A USER GUIDE TO INTERACTIVE (APL/360) PROGRAMS FOR OPERATIONS RESEARCH

by

MASOUD SHAMS AHMADI

B.S.I.E. Kansas State University
Manhattan, Kansas 1974

A MASTER'S REPORT

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

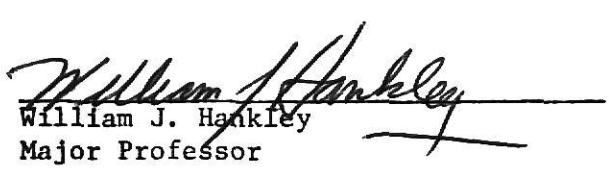
Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1976

Approved by:



William J. Hankley
Major Professor

ACKNOWLEDGMENTS

This author wishes to thank Dr. William J. Hankley for his counsel and encouragement through the preparation of this report. I would also like to thank my supervisory committee members Dr. Richard F. Sincovec and Dr. James Gentry.

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING BEING SKEWED DIFFERANTLY FROM THE TOP OF THE PAGE TO THE BOTTOM.

THIS IS AS RECEIVED FROM THE CUSTOMER.

TABLE OF CONTENTS

GENERAL INTRODUCTION

The purpose of this report is to provide a user documentation for the interactive use of a selection of operations research programs available under APL at Kansas State University. There is some documentation for these programs [5] but it is generally not sufficient and clear enough for the user with a minimum computer and APL background. Therefore the intention of this report is to provide a documentation which is easy to read, aimed at a low level, and which includes solved examples for each program.

Many students in introductory operations research and business courses need to use a program which is easy, accessible and gives them a quick result in solving O.R. problems. The available programs under APL are very useful for tutorial purposes and solving small problems where obtaining a fast solution is very important to a user. In contrast, the batch programs are very inconvenient to use. For instance, batch programs generally require considerable "JCL" header information and there is usually a significant delay for receiving the output. However, it must be noted that batch programs are widely used for solving large problems where they cannot be solved by APL programs.

This report consists of five chapters. Chapter I gives an introduction to the APL and time sharing systems. A brief introduction to each technique, examples, comparison and evaluation of each program is given in chapter II through V of the report. Chapter II includes the discussion of the linear programming problem, assignment problem is

given in chapter III, chapter IV discusses the transportation problem and chapter V is about project scheduling. Chapter II through V of the report are each divided into three sections. Section I contains a brief description of each operations research technique with an example. Section II compares and evaluates each program. Section III gives several examples for each program.

CHAPTER I

INTRODUCTION TO TIME SHARING AND APL

APL is a programming language system designed for interactive time sharing use. To work with APL, the user sits at a tele-typewriter terminal, typically a "selectric" typewriter with some electronics "memory" and some telephone electronics inside the console. The user types instruction and data. Each time the carriage return is struck, the APL processor in the remote computer executes the instruction or processes the input data. Computer responses and results are typed "out" on the typewriter terminal. Also, it may happen that user causes the computer to start typing a very long result, or working on a very lengthy (or even interminable) calculation. If the user decides that he wants to cut short what the computer is doing, the easiest way is to use the "interrupt" feature supplied with the typewriters. In an IBM 2741 terminal, pressing the ATTN key while the keyboard is locked will bring whatever the computer is doing to halt.

Now, let us define a couple of important terms for better understanding time sharing systems. As it has been defined, APL/360 is a time sharing system with remote terminals. Each one of these terms will be defined in turn.

"Remote terminal" means that the user does not have to come to the computer in order to use it. Instead, he uses a typewriter that is connected to a computer. The typewriter is usually referred to as a

"terminal" and it is "remote" because the connection is either by the public or private telephone line, so that the typewriter can be located anywhere that a telephone line can reach.

"Time Sharing" means that the central computer is serving many customers at once. Actually it serves them in rotation, each one gets a tiny ("slice") fraction of the computer's time, but (hopefully) the computer's operating speed is so high that often (but not always) there is no appreciable delay between the time the user types his request and the time the computer types its response. Even for problems of moderate size a response may be received within a few seconds.

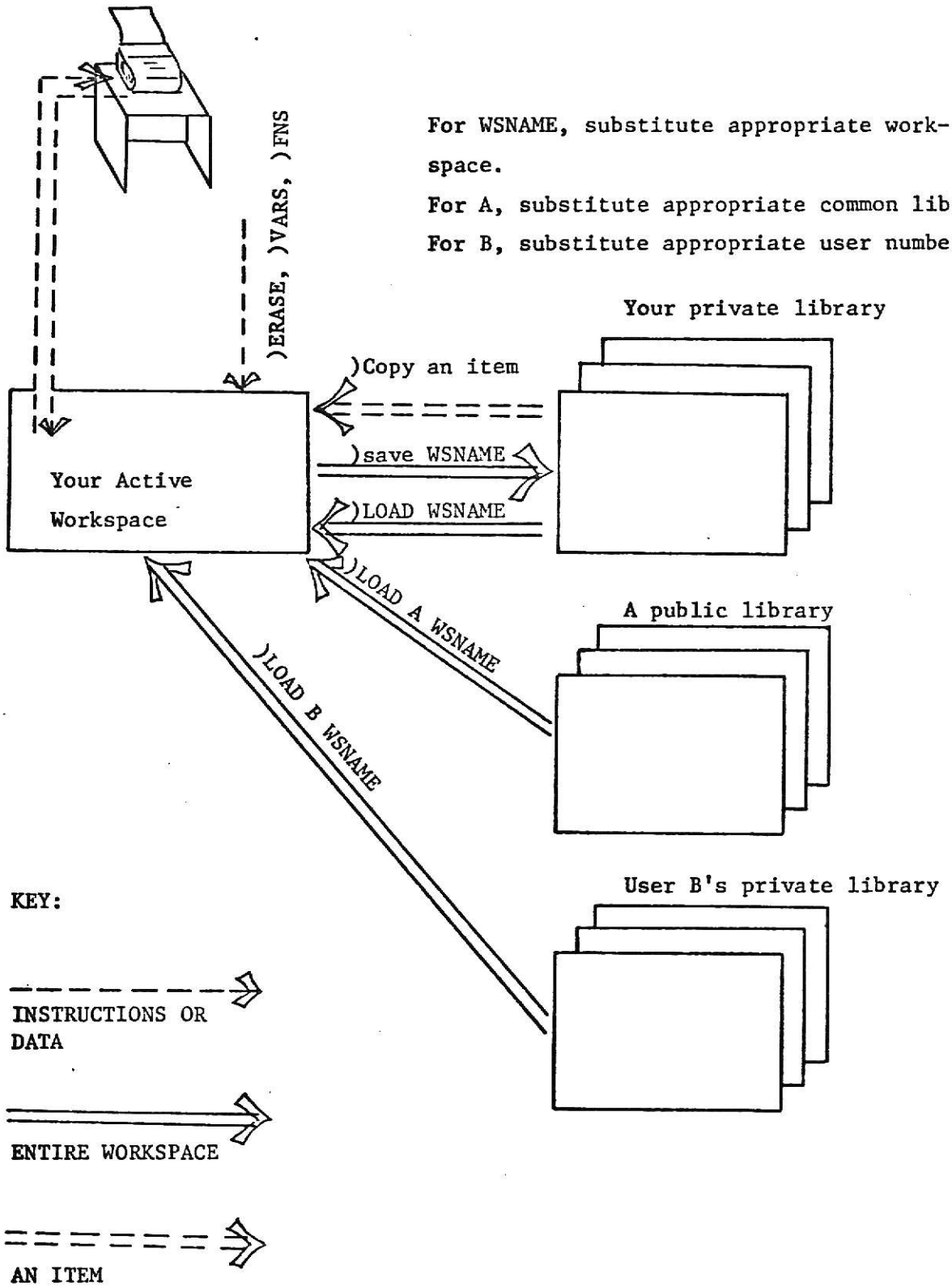
The letters APL designate the programming language that is the outgrowth of the work of K. E. Iverson, first at Harvard and then at IBM. The name comes from the initials of his book A Programming Language [1]. Since the invention of APL/360 there has been several versions of APL (in particular, scientific Time Sharing Corporation's APL*PLUS Time Sharing service and IBM's APLSV program produce). The new versions of APL have character and file processing features which are mostly used in business and other real life applications.

APL is one of the most powerful programming languages for solving mathematical problems. Operations on single items (scalars) extend in a simple and natural way to arrays of any size and shape. Thus, for instance, a matrix addition that in other languages might require two loops and a half dozen statements, become simply A+B in APL. The simplification offered by APL's rich and powerful handling of arrays is central to its strength.

The following diagram represents the flow of information between the user and the computer. It summarizes the following points:

1. The user can see or use only progrmms or data that are in his currently active workspace ()FNS,)VARS).
2. The user can save only what is in his currently active workspace.
3. The user can save only into his own library.
4. The user can load into his active area from his own library, from a common library, or (provided that the user has the necessary information) from the library of another user.
5. If the user gives the command to copy an entire saved workspace, the computer reads into his active workspace all of the programs from whatever saved workspace he calls.
6. The user can copy an item.
7. Global variables, functions or groups may be erased from the workspace by the command)ERASE followed by all the object to be erased.
8. Command)CLEAR clears the active workspace.
9. Command)DROP WSNAME removes the workspace from the user library. e.g. to erase an item from the library.

```
)SAVE A
)LOAD B
)ERASE ITEM
)SAVE B
)LOAD A
```



For WSNAME, substitute appropriate workspace.

For A, substitute appropriate common library.

For B, substitute appropriate user number.

I.A. SIGN ON AND SIGN OFF PROCEDURES:

APL BALL: Make sure that the APL ball (ball number 988) is on before the start of the sign on procedures.

SIGN ON: After dialing in or switching on the hard-wired terminals, hit the ATTN key. The VM/370 ONLINE message should be received. Then type in the following:

DIAL APL

The system will respond with a garbage message and unlock the keyboard. The)INIT command must be issued at this time for an account number which has not been previously used on APL. Prompting is provided to complete the initialization of the account number under APL. An example is given below:

```
VM/370 ONLINE     LJH359 QSYOSU

DIAL APL
⍳⍺⍞∈⊥ ~○ ⍺*⍞        022
)INIT
002) 14.19.49 06/04/76 ΔΔΔ 106171

ENTER (1) )KSUACCT AAAAAAAA III :PPPPPPPP
      (2) )OFF OR )OFF HOLD
      (3) )AAAAAAAA SSN :APLLOCK :PPPPPPPP

WHERE AAAAAAAA IS YOUR ACCOUNT NUMBER, III ARE YOUR INITIALS,
:PPPPPPPP IS YOUR ACCOUNT PASSWORD, SSN IS YOUR SYSTEM SECURITY
NUMBER, AND :APLLOCK IS YOUR APL LOCK.

      )KSUACCT LP09COM5 MSA :C
      )OFF HOLD
002  14.21.06 06/04/76 ΔΔΔ
CONNECTED     0.01.17  TO DATE  602.08.57
CPU TIME      0.00.00  TO DATE    0.24.36
TOTAL COST      $0.07  BALANCE  $8443.26
```

Otherwise, enter:

)MASK

The system will respond with a mask for the sign-on line. Then enter:

)PASSLETTER Account-Number Social-Security-Number

Where each term is defined below:

Passletter: Is an optional character used for security purposes.

Account Number: Is a number which indicates the user can use the computer.

Social Security Number: Is user social security number.

Note: Passletter and social security number must be first registered with the computing center at Kansas State University.

Example:

)ADP69C4F5 577789640

A: Is a passletter (optional)

DP69C4F5: Is an account number.

577789640: Is the user social security number.

After this step, the system will respond with an APL/360 message indicating that sign-on is completed. The following is an example of sign-on procedure for an account number which has already been initialized. Throughout the report, the instructions by the user are shaded.

```
VM/370 ONLINE      LJH359 QSYOSU

DIAL APL
|⍳⍺⌐∈| ~○ ⍺*⌐        021
)MASK
)PASSLETTER 577789640
001) 11.53.22 04/13/76 MSA 103152

   A P L \ 3 6 0
```

The number 103152 is the user workspace identification number. The number could be used by any other user to access this user workspace.

SIGN OFF: When the user has finished working with terminal, he signs-off by entering the command:

) OFF

An example of sign-off procedure is shown below. Accounting information is also provided at the end.

```
      )OFF
001  14.04.29 01/27/76 MSA
CONNECTED      0.03.01  TO DATE      3.40.52
CPU TIME       0.00.01  TO DATE      0.06.43
TOTAL COST       $0.23  BALANCE     $198.16
⌊⍴⍤ _⍴⍤| ⍺⍤□       021
]
```

Important notes about APL:

1. A negative number is indicated by the symbol that means "negative" placed in front of the number.

   Example: ¯2.

2. The input quad ['□'] is very handy for copying with numeric entry that overruns the right margin. A line of numeric input that ends with a quad has the meaning "to be continued." This feature can be used for entering large arrays. An example is given in next section (I.B.).

3. If the user makes a typing error while entering the data, he should back space to where the mistake is and then press the ATTN key and retype from where the mistake has occurred. An example is given below:

```
      A←1 2 3 4 5 6 7
                v
                10 11 12
      A
1  2  3  4  10  11  12
```

4. The user can change the contents of an array or a vector by using the text-editing feature of APL. In many occasions, after the execution of a program, the user finds out that he made a typing error. In this case rather than retyping the whole array, he could just change an element or elements of the array and re-execute the program. An example is given below:

```
      CS←1 2 3 4 5 6
               ∨
               5 8 8
      CS
1  2  3  5  8  8
```

```
      CS[3]←100
      CS
1  2  100  5  8  8
```

```
   CS←3 4 ⍴ 1 2 3 4 5 6 7 8 9 10 11 12
   CS[1;2]←120
   CS[2;4]←200
   CS
1  120    3    4
5    6    7  200
9   10   11   12
```

5. In order to display the contents of a variable or an array, the user must type the variable or array name and press the RETURN key.

6. Suppose a user is through with some programs or variables altogether. He may keep them in his workspace indefinitely if he wishes. But if he no longer wants them cluttering in his workspace, he may delete them by using the system

command)ERASE followed by the names of each of the programs (or variables) he wants to erase.

7. The state indicator shows which programs are suspended by typing an asterisk after their names. The ones without asterisks are pendent.

Example:

```
     )SI
AREA[1]*
WORK[2]*
REPEAT[7]
```

The programs called WORK and AREA are suspended, but the program called REPEAT is pendent. The state indicator may be cleared back to the last previous suspended function by entering a single right pointing arrow with no value to the right of it, like this " →".

A Guide to Error Recovery

The following table is a guide to error recovery. It contains, source of error, corrective action and a page reference in the report for further consultant.

REPORT	PROBLEM	CORRECTIVE ACTION	PAGE REFERENCE
Incorrect Sign On	Use of incorrect command	Enter correctly	7
Number Not In System	Either the number is not in the system or the number has a lock and wrong key was used		7
Incorrect Social Security Or Account Number	Use of incorrect social security or account number	Enter correctly	8
Object Not Found	Workspace does not contain the object; a variable function; or group		-
WS Full	Workspace overloaded	Delete objects no longer needed	15
Shut Down Initiated	System is going down	No problem, everything is saved and will be in user workspace after he signs-on again	-

I.B. GENERAL DISCUSSION OF PROGRAM COSTS AND MEMORY REQUIREMENTS FOR SOLVING O.R. PROBLEMS:

This section is devoted to a discussion of costs and memory requirements for solving O.R. problems under APL.

As soon as the user sign-on is completed, the computer puts at his disposal a block of its internal storage (or "memory"). This block of storage is called the user workspace, since it is where all of his calculations take place. In it will be stored the definitions of programs that the user enters, and the names and values of variables used in his calcuations. An APL system may have several public libraries in which have been stored workspaces containing programs or data that may be useful to many users. Anyone may load one of these public libraries and thus acquire a wide variety of special programs (e.g. O.R. programs) that are already written, tested and ready for use.

When signing-on an IBM 2741 terminal at Kansas State University the user will be provided with 31868 bytes of memory. Allocation of the user workspace in this case is shown below:

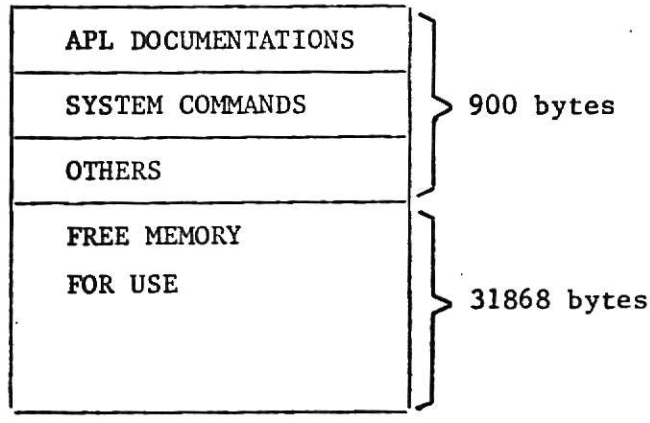


Fig. 1. APL/360 Workspace Organization After Sign-On.

On the other hand, copying a function like SIMPLEX into a clear user workspace will put 31232 bytes at his disposal. The workspace organization is shown in Figure 2.

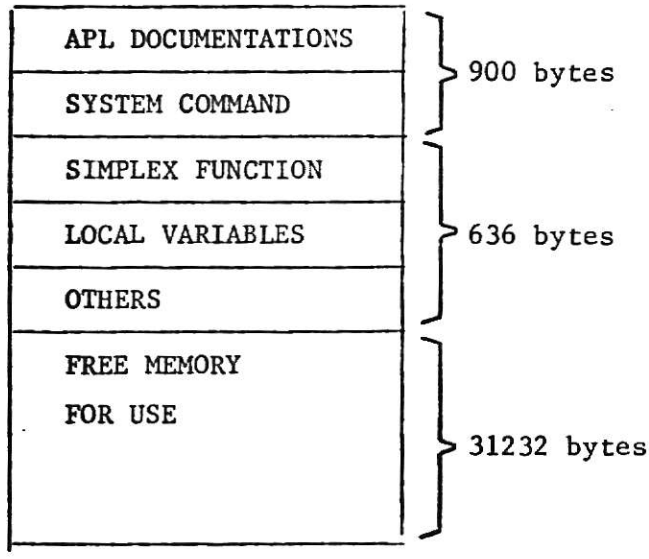


Fig. 2. AP./360 Workspace Organization After Loading SIMPLEX.

When loading a public library like STP4 into a user workspace, the workspace organization is shown in Figure 3.

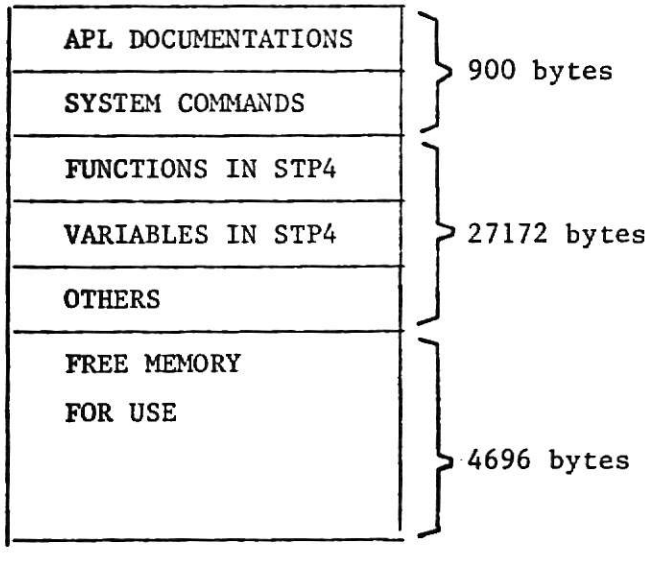


Fig. 3. APL/360 Workspace Organization After Loading STP4.

In many cases the user will receive an error message of WSFULL. This occurs when he has entered an instruction which requires more storage in user workspace than is now available. This may arise because he has assigned a variable a value that involves a large array, or because some of the intermediate steps in his calculation require too much space even for temporary storage during the calculation. Since most of the O.R. programs make use of large arrays, it is very easy to get this error message.

By looking at Figure 1 through 3 and being aware of the above facts, then a user can run bigger problems than those which will be specified in chapters II through V of this report by just copying the required functions into his active workspace. For example, if a user decides to solve a linear programming problem with 25 variables and 15 constraints, he should just copy the function SIMPLEX rather than load the whole public library (STP4) into his workspace and then execute the program. Because loading the whole public library uses a lot of memory and the user gets an error message of WSFULL at the execution time. The list of the required function for executing a desired O.R. program is given in Table 1.

In order to use any one of the programs in Table 1 follow the steps below:

1. Sign-On.
2. Issue Command)CLEAR.
3. Issue)COPY 2 WSNAME FUNCTION

For example, if we want to use the RSIM1 program, the following steps should be followed:

1. Sign-On.
2.)CLEAR.

TABLE 1

LIST OF REQUIRED FUNCTIONS FOR EXECUTION OF EACH O.R. PROGRAM

Program	Workspace Name	Required Functions	Used in Solving
SIMPLEX	STP4	SIMPLEX	L.P. Problem
RSIM	STP4	RSIM	L.P. Problem
RSIM1	STP4	RSIM,RSIM1	L.P. Problem
LINPR	STP6	LIST1,LIST2,LINPR,INPUT, ASSEMBLE,CONVERT,OUTPUT,RSIM	L.P. Problem
LPSOLN	STP4	LPSOLN,INV	L.P. Sensitivity Analysis
TRANSPORT	STP4	TRANSPORT, NETFLOW	Transportation Problem
ASSIGNMENT	STP4	ASSIGN	Assignment Problem
CPM	STP4	CPT	Critical Path Method
CPM1	STP5	CPM1,INPUT,BASICDATA, NODEVECTOR,NUMBEROFNODES, DURATIONVECTOR,PRECEDENCEMATRIX, NETWORKCHECK,NONENUMBERING, INITIALNODES,TERMINALNODES, CONSISTENCY,ERRORREPORT, CPMALGORITHM,EARLYSTART, EARLYFINISH,LATESTART,LATEFINISH, TOTALSLACK,CRITICALPATH, FREESLACK,PREOUTPUTSORT, OUTPUT,TOPOLOGICALSORT,	Critical Path Method

3.)COPY 2 STP4 RSIM

4.)COPY 2 STP4 RSIM1

5. Follow the same procedure for use of another program.

A better way for loading several functions into a user workspace is by the use of GROUP function. An example is given below:

1. Sign-On

2.)CLEAR

3.)LOAD 2 STP4

4.)GROUP NAME RSIM RSIM1

5.)SAVE 2 STP4

6.)CLEAR

7.)COPY STP4 NAME

For better understanding of the above discussion, study the following program which solves a linear programming problem with 25 variables (including auxiliaries) and 15 constraints. (Command I22 is used to specify the amount of unused space remaining in the user workspace, in bytes).

Example:

Maximize: $Z = X_1+X_2+2X_3+3X_4+4X_5+5X_6+X_7+X_8+X_9+X_{10}$

Subject to:

$$X_1 \leq 10$$
$$X_2 \leq 20$$
$$X_3 \leq 30$$
$$X_4 \leq 40$$
$$X_5 \leq 50$$
$$X_6 \leq 60$$
$$X_7 \leq 70$$
$$X_8 \leq 80$$
$$X_9 \leq 90$$
$$X_{10} \leq 100$$
$$X_{11} \leq 24$$
$$X_{12} \leq 45$$
$$X_{13} \leq 55$$
$$X_{14} \leq 65$$
$$X_{15} \leq 75$$

```
VM/370 ONLINE     LJH359 QSYOSU

DIAL APL
⌊⍳⍺⌷⍷⌊ ~○ ⍺*⌷        021
)MASK
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
001) 15.15.38 04/06/76 MSA 103152

   A P L \ 3 6 0

      ⌶22
31868
      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      ⌶22
4696
      )CLEAR
CLEAR WS
      )COPY 2 STP4 SIMPLEX
SAVED  11.51.01 06/18/73
      ⌶22
31232
      B←11 12 13 14 15 16 17 18 19 20,⌷
⌷:
      21 22 23 24 25
      A←16 26 ⍴ 1 1 2 3 4 5 1 1 1 1 0 0,⌷
⌷:
      0 0 0 0 0 0 0 0 0 0 0 0 0 0,⌷
⌷:
      1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0,⌷
⌷:
      0 0 0 0 0 0 0 0 10,⌷
⌷:
      0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0,⌷
⌷:
      0 0 0 0 0 0 0 0 20,⌷
⌷:
      0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0,⌷
⌷:
      0 0 0 0 0 0 0 0 30,⌷
⌷.
```

```
      0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0,⍬
⍬:
      0 0 0 0 0 0 0 0 40,⍬
⍬:
      0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0,⍬
⍬:
      0 0 0 0 0 0 0 0 50,⍬
⍬:
      0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0,⍬
⍬:
      0 0 0 0 0 0 0 0 60,⍬
⍬:
      0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1,⍬
⍬:
      0 0 0 0 0 0 0 0 70,⍬
⍬:
      0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0,⍬
⍬:
      1 0 0 0 0 0 0 0 80,⍬
⍬:
      0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0,⍬
⍬:
      0 1 0 0 0 0 0 0 90,⍬
⍬:
      0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0,⍬
⍬:
      0 0 1 0 0 0 0 0 100,⍬
⍬:
      0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0,⍬
⍬:
      0 0 0 1 0 0 0 0 24,⍬
⍬:
      0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0,⍬
⍬:
      0 0 0 0 1 0 0 0 45,⍬
⍬:
      0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0,⍬
⍬:
      0 0 0 0 0 1 0 0 55,⍬
⍬:
      0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0,⍬
⍬:
      0 0 0 0 0 0 1 0 65,⍬
⍬:
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0,⍬
⍬:
      0 0 0 0 0 0 0 1 75
      R← B SIMPLEX A
      R
1     10
2     20
```

```
      2     20
      3     30
      4     40
      5     50
      6     60
      7     70
      8     80
      9     90
     10    100
     21     24
     22     45
     23     55
     24     65
     25     75
      0   1050
       I22
29272
       )OFF
001  15.27.01 04/06/76 MSA
CONNECTED      0.11.23  TO DATE    23.35.28
CPU TIME       0.00.06  TO DATE     0.18.15
TOTAL COST       $1.71  BALANCE     $152.71
⌊ρo* _ρo| α*⍞       021
```

A general discussion about APL charges is given next. APL costs for running each O.R. program will be computed as follows:

Total cost = Connection cost + Central Processing Unit Cost

Connection charges are directly proportional to the connection time. The IBM 370/156 terminal connect charge at Kansas State University is currently $ 2.5/hour. Clearly the total connect charge is directly related to the total time the user takes to enter data. The CPU time is the time which it takes the computer to execute a program. The CPU charges are directly related to the size of the problem array for the type of algorithms used in O.R. programs. The CPU charges for APL at Kansas State University is currently computed as follows:

Units = 12.00 x (minutes CPU Used)

Computer Charges = $1 per unit

Therefore, from the above discussion, the following equation can be derived for computing the total cost of running any O.R. program:

$$\text{Total cost} = (K_1)(\text{connection times in minutes}) + (K_2)(\text{number of elements in array})$$

As an example, let us compute the value of K_1 and K_2 for solving any linear programming problem using SIMPLEX program. The following table shows the computer costs for solving several L.P. problems:

No. of elements in array	Cost (dollar)	Connection time (minutes)
20	0.16	3
24	0.18	3.5
50	0.27	5

From the above table we conclude that K_1 is equal to 0.04 dollars per minute and K_2 is approximately equal to 0.0030 dollars per array

element. Therefore an L.P. problem with 416 array elements should be solved by SIMPLEX at a cost of about one dollar and seventy two cents.

Total Cost = (0.04)(12)+(0.0030)(416) = 1.7280 dollars

This cost estimates will generally be higher than the actual cost because the cost data for computing K_1 and K_2 included both the cost for running the O.R. problem and the cost for sign-on and sign-off.

I.C. A DISCUSSION OF MAXIMUM PROBLEM SIZE

There are two methods for solving any Operations Research problem using the programs presented in this report. The user can either load the whole public library or just the required function or functions into his workspace. In the first case, he will not be able to solve a big problem because there is not enough space available to him for use; however, the load command is simple to input and fast to execute. Copying a function puts more space at his disposal which enables him to solve bigger problems, but the copy command is more difficult to type and slower to execute. Now let us go through an example and show how a user can determine which method he should use for solving his Operations Research problems.

Example: Solve the following linear programming problem using the function SIMPLEX.

Maximize: $Z = X_1+X_2+X_3+X_4+X_5+X_6+X_7+X_8+X_9+X_{10}+X_{11}+X_{12}+X_{13}+X_{14}+X_{15}+X_{16}+X_{17}+X_{18}+X_{19}+X_{20}$

Subject to:

$$X_1 \leq 10 \qquad X_{11} \leq 110$$
$$X_2 \leq 20 \qquad X_{12} \leq 120$$
$$X_3 \leq 30 \qquad X_{13} \leq 130$$
$$X_4 \leq 40 \qquad X_{14} \leq 140$$
$$X_5 \leq 50 \qquad X_{15} \leq 150$$
$$X_6 \leq 60 \qquad X_{16} \leq 25$$
$$X_7 \leq 70 \qquad X_{17} \leq 35$$
$$X_8 \leq 80 \qquad X_{18} \leq 45$$
$$X_9 \leq 90 \qquad X_{19} \leq 55$$
$$X_{10} \leq 100 \qquad X_{20} \leq 75$$

There are two methods available for solving this problem:

METHOD I - Load the public library STP4 into the workspace.

The following information is available to the user after loading.

1 - Space available after loading = 4696 bytes

2 - Size of initial Simplex matrix = 21x41 = 861

3 - Total space required for storing the matrix

= 861x4 = 3444 bytes

4 - Program needs two copies of initial matrix

5 - There are about 1000 bytes required for storage of temporary variables.

So, from the above discussion we conclude that there are 4696 bytes of memory available for use and the program requires about 10,000 bytes for execution of this problem. Therefore, trying to execute this program will give an error message of WSFULL which means this problem is too big to be solved by this method.

METHOD II - Clear the WS and copy the function SIMPLEX into the workspace.

The following information is available.

1 - Space available after loading = 31000 bytes

2 - Total space required for execution of this program

= 10000 bytes

So, this problem will be solved by this method.

Other Considerations - The Simplex function is defined as follows:

R←B SIMPLEX A

The user can solve bigger problems by not passing A and B as arguments of Simplex, since parameter passing in APL/360 creates a copy of the input arguments. A and B can be created and saved in another workspace.

They can be copyed into the active workspace, and SIMPLEX can be edited so that A and B are global variables, rather than arguments. The table on the next page shows maximum problem size for each program in this report for the first two schemes discussed.

MAXIMUM PROBLEM SIZE FOR EACH PROGRAM IN THIS REPORT

TYPE OF PROBLEM	NAME OF FUNCTION	MAXIMUM PROBLEM SIZE (When Loading the Public Library)		MAXIMUM PROBLEM SIZE (When Copying the Function)	
		Space Available (bytes)	Typical Max. Size	Space Available (bytes)	Typical Max. Size
Linear Programming	SIMPLEX	4696	5 Constraints 10 Variables (Including Slacks)	31232	25 Constraints 50 Variables (Including Slacks)
Linear Programming	RSIM	4696	6 Constraints 12 Variables (Including Slacks)	30604	30 Constraints 60 Variables (Including Slacks)
Linear Programming	RSIM1	4696	6 Constraints 12 Variables (Including Slacks)	29560	30 Constraints 60 Variables (Including Slacks)
Linear Programming	LINPR	22840	15 Constraints 30 Variables (Including Slacks)	23580	15 Constraints 30 Variables (Including Slacks)
Assignment Problem	ASSIGN	4696	15 x 15	30200	30 x 30
Transportation Problem	TRANSPORT	4696	8 x 8	29784	18 x 18
Critical Path	CPM1	23816	20 activities	26180	25 activities
Critical Path	CPM	4696	12 activities	31836	36 activities

CHAPTER II

LINEAR PROGRAMMING

II.A. INTRODUCTION

Linear programming is used to find the values of a set of variables which maximize or minimize a linear combination of the variables, while also satisfying certain other linear equations or inequalities composed of the variables [6]. The following problem will demonstrate the use of linear programming.

II.A.1. EXAMPLE PROBLEM

Suppose a small manufacturing company makes two products, 1 and 2, and that they are fortunate enough to sell all they can produce at present.

Each product requires manufacturing time in the three manufacturing departments as indicated in Table 1. At this time, each department has a fixed amount of man hours available per week, as indicated in Table 2. The problem is to decide how many of each product to manufacture in order to make best use of the limited production facilities. Assume that the profit for each unit of product 1 is \$1 and for product 2 is \$1.50.

TABLE 1

MANUFACTURING-TIME REQUIREMENTS TO PRODUCE 1 UNIT OF PRODUCT PER DEPARTMENT

	Manufacturing time (hours)		
Product	Dept. A	Dept. B	Dept. C
1	2	1	4
2	2	2	2

TABLE 2

MANUFACTURING CAPACITY LIMITS

Dept.	Man-hours/week available
A	160
B	120
C	280

We can represent the problem mathematically as follows:

X_1 = the number of units of product 1 to be made.

X_2 = the number of units of product 2 to be made.

Maximize: Profit = $X_1+1.5X_2$

Subject to the restrictions:

$$2X_1+2X_2 \leq 160$$

$$X_1+2X_2 \leq 120$$

$$4X_1+2X_2 \leq 280$$

$$X_1 \geq 0$$

$$X_2 \geq 0$$

The last two restrictions of nonnegative values for the decision variables are always implied and thus not expressed formally.

The solution to this problem is to produce 40 units of product 1 and 40 units of product 2.

$$X_1 = 40$$

$$X_2 = 40$$

Profit = 100 dollars

Following are some common terms that are used in any linear programming problem.

Basic Variables: The variables in the solution are called basic variables.

Non-Basic Variables: The variables which are not in the solution.

Basic Feasible Solution: A basic solution to the constraint equations in which the nonzero values of the primal variables meet all the constraints. The lowest-cost basic feasible solution is the optimum in the minimization problem.

Slack Variable: An auxiliary variable introduced to convert an inequality constraint to an equation.

Example: $X_1 + X_2 \leq 40$

$X_1 + X_2 + S_1 = 40$

S_1 is being called a slack variable.

Surplus Variable: An auxiliary variable which is used in a greater than or equal constraint to an equation.

Example: $X_1 \geq 45$

$X_1 - U_1 + a_1 = 45$

a_1 is being called an artificial variable.

U_1 is being called a surplus variable.

Artificial Variable: Variable which is added to an equality constraint in order to have an identity matrix.

Example: $X_1 = 25$

$X_1 + a_2 = 25$

a_2 is being called an artificial variable.

II.A.2. SIMPLEX METHOD

One of the most widely used methods for solving the linear programming problems is the simplex method. The simplex method of linear programming uses the basic concepts of matrix algebra to solve for the intersection of two or more lines or planes. It begins at some

feasible solution, one which satisfies all restrictions, and successively obtains solutions at intersections which offer better values in the objective function. Finally this method of solution provides an indicator which determines when an optimum solution has been reached.

The steps required for solving a linear programming problem using simplex method can be summarized as follows:

Step1: Define the objective to be maximized in terms of decision variables.

Step2: Define the problem limitations in terms of constraint inequalities using the decision variables.

Step3: Convert the inequalities by adding slack, artificial or surplus variables.

Step4: Put the constraint equations and objective function in matrix form.

Step5: Select the key column, the largest positive value in objective function row.

Step6: Select the key row, the constraint limit reached first.

Step7: Iterate using row manipulations to make column a part of the identity matrix with the 1 in the key row. Divide the key row by the coefficient in the key row. Perform row operations using the key row to make all other entries in the key column zero.

Step8: Repeat Step7 until all values in the objective function row are nonpositive.

II.A.3. SENSITIVITY ANALYSIS IN L.P.

After a linear programming problem has been solved, it is recommended that the effect of discrete changes in the problem's

parameters on the current optimal solution be studied. Hence, it is usually advisable to perform sensitivity analysis or postoptimality analysis. If the objective function solution is relatively sensitive to changes in certain parameters, special attention must be given to these parameters so that the final solution gives consideration to most of their likely values. For such situations, it is not necessary to solve the problem from the very beginning each time a minor change is made.

There is one APL program available (LPSOLN) for performing sensitivity analysis in L.P. But LPSOLN should not be used if some solumns of the original matrix corresponds to artificial variables. (This constraint is stated in the original documentation and is not further explained [5]). An example is given later in this chapter.

II.A.4. REVISED SIMPLEX METHOD

The Simplex method described in section (II.A.2) is a straightforward algebraic procedure. However, this way of executing the algorithm is not the most efficient computational procedure for electronic digital computers because it computes and stores many numbers that are not needed at the current iteration and which may not become relevant for decision making at subsequent iterations. The only pieces of information relevant at each iteration are the coefficients of nonbasic variables, the coefficients of the entering basic variables in the other equations, and the right-hand side of the equations. It would be very useful to have a procedure that could obtain this information efficiently without computing and storing all the other coefficients. These considerations motivated the development of the revised

simplex method [6]. This method was designed to accomplish exactly the same things as the original simplex method, but in a way that is more efficient for execution on a digital computer. The revised Simplex computes and stores only the information that is currently needed, and it carries along the essential data in a more compact form.

II.B. COMPARISON AND EVALUATION OF EACH PROGRAM

There are currently five programs available (under APL) for solving linear programming problems. There are as follows:

1. SIMPLEX
2. RSIM
3. RSIM1
4. LINPR
5. LPSOLN (For Sensitivity Analysis Only)

For more information about the use of each one of the above programs refer to section II.C. Now, I would like to discuss each of these programs in detail and compare them with the available batch program.

1. SIMPLEX: This function uses the simplex algorithm to solve any small size linear programming problems. The only difficulty with using this program is that the user must set up the initial simplex matrix which includes inserting slacks and artificial variables at appropriate places. Each linear programming problem solved with this function must also be a MAXIMIZING problem.
2. RSIM: This program is almost the same as SIMPLEX except it uses the revised simplex algorithm [4] to solve the linear programming problem. In using this program the user does not need to specify the column number corresponding to the identity matrix as it is needed for SIMPLEX. There is also no need for addition of artificial variables which makes it a little easier

to use. The objective function must also be a MAXIMIZING problem.

3. RSIM1: This function helps a user to assemble the input for the function RSIM, then transfer control to RSIM, and gives the output with suitable labels. So, the program is much easier to use than SIMPLEX or RSIM.

4. LINPR: This program is probably the best and most convenient one for solving any linear programming problem (Maximization or Minimization). The problem is stated in the customary algebraic manner found in most textbooks on linear programming. The program can accomodate a maximum of 30 variables (including slacks and surplus) and 15 constraints (when loading the whole public library into the user workspace). The user does not have to set up the initial matrix.

5. LPSOLN: This function recalculates the optimal solution and does a sensitivity analysis for the price and requirements vectors for linear programming problem. LPSOLN should not be used if some columns of initial matrix correspond to artificial variables. (This constraint is stated in the original documentation and is not further explained [5]).

II.C. EXAMPLES

In this section, several examples for each program are given showing how they can be accessed and used.

There are currently four programs (under APL) for solving linear programming problems, as well as one program for performing sensitivity analysis. There are as follows:

1. SIMPLEX (Linear Programming,Maximization)
2. RSIM (Linear Programming,Maximization)
3. RSIM1 (Linear Programming, Maximization)
4. LINPR (Conversational Linear Programming, Maximization or Minimization)
5. LPSOLN (L.P. Sensitivity Analysis)

In order to obtain information about each of these programs issue the following commands after the Sign-on procedures.

1. SIMPLEX

After the Sign-on issue the following commands:

```
)LOAD 2 STP4

SIMPLEXHOW
```

```
VM/370 ONLINE     LJU359 QSYOSU

DIAL APL
⌊⍳⍺⌈⍷⌋ ~○ ⍺*⌈        021
)MASK
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
001) 10.57.22 01/27/76 MSA 103152

   A P L \ 3 6 0

      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      SIMPLEXHOW

          LINEAR PROGRAMMING
          R←B SIMPLEX A
          ENTERED:24/05/68

   THIS FUNCTION USES THE SIMPLEX ALGORITHM TO SOLVE THE
LINEAR PROGRAMMING PROBLEM SPECIFIED BY THE MATRIX A. THE
ROWS OF A GIVE THE CONSTRAINTS WITH NECESSARY SLACK AND
SURPLUS VARIABLES, AND ALSO ARTIFICIAL VARIABLES WHERE
NECESSARY TO MAKE UP AN IDENTITY MATRIX. THE LAST COLUMN
OF A, EXCEPT FOR THE FIRST ELEMENT WHICH IS ALWAYS ZERO,
GIVES THE REQUIREMENTS VECTOR. THE FIRST ROW OF A, EXCEPT
FOR THE LAST ELEMENT, GIVES THE PRICES. B IS A VECTOR
OF POSITIVE INTEGERS WHICH GIVES THE COLUMN NUMBERS
CORRESPONDING TO THE IDENTITY MATRIX.

   R IS A MATRIX WITH 2 COLUMNS. THE FIRST COLUMN GIVES THE
VARIABLES IN THE OPTIMAL BASIS EXCEPT FOR THE LAST ROW WHICH
IS ALWAYS ZERO. THE SECOND COLUMN GIVES THE VARIABLES IN THE
OPTIMAL BASIS EXCEPT FOR THE LAST ROW WHICH GIVES THE OPTIMAL
VALUE OF THE OBJECTIVE FUNCTION. IF THE PROBLEM HAS AN
UNBOUNDED SOLUTION, THEN THE PROPER INDICATION IS GIVEN IN
R WHICH IS THEN AN ALPHANUMERIC VECTOR.

   EACH LINEAR PROGRAMMING PROBLEM SOLVED WITH THIS FUNCTION
MUST BE A MAXIMIZING PROBLEM.
```

2. RSIM

Issue the following commands:

```
)LOAD 2 STP4

RSIMHOW
```

```
RSIMHOW
     LINEAR PROGRAMMING
     T←RSIM A
     ENTERED:20/12/67

THIS FUNCTION USES THE REVISED SIMPLEX ALGORITHM TO SOLVE
THE LINEAR PROGRAMMING PROBLEM SPECIFIED BY THE MATRIX A.
THE FIRST ROW OF A, EXCEPT FOR THE LAST COLUMN WHICH IS
ALWAYS ZERO, GIVES THE PRICES. THE REMAINING ROWS GIVE THE
CONSTRAINTS WITH NECESSARY SLACK AND SURPLUS VARIABLES, AND
THE REQUIREMENTS VECTOR WITH NON-NEGATIVE COMPONENTS IN THE
LAST COLUMN. NOTE THAT ARTIFICIAL VARIABLES ARE NOT REQUIRED.

T IS A MATRIX WITH 2 COLUMNS. THE FIRST COLUMN GIVES THE
VARIABLES IN THE OPTIMAL BASIS EXCEPT FOR THE LAST ROW WHICH
IS ALWAYS ZERO. THE SECOND COLUMN GIVES THE VARIABLES IN THE
OPTIMAL BASIS EXCEPT FOR THE LAST ROW WHICH GIVES THE OPTIMAL
VALUE OF THE OBJECTIVE FUNCTION. IF THE PROBLEM HAS EITHER
AN UNBOUNDED SOLUTION OR A NON-FEASIBLE SOLUTION, THEN THE
PROPER INDICATION IS GIVEN IN T WHICH IS THEN AN ALPHANUMERIC
MATRIX WITH A SINGLE ROW.

EACH LINEAR PROGRAMMING PROBLEM SOLVED WITH THIS FUNCTION
MUST BE A MAXIMIZING PROBLEM.
```

3. RSIM1

Type the following:

)LOAD 2 STP4

RSIM1HOW

RSIM1HOW

LINEAR PROGRAMMING
RSIM1
ENTERED:20/12/67

THIS FUNCTION ASSEMBLES THE INPUT FOR THE FUNCTION RSIM, TRANSFERS CONTROL TO RSIM AND GIVES THE OUTPUT WITH SUITABLE LABELS. THE DOCUMENTATION FOR RSIM SHOULD BE CONSULTED FOR THE FORM IN WHICH THE OBJECTIVE FUNCTION AND CONSTRAINTS SHOULD BE PUT.

4. LINPR

Type the following:

)LOAD 2 STP6

LINPRHOW

)LOAD 2 STP6
SAVED 11.52.01 06/18/73
LINPRHOW
CONVERSATIONAL L.P.
LINPR
ENTERED:24/05/68

THIS FUNCTION ALLOWS A LINEAR PROGRAMMING PROBLEM TO BE ENTERED IN AN ALGEBRAIC LANGUAGE SIMILAR TO THAT IN WHICH THE PROBLEM IS NORMALLY STATED. TO LEARN HOW TO USE THE FUNCTION TYPE

LINPR

AND ANSWER ALL QUESTIONS WITH YES OR NO.

REQUIRES LINPR, LIST1, LIST2, INPUT, CONVERT, ASSEMBLE, OUTPUT AND RSIM WHICH ARE ALL CONTAINED IN WORKSPACE STP6.

THE GLOBAL VARIABLES C, DATA, M, AND SLACK ARE GENERATED BY EXECUTION OF LINPR.

5. LPSOLN

Issue the following commands:

```
)LOAD 2 STP4

LPSOLNHOW
```

LPSOLNHOW

LINEAR PROGRAMMING
B LPSOLN A
ENTERED:24/05/68

THIS FUNCTION RECALCULATES THE OPTIMAL SOLUTION AND DOES A SENSITIVITY ANALYSIS FOR THE PRICE AND REQUIREMENT VECTORS FOR A LINEAR PROGRAMMING PROBLEM.

THE MATRIX A IS THE SAME MATRIX A AS REQUIRED FOR EITHER SIMPLEX OR RSIM, AND THE VECTOR B GIVES THE VARIABLES IN THE OPTIMAL BASIS. LPSOLN SHOULD NOT BE USED IF SOME COLUMNS OF A CORRESPOND TO ARTIFICAL VARIABLES.

THE OUTPUT CONSISTS OF THE FOLLOWING INFORMATION WITH IDENTIFYING LABELS:

MAXIMUM VALUE OF OBJECTIVE FUNCTION.
VARIABLES IN OPTIMAL BASIS AND THEIR VALUES.
MARGINAL VALUE, LOWER BOUND, RIGHT-HAND SIDE AND UPPER BOUND FOR THE RIGHT-HAND SIDE OF EACH CONSTRAINT.
LOWER BOUND, PRICE AND UPPER BOUND FOR EACH NON-ZERO PRICE.

INFINATE LOWER AND UPPER BOUNDS ARE INDICATED BY VALUES OF -7.237E75 AND 7.237E75. RESPECTIVELY.

Now, let us consider several examples using these programs.

Example 1:

Maximize $Z=X_1+1.5X_2$

Subject to: $2X_1+2X_2 \leq 160$

$X_1+2X_2 \leq 120$

$4X_1+2X_2 \leq 280$

Number of variables = 2

Number of slack variables = 3

The problem can be set up as follows:

$$Z = X_1+1.5X_2+0X_3+0X_4+0X_5$$

$$2X_1+2X_2+X_3 = 160$$

$$X_1+2X_2+X_4 = 120$$

$$4X_1+2X_2+X_5 = 280$$

Now, let us solve this problem using different programs.

Note: In these examples, data is not entered directly in the command lines, which is usually not a good idea except for tutorial problems. Rather the matrix and vector should be stored using a variable name, and the variables used as program arguments. This is given in the example on page 18 of the report.

Instructions for solving Example 1 using APL programs.

1. SIMPLEX

1. Sign-on
2. Issue command)LOAD 2 STP4
3. Specify the vector of positive integers which gives the column numbers corresponding to the identity matrix. In this case they are (3,4,5).
4. Specify the number of rows (including objective function) and columns (including Right hand side of constraints). In this case they are (4,6).
5. Then enter the objective function and constraints coefficients, when you finish entering the numbers hit the RETURN key.
6. Then type R.

```
      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      R←3 4 5 SIMPLEX 4 6 ρ1 1.5 0 0 0 0  2 2 1,□
□:
      0 0 160 1 2 0 1 0 120 4 2 0 0 1 280
      R
     1                40
     2                40
     5                40
     0               100
      )OFF
001  16.43.59 05/25/76 MSA
CONNECTED     0.02.24  TO DATE    25.57.55
CPU TIME      0.00.00  TO DATE     0.19.10
TOTAL COST      $0.17  BALANCE     $305.88
⌊ρ○* _ρ○| α*□       021
```

The output will read as follows:

$$X_1 = 40$$
$$X_2 = 40$$
$$X_5 = 40$$
$$Z = 100$$

2. RSIM

1. Sign-on
2. Issue command)LOAD 2 STP 4
3. Specify the number of rows (including objective function) and columns (including Right hand side of constraints). In this case they are (4,6).
4. Then type in the data.
5. Type T.

```
      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      T←RSIM 4 6 ρ 1 1.5 0 0 0 0 2 2 1 0 0 160,□
□:
      1 2 0 1 0 120 4 2 0 0 1 280
      T
     2                40
     5                40
     1                40
     0               100
      )OFF
001  16.46.56 05/25/76 MSA
CONNECTED     0.02.02  TO DATE    25.59.57
CPU TIME      0.00.01  TO DATE     0.19.10
TOTAL COST      $0.19  BALANCE     $305.69
⌊ρ○* _ρ○| α*□       021
]
```

The output will read as:

$$X_1 = 40$$
$$X_2 = 40$$
$$X_5 = 40$$
$$Z = 100$$

3. RSIM1

1. Sign-on
2. Issue command)LOAD 2 STP4
3. Type RSIM1, then the rest of instructions for using program come to you automatically.

```
      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      RSIM1

ENTER NUMBER OF VARIABLES INCLUDING SLACK AND SURPLUS
□:
      5
ENTER NUMBER OF CONSTRAINTS
□:
      3
ENTER 5 COST COEFFICIENTS
□:
      1 1.5 0 0 0
ENTER 3 BY 5 ACTIVITY MATRIX,ONE ROW AT A TIME
□:
      2 2 1 0 0
□:
      1 2 0 1 0
□:
      4 2 0 0 1
ENTER 3 RIGHT-HAND SIDE COEFFICIENTS
□:
      160 120 280


OPTIMAL VALUE OF OBJECTIVE FUNCTION IS 100

VARIABLE 2 AT LEVEL 40
VARIABLE 5 AT LEVEL 40
VARIABLE 1 AT LEVEL 40
      )OFF
001  10.51.25 01/27/76 MSA
CONNECTED     0.02.56  TO DATE     2.57.47
CPU TIME      0.00.01  TO DATE     0.06.31
TOTAL COST      $0.25  BALANCE     $202.46
⌊⍴⍥* _⍴⍥| ⍺*⊓       021
]
```

4. LINPR

1. Sign-on

2. Issue command)LOAD 2 STP6

3. Type LINPR, then the rest of instructions comes to you

```
VM/370 ONLINE     LJH359 OSYOSU

DIAL APL
⌊⍺⌈∈⌊ ~○ ⍺*⌈        021
)MASK
▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
001) 10.52.36 01/27/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP6
SAVED  11.52.01 06/18/73
       LINPR

   CONVERSATIONAL LINEAR PROGRAMMING

   DO YOU WISH ANY DESCRIPTION OF THE PROGRAM?
NO
MAXIMIZE
Z=X1+1.5X2
SUBJECT
2X1+2X2≤160
X1+2X2≤120
4X1+2X2≤280
END


OPTIMAL VALUE OF OBJECTIVE FUNCTION IS  100

VARIABLE 01 AT LEVEL 40
VARIABLE 02 AT LEVEL 40
VARIABLE 05 AT LEVEL 40



   ANY MORE PROBLEMS?
NO
       )OFF
001  10.56.17 01/27/76 MSA
CONNECTED      0.03.41  TO DATE     3.01.28
CPU TIME       0.00.02  TO DATE     0.06.33
TOTAL COST       $0.54  BALANCE     $201.92
⌊⍴○* _⍴○⌊ ⍺*⌈        021
]
```

5. LPSOLN

1. Sign-on
2. Issue command)LOAD 2 STP4
3. Specify the matrix A which is the same as A for either simplex or RSIM.
4. Specify the variables in the optimal basis.
5. Type in the data.

```
VM/370 ONLINE     LJH359 QSYOSU

DIAL APL
⌊⍳⍺⌈∈⌊ ~○ ⍺*⌈        021
)MASK
▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
001) 14.03.56 02/04/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP4
SAVED  11.51.01 06/13/73
       1 2 5 LPSOLN 4 6⍴ 1 1.5 0 0 0 0 2 2 1 0 0 160,□
□:
       1 2 0 1 0 120 4 2 0 0 1 280


OPTIMAL VALUE OF OBJECTIVE FUNCTION IS  100

VARIABLE 1 AT LEVEL  40
VARIABLE 2 AT LEVEL  40
VARIABLE 5 AT LEVEL  40

CONSTRAINT 1      0.25      120      160      173.3333333
CONSTRAINT 2      0.5       100      120      160
CONSTRAINT 3      0         240      280      7.237005577E75

PRICE 1       ¯7.237005577E75      1      1.5
PRICE 2       ¯7.237005577E75      1.5    2
       )OFF
001  14.06.26 02/04/76 MSA
CONNECTED     0.02.30  TO DATE     5.08.11
CPU TIME      0.00.01  TO DATE     0.06.58
TOTAL COST      $0.21  BALANCE     $190.97
⌊⍴○* _⍴○⌊ ⍺*⌈       021
```

Example 2:

Maximize $Z = X_1+2X_2+X_3+X_4$

Subject to:
$$X_1+2X_2+X_3+3X_4 \geq 400$$
$$X_1+ X_2+X_3+ X_4 \leq 200$$
$$X_1+2X_2 \leq 100$$
$$X_3+2X_4 \leq 200$$

Number of variables = 4

Number of slack variables = 4

Number of artificial variables = 1

The problem can be set as follows:

$$Z = X_1+2X_2+X_3+3X_4+0X_5+0X_6+0X_7+0X_8+0X_9$$
$$X_1+2X_2+X_3+3X_4-X_5+X_6 = 400$$
$$X_1+X_2+X_3+X_4+X_7 = 200$$
$$X_1+2X_2+X_8 = 100$$
$$X_3+2X_4+X_9 = 200$$

The procedure for using each program is given in example 1. The results are given below.

1. SIMPLEX

```
VM/370 ONLINE      LUI359 OSYOSU

DIAL APL
|⍳⍺⍞∈|  ~○ ⍺*⍞        023
)MASK
▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
003) 16.26.00 04/14/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP4
SAVED  11.51.01 06/18/73
       R←6 7 8 9 SIMPLEX 5 10⍴ 1 2 1 3 0 0 0 0 0 0 1 2.⍞
⍞:
       1 3 ¯1 1 0 0 0 400 1 1 1 1 0 0 1 0 0 200 1 2.⍞
⍞:
       0 0 0 0 0 1 0 100 0 0 1 2 0 0 0 0 1 200
       R
      2                50
      7                50
      5                 0
      4               100
      0               400
       )OFF
003  16.28.46 04/14/76 MSA
CONNECTED      0.02.46  TO DATE    23.39.19
CPU TIME       0.00.00  TO DATE     0.18.15
TOTAL COST       ¢0.19  BALANCE     ¢352.34
|⍴○* _⍴○| ⍺*⍞        023
```

2. RSIM

```
VM/370 ONLINE     LJH359 OSYOSU

DIAL APL
⌊⍳⍺⌈∈⌊ ~○ ⍺*⌈       023
)MASK
▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
003) 16.31.45 04/14/76 MSA 103152

   A P L \ 3 6 0

      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      T←RSIM 5 10 ⍴ 1 2 1 3 0 0 0 0 0 0 1 2 1 3 ¯1,⌈
⌈:
      1 0 0 0 400 1 1 1 1 0 0 1 0 0 200 1 2 0 0 0,⌈
⌈:
      0 0 1 0 100 0  0 1 2 0 0 0 0 1 200
      T
     2              50
     7              50
     5               0
     4             100
     0             400
      )OFF
003  16.34.04 04/14/76 MSA
CONNECTED     0.02.19  TO DATE    23.42.48
CPU TIME      0.00.01  TO DATE     0.18.16
TOTAL COST      ¢0.23  BALANCE     ¢351.97
⌊⍴○* _⍴○⌊ ⍺*⌈       023
]
```

3. RSIM1

```
      RSIM1

ENTER NUMBER OF VARIABLES INCLUDING SLACK AND SURPLUS.
□:
      8
ENTER NUMBER OF CONSTRAINTS
□:
      4
ENTER 8 COST COEFFICIENTS
□:
      1 2 1 3 0 0 0 0
ENTER 4 BY 8 ACTIVITY MATRIX,ONE ROW AT A TIME
□:
      1 2 1 3 1 0 0 0
□:
      1 1 1 1 0 1 0 0
□:
      1 2 0 0 0 0 1 0
□:
      0 0 1 2 0 0 0 1
ENTER 4 RIGHT-HAND SIDE COEFFICIENTS
□:
      400 200 100 200


OPTIMAL VALUE OF OBJECTIVE FUNCTION IS 400

VARIABLE 2 AT LEVEL 50
VARIABLE 6 AT LEVEL 50
VARIABLE 8 AT LEVEL 0
VARIABLE 4 AT LEVEL 100
      )OFF
001  16.30.36 01/28/76 MSA
CONNECTED      0.03.16  TO DATE     4.46.33
CPU TIME       0.00.01  TO DATE     0.06.57
TOTAL COST      $0.31   BALANCE     $192.83
⌊⍴○* _⍴○| ⍺*□       021
]
```

4. LINPR

```
      LINPR

   CONVERSATIONAL LINEAR PROGRAMMING

   DO YOU WISH ANY DESCRIPTION OF THE PROGRAM?
NO
MAXIMIZE
Z=X1+2X2+X3+3X4
SUBJECT TO
X1+2X2+X3+3X4≥400
X1+X2+X3+X4≤200
X1+2X2≤100
X3+2X4≤200
END


OPTIMAL VALUE OF OBJECTIVE FUNCTION IS  400

VARIABLE 02 AT LEVEL 50
VARIABLE 04 AT LEVEL 100
VARIABLE 05 AT LEVEL 0
VARIABLE 06 AT LEVEL 50


   ANY MORE PROBLEMS?
NO
      )OFF
002  15.42.10 01/28/76 MSA
CONNECTED      0.03.38  TO DATE     4.12.03
CPU TIME       0.00.02  TO DATE     0.06.53
TOTAL COST       $0.64  BALANCE    $195.05
⌊⍴⍟* _⍴⍟⌊ ⍺*∏       022
⁀
```

Example 3:

Minimize $Z = 2X_1+4X_2+X_3$

Subject to: $X_1+2X_2-X_3 \leq 5$

$2X_1-X_2+2X_3 = 2$

$-X_1+2X_2+2X_3 \geq 1$

Number of variables = 3

Number of slack variables = 2

Number of artificial variables = 1

Number of surplus variables = 1

The problem can be set up as follows:

$$Z = 2X_1+4X_2+X_3+0X_4+0X_5+0X_6+0X_7$$

$$X_1+2X_2-X_3+X_4 = 5$$

$$2X_1-X_2+2X_3+X_5 = 2$$

$$-X_1+2X_2+2X_3-X_6+X_7 = 1$$

Since LINPR is the only program available (under APL) for solving minimization problem, we use it to solve the above problem.

1. LINPR

Steps that should be followed.

1. Sign-on
2. Issue command)LOAD 2 STP6.
3. Type LINPR.
4. Then you will receive the rest of instruction for using the program.

```
      )LOAD 2 STP6
SAVED  11.52.01 06/18/73
      LINPR

   CONVERSATIONAL LINEAR PROGRAMMING

   DO YOU WISH ANY DESCRIPTION OF THE PROGRAM?
NO
MINIMIZE
Z=2X1+4X2+X3
SUBJECT TO
X1+2X2-X3≤5
2X1-X2+2X3=2
-X1+2X2+2X3≥1
END


OPTIMAL VALUE OF OBJECTIVE FUNCTION IS  1

VARIABLE 03 AT LEVEL 1
VARIABLE 04 AT LEVEL 6
VARIABLE 05 AT LEVEL 1


   ANY MORE PROBLEMS?
NO
      )OFF
002  14.00.46 01/27/76 MSA
CONNECTED      0.03.24  TO DATE     3.37.51
CPU TIME       0.00.02  TO DATE     0.06.43
TOTAL COST       $0.59  BALANCE     $198.39
⌊⍴⍵* _⍴⍵| ⍺*⍳       022
]
```

CONCLUSIONS

At the end of this section a discussion of the costs and memory requirements are given. However, it must be noted that the total cost for running APL programs are computed by addition of C.P.U. and connection costs. These costs are not the same in every case because every user could have different connection charges. For example, the user will be charged for sitting at a terminal and doing nothing, or the speed at which he enters the data could be different. But in any case they should be in the same range.

The tables below show the cost and memory requirements for solving several linear programming problems by each program (assuming the whole library is loaded into the user workspace).

1. SIMPLEX

Problem No.	No of Variables*	Cost Dollar	Memory Used Bytes
1	5	0.18	144
2	6	0.16	146
3	9	0.27	156

*Including slacks and surplus.

Results: From the above table we conclude that the costs are directly proportional to number of variables and memory requirements. Any small size problems (up to 10 variables and 5 constraints) could be easily solved by this program at a cost of about 30 cents.

2. RSIM

Problem No.	No. of Variables	Cost Dollar	Memory Used Bytes
1	5	0.19	84
2	6	0.20	88
3	9	0.39	100

Results: By observing the above table we will conclude that costs are approximately proportional to the number of variables. Small size problems (up to 12 variables and 6 constraints) could be solved by this program at a cost of about 45 cents.

3. RSIM1

Problem No.	No. of Variables	Cost Dollar	Memory Used Bytes
1	5	0.25	84
2	6	0.30	88
3	9	0.31	100

Results: RSIM and RSIM1 use the Revised Simplex Method for solving the linear programming problems. So, by looking at the above table, we will conclude that they use the same amount of memory with RSIM1 costing a little more to run.

4. LINPR

Problem No.	No. of Variables	Cost Dollar	Memory Used Bytes
1	5	0.54	252
2	6	0.58	176
3	9	0.64	264

Results: This is probably the best and most convenient program among the mentioned programs for solving any linear programming problems (30 variables including auxiliary variables and 15 constraints). The costs are directly proportional to the number of variables.

5. Batch Program

There is one batch program available in FORTRAN for solving any linear programming problems. This program can be run under WATFIV compiler. The cost for solving several problems are given below:

No. of Variables	Cost Dollar
5	0.46
6	0.46
7	0.46
9	0.46

In all cases these costs are the same because each time we have been charged 25 cents for C.P.U. (Current minimum C.P.U. charges at Kansas State University) and 21 cents for input/output.

CHAPTER III

THE ASSIGNMENT PROBLEM

III.A. INTRODUCTION

The assignment model is concerned with assigning a number of origins to the same number of destinations in order to optimize some function (usually cost or profit) [4]. The assignment model is defined as the optimization of the function:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}$$

Where C_{ij} are the cost (profit) coefficients, subject to the restriction:

$$\sum_{i=1}^{n} X_{ij} = 1.0 \qquad j = 1,2,3,\ldots\ldots,n$$

$$\sum_{j=1}^{n} X_{ij} = 1.0 \qquad i = 1,2,3,\ldots\ldots,n$$

$$X_{ij} = 0 \text{ or } 1 \text{ for all } i \text{ and } j$$

III.A.1. EXAMPLE PROBLEM

Four different buildings A, B, C, and D are to be constructed on a college campus by four different contractors 1, 2, 3 and 4. Because all contractors contribute heavily to the alumni fund, each is to build one building. Each contractor has submitted bids for the four buildings. The bids are shown in the following table.

BIDS FOR BUILDINGS
(00000 Omitted)

Building	Contractor			
	1	2	3	4
A	48	48	50	44
B	56	60	60	68
C	96	94	90	85
D	42	44	54	46

The optimal assignment for this problem is:

Contractor 4 builds building A.

Contractor 1 builds building B.

Contractor 3 builds building C.

Contractor 2 builds building D.

The optimum cost is $2,340,000.

The computer solution to this problem is given later in this chapter.

III.B. EVALUATION OF THE ASSIGNMENT PROGRAM

The assignment problem is a special type of linear programming problem where the resources are being allocated to the activities on a one-to-one basis. Thus, each resource or assignee is to be assigned uniquely to a particular activity or assignment. There is a cost C_{ij} associated with assignee i (i=1,2,...,n) performing assignment j (j=1,2,...,n), so that the objective is to determine how all the assignments should be made in order to minimize total costs.

ASSIGN is currently the only available program (under APL) for solving the assignment problems. It is very easy to use this program. The table below summarizes the computer costs and memory requirements for solving several problems (assuming the public library is loaded into the user workspace). These costs do not stay the same for every user because different connection charges are possible. But the costs should be in the same range.

Number of Constraints	Cost Dollar	Memory Used bytes
4	0.23	400
5	0.30	524
6	0.32	600
7	0.40	660
15	0.65	920

From the above results we conclude that costs are directly proportional to the memory used and number of constraints. This program can handle any size problem of up to 15 constraints at a cost of about 65 cents. ASSIGN will not be able to solve any problems for

which there exists more than one solution. (The example below shows a problem with two optimum solutions.)

Example. There are four qualified employees to be assigned to four different machines. Because of different individual training and experience the cost of successful completion of the given task on each machine is different for the employees. The unit costs to complete each task on each machine by the employees is given below:

(All Costs are in Dollars)

Men	Machine			
	A	B	C	D
1	10	15	16	18
2	14	13	16	10
3	11	9	8	18
4	13	13	11	9

The two optimum assignments are:

Assignment 1

Men	Machine	Cost
1	A	$10
2	B	$13
3	C	$ 8
4	D	$ 9

Total Assignment Cost = $40

Assignment 2

Men	Machine	Cost
1	A	$10
2	D	$10
3	B	$ 9
4	C	$11

Total Assignment Cost = $40

Since there exists two solutions for this assignment problem, we will not be able to solve it by the ASSIGN program. The user should follow the following steps if he does not know whether a problem has one or more solutions.

1. Sign-on
2. Type in)LOAD 2 STP4
3. Type in ASSIGN N

   (N is the number of rows in matrix)
4. Enter the cost matrix row by row.
5. When the user does not get the optimum assignment within a few seconds that means there exists more than one solution to the problem and the program ASSIGN will not be able to solve it.
6. Hit the ATTN key to interrupt the execution.
7. Examine the appropriate matrices to see the partial results. (These results may be useful or not).
8. Hit "→" and return key to cancel the execution.
9. Continue with another problem.

III.C. EXAMPLES

If you would like to get an explanation to the program follow the following steps:

1. Sign-on
2. Issue command)LOAD 2 STP4
3. Type in ASSIGNHOW

An example is given below:

```
vm/370 online      ljh35n qsyosu

DTAL APL
Lin□εL ~o α*□          021
)MASK
**********************
001) 11.04.12 02/17/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP4
SAVED  11.51.01 06/18/73
       ASSIGNHOW

            ASSIGNMENT PROBLEM
            ASSIGN N
            ENTERED:24/05/68

   THIS FUNCTION USES THE HUNGARIAN METHOD TO SOLVE AN N×N
INTEGER ASSIGNMENT PROBLEM WITH COST MATRIX C. THE PRICES
MAY BE ENTERED EITHER ROW-BY-ROW BY EXCUTING THE FUNCTION OR
MAY BE PREVIOUSLY STORED IN THE N×N MATRIX C, WHERE C IS A
GLOBAL VARIABLE WHICH IS UNCHANGED WHEN THE FUNCTION IS
EXECUTED.

   THE OUTPUT CONSISTS OF THE PROBLEM NUMBER AND DATE, THE
COST OF THE OPTIMAL ASSIGNMENT, AND AN N×N LOGICAL ASSIGNMENT
MATRIX IN WHICH ELEMENT (I,J) IS 1 IF AND ONLY IF ROW I IS
ASSIGNED TO COLUMN J IN THE OPTIMAL ASSIGNMENT.
```

Example 1. In order to solve the problem on page 57, follow the following procedures:

1. Sign-on
2. Type in the command)LOAD 2 STP4
3. Type in ASSIGN N

   (Where N is an integer indicating the number of rows which is always equal to the number of columns in the matrix.)
4. The rest of the instructions for using the program will be provided for you.

The solution to the above problem is given below:

```
      A P L \ 3 6 0

      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      ASSIGN 4
ENTER PROBLEM NUMBER.
⎕:
      1
DO YOU WISH TO ENTER THE COST MATRIX?
YES
ENTER ROW  1  OF COST MATRIX.
⎕:
      480000 480000 500000 440000
ENTER ROW  2  OF COST MATRIX.
⎕:
      560000 600000 600000 680000
ENTER ROW  3  OF COST MATRIX.
⎕:
      960000 940000 900000 850000
ENTER ROW  4  OF COST MATRIX.
⎕:
      420000 440000 540000 460000
DO YOU WISH TO LIST THE COST MATRIX?
YES

  480000  480000  500000  440000
  560000  600000  600000  680000
  960000  940000  900000  850000
  420000  440000  540000  460000
```

```
     PROBLEM NUMBER  1    DATE  21776

COST OF OPTIMAL ASSIGNMENT  2340000

  OPTIMAL ASSIGNMENT MATRIX
 0 0 0 1
 1 0 0 0
 0 0 1 0
 0 1 0 0
       )OFF
001   11.09.22 02/17/76 MSA
CONNECTED      0.02.53  TO DATE      5.56.17
CPU TIME       0.00.01  TO DATE      0.07.02
TOTAL COST       $0.25  BALANCE      $183.87
⌊p○* _p○| α*⊓       021
]
```

The solution will read as follows:

Contractor 4 builds building A.

Contractor 1 builds building B.

Contractor 3 builds building C.

Contractor 2 builds building D.

The optimum assignment cost is $2,340,000.

Example II. Four locations A,B,C, and D require a certain spare part. The four spare parts are stored in four different warehouses. If the mileages between warehouses and locations are as shown in the table, determine the least mileage shipping schedule.

MILEAGE

Warehouse	Location			
	A	B	C	D
1	230	200	210	240
2	190	210	200	200
3	200	180	240	220
4	220	180	210	230

For solving this problem use the same procedure as in Example I.

The solution will read as follows:

Location A should receive spare part from warehouse 3.

Location B should receive spare part from warehouse 4.

Location C should receive spare part from warehouse 1.

Location D should receive spare part from warehouse 2.

The least mileage shipping schedule is 790 miles.

```
      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      ASSIGN 4
ENTER PROBLEM NUMBER.
□:
      2
DO YOU WISH TO ENTER THE COST MATRIX?
YES
ENTER ROW  1  OF COST MATRIX.
□:
      230
ENTER ROW  1  OF COST MATRIX.
□:
      230 200 210 240
ENTER ROW  2  OF COST MATRIX.
□:
      190 210 200 200
ENTER ROW  3  OF COST MATRIX.
□:
      200 180 240 220
ENTER ROW  4  OF COST MATRIX.
□:
      220 180 210 230
DO YOU WISH TO LIST THE COST MATRIX?
NO



     PROBLEM NUMBER  2   DATE  21776

COST OF OPTIMAL ASSIGNMENT  790

  OPTIMAL ASSIGNMENT MATRIX
 0 0 1 0
 0 0 0 1
 1 0 0 0
 0 1 0 0
      )OFF
001  11.13.20 02/17/76 USA
CONNECTED     0.02.30  TO DATE     5.58.47
CPU TIME      0.00.01  TO DATE     0.07.03
TOTAL COST      $0.30  BALANCE     $183.57
```

Example III. Consider the situation of assigning 4 jobs to 4 machines. Cost of assigning a job to a machine is given below. The objective is to assign the jobs to machines (one job per machine) at the least total cost.

COST
Dollar

JOB	Machine			
	1	2	3	4
1	1	4	6	3
2	8	7	10	9
3	4	5	11	7
4	6	7	8	5

The computer solution to this problem is given in the next page.

```
      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      ASSIGN 4
ENTER PROBLEM NUMBER.
□:
      3
DO YOU WISH TO ENTER THE COST MATRIX?
YES
ENTER ROW  1  OF COST MATRIX.
□:
      1 4 6 3
ENTER ROW  2  OF COST MATRIX.
□:
      8 7 10 9
ENTER ROW  3  OF COST MATRIX.
□:
      4 5 11 7
ENTER ROW  4  OF COST MATRIX.
□:
      6 7 8 5
DO YOU WISH TO LIST THE COST MATRIX?
YES

   1   4   6   3
   8   7  10   9
   4   5  11   7
   6   7   8   5


     PROBLEM NUMBER  3   DATE  30276

COST OF OPTIMAL ASSIGNMENT  21

  OPTIMAL ASSIGNMENT MATRIX
 1 0 0 0
 0 0 1 0
 0 1 0 0
 0 0 0 1
      )OFF
001  10.47.41 03/02/76 MSA
CONNECTED     0.01.50  TO DATE   11.50.04
CPU TIME      0.00.01  TO DATE    0.13.48
TOTAL COST     $0.23   BALANCE    $236.01
|ρO* _ρO| α*□      021
]
```

The solution will be interpreted as follows:

Assign job 1 to Machine 1.

Assign job 3 to Machine 2.

Assign job 2 to Machine 3.

Assign job 4 to Machine 4.

CHAPTER IV

THE TRANSPORTATION METHOD

IV.A. INTRODUCTION

The transportation model is concerned with the transportation of a good (or service) from a number of sources to a number of destinations with the minimum total cost [2]. Mathematically, the problem is defined in the following manner. The objective equation is:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} X_{ij}$$

Which is subject to the restrictions:

$$\sum_{j=1}^{n} X_{ij} = a_i \qquad i=1,2,\ldots\ldots,m$$

$$\sum_{i=1}^{m} X_{ij} = b_j \qquad j=1,2,\ldots\ldots,n$$

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

$$X_{ij} \geq 0 \text{ for all } i \text{ and } j$$

Where X_{ij} is the amount allocated from origin i to destination j and C_{ij} is the transportation cost. The a_i values are the amounts available at each origin, and the b_j values are the amounts required at each destination.

IV.A.1. EXAMPLE PROBLEM

To illustrate the transportation method, we will examine the following problem. The Blacksburg concrete company has a contract to supply concrete for three construction projects in sites 1, 2, and 3. The amount of concrete required per day at the three construction sites are as follows:

Site	Daily Demand (Truck Load)
1	150
2	70
3	60
Total	280

The Blacksburg Concrete company has three concrete mixing plants in the towns A,B, and C. The Maximum production capacities per day of these three plants are:

Plant	Daily Production (Truck Loads)
A	120
B	80
C	80
Total	280

The cost accounting department of Blacksburg Concrete has estimated the transportation cost per truck load of concrete from the plants to the construction sites shown in the following table.

ESTIMATED TRANSPORTATION COSTS
(In Dollars)

From	Cost Per Truck Load		
	To site 1	To site 2	To site 3
Plant A	8	5	6
Plant B	15	10	12
Plant C	3	9	10

Given the production capacity of the plants, the amount of concrete demanded by the construction projects, and the unit transportation costs, the company seeks to determine the optimum transportation scheme that will minimize the total transportation cost.

The optimum solutions are as follows:

Plant	Transported To	Quantity	Unit Cost	Total Cost
A	1	70	$ 8	$ 560
A	3	50	$ 6	$ 300
B	2	70	$10	$ 700
B	3	10	$12	$ 120
C	1	80	$ 3	$ 240
	Total Transportation Cost		=	$1920

IV.B. EVALUATION OF THE TRANSPORTATION PROGRAM

One of the most important types of linear programming problems is the so-called transportation problem. Because of the special structure that exists in transportation problem (all constraint coefficients are one), this type of problem can be solved more efficiently by streamlined versions of the simplex method which exploits their special structure. These streamlined versions can cut down tremendously on the computer time required for large problems, and they sometimes make it computationally feasible to solve huge problems.

There is currently one program available under APL for solving any transportation problems. The program is very easy to access and economical to use. First the user should determine the size of the cost matrix including one row and one column for each supply and each demand. The user has to add dummy plant or warehouse if supply does not equal demand. Then enter the numbers row by row. After entering the numbers type in S to get the optimum matrix and MINCOST to get the total cost. The program gives one solution for which there exists more than one solution. The table below shows the computer costs and memory requirements for solving several problems (assuming the public library is loaded in user workspace).

Matrix Size	Cost (Dollar)	Memory Used (bytes)
4x4	0.30	428
5x4	0.42	512
4x9	0.62	928
6x7	0.65	952

This preceding table shows that costs are roughly proportional to memory used and the size of the matrix. This program can solve a problem with a maximum of 8 rows and 8 columns at a cost of about $1.50.

IV.C. EXAMPLE

If you would like to get a description to the transportation program, follow the steps below:

1. Sign-on
2. Type in)LOAD 2 STP4
3. Type in TRANSPORTHOW

An example is given below:

```
VM/370 ONLINE     LUH359 OSYOSU

DIAL APL
⌊ι⍺⌈∈⌊ ~○ ⍺*⌈        021
)MASK
▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
001 ) 10.57.21 03/02/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP4
SAVED  11.51.01 06/18/73
       TRANSPORTHOW

           TRANSPORTATION PROBLEM
           S←TRANSPORT COST
           ENTERED:10/11/68

   THIS FUNCTION USES THE PRIMAL-DUAL ALGORITHM TO SOLVE THE
TRANSPORTATION PROBLEM. COST IS AN (M+1)×(N+1) MATRIX, WHERE
COST[I;J] IS THE UNIT COST OF SHIPPING FROM ORIGIN I TO
DESTINATION J, WHERE I=1,...,M AND J=1,...N. COST[I;N+1] IS THE
AMOUNT AVAILABLE AT ORIGIN I, I=1,...,M, AND COST[M+1;J] IS THE
AMOUNT REQUIRED AT DESTINATION J, J=1,...,N. COST[M+1;N+1] IS
ARBITRARY. THE SUM OF THE ORIGIN AVAILABILITIES MUST BE EQUAL
TO THE SUM OF THE DESTINATION REQUIREMENTS.

   S IS AN M×N MATRIX WITH S[I;J] GIVING THE NUMBER OF UNITS
SHIPPED FROM ORIGIN I TO DESTINATION J IN THE OPTIMAL SOLUTION.
THE CORRESPONDING MINIMUM COST IS GIVEN IN THE GLOBAL SCALAR
VARIABLE MINCOST.
```

Example 1. Let us see how we can use the transportation program to solve the problem on page 71.

1. Sign-on
2. Type in)LOAD 2 STP4
3. Type in the cost matrix.

   First you must specify the size of the matrix then enter costs.

   Example: 4 4

   The first 4 is the number of rows.

   The second 4 is the number of columns.
4. Type in S to get optimum matrix.
5. Type in MINCOST to get transportation cost.

```
VM/370 ONLINE     LUM359 OSYOSU

DIAL APL
⌊⍳⍺⍞∈⌊ ~○ ⍺*⍞       022
)MASK
##########################
002) 13.37.22 03/18/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP4
SAVED  11.51.01 06/18/73
       S←TRANSPORT 4 4 ⍴ 8 5 6 120 15 10 12 80,⍞
⍞:
       3 9 10 80 150 70 60 280
       S
  70    0  50
   0   70  10
  80    0   0
       MINCOST
1920
       )OFF
002  13.38.52 03/18/76 MSA
CONNECTED      0.01.30   TO DATE    14.56.48
CPU TIME       0.00.01   TO DATE     0.15.09
TOTAL COST       ¢0.30   BALANCE     ¢212.23
⌊⍴○* _⍴○| ⍺*⍞       022
```

The output will be read as follows:

Plant	Transported to	Quantity
A	1	70
A	3	50
B	2	70
B	3	10
C	1	80

The total transportation cost is $1,920.

Example 2. Consider a transportation problem having the following costs and requirements table:

		DESTINATION						
		A	B	C	D	E	slack	supply
	1	$21	$12	$28	$17	$ 9	0	50
	2	$15	$13	$20	$50	$12	0	60
Sources	3	$18	$17	$22	$10	$ 9	0	40
	4	$999	$ 2	$10	$ 5	$ 1	0	70
	5	$33	$29	$35	$27	$23	0	30
Demand		40	30	50	60	50	20	250

Find the optimum transportation scheme.

In order to equalize the supply and demand we add another column to our cost matrix with zero cell costs and 20 as our demand.

For solving this problem follow the same steps as in Example 1.

```
VM/370 ONLINE     LJH359 OSYOSU

DIAL APL
⍳⍴⍞⍷⍵ ~○ ⍺*⍞       022
)MASK
▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
002) 13.44.14 03/18/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP4
SAVED  11.51.01 06/18/73
       S←TRANSPORT 6 7 ⍴ 21 12 28 17 9 0 50,⍞
⍞:
       15 13 20 50 12 0 60 18 17 22 10 8 0 40,⍞
⍞:
       999 2 10 5 1 0 70 33 29 35 27 23 0 30,⍞
⍞:
       40 30 50 60 50 20 250
       S
   0    0    0    0   50    0
  40    0   20    0    0    0
   0    0    0   40    0    0
   0   30   30   10    0    0
   0    0    0   10    0   20
       MINCOST
2530
       )OFF
002  13.46.56 03/18/76 MSA
CONNECTED     0.02.43   TO DATE    15.01.56
CPU TIME      0.00.03   TO DATE     0.15.14
TOTAL COST      ¢0.65   BALANCE     ¢211.13
⍳⍴○⍺ _⍴○⍳ ⍺*⍞       022
]
```

Optimum transportation scheme is as follows:

Source	Transported to (Destination)	Quantity
1	E	50
2	A	40
2	C	20
3	D	40
4	B	30
4	C	30
4	D	10
5	D	10
5	slack	20

Transportation cost is $2,530.

Example 3. The Arcose Manufacturing Campany has three factories: R,S, and T. These three factories supply one product to seven warehouses: A,B,C,D,E,F, and G. There are some slight differences in manufacturing costs for each factory, but the important factor is the cost of shipping from each factory to a particular warehouse. Each warehouse has a certain sales requirement, which is comparable to each factor having a certain capacity. The following table shows the costs of shipping from each warehouse, monthly factory capacities, monthly warehouse or sales requirements, and slack. The amount in the slack column is defined as the difference between total factory capacity and total warehouse requirements. Find an optimum shipping scheme.

SHIPPING COSTS (In Dollars)

To Warehouse	A	B	C	D	E	F	G	Slack	Factory Capacity
From Factory									
R	6	7	5	4	8	6	5	0	7000
S	10	5	4	5	4	3	2	0	4000
T	9	5	3	6	5	9	4	0	1000
Warehouse or Sales Requirements	1000	2000	4500	4000	2000	3500	3000	1000	21000

To solve this problem, follow the same procedure as in Example 1.

```
vm/370 online     1jh359 qsyosu

DIAL APL
⌊⍳⍺⌈⍷⌊ ~○ ⍺*⌈        022
)MASK
▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
002) 13.20.42 03/18/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP4
SAVED  11.51.01 06/18/73
       S←TRANSPORT 4 9⍴ 6 7 5 4 8 6 5 0 7000,⌈
⌈:
       10 5 4 5 4 3 2 0 4000,⌈
⌈:
       9 5 3 6 5 9 4 0 10000,⌈
⌈:
       1000 2000 4500 4000 2000 3500 3000 1000 21000
       S
  1000     0     0  4000     0     0  1000  1000
     0     0     0     0     0  3500   500     0
     0  2000  4500     0  2000     0  1500     0
       MINCOST
78000
       )OFF
002  13.23.43 03/18/76 MSA
CONNECTED     0.03.02  TO DATE    14.45.02
CPU TIME      0.00.02  TO DATE     0.15.00
TOTAL COST      $0.62  BALANCE     $214.59
⌊⍴○* _⍴○| ⍺*⌈       022
```

The output will read as follows:

Factory	Transportation (Warehouse)	Quantity
R	A	1000
R	D	4000
R	G	1000
R	slack	1000
S	F	3500
S	G	500
T	B	2000
T	C	4500
T	E	2000
T	G	1500

The total cost is $78,000.

CHAPTER V

PROJECT SCHEDULING

V.A. INTRODUCTION

In the development of large scale projects, management has long been seeking a systematic approach that permits it to (1) allocate the resources effectively, (2) maintain control over the entire activities, and (3) assess the results of changes in the rate at which various activities may be performed. Network analysis approach has provided a framework for thorough planning, scheduling, and controlling of such complex projects. The two basic project scheduling techniques are: Program Evaluation and Review Technique (PERT), and Critical Path Method (CPM). In this section, the emphasis will be on the basic concepts of the network analysis rather than on a specific technique.

Network Structure. A project schedule is considered to be structured as a network which embraces events and activities and explicitly identifies the interrelation among all activities. An event is represented as a node in the network. Usually an event indicates the commencement of one or more activities or the termination of one or more activities. Events are distinct points, in time. They do not consume any kind of resources. It is convenient to label events by numbers. An activity is represented in the network by a directed branch leading from one node to another.

An activity is defined as a job, task or operation characterized by an expenditure of resources for some period of time to complete a part of the project. The length of the directed branch has no significance. It does not correspond to the duration or cost of the activity. It is common practice to designate an activity by the number of the events it connects. For example, a direct branch leading from node i to node j, where $i < j$, identifies the corresponding activity $i \rightarrow j$. A dummy activity may be introduced in such a case where a particular event cannot occur before some other event, although is usually represented in the network by a broken directed branch with zero time. The directions of branches show graphically the logical precedence relations between project activities. The construction of a project network is based on the following concepts: (1) an activity cannot be started until its preceding event has occurred; (2) the event succeeding an activity cannot occur until all activities leading to the event are completed, and (3) the network contains no loops, that is, each node appears once and only once when any path is identified.

Important Terms Used in Project Scheduling

ES_{ij} Earliest start time for activity $i \rightarrow j$. It is the earliest time that activity $i \rightarrow j$ can start without interfering with completion times of its preceding activities.

EF_{ij} Earliest finish time for activity $i \rightarrow j$. It is the earliest time that activity $i \rightarrow j$ can be finished, assuming that its preceding activities have started at their earliest times.

LS_{ij} Latest finish time for activity $i \rightarrow j$. It is the time that activity $i \rightarrow j$ can start without interfering with the completion time of its succeeding activities.

LF_{ij} Latest finish time for activity $i \rightarrow j$. It is the latest time that activity $i \rightarrow j$ can be finished without delaying the completion of its succeeding activities.

TS_{ij} Total slack for activity $i \rightarrow j$. It is the excess time available to perform an activity given that all preceding activities start at the earliest time and all succeeding activities start at the latest times.

FS_{ij} Free slack for activity $i \rightarrow j$. It is the part of excess time available assuming that all preceding and succeeding activities start at the earliest times.

S_i Slack for event j. It is the time between the earliest and latest occurrence times for this event.

Critical Path Critical path is defined as the path of operations that must be done on time in order to complete the project on time. If any one of these operations are delayed, the project completion time will be increased by the amount of that delay.

V.A.1. EXAMPLE

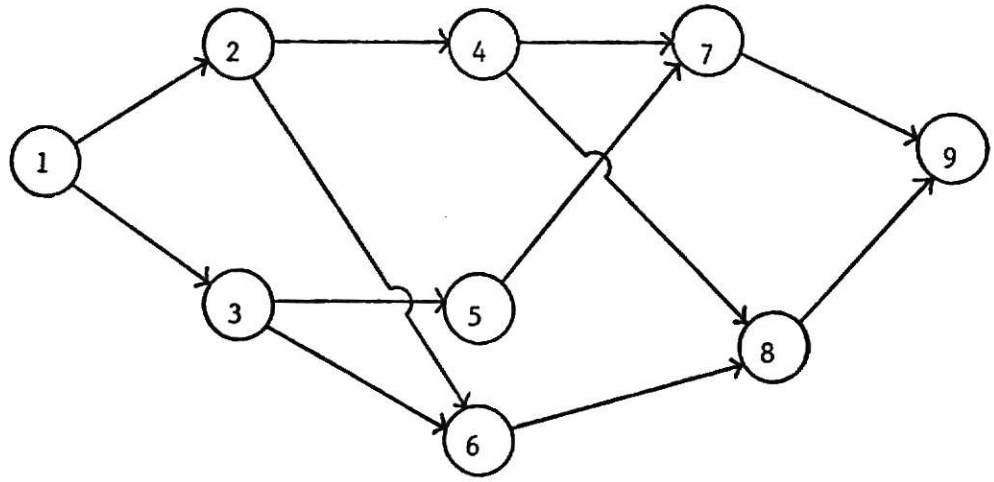
For illustration, consider a small construction project. The table below describes the logical requirements for the project.

Operation	Duration	Is Dependent Only Upon Operations
1	7	---
2	8	1
3	5	1
4	11	2
5	6	3
6	13	2,3
7	9	4,5
8	3	5,6
9 (END)	0	---

a) Use the information given above to construct the "circle" network.

b) Compute length of critical path, critical activities, early start and finish times, late start and finish times, total and free slacks.

Solution to Example Problem:



"circle" Network for construction project

Results of time analysis for construction project.

Operation	Duration	EST	EFT	LST	LFT	TF	FE
1	7	0	7	0	7	0	0
2	8	7	15	7	15	0	0
3	5	7	12	14	19	7	0
4	11	15	26	15	26	0	0
5	6	12	18	20	26	8	8
6	13	15	28	19	32	4	0
7	9	26	35	26	35	0	0
8	3	28	31	32	35	4	4
9	0	35	35	35	35	0	0

V.B. COMPARISON AND EVALUATION OF EACH PROGRAM

There are currently two programs available (under APL) for solving any CPM problems where activities are represented by nodes in the network. The CPM1 program is very easy to use. The information about each node is entered one by one (node number, duration, its successors). The output form the program is very well labeled and easy to read. On the other hand, the use of CPM requires that the user set up the matrix NTW. NTW is a matrix with N rows and M columns where N is the number of nodes numbered from 1 to N inclusive, in any order. Column 1 of NTW gives the node number, column 2 the node times and column 3 to M the immediate predecessors or successors (depending on the value of T). T is a scalar which is 0 (or 1) according as the immediate predecessors (or successors) of each job (node) are specified. The table below summarizes the cost and memory requirements for several examples (assuming the public library is loaded into the workspace).

Number of Activities	CPM1		CPM	
	Cost (Dollar)	Memory Used (bytes)	Cost (Dollar)	Memory Used (bytes)
8	0.27	756	0.45	700
10	0.29	808	0.60	1216
12	0.40	1032	0.80	1260
20	0.87	1516	--	--

The above table shows that CPM and CPM1 costs are roughly proportional to memory used and number of activities, any problems with maximum of 20 activities can be solved by CPM1 at a cost of about $0.90. On the other hand, any problems up to 12 activities can be solved by CPM at a cost of about $0.80.

V.C. EXAMPLES

This section gives procedures and examples for use of each program.

If you would like to get a description of each program, follow the steps below:

FOR CPM 1:

1. Sign-on
2. Type in)LOAD 2 STP5
3. Type in CPM1HOW

Then you will receive the description.

```
      )LOAD 2 STP5
SAVED  11.51.32 06/18/73
      CPM1HOW

          CRITICAL PATH METHOD
          CPM1
          ENTERED:24/05/68

   CPM1 IS A SET OF FUNCTIONS FOR PERFORMING A CRITICAL PATH
ANALYSIS FOR AN ACTIVITY-ORIENTED NETWORK. THE INPUT CONSISTS
OF THE NODE NUMBER, NODE DURATION AND SUCCESSOR NODE NUMBERS
FOR EACH NODE. IF THERE ARE N NODES, THEY SHOULD BE NUMBERED
IN ANY ORDER FROM 1 TO N. FOR CONVENIENCE OF OUTPUT THE
DURATIONS SHOULD BE INTEGERS. THE OUTPUT CONSISTS OF THE
FOLLOWING INFORMATION WITH SUITABLE LABELS: PROBLEM NUMBER
AND DATE, LENGTH OF CRITICAL PATH, CRITICAL ACTIVITIES, NODE
NUMBERS, DURATIONS, EARLY START AND FINISH TIMES, LATE START
AND FINISH TIMES, TOTAL AND FREE SLACKS. ERROR INDICATIONS
ARE GIVEN FOR INCORRECT NODE NUMBERING, MULTIPLE INITIAL AND
TERMINAL NODES, AND NETWORK LOOPS, AND COMPUTATION IS THEN
STOPPED.

   TO USE CPM1 LOAD STP5 AND TYPE
                         CPM1
IF THE INPUT DATA ARE TO BE TYPED IN, ANSWER YES TO THE
QUESTION 'IS THIS A NEW PROBLEM?', AND FOLLOW THE INSTRUCTIONS
THAT ARE TYPED OUT. THE INITIAL DATA ARE STORED, ONE ROW FOR
EACH NODE, IN THE GLOBAL VARIABLE DATA SO THAT CORRECTIONS
MAY BE MADE TO THE DATA WITHOUT ENTERING ALL THE DATA AGAIN.
TO RESTART AFTER MAKING CORRECTIONS TO DATA, ANSWER NO TO THE
QUESTION 'IS THIS A NEW PROBLEM?'.

   ALL OF THE FUNCTIONS REQUIRED FOR CPM1 ARE CONTAINED IN
STP5.

      )OFF
```

FOR CPM:

1. Sign-on
2. Type in)LOAD 2 STP4
3. Type in CPMHOW

```
      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      CPMHOW

          CRITICAL PATH METHOD
          CPT←T CPM NTW
          ENTERED:18/10/67

   CPM PERFORMS A CRITICAL PATH ANALYSIS ON THE NETWORK
REPRESENTED BY THE MATRIX NTW USING THE NODE-ORIENTED METHOD.
T IS A SCALAR WHICH IS 0 (OR 1) ACCORDING AS THE IMMEDIATE
PREDECESSORS (OR SUCCESSORS) OF EACH JOB (NODE) ARE SPECIFIED.
   INPUT: NTW IS A MATRIX WITH N ROWS AND M COLUMNS, WHERE N
IS THE NUMBER OF NODES NUMBERED FROM 1 TO N, INCLUSIVE, IN ANY
ORDER. COLUMN 1 OF NTW GIVES THE NODE NUMBERS, COLUMN 2 THE
NODE TIMES AS INTEGERS AND COLUMNS 3 TO M THE IMMEDIATE
PREDECESSORS OR SUCCESSORS OF THE NODES. IF ANY NODE HAS LESS
THAN M-2 IMMEDIATE PREDECESSORS OR SUCCESSORS, THE REMAINING
COLUMNS IN THE CORRESPONDING ROW MUST BE FILLED OUT WITH
ZEROS.
   OUTPUT: FOR EACH ANALYSIS THE FOLLOWING OUTPUT IS GIVEN AS
A MATRIX WITH ONE ROW FOR EACH NODE: NODE NUMBER IN SUCCESSOR
ORDER (THAT IS NODE I IS GIVEN BEFORE THE NODES WHICH ARE ITS
IMMEDIATE SUCCESSORS), NODE TIME, EARLIEST START TIME, TOTAL
SLACK AND FREE SLACK. EACH ROW ON THE CRITICAL PATH, THAT IS
NODES WITH A TOTAL SLACK OF ZERO, IS PRECEDED BY AN ASTERISK.
THE OUTPUT IS ALSO STORED, APART FROM THE ASTERISKS, IN THE
MATRIX CPT.
   SECONDARY OUTPUT: THE NETWORK NTW, SORTED IN THE SAME NODE
ORDER AS CPT, IS STORED IN THE MATRIX CPΔ. THE LOGICAL MATRIX
PSM WITH N ROWS AND COLUMNS GIVES THE PREDECESSOR-SUCCESSOR
RELATIONSHIPS BETWEEN NODES. ROW I GIVES THE SUCCESSORS OF THE
I-TH NODE OF CPΔ AND COLUMN I GIVES THE PREDECESSORS, WHERE AN
ENTRY OF 1 INDICATES A PREDECESSOR-SUCCESSOR RELATIONSHIP.
   ERROR-EXITS: IF THE N NODES ARE NOT NUMBERED, IN SORT
ORDER, FROM I TO N INCLUSIVE, 'NODE NUMBERING ERROR' FOLLOWED
BY A VECTOR OF SORTED NODE NUMBERS IS GIVEN, AND COMPUTATION
IS ENDED. IF THERE ARE ANY LOOPS OR DISCONTINUITIES IN THE
NETWORK 'LOOP/DISCONTINUITY AT NODE' FOLLOWED BY THE NODE
NUMBERS AT OR PRECEDING WHICH AN ERROR OCCURRED IS GIVEN, AND
COMPUTATION IS ENDED.

   REFERENCE: LEVY, THOMPSON AND WEST, HARVARD BUSINESS
REVIEW, VOL.41, NO.5, PP.98-108, SEPT.-OCT., 1963.
```

Now, let us discuss the procedure for use of each program.

Example I. Consider the construction project presented earlier in this chapter. Follow the steps below for the use of each program.

A. CPM1:

1. Sign-on
2. Type in)LOAD 2 STP5
3. Type in CPM1
4. Follow the instructions that are typed out.

```
VM/370 ONLINE     LJP359 QSYOSU

DIAL APL
\ιαΠε\ ~ο α*Π  .    021
)MASK
[redacted]
001) 42.22.27 03/26/76 MSA 103152

   A P L \ 3 6 0

      )LOAD 2 STP5
SAVED  11.51.32 06/18/73
      CPM1
IS THIS A NEW PROBLEM?
YES
ENTER PROBLEM NUMBER
⎕:
      1
ENTER NODE NUMBER,DURATION AND SUCCESSOR NODES,ONE NODE AT A
TIME IN ANY NODE ORDER.AFTER ALL DATA HAVE BEEN ENTERED,ENTER
A NODE NUMBER OF 0.
⎕:
      1 7 2 3
⎕:
      2 8 4 6
⎕:
      3 5 5 6
⎕:
      4 11 7 8
⎕:
      5 6 7
⎕:
      6 13 8
⎕:
      7 9 9
⎕:
      8 3 9
⎕:
      9 0
⎕:
      0
```

The computer output will be as follows:

```
PROBLEM NUMBER  1        DATE  32676

LENGTH OF CRITICAL PATH:  35

CRITICAL ACTIVITIES:  1   2   4   7   9

NODES:
1   2   3   4   5   6   7   8   9

DURATIONS:
7   8   5   11   6   13   9   3   0

EARLY START TIMES:
0   7   7   15   12   15   26   28   35

EARLY FINISH TIMES:
7   15   12   26   18   28   35   31   35

LATE START TIMES:
0   7   14   15   20   19   26   32   35

LATE FINISH TIMES:
7   15   19   26   26   32   35   35   35

TOTAL SLACK:
0   0   7   0   8   4   0   4   0

FREE SLACK:
0   0   0   0   8   0   0   4   0
        )OFF
001   42.24.52 03/26/76 MSA
CONNECTED      0.02.26   TO DATE    17.53.21
CPU TIME       0.00.01   TO DATE     0.16.23
TOTAL COST      $0.29    BALANCE   $190.21
Lρo* _ρo| α*Γ          021
```

B. CPM:

1. Sign-on

2. Type in <u>)LOAD 2 STP4</u>

3. Type in <u>CPT←T CPM NTW</u>

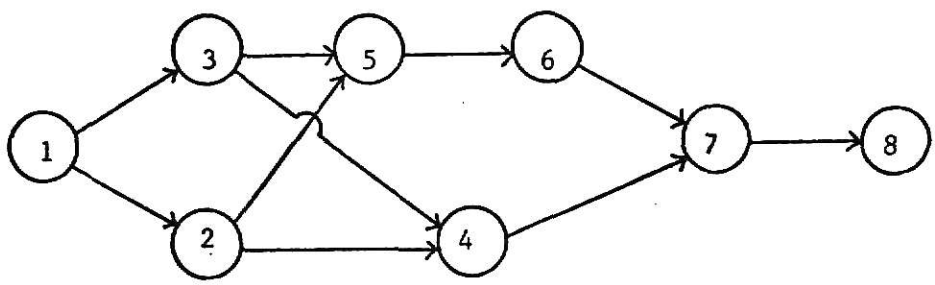
   For definition of T and NTW refer to program explanation.

```
VM/370 ONLINE     LJH359 OSYOSU

DIAL APL
|⍳⍺⌈⍷| ~○ ⍺*⌈       021
)MASK
##########################
001) 10.23.19 04/05/76 MSA 103152

   A P L \ 3 6 0

       )LOAD 2 STP4
SAVED  11.51.01 06/18/73
       CPT←1 CPM 10 4⍴ 1 0 2 0 2 7 3 4 3 8 5 7 4 5 6 7,⌈
⌈:
       5 11 8 9 6 6 8 0 7 13 9 0 8 9 10 0 9 3 10 0 10 0 0 0

  SUCCESSORS SPECIFIED

  NODE   TIME     ES     LS     EF     LF     TS     FS
   * 1      0      0      0      0      0      0      0
   * 2      7      0      0      7      7      0      0
   * 3      8      7      7     15     15      0      0
     4      5      7     14     12     19      7      0
   * 5     11     15     15     26     26      0      0
     6      6     12     20     18     26      8      8
     7     13     15     19     28     32      4      0
   * 8      9     26     26     35     35      0      0
     9      3     28     32     31     35      4      4
   *10      0     35     35     35     35      0      0

   * INDICATES CRITICAL PATH
      )OFF
001  10.26.11 04/05/76 MSA
CONNECTED      0.02.53  TO DATE    20.23.03
CPU TIME       0.00.02  TO DATE     0.17.10
TOTAL COST      ⌀0.60   BALANCE    ⌀173.54
|⍴○* _⍴○| ⍺*⌈       021
]
```

Example II. An assembly is to be made from two parts, A and B. Both parts must be turned on the lathe, and B must be polished while A need not be. The list of jobs to be performed, together with the predecessors of each job and the time in minutes to perform each job is given below:

JOB NO.	Description	Immediate Predecessors	Normal Times (Minutes)
1	START		0
2	Get materials for A	1	10
3	Get materials for B	1	20
4	Turn A on lathe	2,3	30
5	Turn B on lathe	2,3	20
6	Polish B	5	40
7	Assemble A and B	4,6	20
8	Finish	7	0

"Circle" network for example is given below:



Follow the same steps given in Example I for solving this problem.

A. CPM1:

```
      )LOAD 2 STP5
SAVED  11.51.32 06/18/73
      CPM1
IS THIS A NEW PROBLEM?
YES
ENTER PROBLEM NUMBER
□:
      2
ENTER NODE NUMBER,DURATION AND SUCCESSOR NODES,ONE NODE AT A
TIME IN ANY NODE ORDER.AFTER ALL DATA HAVE BEEN ENTERED,ENTER
A NODE NUMBER OF 0.
□:
      1 0 2 3
□:
      2 10 4 5
□:
      3 20 4 5
□:
      4 30 7
□:
      5 20 6
□:
      6 40 7
□:
      7 20 8
□:
      8 0
□:
      0
```

The computer output will be:

```
PROBLEM NUMBER  2      DATE  32676

LENGTH OF CRITICAL PATH:  100

CRITICAL ACTIVITIES:  1  3  5  6  7  8

NODES:
1  2  3  4  5  6  7  8

DURATIONS:
0  10  20  30  20  40  20  0

EARLY START TIMES:
0  0  0  20  20  40  80  100

EARLY FINISH TIMES:
0  10  20  50  40  80  100  100

LATE START TIMES:
0  10  0  50  20  40  80  100

LATE FINISH TIMES:
0  20  20  80  40  80  100  100

TOTAL SLACK:
0  10  0  30  0  0  0  0

FREE SLACK:
0  10  0  30  0  0  0  0
      )OFF
001  42.31.37 03/26/76 MSA
CONNECTED      0.02.28  TO DATE    17.58.21
CPU TIME       0.00.01  TO DATE     0.16.26
TOTAL COST      $0.27   BALANCE     $189.45
```

B. CPM:

```
VM/370 ONLINE     LJH359 QSYOSU

DIAL APL
Lι⍺⎕ϵ| ~○ ⍺*⎕          021
)MASY
[redacted]
001) 42.32.27 03/26/76 MSA 103152

   A P L \ 3 6 0

      )LOAD 2 STP4
SAVED  11.51.01 06/18/73
      CPT←1 CPM 8 4 ρ 1 0 2 3 2 10 4 5,⎕
⎕:
      3 20 4 5 4 30 7 0 5 20 6 0 6 40,⎕
⎕:
      7 0 7 20 8 0 8 0 0 0

  SUCCESSORS SPECIFIED

   NODE    TIME      ES       LS       EF       LF       TS       FS
  *  1        0       0        0        0        0        0        0
     2       10       0       10       10       20       10       10
  *  3       20       0        0       20       20        0        0
     4       30      20       50       50       80       30       30
  *  5       20      20       20       40       40        0        0
  *  6       40      40       40       80       80        0        0
  *  7       20      80       80      100      100        0        0
  *  8        0     100      100      100      100        0        0

  * INDICATES CRITICAL PATH
     )OFF
001  42.35.08 03/26/76 MSA
CONNECTED      0.02.41  TO DATE   18.01.02
CPU TIME       0.00.02  TO DATE    0.16.28
TOTAL COST       $0.47  BALANCE  $188.98
Lρ○* _ρ○| ⍺*⎕        021
```

REFERENCES

1. Iverson, K. E. A Programming Language, Wiley, (New York), 1962.

2. Lee, S. M. and Moore, L. J. Introduction to Decision Science, Petrocellicharter, 1975, p. 243-282.

3. Shaffer, L. R. and Ritter, J. B., and Meyer, W. L. The Critical Path Method, McGraw-Hill, (New York), 1965, p. 89-96.

4. Shamblin, J. E., and Stevens, G. T. Operations Research, A Fundamental Approach, McGraw-Hill, 1974, p. 325-331.

5. Smillie, K. W. An APL Statistical Package. Department of Computer Science, University of Alberta, Edmonton, Alberta. Second edition, 1969, p. 42-61.

6. Taha, H. A. Operations Research, An Introduction, Macmillan Company, (New York), 1971, p. 13-109.

A USER GUIDE TO INTERACTIVE (APL/360) PROGRAMS FOR OPERATIONS RESEARCH

by

MASOUD SHAMS AHMADI

B.S.I.E. Kansas State University
Manhattan, Kansas 1974

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1976

The purpose of the report is to provide a user documentation for the interactive use of a selection of operations research programs available at Kansas State University. The report will include the following information:

1- Sign on and sign off procedures.

2- Brief explanation to each technique used.

3- Procedure for the use of O.R. programs at terminal.

4- Sample program for each technique.

5- Comparison and evaluation of computing cost, merit of each system and user benefits.

The report will provide a nonoriented person with the necessary information to use available computer programs at any one of the Kansas State University's terminals. The report will also compare and evaluate the computational aspects of the O.R. programs which are documented.