converted to equations (except the non-negativity constraints) by adding slack variables to "$\leq$" type constraints and by subtracting surplus variables from "$\geq$" type constraints.
2. The right-hand side constants, the b_i, are all non-negative.

The "Standard Form" has an important feature relating to the feasibility of basic solutions.

1. If the values of all slack and surplus variables (the S_i values) in a basic solution are non-negative the basic solution is feasible.
2. If the values of one or more S_i are negative the basic solution is infeasible.

E.4 Simplex Standard Form of the LP Problem.

Summation Notation

Maximize (or Minimize):

$$Z = \sum_{j=1}^{n} c_j X_j + \sum_{i=m_1+1}^{m} (M) A_i$$

Subject to:

m_1 "$\leq$" constraints $\sum_{j=1}^{n} a_{ij} X_j + S_i = b_i$, $i=1,2,\ldots, m_1$.

m_2 "$\geq$" constraints $\sum_{j=1}^{n} a_{ij} X_j - S_i + A_i = b_i$, $i=m_1+1, \ldots, m_1+m_2$.

m_3 "=" constraints $\sum_{j=1}^{n} a_{ij} X_j + A_i = b_i$, $i=m_1+m_2+1,\ldots, m$.

$X_j \geq 0$, $j=1,2,\ldots, n$.

$b_i \geq 0$, $i=1, ,\ldots, m$, where $m_1 + m_2 + m_3 = m$.

$S_i \geq 0$, $i=1,2,\ldots, m_1 + m_2$, where $m_1 + m_2 = m^*$.

$A_i \geq 0$, $i=m_1 + 1, \ldots, m$.

M is a large number.

The characteristics that distinguish the "Simplex Standard Form" from the "Standard Form" are:

1. The slack and surplus variables, S_i, are always non-negative.
2. Artificial variables, A_i, are added to the "$\geq$" and "=" type constraints to permit the use of the origin (all $X_j=0$) as the first basic solution and still have all S_i and A_i variables be non-negative.
3. The objective function includes the A_i variables with a large penalty of M. In a maximization problem M is negative. In a minimization problem M is positive.

AN USER FRIENDLY FRONTEND FOR A MPS PC PROGRAM

by

JONG-I PERNG

B.S., National Taiwan University, 1960
M.S., University of New Hampshire, 1963

AN ABSTRACT OF A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

ABSTRACT

Linear Programming (LP) is a very important analytical tool for agricultural economic research. The mathematical programming system (MPS) is an eminent computer programming procedure for solving a large linear programming problem. With the development of the simplex algorithm for the LP, the fast growing personal computer (PC) technology, and the advent of software packages for the MPS PC, a new class, learn-by-doing computer users has emerged.

Motivated by the fact that entering a large LP data into a computer is a time consuming and confusing task for the new user class, this report investigates the MPS PC user's desideratum and requirements for a computer program that will ease the LP data entry procedure. The design of the User-Friendly-Frontend program emphasizes the quality of the user-computer interface to provide a friendly environment for the MPS PC user to work in and to increase the data accuracy and integrity.

Four basic steps are taken in developing the program: preliminary design, detail design, coding, and testing. It is written in the LOTUS 1-2-3 macro language and is implemented on a Zenith 386 personal computer. A LP model, studies the impact of energy price on a NW Kansas farm is used to test the program.