LINEAR is an interactive linear programming package for the IBM PC/XT/AT family and compatibles. Its features include the following:

— Natural format for problem files.
— Extensive sensitivity analysis to answer "what-if" questions.
— Support of math coprocessor.
— On-line help.

Don't be misled by the word **programming** — this has nothing to do with **computer programming**. The "programming" in linear programming simply means "planning". I did the computer programming for you in producing this package. You don't need to know anything about computers to use it.

No mathematical training beyond high-school algebra is necessary to use LINEAR. However, a college-level course in linear programming, or at least some knowledge of linear algebra, would be helpful for the more advanced techniques. This manual is not a linear programming textbook. It and the software could be used, however, as a supplement to a linear programming or mathematical modelling course. For the mathematical theory of linear programming, see the following references:

V. Chvátal [1983]. *Linear Programming.* W.H. Freeman: New York/San Francisco

S.I. Gass [1975]. *Linear Programming.* McGraw-Hill: New York

G.B. Dantzig [1963]. *Linear Programming and Extensions.* Princeton U. Press: Princeton

This software was written by Robert B. Israel using Turbo Pascal. Please address any comments, questions or reports of bugs to him at

Department of Mathematics
University of British Columbia
Vancouver, BC, Canada V6T 1Y4

System Requirements

— IBM PC or compatible
— PC-DOS/MS-DOS 2.0 or above
— at least 128 KB RAM memory
— 80 column display
— at least one floppy disk drive (to read the distribution disk).

The following are recommended:

— Any full-screen editor or word processor program that produces ordinary text files.
— Math coprocessor chip.
— Second floppy disk drive, hard disk or RAMdisk.

Maximum problem size depends on available memory, so if you want to handle large problems you should get as much system memory as possible (up to 640K). For example, with 150,000 bytes of free memory (as indicated by `CHKDSK`), LINEAR can handle 86 constraints in 86 variables, while with 600,000 bytes free it can handle 283 constraints in 283 variables.

If your computer has a numeric coprocessor chip (8087, 80287 or 80387), or is a 486 (not 486SX), which has a coprocessor built in, LINEAR will automatically detect and use it. This makes solving problems much faster — sometimes over 10 times as fast.

Before you start

Before doing anything else, you should be sure to back up the distribution disk. This is, of course, a good idea for any piece of valuable software or data. The distribution disk is not copy-protected, and may be copied with the `COPY` or `DISKCOPY` commands. Keep the original in a safe place, and work with the copy.

The files you will need on your working disk are as follows:

— `LINEAR.EXE`.

— `LINEAR.HLP`.

— `LIN.BAT` and `LIN1.BAT` (if you want to use the '`Edit`' command).

You may also want to put `LINEAR.CFG` there. See the section **Configuring** LINEAR in Chapter 4.

If you have a hard disk, put these in the same directory. The distribution disk also contains files for several examples which will be referred to in this manual.

What is Linear Programming?

The best way to explain linear programming is with an example.

A manufacturer can produce three products: P_1, P_2 and P_3. The manufacturer would like to make as much profit as possible. However, production capacity is limited by the amount of processing time available on two machines M_1 and M_2, which are used in making the products.

Each unit of P_1 requires 1.5 hours on M_1 and 1 hour on M_2, and brings in \$20 of profit.

Each unit of P_2 requires 1 hour on M_1 and 2 hours on M_2, and brings in \$30 of profit.

Each unit of P_3 requires 1 hour on M_1 and 1 hour on M_2, and brings in \$15 of profit.

In a week there are 100 hours available on each machine.

How much of each product should be produced?

The problem can be formulated mathematically as follows. Let x_1,x_2 and x_3 be the amounts of the three products produced each week. These are the **decision variables** of the problem. Then the profit is given by the expression $20x_1 + 30x_2 + 15x_3$. This is called the **objective function**. We want to find values of x_1, x_2 and x_3 that will **maximize** the objective, subject to several **constraints**. To produce the amounts x_1, x_2 and x_3 of the products will require a total of $1.5x_1 + x_2 + x_3$ hours of M_1 time and $x_1 + 2x_2 + x_3$ hours of M_2 time. These amounts must not exceed the 100 hours available. This produces two constraints, $1.5x_1 + x_2 + x_3 \leq 100$ and $x_1 + 2x_2 + x_3 \leq 100$. There are three more constraints, which simply express the fact that we can't produce a negative amount of a product: $x_1 \geq 0$, $x_2 \geq 0$, and $x_3 \geq 0$. Here, then, is the mathematical statement of the problem:

$$\begin{gathered}
\text{maximize}\quad 20x_1 + 30x_2 + 15x_3 \\
\text{subject to} \\
\begin{aligned} 1.5x_1 + x_2 + x_3 &\leq 100 \\ x_1 + 2x_2 + x_3 &\leq 100 \end{aligned} \\
x_1 \geq 0, x_2 \geq 0, x_3 \geq 0
\end{gathered}$$

The objective function and constraints are all **linear**: they involve sums of terms, each of which can be a constant, a variable or a constant times a variable, but nothing else. For example, a variable times a variable is not allowed. This is why the subject is called "linear" programming. To solve problems involving objectives or constraints that are not linear, **nonlinear programming** must be used. Fortunately, many problems of practical interest involve only linear objectives and constraints.

Our example involves maximizing the objective, but in other problems you may want to **minimize** an objective (perhaps representing cost) instead. The constraints in the example all involved $\leq$ or $\geq$ signs, but $=$ signs may also occur. For example, you might want to insist that all available time on machine M_1 be used up: the first constraint would become $1.5x_1 + x_2 + x_3 = 100$.

A linear programming problem, then, is one in which a linear objective function of several variables must be maximized or minimized subject to a number of linear constraints.

Solving the Problem

Before a problem can be solved, it must be put into a form that LINEAR can read. I have already done this for our example in the file `EXAMPLE1.PRB`. Here are the contents of that file:

```
constraints 2;
variables 3 x1 x2 x3;
maximize
20 x1 + 30 x2 + 15 x3 [ profit ];
1.5 x1 + x2 + x3 <= 100 [ m1 ];
x1 + 2 x2 + x3 <= 100 [ m2 ];
end
```

This is quite similar to our mathematical statement of the problem. Note, however, that the constraints $x_i \geq 0$ are not included: LINEAR automatically assumes that all variables must be ≥ 0.

The problem file can be produced with any text editor or word processor that produces ordinary text files. Alternatively, you can enter the problem directly from the keyboard.

- Turn the machine on.
- Put your working disk in the drive (or on a hard disk system, go to the directory containing the LINEAR files).
- Enter the command '`linear`'.

NOTE: *In this manual, to "enter" something means that you type it, without quotation marks of course, and press* $\boxed{\hookleftarrow}$.

You should see a menu at the bottom of the screen:

```
Load       Edit       Record       Path       Options       Quit
```

The word '**L**oad' should be in reverse video (black on white), and the capital letters should be highlighted. You can use the ← and → cursor control keys (on the numeric keypad in most keyboards) to move the reverse- video bar to any of the five choices. If it doesn't move, you probably have Num Lock on: pressing the Num Lock key once should correct the situation. To select a choice in a menu, either press the key for its capital letter, or move the bar to it and press ↵ , (the large key called 'Return' or 'Enter').

- Choose '**L**oad'.

  Now you should see the following menu:

```
load                Problem                Tab                Dif
```

  LINEAR is asking for the type of input file.

- Choose '**P**roblem'. Next you are asked for the name of the input file.
- To use the prepared example, enter 'example1'.
- If you prefer to type in the problem yourself, enter 'con:', and then enter each line of the problem.

  After the last line, LINEAR should signal 'Loaded' and the name of the file. You are now ready to solve the problem.

- Choose '**S**olve'.

  The output will look like this:

```
Solving problem
Pivot #      Entering             Leaving              PROFIT
Phase 2 begins
    1          X2            SLACK M2               1500.0000
    2          X1            SLACK M1               1750.0000
Optimal solution found after  2 iterations
Optimal value of PROFIT:     1750.0000
```

  The best solution, as found by LINEAR, yields a profit of $1750 per week.

- To see what values for the variables produce this amount of profit, choose '**V**alues **V**ariables'.

  The output is as follows:

```
Values of basic variables
      PROFIT      =    1750.0000
      X1          =      50.0000
      X2          =      25.0000
```

X3 is not mentioned, because it is a **nonbasic variable**: this means that its value in the solution is 0. Thus the best plan for the manufacturer is to produce 50 units of P_1, 25 units of P_2 and no P_3 each week.

There is a good deal of additional information that LINEAR can tell about the problem, as will be explained in later chapters. But for now, you can exit from LINEAR and return to DOS.

- Choose '`Quit Exit`'.

Trademark Acknowledgements