NBS TECHNICAL NOTE 552

OMNITAB II
User's Reference Manual

U.S.
PARTMENT
OF
COMMERCE
National
Bureau
of
Standards

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Center for Computer Sciences and Technology, and the Office for Information Programs.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of a Center for Radiation Research, an Office of Measurement Services and the following divisions:

Applied Mathematics—Electricity—Heat—Mechanics—Optical Physics—Linac Radiation[2]—Nuclear Radiation[2]—Applied Radiation[2]—Quantum Electronics[3]—Electromagnetics[3]—Time and Frequency[3]—Laboratory Astrophysics[3]—Cryogenics[3].

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials and the following divisions:

Analytical Chemistry—Polymers—Metallurgy—Inorganic Materials—Reactor Radiation—Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services to promote the use of available technology and to facilitate technological innovation in industry and Government; cooperates with public and private organizations leading to the development of technological standards (including mandatory safety standards), codes and methods of test; and provides technical advice and services to Government agencies upon request. The Institute also monitors NBS engineering standards activities and provides liaison between NBS and national and international engineering standards bodies. The Institute consists of the following technical divisions and offices:

Engineering Standards Services—Weights and Measures—Flammable Fabrics—Invention and Innovation—Vehicle Systems Research—Product Evaluation Technology—Building Research—Electronic Technology—Technical Analysis—Measurement Engineering.

THE CENTER FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus within the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Center consists of the following offices and divisions:

Information Processing Standards—Computer Information—Computer Services—Systems Development—Information Processing Technology.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS and other agencies of the Federal Government; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world, and directs the public information activities of the Bureau. The Office consists of the following organizational units:

Office of Standard Reference Data—Office of Technical Information and Publications—Library—Office of Public Information—Office of International Relations.

---

[1] Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.
[2] Part of the Center for Radiation Research.
[3] Located at Boulder, Colorado 80302.

UNITED STATES DEPARTMENT OF COMMERCE
Maurice H. Stans, Secretary
NATIONAL BUREAU OF STANDARDS • Lewis M. Branscomb, Director

NBS TECHNICAL NOTE 552

ISSUED OCTOBER 1971

Nat. Bur. Stand. (U.S.), Tech. Note 552, 264 pages (Oct. 1971)
CODEN: NBTNA

OMNITAB II
User's Reference Manual

David Hogben, Sally T. Peavy, and Ruth N. Varner

Statistical Engineering Laboratory
Applied Mathematics Division
Institute for Basic Standards
National Bureau of Standards
Washington, D.C. 20234

For sale by the Superintendent of Documents, U.S. Government Printing Office
Washington, D.C. 20402 - Price $2 $3
Stock Number 0303-0918

OMNITAB II User's Reference Manual

David Hogben, Sally T. Peavy and Ruth N. Varner

OMNITAB II, a highly user-oriented system for a large computer, is designed to make computing easy, accurate and effective, particularly for persons who are not programmers. It is a general-purpose program, which can be learned quickly, for both simple and complex numerical, statistical and data analysis. OMNITAB executes instructions written in the form of simple English sentences. Problem-solving is further enhanced by the natural structure of the system and its many features. OMNITAB has been used successfully in government, industry and universities across the country and in several centers abroad. The system has been implemented on large computers of at least seven different manufacturers.

The original version of OMNITAB has been completely rewritten to make it as machine independent as possible and to implement many improvements. This manual describes Version 5.0. Details are presented so that the user can easily find the specific information needed in any particular instance. PART A is a simple, compact introduction to OMNITAB for people who have had no experience using a large computer. PART B describes the general and special features of the OMNITAB system. PART C gives explanations, with short examples, for the use of specific instructions. PART D is a complete alphabetical list of the instructions which are in the system.

Key words: Automatic printing, Bessel functions, data analysis, data manipulation, easy and effective programming in English, list of instructions, matrix operations, numerical analysis, OMNITAB II user oriented computing system, self-teaching, statistical analysis.

Introduction

OMNITAB is an interpretative computing system developed and maintained by the National Bureau of Standards to enable scientists to use a large computer easily, effectively and accurately for numerical, statistical and data analysis without having to become professional programmers. It was conceived by Joseph Hilsenrath about ten years ago and subsequently developed by him and his colleagues. A complete description of the first Version of OMNITAB is given in

HILSENRATH, J., ZIEGLER, G. G., MESSINA, C. G., WALSH P. J. and HERBOLD, R. J. (1966). OMNITAB: A Computer Program for Statistical and Numerical Analysis, National Bureau of Standards Handbook 101, Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402. Reissued 1968 with corrections.

During the past five years, the OMNITAB program has been completely rewritten to make it as machine independent as the state of the art will permit. In the course of this revision,

numerous additions and improvements have been made which necessitate complete documentation, including a new user's manual. The basic philosophy and spirit of OMNITAB, however, remain the same. After the rewriting of OMNITAB had begun it was decided to refer to the newer versions as OMNITAB II.

OMNITAB II is characterized by continual development with new instructions being added and existing instructions being improved. Hence, the present version of OMNITAB II can not be considered complete. In fact, there are some important instructions which were previously available, but so far have not been implemented. After some deliberation, the OMNITAB Steering Committee decided that any further delay in documentation would be unwise and further improvements should be postponed until this documentation is made available. After this manual is published, we shall resume development of OMNITAB II and continually try to respond to users' needs.

The manual is divided into four parts. PART A is a simple, compact introduction to OMNITAB II for people who have had no experience using a large computer or OMNITAB. PART B describes the general features of the OMNITAB II computing system. PART C gives detailed explanations, with short examples, for the use of specific instructions. PART D is a complete alphabetical list of all the instructions which are in the system. A comprehensive table of contents is given at the beginning. In addition, at the beginning of PART C there is a list of commands under functional titles and at the end of PART C there is an index to the commands which are described. The title page of each PART contains a short abstract of the contents.

The early history of OMNITAB was described in NBS Handbook 101 and will not be repeated here. Also, there will be no serious attempt to present examples of sets of OMNITAB instructions to solve problems as these appear in Handbook 101. Rather, we shall concentrate on presenting short examples to explain the use of particular instructions and to present more details of the instructions based on accumulated knowledge gained from the widespread use of OMNITAB II.

This manual is intended to be a reference manual to accompany the magnetic tape copy of OMNITAB II deposited with the National Technical Information Service (formerly the "Clearinghouse") where the OMNITAB II program can be obtained. See section B5.1 for further details, including a description of the other documentation of OMNITAB II.

This manual describes Version 5.0 of OMNITAB II which is in operation at the Washington D.C. site of the National Bureau of Standards in the batch processing mode. Users of OMNITAB II Version 5.0 at other computer centers may find some minor differences. Certainly, the control cards (section B1.4) will be different. Certain parameters, such as the size of the worksheet, may vary. Constants affecting overflow and underflow vary from one computer to another. The use of magnetic tapes (section C1.10) may be different. When operating in the time-sharing mode the action taken when fatal errors occur will be different.

The contributions of many NBS mathematicians and scientists were acknowledged in NBS Handbook 101. Many more have contributed to the present Version of OMNITAB. A great deal of the early work of converting OMNITAB to a machine independent form was done by Walter J. Gilbert who also contributed several new features. Much of the present programming and maintenance is done by Sally T. Peavy and Ruth N. Varner with help from Shirley Bremer. David Hogben and Sally T. Peavy, Statistical Engineering Laboratory, direct the development and maintenance of the OMNITAB II computing system. Many users have made valuable suggestions which have led to improvements. Other members of the Statistical Engineering Laboratory have made contributions, particularly to this manual and the formatting for the statistical instructions with automatic printing. Joseph M. Cameron did much to improve the set of instructions for matrix operations, to revise the curve fitting instruction, and to influence the accuracy of the computing algorithms. The following persons have contributed to one or more subprograms for the current Version of OMNITAB II: Walter J. Gilbert, William G. Hall, David Hogben, Robert C. McClenon, Carla G. Messina, Bradley A. Peavy, Sally

T. Peavy, M. Stuart Scott, Irene A. Stegun, Ruth N. Varner, Philip J. Walsh, Roy H. Wampler and Ruth Zucker.

We thank Bradley A. Peavy for considerable assistance in writing section 9 of PART C, Robert C. McClenon for writing section 10 of PART C and M. Stuart Scott for assistance in writing section C4.6. We thank Shirley G. Bremer for help in proof reading and in the writing of sets of instructions for verifying many of the statements made in the manual. Also, we thank the many readers of a preliminary draft for their comments and corrections.

It is impossible to write a manual such as this which is free of errors. Hopefully, the number of errors is small, but we will apprectiate any comments from readers so that we can pass on information to users.

How To Read This Manual

This is a reference manual; not a textbook. The style is not conducive to cover to cover reading. Rather, it is designed so that the user can easily and quickly locate the information he requires in any specific instance.

References are not given at the end of the manual, but appear in the text. There are two exceptions to this rule. Frequent reference is made to Hilsenrath et al. or NBS Handbook 101. The complete reference is given above. In section 4 of PART C, all the references for the statistical instructions are collected together and put in section C4.8. We have tried to supply a liberal amount of cross referencing. References to another section in the same PART of the manual give the section and sub-section number. Reference to a section in another PART prefix the section number by the PART letter as in "C4.8".

The novice should study PART A carefully and then peruse the table of contents. By thumbing through PART D, he will quickly grasp the scope of OMNITAB II. He can then proceed at his own pace to learn as much or as little as he requires. Particular attention should be given to the discussion of self-teaching in section B4.1. Sections 1.1, 1.2, 1.3 and 1.4 of PART B and sections 1.1, 1.2 and 1.3 of PART C need to be studied before attempting any computing. No attempt should be made to read the manual from cover to cover. Sections such as B1.8 and B4.4 are best read after the user has gained some experience using OMNITAB II.

The descriptions of instructions in PART C are written so that the most important information appears first and secondary information follows. The novice may wish to read only the first few sentences and defer further reading to a later use of the instruction. On the other hand, experienced users may find the opening remarks obvious and find the information they need to clear up some difficulty at the end of the discussion. In several places there is a general discussion at the beginning of a section or sub-section which applies to a group of instructions and this should not be overlooked.

The more experienced user may find section C11, Index Of Commands Described in PART C, most useful. He may also wish to take advantage of some of the special features described in PART B.

The experienced user will primarily use PART D. Section B4.1 should also be useful. He may need only occasional reference to PART B and PART C.

If OMNITAB is easy to learn and use, why is this manual so long? There are several reasons. First, we have tried to make the manual comprehensive so that virtually all questions will be answered. Second, OMNITAB is very flexible. For example, one can simply PRINT columns of numbers or one can print with a specified number of significant digits, print arrays, print matrices and even print matrices according to a specified FORMAT. Hence, many users' needs are satisifed. For simplicity, the user should use PRINT, but many other options are available. This flexibility in the system necessitates a longer manual. Third, some of the statistical instructions are simple to use, but provide a comprehensive automatic printing which requires a detailed explanation.

TABLE OF CONTENTS

PART A

BEGINNER'S OMNITAB

The National Bureau of Standards OMNITAB II computing system is designed to make computing easy, accurate and effective for scientists who are not programmers. OMNITAB is most useful for numerical analysis, data manipulation and statistical analysis. PART A is a compact, simple introduction to OMNITAB for people who have had no experience using a large computer. The material presented can be digested quickly and then the reader will be prepared to use OMNITAB and a computer to perform any set of calculations that can be done using a desk calculator plus a number that can not be done easily with a desk calculator. Only the bare essentials are described and no attempt is made to give a complete description of the OMNITAB system. Complete details are given in the other parts of this manual. PART A stands alone as a self-contained section. The discussion centers around an example. The example was selected because it is interesting and sufficiently non-trivial to be instructive. Thus, we could not avoid introducing technical material which is not germane to the explanation of OMNITAB. The computations are actually simple and the reader can easily gloss over technical expressions such as "cumulative distribution" without impairing his train of thought.

1. An Example.

"It has been noticed by astute observers that well used tables of logarithms are invariably dirtier at the front than at the back. Upon reflection one is led to inquire whether there are more physical constants with low order first significant digits than high." Thus, starts a paper by Pinkham (1961), ("On the distribution of first significant digits." Ann. Math. Statist., 32, 1223-1230). This provides the background for an interesting example to illustrate the basic features of the OMNITAB II computing system. Pinkham gives a theoretical discussion of why and to what extent the cumulative distribution of initial digits compares with the law log(n+1), here log means log to the base ten. By the law log(n+1) and cumulative distribution we simply mean that the digit 1 should occur 100xlog(2) percent of the time, both the digits 1 and 2 should occur 100xlog(3) percent of the time, the digits 1, 2 and 3 should appear 100xlog(4) percent of the time, etc. (The reader should not be distracted by the statistical aspects of this example.)

Let us examine the initial digit of the values of the 50 fundamental physical constants given in NBS Handbook 102, pages 42 and 43, (ASTM Metric Practice Guide, U. S. Government Printing Office) and see how well the law log(n+1) behaves. The fundamental physical constants are the Speed of light in vacuum, Faraday constant, Gravitational constant, etc.. The initial digits of the 50 values given in Handbook 102 are:

2, 1, 4, 6, 9, 5, 1, 1, 1, 1, 9, 2, 6, 1, 7, 1, 1, 5, 1, 5, 4, 1, 2, 3, 1,
2, 1, 5, 2, 7, 6, 2, 4, 2, 4, 9, 5, 1, 2, 2, 1, 4, 8, 2, 1, 3, 1, 2, 5, and 6.

One thing that we can do with these data is construct a frequency distribution showing the number of times 1 appears as the initial digit, the number of times 2 appears as the initial digit, and so on. We can then compare these observed frequencies with the expected (or theoretical) frequencies. To do this statisticians sometimes compute a statistic called chi-squared from the formula:

$$T = \text{SUM } (\text{observed-expected})^2/\text{expected}.$$

Here, the expected individual frequencies are derived from the cumulative relative frequencies by using the relation 50x(log(n+1)-log n). The simple calculations required to compute T are laid out below in a familar form. It is common practice in laboratories to write down the results of hand calculations on a multi-columned pad of ruled paper, or worksheet, as simulated in the following table. The basic data, in this case, are the observed frequencies in column (2). After this, calculations were performed on successive columns and the results put in columns (3) through (10).

n	Obs'd Freq.	n+1	log (n+1)	log (n)	diff. (4)-(5)	exp'd 50.x(6)	diff. (2)-(7)	(8)x(8)	T (9)/(7)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
1	16	2	.3010	.0000	.3010	15.05	0.95	0.902	0.06
2	11	3	.4771	.3010	.1761	8.80	2.20	4.840	0.55
3	2	4	.6021	.4771	.1250	6.25	-4.25	18.062	2.89
4	5	5	.6990	.6021	.0969	4.85	0.15	0.022	0.00
5	6	6	.7782	.6990	.0792	3.96	2.04	4.162	1.05
6	4	7	.8451	.7782	.0669	3.35	0.65	0.422	0.13
7	2	8	.9031	.8451	.0580	2.90	-0.90	0.810	0.28
8	1	9	.9542	.9031	.0511	2.56	-1.56	2.434	0.95
9	3	10	1.0000	.9542	.0458	2.29	0.71	0.504	0.22
	--				------	-----	-----		----
	50				1.0000	50.01	-0.01		6.13

We might expect the initial digits 1, 2, and 3 to appear approximately one third of the time, but the law log(n+1) = log(4) = 0.6, says we should expect them to appear closer to two thirds of the time. Note, the value of T = 6.13 is not at all unusually large thus indicating the law log(n+1) is reasonable in this case. (This is actually a chi-squared goodness-of-fit test which is explained, for example, in "Introduction to Statistical Analysis," W. J. Dixon and F. J. Massey, McGraw-Hill, p 209 ff.)

If one were to ask someone to perform the above calculations using a desk calculator, it is possible that the step-by-step instructions would be some verbalization of the headings that appear at the top of the columns. This is pretty much what you do to write a set of instructions in OMNITAB to have the calculations performed by a computer.

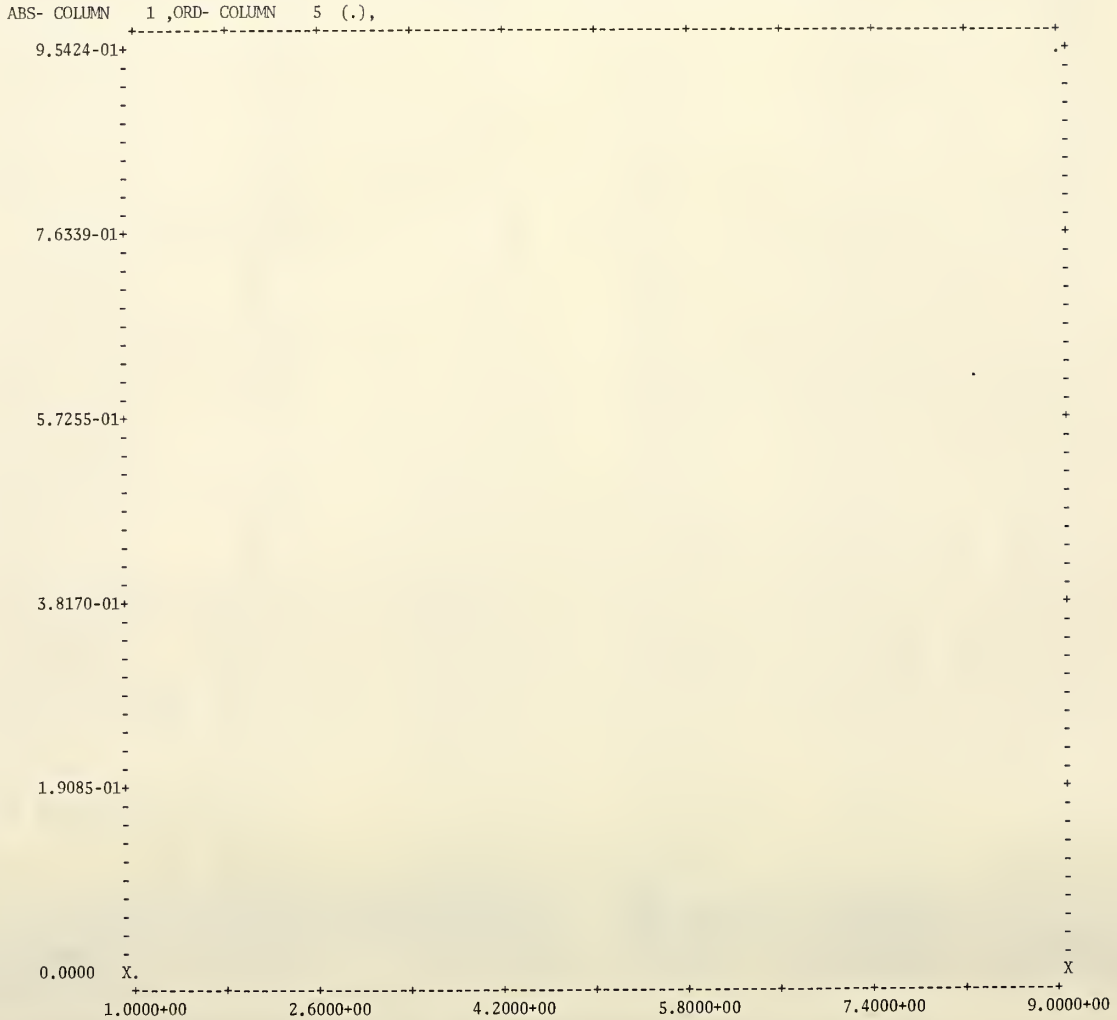
It is very helpful to imagine a large worksheet consisting of 201 rows by 62 columns. Operations are performed on the numbers in a column by writing instructions which closely resemble English, or at least technical English. The rules that govern the writing of the instructions are fairly simple and permit a wide latitude in form. The most important rule is one which enables the computer to distinguish between column numbers, such as 5, and constants, such as 50.0 in an instruction. All column numbers must be written without a decimal point and all constants must have a decimal point. Imagine for a moment, if you will, writing out a set of step-by-step instructions to perform the above computations and now examine a set of OMNITAB instructions to perform these calculations.

```
OMNITAB  5/19/70  distribution of initial digit of physical contants
SET initial digit of physical constants in column 21
2 1 4 6 9 5 1 1 1 1 9 2 6 1 7 1 1 5 1 5 4 1 2 3 1
     2 1 5 2 7 6 2 4 2 4 9 5 1 2 2 1 4 8 2 1 3 1 2 5 and 6.0
GENERATE n from 1.0 in steps of 1.0 to 9.0, put in column 1
FREQUENCY distribution for column 21, using 9 cells, put obs'd freq in column 2
     ADD 1.0 to column 1 and put result in column 3
     LOGTEN of column 3, put in column 4
     LOGTEN 1, 5
SUBTRACT column 5 from column 4 and put in column 6
MULTIPLY column 6 by 50.0 and put expected frequencies in colomn 7
SUBTRACT col 7 from col 2 and put differences in col 8
SQUARE column 8 and put in column 9
DIVIDE column 9 by column 7, put in column 10
SUM column 10 and put chi-squared in column 11
PLOT relative cumulative frequencies in column 5 against n in column 1
PRINT columns 1, 2, 7, 8, 10 and 11
STOP
```

The results obtained by using this set of instructions are shown on pages 4 and 5. The OMNITAB results differ in a few respects from the hand-calculated results above. The frequency distribution was computed, a plot of the theoretical, cumulative distribution is given and the amount of information printed is slightly different. Now, let us examine the set of instructions in some detail.

2. Discusssion Of Example.

Each instruction must be punched on a single Hollerith data processing card which has 80 columns. The first word of each instruction must be one of the valid command names in the OMNITAB vocabulary. To emphasize this point, the command name is written in capital letters. On the remaining portion of the card, only the numbers are important and required. The words in lower case letters are descriptive words which help the user understand the meaning of the instruction, but are ignored by the computer for computing purposes. (On a punched Hollerith card, only capital letters are possible.) Each instruction is interpreted and executed by the computer as it is encountered. Let us examine the instructions one by one in some detail.


ABS- COLUMN 1 ,ORD- COLUMN 5 (.),
9.5424-01
7.6339-01
5.7255-01
3.8170-01
1.9085-01
0.0000
X.
X
1.0000+00
2.6000+00
4.2000+00
5.8000+00
7.4000+00
9.0000+00

COLUMN 1	COLUMN 2	COLUMN 7	COLUMN 8	COLUMN 10	COLUMN 11
1.0000000	16.000000	15.051499	.94850063	.059771682	6.1281340
2.0000000	11.000000	8.8045628	2.1954372	.54743711	6.1281340
3.0000000	2.0000000	6.2469367	-4.2469367	2.8872505	6.1281340
4.0000000	5.0000000	4.8455003	.15449965	.0049262491	6.1281340
5.0000000	6.0000000	3.9590627	2.0409373	1.0521240	6.1281340
6.0000000	4.0000000	3.3473395	.65266055	.12725503	6.1281340
7.0000000	2.0000000	2.8995972	-.89959720	.27909915	6.1281340
8.0000000	1.0000000	2.5576260	-1.5576260	.94861355	6.1281340
9.0000000	3.0000000	2.2878744	.71212563	.22165680	6.1281340

LIST OF COMMANDS, DATA AND DIAGNOSTICS

```
SET INITIAL DIGIT OF PHYSICAL CONSTANTS IN COLUMN 21
2 1 4 6 9 5 1 1 1 1 9 2 6 1 7 1 1 5 1 5 4 1 2 3 1
     2 1 5 2 7 6 2 4 2 4 9 5 1 2 2 1 4 8 2 1 3 1 2 5 AND 6.0
GENERATE FROM 1.0 IN STEPS OF 1.0 TO 9.0, PUT N IN COLUMN 1
FREQUENCY DISTRIBUTION FOR COLUMN 21, USING 9 CELLS, PUT OBS'D FREQ IN COLUMN 2
     ADD 1.0 TO COLUMN 1 AND PUT RESULT IN COLUMN 3
     LOGTEN OF COLUMN 3, PUT IN COLUMN 4
     LOGTEN 1,5
SUBTRACT COLUMN 5 FROM COLUMN 4 AND PUT IN COLUMN 6
MULTIPLY COLUMN 6 BY 50.0 AND PUT EXPECTED FREQUENCIES IN COLOMN 7
SUBTRACT COL 7 FROM COL 2 AND PUT DIFFERENCES IN COL 8
SQUARE COLUMN 8 AND PUT IN COLUMN 9
DIVIDE COLUMN 9 BY COLUMN 7, PUT IN COLUMN 10
SUM COLUMN 10 AND PUT CHI-SQUARED IN COLUMN 11
PLOT RELATIVE CUMULATIVE FREQUENCIES IN COLUMN 5 AGAINST N IN COLUMN 1
PRINT COLUMNS 1, 2, 7, 8, 10 AND 11
STOP

 NATIONAL BUREAU OF STANDARDS, WASHINGTON, D. C.  20234, OMNITAB II VERSION 5.00 MAY 15, 1970
```

OMNITAB is the first instruction of any set. The date and other information on the card serves as a title which appears at the top of each page that is printed, along with the page number, as shown on pages 4 and 5. The OMNITAB instruction does a certain amount of initialization; in particular, it sets every entry in the 201 x 62 worksheet equal to zero.

SET is one of the instructions to get (enter) data into the worksheet. All the data that follow are read into the designated column of the worksheet until an OMNITAB instruction is encountered. The number 2 goes into row 1 of column 21, the number 1 into row 2, 4 into row 3, etc., and finally 6 into row 50. Note, we have not indicated anywhere that 50 numbers have to be entered. The computer automatically determines the number 50 for us. Considerable freedom is allowed in punching the numbers on a card. At the end of the second data card 6.0 is written, whereas all the other numbers do not have a decimal point. It is obvious when you read this sentence that 6 and 6.0 represent the same quantity and so they do when data are entered by OMNITAB. The user is free to put the decimal point in a data value or leave it out. Any words, that are not part of the OMNITAB vocabulary, can be entered anywhere on the card as comments, e.g. the word "and" at the end of the second data card. Also, data may start and appear anywhere on the card. All that is required is a comma, space or word to separate the numbers. One of the instructions for entering data, such as SET, must be used before any arithmetic is done.

GENERATE is another of the instructions for getting data into the worksheet (computer). The first number, 1.0, goes into row 1 and each succeeding row is obtained by adding the increment, the second 1.0, to the preceding row. The process is continued until the final value is reached, the third number 9.0 Thus, we obtain the numbers 1.0, 2.0 ... 9.0 in the first nine rows of column 1. In contrast to SET, the constants in this instruction must be written with a decimal point. Some people use the mathematical notation with parentheses as in

GENERATE 1.(1.)9. in column 1

FREQUENCY creates a frequency distribution using the specified number of cells or classes. This instruction produces the numbers in column (2) of the hand-calculated table of section 1. It gives the number of times (frequency) the digit 1 appears, the number of times the digit 2 appears and so on.

ADD is one of the arithmetic instructions. The first two numbers can be either constants (1.0) or column numbers which do not contain a decimal point (1). Note, ADD does not start in the first column of the card. Any instruction can appear anywhere on the card as long as one and only one instruction is on the card. Two cards can not be used for a single instruction. Of course, the instruction is equivalent to

ADD column 1 to 1.0 and put result in column 3

LOGTEN is another arithmetic instruction. Note, the difference in the two LOGTEN instructions. The words are superfluous, except for ease of reading and the first instruction could be written LOGTEN 3,4. Although the latter form is less desirable, the flexibility permits a great deal of freedom in writing an instruction.

SUBTRACT is clear. Notice the word "from" in the instruction which, although not necessary, explains the structure of the instruction. The instruction

SUBTRACT 4.0 from 7.0 put in columnn 56 would put the number 3.0 into each row of column 56.

MULTIPLY as punched on the cards listed on page 5 has the word column misspelled, but this will not affect the execution of the instruction. The command name MULTIPLY must be spelled correctly, but since the words which follow are ignored and are not necessary, misspellings, although not encouraged, are allowed.

SUBTRACT uses the abbreviation col for column which is very common with OMNITAB users.

SQUARE is another arithmetic instruction whose meaning is clear. The results in column 8 are needed to obtain column 9, but are of no intrinsic interest. Thus, they can be destroyed and we could have used the instruction SQUARE 8, 8. The last number in any instruction is always a column number indicating where the results are to be put in the worksheet after computations are completed. The results are put in the designated column and at the same time the previous numbers are erased, but not before the calculations are complete. The instruction ADD 1,1,1 is valid and would replace the numbers in column 1 by numbers having twice their value.

DIVIDE one column by another column means perform division row by row. In this case, divide the number in row 1 of column 9 by the number in row 1 of column 7 and put the result in row 1 of column 10; divide the number in row 2 of column 9 by the number in row 2 of column 7 and put the result in row 2 of column 10; and so on.

SUM is one of several data summarization commands. The result 6.13 is put in each row of column 11. Some people are puzzled to learn that the same number is put in each row. But OMNITAB works on columns of numbers and it would be more confusing to have different rules when the result of an operation is a single number rather than a column of numbers.

PLOT provides a convenient graphical display of the data via the printer. The scales along the axes of the plot are automatically determined by the computer.

PRINT is the basic instruction for obtaining printed results from the computer. Notice that the numbers are printed in a readable form with the decimal points lined up. This is a unique feature of OMNITAB which makes reading of results easy. The position of the decimal point is automatically determined by the magnitudes of the numbers in a column. Printing starts on a new page and at the top of each column appears the heading COLUMN with the appropriate column number.

STOP is always the last instruction. It tells the computer that you have finished using the OMNITAB system. At the same time it causes a LIST OF COMMANDS, DATA AND DIAGNOSTICS to be printed for reference purposes.

3. A Few Simple Rules.

(1) One and only one instruction is punched on a single Hollerith card.

(2) The first word of any instruction must be a valid command name in the OMNITAB vocabulary.

(3) Column numbers must be written (punched) without a decimal point.

(4) Constants which are part of an instruction must have a decimal point. (The decimal point is not necessary in data that are read into the worksheet.)

(5) OMNITAB must be the first instruction.

(6) STOP must be the last instruction.

(7) Any data value less than -8191 must have a decimal point, e.g. -9328 must be punched as -9328.0.

4. A Beginning List Of Instructions.

Each OMNITAB instruction consists of (i) a valid command name in the vocabulary such as ADD which is usually an imperative verb, (ii) one or more arguments, or numbers, which indicate the specific data to be operated on, e.g. the numbers 1.0, 2 and 3 in ADD 1.0,2,3 and (iii) descriptive words or phrases for the user's benefit as in ADD the number 1.0 to

column 2 and put the results in column 3. The last two, (ii) and (iii), are not always present as in STOP. The number of arguments allowed varies from instruction to instruction and depends on the form of the instruction. In the list below, the meaning or type of argument allowed should be clear from the context, but to remove any doubt four conventions are used:

(C) = a COLUMN number which must not have a decimal point
(K) = a CONSTANT which must contain a decimal point
(E) = EITHER a column number or a constant
(k) = an integer, other than a column number, without a decimal point.

SET the data on the following card(s) into column (C)
READ data into columns (C), (C), ... (C) one card per row
GENERATE from (K), in steps of (K), to (K) in column (C)

ADD (E) to (E) and put the results in column (C)
SUBTRACT (E) from (E) and put the results in column (C)
MULTIPLY (E) by (E) and put the results in column (C)
DIVIDE (E) by (E) and put the results in column (C)

SQUARE of (E) put in column (C)
SQRT of (E), put square root in column (C)
ABSOLUTE value of (E), put in column (C)
RAISE (E) to the (E) power and put results in column (C)

LOGTEN of (E), put in column (C) the logarithm to base ten
ANTILOG of (E) put in column (C)
LOGE of (E), put in column (C) natural logarithm
EXPONENTIAL of (E) put in column (C)

SIN of (E) radians put in column (C)
COS of (E) radians in column (C)

SUM all the values in column (C), put result in column (C)
AVERAGE the values in column (C), put result in column (C)
MAXIMUM value of column (C), put in column (C)
MINIMUM value of column (C), put in column (C)
DEFINE (E) into column (C)
SORT column (C), carry along columns (C), (C), ... (C)

FREQUENCY distribution for column (C), using (k) cells, put in column (C)
STATISTICAL analysis of column (C)

PLOT column (C) using vertical scale against column (C) using horizontal scale
PRINT columns (C), (C), ... (C)

OMNITAB (this must be the first instruction)
STOP (this must be the last instruction)

5. How To Use OMNITAB II.

A list of instructions and a list of rules has been given with a discussion of how to use them. A few additional comments about the use of instructions are now needed.

The READ instruction is similar to the SET instruction except data is entered into several columns one row at a time. The data on the first card goes into row 1 of the designated columns, the data on the second card into the second row and so on. For the READ instruction you punch one card for each row of data, whereas for the SET instruction you keep punching on a card and use only as many cards as you need to punch all the data for a single column.

The instruction STATISTICAL analysis is one of several instructions which give an automatic printing of a comprehensive set of results. This particular instruction prints a frequency distribution; 43 different statistics on measures of location, measures of dispersion, confidence intervals, linear trend statistics, tests for non-randomness, deviations from mean and other statistics; and the data with ranks, deviations from the mean and differences between adjacent ordered observations. An example is given in PART C.

The instruction DEFINE can be used to put a single value into every row as in DEFINE 2.0 into column 3, or it can be used to move a column from one part of the worksheet to another as in DEFINE column 2 into column 3. The instruction SORT puts data in increasing order and at the same time carries along data from the other columns which are specified. If the baseball batting averages .285, .310, .239, .268 and .293 are in column 36 and the number of walks 18, 26, 5, 3 and 12 are in column 38, then the instruction

SORT column 36 and carry along column 38

would change the numbers in column 36 to .239, .268, .285, .293 and .310. The numbers in column 38 would be changed to 5, 3, 18, 12 and 26.

The user may make errors, but they usually can be easily spotted and corrected. If a command name is not spelled correctly or has the wrong number or type of arguments, a FATAL ERROR will result and the instructions which follow are not executed. A message indicating the type of error is given in the LIST OF COMMANDS, DATA AND DIAGNOSTICS which is printed after the execution of the last instruction. Sometimes arithmetic errors occur as in attempting to take the square root of a negative number. This results in a diagnostic also, but does not affect the execution of subsequent instructions. A complete list of possible errors is given in PART B.

In addition to the OMNITAB control cards OMNITAB and STOP, a few control cards are necessary for accounting and administrative purposes. These cards differ from one computer center to another. Specific details should be obtained from your computer center. NBS users can find the necessary information in PART B. Information on how to punch cards, send cards to the computer to be run and receive printed results should also be obtained from your computer center.

The material provided in this part of the manual will enable you to solve a number of different problems easily and quickly. However, we have only scratched the surface of what can be done using OMNITAB. After using OMNITAB for awhile the beginning user will have many questions.

--- Are there other ways of printing data?
--- Can I modify the printing to have titles, column headings, etc.?
--- Can I choose the scales of a PLOT?
--- Can I work with matrices rather than columns?
--- Can I invert a matrix?
--- Can I do any numerical analysis such as interpolation?
--- Can I do more sophisticated statistical analysis using least squares?
--- Can I handle more than 201 data values?

The answer to each of these questions is "Yes, in several different ways." The answers are given in the remainder of the manual. The beginner should not be overwhelmed, but should proceed at his or her own pace. Examine the table of contents and glance through the complete LIST OF INSTRUCTIONS (PART D) at the end of this manual. In particular, study carefully the section on self-teaching. Then proceed to learn the material as needed and don't try to read it all at once. PART A stands alone as a self-contained section.

PART B

THE OMNITAB II COMPUTING SYSTEM

A general description of the National Bureau of Standards OMNITAB II computing system is given with illustrations. General features are described. Details on specific instructions are given in PART C.

1. How To Use OMNITAB II
2. Repeated Use Of Commands
3. Diagnostic Features And Accuracy
4. For More Effective Use Of OMNITAB II
5. The OMNITAB II Project

1. HOW TO USE OMNITAB II

1.1 Introduction

OMNITAB is a highly user-oriented general-purpose computing system. It is especially suited to problem solving in areas of data analysis, data manipulation, statistical analysis and numerical analysis.

The name OMNITAB comes from omni and tab. The former because it is one in a series of omnibus programs and the latter because it can handle a wide variety of tabular data; see Hilsenrath et al. (1966).

The central theme behind OMNITAB is that considerable extra effort should be expended by specialists in writing the master program so that users have only to exert a minimum amount of effort to use the computer easily and effectively. Every effort is made to utilize the computer to free the user from annoying, tiresome chores and to enable him or her to do the type of problem solving and data interpretation that is normally required in the chosen field.

A major consequence of this theme is that computing is done in a natural way by writing instructions to the computer in simple English sentences or abbreviations thereof. The simplicity of the system makes a more direct and immediate access to the computer possible for both non-programmers as well as experienced programmers.

It is very helpful to imagine a large worksheet consisting of 201 rows and 62 columns. Operations are performed on the numbers in columns by writing a series of instructions which closely resemble English or at least technical English sentences. Each instruction is punched on a single Hollerith card which has 80 columns. Each instruction is executed (except as in section 2) as it is encountered and interpreted by the computer. Hence, the order of the instructions controls the flow of computations in much the same way that hand calculations are performed and recorded on a multi-columnar pad. The instruction

MULTIPLY column 3 by 6.2 and put the result in column 5

causes the numbers in each row of column 3 to be multiplied by 6.2 and the results to be placed in the corresponding rows of column 5. The numbers which were in column 5 are replaced by the new results (only) after the operation is complete. Complete details on the writing and general use of instructions are given in the following sections.

In PART A a simple compact introduction to OMNITAB was given for the novice. In some instances there was a slight over-simplification. In the remaining portions of this manual details are given for both the novice and the experienced user. Some of the points made in PART A will be slightly modified to make them agree with the actual characteristics. It is assumed that the reader is familiar with the material in PART A.

1.2 Conventions, Definitions.

An OMNITAB instruction is the information punched on a single Hollerith card as in:

ADD 1.0 to column 24 and put the result in column 37 $ Y=X+1

For clarity of exposition, an instruction is said to consist of one or more of the following:

(a)	The command:	ADD
(b)	Arguments:	1.0, 24 and 37
(c)	Descriptive words:	to column, and put the result in column
(d)	Comments:	$ Y=X+1

Formerly, the terms command and instruction were used interchangeably, but here the above distinction is made. The command must be a valid name in the OMNITAB vocabulary; see PART D for a complete list. If the command is misspelled or is in any way incorrect the following FATAL ERROR will occur

NAME NOT FOUND IN LIBRARY

A complete discussion of error messages is given in section 3. All commands, except those listed in sections 1.11 and 1.16 consist of a single word. If the name has more than six letters, only the first six are necessary. For example, the command

STATISTICAL analysis

is often abbreviated to

STATIS

The word "analysis" is not part of the command, but may be used to clarify the meaning of the command.

Arguments are numbers which are either constants, row numbers, column numbers, etc. which indicate what specific numbers the operation (ADD) is to work on. Arguments must be separated from each other. This can be done by either (a) using one or more blank spaces, (b) using a comma or (c) using a descriptive word or phrase.

The descriptive words are helpful for understanding the meaning of the instruction, but are not used by the computer except in the printing of the LIST OF COMMANDS, DATA AND DIAGNOSTICS which is given at the end of the execution of a set of instructions. The use of comments is discussed in section 1.5.

Instructions are of three types: executable, non-executable and stored. Executable instructions are those which perform some type of operation immediately as in ADD. Non-executable instructions are used in detailed printing, for example FORMAT. The use of stored instructions for repeated execution is discussed in section 2.

In addition to instructions, the two other types of punched cards used with OMNITAB are data cards and operating system control cards. The operating system control cards are described in section 1.4.

Throughout this manual commands are written in capital letters. Non-essential descriptive words in an instruction are written in lowercase letters. No numerals are used in a command except in TITLE1, TITLE2, TITLE3, TITLE4, NOTE1 and NOTE2.

Some commands have a qualifier denoted by "L", where "L" indicates any one of the letters A, B, C, D, E or F; see section 1.16. These qualifiers are used to distinguish between formats and tapes (except in DUMMY "L"); see section 4.4. The qualifier (without quotation marks) is part of the command. One blank space must precede and follow the qualifier without any additional characters. All of the magnetic tape operation commands have either one or two qualifiers. One instruction, RESET "V", has the qualifier "V", where "V" denotes the letter V, W, X, Y or Z; see section 1.7.

Parentheses enclosing a letter indicate the type of argument allowed in an instruction. Lower case letters always represent integers which must be written without a decimal point. Examples are (r) = the number of rows and (c) = the number of columns. Capital letters are used as follows:

(C) = a COLUMN number which must be written without a decimal point
(E) = EITHER a column number or a constant
(K) = a constant which must be written with a decimal point
(N) = an instruction NUMBER with or without a decimal point
(R) = a ROW number which must be written without a decimal point

Constants can be written in many different ways. The number 123.4 can be written simply as 123.4 or it can be written using some kind of notation indicating the size of the exponent as in 1.234E+2, 1.234E2 or even 1.234+2.

In some instructions, like PRINT, it is clear from the context that the arguments must be column numbers. In some instructions, like ADD, the form of the instruction indicates that either constants or column numbers may be used as arguments. To make the distinction between constants and column numbers unambiguous, constants are always written with a decimal point and column numbers are always written as integers without a decimal point.

The word OMNITAB is used for several different purposes and some ambiguity may result. Also, there has been some discussion and comment on whether OMNITAB is a program, a language or even a statistical package. To alleviate these difficulties, the following definitions will be used:

OMNITAB master program: The entire set of ANSI FORTRAN subprograms which exists on magnetic tape or punched cards.

OMNITAB language: The vocabulary of imperative words and the grammar, syntax and rules for its use.

OMNITAB vocabulary: The list of commands, largely imperative verbs, which comprise the first word of an OMNITAB instruction.

OMNITAB instruction: The combination of a command from the vocabulary, numerical constants and column numbers, with or without intervening "noise" words.

OMNITAB set of instructions: A set of instructions beginning with the command OMNITAB, including data cards, written by a user to solve a specific problem.

OMNITAB philosophy: The strong user-oriented features of OMNITAB such as automatic printing and printing of error bounds, without which OMNITAB could exist, but would be substantially less effective.

OMNITAB project: The continuing activity designed to make the use of OMNITAB more effective. These activities include maintenance of the system, programming of new features, operation of a feedback system from users, distribution of program tapes, preparation of the OMNITAB Newsletter, and planning and coordination of workshops, seminars and lectures.

The word OMNITAB by itself may signify any of the above, but usually refers to the OMNITAB computing system which encompasses all of the above.

1.3 A Few Simple Rules.

A few restrictions are imposed upon the user by the rules below. However, the user is allowed wide latitude in writing instructions or data cards. For example:

(a) Any instruction can appear anywhere on the card. Punching does not have to start in card column 1.

(b) Data can be spaced in any way on a card and descriptive words or phrases (without numerals) may appear anywhere on the card. If a descriptive word is used at the beginning of a data card, it must not be a valid name in the OMNITAB vocabulary.

(c) The extent to which descriptive words are or are not used in an instruction is up to the user. Liberal use of descriptive words is often helpful.

(d) When the last argument in an instruction is a column number, as is usually the case, indicating where results are stored, the column number does not have to differ from column numbers previously used in the instruction. E.g.,

SQUARE 8,8

is a valid instruction and would replace all the numbers in column 8 by the same numbers squared.

(e) The length of a column is usually automatically set when data are entered into the worksheet and need be of no concern to the user. For a few exceptions see section 1.6.

(f) The master program automatically makes the distinction between data and instructions.

(g) FORMAT statements are not required. They may be used, if desired.

The following is a short list of simple rules which apply in general. Rules which apply only to a single instruction are given with the description of the instruction in PART C.

(1) One and only one instruction is punched on a single Hollerith card.

(2) The first word of any instruction must be a valid command in the OMNITAB vocabulary.

(3) Column numbers must be written (punched) without a decimal point.

(4) Constants which are part of an instruction must have a decimal point. (The decimal point is not necessary in data that are read into the worksheet.)

(5) OMNITAB must be the first instruction in any set of instructions.

(6) STOP must be the last instruction.

(7) The maximum number of arguments allowed in a single instruction is 100.

(8) Arguments must be separated from each other by a blank space, comma, or any character other than an asterisk or a dollar sign.

(9) All matrix and array operation commands, except AAVERAGE, ACOALESCE and MMATVEC, use the first four arguments to determine the location and size of the array or matrix.

(10) The use of asterisks, see section 1.8, is governed by special rules. In particular, a space cannot be used for a comma to separate numbers between asterisks. An asterisk should not be used in a descriptive word or phrase.

(11) The first instruction following a data card must not be a stored instruction.

(12) Any data value less than -8191 must have a decimal point, e.g., -9328 must be punched as -9328.0.

(13) Data must be within the range of the computer. For NBS users this means approximately 11 digits or less for integers (2^{35}-1) and floating-point numbers should be less than 10^{38}.

1.4 NBS Operating System Control Cards.

OMNITAB has two essential control commands OMNITAB and STOP, which are discussed in PART C. In addition, control cards are required by the computer center for administrative and accounting purposes. Users of any computer other than the NBS computer should consult their computer center staff for assistance. Users of the NBS computer have to use the following three control cards. The symbol, 8, represents the multiple 7-8 control punch which must be in card column 1.

```
8P RUN NAMEXX,00000,MT,MP,MC
8Ø   XQT ØMNITAB

     (your set(s) of OMNITAB instructions)

8 FIN
```

On the first card:

P = Priority which is either the letter A, B, C or D as you choose. The priority affects (i) the computing cost, (ii) how quickly your sets of instructions will be processed and (iii) sets limits on the execution time, number of pages that can be printed, number of cards that can be punched and number of magnetic tapes that can be used. (If a blank space appears, D priority is assumed.)

00000 = the five digit task number assigned to you by the Computer Services Division.

NAMEXX = the six character identification assigned to you by the Computer Services Division.

MT = the maximum number of minutes of computing time expected.

MP = the maximum number of pages to be printed.

MC = the maximum number of cards to be punched.

If the priority, name or task number is incorrect, your computer run (set of instructions) will be aborted. In other words, the computer will not accept your set of instructions to be processed. Similarly, if there is a conflict between the priority and either MT, MP or MC, then the run will be aborted. If MT, MP or MC is reached, your computing, printing or punching will all cease and you will only get back whatever has been produced up to this point.

Usually, 1 or 2 is used for MT. A great deal of computing can be done in 1 minute. Occasionally several minutes are needed, particularly if several sets of instructions are being used or if an iterative procedure is being used. The computing time almost never exceeds 10 minutes. The run time can now be given to the nearest tenth of a minute. Hence, the following are all acceptable times: 0.6, 1, and 2..

The number of pages printed is often less than 50 and rarely greater than 100. Keep in mind that about five extra pages are printed by the executive system for administrative purposes.

Usually cards are not punched and MC can be left blank. However, when you do have cards to be punched, be sure to set MC to equal or exceed the number of cards that have to be punched.

The priority establishes limits on MT, MP, and MC as follows:

PRIORITY	MAXIMUM TIME	MAXIMUM PAGES	MAXIMUM CARDS	MAXIMUM TAPES
A	2 min.	75	0	0
B	5 min.	125	500	3
C	10 min.	175	500	5
D	unlimited	2500	20,000	7

A slash is used to distinguish the letter, Ø, from the number zero. If the letter, Ø, in the second card column of the second system control card is missing, overflow and underflow diagnostics will be printed by the executive system. These messages are sometimes informative, but are usually difficult to relate to OMNITAB.

In addition to the above system control cards, the user must precede all cards by a completed FORM NBS-112, Computer Services Instruction Card. An example of a completed form (with fictitious data) follows. This white card immediately precedes the "RUN" card.

PRIORITY: D | NAME: HOGBEN | TASK: 12345 | DATE: 9/30/70 | BIN (DO NOT WRITE)

REEL NO	RING IN	RING OUT

MAX RUN TIME 1 MIN.
MAX PRINT OUT 50 PAGES
MAX. CARDS OUT _____ CARDS

☐ PICKUP 2ND FLOOR ☒ PICKUP 3RD FLOOR
☐ MAIL TO:

SPECIAL INSTRUCTIONS

NUMBER SCRATCH TAPES NEEDED
☐ TPR OUTPUT
☐ PRINT & RELEASE
☐ PRINT & SAVE

OPERATOR COMMENTS

REEL (DO NOT WRITE)

FORM NBS-112 (8/67) | U.S. DEPARTMENT OF COMMERCE NATIONAL BUREAU OF STANDARDS | PHONE

COMPUTER SERVICES INSTRUCTION CARD

1.5 Numbers In Comments.

Explanatory words or phrases are always allowed in an instruction, i.e.,

```
SQUARE 1,2
```

can be written as

```
SQUARE column 1 and put the results in column 2
```

If numbers are used in comments, additional steps must be taken to avoid having the numbers mistaken for arguments. A number in a comment should always be at the end of the instruction and follow the last argument. If there are only a few such numbers, the simplest procedure is to precede the comment by a dollar sign, $, as in:

```
ADD 1.0 to column 24 and put in column 37 $  Y=X+1
```

The dollar sign is a signal to the computer to ignore all information which follows it. The dollar sign can be used to make an entire card a comment card by putting the dollar sign first, usually in the first card column, as in:

$ the following group of instructions computes the geometric mean.

If numbers in comments occur in many cards, a simpler procedure is to use the instruction SCAN. (See PART C.) The instruction

SCAN only the first (c) card columns on the following cards

instructs the computer to ignore all information on any card that follows beyond the c-th. column.

1.6 NRMAX

NRMAX stands for maximum number of rows. It is the number of rows operated upon by an instruction. It is not the number of rows in the worksheet. Usually the user need have no concern about NRMAX. Its value is usually automatically set by one of the instructions for entering data into the worksheet (READ, SET, GENERATE). In the instructions:

```
OMNITAB
GENERATE 1.(1.)5. in column 1
ADD 1. to column 1 and put in column 2
PRINT columns 1 and 2
STOP
```

the command OMNITAB automatically sets the number of rows in the worksheet equal to 201 and NRMAX equal to zero. The command GENERATE automatically sets NRMAX to equal 5. Hence, when the ADD instruction is executed, only the first five rows are operated on. The numbers 2., 3., 4., 5., and 6. will be put in the first five rows of column 2 and the remaining rows will have the value zero which was set by the command OMNITAB.

The value of NRMAX may be changed, but not necessarily, only by the following instructions:

Increase NRMAX	Decrease NRMAX	Either Increase or Decrease NRMAX
DEMOTE	ERASE (with no arguments)	ISOLATE
DUPLICATE	FREQUENCY (usually)	ISETUP
GAUSS QUADRATURE	OMNITAB (NRMAX=0)	ITERATE
GENERATE	SHORTEN	RESET nrmax
INSERT		
READ		
SET		
CSET TAPE "L"		
READ "L"		
READ TAPE "L"		
READ TAPE "L","L"		
SET TAPE "L"		

1.7 Variables V, W, X, Y and Z.

If a constant, such as 3.7, is to be used repeatedly, it is sometimes convenient to give it a name. This is particularly true if the constant is used often in a set of instructions, but the value of the constant is changed when the set of instructions is used on different days. Then the value of the constant has to be changed in only one place rather than many different places. Five variables are available for this purpose; V, W, X, Y and Z. The value of a variable is set by using the instruction

RESET "V"

where the qualifier "V" can be either V, W, X, Y or Z. For example, we could use

RESET W to 3.7

When a variable is referenced in an instruction, it must be enclosed by asterisks; see section 1.8. The instruction

ADD the value *W* to column 2 and put in column 3

adds 3.7 to each value in column 2 and puts the results in the corresponding rows of column 3.

If more than 5 constants are needed one can put all the constants in a column and then use asterisks as described in section 1.8. Actually, the decimal point is not necessary as the instruction implies that the number after the qualifier has to be a constant, i.e., RESET X to 12 is automatically converted to RESET X to 12.0.

1.8 Use Of Asterisks.

(a) Three asterisks can be used to designate an implied "through". In PRINT 1 *** 8 the three asterisks indicate that we want to use all the integers between 1 and 8. This is merely a shorthand way of writing PRINT 1,2,3,4,5,6,7,8. The numbers on either side of the three asterisks must always be integers. The three asterisks must not be separated. They must not be used to mean "thru" when "thru" is implied by the instruction as in the instructions PERFORM, ROW SUM, and SUM. The number on the left of the three asterisks should be less than or equal to the the number on the right, i.e., PRINT 8 *** 1 should not be used.

(b) Single asterisks enable you to use a number in a particular part of the worksheet without actually knowing its specific value. The instruction

ADD value *2,3* to column 17 and put in column 35

adds the number which is in row 2 of column 3 to every number in column 17 and puts the results in column 35.

The argument defined by a pair of single asterisks must be defined by the following, no more and no less: (i) an asterisk, (ii) (R) = row number, (iii) a comma, (iv) (C) = a column number, and (v) an asterisk. No additional commas, spaces or any other characters may be used.

(c) Double asterisks are used in much the same way that single asterisks are used, except the argument defined is an integer rather than a constant. If the number in the worksheet is not an exact integer, it is truncated to an integer for use in the instruction. For example, if the number 3.7654289 is in row 7 of column 8, then the instruction

ADD column **7,8** to column 16 and put in column 17

would add the numbers in column 3 to the numbers in column 16 and put the results in column 17. The number 3.7654289 is truncated to 3 for defining the argument in this instruction, but the number in row 7 of column 8 in the worksheet is unchanged.

NRMAX and the variables V, W, X, Y and Z may be referenced using either single or double asterisks.

1.9 The Size Of The Worksheet.

The instruction OMNITAB automatically sets the size of the worksheet to be 201 rows by 62 columns. This shape is not always the most desirable. At times all one needs is a simple analysis of several hundred measurements. At other times one needs to perform many computations for a small set of data. In these situations the shape of the worksheet can be easily changed by using the instruction

DIMENSION the worksheet to be (r) rows by (c) columns

The integers (arguments) (r) and (c) can have any positive value as long as the product does not exceed 12,500. When DIMENSION is used, it should immediately follow the OMNITAB instruction or at least precede any executable instruction. It should not be placed in the middle of a set of instructions.

1.10 Automatic Printing.

Several instructions for statistical analysis automatically produce the printing of a comprehensive set of results. In contrast to other statistical programs, the user is not asked to choose between many different options. He gets everything that the instruction produces. Invariably, the exercise of an option implies an a priori judgement of the data which the OMNITAB user is spared. Considerable thought was given to the method of presenting results and to the selection of items to be printed. An effort was made to give non-statisticians, in particular, enough information to use their results thoughtfully. For example, the FIT instruction prints and plots the standardized residuals to enable the user to assess the adequacy of his statistical model. Careful use of the printed results can be educational.

The instructions which have automatic printing are:

STATISTICAL analysis	CORRELATION
FIT	POLYFIT
ONEWAY	TWOWAY
APROPERTIES	MPROPERTIES

Each of these instructions can also be used to store results in the worksheet. The same command preceded by an S, as in SFIT, suppresses the automatic printing and gives only the stored results. The use of S to suppress the automatic printing can be useful if a few of the many results are desired on several sets of data, rather that the full set of results for a single set of data.

1.11 Two-word Commands.

Most instructions have a single command, e.g., MULTIPLY. Some instructions may seem to have two words in the command, as in STATISTICAL analysis, but the second word is used only for clarification and can be omitted. The following commands have two words separated by at least one space. Both words must be used. Except, the command ROWSUM is allowed as a synonym for the command ROW SUM. The magnetic tape operation commands described in section C1.10 are not listed here.

NO LIST	ELLIPTICAL FIRST
CLOSE UP	ELLIPTICAL SECOND
NEW PAGE	ZEROS BJZERO
ROW SUM	ZEROS BJONE
F PROBABILITY	STRUVE ZERO
PAGE PLOT	STRUVE ONE
GAUSS QUADRATURE	
UNIFORM RANDOM	
PRINT NOTE	

If a phrase normally occurs as two words in the English language, e.g. new page, then the OMNITAB command usually has two words, e.g., NEW PAGE. This is not always true in technical expressions, as in NORMLAGUERRE.

1.12 Abbreviations.

Abbreviations are allowed for some of the more commonly used commands. For reference, a complete list is given here. All the commands on the right are array or matrix operation commands.

FULL COMMAND	ABBREVIATION	FULL COMMAND	ABBREVIATION
SUBTRACT	SUB	ASUBTRACT	ASUB
MULTIPLY	MULT	AMULTIPLY	AMULT
DIVIDE	DIV	ADIVIDE	ADIV
ABSOLUTE	ABS	MSUBTRACT	MSUB
EXPONENTIAL	EXP	MMULTIPLY	MMULT
LOGE	LOG		
MAXIMUM	MAX		
MINIMUM	MIN		
DIMENSION	DIM		

Caution: abbreviations are not allowed for the instructions to perform complex arithmetic. E.g., CDIV cannot be used as an abbreviation for CDIVIDE. Only the above abbreviations are allowed.

1.13 Synonyms.

Certain commands have synonyms, particularly if an instruction can be used in more than one context. For example, an array can also be thought of as a matrix. A complete list is given below.

COMMAND	SYNONYM(S)
PERFORM	REPEAT, EXECUTE
ORDER	SORT (C), only one argument
MAXMIN	EXTREMA
MOVE	AMOVE, MMOVE
INVERT	MINVERT
AADD	MADD
ASUBTRACT	MSUBTRACT
AERASE	MERASE, AZERO, MZERO
ATRANSPOSE	MTRANSPOSE
ADEFINE	MDEFINE
APRINT "L"	MPRINT "L"

The list does not include commands where a special case of one instruction may be synonymous with another instruction, as

```
MULTIPLY 1, 1, 2
```

is equivalent to

```
SQUARE 1, 2
```

1.14 Named Constants.

Two mathematical constants which occur frequently in scientific work can be referenced using asterisks without actually writing the number explicitly. The constants are, PI = 3.1415927, the ratio of the circumference of a circle to its diameter and, E = 2.7182818, the base of the natural system of logarithms. The instructions

```
MULTIPLY  *PI* by 0.5 and put in column 2
SQUARE of *E* put in column 3
```

would put 1.5707963 and 7.3890561 into columns 2 and 3. A single asterisk is used directly on each side of the symbol. No character should be used between the asterisks other than the named constant (PI or E).

Similarly, 18 fundamental physical constants can be used by enclosing the appropriate symbol in asterisks. The value of a constant depends upon the system of units in use. The instruction OMNITAB initially sets the value of the constants in SI (Systeme Internationale) units (metric system). The system of units is easily changed by using either of the instructions (with no arguments)

```
CGS       (centimeter-gram-second)
```

or

```
SI        (Systeme Internationale)
```

The OMNITAB symbol, the name and the current value in both SI and CGS units are given for each constant in the following table. SI units were formerly called MKSA units.

OMNITAB	Physical Constant	Value SI Units	CGS Units
ALPHA	Fine structure constant	7.29720E-3	7.29720E-3
C	Speed of light in vacuum	2.997925E+8	2.997925E+10
CONE	First radiation constant	3.7415E-16	3.7415E-5
CTWO	Second radiation constant	1.43879E-2	1.43879
F	Faraday constant	9.64870E+4	9648.70
G	Gravitational constant	6.670E-11	6.670E-8
GAMMA	Gyromagnetic ratio of proton	2.67519E+8	26751.9
H	Plank constant	6.6256E-34	6.6256E-27
K	Boltzmann constant	1.38054E-23	1.38054E-16
ME	Electron rest mass	9.1091E-31	9.1091E-28
MP	Proton rest mass	1.67252E-27	1.67252E-24
MUB	Bohr magneton	9.2732E-24	9.2732E-21
N	Avogadro constant	6.02252E+23	6.02252E+23
Q	Elementary charge	1.60210E-19	1.60210E-20
QME	Charge to mass ratio for electron	1.758796E+11	17587960.
R	Gas constant	8.3143	8.3143E+7
RINF	Rydberg constant	10973731.	109737.31
SIGMA	Stephan-Boltzmann constant	5.6697E-8	5.6697E-5

For further information on the use of fundamental physical constants see ASTM Metric Practice Guide, Second Edition, NBS Handbook 102, U. S. Government Printing Office (1966). Values of the constants are given on pages 42 and 43.

1.15 Characters Recognized.

The OMNITAB computing system recognizes and uses the letters A through Z, the digits 0 through 9 and the 12 special characters:

```
=          +          -          .          ,          *
(blank)    /          (          )          '          $
```

All other characters, such as the one given by a multiple 11-8-2 punch, are made equivalent to $. See section 1.5 for possible effects.

1.16 Commands With Qualifiers.

Some commands have a qualifier denoted by "L", where "L" indicates any one of the letters A, B, C, D, E or F. These qualifiers are used to distinguish between formats and tapes (except in DUMMY "L"). The qualifier (without quotation marks) is part of the command. One blank space must precede and follow the qualifier without any additional characters. All of the magnetic tape operation commands have either one or two qualifiers. One instruction, RESET "V", has the qualifier "V", where "V" denotes the letter V, W, X, Y or Z. The commands with qualifiers are:

FORMAT "L"
PRINT "L"
PUNCH "L"
READ "L"
ABRIDGE "L"
NPRINT "L"
APRINT "L"
MPRINT "L"
RESET "V"
DUMMY "L"

READ TAPE "L"
CREAD TAPE "L"
WRITE TAPE "L"
SET TAPE "L"
CSET TAPE "L"
ENDFILE TAPE "L"
SKIP TAPE "L"
BACKSPACE TAPE "L"
REWIND TAPE "L"

READ TAPE "L" "L"
CREAD TAPE "L" "L"
WRITE TAPE "L" "L"

2. REPEATED USE OF COMMANDS.

Ordinarily, each instruction is executed once and only once as soon as it is encountered. Often, an instruction has to be repeated several times and it would be cumbersome to have to write out an instruction each time. For example, to compute the average value of the numbers in each of the nine columns 11 through 19 and put the results in columns 21 through 29, it would be annoying to write

```
AVERAGE 11,21
AVERAGE 12,22
AVERAGE 13,23
AVERAGE 14,24
AVERAGE 15,25
AVERAGE 16,26
AVERAGE 17,27
AVERAGE 18,28
AVERAGE 19,29
```

If we wanted to do something similar 100 times rather than 9 the task would be formidable. A much simpler procedure exists which involves storing, or saving, instructions for repeated use.

Instructions which are to be repeated are saved for later use. They are not executed when encountered but are only executed when the appropriate instruction is given. The command of a stored instruction must be preceded by a valid instruction number and a slash, /, as in

```
1/ AVERAGE 11,21
```

A stored instruction would have limited utility if there were not also an easy way of modifying the instruction. In our example, the second time the instruction AVERAGE is used the arguments have to be changed from 11 and 21 to 12 and 22. Modifying instructions is accomplished by using the INCREMENT instruction. We now have

```
1/ AVERAGE the values in column 11 and put in col 12
2/ INCREMENT instruction 1 by    1                  1
```

These instructions will be stored but no computing will result. To execute the instructions and compute we must use the instruction PERFORM, indicating which instructions are to be executed and how often. Thus, our complete set of instructions is

```
1/ AVERAGE the values in column 11 and put in col 12
2/ INCREMENT instruction 1 by    1        and       1
      PERFORM instructions 1 thru 2, 9 times
```

General information for the use of stored instructions is given in the sections which follow. Some of the information on specific instructions is repeated in PART C. A complete discussion of error diagnostics, and in particular of FATAL ERRORS, is given in section 3. Several of them pertain to the repeated use of the commands and these are given in the following discussions. In the REPEAT mode (repeated use of instructions) the error message comes after the PERFORM instruction and also gives the instruction (statement) number and the time, or cycle, through the stored instructions in which the error occurred. A complete message is shown in the following example:

```
OMNITAB  8/18/70
GENERATE 1.(1.)100. into column 1
1/ PROMOTE              25 rows, col 1 into col 11
2/ INCREMENT instr 1 by 20,          0,          1
PERFORM instructions 1 thru 2, 7 times

* INFORMATIVE DIAGNOSTIC IN ABOVE COMMAND
* ATTEMPT TO PROMOTE FROM BELOW NRMAX. FIRST ARGUMENT IS RESET TO NRMAX.
IN COMMAND AT STATEMENT NUMBER 1.0
CYCLE NO.   5 OF    7 OF EXTERNAL PERFORM STATEMENT.
```

2.1 Numbering Instructions.

An instruction number is a number between 0.1 and 999.9 inclusive. Normally, integers are used without a decimal point. One digit is allowed after a decimal point. If the number is less than 1 a zero must precede the decimal point, e.g., 0.6 is a valid instruction number, but .6 is not. An instruction number which is an exact integer must not be written with a decimal point unless a zero follows the decimal point. E.g., 28 and 28.0 are valid instruction numbers, but 28. is not. Any statement number which does not conform to these rules produces the FATAL ERROR

*** ILLEGAL STATEMENT NUMBER

Decimal points in an instruction number are usually used only when instructions are added as an afterthought. To insert an ADD instruction between instructions 23/ and 24/ we might for example use

23.5/ ADD column 3 to column 4 and put in column 5

and avoid having to renumber all the instructions which follow.

A numbered instruction must not immediately follow a data card. If one wants to put numbered instructions right after data cards, the first numbered instruction should be preceded by the command NULL. Also, a numbered instruction must precede the PERFORM instruction which executes it.

An instruction number can be used more than once in a set of instructions. After a stored instruction has been used, i.e., executed by a PERFORM instruction, it is sometimes preferable to reuse the number in the work that follows rather than to use a new number. See, also, section 2.3. A numbered instruction is saved, hence when a number is reused the new instruction replaces the previous one. The previous instruction is now destroyed and the new one is saved. The instruction OMNITAB automatically removes all previously saved instructions.

Any unnumbered instruction which appears between two numbered instructions is executed at once as usual. The same result would be obtained if the unnumbered instruction appeared just before or just after the numbered instructions. For reasons discussed below, the command PERFORM is an exception to this rule.

One or more blank spaces can be used on either side of the slash following an instruction number.

Instructions are executed in the same order they are numbered regardless of their physical order. If ten cards with instruction numbers 1, 2 ... 10 were shuffled and then put back in the set of instructions, the results would be exactly the same as those obtained had the cards been supplied in order.

There is a limit to the number of instructions which can be stored in the computer. The limit is rarely exceeded, but when it is the following FATAL ERROR results

*** COMMAND STORAGE AREA OVERFLOW

A precise statement cannot easily be made about the size of this limit. It depends on the number of arguments in each of the stored instructions. The number is roughly 250.

2.2 Use Of PERFORM.

The instruction PERFORM causes the execution of all stored instructions with numbers between the first and second arguments, inclusive. The third argument indicates the number of times the instructions are to be executed. The PERFORM instruction must follow the stored instructions which are to be executed. The commands EXECUTE and REPEAT are identical in every way to PERFORM.

PERFORM usually has three arguments, but may have only two or one. The 1st argument must be less than or equal to the 2nd argument. If the third argument is missing it is assumed to be 1. Thus,

PERFORM instructions 11 thru 17, 1 time

and

PERFORM instructions 11 thru 17

are equivalent. If both the second and third arguments are missing, the third argument is assumed to be 1 and the second argument is assumed to be the same as the first. Thus,

PERFORM 37

is equivalent to

PERFORM 37 through 37

and is also equivalent to

PERFORM instructions 37 thru 37, 1 time

Note that in the above examples the first instruction number is not 1. The instruction numbers in a PERFORM instruction can be any valid instruction number providing it has been used. The first argument does not have to correspond with the first numbered instruction and the second argument does not have to correspond with last numbered instruction. Any instruction number that is referred to must exist. In the above example if no instruction numbered 37 exists, the following FATAL ERROR will occur

*** STATEMENT NUMBER NOT FOUND

PERFORM can be a stored instruction to be executed by another PERFORM instruction. But an instruction cannot repeat itself. The instruction

17/ PERFORM 15 thru 20, 5 times

would produce the FATAL ERROR

*** STORED PERFORM STATEMENT WILL EXECUTE ITSELF

Caution: Do not make the common mistake of using *** in place of thru. The instruction

PERFORM 1 *** 5, 15 times

is interpreted as

PERFORM 1,2,3,4,5, 15 times

and would produce a FATAL ERROR because there is an illegal number of arguments.

When PERFORM is a numbered instruction we refer to it as a PERFORM within a PERFORM. The maximum number of PERFORM commands allowed within a PERFORM is eight.

Care should be exercised in using PERFORM as a numbered instruction. Following the instructions

```
1/ .............
2/ .............
3/ PERFORM 1,2,5
4/ .............
5/ .............
```

the instruction

```
PERFORM 1 thru 5, 10 times
```

may possibly be correct. However, often it is used mistakenly in place of

```
PERFORM 3 thru 5, 10 times
```

2.3 Use Of INCREMENT.

The INCREMENT instruction has exactly one more argument than the instruction that is to be modified. The extra argument is the first one which is the instruction number of the modified instruction. All arguments must agree in kind with the instruction referred to. If a decimal point is used in an argument of an instruction, a decimal point must be used in the corresponding argument in the INCREMENT instruction. Similarly, if a decimal point is not used, then a decimal point must not be used in the corresponding argument of the INCREMENT instruction. For the instruction

```
27/ ADD  the value  1.0 to column 2 and put in column 3
```

the instruction

```
28/ INCREMENT 27 by 3.0          0                  1
```

is correct, but the instructions

```
28/ INCREMENT 27 by 1            0                  1
```

and

```
28/ INCREMENT 27 by 1.0          1.0                1
```

are both incorrect.

An INCREMENT instruction can be incremented by another INCREMENT instruction. If it is, care should be used. Naturally, an INCREMENT instruction cannot increment itself, if such an attempt is made the following FATAL ERROR will occur:

```
*** AN INCREMENT COMMAND CAN NOT INCREMENT ITSELF
```

If asterisks are used to define an argument (see section 1.8) the INCREMENT instruction must also contain asterisks. The instruction

```
18/ ADD the value defined by *1,2* to col 11 and put in col 12
```

adds the number in row 1 of column 2 to every number in column 11 and puts the result in column 12. The defining row and column numbers in the argument *1,2* can be incremented individually using asterisks as in

```
19/ INCREMENT instr 18 by    *1,0*        0                  1
```

Do not make the mistake of thinking that because *1,2* represents a number, that instruction 18 could be incremented by a number as in

```
19/ INCREMENT instr 18 by     3.7         0                  1
```

This instruction would produce a fatal error because the argument is improper.

The INCREMENT instruction can be used to increment instructions which contain NRMAX or one of the variables enclosed in either single or double asterisks. But the variable itself cannot be incremented this way. In the INCREMENT instruction, 0.0, without asterisks, should be used in the corresponding argument for NRMAX or "V". A decimal point must always be used with zero, even if the argument which is being incremented is an integer; as in **V**. The instruction

```
35/ DIVIDE  column 27 by *NRMAX* and put in col 28
```

could be incremented by

```
36/ INCREMENT 35 by 1       0.0                    1
```

An instruction containing triple asterisks to imply through can also be incremented, but care should be used. The instructions

```
31/ PRINT the numbers in columns 1 ***  5
32/ INCREMENT instr 31 by       10 *** 11
      PERFORM 31 thru 32, 2 times
```

would print the contents of columns 1,2,3,4,5 and 11,12,13,14,15 and 16. The two numbers on either side of the asterisks do not have to agree.

Some care should be exercised in choosing the location of an INCREMENT instruction. The usual and best practice is to have the INCREMENT instruction follow the instruction that is being incremented as in the above examples. In the opening example on page 24

```
1/ INCREMENT instruction 2 by    1                    1
2/ AVERAGE the values in column 11 and put in column 12
       PERFORM instructions 1 thru 2, 9 times
```

would work, but would yield the wrong answers. If the instructions were to be used this way, instruction 2 should read

```
2/ AVERAGE the values in column 10 and put in column 11
```

In this connection, note that

```
13/ SQUARE column 0 and put in column 1
```

is not necessarily incorrect. Although 0 is an illegal column number, the instruction would be valid if 0 were incremented correctly prior to the execution of instruction 13.

After an instruction has been incremented, it is often desirable to restore the instruction to its original form for further use. This may happen, for example, if a set of operations is used for several sets of data or when one has a PERFORM within a PERFORM. It can be done in three different ways. The INCREMENT instruction can be used, possibly with negative arguments, but this method is sometimes tricky and is prone to errors. One can simply rewrite the numbered instruction in its original form (or any other form for that matter). This wipes out the instruction you had and replaces it with the new one. The third method is to use the instruction RESTORE. The instruction RESTORE does not have to be stored (numbered). Suppose we want to read into the worksheet eight columns of data to find the averages and to repeat this 15 times. Then, we could use the following instructions:

```
                1/ AVERAGE      column 11 put in column 21
                2/ INCREMENT instr 1 by 1                1
                3/ PERFORM instrs 1 to 2, 8 times
                4/ ABRIDGE row 1 of columns 21 *** 28
                5/ RESTORE instr 1 to  11               21
        READ data into columns 11 *** 18
                ( data cards )
        PERFORM instrs 3 to 5
        READ data into columns 11 *** 18
                ( data cards )
        PERFORM instrs 3 thru 5
                ..... etc. ( repeallast three lines 12 times )
```

An exception exists for instructions for using magnetic tapes. The qualifier of a tape operation instruction, which refers to a tape number, must be incremented even though it is not an argument. This is true for tape operation instructions only. If the command has two qualifiers, the first qualifier refers to a tape and the second to a format. Only the first qualifier should be incremented. Usually the number zero is used in the INCREMENT instruction to increment the tape number. See PART C for further details and, in particular, non-NBS users should verify that the magnetic tape operation instructions operate with their computer as described herein.

2.4 Instructions Which Must Be Stored.

Most instructions can be either stored or not stored. Because of their form, the following instructions must always be stored for later use:

```
        COMPARE     ISOLATE      ITERATE
IFEQ        IFNE        IFGE        IFGT        IFLE        IFLT
```

Actually, ISOLATE and ITERATE do not have to be stored, but in almost all applications they should be stored.

When an instruction which must be stored appears without an instruction number, the following FATAL ERROR occurs

*** COMMAND MUST BE STORED

The instruction NULL does not have to be stored, but its chief use is as a stored instruction to wipe out an instruction that is no longer wanted.

2.5 Instructions Which Cannot Be Stored.

Most instructions can be stored for repeated execution. Because of their form, the following instructions must not be stored:

```
FORMAT "L"      HEAD        NOTE        NOTE1       NOTE2       OMNITAB     STOP        READ        SET
TITLE1          TITLE2      TITLE3      TITLE4      TITLEX      TITLEY      BEGIN       FINISH
```

If the command BEGIN is stored (preceded by an instruction number), the following FATAL ERROR will occur:

*** COMMAND NOT ALLOWED IN THE REPEAT MODE

If any of the commands FORMAT, NOTE, TITLE or HEAD are stored, the following INFORMATIVE DIAGNOSTIC (see section 3.4) will be given:

* COMMAND NOT ALLOWED IN THE REPEAT MODE. EXECUTED BUT NOT STORED

No diagnostic is given if either READ, SET or FINISH is stored.

Although NOTE1 and NOTE2 cannot be stored, the instruction PRINT NOTE can be stored and it is PRINT NOTE which actually executes NOTE1 and NOTE2. Also, although READ cannot be stored, the command READ "L" format can be stored. If the command READ "L" is stored, the data should immediately follow the external PERFORM instruction which executes READ "L". The data should not immediately follow the READ "L" command as would be the case if the command were not stored. External means that the PERFORM command is not stored as opposed to an internal (with respect to the REPEAT mode) PERFORM which is stored. READ "L" can be used effectively as a stored instruction when it is necessary to perform the same set of operations on several different sets of data, but extreme care should be exercised. If a FATAL ERROR occurs before the READ "L" instruction is executed, the external PERFORM instruction will not execute the READ "L" instruction. As a severe consequence, the data which follow the external PERFORM will be treated as instructions rather than data and FATAL ERRORS will result. If there is a large number of data cards, a large number of FATAL ERRORS will result which also can consume a lot of computer time. Consequently, every instruction should be very carefully checked to ensure that it is correct. It may even be advisable to make a trial run without data just to be sure there are no FATAL ERRORS.

When identical operations are performed on different sets of data, each set is preceded by an appropriate READ instruction and then followed by the appropriate PERFORM instruction.

2.6 Use Of BEGIN And FINISH.

The technique of using BEGIN and FINISH is retained from the original OMNITAB. Although it is simple, it has several disadvantages. If an instruction is added later, the numbering sequence will change and all the INCREMENT instructions may have to be changed accordingly. An instruction which is not to be stored cannot be used between BEGIN and FINISH.

The method of numbering instructions directly can be avoided by simply using the instruction

BEGIN storing instructions

immediately before the first instruction to be stored and using the instruction

FINISH storing instructions

immediately after the last instruction to be stored.

Instructions are automatically numbered by the computer starting with 1 and proceeding in steps of 1 until the last instruction, before FINISH, has been numbered. In the LIST OF COMMANDS, the instruction number assigned by the computer is listed on the extreme left.

If you want the instructions numbered starting with some number other than 1, use the instruction

BEGIN storing instructions starting with number (N)

This instruction should be used if BEGIN is used more than once in the same set of instructions and earlier instructions are not to be wiped out. The instructions

BEGIN

and

BEGIN 1

are equivalent.

Manually numbered instructions can be used in conjunction with BEGIN and FINISH. But they must not be placed between BEGIN and FINISH or the FATAL ERROR

*** STATEMENT NUMBER MAY NOT BEGIN ANY CARD BETWEEN BEGIN AND FINISH CARDS

will result. Any instruction numbered manually automatically wipes out the instruction which has been given the same number by the computer. The instructions

```
BEGIN
ADD              1.0   1.0   11
INCREMENT 1 by 0.0   1.0    1
FINISH
PERFORM 1,2,4
1/ MULTIPLY 2.0, 1.0, 15
PERFORM 1,2,4
```

will put the numbers 2,3,4,5 and 2,4,6,8 into columns 11 through 18.

If, in the above example, we were to replace

```
1/ MULTIPLY 2.0, 1.0, 15
```

by

```
BEGIN
MULTIPLY 2.0, 1.0, 15
FINISH
```

the original ADD instruction would be deleted as soon as the second BEGIN was encountered, but not before the first PERFORM had been executed. The INCREMENT instruction, instruction numbered 2, would remain intact as only one instruction had been used after the second BEGIN.

2.7 Branching.

Branching (or logical branching) is a term used to describe a point in a set of instructions where one group of instructions is executed if a certain condition is true and a different set of instructions is executed if the condition is false. Branching is common in most computer languages such as FORTRAN. In OMNITAB, branching is seldom necessary. It primarily occurs in iterative procedures used in non-linear least squares, inverse interpolation, finding the values of an inverse function, etc. The seven instructions used in branching

COMPARE IFEQ IFNE IFGE IFGT IFLE IFLT

are described in PART C. Of these instructions, COMPARE is probably the one most often used. Note, all branching instructions must be stored.

2.8 Additional Comments.

Earlier it was explained how to find the average value of the numbers in several columns using stored instructions or the REPEAT MODE. Notice that it was necessary to have the original data in consecutive columns (or have adjacent column numbers differ by a constant amount); similarly for the results. This type of procedure is not always necessary. Clearly, most of the instructions for entering and printing data, such as READ and PRINT, will operate on several different columns whose numbers are not necessarily evenly spaced (or even monotonic). In addition, some of the manipulative and arithmetic instructions can perform operations on several different columns. For example,

FLIP column 11 into 2, 3 into 17, 8 into 5, and 31 into 12

The following is a list of such commands:

CHANGE	CLOSE UP	DEMOTE	ERASE
EXCHANGE	FLIP	ORDER	PRODUCT
PROMOTE	ROW SUM	SORT	

Care should be exercised in the use of these instructions if any column number is used more than once. For example, the instruction

FLIP column 11 into column 12 and column 12 into column 13

is valid, but all it does in effect is move column 11 into column 13.

The more experienced user would do well to note that the array and matrix operation instructions can be used very effectively for data manipulation without resorting to the use of the REPEAT mode. Matrix operation instructions can be used to perform operations usually associated with matrix algebra (MINVERT), but are perhaps more often used for data manipulation. As an illustrative example solely, and not particularly useful in this case, consider the earlier problem of finding the averages for columns 11 through 19. If NRMAX = 50, we could use the following set of instructions:

```
DEFINE 1.0 into column 20
M(V'A) 1,11 size 50x9 by column 20, store 1,21
ADIVIDE the array in 1,21 by 50.0 and put in 1,21
DUPLICATE 50 times the array in 1,21 size 1,9 put in 1,21
```

The last instruction would only be desirable, or necessary, if further calculations were to be performed on the averages. Note, **NRMAX** could be used for 50, if NRMAX was unknown; and instead of 50.0 we could use *NRMAX* (see section 1.8).

A good use of the PERFORM instruction is in the generation of ad hoc subprograms or subroutines as they are usually called. It is not uncommon to write a set of instructions which can be used as a subprogram or may be used in different sets of instructions at different times by different people. Often the subprogram takes the place of an instruction that does not exist. There is no instruction to compute the harmonic mean of a column of numbers. If there were a great demand for the instruction, the command HARMONIC MEAN might be added to the vocabulary, But until such time, cards could be punched for the following instructions

```
501/ DIVIDE 1.0 by column 1 and put in column 2
502/ AVERAGE column 2 and put in column 2
503/ DIVIDE 1.0 by column 2 and put in column 2
```

These cards could be kept and every time we want the harmonic mean of column 1 put in column 2 we would use the cards with

PERFORM instructions 501 thru 503

The subroutine above can be made more general by using **X** and **Y** in place of column 1 and column 2. The variables X and Y could then be RESET to the desired value each time the subroutine is used. This procedure is particular useful if the subroutine is lengthy or complex. With a little extra effort and cooperation among colleagues, a small local subroutine library can be maintained for special purposes. The NBS Statistical Engineering Laboratory maintains a small library for its own use.

3. DIAGNOSTIC FEATURES AND ACCURACY

3.1 Diagnostic Features.

After a set of instructions has been executed, the master program prints the

LIST OF COMMANDS, DATA AND DIAGNOSTICS

subject to certain restraints which may be imposed by the user. See LIST, NO LIST and READ "L" in PART C. Each card that is processed is listed and any errors in an instruction that have been detected are listed just below the instruction.

The simplicity of OMNITAB and the natural structure of the language make it quite possible to write a set of instructions free of errors. But when a set of instructions is written in haste or if the set is complicated, the user may make some errors. The diagnostics which are printed make it easy to spot an error and correct it. Considerable care and attention has been given to the proper detection and identification of errors by the master program.

There are three levels of errors and diagnostic messages: (i) fatal errors, (ii) arithmetic faults and (iii) informative diagnostics. An example of each is given in the following example.

```
OMNITAB   6/12/70  EXAMPLE OF ERROR DIAGNOSTICS

LIST OF COMMANDS, DATA AND DIAGNOSTICS

    TITLE7  PLOT OF LOG OF X

* INFORMATIVE DIAGNOSTIC IN ABOVE COMMAND
* IMPROPER TITLE NUMBER, ASSUMED 1

    GENERATE 0.0(.01)1.0 IN COLUMN 1
    LOGTEN OF COLUMN 1 PUT IN COLUMN 2

** ARITHMETIC FAULT IN ABOVE COMMAND, ZERO RETURNED   1 TIMES
** NEGATIVE ARGUMENT TO SQRT, LOG OR RAISE

    PLOT COLUMN 2 VERSUS COLUMN 1
    SQUARE 2,3
    INVERT 1,1 SIZE 3,3 STORE 1,4
      ++++++++++++++++++++ SMALLEST ERROR BOUND ON INVERTED MATRIX IS   .5-03   +++
    PRINT COLUMNS 4, 5 AND 6

*** FATAL ERROR IN ABOVE COMMAND
*** NAME NOT FOUND IN LIBRARY

                    ONLY ONE FATAL ERROR
```

A single asterisk is printed at the extreme left of an informative diagnostic, two asterisks for an arithmetic fault, and three asterisks for a fatal error message. In addition, an error bound follows a series of +'s after the command INVERT as shown above. Each type of error message is discussed separately in the next three sections.

A fatal error in an instruction results in a failure to execute that instruction and all subsequent instructions. However, the execution of instructions is not affected by arithmetic faults or informative diagnostics. The guiding principle is that computation should be allowed to continue as long as at least part of the results are likely to be useful. (The command NOTE is an exception; see PART C for details.)

3.2 Fatal Errors.

A FATAL ERROR occurs if an instruction is incorrectly written and the consequences would seriously affect the results of executing subsequent instructions. When a FATAL ERROR occurs in an instruction, that instruction and all subsequent instructions are not executed. However, the remaining instructions are examined to detect any other errors that can be found simply by reading the instruction. Errors which can only be detected when the instruction is executed will not be found if a FATAL ERROR has already occurred.

If a FATAL ERROR is detected in a stored (numbered) instruction, the remaining stored instructions are not checked for errors. This is worth noting for the correction of a FATAL ERROR in a stored instruction does not necessarily mean that there are no more errors. The user should double check the remaining stored instructions to assure that they are correctly written.

If a FATAL ERROR occurs in a set of instructions, the number of FATAL ERRORS is given at the end of the LIST OF COMMANDS, DATA AND DIAGNOSTICS as in the example of section 3.1. The notation (n) is used to indicate a number which is printed by an error message.

The following are the most common FATAL ERRORS and can occur in many different ways:

```
*** NAME NOT FOUND IN LIBRARY
*** ILLEGAL ARGUMENT ON CARD
*** NRMAX=0
*** (n) IS AN ILLEGAL NUMBER OF ARGUMENTS
*** COLUMN NUMBER TOO BIG OR LESS THAN 1
*** IMPROPER TYPE OF ARGUMENT
```

The following FATAL ERRORS can occur in the use of stored instructions. See section 2 and PART C for further details.

```
*** ILLEGAL STATEMENT NUMBER
*** COMMAND NOT ALLOWED IN THE REPEAT MODE
*** STATEMENT NUMBER MAY NOT BEGIN ANY CARD BETWEEN BEGIN AND FINISH CARDS
*** COMMAND STORAGE AREA OVERFLOW
*** STATEMENT NUMBER NOT FOUND
*** STORED PERFORM STATEMENT WILL EXECUTE ITSELF
*** COMMAND MUST BE STORED
*** AN INCREMENT COMMAND CAN NOT INCREMENT ITSELF
*** ILLEGAL *STATEMENT*
```

The following FATAL ERRORS can only occur in a particular instruction or group of instructions:

```
*** DIMENSIONED AREA EXCEEDS LIMIT
*** ILLEGAL SIZE ROW NUMBER
*** DEFINED MATRIX OVERFLOWS WORKSHEET
*** INTEGER ARGUMENT LESS THAN -8191
*** MATRIX IS (NEARLY) SINGULAR
*** INSUFFICIENT SCRATCH AREA
*** DEGREE IS LARGER THAN NO. OF NON-ZERO WEIGHTS
*** NEGATIVE WEIGHTS MAY NOT BE USED
*** NUMBER OF COLUMNS IS GREATER THAN NUMBER OF ROWS
```

```
*** FORMAT NOT FOUND
*** INCORRECT TAPE UNIT, COMMAND IS NOT EXECUTED
*** NUMBER OF ARGUMENTS SHOULD BE (n)
*** MATRIX IS NOT SYMMETRIC
```

3.3 Arithmetic Faults.

An arithmetic fault occurs in an attempt to perform an operation which is normally undefined. For example, division by zero is not defined mathematically. In OMNITAB there are nine different types of arithmetic faults which cause the printing of a diagnostic message. Arithmetic errors may result from (i) accidental use by a user of an improper value, (ii) intentional use (see section 4.6), or (iii) an occasional occurrence of an improper number in calculations performed internally by an instruction.

Below each instruction which produces an arithmetic fault a diagnostic message is printed showing the number of times the error occurred. In each case the functional value is set equal to zero and the computation continues. Often, zero is the result desired.

In particular, the result of dividing any number by zero is defined to be zero. This is always true for instructions which are directly concerned with division such as DIVIDE, ADIVIDE and CDIVIDE.

Division by zero may accidentally occur somewhere in the middle of a set of calculations controlled by an instruction where the division operation is not obviously indicated, as in FIT. Here, most computers, including the NBS computer, will set the result equal to zero. But a few computers give an error termination and all computation ceases. Where it is known that division by zero may occur, steps have been taken to avoid an error termination, but this problem has not been eliminated completely. An error termination will occur only rarely.

Arithmetic errors which appear to be beyond the user's control can occur in instructions like FIT and INVERT. Sometimes the errors are of little significance and can be essentially ignored. At other times it is an indication of serious difficulty. The difficulty may be inherent, but sometimes it can be removed by a reformulation of the problem. For example, a FIT using a set of vectors which are almost dependent may produce serious round-off errors which can be eliminated by using a different model having a set of vectors which are more independent.

To avoid the excessive printing of diagnostics, a tally is kept. After the first 100 arithmetic faults have been found, the following message is printed:

```
*   100 INFORMATIVE AND ARITHMETIC DIAGNOSTICS HAVE BEEN ENCOUNTERED.
*   ANY SUCH ADDITIONAL DIAGNOSTICS FOR THIS COMMAND OR REPEAT MODE ARE DISREGARDED.
```

The most common arithmetic faults occur in division by zero, taking the square root of a negative number and in attempting to take the logarithm of a non-positive number. The folowing are the nine possible arithmetic faults which can occur:

```
** NEGATIVE ARGUMENT TO SQRT,LOG OR RAISE
** EVALUATION OF EXPONENT PRODUCES OVERFLOW
** ARGUMENT OUT OF BOUNDS TO INVERSE FUNCTION
** ARGUMENT TOO LARGE FOR SIN OR COS, ZERO RETURNED (n) TIMES
** BESSEL ARGUMENTS SCALED TO AVOID OVER/UNDER FLOW. RETURNED (n) TIMES
** DIVISION BY ZERO, RESULT SET=0, (n) TIMES
** TRIG FUNCTION NOT DEFINED RESULTS SET=0 (n) TIMES
** ONE OF THE VALUES COMPARED IS ZERO, ABSOLUTE TOLERANCE WAS USED (n) TIMES
** X FOR ELLIPTICAL INTEGRALS IS = 1.0 OR GREATER. RESULT IS SET TO 0.0 (n) TIMES
```

3.4 Informative Diagnostics.

Individual instructions may impose certain restrictions which when violated produce the printing of an informative diagnostic. An appropriate adjustment is made by the master program and computation continues. As an example consider:

```
      OMNITAB
      GENERATE 1.(1.)300. IN COLUMN 1

* INFORMATIVE DIAGNOSTIC IN ABOVE COMMAND
* TOO MUCH DATA IN SET, READ OR GENERATE, SPILL LOST
```

Either the user forgot to put in a DIMENSION instruction or the 3 was incorrectly punched for a 2, say.

An informative diagnostic is given rather than a fatal error on the basis that the results may be at least partially useful, although not necessarily. Often the error is of minor or no significance. But the error can be serious and the user should carefully check the importance of the diagnostic.

As was true for arithmetic faults, see section 3.3, after the first 100 informative diagnostics occur, the following message is printed:

```
*   100 INFORMATIVE AND ARITHMETIC DIAGNOSTICS HAVE BEEN ENCOUNTERED.
*   ANY SUCH ADDITIONAL DIAGNOSTICS FOR THIS COMMAND OR REPEAT MODE ARE DISREGARDED.
```

In most cases, a specific informative diagnostic can only result from the use of one particular instruction or group of instructions. A complete list of the informative diagnostics which are possible is given below.

```
* TOO MUCH DATA IN SET, READ OR GENERATE, SPILL LOST
* COMMAND NOT ALLOWED IN REPEAT MODE. EXECUTED BUT NOT STORED
* VALUE REQUESTED IN SHORTEN, ACOALESCE OR AAVERAGE NOT FOUND.
* BAD HEAD. COLUMN GT 50 OR NO /
* THIS COMMAND WAS NOT EXECUTED BECAUSE ITS MEANING WAS QUESTIONABLE
* F LESS THAN 0, SET = 0
* NU1 OR NU2 LESS THAN 1
* NU1 OR NU2 TRUNCATED TO INTEGER
* IMPROPER TITLE NUMBER, ASSUMED 1
* NO OF ROWS NOT = TO COLS. MATRIX USED LARGEST SQUARE
* ASTERISK STRING IMPLYING 'THRU' INCORRECT, IGNORED
* UNNECESSARY ARGUMENTS IN COMMAND IGNORED
* PARTIAL STORAGE OF MATRIX
* INSUFFICIENT SCRATCH AREA
* NRMAX IS NOT LARGE ENOUGH TO ALLOW ITERATION
* 1ST COLUMN OF ISETUP OR ISOLATE IS NOT MONOTONIC OR IS CONSTANT
* ITERATION HAS FOUND NO VALUES
* WORKSHEET IS TOO SHORT TO ACCOMMODATE ALL THE VALUES GENERATED BY THIS COMMAND.
* MAXMIN HAS FOUND NO EXTREMA
* MAXMIN HAS FOUND AND IGNORED A TRIAD OF X'S WITH AT LEAST TWO IDENTICAL VALUES.
* MORE THAN ONE ARGUMENT IN COMMAND. ONLY FIRST ONE IS USED
* FORMAT NOT FOUND. READABLE FORMAT IS USED
* ONE, SOME OR ALL WEIGHTS ARE NEGATIVE
* ALL WEIGHTS ARE ZERO. COMMAND IS NOT EXECUTED
* ARG FOR BESIN,BESJN,BESKN GIVES A RESULT TOO LARGE/SMALL. COMMAND NOT EXECUTED.
* COLUMN NOT LONG ENOUGH TO STORE ALL ELEMENTS. ONLY NROW WILL BE STORED.
* NOT ENOUGH DATA ON COL TO RESTORE MATRIX/ARRAY. DATA AVAILABLE WILL BE USED.
* SUM OF SQRS DO NOT ADD UP-ABS. VALUE OF (TOTAL-ROW-COL-RES.)/TOTAL EXCEEDS 5.E-7)
* MORE THAN 50 HEAD COLUMN COMMANDS HAVE BEEN USED.
* ATTEMPT TO PROMOTE FROM BELOW NRMAX. FIRST ARGUMENT IS RESET TO NRMAX.
```

* ATTEMPT TO DEMOTE OFF THE WORKSHEET. SPILL IS LOST.
* NEGATIVE VALUE(S) WERE ENCOUNTERED BY PARTITION FUNCTION. ZEROES STORED.
* NEGATIVE ABSOLUTE TEMPERATURES CONVERTED.
* CAUTION, USE EXPERIMENTALLY ONLY. NOT OPTIMUM IN ORDER TO MAKE IT MACHINE INDEPENDENT.
  REFERENCES - J.B.KRUSKAL, ACM,12,92. AND J.H. HALTON,SIAM REV.,12,1.
* COMMAND IGNORED - S BEFORE COMMAND NAME MEANINGLESS IF NO STORAGE REQUESTED.
* NUMBER OF SIGNIFICANT DIGITS AFTER DECIMAL PT. HAS BEEN SET TO (n)

3.5 Accuracy In The Use Of Instructions.

Each of two or more instructions can give fully accurate results, but when used together inaccuracies can occur. Consider

DIVIDE 2. by 3. and put in column 1
DIVIDE 1. by 3. and put in column 2
SUBTRACT column 2 from column 1 and put in column 3

The results correct to 8 digits in columns 1 and 2 are 0.66666667 and 0.33333333. The result in column 3, which is supposed to be 1/3. is 0.33333334. An error of one has been introduced in the eighth digit in a trivial operation. This example is an extreme over-simplification of situations which can cause serious problems. A main source of error in computing is in the subtraction of two quantities which are almost equal. The operation

$$1965.3289 - 1965.3276 = 0.0013$$

starts with two numbers having 8 significant digits and yields a result which is only accurate to two significant digits.

Small errors should not be dismissed lightly. The essential charateristic of a computer is its ability to perform millions of calculations in a small time. The small errors can accumulate into very large errors quickly. This is dramatically illustrated in the following example taken from "Topics in the investigation of linear relations fitted by the method of least squares." F. J. Anscombe, J. R. Statist. Soc., B, 29, 1-29 (1967).

The so-called computational formula for the variance of a set of measurements appears in most elementary books on statistics and statistical computing. If the formula

$$\frac{\sum_{i=1}^{n} x_i^2 - (\sum x_i)^2/n}{(n-1)}$$

is applied to the measurements

9000, 9001 and 9003

the result is in error in the first digit. But if the definition of the variance

$$\sum_{i=1}^{n} (x_i - \bar{x})^2/(n-1)$$

is used, the result is correct to 8 digits. This example shows how easy it is to get inaccurate results with a poor formula and at the same time demonstrates how accurate results can be achieved by choosing the appropriate algorithm (formula).

The novice, who is familiar with the use of a desk calculator, should realize that computational difficulties that arise in hand calculations may easily be spotted and corrected. But when the same calculations are done using a computer, the user may only see the final result and errors may go undetected.

In the days before the computer was invented, people went to some length to provide excellent cross checks on computations. For some reason, these cross checks are often ignored when a computer is used. Perhaps people falsely assume that humans make mistakes and computers don't. We should re-introduce cross checking. Whenever feasible, two independent methods should be used. The command APROPERTIES can often be used effectively to check the correctness of data entered into the worksheet and subsequent calculations. The instruction APROPERTIES produces most of the information formerly given by the old command SUMMARIZE, which it replaces, such as sum of values, minimum value, etc..

3.6 Accuracy Of Instructions.

The developers of OMNITAB consider accuracy to be of paramount importance. The idiosyncrasies of a computer make it particularly important to exercise extreme care and caution in writing the master program. At times some loss in efficiency results in an attempt to provide greater accuracy. It is small comfort to know that you were able to do your computing in one or two seconds less time if the answers are incorrect. Seldom is the increase in computing time (or cost) to achieve greater accuracy of any consequence. Often the improved accuracy avoids considerable embarassment resulting from publication of meaningless results. In problems for which OMNITAB is useful, the computing cost is usually a small fraction of the total cost of the scientific effort and a few more seconds of computer time to obtain accurate results is a small price to pay. An emphasis on accuracy is particularly important in a system like OMNITAB which may be used by persons unfamiliar with the internal workings of a computer.

When it is known that an instruction can produce inaccurate results in certain circumstances, an indication of the accuracy of the instructions should be printed. Notable examples appear in the commands FIT and INVERT. The FIT instruction indicates the computing accuracy of the least-squares coefficients in the automatic printing. (See PART C for further details.) Under rather general conditions, a matrix can be inverted exactly. (See "Solving equations exactly," Morris Newman, J. Res. NBS 71B (1967), 4, 171-179.) But in this case, the additional cost is considerable and not deemed justified for routine use. The algorithm that is used produces an error bound which is printed in the LIST OF COMMANDS, DATA AND DIAGNOSTICS immediately after the listing of the command INVERT (or MINVERT).

The original OMNITAB had its own programs to compute the elementary functions (SIN, LOGE, etc.) which were of known accuracy. In order to make OMNITAB as machine independent as possible, we are unable to follow this procedure and must rely on the system library of the particular computer system in use. The accuracy of system library subroutines varies from one computer to another. Some are quite accurate. Others are surprisingly inaccurate as was explained excellently by W. J. Cody, "Software for the Elementary Functions," Mathematical Software Symposium, Purdue University, April 1-3, 1970. The user should not blithely assume that all 8 digits are correct (or that full machine accuracy is obtained).

Because accuracy is considered very important, the user should not expect that full accuracy is always achieved. OMNITAB is a large system and it would be unheard of with the present state of the computing art to have complete accuracy throughout. The master program is continually being revised to obtain greater accuracy. Although the FIT instruction is accurate ("An Evaluation of Linear Least Squares Computer Programs," R. H. Wampler, NBS J. Res., 74B, 59-90, 1969.), it has been modified two or three times to improve its accuracy. We have already begun to take advantage of recent advances in numerical analysis to make some further improvement.

No command is added to the vocabulary unless the algorithm used is good, if not the best. The implementation of some commands has been delayed, e.g., GAMMA, partly because a satisfactory computing algorithm is not available.

All of this is intended neither to unduly frighten the user nor to make him over-confident. There are those who approach the computer with blind faith. There are those who have extreme skepticism. There is a middle road where the computer can be used sensibly and very effectively.

4. FOR MORE EFFECTIVE USE OF OMNITAB II.

4.1 Self-Teaching.

No manual can hope to answer every single question. This manual is certainly no exception. Despite the simplicity of OMNITAB, users may have many questions. What then? A common practice is to find a neighborhood "expert" and ask him or her the question. This practice has several disadvantages despite its apparent simplicity. It consumes a lot of valuable time, the "expert" may not know the answer, it doesn't help answer the next question, the answer is not always complete and it may not fully sink in. A method that works very effectively is self-teaching. The importance of self-teaching cannot be over-emphasized. It is a useful, inexpensive, simple and often exciting means to clarify the meaning or properties of an instruction.

Before using the STOP command, examine the LIST OF INSTRUCTIONS in PART D to find an instruction (or a few) which (a) you have not used before, (b) you might want to use in the near future, and (c) one whose meaning is not clear. Then write a short set of instructions to exhibit the meaning of the new instruction and insert this short set just before your STOP command. This will take very little time and will cost practically nothing.

For example, from the LIST OF INSTRUCTIONS in PART D it is not clear what the instruction RMS actually does. However, the user can readily find out for himself by using a set of instructions such as:

```
OMNITAB  6/13/70  TEST RMS
GENERATE 1.(1.)5. in column 1
RMS column 1 put in column 2
PRINT columns 1 and 2
STOP
```

By working with integers one can quickly do a few hand calculations to compare with the printed results. Note, the sum of the first n squared integers is $n(n+1)(2n+1)/6$. If you know that RMS stands for something like root mean square you should be able to figure out exactly what operations the instruction performs.

The manipulative instructions are very powerful, yet they are not quite as easy to understand as the arithmetic instructions. One can write a short set of instructions which will help clarify the meaning of several of them at one time. Witness the following set of instructions:

```
OMNITAB  5/28/69 SELF-TEACHING INSTRUCTION SET
GENERATE 1.(1.)10. in column 1
FLIP column 1 into column 2
CENSOR column 1 for 7.0, replace by 2.0, put in col 3
PROMOTE 1 row, column 1 into column 4
MOVE the array in 3,1 of size 6x1 to 2,5
ROW SUM columns 1,2,3 and put in col 6
PRINT columns 1 *** 6
```

The results of using this set of instructions are:

COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4	COLUMN 5	COLUMN 6
1.0000000	10.000000	2.0000000	2.0000000	0.	13.000000
2.0000000	9.0000000	2.0000000	3.0000000	3.0000000	13.000000
3.0000000	8.0000000	2.0000000	4.0000000	4.0000000	13.000000
4.0000000	7.0000000	2.0000000	5.0000000	5.0000000	13.000000
5.0000000	6.0000000	2.0000000	6.0000000	6.0000000	13.000000
6.0000000	5.0000000	2.0000000	7.0000000	7.0000000	13.000000
7.0000000	4.0000000	2.0000000	8.0000000	8.0000000	13.000000
8.0000000	3.0000000	8.0000000	9.0000000		19.000000
9.0000000	2.0000000	9.0000000	10.000000		20.000000
10.000000	1.0000000	10.000000	0.		21.000000

Self-teaching is also effective in determining whether an instruction allows certain values for the argument of a function. For example, what happens if you use the instruction ERROR when some of the values in the column are negative? Also, self-teaching is useful in obtaining a full understanding of the instructions which have a comprehensive automatic printing (see section 1.10).

Hopefully, this manual will answer the major questions and prove useful. But the user, armed with the LIST OF INSTRUCTIONS and an appreciation of self-teaching, could quickly become an "expert" by continually writing short sets of instructions like the above. The sets of instructions for testing OMNITAB in "Test Problems and Results for OMNITAB II," Ruth N. Varner and Sally T. Peavy, NBS Technical Note 551, U. S. Government Printing Office, should also be of help (see section 5.1).

Self-teaching is an effective supplement or even replacement for a manual, but OMNITAB itself can be a useful teacher. If the instruction POLYFIT were used to fit a straight line, the automatic printing would give the coefficients from the straight line fit and also the coefficient (omitting the last term) from fitting just a constant. One could see immediately that the estimate of the constant term is different in the two fits, which dramatizes the fact that the two vectors in the straight line fit are not orthogonal. See PART C for further details.

4.2 A Few Common Errors.

Listed below are some comments on some of the more common errors that are made which one should be especially careful to avoid.

(a) Incorrect number of arguments. It is a good idea to check the number of arguments in your written instruction against the number of arguments given in the column of notes at the extreme right in the LIST OF INSTRUCTIONS in PART D.

(b) Be sure data have been put into the worksheet before using an executable instruction.

(c) Check to make sure all arguments are of the right kind: column numbers must not have a decimal point and constants must have a decimal point. Check against the LIST OF INSTRUCTIONS if in doubt.

(d) The first command after READ or SET must be spelled correctly and must not be a stored instruction, otherwise it will be treated as data.

(e) The use of three asterisks (***) to mean thru in a PERFORM instruction is not allowed.

(f) When writing stored instructions, numbers less than one must be written with a zero before the decimal point and numbers which have a decimal point must have a digit to the right of the decimal point.

(g) Any INCREMENT or RESTORE instruction must have exactly one more argument (the stored instruction number) than the instruction (except tape commands) referred to and the other arguments must agree in kind.

(h) When using the command READ "L" remember to indicate in the first argument the number of data cards that are to be read into the worksheet.

4.3 Combining Sets Of Instructions.

Often it is desirable to process several sets of OMNITAB instructions (problems) simultaneously. It is substantially cheaper to process sets of instructions together rather than separately. When sets of instructions are combined, there should be only one STOP command which appears at the end of the last set of instructions. The OMNITAB command must be used as the first card of each set of instructions. The OMNITAB command, see PART C, initializes everything so that each new set is processed independently of the other sets. In particular, if a FATAL ERROR occurs in one set it has no bearing on the execution of another set of instructions.

4.4 Use Of FORTRAN Formats.

OMNITAB provides considerable flexibility for entering and printing data without using formats. The regular PRINT instruction prints data in a readable form with the decimal point in a fixed position. The number of significant digits printed is easily changed from 8 to any other desired number. The instruction can be modified by preceding it with either the instruction FIXED or the instruction FLOATING. If only one row at a time is to be printed, one can use ABRIDGE. To print arrays or matrices one can use APRINT or MPRINT. In addition, there are a number of commands for improving printing such as HEAD, TITLE, NOTE, PRINT NOTE and SPACE.

In addition to this flexibility, there is provision for using regular FORTRAN formats to meet more exacting requirements. The instruction

FORMAT "L" (user's own format)

can be used by anyone having a knowledge of a FORTRAN language in conjunction with any of the following commands:

READ "L"	APRINT "L"	CREAD TAPE "L", "L"
PRINT "L"	MPRINT "L"	READ TAPE "L", "L"
NPRINT "L"		WRITE TAPE "L", "L"
ABRIDGE "L"		
PUNCH "L"		

There are five commands in the first column for entering and printing data, two in the second column for printing arrays, and three in the third column for magnetic tape operations.

The qualifier "L" represents anyone of the first six letters A, B, C, D, E or F. Thus, as many as six different formats can be used at one time. Note, all magnetic tape operation commands have one or two qualifiers. Only those that are used in conjunction with FORMAT "L" have two qualifiers. The first qualifier refers to the tape in use. The second qualifier refers to the FORMAT.

Whenever a FORMAT command is used:

(i) The FORMAT "L" command must precede (anywhere) the executable command which refers to it.

(ii) The qualifier of the executable instruction must agree with the qualifier of the FORMAT instruction.

(iii) More than six formats can be used in any one set of instructions simply by re-using any of the qualifiers in FORMAT "L", but only six can be used at one time. The master program always uses the last written FORMAT "L".

In the FORMAT "L" instruction, a regular FORTRAN format is inserted between parentheses. Usual FORTRAN rules apply, except continuation cards are not allowed. Also, a format should be used for each single card (line) when entering or printing (punching) data. The regular OMNITAB rule that an instruction must be punched on a single card holds for FORMAT "L" instructions also. A discussion of how to construct FORTRAN format statments is beyond the scope of this manual. It is assumed that anyone using the FORMAT "L" command is familiar with the FORTRAN language. A discussion of FORTRAN is found, for example, in "A Primer For FORTRAN IV," E. I. Organick, Addison-Wesley (1966).

The FORTRAN I and A format specifications may be used to input data, manipulate data and output data of the same type. However, the E or F format specification must be used for any kind of arithmetic operation. The X and H format specifications may be used as in FORTRAN. The following commands can be used with data that has been entered using the A or I format specification: DEMOTE, DUPLICATE, EXCHANGE, FLIP, INSERT, MMATVEC, MOVE (and AMOVE or MMOVE), MTRANSPOSE (and ATRANSPOSE), MVECMAT, ORDER, PROMOTE, SEPARATE, and SORT. However, caution should be exercised with the use of the commands DEMOTE, PROMOTE and SEPARATE as difficulties may be encountered in printing the value zero.

4.5 Organizing A Set Of Instructions.

Flow charts are not necessary in writing a set of OMNITAB instructions. Perhaps this frees the problem solver from a rigid approach to a problem. Sometimes it helps to keep a record of how the columns in the worksheet are used and which columns contain what information. This is particularly true if many operations are being performed or if you need added assistance in interpreting a set of instructions some months later. An example of a coding chart is given in NBS Handbook 101, page 261. Sometimes it helps to divide the columns into multiples of ten and use a separate multiple for each different logical unit of computations.

Cards which are completely blank are ignored in the execution of instructions or in the interpretation of data (except when using READ "L"). In the printing of the LIST OF COMMANDS, DATA AND DIAGNOSTICS the presence of a blank card causes a blank line to be printed. In a lengthy set of instructions it is sometimes helpful to insert blank cards between logical units (data, arithmetic, printing, etc.) to separate them.

Liberal use of descriptive words in writing instructions has many advantages despite the temptation to avoid their use. Punching an instruction tends to flow more smoothly. It is easier to read a set of instructions six months later. It is easier to communicate with someone else. The fact that an OMNITAB set of instructions is often concise makes it possible to use the LIST OF COMMANDS, DATA AND DIAGNOSTICS in a written report, particularly if descriptive words have been used liberally.

Although an instruction can be punched anywhere on a card it is usually easiest to start punching in card column 1. If this practice is generally followed, the indenting of instructions from the left, say to card column 6, helps to offset one or more instructions from the rest of the instructions. This can be done for intermediate calculations.

Another trick is to use cards of different colors for different units of a set of instructions. Control cards might be punched on one colored card, data on another color, executable instructions on another color, etc.

For greater emphasis, one user, in particular, inserts one space between letters and four spaces between words in titles to be printed by OMNITAB, HEAD, TITLE and NOTE instructions. (Don't put a space between letters in a command.)

4.6 Some Aids For Writing Sets Of Instructions.

(a) Large Amounts of Data. OMNITAB is basically designed to handle small to moderate amounts of data. However, there are ways of handling large amounts of data. Often, all that is required is to change the shape of the worksheet by using a DIMENSION instruction. This will not be enough if one has more than 12,500 measurements. In this case there are two tricks which are sometimes successful.

(i) Enter subsets of data one at a time and perform necessary calculations to obtain partial results. After the last group has been processed, complete any further calculations and print final results. With large amounts of data one often wants the data summarized by groups and this technique can be useful.

(ii) Sometimes the data is punched on cards in the form of an array and the size of the array exceeds the size of the worksheet. Furthermore, to obtain partial results entire columns are needed so that it appears necessary to enter the entire array. In this case, a simple trick is to reproduce the data deck and enter the data into the worksheet twice. The first time, the first half of the data is read into the worksheet and the second time the remaining half is read into the worksheet. As an illustration consider the following problem and solution.

Suppose there are 1000 cards and twenty numbers on each card. The average (mean) and standard deviation of the average are desired for each set of 1000 numbers. The total of 20,000 numbers is too large to go into the worksheet. A way to handle the problem is to first reproduce the data deck. This is often more satisfactory than making two separate runs (passes). Then the following concise set of instructions could be used. The set of instructions appears to be far more lengthy than it really is because a liberal number of comment cards has been used.

```
OMNITAB  6/16/70  compute means and their std. dev's for 20 sets of 1000 values
DIMENSION the worksheet to be 1000 rows by 12 columns
1/ SPOLYFIT column 1, weights 1.0, degree 0, x in column 1, put coeffs in col 1
$  S before polyfit is used to suppress automatic printing
$  polyfit of degree zero gives mean in row 1 and std dev of mean in row 2 of
$     column where coefficients are stored.
$  fourth argument is a dummy argument in polyfit of degree zero and can be any
$     valid column number.
$  note that column where coefficients are stored is the same as that where data
$     is stored.  This is valid and saves space.
$  in this case it is more efficient to use polyfit than statis.
2/ ATRANSPOSE the array in 1,1 size 2x1 and put in 1,11
$  this changes a column into a row
3/ INCREMENT instruction 1 by 1, 0.0, 0, 0, 0
4/ INCREMENT instruction 2 by 0,0   0,0   1,0
$  we now have results for second set in second row
READ data into columns 1 *** 10
$  1 *** 10 is equivalent to 1,2,3,4,5,6,7,8,9,10

     ( follow with data deck )

PERFORM instructions 1 thru 4, 10 times
READ data into columns 1***10 and 1 *** 10
$  a column number can be used twice. when 1 is used the second time, the 11th
$  column of data is put into column 1 replacing the data
$  which was entered there first.

     ( follow with second copy of data deck )

RESTORE instruction 1 to 1, 1.0, 0, 1, 1
$ this is necessary or the first argument of instr 1 would become illegal
```

```
PERFORM instructions 1 thru 4, 10 times
RESET 20
$ remember nrmax was 1000
PRINT columns 11 and 12
```

(b) Row Titles. Row titles can be obtained by reading in and printing the titles using an A format, see section 4.4. One column should be allotted for each three characters in the title, including blank spaces. Users of the NBS computer can allot one column for each six characters. To print the coefficients from a quadratic least squares fit, which are stored in column 45, one could use the instructions:

```
FORMAT A (6A3)
READ A format, 3 cards into columns 11 *** 16
estimate of a
estimate of b
estimate of c
FORMAT B (1X,6A3,1PE15.6)
RESET 3
PRINT B format, columns 11 *** 16 and 45
```

(c) Data Manipulation. In general, OMNITAB instructions operate on an entire column down to NRMAX as in

```
ADD column 1 to column 2 and put the results in column 3
```

A number of instructions allow one to operate on only part of a column as in

```
SUM column 1, rows 3 thru 7, put result in column 2
```

Often, however, one needs to perform an operation on certain rows of a column when the row numbers are only known through some property of the data. For example, one might wish to find the sum of all numbers which have a positive value, but not know the row numbers associated with these numbers. Moreover, the values which are needed might not be in consecutive rows of the column.

This sort of problem occurs frequently, but can be resolved easily by a problem solver using a basic technique. The trick is to construct a weight function which is a column of ones and zeros. A one indicates the corresponding value in the data column has a certain property and a zero indicates it does not have that property. In constructing the column of weights one frequently uses the fact that, in OMNITAB, any number divided by itself equals one, except when the number is zero, in which case the result is zero.

Suppose we wish to do a statistical analysis of the even values in column 6, assuming all the numbers are integers. The following set of instructions could be used:

```
DIVIDE column 6 by 2.0, put in column 7
FRACTIONAL part of column 7 put in column 7
DIVIDE column 7 by col 7 and put in col 7
SUBTRACT col 7 from 1.0 and put weights in col 7
STATISTICAL analysis of column 6 using weights in column 7 put in col 41
```

Often, the CENSOR command is very useful in this type of problem and also the command MATCH. To find the sum of all positive values in column 14, one could use

```
CENSOR column 14 for 0.0, replace by 0.0, store in col 15
SUM col 15 put in column 15
```

Suppose we have a column of numbers, X, and we need to know the number of values in the column which are less than some constant, k, (or greater than, equal to, etc.) Denote this

number by n(X LT k). This problem arises frequently in one form or another. A solution is given below.

First, compute Y = (X-k)/ABS(X-k), which equals minus one if a value is less than k, equals zero if the value equals k, and equals plus one if the value is greater than k. Then proceed as follows:

To find:	Perform operations successively on column Y
n(X GT k)	(i) CENSOR for 0.0, replace by 0.0 (ii) SUM
n(X GE k)	(i) CHANGE sign (ii) CENSOR for 0.0 replace 2.0 (iii) SUBTRACT 1.0 (iv) SUM
n(X EQ k)	(i) CENSOR for -1.0, replace 1.0 (ii) SUBTRACT 1.0 (iii) CHANGE sign (iv) SUM
n(X LT k)	(i) CHANGE sign (ii) CENSOR for 0.0, replace by 0.0 (iii) SUM
n(X LE k)	(i) CENSOR for 0.0, replace by 2.0 (ii) SUBTRACT 1.0 (iii) SUM

There is a tendency to think that the instructions for array and matrix operations are only of use in problems related to matrix algebra. However, they can be especially useful in data manipulation and sometimes to remove the need for using stored instructions. Commands like ATRANSPOSE, ADEFINE and MMATVEC are often helpful. As a further note, the instructions APROPERTIES and MPROPERTIES can be used effectively to check calculations and the punching of data cards after data has been entered into the worksheet.

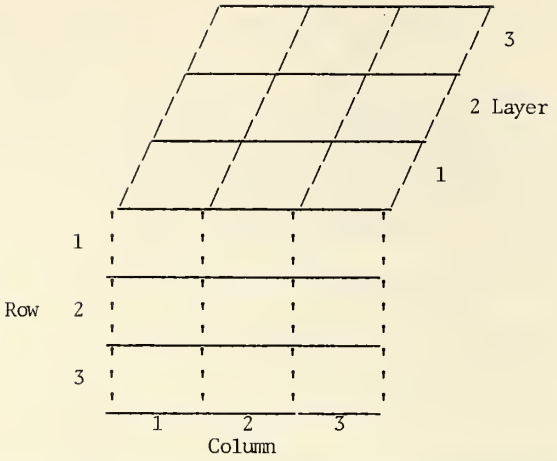
(e) Branching. The very nature of OMNITAB makes logical branching seldom necessary or desirable in the usual sense. Often the equivalent of logical branching can be obtained by using a weight function as implied above. In PART C instructions are described for branching in the REPEAT mode. The following is an example of a situation where logical branching is normally required, but is only used indirectly in OMNITAB. Suppose we want to take the logarithm to the base ten of the quotient of numbers stored in columns 11 and 12. Except, that if the quotient is less than or equal to zero, we want to replace the logarithm by a constant, say 3.8. These instructions could be used:

```
ANTILOG of 3.8 put in column 10
DIVIDE col 11 by col 12 and put in col 13
CENSOR col 13 for 0.0, replace by col 10, put in col 13
LOGTEN of column 13 is put in column 13
```

(f) Listing Of Instructions. To avoid the listing of a lengthy set of data or to avoid the printing of arithmetic faults, particularly when it is known that division by zero will happen frequently, one can use the instructions NO LIST or LIST (n) which are described in section C1.1.

4.7 An Example Of Table Making.

The need to form sub-tables from a set of data is common. It is easier to do in OMNITAB than might appear. This is due to the basic structure of the system and the large set of manipulative instructions available. A study of a problem posed by a potential user illustrates some of the things that can be done. Consider the three-dimensional array below, with data in rows, columns and layers:



The problem is to present a reduced two-way table (rows by columns) having in each cell the mean and standard deviation of the measurements in the three layers. Four solutions are given below for data taken from "An analysis of transformations," G. E. P. Box and D. R. Cox, J. Royal Statist. Soc., B, 26, 223. (1964). The data occurred in a random order. Solution 4 appeared in "OMNITAB - Rapid Statistical Manipulation," J. R. Rosenblatt, B. L. Joiner and David Hogben, U. S. Bureau of the Census, GE40, No. 6, Final 1970 Census Plans and Four Programming Systems for Computerized Data Retrieval and Manipulation, New York, N.Y., August 21, 1969. U.S. Government Printing Office, Washington, D.C., 1970.

People think differently and tend to approach a problem in different ways. It is characteristic of OMNITAB that a problem can be solved in many different ways as is shown here. This is important because it means that the user is free to problem solve in his own way and is not tied to a rigid programming structure. By presenting four different solutions to the same problem, it is hoped the flexibility of approach will be demonstrated.

The instructions which enter the raw data into the worksheet are given only once. A few comments are included to aid the reader, but a complete understanding requires a careful study and possibly some use of self-teaching. Solution 4 uses a PERFORM within a PERFORM. Solutions 2 and 3 avoid one PERFORM by using the instruction MATCH. Solution 1 uses the instruction ONEWAY and avoids the use of PERFORM completely. (The instructions MATCH and ONEWAY were not in the system when solution 4 was written.) Solution 2 stores the means in consecutive columns, whereas solution 3 stores the means in consecutive rows of the same column. The outputs are the same for each solution; except the formats differ. Only the output for solution 2 is shown. Some may object to the excessive amount of computing resulting from the use of STATISTICAL analysis or ONEWAY analysis, but the computing time is very small and costs little. The saving in programming effort more than offsets the increase in computing cost unless, of course, one were to use the same set of instructions many times. The NEW PAGE and ERASE instructions in solutions 2, 3 and 4 are only necessary because all four solutions are given in the same set of instructions.

Output:

```
    445.33333          271.33333          126.00000
    201.83492           64.166450          40.595566

    1082.0000          693.33333          331.33333
     402.73068         298.82659          111.00150

    2940.0000          1068.0000          794.66666
     844.85265          501.00299          397.59946
```

List of Commands:

```
     OMNITAB   8/4/70   CONSTRUCT TWO-WAY TABLE OF MEANS AND S.D.'S
     SET row numbers in column 1
     1 2 1 2 3 2 1 1 3 3 2 2 1 1 1 2 2 2 1 3 3 3 3 3 1 3 2 3
     SET column numbers in column 2
     1 2 3 3 3 1 1 2 2 1 3 1 3 2 2 3 2 2 3 3 3 2 1 1 1 1 2
     SET layer numbers in column 3
     3 2 3 1 1 3 2 1 3 2 3 2 1 2 3 2 1 3 2 3 2 1 1 1 3 1 2
     SET data in column 4
     292 620 90 442 1140 634 370 338 566 3184 220 1198 170 266 210
     332 1022 438 118 360 884 1568 3636 674 2000 1414 1070

$    begin solution 1
$    first instruction produces numbers 1 to 9 corresponding to each of 9 cells
     SUBTRACT 1.0 from column 1, multiply by 3.0, add column 2, put in col 5
     SONEWAY anal. of col 4, identification in col 5, store in cols 11 *** 14
     MTRANSPOSE array in 1,13 size 9x2 to 1,21
     APRINT array in 1,21 of size 2x3
     SPACE 2
     APRINT array in 1,24 of size 2x3
     SPACE 2
     APRINT array in 1,27 of size 2x3

$    begin solution 2
     NEW PAGE
     ERASE columns 5 *** 62
     SUBTRACT 1.0 from col 1, mult by 3.0, add col 2, put in col 5
$    next instr. suppresses printing of arithmetic faults caused by div by zero
     LIST 1
        1/ MATCH col 5 with 1.0, extract from col 4, put in col 11
        2/ DIVIDE col 11 by col 11 and put in column 10
        3/ SSTATISTICAL anal. of col 11, weights in col 10, store in col 41
        4/ INCREMENT instr 1 by 0, 1.0, 0, 1
        5/ INCREMENT instr 2 by 1, 1, 0
        6/ INCREMENT instr 3 by 1, 0, 1
     PERFORM instrs 1 thru 6, 9 times
     MOVE array in 9,41 size 1x9 to 4,41
     APRINT 3,41 size 2x3
     SPACE 2
     APRINT 3,44 size 2x3
     SPACE 2
     APRINT 3,47 size 2x3

$    begin solution 3
     NEW PAGE
     ERASE columns 5 *** 62
     LIST 1
     SUBTRACT 1.0 from col 1, mult by 3.0, add to col 2, put in col 5
```

```
      1/ MATCH column 5 with 1.0, extract col 4, put in col 6
      2/ DIVIDE col 6 by col 6 and put in column 10
$   STATIS stores mean in row 3 of first storage col (41)
      3/ SSTATISTICAL anal. of col 6, weights in col 10, store in col 41 on
      4/ DEFINE 3,41 as 1,7
$   STATIS stores standard deviation in row 9 of first storage col (41)
      5/ DEFINE 9,41 as 1,8
      6/ INCREMENT instr 1 by 0, 1.0, 0, 0
      7/ INCREMENT instr 4 by 0,0   1,0
      8/ INCREMENT instr 5 by 0,0   1,0
$   next instruction is necessary, see PART C for definition of MATCH
      9/ ERASE column 6
    PERFORM instructions 1 thru 9, 9 times
$   next 4 instrs. turn a 9x2 matrix into a 6x3 matrix
    MMATVEC column 7 into 1,11 size 3x3
    MMATVEC column 8 into 1,14 size 3x3
    MVECMAT 1,11 size 3x6 into column 20
    MMATVEC column 20 into 1,21 size 6x3
    MPRINT matrix in 1,21 size 6x3

$   begin solution 4
    NEW PAGE
    ERASE columns 5 *** 62
$   next two instructions get data into standard order
$   note, this is the first time column 3 is used.
    SORT column 2 and carry along columns 1, 3 and 4
    SORT column 1 and carry along columns 2, 3 and 4
      1/ MOVE 1,4 size 3x1 to 1,5
      2/ SSTATISTICAL analysis of column 5, store in col 41 on
      3/ DEFINE row 3 of col 41 as row 1 of column 6
      4/ DEFINE row 9 of col 41 as row 2 of column 6
      5/ INCREMENT instruction 1 by 3,0   0,0   0,0
      6/ INCREMENT instruction 3 by 0,0   0,1
      7/ INCREMENT instruction 4 by 0,0   0,1
    8/ PERFORM instructions 1 thru 7, 3 times
      9/ APRINT 1,6 size 2x3
     10/ SPACE 1 row
     11/ RESTORE instruction 3 to 3,41   1,6
     12/ RESTORE instruction 4 to 9,41   2,6
$   next instr is necessary because we are working with sets of 3
    RESET 3
    FIXED 4
    PERFORM instructions 8 thru 12, 3 times

    STOP
```

5. THE OMNITAB II PROJECT

This section contains some brief notes on the development and management of OMNITAB II which are not directly concerned with the use of OMNITAB II. Sections 5.1 through 5.3 contain a capsule summary of the documentation available to potential users of OMNITAB II. Sections 5.4 and 5.5 contain a few remarks on questions often raised by programmers and managers of computation facilities. Sections 5.6 through 5.10 describe some services, intended to make the system more effective, which are provided by the user-oriented management of the system at the NBS Gaithersburg laboratories.

5.1 Availability Of OMNITAB II.

"The OMNITAB II Magnetic Tape and Documentation Parcel", NBS Magnetic Tape 1 (1970) by David Hogben, Sally T. Peavy and Ruth N. Varner is available from (formerly the "Clearinghouse")

National Technical Information Service
Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151

The complete parcel consists of one reel of magnetic tape and documentation as follows:

(a) Magnetic Tape.

File 1.	OMNITAB Master Program	(16,806 records,	84 characters/record)
File 2.	Test Problems	(2,788 records,	84 characters/record)
File 3.	XREF	(210 records, blocked 10 cards	840 characters/record, /record)
File 4.	Test Results	(718 records, blocked 10 lines	1320 characters/record, /record)

File 1 contains the main program and subprograms of the OMNITAB II system. The test problems (File 2), which are sets of OMNITAB instructions, are designed as bench marks for use in the implementation of the system. XREF (File 3) consists of FORTRAN comment statements which provide a guide for implementing OMNITAB II. File 4 is the output obtained by using the test problems on the NBS computer.

The magnetic tape was prepared on a certified 2400 foot reel in BCD mode, 556 bpi density, even parity, and 7-track. Tapes will be generated by NTIS in other modes, other densities, even or odd parity, and 7 or 9-track, if so requested.

(b) Documentation:

1. Source Listing Of OMNITAB II Program. Sally T. Peavy, Ruth N. Varner, and David Hogben. NBS Special Publication 339 (1970), 371 pages; $4.75.

2. A Systems Programmer's Guide for Implementing OMNITAB II. Sally T. Peavy, Ruth N. Varner, and Shirley G. Bremer. NBS Technical Note 550 (1970), 43 pages; 50 cents.

3. Test Problems and Results for OMNITAB II. Ruth N. Varner and Sally T. Peavy. NBS Technical Note 551 (1970), 190 pages; $1.50.

4. OMNITAB II User's Reference Manual. David Hogben, Sally T. Peavy and Ruth N. Varner. NBS Technical Note 552 (1971), 262 pages; $2.00.

The publications listed in the documentation are for sale separately by the

Superintendent of Documents
U. S. Government Printing Office
Washington, D.C. 20402

The first publication should be ordered by SD Catalog No. C13.10:339. For the last three publications, order by SD Catalog No. C13.46:550, C13.46:551 and C13.46:552, respectively.

Computers were used, almost exclusively, to prepare all the above four publications. The main sections of Special Publication 339 and Technical Note 551 are actual computer output which was computer assisted phototypeset. The main part of Technical Note 550 and some of the examples in PART C of this manual were put on magnetic tape and then printed by a computer terminal.

A 12 pitch IBM 2741 terminal with ½ spacing was used, from start to finish, to prepare this manual and the introductions to the other three publications. An ATS text-editing system was used to go from one draft to another. Most of the text, by far, was printed using a 10 pitch Delegate element (035) and reduced 16%. In some of the formulas other typing elements were used (Delegate (070), 10 pitch Symbol (061) 12 pitch Symbol (004) and Prestige Elite (012)). A few of the formulas were typed separately on a conventional typewriter. Some of the actual computer output in the examples was reduced 25%.

5.2 OMNITAB II Master Program.

The OMNITAB II master program consists of 177 ANSI FORTRAN subprograms. A complete listing of the program is available in NBS Special Publication 339 listed in section 5.1. Considerable effort has been made to make OMNITAB II as machine independent as possible. In addition to restricting the writing of subroutines to the ANSI FORTRAN language, several additional steps have been taken. We exclude, to the best of our knowledge, any statements which are permissible in ANSI FORTRAN, but which are not acceptable in any computing system which might use OMNITAB. In the internal use of alphabetical information, the word length has been restricted to three characters. The length of a printed page line has been restricted to 120 characters.

5.3 Implementation Of OMNITAB II.

Because of the size of the OMNITAB II program, it requires a large computer. It has been implemented on at least five different makes of computers (e.g., IBM 360/50 up, GE 625, CDC 3800 and 6600, Burroughs 5500, and UNIVAC 1108). Since considerable effort has been made to make OMNITAB as machine independent as possible, implementation should have minimal difficulties. The implementation of OMNITAB, however, still remains a task for a systems programmer.

The major problem in implementing OMNITAB relates to the size of the system. Overlay and/or segmentation are necessary with most computer systems and probably desirable with the largest computers or with computers using a time-sharing system. On most computers the overlay problem is more or less solved.

The publication "A Systems Programmer's Guide to OMNITAB II," described in section 5.1, should be helpful for the implementation of OMNITAB. OMNITAB has been successfully implemented on several different computers without the aid of this publication. The present version, Version 5.0, is more readily adaptable and together with the above documentation, a systems programmer should not have too much difficulty.

5.4 Operating Mode.

There has been much discussion in the computing literature concerning the pros and cons of batch processing, remote batch processing and time-sharing. Although the discussion is important, it is somewhat irrelevant as far as the implementation and use of OMNITAB is concerned. This manual describes a version of OMNITAB designed for use in the batch processing mode. However, the very nature of OMNITAB is such as to make it readily adaptable for time-sharing or remote batch processing. A time sharing version was developed by Walter J. Gilbert and K. B. Weiner for the University of Maryland. A similar version is in operation at the University of Rome, Italy. International Telecommunications Network, Inc. adapted an earlier version of OMNITAB for use in the remote batch processing mode.

5.5 Efficiency.

The efficiency of a computer program has long been a subject of interest. Actually, efficiency has only a limited meaning, except when the entire scientific effort is considered. The developers of OMNITAB have been primarily concerned with the efficient and accurate use of the computer to solve scientific problems, and to a lesser extent with the computing efficiency of a particular subprogram. Although the emphasis has been on developing accurate subprograms or subprograms which make the use of OMNITAB easier, indications are that OMNITAB is quite efficient in the use of computer time.

5.6 Development Of OMNITAB II.

The basic spirit and philosophy of OMNITAB remains virtually the same as it was eight years ago. However, it is in a continual state of development. New commands and new features are frequently added to the system. Existing commands are often revised and improved. There are a few commands in the original OMNITAB that remain to be implemented in OMNITAB II. Some of these commands have been superseded by new commands, the rest we hope to implement soon. The current version contains many commands which were unavailable three years ago.

5.7 Comments From Users.

During the past two years 75 comments from NBS users have been responded to in a semi-formal manner. The comments have been valuable and have led to the addition of new commands and several improvements in existing commands. It is essential that a highly user-oriented system like OMNITAB be responsive to users' needs. Consequently, comments from users have been encouraged as much as possible.

5.8 Notices.

For users of the NBS computer only, a notice appears at the end of the LIST OF COMMANDS, DATA AND DIAGNOSTICS which describes the changes made in the current version of OMNITAB. Each notice begins with *** WATSNU IN VERSION X.XX ***, where X.XX is the current version number. This procedure replaces the command WATSNU of the original OMNITAB. In this way users are kept informed of the latest developments. A similar procedure could easily be adopted at other computer centers.

5.9 Newsletters.

To supplement manuals and the notices, newsletters are distributed to users of the NBS computer describing recent developments in the OMNITAB system. Included in the newsletters are descriptions of new instructions, modifications of existing instructions, errors detected and corrected, novel uses of instructions, comments by users, etc..

5.10 Recorded Telephone Messages.

The Computer Services Division records messages giving the current status of the NBS operating system. OMNITAB news may be given at the end of a message. The purpose of these

messages is to provide users with the very latest information before the computer is used. Any difficulties will be reported as soon as they are detected. The current message is obtained by dialing extension 3261.

PART C

DESCRIPTIONS OF INSTRUCTIONS

Explanations are given on how to use each instruction. Examples are given to further illustrate the use of a particular instruction or group of instructions. Examples are tutorial and are not necessarily given to solve a particular problem. Synonyms and abbreviations are not treated separately, but are listed under the principal instruction. Each principal instruction is enclosed in a rectangular box. If an instruction has more than one form, each optional form is enclosed in a trapezoidal box. Comments pertaining to a group of instructions may appear at the beginning of a section or subsection. Within each subsection, instructions are listed alphabetically. The commands appear alphabetically under the subsection title.

Diagnostics (section B3) which may be printed by a particular instruction are included in the description of that instruction. Diagnostics which may be printed by any one of a particular group of instructions are given in the general description of the group at the beginning of the section. Diagnostics which are applicable to a large number of instructions are not discussed.

Instructions are grouped into ten sections which are further subdivided into subsections. Thus, anyone interested in printing, for example, can immediately turn to the appropriate subsection(s) to find which instructions can be used. A capsule summary showing the commands in each section is given on the next two pages. Commands to suppress automatic printing (section B1.10), abbreviations (section B1.12) and synonyms (section B1.13) are not listed. Section 11 contains an index.

See section B1.2 and the beginning of PART D for a description of the notation used to define the arguments of an instruction.

List Of Commands

1. Entering and Printing Data.

Control instructions:	OMNITAB	STOP		
miscellaneous:	DIMENSION	SCAN	NULL	DUMMY "L"
listing:	LIST	NO LIST		
Entering data:	SET	READ	GENERATE	
Printing data:	PRINT	ABRIDGE	NPRINT	
to modify:	FIXED	FLOATING	FLEXIBLE	
Detailed printing:	HEAD	NEW PAGE	SPACE	
notes:	NOTE	NOTE1	NOTE2	PRINT NOTE
page titles:	TITLE1	TITLE2	TITLE3	TITLE4
Plotting:	PLOT	PAGE PLOT	TITLEX	TITLEY
Printing arrays,matrices	APRINT	MPRINT	APRINT "L"	MPRINT "L"
Punching cards:	PUNCH	PUNCH "L"		
Format:	FORMAT "L"			
read with format:	READ "L"			
print with format:	PRINT "L"	ABRIDGE "L"	NPRINT "L"	
Magnetic tape, set:	SET TAPE "L"		CSET TAPE "L"	
read from tape:	READ TAPE "L"		READ TAPE "L", "L"	
read with count:	CREAD TAPE "L"		CREAD TAPE "L", "L"	
write onto tape:	WRITE TAPE "L"		WRITE TAPE "L", "L"	
manipulations:	SKIP TAPE "L"		ENDFILE TAPE "L"	
	REWIND TAPE "L"		BACKSPACE TAPE "L"	

2. Arithmetic Operations.

Simple arithmetic:	ADD	SUBTRACT	MULTIPLY	DIVIDE
power:	SQUARE	SQRT	RAISE	
change sign:	ABSOLUTE	CHANGE		
logarithms:	LOGTEN	LOGE		
antilogs:	ANTILOG	EXPONENTIAL	NEGEXPONENTIAL	
Trigonometric:	SIN	COS	TAN	COT
degrees:	SIND	COSD	TAND	COTD
inverse:	ASIN	ACOS	ATAN	ACOT
degrees:	ASIND	ACOSD	ATAND	ACOTD
hyperbolic:	SINH	COSH	TANH	COTH
inverse:	ASINH	ACOSH	ATANH	ACOTH
Data summarization:	INTEGER	FRACTIONAL	ROUND	ACCURACY
sum:	SUM	PARSUM	ROW SUM	
product:	PRODUCT	PARPRODUCT	EXPAND	
properties:	AVERAGE	RMS	MAXIMUM	MINIMUM
Complex arithmetic:	CADD	CSUBTRACT	CMULTIPLY	CDIVIDE
change coordinates:	CPOLAR	CRECTANGULAR		

3. Data Manipulation.

Defining operations:	RESET	RESET "V"		
	COUNT	DEFINE	ERASE	
Moving data:	MOVE	DUPLICATE	PROMOTE	DEMOTE
	EXCHANGE	CLOSE UP	SEPARATE	INSERT
Manipulative:	FLIP	CENSOR	SHORTEN	
sort:	ORDER	SORT	HIERARCHY	
search:	MATCH	SEARCH	SELECT	

4. Statistical Analysis.

Elementary:	FREQUENCY	HISTOGRAM	NHISTOGRAM	RANKS
Analysis:	STATISTICAL	ONEWAY	TWOWAY	
Regression:	POLYFIT	FIT	CORRELATION	
Probability:	F PROBABILITY	UNIFORM RANDOM		

5. Numerical Analysis.

Error functions:	ERROR	CERF		
Special integrals:	ELLIPT FIRST	ELLIPT SECOND	STRUVE ZERO	STRUVE ONE
Polynomials:	HERMITE	LEGENDRE	LAGUERRE	NORMLAGUERRE
Chebyshev:	TCHEBYSHEV	UCHEBYSHEV		
Iteration:	ISOLATE	ISETUP	ITERATE	
Analysis:	INTERPOLATE	SOLVE	MAXMIN	HARMONIC
Integration:	GAUSS QUADRATURE			

6. Repeat Mode.

Repeated execution:	PERFORM	INCREMENT	RESTORE	
alternative:	BEGIN	FINISH		
Branching:	COMPARE	IFEQ	IFNE	
two arguments:	IFGE	IFGT	IFLE	IFLT

7. Array Operations.

Arithmetic:	AADD	ASUBTRACT	AMULTIPLY	ADIVIDE
	ARAISE			
Define, move operations:	AERASE	ADEFINE	AMOVE	ATRANSPOSE
Properties:	AAVERAGE	ACOALESCE	APROPERTIES	

8. Matrix Operations.

Defining operations:	MDEFINE	MERASE	MIDENTITY	
Moving operations:	MMOVE	MTRANSPOSE	MMATVEC	MVECMAT
	MDIAGONAL	MVECDIAGONAL		
Matrix algebra:	MADD	MSUBTRACT		
multiplication:	MMULTIPLY	MRAISE	MSCALAR	MKRONECKER
	M(AD)	M(DA)	M(AV)	M(V'A)
	M(X'X)	M(XX')	M(X'AX)	M(XAX')
Matrix analysis:	MINVERT	MORTHO	MEIGEN	MPROPERTIES
	MTRIANGULARIZE			

9. Bessel Functions.

First, second of order 0,1:	BJZERO	BJONE	BYZERO	BYONE
modified:	BIZERO	BIONE	BKZERO	BKONE
extreme argument:	EXIZERO	EXIONE	EXKZERO	EXKONE
Complex, angle=PI/4:	KBIZERO	KBIONE	KBKZERO	KBKONE
extreme real argument:	KEXIZERO	KEXIONE	KEXKZERO	KEXKONE
arbitrary angle:	CIZERO	CIONE	CKZERO	CKONE
extreme real argument:	CEIZERO	CEIONE	CEKZERO	CEKONE
Zeros:	ZEROS BJZERO	ZEROS BJONE		
Order n, integral	BESIN	BESJN	BESKN	INTJO

10. Thermodynamics.

Units, temp. conversion:	SI	CGS	FTOC	CTOF
Molecular weight,	ATOMIC	MOLWT	PARTFUNCTION	
properties of state:	BOLDIST	EINSTEIN	PFTRANS	PFATOMIC

1. ENTERING AND PRINTING DATA

1.1 Control Instructions.

DIMENSION, DUMMY "L", LIST, NO LIST, NULL, OMNITAB, SCAN, STOP

DIMENSION the worksheet to have (r) rows and (c) columns

This changes the dimensions of the worksheet from 201 rows and 62 columns to the specified number of rows and columns. The product of the number of rows and number of columns must not exceed 12,500. The instruction DIMENSION 500x30 would be illegal. The instruction is usually used only when one wishes to perform a few calculations on a large set of data, e.g., DIMENSION 1000 x 12, or perform a lot of calculations on a small set of data, e.g., DIMENSION 10 x 1000. The DIMENSION instruction is best put immediately after the OMNITAB instruction and should never be put in the middle of a set of instructions as it changes the entire configuration of the worksheet and could cause a value in one column to end up in some other column. If the product (r)x(c) exceeds 12,500; the following fatal error will occur:

*** DIMENSIONED AREA EXCEEDS LIMIT

The command DIM is allowed as an abbreviation for DIMENSION.

DUMMY "L"

This command was added to the system to allow users to add their own subroutines. The use of DUMMY "L" requires an extensive knowledge of the internal structure of OMNITAB and the command is not discussed any further in this manual.

LIST (n)

The instruction LIST with an argument (n) controls the printing of diagnostics in the LIST OF COMMANDS, DATA, AND DIAGNOSTICS according to the value of the integer n as follows:

0. Print nothing. LIST 0 is synonymous with NO LIST; see below.
1. Suppress the printing of arithmetic diagnostics only.
2. Suppress the printing of informative diagnostics only.
3. Suppress nothing. LIST 3 is synonomous with LIST (see below).
4. Suppress the printing of both arithmetic and informative diagnostics.

As soon as a fatal error occurs, the effect of using NO LIST or LIST (n) is changed to LIST for the remainder of the set of instructions.

LIST 1 (or LIST 2) is particularly useful when it is known that a large number of diagnostics will occur and one does not want to have them printed. For example, in data manipulation, it is common to divide numbers by themselves to obtain a column of weights. If some of the numbers are zero, arithmetic diagnostics will result which may be numerous if

instructions are stored. These diagnostics can be avoided by using LIST 1 before the DIVIDE command. (LIST could be used right after DIVIDE, if further diagnostics are needed.)

If the argument of LIST is less than zero or greater than four, the following informative diagnostic is given:

* THIS COMMAND WAS NOT EXECUTED BECAUSE ITS MEANING WAS QUESTIONABLE

LIST all diagnostics

The command LIST without an argument is used to negate LIST (n) or NO LIST, see below, and all subsequent instructions and diagnostics are printed. The instruction LIST is not printed, however.

NO LIST

All instructions and data which appear after this card are not printed in the LIST OF COMMANDS, DATA AND DIAGNOSTICS until the instruction is countermanded by LIST, see above, or until a fatal error occurs. The instruction only affects the listing of instructions. The most common use of NO LIST is just before READ to avoid the printing of a large set of data. NO LIST is synonomous with LIST 0 described above.

NULL

This instruction does nothing. It can be used as a stored instruction to erase an existing stored instruction which is no longer needed.

OMNITAB

This must be the first instruction of any set of instructions. All information on the card is printed on each page. The information is printed in positions 21 through 100 on the first printed line of each page. Hence, this instruction can be used to supply the date and title for each set of instructions. More extensive titles can be printed as described in section 1.4. The word PAGE and the page number is printed in positions 111 through 118.

OMNITAB initializes a set of instructions as follows:

(1) Each entry in the worksheet is set equal to zero.
(2) NRMAX is set equal to zero.
(3) The worksheet is dimensioned to have 201 rows and 62 columns.
(4) The variables V, W, X, Y and Z are set equal to zero.
(5) All stored instructions are destroyed.
(6) If it is not the first in a series of sets of instructions, it signals the end of the previous set of instructions and causes the LIST OF COMMANDS, DATA, AND DIAGNOSTICS to be printed.
(7) The values of the fundamental physical constants are set in the SI system.
(8) The argument of LIST (n) is set equal to 3.
(9) The argument of SCAN (c) is set equal to 80.
(10) All FORMATs are removed.
(11) The command PRINT is set in normal mode (readable printing).
(12) All TITLEs are erased.
(13) All NOTEs are erased.

Caution: Since NRMAX is set equal to zero, data must be entered (e.g., using READ, SET or GENERATE) before any executable instruction is used (e.g., ADD, STATIS or MINVERT). Non-executable instructions (e.g., NULL, FORMAT etc.) and stored instructions may precede the entry of data. If an executable instruction is used before NRMAX is reset, the following FATAL ERROR will occur (in most cases):

*** NRMAX=0

SCAN the first (c) columns on the following Hollerith cards

Normally, all 80 columns of a Hollerith card are examined in OMNITAB. However, if certain information, particularly numbers, at the end of a card is to be ignored then the SCAN (c) instruction can be used to do this. For example, if data in card columns 73 through 80 are to be ignored by the computer use SCAN 72. The information although not scanned will, however, be printed in the LIST OF COMMANDS. If the SCAN instruction has more than one argument, the following informative diagnostic will be given:

* MORE THAN ONE ARGUMENT IN COMMAND. ONLY FIRST ONE IS USED

This could happen, for example, if the card containing SCAN 72 had a number in card columns 73-80.

STOP

This must be the last card at the end of the last set of instructions. It signals the end of the use of OMNITAB and returns control of the computer to the executive system. It causes the last LIST OF COMMANDS, DATA AND DIAGNOSTICS to be printed. Forgetting to use a STOP card is the major cause of not having a list of commands printed.

1.2 Entering Data Into The Worksheet.

GENERATE, READ, SET.

Data on cards following the READ or SET instructions may appear anywhere on the card and either with or without a decimal point. Data must be separated by a space, comma or word (non-numeric characters). Integers less than -8191 must be punched with a decimal point.

Each of these commands may affect the value of NRMAX. If the number of rows required to enter data into the worksheet exceeds NRMAX, then NRMAX will be automatically reset to agree with the new number of rows. The value of NRMAX is never decreased by these instructions. If fewer than NRMAX rows of data are entered into the worksheet, the numbers in the remaining rows of the worksheet remain unchanged.

Comments are allowed on data cards. Consequently, the first command after data cards must be spelled correctly. Otherwise, the card will be mistaken for a data card. Also, if a comment is at the beginning of a card, it should not be one of the OMNITAB commands or it will be treated as an instruction card.

In each of these commands, the following informative diagnostic will be given if an attempt is made to enter too much data into the worksheet:

* TOO MUCH DATA IN SET, READ OR GENERATE, SPILL LOST

Stored instructions should not immediately follow data cards used by a READ or SET instruction becuase instruction numbers may be interpreted as data.

Other forms of READ and SET are described in sections 1.7 and 1.10.

GENERATE from (K) in steps of (K) to (K) in steps of (K) to (K) ... in col (C)

This instruction generates a sequence of numbers with differences specified by the even arguments in the instruction. To enter the consecutive numbers 11 through 20 in column 30, one could use the instruction

GENERATE 11.(1.)20. in column 30

Here, the parentheses are used in the usual mathematical context. Note, the number of arguments in the instruction must be even and at least 4. The even arguments which determine the step size can be positive or negative, but not zero. The instruction

GENERATE 1.(1.)10.(-1.)1. in column 32

could be written

GENERATE 1.(1.)10.(1.)1. in column 32

since the context dictates that the second step size must be negative. (In this instruction the decimal points are not needed since the form of the instruction dictates that all arguments except the last are constants.)

The difference between any two arguments on both sides of a step size should be an integral multiple of the step size, indicating the number of steps to be taken. For example, in GENERATE 11.(2.)21. IN COL 42, 21.-11. = 10. = 2(increment)x5(steps). If such is not the case, the last increment will not equal the designated step size. In the instruction

GENERATE 11.(2.)21.7 into column 42

the numbers 11., 13., 15., 17., 19., 21. and 21.7 would be put in column 42. The last number generated always equals the number on the right of the step size in the instruction.

READ data on following cards into columns (C), (C) ..., (C) row by row

The data on the cards which follow are read into the specified columns, one row at a time. Each card contains the data for one row. The numbers on the first card go into row 1 of all the specified columns; the numbers on the second card go into row 2, etc. This continues until a valid instruction is encountered or until the columns are completely filled.

(1) If any card is partially complete, zeros are entered in the remaining columns. Blank cards are ignored as usual.
(2) The value of NRMAX is increased, if necessary, to agree with the number of cards read.
(3) Any extra numbers on a card are ignored.
(4) Stored instructions should not immediately follow the data cards.

If NRMAX = 2:

```
READ data into columns 41, 42 and 43
    11 12 13
    21 22 23 24
    31 32
```

would cause NRMAX to be reset to 3; and the numbers 11.0, 12.0 and 13.0 to be put in row 1 of columns 41, 42 and 43 respectively; the numbers 21.0, 22.0 and 23.0 to be put in row 2 of columns 41, 42 and 43; and the numbers 31.0, 32.0 and 0.0 to be put in row 3 of columns 41, 42 and 43. Only three column numbers are given in the READ instruction, so the fourth number on the second card, 24, is ignored. Since there are only two numbers on the third card, the third number is set equal to zero. If NRMAX had been 5 originally, it would not be decreased to 3. The value of NRMAX is increased, if necessary, but never decreased by the READ instruction.

SET data on following cards into column (C)

The numbers on the following card(s) are put into the rows of the specified column. The first number is put into the first row, the second number into the second row, etc. until a valid instruction is encountered. The SET instruction is similar to the READ instruction, except SET can only be used for one column at a time and the number to be put in a row does not have to be put on a separate card but can follow the previous number on the same card. Consequently, many numbers can be punched on a few cards and it is often preferable to use SET when entering data into just a few columns. As in READ, the value of NRMAX is increased if necessary. Also, stored instructions should not immediately follow the data. If there are fewer numbers on the card than the value of NRMAX, the remaining numbers in the column are unchanged. For example, if NRMAX =5 and column 27 contains the numbers 11.0, 12.0, 13.0, 14.0 and 15.0, then the result of using

```
SET into column 27
    31   32   33
```

would be to put the numbers 31.0, 32.0, 33.0, 14.0 and 15.0 in the first five rows of column 27.

SET the data on the following cards, starting with row (R) of column (C)

The SET instruction with two arguments performs exactly like the SET instruction with one argument described above, except the entering of data begins with row (R) instead of row 1. All the rows before (R) remain unchanged. If NRMAX = 5 and column 27 contains the numbers 11.0, 12.0, 13.0, 14.0 and 15.0, then the result of using

```
SET data into row 3 of column 27
    33   34   35   36
```

would be to put the numbers 11.0, 12.0, 33.0, 34.0, 35.0, and 36.0 in column 27. NRMAX would be reset to 6. Note, if R = 1, this instruction is equivalent to the SET instruction with only one argument.

1.3 Common Printing Instructions.

ABRIDGE, FIXED, FLEXIBLE, FLOATING, NPRINT, PRINT.

The basic command for printing data in the worksheet is PRINT, which simply prints columns of data in "readable form", a feature unique to OMNITAB II. Numbers in a column are

printed with the decimal point in a constant position determined by the values of the data in a column. Traditional forms of printing are possible by using either of the commands FIXED or FLOATING. Methods of obtaining detailed printing are described in section 1.4. Optional forms of the printing commands are described in section 1.6.

```
ABRIDGE row (R) of columns (C), (C), ... (C)
```

Whereas PRINT causes an entire column to be printed, the command ABRIDGE prints only a single row. The command is often useful in the repeat mode for printing results for each iteration. Since only one row is printed, all of the features of PRINT are unavailable; except numbers are still printed in "readable form" unless FIXED or FLOATING has been used. If rows are printed successively, the decimal points will not necessarily line up as would be the case with PRINT.

```
FIXED with (d) digits after the decimal point
```

Command forces any of the commands ABRIDGE, APRINT, MPRINT, NPRINT, or PRINT, which follow, to print numbers in a column with exactly (d) digits after the decimal point. The command remains in effect until countermanded by either FLEXIBLE or FLOATING (or another FIXED). The single argument (d) must be an integer between 0 and 8, inclusive. If not, (d) is set equal to 8.

```
FLEXIBLE to return to readable printing
```

Removes the effect of using either FIXED or FLOATING. It returns the operation of ABRIDGE, APRINT, MPRINT, NPRINT and PRINT to the normal mode of printing numbers in "readable form". No arguments are used in the instruction. The command OMNITAB automatically puts FLEXIBLE into effect.

```
FLOATING with (s) significant digits
```

Forces the commands ABRIDGE, APRINT, MPRINT, NPRINT, and PRINT to print numbers using the scientific notation or floating-point form. The argument (s) determines the number of digits printed. It must be an integer between 1 and 8, inclusive. If not, (s) is set equal to 8. Numbers are automatically rounded to the specified number of digits.

A floating-point number is one which has been normalized to be a number, greater or equal to one but less than ten, times the suitable power of ten. Each number consists, in order from left to right, of (i) blank spaces, (ii) a minus sign if the number is negative and a blank space if the number is zero or positive, (iii) the first significant digit, (iv) the decimal point, (v) (s-1) digits after the decimal point, (vi) a plus or minus sign of the power of ten needed to multiply the normalized number to obtain the number in usual form, and (vii) two digits giving the power of ten. The total number of characters, including blanks, is 15. Let the letter b denote a blank space. If s=6, the number -762.89357 would by printed as

bbbb-7.62894+02

Some computers may put the letter E before the sign as in bbb-7.62894E+02.

FLOATING with eight significant digits

The command without an argument is synonomous with the above command with the argument (s) automatically set equal to 8.

NPRINT columns (C), (C), ... (C)

The letter N (for no new page) before PRINT implies the command (a) does not start printing on a new page and (b) ignores all HEAD column commands (see section 1.4).

If the number of columns to be printed exceeds 8, columns are printed in blocks of 8 (or less). The entire NRMAX rows are printed before proceeding to the next block. A blank line is inserted between blocks. If NRMAX is less than or equal to 48, rows are printed in blocks of five with a blank line between each block. If NRMAX exceeds 48, rows are printed in blocks of ten. In all other respects NPRINT works like PRINT.

PRINT columns (C), (C), ... (C)

All NRMAX values in a column are printed in "readable form" unless FIXED or FLOATING is in effect. All numbers are printed with 8 significant digits. (See section 1.6 for other possibilities.)

(1) If NRMAX is less than or equal to 48, rows are printed in blocks of five with a blank line between blocks. If NRMAX exceeds 48, rows are printed in blocks of ten.

(2) The word COLUMN and the column number are printed two lines above each column. For an alternative, see discussion of HEAD in section 1.4. Fifteen positions (1½") are needed for each column. At most 5 columns will fit on 8½ x 11" paper. COLUMN starts in position 4.

(3) Printing always starts on a new page.

(4) If there are three or more consecutive zeros at the bottom of a column, the zeros are not printed. (Blank spaces are supplied.) If every value in a column is zero, the entire column will be blank and no column heading will be supplied. This feature enables users to space columns of data apart, if desired.

(5) Zero is printed as 0., rather than 0.000000+00.

(6) If the number of arguments exceeds 8, columns are printed in blocks of 8 (or less). The instruction

```
PRINT columns 1 *** 19
```

would give the same results as the three instructions

```
PRINT columns  1 ***  8
PRINT columns  9 *** 16
PRINT columns 17 *** 19
```

(7) If the range of the numbers in a column is too large to enable all numbers to be printed in "readable form", then some of the numbers will be printed in floating-point form. To emphasize this condition, an asterisk (*) is printed on the left of floating-point numbers. If there are 3 or less orders of magnitudes, all numbers will be printed in "readable form".

1.4 Detailed Printing.

HEAD, NEW PAGE, NOTE, NOTE1, NOTE2, PRINT NOTE, SPACE,
TITLE1, TITLE2, TITLE3, TITLE4

The commands in this section provide added flexibility in the printing of results, particularly in the generation of reports. The commands NOTE1, NOTE2, and PRINT NOTE are associated and are described jointly. Similarly, the commands TITLE1, TITLE2, TITLE3, TITLE4 are described jointly. Only the commands in this section have a numeral in the command. Care should be used in using these commands as experience shows that attempts to provide special printing are prone to errors. If a dollar sign ($) (see section B1.5) is any one of the characters in any of the instructions in this section, it does not stop scanning of the rest of the card. It is treated like any other character. A dollar sign ($) in an instruction indicates that the phrase which follows is a descriptive aid to the user and is not used by the instruction.

HEAD column (C)/ $ the 12 characters after / are used as column heading

The 12 characters immediately following the slash (/) are used as a column heading in place of the usual 12 character heading "COLUMN (C) " provided by OMNITAB. All characters (including blanks) described in section B1.15 are counted. The command can be used to provide headings for any of the commands FIT, PLOT, POLYFIT or PRINT. In the PRINT command, three blanks appear on the left and the column heading appears on the right at the top of each column. In the plot command, the column heading will follow ORDINATE and ABSCISSA. The column heading will also be used by the commands FIT and POLYFIT in the title and data headings. Any characters after the 12th character, following the slash, will be ignored.

The HEAD command may be used in conjuction with the first two optional forms of PRINT described in section 1.6. It can not be used with a PRINT "L" instruction. The column heading will be ignored and no diagnostic will be given.

The headings may be updated at any time by using a new HEAD instruction. Only 50 HEAD commands are allowed at any one time. The use of additional HEAD instructions wipes out the original ones to the extent that an excess number of headings is used. The 51st HEAD instruction destroys the 1st, the 52nd destroys the 2nd and so on. If more than 50 HEAD instructions are used, the following informative diagnostic is given

* MORE THAN 50 HEAD COLUMN COMMANDS HAVE BEEN USED.

The command HEAD cannot be stored for repeated execution. If it is included in a set of stored instructions, the instruction is carried out at the time it is first encountered and is then deleted from the set of stored instructions. The following informative diagnostic is given:

* COMMAND NOT ALLOWED IN THE REPEAT MODE. EXECUTED BUT NOT STORED.

If the instruction is punched incorrectly, e.g., column number or slash is omitted, the instruction is ignored and the following informative diagnostic is given:

* BAD HEAD. COLUMN GT 50 OR NO /

(The diagnostic GT 50 should be NUMBER INCORRECT.)

```
NEW PAGE
```

Assures printing will start on a new page. The command can be used with any of the print commands which do not otherwise start printing on a new page: ABRIDGE, ABRIDGE "L", APRINT, APRINT "L", MPRINT, MPRINT "L", NPRINT, and NPRINT "L".

```
NOTE   $  information in Hollerith card columns is printed immediately
```

One blank space should follow the command NOTE. The characters following NOTE through card column 80, including the blank, are printed immediately. This instruction allows additional details in printing. It is mainly used in conjunction with SPACE and ABRIDGE. It may also be used with ABRIDGE "L", APRINT, APRINT "L", MPRINT, MPRINT "L", PLOT (at bottom of page), PRINT, PRINT "L" and any of the commands which provide a comprehensive automatic printing of results (see section B1.10).

The command NOTE must not be stored. However, the command PRINT NOTE, which executes NOTE1 and NOTE2, (see below) can be stored for repeated execution.

```
NOTE1   $   next 60 characters are stored for printing first half of note
```

```
NOTE2   $   next 60 characters are stored for printing second half of note
```

```
PRINT NOTE
```

The commands NOTE1 and NOTE2 are automatically stored and are not executed until the command PRINT NOTE is used. The commands may be used together to give a note occupying a full page line (120 characters). Either or both of the commands may be revised at any time by simply writing a new instruction(s).

The commands can not be stored for repeated use. However, the command PRINT NOTE, which executes the commands NOTE1 and NOTE2, may be stored.

One blank space should follow the command NOTE1. This is not necessary for NOTE2. For NOTE1 and/or NOTE2, the first 60 characters after the numeral 1 or 2 are stored for later printing. If less than 60 characters are used, blanks are supplied at the end. Actually, only 59 characters of NOTE1 are printed; 60 of NOTE2.

The two-word command PRINT NOTE simply causes the information in the instructions NOTE1 and NOTE2 to be printed immediately. It is similar to NOTE; except it can provide a longer note and can be stored for repeated use. If NOTE1 has not been used, the 60 characters after NOTE2 will appear on the right hand half of the printed line and the first half will be blank. Similarly, if NOTE2 has not been used, the 60 characters after NOTE1 will appear on the left half of the line and the right half will be blank.

SPACE (p) lines on printed page

The specified number (p) of blank lines appear on a printed page. This command is used chiefly in conjuction with any of the commands ABRIDGE, ABRIDGE "L", APRINT, APRINT "L", MPRINT, MPRINT "L", NOTE, NPRINT, NPRINT "L", and PRINT NOTE. The command SPACE has no effect on any command which starts printing on a new page; such as PLOT, PRINT and PRINT "L". If (p) is omitted, one blank line will be printed.

TITLE1	$	next 60 characters printed on first half of second line
TITLE2	$	next 60 characters printed on second half of second line
TITLE3	$	next 60 characters printed on first half of third line
TITLE4	$	next 60 characters printed on second half of third line

The four commands TITLE1, TITLE2, TITLE3, and TITLE4 provide a two line title which appears on each page immediately after the first printed line containing "OMNITAB" and "PAGE". Any or all of the instructions may be used. The 60 characters, including blanks, after the numeral (1, 2, 3 or 4) are saved. Any or all of the instructions may be revised by simply writing a new instruction. A title may be deleted by using an instruction(s) with 60 blanks after the command.

If a TITLE number is punched incorrectly, the following informative diagnostic is given:

* IMPROPER TITLE NUMBER, ASSUMED 1

The TITLE instructions can not be stored for repeated use. If an instruction is numbered, the following informative diagnostic is given:

* COMMAND NOT ALLOWED IN THE REPEAT MODE. EXECUTED BUT NOT STORED

1.5 Plotting Data.

PAGE PLOT, PLOT, TITLEX, TITLEY

There are two commands, PAGE PLOT and PLOT, which enable the user to plot data using the high speed printer. The two commands are essentially the same; except PLOT uses all 120 spaces of a printed line, whereas PAGE PLOT uses only 72 spaces on any line. Hence, a PAGE PLOT will fit on an 8½ x 11 piece of paper. Each of these commands has five different forms which control the scale of the vertical and/or horizontal axes. The commands TITLEX and TITLEY, described jointly, enable the user to supply his own titles on the horizontal and vertical axes respectively.

Each plot has 51 positions vertically and 101 positions horizontally. On each scale plus signs are printed at every tenth position (left and right, top and bottom). Minus

signs are printed in the other positions. Six values are printed along each axis at equal intervals. Floating-point numbers with 5 digits are used. If zero is on a scale, an X is printed in the proper position.

At the top of each plot, a short description is given of the columns plotted. The same column headings used by PRINT are given for ABS- (abscissa) and ORD- (ordinate), i.e., "COLUMN (C)" if a HEAD instruction has not been used or the heading if a HEAD instruction has been used. In addition, the plotting symbol(s) used are printed.

In each option any number of columns (functions) from one to five can be plotted against the single abscissa. Plotting symbols are assigned as follows:

Column	Plotting Symbol
1st	.
2nd	*
3rd	+
4th	,
5th	-

Where two or more symbols would coincide, a digit showing the actual number of points is printed rather than any one of the above symbols. If 10 or more points coincide, then an X is printed.

In the first option, the user does not have to be concerned about the scales. Both the vertical and horizontal scales are automatically determined by the instruction. The program determines the largest and smallest values and uses these values as the extreme values of each scale. If more than one function is plotted, the program finds the largest and smallest values of all columns combined. The remaining four options provide the user with flexibility in choosing scales which often improve the appearance of the plot. If a scale is determined by the user, it may happen, accidently or intentionally, that some values fall outside the specified range and cannot be plotted. In this case a diagnostic is printed at the top of the plot showing the number of points plotted and the number which fall outside the bounds.

Two lines are available at the bottom of a plot for printing additional information. Often one or two of the commands TITLEX, NOTE or ABRIDGE are used.

The PAGE PLOT instructions are listed, but are not described as they closely resemble the corresponding PLOT instructions below. The only difference is PAGE PLOT has 61 plotting positions on the horizontal scale instead of 101. Scale values are printed at the 0th, 20th, 40th, and 60th positions. However, the commands TITLE2, TITLE3 and TITLE4 are ignored by the command PAGE PLOT.

PAGE PLOT columns (C), (C), ... (C) against column (C)

PAGE PLOT cols (C) ... (C), with vertical scale from (K) to (K), against col (C)

PAGE PLOT cols (C) ... (C) against col (C) with horizontal scale from (K) to (K)

PAGE PLOT cols (C)...(C) vertical (K) to (K) vs col (C) horizontal (K) to (K)

PAGE PLOT cols (C)...(C) vs col (C), horizontal (K) to (K), vertical (K) to (K)

Page 68 shows an example of the use of the PAGE PLOT instruction. The plot shows 5 straight lines, y = a+bx, with a = 4, 9, 16, 25 and 36; b = 2, 3, 4, 5 and 6; and x = 0.(1.)100.. This example of PAGE PLOT has been used to illustrate what happens when points fall out of bounds. Consecutive digits have been used in TITLEX and TITLEY to show exactly where the characters appear on the plot. The actual set of instructions used was:

```
OMNITAB   11/27/70   EXAMPLE OF PAGE PLOT
    GENERATE 0. (1.0) 100. IN COLUMN 1
    1/ ADD 2.0 TO COL 1, MULT BY 2.0, ADD 0.0, STORE IN COL 11
    2/ INCREMENT 1 BY 1.0, 0, 1.0, 0.0, 1
    PERFORM INSTRUCTIONS 1 THRU 2, 5 TIMES
    TITLE1 PAGE PLOT OF FIVE STRAIGHT LINES USING 5TH. OPTION
    TITLE3 Y = A+BX, A=4,9,16,25,36   B=2,3,4,5,6   X=0(1)100
    TITLEX  3456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
    TITLEY  3456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
    PAGE PLOT COLUMNS 11 12 13 14 15 VS COL 1 0.0 TO 60.0, 0.0 TO 250.0
```

*****

PLOT columns (C), (C), ... (C) against column (C)

Both the horizontal and vertical scales are determined by OMNITAB. The instruction is easy to use and requires no thought in planning. The options below provide more flexibility.

PLOT columns (C) ... (C), with vertical scale from (K) to (K), against col (C)

This option is identical to the first; except the range of the vertical scale (ordinate) is chosen by the user rather than by the instruction. The scale does not have to be increasing, i.e., the first (K), which specifies the value at the bottom of the plot, can be greater than the second (K), which specifies the value at the top of the plot. The following is a valid instruction:

```
PLOT column 41 from 10.0 to -10.0 versus column 14
```

Page 69 shows an example of the use of the second form of the PLOT instruction. The functions sine, cosine, tangent and cotangent (see section 2.4) are plotted for x = -5.0(0.1)5.0. The actual set of instructions used was:

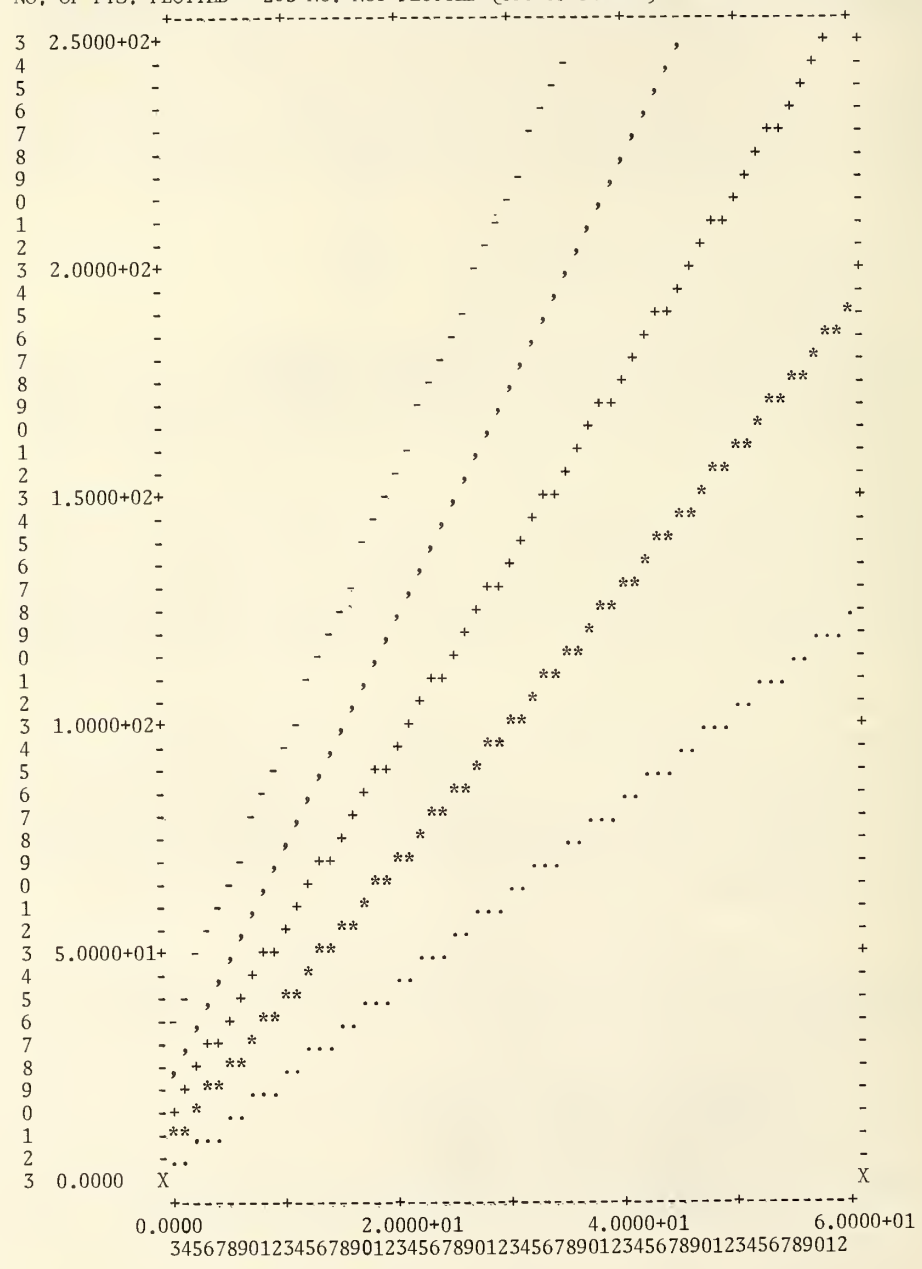
```
OMNITAB   11/27/70   EXAMPLE OF PLOT
    GENERATE -5.0 (0.1) 5.0 IN COLUMN 10
    SIN OF COLUMN 10 PUT IN COLUMN 1
    COS OF COLUMN 10 PUT IN COLUMN 2
    TAN OF COLUMN 10 PUT IN COLUMN 3
    COT OF COLUMN 10 PUT IN COLUMN 4
    PLOT COLUMNS 1, 2, 3, AND 4 FROM -1.0 TO 1.0 VS COLUMN 10
```

PAGE PLOT OF FIVE STRAIGHT LINES USING 5TH. OPTION

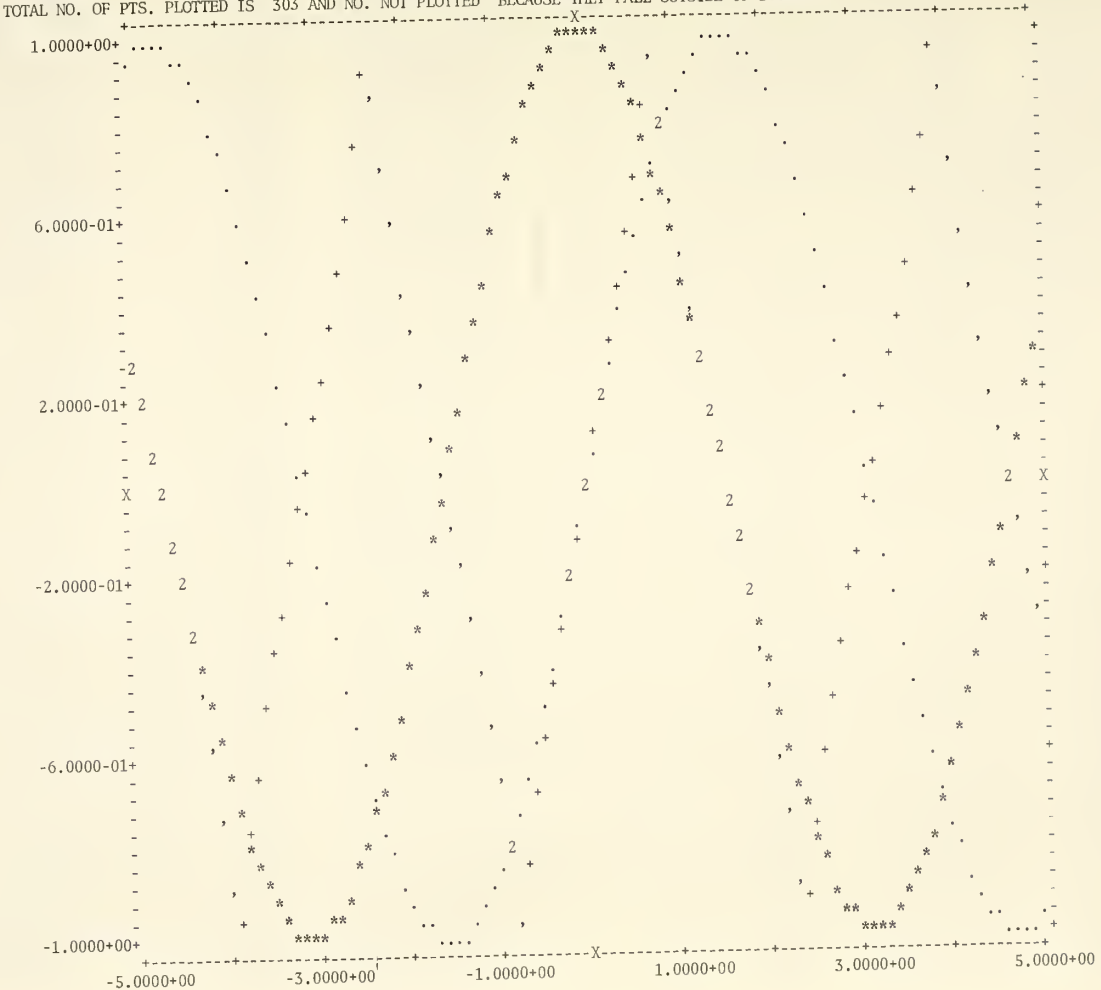
ABS- COLUMN 1

ORD- COLUMN 11 (.), COLUMN 12 (*), COLUMN 13 (+), COLUMN 14 (,), COLUMN 15 (-),

NO. OF PTS. PLOTTED 263 NO. NOT PLOTTED (OUT OF BOUNDS) 242



ABS- COLUMN 10 ,ORD- COLUMN 1 (.), COLUMN 2 (*), COLUMN 3 (+), COLUMN 4 (,),
TOTAL NO. OF PTS. PLOTTED IS 303 AND NO. NOT PLOTTED BECAUSE THEY FALL OUTSIDE OF BOUNDS IS 101



```
PLOT cols (C)...(C) against column (C) with horizontal scale from (K) to (K)
```

Here, the range of the horizontal scale (abscissa) is selected by the user and the vertical scale is determined by the instruction.

```
PLOT cols (C)...(C) vertical (K) to (K) vs col (C) horizontal (K) to (K)
```

Here, both the vertical scale and the horizontal scale are selected by the user.

```
PLOT cols (C)...(C) vs col (C), horizontal (K) to (K) vertical (K) to (K)
```

This option is a slight modification of the preceding option. The only difference is that the four arguments, which specify the vertical and horizontal scales, appear at the end of the instruction. Warning: unlike the previous instruction, the two arguments which specify the horizontal scale come before the two arguments which specify the vertical scale.

```
TITLEX $ 60 characters after 2nd space following X are printed horizontally
```

```
TITLEY $ 51 characters after 2nd space following Y are printed vertically
```

Two spaces should follow either command. The dollar sign ($) in the instruction indicates that the information which follows is for the aid of the user and is not part of the instruction. The 60 characters following the second space after the X of TITLEX will be printed, centered, below the horizontal axis (x-axis or abscissa) on every subsequent page printed by PLOT or PAGE PLOT. The 51 characters following the second space after Y of TITLEY will be printed vertically to the left of the vertical axis (y-axis or ordinate) of any subsequent plot. Note, TITLEX allows the use of 60 characters, but TITLEY only allows the use of 51 characters. Either instruction may be revised at any time by simply rewriting the instruction.

The TITLEX and TITLEY instructions must not be stored for repeated use. If they are stored (numbered), the instruction is performed when it is first encountered and then deleted from the set of stored instructions. As is the case with the other TITLE instructions (section 1.4), an informative diagnostic is given.

1.6 Optional Forms Of Readable Printing.

ABRIDGE, NPRINT, PRINT

The basic forms of ABRIDGE, NPRINT and PRINT were described in section 1.3. For each of these commands there are five additional forms. The options are described below for PRINT, but are merely listed for ABRIDGE and NPRINT. The first two options of each instruction are fairly simple. They simply change the number of digits printed from 8 to the specified number. The last three options provide considerable flexibility at the expense of simplicity. They are not recommended for the beginner. A liberal amount of self-t ching (section B4.1) should help in understanding how to use these options effectively.

The first two options are closely related; actually, the first option is a simple form of the second option. Also, the last three options are closely related; the third and fourth options being special cases of the fifth option. The last three options are described jointly.

The HEAD command is ignored by the last three options. None of the five options can be used if FIXED or FLOATING is in effect. If FIXED or FLOATING has been used by mistake, the following informative diagnostic is given:

* THIS COMMAND WAS NOT EXECUTED BECAUSE ITS MEANING WAS QUESTIONABLE

Each option has an argument which specifies the number of digits to be printed. If the number exceeds 8, it is automatically reset to 8, but no diagnostic is given. If the argument is less than 1, it is automatically reset to one, but no diagnostic is given.

ABRIDGE row (R) of columns (C),(C), ... (C) with (K) significant digits

ABRIDGE row (R) of (C) ... (C) with (K) s. digits, (C) ... (C) with (K), etc.

ABRIDGE row (R), (K) cols, (C) (s), (C) (s), etc. $ max width 22, 3 blanks

ABRIDGE row (R), (K) cols, (C) (s) (m) max width, (C) (s) (m), etc. $ 3 blanks

ABRIDGE row (R) of (K) cols, (C) (s) (m) (b) blanks, (C) (s) (m) (b), ...

*****

NPRINT columns (C), (C), ... (C) with (K) significant digits

NPRINT columns (C)...(C) with (K) s. digits, (C)...(C) with (K) s. digits etc.

NPRINT (K) cols, (C) with (s) s.d., (C) with (s), etc $ max width 22, 3 blanks

NPRINT (K) cols, (C) with (s) s.d. and (m) max width, (C) (s) (m) etc $ 3 blanks

```
/------------------------------------------------------------------------\
/ NPRINT (K) cols, (C) with (s) s.d. (m) max width (b) blanks, (C) (s) (m) (b) etc /
/------------------------------------------------------------------------/
```

*****

```
/------------------------------------------------------------------------\
/ PRINT columns (C), (C), ... (C) with (K) significant digits            /
/------------------------------------------------------------------------/
```

This instruction is the same as the normal PRINT instruction; except the number of significant digits in each value of each column is changed from 8 to (K). Although the argument (K) represents a mathematical integer, it must be written with a decimal point to avoid being mistaken for a column number. It can be any of the values 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, or 8.0. If (K) is not an exact integer, it is truncated to an integer, but no diagnostic is given. For example, if K=2.3, K is automatically reset to 2.0.

```
/------------------------------------------------------------------------\
/ PRINT columns (C)...(C) with (K) s. digits, (C)...(C) with (K) s. digits, etc. /
/------------------------------------------------------------------------/
```

This is a special case of the option above. Here, the number of significant digits printed can vary from one column (or group of columns) to another.

```
/------------------------------------------------------------------------\
/ PRINT (K) cols, (C) with (s) s.d., (C) with (s), etc $ max width 22, 3 blanks /
/------------------------------------------------------------------------/
```

```
/------------------------------------------------------------------------\
/ PRINT (K) cols, (C) with (s) s.d. (m) max width, (C) (s) (m), etc. $ 3 blanks /
/------------------------------------------------------------------------/
```

```
/------------------------------------------------------------------------\
/ PRINT (K) cols, (C) with (s) s.d. (m) max width (b) blanks, (C) (s) (m) (b), etc /
/------------------------------------------------------------------------/
```

These three options allow the user to change the width of a column and the number of blank spaces between columns, in addition to varying the number of digits printed. In each case the first argument, (K), specifies the number of columns to be printed. In the last option the argument (K) is followed by (K) sets of four arguments. Hence, the total number of arguments in an instruction is (4K+1). The arguments in each set, all integers, are:

(C) - the number of the column to be printed.
(s) - the number of digits to be printed.
(m) - the maximum width of a column, excluding blanks on the left.
(b) - the number of blank spaces to appear at the left of the column.

In the fourth option, the fourth argument of a set, (b), is missing and it is automatically set equal to (assumed to be) 3. Here, we have a total of (3K+1) arguments. In the third option, both of the arguments (m) and (b) are missing. Automatically, (m) is set equal to 22 and (b) is set equal to 3.

The argument (b) allows the user to provide different spacings between columns.

The meaning of the argument (m) can be easily misunderstood. It is not the actual column width, but rather the maximum width that the user will tolerate. The actual width

may be smaller. Each column of data is examined to determine how many spaces are needed to print all the numbers in the column in "readable form", i.e., without using floating-point notation. For example, suppose s=4, NRMAX=2 and the numbers in column 7 are:

```
-76.24
  .001593
```

then 9 spaces would be needed. Two spaces are needed for the sign and the decimal point. Two spaces are needed on the left of the decimal point (for the largest number in absolute value) and six places are needed after the decimal point (for the smallest number in absolute value). In this example, if the maximum width (m)=10, the actual width would be 9. On the other hand, if (m) were less than 9, the actual width would be the minimum width required to print the numbers in floating-point form; in this case 9. The value of (m) should always be greater than or equal to (s)+5. When numbers are printed in floating-point form, the first blank space is used to print an asterisk.

In a normal PRINT instruction: (s)=8, (m)=13 and (b)=15 minus actual width.

In the last three options, the normal column heading "COLUMN (C) " is used if the column width is at least 12. If the width is less than 12, but greater than 6, the column number (C) is printed, but the word "COLUMN" is omitted. If the column width is less than 6, no column heading is given.

Additional flexibility is provided in the last two options by allowing the user to print numbers in either floating-point, fixed or integer form. Values of the arguments (s) and (m) should be set as follows:

Argument	Floating-point	Fixed	Integer
(s)	number of digits	no. of decimal places	0
(m)	0	-m	-(m+1)

To obtain numbers in fixed or integer form, (m) should be large enough so that the largest number can be printed in the allotted space. Asterisks are not printed when floating-point numbers are requested. When numbers are printed as integers, a single blank space appears on the right. An example is given below.

Suppose the numbers -1.2345678, 12.345678, 123.45678, and 1234.5678 are in row 1 of columns 31 through 34 and NRMAX=1. The instruction:

```
PRINT 4.0 cols 31, 8, 0 FLOATING 32, 4, -13 FIXED 33, 8, 22 READABLE 34, 0, -9 INTEGER
```

would give the following result. Digits are printed under the result so that the reader can see how many blank spaces are printed and where they are located.

```
                        OMNITAB   11/24/70   TEST PRINT OPTION

COLUMN   31     COLUMN   32          33           34

-1.2345678+00        12.3457    123.45678       1235

12345678901234567890123456789012345678901234567890123456
```

The fourth option has been used, so (b)=3. The total column widths, including blanks, are 16, 16, 13, and 12. (The total width is 57.) Note, the word COLUMN does not appear above columns 33 and 34 because the widths, excluding blanks, are only 9 and 10 respectively.

1.7 Formatted Printing And Reading.

ABRIDGE "L", FORMAT "L", NPRINT "L", PRINT "L", READ "L"

For added flexibility, these commands are available for printing and reading data according to the user's own format. They are intended for use by anyone with a knowledge of the FORTRAN language. Their use is not recommended for the novice. See section B4.4 for a discussion of the use of formats. Six additional commands are described in sections 1.8, 1.9 and 1.10.

Each instruction with a qualifier "L" must be preceded by the corresponding command FORMAT "L". If the qualifiers do not agree, the following fatal error occurs with READ "L":

*** FORMAT NOT FOUND

and the following informative diagnostic appears with any print command:

* FORMAT NOT FOUND. READABLE FORMAT IS USED.

The HEAD instruction described in section 1.4 is ignored when printing according to a specified format. The options described in section 1.6 are unavailable.

Remark.

When a format is used, values for all column numbers are either read or printed. This means that the format may pertain to more than one card or line. For example, for the instructions

```
FORMAT A (20F4.0/20F4.0/5F4.0)
READ A 9 cards into cols 1 *** 45
       (27 cards)
```

the data on card 1 would go into columns 1 - 20, the data on card 2 would go into columns 21 - 40 and the data on card 3 would go into columns 41 - 45. The remaining 24 = 3x8 cards would be read in a similar manner. Note, the first argument of the READ A instruction is 9 not 27.

ABRIDGE "L" format, row (R) of columns (C), (C), ... (C)

Print the designated row of the indicated columns according to the indicated format.

FORMAT "L" ( )

Put regular FORTRAN format specification inside the parentheses. The format specification can not be continued onto the next Hollerith card. Also, the format should specify only one line of printing.

NPRINT "L" format, columns (C), (C), ... (C)

This command bears the same relation to PRINT "L" (below) that NPRINT does to PRINT. Printing does not start on a new page.

```
PRINT "L" format, columns (C), (C), ... (C)
```

Prints in accordance with the specified format. Printing starts on a new page. Most of the special features of PRINT are not available.

```
READ "L" format, (n) cards into columns (C), (C) ... (C)
```

One card is read, in accordance with the specified format, for each row of the indicated columns. Note carefully, this command has one more argument than READ. The first argument determines how many cards are to be read. Note, the data cards after READ "L", unlike READ, are not listed in the LIST OF COMMANDS, DATA AND DIAGNOSTICS.

Unlike READ, READ "L" may be used in the repeat mode. Data to be entered into the worksheet should immediately follow the external (unnumbered) PERFORM which executes the READ "L" command. Extreme caution should be used (see section B2.5) when READ "L" is stored for repeated use.

1.8 Printing Arrays And Matrices.

APRINT, APRINT "L", MPRINT, MPRINT "L"

The commands in this section permit one to print data in any part of the worksheet as long as the rows and columns are consecutive. Printing does not have to start with row 1 and one can print from below NRMAX. Although the array and matrix operation commands are described in sections 7 and 8, the printing commands are described here.

Each of the four instructions has four arguments. The first two arguments determine the location of the array (or matrix) in the worksheet by specifying the row-column location of the value in the upper left-hand corner of the rectangular array. The last two arguments determine the size of the array (or matrix) by giving the number of rows and number of columns.

Each of the four commands causes printing to start where the last printing ended and does not start printing on a new page. For the commands APRINT "L" and MPRINT "L", see section 1.7 for further discussion on the use of formats. The HEAD command is ignored by all four of the commands described here.

An error in any one of the four arguments, which defines an array (matrix) that will not fit in the worksheet, will produce the following fatal error:

*** DEFINED MATRIX OVERFLOWS WORKSHEET

The commands APRINT and MPRINT are similar, but have a few minor differences. The commands APRINT "L" and MPRINT "L" are synonomous.

```
APRINT the array in (R),(C) of size (r)x(c)
```

Printing is according to "readable form" unless FIXED or FLOATING is in effect. No headings are supplied. If the number of columns (c) exceeds 8, the array is printed in blocks of 8 columns (or less). A space is inserted between each block.

APRINT "L" format, the array in (R),(C) of size (r)x(c)

Print the specified array in accordance with the prescribed FORMAT "L".

MPRINT the matrix in (R),(C) of size (r)x(c)

MPRINT differs from APRINT in two respects. First, numbers are printed in readable form only if every number in the matrix can be printed without using floating-point notation. If the range in the magnitudes of the values forces some values to be printed in floating-point form, then all numbers are printed in floating-point form. Second, row/column number headings are provided as in the example which follows. Since space is required for headings, the matrix is printed in blocks of seven (or less) rather than 8.

Suppose, in columns 41 to 44, the numbers 11 to 14 are in row 6, the numbers 21 to 24 are in row 7 and the numbers 31 to 34 are in row 8. Then the instruction

MPRINT 6,41 size 3x4

would yield

ROW/COL	41	42	43	44
6	11.000000	12.000000	13.000000	14.000000
7	21.000000	22.000000	23.000000	24.000000
8	31.000000	32.000000	33.000000	34.000000

MPRINT "L" format. the matrix in (R),(C) of size (r)x(c)

MPRINT "L" is a synonym for APRINT "L".

1.9 Punching Data Onto Cards.

PUNCH, PUNCH "L"

Data can be punched onto cards using either of the above commands.

PUNCH data in columns (C), (C) ... (C) onto Hollerith cards

Provides NRMAX cards with numbers punched according to floating-point notation with seven significant digits. A maximum of four columns can be punched at one time. If more columns are needed, PUNCH "L" should be used. The first column of data punched appears in the first 15 card columns, the next column of data in card columns 16 to 30, and so on up to a maximum of 60 card columns. Each field of 15 characters constists of (i) three blanks, (ii) a minus sign if the number is negative, otherwise an additional blank, (iii) the first significant digit, (iv) the decimal point, (v) the last six significant digits, (vi) a plus or minus sign denoting sign of the exponent of 10, and (vii) two digits giving the power of ten. The commands FIXED and FLOATING govern the operation of PUNCH.

PUNCH "L" format, data in columns (C), (C) .... (C)

Data in the specified columns is punched according to the indicated format. Unlike the command PUNCH, this command allows the punching of more than four columns.

1.10 Use Of Magnetic Tapes.

BACKSPACE TAPE "L",	CREAD TAPE "L",	CREAD TAPE "L" "L",	CSET TAPE "L",
ENDFILE TAPE "L",	READ TAPE "L",	READ TAPE "L" "L",	REWIND TAPE "L",
SET TAPE "L",	SKIP TAPE "L",	WRITE TAPE "L",	WRITE TAPE "L" "L"

Each tape operation command has two words and at least one qualifier. The second word is always the word TAPE. The qualifier, see section B1.16, is always any one of the six letters A, B, C, D, E, or F (without quotation marks) and refers to the tape unit being used. If there are two qualifiers, the first qualifier refers to the tape unit and the second qualifier refers to the corresponding FORMAT "L". Five of the instructions use the argument (n) records. Here, a record is 80 characters or one complete Hollerith card image.

If the tape unit specified is anything other than one of the letters A through F, the following fatal error occurs:

*** INCORRECT TAPE UNIT. COMMAND IS NOT EXECUTED

If two qualifiers are used and the instruction FORMAT "L" with the same qualifier has not been used before, then the following fatal error occurs with input commands (read, set):

*** FORMAT NOT FOUND

and the following informative diagnostic is given with the WRITE TAPE commands:

* FORMAT NOT FOUND. READABLE FORMAT IS USED.

(See section 1.7.)

As in section 1.2, input instructions (CREAD TAPE, CSET TAPE, READ TAPE and SET TAPE) may affect the value of NRMAX.

When using tape operation commands with the NBS computer an ASG control card must be used for tape assignment. Users of other computers should check with their computer services staff to find out what control cards are needed, if any. The ASG card must appear immediately after the RUN card (see section B1.4). It has the form

```
$WS ASG "L"
```

or

```
$WS ASG "L"=USERTP
```

where "L" is the appropriate tape unit (one of the letters A through F) and USERTP is the name (six characters) or number of the magnetic tape reel which is to be used. If the second form is not used, then a scratch tape is used. The option W, after the multiple 7-8 punch, permits tape writing. The option S indicates that the tape is to be saved.

The following commands may not be used in the REPEAT MODE (see section B2.5): CREAD TAPE "L", CSET TAPE "L", READ TAPE "L", and SET TAPE "L". When incrementing tape operation commands, which have been stored, the first qualifier (tape unit) should be incremented by zero (without a decimal point).

BACKSPACE TAPE "L" unit, (n) records

This instruction backspaces tape "L" by (n) physical tape records.

```
CREAD TAPE "L" unit, using (n) records, into columns (C), (C), ... (C)
```

This command is similar to the READ TAPE "L" command; except the value (n) specifies the number of records to be read instead of terminating input with a record of zeros as in READ TAPE "L".

```
CREAD TAPE "L" "L" unit and format, using (n) records, into cols (C) ... (C)
```

Similar to command above; except a FORMAT "L" is referenced.

```
CSET TAPE "L", using (n) records, into column (C)
```

This command is similar to the SET TAPE command (see below). Here, the indicated number of records is read, whereas the SET TAPE command reads data until a record of zeros is encountered.

```
CSET TAPE "L" unit, using (n) records, into row (R) of column (C)
```

Similar to above command; except data is entered into row (R) and below rather than starting with row 1.

```
ENDFILE TAPE "L" unit
```

This instruction writes an end-of-file mark on the specified tape, "L". The use of the instruction is highly recommended, if the tape is to be reused.

```
READ TAPE "L" unit into columns (C), (C) ... (C)
```

This command is similar to the ordinary READ command; except data is read from the specified tape unit rather than from cards. Reading of data continues until a record of zeros is encountered; not a blank record.

```
READ TAPE "L" "L" unit and format into columns (C), (C), ... (C)
```

Same as above; except tape is read using the specified format.

```
REWIND TAPE "L" unit
```

The tape specified by "L" is rewound to the beginning of the tape position.

SET TAPE "L" unit into column (C)

This command is similar to the SET command; except data is read from the specified tape unit rather than cards. Reading of data continues until a record of zeros is encountered; not a blank record.

SET TAPE "L" unit starting with row (R) of column (C)

Same as above; except first datum is entered into row (R) instead of row 1.

SKIP TAPE "L" unit, forward (n) records

The specified tape "L" is moved forward (n) physical records.

WRITE TAPE "L" unit from columns (C), (C) ... (C)

This command is similar to the PUNCH command; except data is recorded on magnetic tape rather than cards. Each card image, as in PUNCH, is set up as an output record on the tape. NRMAX records are written as usual and in addition a record of zeros is written at the end. As in PUNCH, the maximum number of arguments (column numbers) is four.

WRITE TAPE "L" "L" unit and format, from columns (C), (C) ... (C)

Same as above; except records are written according to the referenced format.

2. ARITHMETIC OPERATIONS.

This section describes the instructions for: performing simple arithmetic, the use of logarithms, computing trigonometric functions, data summarization, and performing complex arithmetic. Many of these instructions can produce arithmetic diagnostics. See section B3.3 for further details.

None of the instructions in this section have any effect on the value of NRMAX. Be sure that data has been entered into the worksheet by using one of the instructions described in section 1.

2.1 Simple Arithmetic (3 Arguments).

ADD, DIVIDE, MULTIPLY, RAISE, SUBTRACT

Each of these commands has three arguments. The first two arguments may be either constants or column numbers. The third argument is always a column number.

ADD (E) to (E) and put results in column (C)

Performs the indicated addition, row by row.

DIVIDE (E) by (E) and put the results in column (C)

If division by zero is attempted, the result is set equal to zero and the following arithmetic diagnostic is given:

** DIVISION BY ZERO, RESULT SET=0, (n) TIMES

The command DIV is an acceptable abbreviation of DIVIDE.

MULTIPLY (E) by (E) and put the results in column (C)

Performs indicated multiplication. The command MULT is an acceptable abbreviation of MULTIPLY.

RAISE (E) to (E) and put in column (C)

Either of the first two arguments can be negative, zero or positive. However, if the first argument is less than zero and the second argument is not an integer, or if the first argument is zero and the second argument is negative and not an integer, the exponent is set equal to zero, the result is set equal to one, and the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG OR RAISE

The instruction

RAISE -2.0 to -3.0 and put in column 45

is valid and puts -1/8 into column 45. But the instruction

RAISE -2.0 to -3.5 and put in column 45

would produce the above arithmetic fault and diagnostic.

Remember, zero raised to any power is zero and any non-zero value raised to the zero power is one. (Also, 1.0 raised to any power is 1.0.)

RAISE can produce numbers which are too big (overflow) or too small (underflow). Let E1 and E2 be any values specified by the first two arguments. If E2 is a positive integer, then the absolute value of E1 raised to E2 must not exceed the largest number allowed in the computer (i.e., about 10^{38} for the NBS computer). Otherwise, overflow will occur. If E2 is not an integer and E1 is positive, overflow will also occur when E2xlog(E1) exceeds 88.0. If E2 is an integer greater than or equal to 60, then E1 must be positive.

SUBTRACT (E) from (E) and put in column (C)

Carefully note the use of "from" in the instruction which indicates the direction of the subtraction. The instruction

SUBTRACT the value 4.0 from 7.0 and put in column 26

would put the value 3.0 into column 26, whereas the instruction

SUBTRACT the value 7.0 from 4.0 and put in column 26

would put the value -3.0 into column 26. The command SUB is an acceptable abbreviation of SUBTRACT.

2.2 More Simple Arithmetic.

ABSOLUTE, CHANGE, SQRT, SQUARE

The instructions in this section do not have three arguments. ABSOLUTE, SQRT and SQUARE have two arguments. The first argument can be either a column number or a constant. The second argument is always a column number. CHANGE has a variable number of arguments, which are all column numbers.

ABSOLUTE value of (E) put in column (C)

All values are made positive (non-negative) and put in the designated column. If any value is positive or zero, it remains unchanged. If any value is negative, it is multiplied by -1.0 to make it positive. The command ABS can be used as an abbreviation of ABSOLUTE.

CHANGE the sign of values in columns (C), (C) ... (C)

This command has the effect of multiplying each designated column by -1.0. Hence, the instruction

CHANGE the sign of columns 17, 36 and 48

would be equivalent to the three instructions

```
MULTIPLY 17, -1.0, 17
MULTIPLY 36, -1.0, 36
MULTIPLY 48, -1.0, 48
```

Note, this is the only instruction in section 2 which can perform an operation on several columns simultaneously. (See section B2.8 for a complete list of similar instructions.) All the arguments in this instruction are column numbers. Note also, that the results are put back into the designated columns and not into new columns.

SQRT of (E) put in column (C)

Computes the square root (SQRT) of a number or column of numbers. If any number is less than zero, the result is set equal to zero and the following arithmetic diagnostic is given:

```
** NEGATIVE ARGUMENT TO SQRT, LOG OR RAISE
```

The instruction

```
SQRT of 25.0 put in column 7
```

would put the value 5.0 into each row of column 7 and is equivalent to the (more complicated) instruction

```
RAISE 25.0 to the 0.5 power and put in column 7
```

SQUARE of (E) put in column (C)

Multiplies the first argument by itself. The instruction

```
SQUARE column 1 and put in column 2
```

is equivalent to both of the instructions

```
MULTIPLY 1, 1, 2
```

and

```
RAISE 1, 2.0, 2
```

2.3 Logarithms, Exponentiation.

ANTILOG, EXPONENTIAL, LOGE, LOGTEN, NEGEXPONENTIAL

Each of these commands has two arguments. The first argument can be either a constant or a column number. The second argument is always a column number.

ANTILOG of (E) put in column (C)

This is the inverse function of LOGTEN, below. If $y = \log_{10}(x)$, then x is the antilogarithm of y.

```
: EXPONENTIAL of (E) put in column (C)                                    :
```

Computes e^x, for x = (E). (The value of x may be negative.) The command EXP can be used as an abbreviation of EXPONENTIAL. If any value of x exceeds 88.0 (in NBS UNIVAC 1108 computer), overflow occurs. The result is set equal to zero and the following diagnostic is given:

** EVALUATION OF EXPONENT PRODUCES OVERFLOW

The EXPONENTIAL instruction can be considered the antilogarithm (to base e) of LOGE (below).

```
: LOGE of (E) put in column (C)                                          :
```

Computes the natural logarithm or logarithm to the base e. If an attempt is made to compute the natural logarithm of zero or a number less than zero, the result is set equal to zero and the following arithmetic diagnostic given:

** NEGATIVE ARGUMENT TO SQRT, LOG OR RAISE

The command LOG is an acceptable abbreviation for LOGE. It should not be mistaken as an abbreviation for LOGTEN.

```
: LOGTEN of (E) put in column (C)                                        :
```

Computes the common logarithm (logarithm to the base ten). If any value is less than or equal to zero, the result is set equal to zero and the following arithmetic diagnostic given:

** NEGATIVE ARGUMENT TO SQRT, LOG OR RAISE

```
: NEGEXPONENTIAL of (E) put in column (C)                                 :
```

Similar to the EXPONENTIAL instruction above; except the exponent is -(E). The instructions

EXPONENTIAL of -3.6 put in column 54

and

NEGEXPONENTIAL of 3.6 put in column 54

are equivalent.

2.4 Trigonometric Functions.

COS , COT , SIN , TAN	- Radians
COSD, COTD, SIND, TAND	- Degrees
ACOS , ACOT , ASIN , ATAN	- Inverse in radians
ACOSD, ACOTD, ASIND, ATAND	- Inverse in degrees
COSH, COTH, SINH, TANH	- Hyperbolic
ACOSH, ACOTH, ASINH, ATANH	- Inverse hyperbolic

The basic trigonometric functions available are cosine (COS), cotangent (COT), sine (SIN) and tangent (TAN). There are five additional variations of each basic command. If the argument of a trigonometric function is in degrees, the letter D is appended to the command. For the evaluation of the inverse trigonometric functions, the letter A must precede and be attached to the command. The letter H appended to a command will indicate that the hyperbolic function is requested.

The following three arithmetic faults are possible in the execution of these instructions:

```
** ARGUMENT OUT OF BOUNDS TO INVERSE FUNCTION
** ARGUMENT TOO LARGE FOR SIN OR COS, ZERO RETURNED (n) TIMES
** TRIG FUNCTION NOT DEFINED RESULTS SET=0, (n) TIMES
```

The NBS computer (UNIVAC 1108) bound for cos(x), cot(x), sin(x) and tan(x) is $|x| < 10^7$.

Each instruction has two arguments. The first argument can be either a constant or a column number. The second argument is always a column number. In the descriptions below, let x be a number determined by the first argument in an instruction and let y by the corresponding number in column (C). Here, log is used to imply log to the base e.

ACOS of (E) put in column (C)

The principal value (between 0 and $\pi/2$ radians) of arc cosine is computed; $y = \cos^{-1}(x)$.

ACOSD of (E) put in column (C)

The principal value (between 0 and 90 degrees) of arc cosine is computed.

ACOSH of (E) put in column (C)

Computes the inverse hyperbolic cosine; $y = \log\{x + \sqrt{(x^2 - 1)}\}$.

ACOT of (E) put in column (C)

The principal value (between $-\pi/2$ and $+\pi/2$ radians) of arc cotangent is computed; $y = \cot^{-1}(x)$.

ACOTD of (E) put in column (C)

The principal value (between -90 and +90 degrees) of arc cotangent is computed.

ACOTH of (E) put in column (C)

Computes the inverse hyperbolic cotangent of (E); $y = \frac{1}{2}\log\{(x+1)/(x-1)\}$.

ASIN of (E) put in column (C)

The principal value (between 0 and $\pi/2$ radians) of arc sine is computed; $y = \sin^{-1}(x)$.

ASIND of (E) put in column (C)

The principal value (between 0 and 90 degrees) of arc sine is computed.

ASINH of (E) put in column (C)

Computes the inverse hyperbolic sine of (E); $y = \log\{x + \sqrt{(x^2 + 1)}\}$.

ATAN of (E) put in column (C)

The principal value (between $-\pi/2$ and $+\pi/2$) of arc tangent is computed; $y = \tan^{-1}(x)$.

ATAND of (E) put in column (C)

The principal value (between -90 and +90 degrees) of arc tangent is computed.

ATANH of (E) put in column (C)

Computes the inverse hyperbolic tangent of (E); $y = \frac{1}{2}\log((1+x)/(1-x))$.

COS of (E) put in column (C)

Computes the cosine of angle(s) in radians; $y = \cos(x)$.

COSD of (E) put in column (C)

Computes the cosine of angle(s) in degrees.

COSH of (E) put in column (C)

Computes the hyperbolic cosine of (E); $y = \frac{1}{2}(e^x + e^{-x})$.

COT of (E) put in column (C)

Computes the cotangent of angle(s) in radians; $y = \cot(x) = \cos(x)/\sin(x)$.

COTD of (E) put in column (C)

Computes the cotangent of angle(s) in degrees.

COTH of (E) put in column (C)

Computes the hyperbolic cotangent of (E); $y = (e^{x} + e^{-x})/(e^{x} - e^{-x})$.

SIN of (E) put in column (C)

Computes the sine of angle(s) in radians; $y = \sin(x)$.

SIND of (E) put in column (C)

Computes the sine of angle(s) in degrees.

SINH of (E) put in column (C)

Computes the hyperbolic sine of (E); $y = \frac{1}{2}(e^{x} - e^{-x})$.

TAN of (E) put in column (C)

Computes the tangent of angles in radians; $y = \tan(x) = \sin(x)/\cos(x)$.

TAND of (E) put in column (C)

Computes the tangent of angle(s) in degrees.

TANH of (E) put in column (C)

Computes the hyperbolic tangent of (E), $y = (e^{x} - e^{-x})/(e^{x} + e^{-x})$.

2.5 Triple Operations.

All the commands in sections 2.1, 2.2 (except CHANGE), 2.3 and 2.4 have an additional form with two additional arguments for triple operations. (Hence, commands in section 2.1 will have 5 arguments, whereas those in sections 2.2, 2.3 and 2.4 will have 4 arguments.) Also, there are two commands in section 2.6, FRACTIONAL and INTEGER, which have this added form. The three operations are:

1. The operation specified by command (e.g., DIVIDE)
2. Multiplication
3. Addition

For example, if the numbers 1, 2 and 3 are in column 17, the instruction

SQUARE column 17, multiply by 2.0, add 1.2, put in column 18

would put the numbers 3.2, 9.2 and 19.2 into column 18. This instruction is equivalent to the three instructions:

```
SQUARE 17, 18
MULTIPLY 18, 2.0, 18
ADD 1.2, 18, 18
```

The abbreviations described in the other sections can also be used in the triple operation instructions.

If the number of arguments in an instruction is incorrect (should be 3 or 5 for those in section 2.1 and 2 or 4 for those in sections 2.2, 2.3 and 2.4), the following fatal error occurs:

*** (n) IS AN ILLEGAL NUMBER OF ARGUMENTS

Since all triple operation instructions bear the same relationship to the single operation instructions described previously, all the optional forms are merely listed on the next page (alphabetically) without a description. The five argument instructions are separated from the four argument instructions. The first instruction in each group is boxed as usual, but the remaining instructions are just listed.

5 Arguments

ADD	(E) to (E), multiply by (E), add (E), put in column (C)
DIVIDE	(E) by (E), multiply by (E), add (E), put in column (C)
MULTIPLY	(E) by (E), multiply by (E), add (E), put in column (C)
RAISE	(E) to (E), multiply by (E), add (E), put in column (C)
SUBTRACT	(E) from (E), multiply by (E), add (E), put in column (C)

4 Arguments

ABSOLUTE value	of (E), multiply by (E), add (E), put in column (C)
ACOS	of (E), multiply by (E), add (E), put in column (C)
ACOSD	of (E), multiply by (E), add (E), put in column (C)
ACOSH	of (E), multiply by (E), add (E), put in column (C)
ACOT	of (E), multiply by (E), add (E), put in column (C)
ACOTD	of (E), multiply by (E), add (E), put in column (C)
ACOTH	of (E), multiply by (E), add (E), put in column (C)
ANTILOG	of (E), multiply by (E), add (E), put in column (C)
ASIN	of (E), multiply by (E), add (E), put in column (C)
ASIND	of (E), multiply by (E), add (E), put in column (C)
ASINH	of (E), multiply by (E), add (E), put in column (C)
ATAN	of (E), multiply by (E), add (E), put in column (C)
ATAND	of (E), multiply by (E), add (E), put in column (C)
ATANH	of (E), multiply by (E), add (E), put in column (C)
COS	of (E), multiply by (E), add (E), put in column (C)
COSD	of (E), multiply by (E), add (E), put in column (C)
COSH	of (E), multiply by (E), add (E), put in column (C)
COT	of (E), multiply by (E), add (E), put in column (C)
COTD	of (E), multiply by (E), add (E), put in column (C)
COTH	of (E), multiply by (E), add (E), put in column (C)
EXPONENTIAL	of (E), multiply by (E), add (E), put in column (C)
FRACTIONAL part	of (E), multiply by (E), add (E), put in column (C)
INTEGER part	of (E), multiply by (E), add (E), put in column (C)
LOGE	of (E), multiply by (E), add (E), put in column (C)
LOGTEN	of (E), multiply by (E), add (E), put in column (C)
NEGEXPONETIAL	of (E), multiply by (E), add (E), put in column (C)
SIN	of (E), multiply by (E), add (E), put in column (C)
SIND	of (E), multiply by (E), add (E), put in column (C)
SINH	of (E), multiply by (E), add (E), put in column (C)
SQRT	of (E), multiply by (E), add (E), put in column (C)
SQUARE	of (E), multiply by (E), add (E), put in column (C)
TAN	of (E), multiply by (E), add (E), put in column (C)
TAND	of (E), multiply by (E), add (E), put in column (C)
TANH	of (E), multiply by (E), add (E), put in column (C)

2.6 Data Summarization.

ACCURACY, FRACTIONAL, INTEGER, ROUND	- Numbers
PARSUM, ROW SUM, SUM	- Sums
EXPAND, PARPRODUCT, PRODUCT	- Products
AVERAGE, MAXIMUM, MINIMUM, RMS	- Properties

Numbers

ACCURACY of (E) compared with (E) put in column (C)

This instruction is used to see how closely two (sets of) numbers agree. In ACCURACY X, Y, Z

$$Z = -\text{LOGTEN}\ |(X-Y)/Y|, \quad \text{if } X \neq Y \text{ and } Y \neq 0$$
$$Z = -\text{LOGTEN}\ |X-Y|, \quad \text{if } X \neq Y \text{ and } Y = 0$$
$$Z = 8, \quad \text{if } X = Y$$

But Z is never less than -8.0 or greater than +8.0 (in the NBS computer).

The instruction gives a measure of the number of leading digits in X which are the same as the number of leading digits in Y. If the numbers 1.2347680, 1.2345378, 2.2234568, 1.2345678, 1.2345679, -1.2345678, 76.234567, 2.4691356 and 0.0 are in column 31, the instruction

ACCURACY of column 31 compared to 1.2345678 put in column 32

would put the numbers 3.7900571, 4.6144509, .096367388, 8.0000000, 7.0731966, -.30102999, -1.7835463, 0.0 and 0.0 into column 32. This instruction will give different answers from the instruction

ACCURACY of 1.2345678 compared to column 31 put in column 32

The answers here would be 3.7901275, 4.6144404, .35188114, 8.0000000, 7.0731966, -.30102999, .0070906878, .30102999 and -.091514938. A negative result indicates the two values being compared either do not agree in the first digit or they differ with respect to sign. An example of two numbers which do not agree in their leading digit (and actually differ in order of magnitude) is given above by X = 76.234567 and Y = 1.2345678; the accuracy reported is -1.7835463. An example of two numbers which do not agree in sign is X = -1.2345678 and Y = 1.2345678; here the accuracy reported is -.30102999.

This instruction was added to aid the developers of OMNITAB in testing the accuracy of instructions, but it is available for use by all users. The above description of accuracy was used extensively by R. H. Wampler in "A report on the accuracy of some widely used least squares computer programs.", J. Am. Statist. Assoc., 65, 549-565 (1970). It is also used by the FIT and POLYFIT instructions.

FRACTIONAL part of (E) put in column (C)

The portion of each value to the left of the decimal point is dropped and the remainder is put in the designated column. The sign of each number is kept. The instruction

FRACTIONAL part of column 7 put in column 8

applied to the numbers 7.35, -12.82, 26.00 and 0.96 in column 7, would put 0.35, -0.82, 0.00 and 0.96 into column 8. The instruction is the complement of INTEGER, below.

INTEGER part of (E) put in column (C)

INTEGER is the complement of the previous instruction, FRACTIONAL. The portion of each value on the right of the decimal point is chopped and the remainder is put in the designated column. For the above numbers in column 7, namely 7.35, -12.82, 26.00 and 0.96, the instruction

INTEGER part of column 7 put in column 9

would put the numbers 7., -12., 26. and 0. into the first four rows of column 9.

ROUND the numbers in column (C) to (n) digits and put in column (C)

The numbers are rounded to the specified number of digits in accordance with the standard rules for rounding. The argument (n) must be an integer from 1 to 8.

(1) If the (n+1)st digit is less than 5, the portion beyond the nth digit is dropped and the nth digit is unchanged. E.g., 1234.5678 rounded to 3 digits is 1230.

(2) If the (n+1)st digit is greater than 5, the nth digit is increased by 1. Or, if the (n+1)st digit equals 5 and any digit on the right is not zero, the nth digit is increased by 1. E.g., 1234.5678 rounded to 5 digits is 1234.6.

(3) If the (n+1)st digit is 5 and all digits on the right are zero, the nth digit is rounded to the nearest even digit, i.e., if the nth digit is even, it is unchanged and if the nth digit is odd, it is increased by 1. The number 123.4500 rounded to 4 digits is 123.4, whereas the number 986.75 rounded to 4 digits is 986.8.

*****

Sums

PARSUM column (C) and put in column (C)

The partial sums of the first named column are put in each row of the second named column. If the numbers 1, 7, 0, 3 and 5 are in column 1, the instruction

PARSUM 1,2

would put the numbers 1, 8, 8, 11 and 16 into the first five rows of column 2.

ROW SUM columns (C), (C), ... (C) and put in column (C)

The instruction must have at least 4 arguments. Each row of the last named column contains the sum of the numbers in the corresponding row of the other columns. For the following numbers:

COLUMN 2	COLUMN 4	COLUMN 1	COLUMN 6	COLUMN 11
2.0	1.0	3.0	2.0	8.0
3.0	4.0	0.0	3.0	10.0
5.0	2.0	4.0	1.0	12.0

the numbers in column 11 would result from using the instruction

ROW SUM columns 2, 4, 1 and 6 and put in column 11

The command ROWSUM (one word) is synonymous with ROW SUM (two words) both here and in the two options which follow.

ROW SUM columns (C) through (C) and put in column (C)

This instruction performs the same operation as the one above. It has exactly 3 arguments (all column numbers). The instruction finds the sum, row by row, for all columns between the first named column and the second named column, inclusive, and puts the result in the third named column. In the example above, if columns 4, 1 and 6 were changed to 3, 4 and 5, the instruction

ROW SUM columns 2 through 5 and put in column 11

would give the same results in column 11.

ROW SUM the entire worksheet and put results in column (C)

Each row of the worksheet is summed and put in the corresponding row of the named column. For example, row 17 of the named column (if NRMAX is greater than 16) would contain the sum of the numbers in all 62 columns of row 17 (unless DIMENSION has been used) of the worksheet, including row 17 of the named column. The instruction

ROW SUM 23

is equivalent to the instruction

ROW SUM 1,62,23

and also to the instruction

ROW SUM 1 *** 62, 23

SUM the rows of column (C) and put in column (C)

Finds the sum of numbers in the first named column and puts the single result into every row of the second named column. If the numbers 2.0, 7.2 and 1.3 are in column 34, the instruction

SUM column 34 and put in column 56

would put 10.5, 10.5 and 10.5 into column 56. Two additional forms of SUM are described below.

SUM column (C), rows (R) through (R), and put in column (C)

This instruction is the same as the one above; except only the indicated consecutive rows are summed. If the values 6.3, 2.0, 7.2, 1.3 and 4.8 are in column 11, the instruction

SUM column 11, rows 2 through 4, and put in column 12

would put 10.5, 10.5, 10.5, 10.5 and 10.5 into the first five rows of column 12. The second argument does not have to be less than the third argument, i.e. the following instruction would be equivalent to the previous one

SUM 11, 4, 2, 12

SUM column (C), rows (R), (R), ... (R) and put in column (C)

The instruction must have at least 5 arguments. It is similar to the two instructions above; only the specified rows are summed and they need not be consecutive. The instruction above could be written:

SUM column 11, rows 3, 2 and 4, and put result in column 12

*****

Products

EXPAND (E) to power (p) in increments (i) and put in column (C) and succ. cols

Provides for the exponentiation of a constant or column to the integral power (p) in equal, integral steps (i). Results are put in successive columns starting with the column designated by the 4th (last) argument. The number of columns used is the same as the number of steps, which is (p)/(i). The exponent is always equal to the increment size (i). The instruction

EXPAND 2.0 to the power 8 in increments of 2 and put in column 11

would put the values 4., 16., 64. and 256. into each row of columns 11 through 14. The number of columns used for storing results is 8/2 = 4. An additional form of the instruction is described below.

Caution: The power (p) should be an integral multiple of the increment (i). If not, an additional increment is used. E.g., the instruction

EXPAND 2.0, 7, 2, 11

would produce the same answers as the instruction above.

EXPAND (E) to power (K) in steps of (K) and put in col (C) and successive cols

Same as above, but the power and the increment do not have to be an integer. The instruction

EXPAND 4.0, 2.5, 0.5, 11

would put the 0.5, 1.0, 1.5, 2.0 and 2.5 powers of 4.0, namely 2., 4., 8., 16. and 32., into each row of columns 11 through 15. The number of columns used for storing results is 2.5/0.5 = 5.

Caution: The quotient of the power (2nd argument) and the increment size (3rd argument) should be an integer. If the quotient is not an integer, an additional step is taken by the instruction, but no diagnostic is given. For example, the instruction

EXPAND 4.0, 2.7, 0.5, 11

would be performed like

EXPAND 4.0, 3.0, 0.5, 11

PARPRODUCT of column (C), put partial products in column (C)

Puts partial products of successive rows of the 1st named column into the 2nd named column. If the numbers 1., 3., 2., 5., 0. and 4. are in column 23, the instruction

PARPRODUCT of column 23 put in column 24

would put the numbers 1., 3., 6., 30., 0. and 0. into the first six rows of column 24. If any product exceeds the capacity of the computer, zero is returned for that particular row and all subsequent rows. No diagnostic is given. If the numbers in the first named column are consecutive integers (starting with 1 or 2), the nth row will contain n factorial.

PRODUCT row by row of cols (C), (C), ... (C) put into column (C)

The instruction must have at least four arguments, all column numbers. For the following numbers

COLUMN 2	COLUMN 4	COLUMN 1	COLUMN 6	COLUMN 11
2.0	1.0	3.0	2.0	12.0
3.0	4.0	0.0	3.0	0.0
5.0	2.0	4.0	1.0	40.0

the numbers in column 11 would be the result of using the instruction

PRODUCT row by row of columns 2, 4, 1 and 6 put in column 11

PRODUCT row by row of columns (C) through (C) put in column (C)

This instruction performs the same type of operation as the one above. It has exactly three arguments. The instruction finds the product of the the numbers in each row of the consecutive columns from the 1st named column to the 2nd named column, inclusive, and puts the results into the 3rd named column. If columns 4, 1 and 6 were changed to 3, 4 and 5 in the example above, the instruction

PRODUCT row by row of columns 2 through 5 put in column 11

would give the same answers in column 11. The 1st argument should be less than the 2nd argument.

*****

Properties

AVERAGE the values in column (C) and put the result in column (C)

For the n=NRMAX numbers x_i in the 1st named column, the instruction computes

$$\bar{x} = \Sigma_{i=1}^{n} x_i/n$$

and puts the result into each row of the 2nd named column.

MAXIMUM value of the numbers in column (C) put in column (C)

Finds the largest value in the 1st named column and puts the single result into each row of the 2nd named column. The command MAX is an acceptable abbreviation of MAXIMUM, both here and in the option below.

MAXIMUM of col (C) put in col (C), corresp. value of (C) in (C), (C) in (C) ...

If the numbers 3.1, 17.6, 9.2, 11.6 and 8.3 are in column 51 and the numbers 127.8, 92.3, 15.3, 224.7 and 75.4 are in column 61, the instruction

MAXIMUM of col 51 put in col 52, corresponding value of 61 put in col 62

would put the number 17.6 (row 2 of column 51) into each of the five rows of column 52 and the number 92.3, in the corresponding 2nd row of column 61, into the first five rows of column 62. The instruction must have an even number of arguments.

MINIMUM value of the numbers in column (C) put in column (C)

Finds the smallest value in the first named column and puts the single result into each row of the second named column. The command MIN is an acceptable abbreviation of MINIMUM, both here and in the option below.

MINIMUM of col (C) put in col (C), corresp. value of (C) in (C), (C) in (C) ...

If the numbers 3.1, 17.6, 9.2, 11.6 and 8.3 are in column 51 and the numbers 127.8, 92.3, 15.3, 224.7 and 75.4 are in column 61, the instruction

MINIMUM of col 51 put in col 52, corresponding value of col 61 put in col 62

would put the number 3.1 (row 1 of col 51) into each of the five rows of column 52 and the number 127.8, in the corresponding row (1) of column 61, into the first five rows of column 62. The instruction must have an even number of arguments.

RMS of column (C) put in column (C)

For the n=NRMAX values x_i in the 1st named column, the instruction puts the root mean square (RMS)

$$\sqrt{\Sigma_{i=1}^{n} x_i^2/n}$$

into each row of the 2nd named column. For the numbers 1., 2., 3., 4. and 5. in column 28, the instruction

RMS of column 28 put in column 29

would put the value 3.3166248 (= $\sqrt{55}/5 = \sqrt{11}$) into the first five rows of column 29.

2.7 Complex Arithmetic.

CADD, CDIVIDE, CMULTIPLY, CPOLAR, CRECTANGULAR, CSUBTRACT

Any complex number has (in rectangular coordinates) a real and an imaginary part. In OMNITAB, two columns are needed for each set of complex numbers. The first number in each pair is the real part and the second number of the pair is the imaginary part. Otherwise, the arithmetic operations are similar to those in section 2.1. However, triple operation instructions are not available and no abbreviations are allowed. Two commands are available for changing from one coordinate system to the other; CPOLAR and CRECTANGULAR.

The last two arguments of each instruction are always column numbers. The remaining arguments may be either constants or column numbers.

A complex number z can be written as $z = x + iy$, where x is the real part and y is the imaginary part. This notation will be used in the descriptions below. If there are two complex numbers, they will be represented by $z_1 = x_1 + iy_1$ and $z_2 = x_2 + iy_2$. For further details the reader may consult, for example, R. V. Churchill, "Introduction to Complex Variables and Applications", McGraw-Hill Book Company (1948).

To assure that the complex arithmetic instructions give accurate results, internal calculations are done using double precision arithmetic.

CADD real (E) imag (E) to real (E) imag (E) put real in col (C) imag in col (C)

The two (sets) of complex numbers specified by the first four arguments are added and then put in the columns indicated by the last two arguments. If the two numbers added are z_1 and z_2, then the result $z = z_1 + z_2$ is given by:

$$x = x_1 + x_2 \text{ and } y = y_1 + y_2.$$

CDIVIDE real (E) imag (E) by real (E) imag (E) put real in col (C) imag in (C)

Performs the indicated complex division. For $z = z_1/z_2$, $z_2 \neq 0$,

$$x = (x_1x_2 + y_1y_2)/(x_2^2 + y_2^2) \text{ and } y = (x_2y_1 - x_1y_2)/(x_2^2 + y_2^2).$$

If division by zero is attempted, the result is set equal to zero and an arithmetic diagnostic is given.

CMULTIPLY real (E) imag (E) by real (E) imag(E) put real in col (C) imag in (C)

Performs the indicated complex multiplication. For $z = z_1 z_2$,

$$x = x_1x_2 - y_1y_2 \text{ and } y = x_1y_2 + x_2y_1.$$

CPOLAR for x = (E), y = (E) put rho in col (C), theta in col (C)

This command changes complex numbers in the rectangular coordinate system to numbers in the polar coordinate system. For $x = r\cos\theta$ and $y = r\sin\theta$, the instruction puts

$$r = \sqrt{(x^2 + y^2)} \text{ and } \theta = \tan^{-1}(y/x)$$

in the last two columns. The principal value of the arctan is taken between $-\pi/2$ and $+\pi/2$. If $x = 0$, θ is set equal to zero and an arithmetic diagnostic is given.

CRECTANGULAR for rho = (E), theta = (E) put x in col (C), y in col (C)

Changes from polar coordinates to rectangular coordinates. This instruction is the inverse of CPOLAR above, The results

$$x = r\cos\theta \text{ and } y = r\sin\theta$$

are put in the last two columns.

CSUBTRACT real (E) imag (E) from real (E) imag (E) put real in (C) imag in (C)

Performs the indicated subtraction. For $z = z_2 - z_1$, the instruction puts

$$x = x_2 - x_1 \text{ and } y = y_2 - y_1$$

in the last two columns.

3. DATA MANIPULATION.

This section contains five types of instructions for data manipulation which enhance problem solving considerably. Caution. Several of the instructions in this section affect the value of NRMAX. Also, in several instructions it is possible to make a mistake in using an argument, which will define a location outside the worksheet. When this happens an appropriate diagnostic will be given. Some of the instructions in this section, although powerful, are a little tricky to comprehend. A liberal amount of self-teaching can help immensely; see section B4.1.

3.1 Defining Operations.

COUNT, DEFINE, ERASE, RESET, RESET "V"

Care should be exercised in using RESET and the last three forms of DEFINE to avoid using an argument which defines a number of rows which exceeds the number of rows in the worksheet (normally 201) or to define a row number outside the worksheet.

COUNT length of column (C) and put in column (C)

Starting from the bottom, row NRMAX, the instruction searches for the first row of the first named column containing a value which is not equal to zero. The number of this row is the "count", which is put into each row of the second named column. If NRMAX=5 and the numbers 1, 2, 3, 4 and 0 are in column 11, then the instruction

COUNT of column 11 put in column 12

would put the number 4.0 into the first five rows of column 12.

DEFINE (E) into column (C)

If the first argument is a constant, the constant is put into each row of the designated column. The instruction

DEFINE the value 9.6 into column 3

is equivalent to the instruction

ADD the value 0.0 to the value 9.6 and put in column 3

If the first argument is a column number, the instruction simply moves one column of numbers to another column. The instruction

DEFINE column 11 into column 33

is equivalent to the instruction

ADD the value 0.0 to column 11 and put in column 33

and also to the instruction (see section 3.2)

MOVE the array in 1,11 of size **NRMAX** x 1 to 1,33

DEFINE the constant (K) into row (R) of column (C)

The constant (K) is put into a single location in the worksheet. The row number (R) can exceed NRMAX.

DEFINE the value in row (R) of column (C) into column (C)

The number in row (R) of column (C) is put into each row of the second named column. The number may be located below NRMAX, nevertheless it is put in rows 1 to NRMAX of the second named column. If 3.6 is in row 7 of column 11 and NRMAX = 5, the instruction

DEFINE the value in 7,11 into column 12

would put the numbers 3.6, 3.6, 3.6, 3.6 and 3.6 into the first five rows of column 12.

DEFINE the value in row (R) of column (C) into row (R) of column (C)

A single number is moved from any specified location in the worksheet to any other location. A number can be moved from above NRMAX to below NRMAX, or vice versa. The instruction

DEFINE the value in 7,11 into 4,32

is equivalent (see section 3.2) to the instruction

MOVE the array in 7,11 of size 1x1 to 4,32

ERASE columns (C), (C) ... (C)

Zeros are put into each row of the designated columns. The number of arguments is variable. Unlike the option below, which has no arguments, the value of NRMAX is unaffected by the instruction.

ERASE the entire worksheet

Each entry in the entire worksheet is set equal to zero. The value of NRMAX is reset to zero. Unlike an OMNITAB instruction, this is the only initialization that is performed. This instruction may be used at the start of a new subset of instructions when one does not want to destroy any previously used titles, notes or stored instructions.

RESET nrmax to equal (r) rows

Simply resets the value of NRMAX to the specified number of rows. (If the argument is not an integer, it is truncated to an integer.) Sometimes it is wise to use the instruction RESET 0 before using a READ or SET instruction. See, also, section B1.6.

RESET "V" variable equal to (K)

The qualifier "V" can be any one of the letters (without quotation marks) V, W, X, Y or Z. (If the argument in the instruction is an integer (without a decimal point), it is automatically converted to a constant.) See sections B1.7 and B1.16 for further details.

3.2 Moving Data.

DEMOTE, DUPLICATE, EXCHANGE, MOVE, PROMOTE

The commands DUPLICATE and MOVE actually operate on arrays. However, since they are commonly needed and used, they are described here rather than in section 7 along with the regular array operation commands. These commands are often used by those who are unfamilar with the array operation commands described in section 7. See section 7 for further details.

DEMOTE by (r) rows, col (C) into col (C), col (C) into col (C), ... etc.

For each pair of columns, each value in the 1st named column is moved (pushed) down (r) rows and put in the 2nd named column. The first (r) rows of the 2nd named column will remain unchanged. NRMAX is increased by (r). The number of arguments is always odd and at least 3. This instruction is essentially the reverse of PROMOTE, described below. If the numbers 1 to 8 are in column 26 and the numbers 41 to 48 are in column 27, the instruction

DEMOTE by 3 rows column 26 into column 27

would put the numbers 41, 42, 43, 1, 2, 3, 4, 5, 6, 7 and 8 into column 27. NRMAX would be increased by 3 = (r) from 8 to 11. This instruction is actually equivalent to the two instructions

```
MOVE 1,26 size 8x1 to 4,27
RESET 11
```

Caution, the instruction demotes each column (denoted by the even arguments) independently and consecutively. Hence, for the numbers above, the instruction

DEMOTE by 3 rows col 26 into 27, 27 into 27

would put the numbers 41, 42, 43, 41, 42, 43, 1, 2, 3, 4 and 5 into column 27. NRMAX would be 11.

(If the argument (r) is negative, the DEMOTE instruction is synonymous with the instruction using PROMOTE for DEMOTE and r for -r.)

The value of (r) plus NRMAX should not exceed the number of rows in the worksheet (normally 201); otherwise the following informative diagnostic is given

* ATTEMPT TO DEMOTE OFF THE WORKSHEET. SPILL IS LOST.

DEMOTE all values in the worksheet by (r) rows

Similar to the above instruction, but the entire worksheet is demoted by (r) rows and only one argument is used. The results are put back into the original columns. The new

value of NRMAX is the old value of NRMAX plus (r). The numbers in the first (r) rows remain unchanged.

DUPLICATE (t) times the array starting in (R),(C) of size (r)x(c) put in (R),(C)

The two pairs of arguments, after the first argument, designate a rectangular (or square) array of consecutive rows and columns. The first pair gives the row-column location of the number in the upper left-hand corner of the array and the second pair determines the size of the array. The last pair of arguments specify the location of the element in the upper left-hand corner of the new array. The new array will be the original array reproduced (t) times. Each copy of the original array is put immediately below the previous one. Hence, the new array will have (t)x(r) rows and the same number of columns (c) as the first array. The new array may overlap the original array. Unless there is an overlap, the original array remains unchanged. NRMAX can be changed by this instruction. For the following data in the worksheet

Row/Column	11	12	13
7	2.0	0.0	8.0
8	6.0	4.0	-2.0

the instruction

DUPLICATE 3 times the array starting in 7,11 of size 2x3 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	2.0	0.0	8.0
5	6.0	4.0	-2.0
6	2.0	0.0	8.0
7	6.0	4.0	-2.0
8	2.0	0.0	8.0
9	6.0	4.0	-2.0

The above instruction is equivalent to the 3 = (t) MOVE (see next page) instructions

MOVE the array starting in 7,11 of size 2x3 to 4,32
MOVE the array starting in 7,11 of size 2x3 to 6,32
MOVE the array starting in 7,11 of size 2x3 to 8,32

If (t)x(r) exceeds the number of rows in the worksheet, the following fatal error occurs

*** ILLEGAL SIZE ROW NUMBER

EXCHANGE col (C) with col (C), col (C) with col (C), etc.

Interchanges the numbers in the rows of each pair of designated columns. The number of arguments is always even. The exchanges are made on each pair consecutively. Hence, care is needed if all the arguments are not different. For the numbers

Column 1: 4, 2, 7, 3 and 8
Column 2: 11, 17, 19, 12 and 16
Column 3: 125, 123, 128, 126 and 134

the instruction

EXCHANGE column 1 with column 3, and column 3 with column 2

would first exchange columns 1 and 3 and then exchange the new column 3 (the original column 1) with column 2 producing the results:

Column 1:	125, 123, 128, 126 and 134
Column 2:	4, 2, 7, 3 and 8
Column 3:	11, 17, 19, 12 and 16

MOVE the array starting in (R),(C) of size (r)x(c) to the array in (R),(C)

The first four arguments designate a rectangular array of consecutive rows and columns. The first two arguments give the row-column location of the number in the upper left-hand corner of the array. The second two arguments give the size of the array. The last two arguments indicate where the array is to be moved. Arguments are not used to specify the size of the array in the new locations since the size is always the same as that of the original array. The new location may overlap the original array. For the following data in the worksheet

Row/Column	11	12	13
7	2.0	0.0	8.0
8	6.0	4.0	-2.0

the instruction

MOVE the array starting in 7,11 of size 2x3 to 4,32

would give the following result:

Row/Column	32	33	34
4	2.0	0.0	8.0
5	6.0	4.0	-2.0

See, also, the discussion of arrays in section 7 and the description of the synonym AMOVE in section 7.2.

If an attempt is made to move a rectangular array outside the worksheet, a fatal error will result.

PROMOTE by (r) rows, col (C) into col (C), col (C) into col (C), ... etc.

This instruction is essentially the reverse of DEMOTE, which is described above. The number of arguments is always odd and greater than 2. For each pair of columns, each value in the 1st named column is moved up (r) rows into the second named column. The bottom (r) rows of the 2nd named column remain unchanged. The first (r) rows of the 1st named column will not appear in the 2nd named column. The number of values actually moved is NRMAX - (r). If the numbers 1 to 8 are in column 26 and 41 to 48 are in column 27, the instruction

PROMOTE by 3 rows column 26 into column 27

would put the numbers 4, 5, 6, 7, 8, 46, 47 and 48 into column 27.

Caution, the instruction promotes each column (even arguments) independently and consecutively. Hence, for the numbers above and the numbers 31 to 38 in column 28, the instruction

PROMOTE by 3 rows column 26 into column 27, and column 27 into column 28

would give the same results in column 27 and would put the numbers 7, 8, 46, 47, 48, 36, 37 and 38 into column 28.

(If the argument (r) is negative, the PROMOTE instruction is synonymous with the same instruction using DEMOTE for PROMOTE and r for -r.)

If the argument (r) exceeds NRMAX, the following informative diagnostic is given

* ATTEMPT TO PROMOTE FROM BELOW NRMAX. FIRST ARGUMENT IS RESET TO NRMAX

PROMOTE all values in the worksheet by (r) rows

Similar to the above instruction, but the entire worksheet is promoted and when each column is promoted the result is put back into the same column. The instruction has exactly one argument. Zeros are put in the bottom NRMAX-(r) rows of the worksheet.

3.3 Manipulative Operations.

CENSOR, CLOSE UP, FLIP, INSERT, SEPARATE, SHORTEN

The instructions described here enhance problem solving considerably. The CENSOR instruction is related to the instructions described in section 3.5

CENSOR col (C) for values less than or equal to (E), replace by (E), put in (C)

Each number, X, in the 1st named column is compared with the value(s), Y = (E), designated by the 2nd argument. If X is less than or equal to Y, X is replaced by the number designated by the 3rd argument and put in the corresponding row of the column designated by the 4th (last) argument. If X is greater than Y, then X is simply moved to the column designated by the last argument. The original numbers in the 1st column remain unchanged. To censor values larger than a particular value, change the sign of the values, censor for the negative of the desired value, and change the sign back. This instruction has many uses in treating subsets of data or in defining properties of numbers. See section B4.6 for some examples and see also MATCH in section 3.5. For the numbers

Column 11: 2.3, 7.6, 5.2, 8.3 and 4.2
Column 12: 3.4, 5.8, 5.2, 4.7 and 6.3

the instruction

CENSOR col 11 for values less than or equal to col 12, replace by -1.2, put in col 13

would leave columns 11 and 12 unchanged and would put the numbers -1.2, 7.6, -1.2, 8.3 and -1.2 into column 13. Whereas, the instructions

CHANGE sign of columns 11 and 12
CENSOR col 11 for col 12, replace 1.2, put in column 13
CHANGE sign of columns 11, 12 and 13

would leave columns 11 and 12 unchanged and would put the numbers 2.3, -1.2, -1.2, -1.2 and 4.2 into column 13.

```
:-------------------------------------------------------------------------:
:                                                                         :
:  CLOSE UP rows with (K) in columns (C), (C) ... (C)                     :
:                                                                         :
:-------------------------------------------------------------------------:
```

Each row containing the number (K) is moved to the bottom of the column and the number in the row is changed to zero. The numbers, which were below (K), are all moved up one row. Operations are performed on each column independently. For the numbers

Column 3: 2, 1, 3, 2, 1, 0, 3, 1, 4 and 5
Column 4: 1, 4, 5, 0, 6, 1, 7, 2, 3 and 5

the instruction

CLOSE UP rows having 1.0 in columns 3 and 4

would change the numbers in columns 3 and 4 to

Column 3: 2, 3, 2, 0, 3, 4, 5, 0, 0 and 0
Column 4: 4, 5, 0, 6, 7, 2, 3, 5, 0 and 0

```
:-------------------------------------------------------------------------:
:                                                                         :
:  FLIP column (C) into col (C), col (C) into col (C), ... etc.           :
:                                                                         :
:-------------------------------------------------------------------------:
```

In each pair of columns, the second named column is the first named column turned upside down. If the numbers 2, 0, 1, 9 and 3 are in column 17, the instruction

FLIP column 17 into column 18

would put the numbers 3, 9, 1, 0 and 2 into column 18. The instruction

FLIP 17, 18, 18, 18

would leave the original numbers in columns 17 and 18. The number of arguments in the instruction is always even.

```
 /-------------------------------------------------------------------------/
/  FLIP the entire worksheet upside down                                  /
/-------------------------------------------------------------------------/
```

The instruction without an argument turns the entire worksheet upside down. Row 1 goes into row NRMAX, row 2 goes into row NRMAX-1, row 3 goes into row NRMAX-2, etc..

```
:-------------------------------------------------------------------------:
:                                                                         :
:  INSERT into col (C) from (C) at every (i)th row, start with row (R), put in (C)  :
:                                                                         :
:-------------------------------------------------------------------------:
```

The values in the second named column are inserted before rows R, R+i, R+2i, ... etc. of the first named column and the results are stored as indicated. The first two columns remain unchanged. The value of NRMAX is increased by

$$1 + (NRMAX-R+1)/(i-1)$$

However, NRMAX never exceeds the number of rows in the worksheet (normally 201). The value of NRMAX must be at least 2 to begin with. The values of both (i) and (R) must be greater than one. For the numbers (NRMAX=6)

Column 57: 1, 2, 4, 6, 8 and 10
Column 58: 3, 5, 7 and 9

the instruction

INSERT in col 57 from col 58 at every 2nd. row, starting with row 3, put in col 60

would put the numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and 0 into column 60. Here, NRMAX would be increased by 1+(6-3+1)/(2-1) = 1+4/1 = 5. Hence, the new value of NRMAX is 11=6+5.

SEPARATE from column (C) every (r)th row, start with row (R), put in column (C)

Extracts every (r)th row from the first named column, starting with row (R), and puts the result in the column designated by the fourth (last) argument. If the numbers 1 through 10 are in column 1, the instruction

SEPARATE from col 1 every 2nd row starting with row 3 and put in column 4

would put the numbers 3, 5, 7 and 9 into the first four rows of column 4; leaving the remaining 6 rows intact. Caution, do not misspell the command by using SEPERATE. The instruction performs virtually the reverse operation of INSERT, above).

SHORTEN column (C) for col (C) equal to (K), put shortened cols in (C) and (C)

The instruction operates somewhat differently depending upon whether the numbers in the second named column are (1) in increasing order, (2) in decreasing order, or (3) in neither increasing or decreasing order.

(1) If the numbers in the second named column are in increasing order, the instruction, starting with row 1, looks for the first row containing the value (K). If there is no value in the column equal to (K), it finds the row containing the smallest number larger than (K). NRMAX is reset to agree with the row number selected and the first two columns are put in the columns designated by the last two arguments. The first two columns remain unchanged. For the numbers

Column 11: 1.0, 2.0, 3.0, 4.0 and 5.0
Column 12: 2.3, 4.2, 5.2, 7.6 and 8.3

the instruction

SHORTEN col 11 for col 12 equal to 5.3 and put shortened cols in cols 21 and 22

would set NRMAX=4 and give the following results

Column 21: 1.0, 2.0, 3.0 and 4.0
Column 22: 2.3, 4.2, 5.2 and 7.6

If the argument (K) in the above instruction was 5.2 instead of 5.3, NRMAX would equal 3 and the results would be same as above except 4.0 and 7.6 would not be put into row 4 of columns 21 and 22.

(2) If the numbers in the second named column are in decreasing order, the operation is similar to that described in (1). However, if the value (K) is not found in the second named column, the instruction looks for the row containing the largest number less than (K). Hence, if the numbers in columns 11 and 12, above, were in reverse order, the instruction above would set NRMAX=3 and give the results

Column 21: 5,0, 4.0 and 3.0
Column 22: 8.3, 7.6 and 5.2

(3) If the numbers in the second named column are in no particular order, the instruction looks for the first pair of consecutive rows containing values which bracket (K), i.e. the value in one row is less than or equal to (K) and the value in the other row is greater than or equal to (K). The value of NRMAX is reset to the higher of the two consecutive rows selected. For the numbers

Column 11: 1.0, 2.0, 3.0, 4.0 and 5.0
Column 12: 7.6, 2.3, 5.2, 8.3 and 4.2

the instruction

SHORTEN col 11 for col 12 equal to 5.2 and put shortened cols in cols 21 and 22

would set NRMAX=2 and give the results

Column 21: 1.0 and 2.0
Column 22: 7.6 and 2.3

Note, in this case, that the columns are not shortened at 5.2 and NRMAX is set equal to 2, because the values 7.6 and 2.3 bracket 5.2.

In any of the above three cases, if the value (K) is not found in the second named column, the following informative diagnostic is given

* VALUE REQUESTED IN SHORTEN, ACOALESCE OR AAVERAGE NOT FOUND.

NRMAX remains unchanged and the first two columns are put in the columns designated by the last two arguments.

3.4 Sorting Data.

HIERARCHY, ORDER, SORT

The instruction HIERARCHY does not sort directly, but is related to both sorting and ranking. The ORDER instruction operates on several columns simultaneously. On the other hand, the SORT instruction only sorts one column, but several columns can be carried along during the sort.

HIERARCHY of column (C), put locations of smallest thru largest in column (C)

Finds which row the smallest number is in and puts this row number in the first row of the 2nd named column; finds which row the second smallest number is in and puts this value into the second row of the 2nd named column; and so on. HIERARCHY is related to RANKS, described in section 4.1. If the numbers 7.38, 4.29, 2.15, 8.54 and 6.47 are in column 23, the instruction

HIERARCHY of column 23 put in column 28

would put the numbers 3, 2, 5, 1 and 4 into column 28.

ORDER independently columns (C), (C) ... (C) smallest to largest

Reorders each column separately, putting the smallest number in the first row, the second smallest in the second row, and so on until the largest number is put in row NRMAX. For the numbers

Column 12: 4.2, 7.6, 1.3, 9.8 and 5.7
Column 43: 11.3, 15.4, 19.6, 12.8 and 17.5

the instruction

ORDER columns 12 and 43

would change the numbers in columns 12 and 43 to

Column 12: 1.3, 4.2, 5.7, 7.6 and 9.8
Column 43: 11.3, 12.8, 15.4, 17.5 and 19.6

SORT column (C) min to max, carry along columns (C), (C) ... (C)

Sorts (orders) the numbers in the first named column in increasing order and carries along the numbers in the corresponding row of the other designated columns. The results are put back into the same columns. For the numbers (see HIERARCHY above)

Column 23: 7.38, 4.29, 2.15, 8.54 and 6.47
Column 11: 1.00, 2.00, 3.00, 4.00 and 5.00

the instructions

HIERARCHY of column 23 put in column 28
SORT column 28 and carry along column 11

would yield the results:

Column 28: 1.00, 2.00, 3.00, 4.00 and 5.00
Column 11: 4.00, 2.00, 1.00, 5.00 and 3.00

The numbers in column 11 are actually the ranks of the numbers in column 23. (See section 4.1 for a description of RANKS.)

Care should be used in sorting alphabetical information which has been entered into the worksheet using a FORMAT, since different computers handle blanks (and other special characters) in different ways. A (reasonably) safe, but inefficient, method is to use an A1 format specification.

SORT column (C)

This option is synonymous with the instruction ORDER, with one argument, described above.

3.5 Search Operations.

MATCH, SEARCH, SELECT

The three instructions described here perform three different kinds of searching. The instruction MATCH is particularly useful in treating subsets of data. The instruction ONEWAY, described in section 4.3, although primarily an instruction for statistical analysis, can be used to perform certain types of matching operations.

MATCH column (C) with (E), extract from (E) and put in column (C)

Each row of the first named column is compared with the second argument (E). If two values are exactly equal, the third argument (E) is put in the corresponding row of the column designated by the fourth argument. If the numbers are unequal the value in the column designated by the fourth argument remains unchanged. This is a powerful manipulative instruction, which is very often useful for analyzing subsets of data.

Suppose the salaries 9,881.00, 5,212.00, 6,548.00, 5,212.00 and 8,098.00 are in column 21 and the corresponding (GS) grades 9, 3, 5, 3 and 7 are in column 22. Suppose we want the total salary for all persons in grade 3 (and also suppose that column 31 contains all zeros.) Then the instructions

```
MATCH column 22 with 3.0, extract from col 21, put in col 31
SUM column 31 and put into column 31
```

would put the numbers 0.00, 5,212.00, 0.00, 5,212.00 and 0.00 in column 31 after the execution of MATCH and would put the desired result 10,414.00 in the first five rows of column 31 after the execution of SUM. If similar information was required for all grade levels the MATCH instruction could be used in the repeat mode or a ONEWAY instruction (see section 4.2) could be used.

Note, that if a match does not occur in any particular row, the corresponding row of the column designated by the fourth argument remains unchanged. Zero is not put into the row. If zero is the desired result, the user should either use an unused column or use an ERASE instruction. Note also, that two numbers must agree exactly (no tolerance is given) for a match to occur. Usually, but not necessarily, the second argument is either an integer or a column of integers. If integers are not involved, caution should be exercised since numbers may differ slightly due to round-off or conversion.

SEARCH col (C) equal col (C), move corresp nos in (C) into (C), (C) to (C) etc.

Searches down the first named column until it finds the first number equal to the number in row 1 of the second named column. If it finds the number in the kth. row, it transfers the number in row k of the 3rd named column to the first row of the 4th named column, row k of the 5th named column to the first row of the 6th named column, and so on. If the number is not found, a zero is put into the first row of the 4th, 6th, etc. named columns. This process is repeated using the value in row 2 of the second named column, then repeated again using the value in row 3 of the second named column, and so on. For the numbers

```
Column 1:      9,    3,    5,    3 and    7
Column 2:      3,    3,    3,    3 and    3
Column 3:   9881, 5212, 6548, 7864 and 8098
```

the instruction

```
SEARCH col 1 equal to col 2, move corresponding nos in col 3 into col 4
```

would put 5212 into each row of column 4, whereas the instruction

```
SEARCH col 2 equal to col 1, move corresponding nos in col 3 into col 4
```

would put the numbers 0, 9881, 0, 9881 and 0 into column 4. The instruction should not be confused with MATCH, described above. The instruction always has an even number of

arguments. The numbers in the columns designated by the odd numbered arguments and the second column remain unchanged.

SELECT in col (C) nos approx col (C) within absolute tol (K) and put in col (C)

The instruction looks at the absolute value of the difference between numbers in the 1st named column and the number in row 1 of the 2nd named column. It selects the row, call it row m, for which this difference is less than or equal to the absolute tolerance (K). If there is more than one difference which satisfies the tolerance, row m will be the one with the smallest difference. Only one row is selected. Then the value in row m of the 1st named column is put in row 1 of the column designated by the fourth argument. If the tolerance is never satisfied, zero is put into row 1 of the designated column. This process is repeated using the value in row 2 of the second named column, then for the value in row 3, and so on. The numbers in the first two designated columns remain unchanged. For the numbers

Column 31: 5, 4, 2.4, 2.3 and 1
Column 32: 2, 7, 5, 1 and 8

the instruction

SELECT in col 31 nos approx col 32 within absolute tolerance 0.5 and put in col 37

would put the numbers 2.3, 0, 5, 1 and 0 into column 37.

SELECT in (C) nos approx col (C) within absolute tol (K) put in cols (C) to (C)

Whereas the above form of SELECT only searches for the smallest difference which satisfies the specified tolerance, this form of the instruction searches for as many values as are determined by the last two arguments. The values in row (R) are stored according to the increasing size of the absolute difference between the number in row (R) of the 2nd named column and the numbers in the 1st named column. For the numbers

Column 31: 5.3, 13.8, 12.6, 12.2, 28.7, 13.4, 4.6, 3.9 and 61.7
Column 32: 12.0, 13.0, 14.0, 3.0, 4.0, 5.0, 26.0, 27.0 and 28.0

the instruction

SELECT in col 31 nos approx col 32 within 1.5 and put in cols 37 to 39

would yield

Column 37: 12.2, 13.4, 13.8, 3.9, 3.9, 5.3, 0.0, 0.0 and 28.7
Column 38: 12.6, 12.6, 13.4, 0.0, 4.6, 4.6, 0.0, 0.0 and 0.0
Column 39: 13.4, 13.8, 12.6, 0.0, 5.3, 3.9, 0.0, 0.0 and 0.0

Note that only 3 columns were designated for storage. Hence, the fourth value in column 31, 12.2, was not selected as approximating 13.0 in row 2 of column 32 within the tolerance 1.5.

SELECT in (C) nos approx (C) within abs tol (K) put in (C) to (C) count in (C)

This form of the SELECT instruction operates exactly like the one immediately above, but, in addition, the number of values selected from the first named column (count or frequency), for each row of the second named column, is put into each row of the column

designated by the last (6th) argument. For the numbers in the worksheet and the instruction in the form above, the instruction

SELECT in col 31 nos approx col 32 within 1.5, put in cols 37 to 39 and freq in col 40

would put the numbers 3, 4, 3, 1, 3, 3, 0, 0 and 1 into column 40.

4. STATISTICAL ANALYSIS.

The instructions in this section are described in sufficient detail to enable anyone familar with the statistical terms to use the instructions effectively. All the instructions in sections 4.2 through 4.6 can automatically produce a comprehensive set of statistics. It is beyond the scope of this manual to describe the automatic output of these instructions in complete detail. Plans are being made to publish separate papers, which will describe these instructions in more detail and will include a discussion of the uses and misuses of the various statistics. To supplement the material here and to aid the non-statistician, a set of references is provided in section 4.8. (An alternative set of references is given in Hogben (1969).)

Sections 4.2 through 4.6 contain optional forms of the basic instructions, which provide varying amounts of automatic storage. These forms also have additional forms with the letter S before the command to suppress the automatic printing. (See section B1.10.) In each case these options are merely listed and no description is given. The reader, in doubt, should refer to the previously described instruction without the letter S at the beginning of the command. If an attempt is made to put an S before the command in an instruction which does not provide storage of results, the instruction will be ignored and the following informative diagnostic will be given

```
* COMMAND IGNORED - S BEFORE COMMAND NAME MEANINGLESS IF NO STORAGE REQUESTED.
```

The instructions in sections 4.2, 4.3, 4.4 and 4.5 can be used to perform a weighted (unequal weights) analysis by specifying a column of weights. The use of negative weights is not allowed and causes the following fatal error

```
*** NEGATIVE WEIGHTS MAY NOT BE USED
```

If all the weights are equal to zero, the following informative diagnostic is given

```
* ALL WEIGHTS ARE ZERO. COMMAND IS NOT EXECUTED
```

A FREQUENCY instruction is the only one in this section which changes the value of NRMAX. Several of the instructions, which store results in the worksheet, may put results in rows below NRMAX, but the value of NRAMX is not changed.

OMNITAB may seem to have relatively limited capability for statistical analysis. There are only 12 basic instructions in this section. However, the instructions are very powerful and many have several options. Also, the analysis of real data often requires that the analyst look at the data in several different ways. The instructions here and in the other sections provide considerable power and flexibility to do just this.

4.1 Elementary Analysis.

FREQUENCY, HISTOGRAM, NHISTOGRAM, RANKS

Although the concept of a frequency distribution is very elementary, the computational problems are not so simple. As a result, there are eight different forms of the FREQUENCY instruction. The first four forms of the instruction do not store the lower and upper boundaries for each class, whereas the last four do store the boundaries. In all other respects the last four options are the same as the first four options.

In order to construct a frequency distribution for a set of numbers, it is necessary to determine (1) the number of classes which will be used to group the data, (2) the length of

each class, and (3) the lower boundary of the first class. Freund and Williams (1958) give a clear discussion of how this may be done. It is always assumed that all the classes have the same length.

In the first (and fifth) form the number of classes (k), the class length (K), and the lower boundary of the first class, (K), are all determined by the instruction. In the second (and sixth) form, the number of classes, (k), is specified by the user. In the third (and seventh) form, both the number of classes, (k), and the class length, (K), are specified by the user. In the fourth (and eighth) form, all three numbers are specified by the user. The argument (K) which specifies the class length and/or the lower boundary of the first class is a constant. All other arguments are integers.

Warning: In each form NRMAX is reset to agree with the number of classes, (k). Also, the number and type of arguments in the instruction determine which form is to be used, so the user should be careful. If the class length or lower class boundary of the first class is specified by the user, he should make sure that all the data are included in the frequency distribution.

When the number of classes is determined by the program, it is computed to be the integral part of $1.0 + 3.3\log_{10}(NRMAX)$, but never less than 5. The class length, unless otherwise specified, is equal to the absolute value of R/(k-1), where R is the range of the data (difference between the largest and the smallest) and k is the number of classes. The lower boundary of the first class, unless otherwise specified, is equal to the smallest datum minus one half the class length.

The instruction STATISTICAL analysis (in the next section) automatically prints a frequency distribution using 10 classes. The computational method is different from that described here. The different forms of FREQUENCY provide more flexibility.

No further descriptions of the forms of FREQUENCY are given below, but results of using the instruction on a specific set of data are presented. The data in column 1 are the actual 39 measurements of the velocity of light given by Mandel (1964) (coded by subtracting 299,799.0 from each measurement), namely

```
0.4,  0.6,  1.0,  1.0,  1.0,  0.5,  0.6,  0.7,  1.0,  0.6,  0.2,  1.9,  0.2,
0.4,  0.0, -0.4, -0.3,  0.0, -0.4, -0.3,  0.1, -0.1,  0.2, -0.5,  0.3, -0.1,
0.2, -0.2,  0.8,  0.5,  0.6,  0.8,  0.7,  0.7,  0.2,  0.5,  0.7,  0.8,  1.1
```

FREQUENCY distribution of column (C) put in column (C)

The instruction

FREQUENCY of column 1 put in column 13

would reset NRMAX to 6 and put the numbers 5, 11, 10, 12, 0 and 1 into column 13.

FREQUENCY distribution of column (C), using (k) classes, put in column (C)

The instruction

FREQUENCY of column 1 using 10 classes put in column 23

would reset NRMAX to 10 and put the numbers 3, 3, 5, 8, 7, 7, 5, 0, 0 and 1 into column 23.

FREQUENCY dist'n of col (C), using (k) classes of length (K), put in column (C)

The instruction

FREQUENCY of column 1 using 13 classes of length 0.2 put in column 33

would reset NRMAX to 13 and put the numbers 3, 2, 5, 6, 3, 7, 7, 0, 5, 0, 0, 0 and 1 into column 33.

FREQUENCY of (C), use (k) classes of length (K) starting at (K) put in col (C)

The instruction

FREQUENCY of column 1 using 5 classes of length 0.5 starting at -0.5 put in column 43

would reset NRMAX to 5 and put the numbers 8, 11, 14, 5 and 1 into column 43.

FREQUENCY of (C), put lower boundaries in (C) upper in (C) frequencies in (C)

The instruction

FREQUENCY of column 1, put boundaries and frequencies in cols 11, 12 and 13

would reset NRMAX to 6 and put the following numbers in the worksheet:

Column 11:	-.74, -.26, 0.22, 0.70, 1.18 and 1.66
Column 12:	-.26, 0.22, 0.70, 1.18, 1.66 and 2.14
Column 13:	5.00, 11.00, 10.00, 12.00, 0.00 and 1.00

The number of classes equals the integral part of $1+3.3 \times \log_{10}(39) = 1+3.3(1.59016) = 6.2475$ or 6. The class length is (1.9+.5)/5 = 2.4/5 = 0.48. The lower class boundary of the first class equals -0.5-0.48/2 = -0.74.

FREQUENCY of (C) use (k) classes, put bounds and freq's in cols (C), (C) and (C)

The instruction

FREQUENCY of column 1 using 10 cells put in columns 21, 22 and 23

would reset NRMAX to 10 and put the following numbers, correct to 4 decimals, in the worksheet:

Column 21:	-.6333, -.3667, -.1000, .1667, .4333, .7000, .9667, 1.2333, 1.5000 and 1.7667
Column 22:	-.3667, -.1000, .1667, .4333, .7000, .9667, 1.2333, 1.5000, 1.7667 and 2.0333
Column 23:	3.0, 3.0, 5.0, 8.0, 7.0, 7.0, 5.0, 0.0, 0.0 and 1.0

The class length is (1.9-(-.5))/(10-1) = 2.4/9 = 0.26666667. The lower boundary of the first class is -0.5 - 0.2666667/2 = -0.5 - 0.13333333 = -0.63333333.

FREQUENCY of (C) using (k) classes of length (K) put in cols (C), (C) and (C)

The instruction

FREQUENCY of col 1 using 13 classes of length 0.2 put in cols 31, 32 and 33

would reset NRMAX to 13 and put the following numbers in the worksheet:

Column 31:	-0.6,	-0.4,	-0.2,	0.0,	0.2,	0.4,	0.6,	0.8,	1.0,	1.2,	1.4,	1.6 and 1.8
Column 32:	-0.4,	-0.2,	0.0,	0.2,	0.4,	0.6,	0.8,	1.0,	1.2,	1.4,	1.6,	1.8 and 2.0
Column 33:	3.0,	2.0,	5.0,	6.0,	3.0,	7.0,	7.0,	0.0,	5.0,	0.0,	0.0,	0.0 and 1.0

The lower boundary of the first class equals -0.5 - 0.2/2 = -0.6.

FREQUENCY of (C), classes (k), length (K), start at (K), put in (C), (C) and (C)

The instruction

FREQUENCY of col 1 use 5 classes of length 0.5 start at -0.5 put in cols 41,42,43

would reset NRMAX to 5 and put the following numbers in the worksheet:

Column 41:	-0.5,	0.0,	0.5,	1.0 and 1.5
Column 42:	0.0,	0.5,	1.0,	1.5 and 2.0
Column 43:	8.0,	11.0,	14.0,	5.0 and 1.0

HISTOGRAM using mid-points in column (C) and frequencies in column (C)

Automatically prints a histogram using one line for each class and representing each datum by a plus sign. An example is given on page 114, showing the histogram and the LIST OF INSTRUCTIONS, DATA AND DIAGNOSTICS. The data are the 39 measurements of the velocity of light used in the examples of FREQUENCY, above, and also in the example of STATISTICAL analysis in the next section. The instruction automatically prints the title HISTOGRAM FOR FREQUENCIES ... and the column headings MID-POINTS and FREQUENCY. The instruction is often, but not necessarily, used in conjunction with the last four forms of FREQUENCY, The mid-points, the points half-way between the lower and upper class boundaries, were obtained in this example by using the instruction

ADD col 41 to col 42, mult by 0.5, add 0.0 and put in col 44

A maximum of 95 plus signs are printed on any one line. If the frequency in any row exceeds 95, printing continues on the next line(s).

NHISTOGRAM using mid-points in column (C) and frequencies in column (C)

Same as the above instruction; except (1) printing does not start on a new page, (2) the headings MID-POINTS and FREQUENCY are not printed, and (3) the title HISTOGRAM FOR ... is not printed.

HISTOGRAM FOR MID-POINTS IN COLUMN 44, FREQUENCIES IN COLUMN 43

MID-POINTS	FREQUENCY	
-.25000000	8	++++++++
.25000000	11	++++++++++
.75000000	14	++++++++++++
1.2500000	5	+++++
1.7500000	1	+

LIST OF COMMANDS, DATA AND DIAGNOSTICS

```
SET VELOCITY OF LIGHT IN COLUMN 1
.4,.6,1,1,1,.5,.6,.7,1,.6,.2,1.9,.2
.4,0,-.4,-.3,0,-.4,-.3,.1,-.1,.2,-.5,.3,-.1
.2,-.2,.8,.5,.6,.8,.7,.7,.2,.5,.7,.8,1.1
FREQUENCY OF COL 1 USING 5 CELLS OF LENGTH .5 START AT -.5 PUT IN COLS 41,42,43
ADD COL 41 TO COL 42, MULT BY 0.5, ADD 0.0 AND PUT IN COL 44
HISTOGRAM FOR MID-POINTS IN COL 44 AND FREQUENCIES IN COL 43
STOP
```

NATIONAL BUREAU OF STANDARDS, WASHINGTON, D. C. 20234, OMNITAB II VERSION 5.00 MAY 15, 1970

RANKS of column (C) put in column (C)

Computes the ranks of a column of numbers. Whenever two or more numbers are tied (equal), each is assigned the rank equal to the average of the ranks if they were not equal. E.g., the numbers 3, 3, 7, 7 and 7 would have ranks 1.5, 1.5, 4, 4 and 4. The value of

$$T = (1/12) \underset{\text{all ties}}{\text{Sum}} (t-1)t(t+1),$$

where t is the number tied, is put into the first row below row NRMAX (unless NRMAX equals the number of rows in the worksheet). Thus, in the above example T = (1/12)x(1x2x3 + 2x3x4) = (1/12)x30 = 2.5 would appear in row 6. The value of NRMAX remains unchanged. The value of T may be used to make adjustments for ties in statistics such as the rank correlation coefficient. It is used by a CORRELATION instruction (see section 4.6). See, also, Kendall (1948). If the numbers 4.0, 9.0, 7.0 and 1.0 are in column 53, the instruction

```
RANKS of column 53 put in column 52
```

would put the numbers 2.0, 4.0, 3.0, 1.0 and 0.0 into the first five rows of column 52. NRMAX remains equal to 4.

4.2 Analysis Of One Column Of Data.

STATISTICAL, SSTATISTICAL

A STATISTICAL analysis instruction automatically prints (unless suppressed by using SSTATISTICAL), on one page, 43 statistics for a single column of data and also a frequency distribution. This page is followed by the printing of the data, the ranks of the data, the deviations (residuals) about the mean (average), the ordered observations and the differences between adjacent ordered observations. Pages 117 and 118 give the results of using a STATISTICAL analysis instruction on 39 actual measurements of the velocity of light given in Mandel (1964). The actual set of instructions used in this example is

```
OMNITAB 6/4/69 VELOCITY OF LIGHT MEASUREMENTS
    SET VELOCITY IN COL 1 $ -299799.0
    .4,.6,1,1,1,.5,.6,.7,1,.6,.2,1.9,.2
    .4,0,-.4,-.3,0,-.4,-.3,.1,-.1,.2,-.5,.3,-.1
    .2,-.2,.8,.5,.6,.8,.7,.7,.2,.5,.7,.8,1.1
    STATISTICAL ANALYSIS OF COLUMN 1
```

There are 6 different forms of STATIS, plus 4 more (SSTATIS) which suppress the automatic printing. STATIS is a frequently used shortened form of STATISTICAL analysis. The six forms differ depending upon whether (1) weights are or are not used, (2) there is or is not automatic storage of results, and (3) storage of results is in consecutive columns or in specified columns. Each form is differentiated from the others by the number of arguments (column numbers) in the instruction. The following table summarizes the number of arguments used in each form of the instruction.

Number of Arguments in STATIS Instruction

	No Storage	Consecutive Storage	Specified Storage
No weights (weights=1)	1	2	5
Weights specified	-3*	3	6

* Last column number is preceded by a minus sign

When results are stored, the statistics appear in the first column, the ranks of the measurements in the second column, the ordered measurements in the third column and the residuals (deviations from the mean) in the fourth column. For example,

In the instructions:-	STATIS 1, 11	STATIS 3, 17, 21, 47, 34
Measurements are in column	1	3
Statistics (page 1)	11	17
Ranks (page 2)	12	21
Ordered measurements (page 2)	13	47
Residuals (page 2)	14	34

Page 119 shows the order in which the statistics in the automatic printing are stored in the worksheet when the 2nd, 3rd, 5th or 6th forms of STATIS are used. The automatic printing gives some references to Natrella (1963). Page 119 gives additional references. In each case, a single reference is given, yet often many of the references in section 4.8 discuss the statistic.

Page 120 gives the formulas used to compute each of the statistics. Note, some of the formulas are defined in terms of some of the other formulas. Special algorithms were developed to compute the confidence intervals for the mean and standard deviation (Hogben (1968)).

Remark 1.

When weights equal to zero are used, the analysis is exactly the same as if the corresponding measurements were not used. Note, if in one case weights are not specified (all weights equal 1.0) and in a second case weights equal to 3.0 are used, the value of the sample variance in the second case will be three times as large as in the first case. Let the variance of a measurement be

$$\sigma_i^2 = k_i\sigma^2 \text{ and let } w_i = 1/k_i \text{ and } \sigma_i^2 = 2.3.$$

In the first case, $k_i = 1$, and the sample variance estimates 2.3. In the second case, $k_i = 1/3$ and the variance estimates 6.9 (2.3 = 6.9/3).

Remark 2

Sometimes it is desirable to compare the statistics from several sets of measurements. One way to do this is to use SSTATIS and put the results into adjacent columns. If the identification of each row is desired, titles can be read and printed using (for example) a 3A4 format specification. The following is an example of a set of instructions for 4 sets of 100 measurements.

```
FORMAT A (3A4)
READ A 43 cards into columns 11, 12 and 13
     (43 cards with titles in card columns 1 to 12)
GENERATE 1.(1.)43. into column 10
READ data into columns 21, 22, 23 and 24
     (100 cards, 4 numbers on each card)
SSTATIS 21, 31
SSTATIS 22, 32
SSTATIS 23, 33
SSTATIS 24, 34
FORMAT B (1X,F3.0,3X,3A4,1P4E15.6)
RESET 43
PRINT B columns 10, 11, 12, 13 and 31, 32, 33, 34
```

OMNITAB 6/4/69 VELOCITY OF LIGHT MEASUREMENTS

STATIS. AL ANALYSIS OF COL 1 N = 39

FREQUENCY DISTRIBUTION (1-6) 5 3 8 3 7 7 5 0 0 1

MEASURES OF LOCATION (2-2)

UNWEIGHTED MEAN	=	4.1025639-01
WEIGHTED MEAN	=	4.1025639-01
MEDIAN	=	5.0000000-01
MID-RANGE	=	7.0000000-01
25 PCT UNWTD TRIMMED MEAN	=	4.2380951-01
25 PCT WTD TRIMMED MEAN	=	4.2380951-01

MEASURES OF DISPERSION (2-6)

STANDARD DEVIATION	=	5.0668936-01
S.D. OF MEAN	=	8.1135232-02
RANGE	=	2.4000000+00
MEAN DEVIATION	=	4.0486518-01
VARIANCE	=	2.5673411-01
COEFFICIENT OF VARIATION	=	1.2350554+02

A TWO-SIDED 95 PCT CONFIDENCE INTERVAL FOR MEAN IS 2.4601-01 TO 5.7451-01 (2-2)
A TWO-SIDED 95 PCT CONFIDENCE INTERVAL FOR S.D. IS 4.1071-01 TO 6.4762-01 (2-7)

LINEAR TREND STATISTICS (5-1)

SLOPE	=	-4.0080933-03
S.D. OF SLOPE	=	7.2760492-03
SLOPE/S.D. OF SLOPE = T	=	-5.5086122-01
PROB EXCEEDING ABS VALUE OF OBS T	=	.585

OTHER STATISTICS

MINIMUM	=	-5.0000000-01
MAXIMUM	=	1.9000000+00
BETA ONE	=	9.6501456-02
BETA TWO	=	3.3379331+00
WTD SUM OF VALUES	=	1.5999999+01
WTD SUM OF SQUARES	=	1.6319999+01
WTD SUM OF DEVS SQUARED	=	9.7558964+00
STUDENT'S T	=	5.0564517+00
WTD SUM ABSOLUTE VALUES	=	2.0599998+01
WTD AVE ABSOLUTE VALUES	=	5.2820507-01

TESTS FOR NON-RANDOMNESS

NO OF RUNS UP AND DOWN	=	23
EXPECTED NO OF RUNS	=	25.7
S.D. OF NO OF RUNS	=	2.57
MEAN SQ SUCCESSIVE DIFF	=	2.8289471-01
MEAN SQ SUCC DIFF/VAR	=	1.102

DEVIATIONS FROM WTD MEAN

NO OF + SIGNS	=	20
NO OF - SIGNS	=	19
NO OF RUNS	=	8
EXPECTED NO OF RUNS	=	20.5
S.D. OF RUNS	=	3.08
DIFF./S.D. OF RUNS	=	-4.056

NOTE - ITEMS IN PARENTHESES REFER TO PAGE NUMBER IN NBS HANDBOOK 91

OBSERVATIONS

I	X(I)	RANK	X(I)-MEAN
1	4.0000000-01	18.5	-1.0256387-02
2	5.9999999-01	24.5	1.8974361-01
3	1.0000000+00	35.5	5.8974361-01
4	1.0000000+00	35.5	5.8974361-01
5	1.0000000+00	35.5	5.8974361-01
6	5.0000000-01	21.0	8.9743614-02
7	5.9999999-01	24.5	1.8974361-01
8	7.0000000-01	28.5	2.8974361-01
9	1.0000000+00	35.5	5.8974361-01
10	5.9999999-01	24.5	1.8974361-01
11	2.0000000-01	14.0	-2.1025639-01
12	1.9000000+00	39.0	1.4897436+00
13	2.0000000-01	14.0	-2.1025639-01
14	4.0000000-01	18.5	-1.0256387-02
15	0.0000000	9.5	-4.1025639-01
16	-4.0000000-01	2.5	-8.1025638-01
17	-3.0000000-01	4.5	-7.1025638-01
18	0.0000000	9.5	-4.1025639-01
19	-4.0000000-01	2.5	-8.1025638-01
20	-3.0000000-01	4.5	-7.1025638-01
21	1.0000000-01	11.0	-3.1025639-01
22	-1.0000000-01	7.5	-5.1025638-01
23	2.0000000-01	14.0	-2.1025639-01
24	-5.0000000-01	1.0	-9.1025639-01
25	3.0000000-01	17.0	-1.1025639-01
26	-1.0000000-01	7.5	-5.1025638-01
27	2.0000000-01	14.0	-2.1025639-01
28	-2.0000000-01	6.0	-6.1025638-01
29	8.0000000-01	32.0	3.8974361-01
30	5.0000000-01	21.0	8.9743614-02
31	5.9999999-01	24.5	1.8974361-01
32	8.0000000-01	32.0	3.8974361-01
33	7.0000000-01	28.5	2.8974361-01
34	7.0000000-01	28.5	2.8974361-01
35	2.0000000-01	14.0	-2.1025639-01
36	5.0000000-01	21.0	8.9743614-02
37	7.0000000-01	28.5	2.8974361-01
38	8.0000000-01	32.0	3.8974361-01
39	1.1000000+00	38.0	6.8974361-01

ORDERED OBSERVATIONS

NO.	X(J)	X(J+1)-X(J)
24	-5.0000000-01	1.0000000-01
16	-4.0000000-01	0.0000000
19	-4.0000000-01	1.0000000-01
17	-3.0000000-01	0.0000000
20	-3.0000000-01	9.9999998-02
28	-2.0000000-01	1.0000000-01
22	-1.0000000-01	0.0000000
26	-1.0000000-01	1.0000000-01
15	0.0000000	0.0000000
18	0.0000000	1.0000000-01
21	1.0000000-01	1.0000000-01
11	2.0000000-01	0.0000000
13	2.0000000-01	0.0000000
23	2.0000000-01	0.0000000
27	2.0000000-01	0.0000000
35	2.0000000-01	9.9999998-02
25	3.0000000-01	1.0000000-01
1	4.0000000-01	0.0000000
14	4.0000000-01	1.0000000-01
6	5.0000000-01	0.0000000
30	5.0000000-01	0.0000000
36	5.0000000-01	9.9999994-02
2	5.9999999-01	0.0000000
7	5.9999999-01	0.0000000
10	5.9999999-01	0.0000000
31	5.9999999-01	1.0000000-01
8	7.0000000-01	0.0000000
33	7.0000000-01	0.0000000
34	7.0000000-01	0.0000000
37	7.0000000-01	1.0000000-01
29	8.0000000-01	0.0000000
32	8.0000000-01	0.0000000
38	8.0000000-01	2.0000000-01
3	1.0000000+00	0.0000000
4	1.0000000+00	0.0000000
5	1.0000000+00	0.0000000
9	1.0000000+00	9.9999994-02
39	1.1000000+00	8.0000000-01
12	1.9000000+00	

Row	Name Of Statistic	Reference
1	N = LENGTH OF COLUMN, NRMAX	Section B1.6
2	NUMBER OF NON-ZERO WEIGHTS	Section 4.2
MEASURES OF LOCATION		
3	UNWEIGHTED MEAN	Dixon and Massey (1957), 14
4	WEIGHTED MEAN	Brownlee (1965), 95-97
5	MEDIAN	Dixon and Massey (1957), 70
6	MID-RANGE	Dixon and Massey (1957), 71
7	25 PCT UNWTD TRIMMED MEAN	Crow and Siddiqui (1967)
8	25 PCT WTD TRIMMED MEAN	
MEASURES OF DISPERSION		
9	STANDARD DEVIATION	Snedecor and Cochran (1967), 44
10	S.D. OF MEAN	Brownlee (1965), 80
11	RANGE	Snedecor and Cochran (1967), 39
12	MEAN DEVIATION	Duncan (1965), 50
13	VARIANCE	Snedecor and Cochran (1967), 44
14	COEFFICIENT OF VARIATION	Snedecor and Cochran (1967), 62
CONFIDENCE INTERVALS		
15	LOWER CONFIDENCE LIMIT, MEAN	Natrella (1963), 2-2,2-3
16	UPPER CONFIDENCE LIMIT, MEAN	Natrella (1963), 2-2,2-3
17	LOWER CONFIDENCE LIMIT, S.D.	Natrella (1963), 2-7
18	UPPER CONFIDENCE LIMIT, S.D.	Natrella (1963), 2-7
LINEAR TREND STATISTICS		
19	SLOPE	Fisher (1950), 136
20	S.D. OF SLOPE	
21	SLOPE/S.D. OF SLOPE = T	
22	PROB EXCEEDING ABS VALUE OF OBS T	Brownlee (1965), 344
TESTS FOR NON-RANDOMNESS		
23	NO OF RUNS UP AND DOWN	Brownlee (1965), 223
24	EXPECTED NO OF RUNS	Bradley (1968), 279
25	S.D. OF NO OF RUNS	Bradley (1968), 279
26	MEAN SQ SUCCESSIVE DIFF	Brownlee (1965), 222
27	MEAN SQ SUCC DIFF/VAR	Brownlee (1965), 222
DEVIATIONS FROM WTD MEAN		
28	NO OF + SIGNS	
29	NO OF - SIGNS	
30	NO OF RUNS	Brownlee (1965), 224
31	EXPECTED NO OF RUNS	Brownlee (1965), 227
32	S.D. OF RUNS	Brownlee (1965), 230
33	DIFF./S.D. OF RUNS	Brownlee (1965), 230
OTHER STATISTICS		
34	MINIMUM	Natrella (1963), 19-1
35	MAXIMUM	Natrella (1963), 19-3
36	BETA ONE	Snedecor and Cochran (1967), 86
37	BETA TWO	Snedecor and Cochran (1967), 87
38	WTD SUM OF VALUES	
39	WTD SUM OF SQUARES	
40	WTD SUM OF DEVS SQUARED	
41	STUDENT'S T	Brownlee (1965), 296
42	WTD SUM ABSOLUTE VALUES	
43	WTD AVE ABSOLUTE VALUES	
.		
51-60	FREQUENCY DISTRIBUTION	Freund and Williams (1958), 17

Row	Formula
1.	n = NRMAX
2.	k (k=n, if weights are not specified)
3.	$\bar{x} = \sum_{i=1}^{n} x_i/n$
4.	$\bar{x}_w = \sum_{i=1}^{n} w_i x_i / \sum_{i=1}^{n} w_i$
5.	$x_{[(k+1)/2]}$, k odd, or $\frac{1}{2}(x_{[k/2]} + x_{[(k+1)/2]})$, k even
6.	$\frac{1}{2}(x_{[1]} + x_{[k]})$
7.	$\sum_{i=1+[n/4]}^{n-[n/4]} x_i/(n-2[n/4])$
8.	$\sum_{i=1+[k/4]}^{k-[k/4]} w_i x_i / \sum_{i=1+[k/4]}^{k-[k/4]} w_i$
9.	s
10.	$s_{\bar{x}} = s/\sum_{i=1}^{k} w_i$
11.	$R = x_{[k]} - x_{[1]}$
12.	$\sum_{i=1}^{n} \lvert x_i - \bar{x}_w \rvert /n$
13.	$s^2 = \sum_{i=1}^{n} w_i (x_i - \bar{x}_w)^2/(k-1)$
14.	$100s/\bar{x}_w$
15.	$\bar{x} - t_{.025} s_{\bar{x}}$, using (n-1) degrees of freedom
16.	$\bar{x} + t_{.025} s_{\bar{x}}$, using (n-1) degrees of freedom
17.	$s\sqrt{(k-1)/\chi^2_{.975}}$, with (k-1) degrees of freedom
18.	$s\sqrt{(k-1)/\chi^2_{.025}}$, with (k-1) degrees of freedom
19.	$B = 12\sum_{i=1}^{k} i(x_i - \bar{x}_w)/k(k^2-1)$
20.	$[12\sum(x_i - \bar{x}_w)^2/k(k^2-1) - B^2]^{\frac{1}{2}}/(k-2)^{\frac{1}{2}}$
21.	t_0 = (19)/(20), with (k-2) degrees of freedom
22.	$Pr\{t<-\lvert t_0 \rvert$ and $t>+\lvert t_0 \rvert\}$
23.	r
24.	(2k-1)/3
25.	$[(16k-29)/90]^{\frac{1}{2}}$
26.	$D^2 = \sum_{i=1}^{k-1} (x_{i+1} - x_i)^2/(k-1)$
27.	D^2/s^2
28.	u
29.	v
30.	1 + no. of changes in sign of $(x_i - \bar{x}_w)$
31.	1 + (2uv)/k
32.	$[2uv(2uv-u-v)/(u+v)^2(k-1)]^{\frac{1}{2}}$
33.	[(30)-(31)]/(32)
34.	$x_{[1]}$, smallest value with non-zero weight
35.	$x_{[k]}$, largest value with non-zero weight
36.	$[\sum_{i=1}^{k} (x_i - \bar{x}_w)^3/k]^2 / [(k-1)s^2/k]^3$
37.	$[\sum_{i=1}^{k} (x_i - \bar{x}_w)^4/k] / [(k-1)s^2/k]^2$
38.	$\sum_{i=1}^{n} w_i x_i$
39.	$\sum_{i=1}^{n} w_i x_i^2$
40.	$\sum_{i=1}^{n} w_i (x_i - \bar{x}_w)^2$
41.	$t = (\sqrt{\sum_{i=1}^{n} w_i})\bar{x}_w/s$, with (k-1) degrees of freedom
42.	$\sum_{i=1}^{n} w_i \lvert x_i \rvert$
43.	$\sum_{i=1}^{n} w_i \lvert x_i \rvert / \sum_{i=1}^{n} w_i$

```
STATISTICAL analysis of column (C)
```

The STATIS instruction with 1 argument prints the comprehensive set of statistics without using unequal weights and does not store any results in the worksheet.

```
STATISTICAL analysis of column (C), put statistics in (C) and next three cols
```

The STATIS instruction with two arguments performs an unweighted (equal weights) analysis and puts results in the column designated by the second argument and the three successive columns which follow.

```
STATISTICAL analysis of (C), weights (C), put in (C) and next three cols
```

The STATIS instruction with three arguments uses weights and stores results in four columns. Results are stored in the column designated by the last argument and in the three successive columns which follow.

```
STATISTICAL anal of (C) wts in (C) don't put in (-C)
```

This form of the STATIS instruction uses weights and does not provide any storage of results. The last argument must be preceded by a minus sign to indicate that storage of results is not requested. This is the only instruction in OMNITAB which uses a negative column number. This technique was necessary to differentiate this form from the one above.

```
STATISTICAL analysis of col (C), put statistics in cols (C), (C), (C) and (C)
```

The STATIS instruction with 5 arguments does not use (unequal) weights and stores statistics in the four columns designated by the last four arguments.

```
STATISTICAL anal of (C) wts in (C), put statis in cols (C), (C), (C) and (C)
```

The STATIS instruction with 6 arguments uses weights and puts statistics in the four columns designated by the last four arguments.

*****

```
SSTATISTICAL analysis of column (C), put statistics in (C) and next three cols
```

```
SSTATISTICAL analysis of (C), weights (C), put in (C) and next three cols
```

```
SSTATISTICAL analysis of col (C), put statistics in cols (C), (C), (C) and (C)

SSTATISTICAL anal of (C) wts in (C), put statis in cols (C), (C), (C) and (C)
```

4.3 Analysis Of Groups Of Data.

ONEWAY, SONEWAY

The instruction ONEWAY automatically prints a comprehensive set of results for analyzing a one-way classification. Although primarily an instruction for statistical analysis, ONEWAY has other uses in data manipulation. It can be used simply to determine the number of values in each of several groups.

Use of Instruction.

There are three forms of the instruction with 2, 3 and 6 arguments, which are all column numbers. The first argument is the number of the column containing the data or measurements. The second argument is the number of the column containing the corresponding group number (identification or tag). Suppose six measurements have been made using three different types of equipment; two using the first, one using the second and three using the third type of equipment. Then, the group numbers, or tags, would be 1, 1, 2, 3, 3 and 3. These numbers should be used if the equipment (groups) have been labeled A, B and C.

The group numbers should be integers.* Every integer from 1 to the largest should be represented; i.e., there should be at least one measurement in each group.* No diagnostic is given if these conditions are not met. If any tag is less than 1.0, it is set equal to 0.0 and the corresponding measurement is ignored. (The measurement is given the weight zero.) The group numbers do not have to be in increasing order. They can be in any order as long as it matches the order in which the measurements are entered into the worksheet.

* These restrictions will be lifted in the near future.

Restrictions.

(1) The value of NRMAX must not exceed 2700.
(2) The number of groups must not exceed 540.
(3) The number of groups must be at least 2.
(4) The number of groups must not exceed the number of measurements with positive tag.

Violations of these restrictions result in the following fatal error:

*** COLUMN NUMBER TOO BIG OR LESS THAN 1

(Restrictions (1) and (2) apply to NBS worksheet of size 12,500.)

Automatic Printing.

Each of the five sections of the automatic printing is described below under the headings which appear on the printed page as shown in the example on page 126.

ANALYSIS OF VARIANCE

The traditional analysis of variance for a one-way classification is printed, which shows the source of variation, degrees of freedom (D.F.), the sums of squares, the mean squares, the F-ratio for testing for differences between group means, and the significance

level (F PROB.) of the F-ratio. The usual assumptions of normality, independence and constant variance of measurement errors are made. See, for example, section 10.2 of Brownlee (1965) for a discussion of the statistical treatment of a one-way classification.

If the significance level (F PROB.) for

F = Between Groups Mean Square/Within Groups Mean Square

is less than 0.10 and the number of groups exceeds 2, additional results are printed, which do not appear in the traditional analysis of variance. The Between Groups (means) sum of squares is separated into two components; the slope with 1 degree of freedom and the balance representing deviations about the straight line regression of group averages on group number. This information may or may not be useful. Often, time has an important effect, which may be revealed in these results. See section 11.12 of Brownlee (1965) for a discussion of some of the statistical aspects of this procedure.

Following the above analysis of variance, results are printed for the Kruskal-Wallis non-parametric H-test for testing for differences between group means (averages). The value of H is printed along with its significance level (F PROB.). The H-test uses the ranks of the measurements and avoids any assumption about the distribution of measurement errors. Details of the test may be found in section 7.7 of Brownlee (1965) and in the original paper by Kruskal and Wallis referenced therein.

ESTIMATES

The following items are printed for each group: the (1) group number, (2) number of measurements, (3) average (mean), (4) within standard deviation, (5) standard deviation of the mean, (6) minimum or smallest measurement, (7) maximum or largest measurement, (8) S(R) equal to the sum of the ranks of the measurements, and (9) a 95% confidence interval for the mean. Six significant digits are used to print items (3), (4), (5), (6), (7) and (9) in floating-point (scientific) notation. The values printed in items (1), (2) and (8) are exact. The results are printed with the group numbers (tags) in consecutive, increasing order; regardless of the order in which the numbers were entered into the worksheet.

In the printing of the averages (means) and standard deviations, the letters H and L are put immediately after the high (largest) and low (smallest) values. If two values are tied for the largest (smallest), the letter H (L) is put immediately after both values. If the number of measurements in a group equals 1, ESTIMATE NOT AVAILABLE is printed under WITHIN S.D. and S.D. OF MEAN. Also, ********** TO ********** appears under 95PCT CONF INT FOR MEAN.

The number of measurements, mean, minimum and maximum are also printed for the total number of measurements. In addition, the within standard deviation, standard deviation of the mean and 95% confidence interval for the mean are printed for three different models: the fixed effects model (Model I), the random effects model (Model II), and the model which assumes that all the groups are the same so that in effect all measurements were taken from a single group.

The confidence limits are formed by taking the grand mean and first subtracting and then adding the product of the percentage point of Student's t-distribution and the standard deviation of the mean. Let k = number of groups and n = total number of measurements (with positive tag). Then, the standard deviation of the mean is the square root of the variance of the mean formed as follows:

Model	Variance	Variance of Mean	Degrees of Freedom
I	v = Within groups mean square	v/n	n-k
II	$v = \sum_{i=1}^{k}(x_i-\bar{x})^2/(k-1)$	v/k	k-1
Ungrouped	v = Total mean square	v/n	n-1

PAIRWISE MULTIPLE COMPARISION OF MEANS.

This section only appears if the significance level (value under F PROB.) of the Between Groups F-ratio is less than 0.10. The Newman-Keuls-Hartley procedure is not performed if the number of measurements, with positive tag, is less than 4 plus the number of groups.

The purpose here is to divide the averages into groups in such a way that all averages within a group are not significantly different at the .05 significance level, whereas averages in different groups are significantly different. Two different procedures are used; the Newman-Keuls-Hartley method and the Scheffé method. The two methods are similar, but not identical and frequently give slightly different results. The Newman-Keuls-Hartley method is described in section 10.6 of Snedecor (1956) and section 10.8 of Snedecor and Cochran (1967). The Scheffé method is discussed in section 10.3 of Brownlee (1965). Groups are separated by a string of 5 asterisks. If adjacent groups have no average in common, the two groups are separated by two strings of 5 asterisks.

Both methods require percentage points of the studentized range. Here, a good approximation developed by John Mandel is used. The Newman-Keuls-Hartley method is designed for use when the number of measurements in each group is the same. An approximation is used in comparing two means by setting

$$1/n = \tfrac{1}{2}(1/n_i + 1/n_j)$$

TEST FOR HOMOGENEITY OF VARIANCES.

The usual analysis of variance for a one-way classification assumes that the variance of each group is the same. This section provides information for assessing the validity of this assumption. Small values of the significance level P indicate lack of homogeneity of variance. Cochran's C is discussed on page 180 of Dixon and Massey (1957) and in more detail in Chapter 15 of Eisenhart et al. (1947). The Bartlett-Box F-test is a modification of Bartlett's test which uses the F-distribution rather than the chi-squared distribution and is less sensitive to non-normality. It is discussed on pages 179 and 180 of Dixon and Massey (1957). A table of critical values of maximum variance/minimum variance for equal sample sizes is given on pages 100 and 101 of Owen (1962).

If either P value is less than or equal to 0.10, the following is also printed

APPROX BETWEEN MEAN F-TEST IN PRESENCE OF HETEROGENEOUS VARIANCE

followed by the F-value and significance level P. This approximate F-test for testing for differences between means is described on pages 287-289 of Snedecor (1956). Note, this information does not appear in the example on page 126, because both P values (significance levels) exceed 0.10.

MODEL II - COMPONENTS OF VARIANCE.

This is the usual analysis of variance estimate for the between component in a random effects model (Model II). See, for example, sections 10.6 and 10.7 of Brownlee (1965).

ONEWAY analysis for data in column (C) with group number in column (C)

Provides the comprehensive automatic printing described above, but does not give any automatic storage of results. The following set of instructions, to analyze the data on page 315 of Brownlee (1965), give the automatic printing shown on page 126.

```
OMNITAB 2/22/71 EXAMPLE OF ONEWAY - BROWNLEE DATA PAGE 315
SET data in column 1
   83   81   76   78   79   72
   61   61   67   67   64
   78   71   75   72   74
SET group number in column 2
    1    1    1    1    1    1
    2    2    2    2    2
    3    3    3    3    3
ONEWAY with data in col 1 and group no. in col 2
```

ONEWAY for (C) group no in (C) put statistics in (C) and next three columns

This form of ONEWAY, with an additional argument, provides automatic storage in addition to the automatic printing. If the last column number is X, then results are stored in the four consecutive columns X, X+1, X+2 and X+3 as follows:

Column	Contents
X	group number
X+1	number of measurements in each group
X+2	group averages (means)
X+3	group standard deviations

These are the results in the first four columns in the ESTIMATES section of the automatic printing. The instructions, applied to the above example,

```
ONEWAY   1,2 and store in col 11 and next three cols
          PRINT columns 11, 12, 13 and 14
```

would yield

COLUMN 11	COLUMN 12	COLUMN 13	COLUMN 14
1.0000000	6.0000000	78.166666	3.8686776
2.0000000	5.0000000	64.000000	3.0000000
3.0000000	5.0000000	74.000000	2.7386128

ONEWAY for (C) with (C), put group in (C), number in (C), means (C), s.d. (C)

Same as above; except the last three columns for storage are specified by the user instead of implied by the instruction. The last four column numbers do not have to be either consecutive or equal.

Storage of results is done sequentially in the same order as in the form above. Hence, if any column number is used more than once, results stored first will be erased. The instruction

```
ONEWAY  1,2   and   11, 11, 12, 12
```

would put the numbers in each group in column 11 and the standard deviations in column 12. The group identification and means would not be stored.

ANALYSIS OF VARIANCE

SOURCE	D.F.	SUM OF SQUARES	MEAN SQUARES	F RATIO	F PROB.
BETWEEN GROUPS	2	5.651041+02	2.825521+02	26.082	.000
SLOPE	1	6.450891+01	6.450891+01	1.408	.257
DEVS. ABOUT LINE	1	5.005952+02	5.005952+02	46.209	.000
WITHIN GROUPS	13	1.408333+02	1.083333+01		
TOTAL	15	7.059375+02			

KRUSKAL-WALLIS RANK TEST FOR DIFFERENCE BETWEEN GROUP MEANS * H = 11.391, F PROB = .000 (APPROX.)

ESTIMATES

GROUP	NO.	MEAN	WITHIN S.D.	S.D. OF MEAN	MINIMUM	MAXIMUM	S(R)	95PCT CONF INT FOR MEAN
1	6	7.81667+01H	3.86868+00H	1.57938+00	7.20000+01	8.30000+01	76.0	7.41068+01 TO 8.22265+01
2	5	6.40000+01L	3.00000+00	1.34164+00	6.10000+01	6.70000+01	15.0	6.02751+01 TO 6.77249+01
3	5	7.40000+01	2.73861+00L	1.22474+00	7.10000+01	7.80000+01	45.0	7.05996+01 TO 7.74004+01
TOTAL	16	7.24375+01			6.10000+01	8.30000+01		
FIXED EFFECTS MODEL			3.29140+00	8.22851-01				7.06598+01 TO 7.42152+01
RANDOM EFFECTS MODEL			7.29576+00	4.21221+00				5.43136+01 TO 9.05614+01
UNGROUPED DATA			6.86021+00	1.71505+00				6.87820+01 TO 7.60930+01

PAIRWISE MULTIPLE COMPARISON OF MEANS. THE MEANS ARE PUT IN INCREASING ORDER IN GROUPS SEPARATED BY *****. A MEAN IS ADJUDGED NON-SIGNIFICANTLY DIFFERENT FROM ANY MEAN IN THE SAME GROUP AND SIGNIFICANTLY DIFFERENT AT THE .05 LEVEL FROM ANY MEAN IN ANOTHER GROUP. ***** ***** INDICATES ADJACENT GROUPS HAVE NO COMMON MEAN.

NEWMAN-KEULS TECHNIQUE, HARTLEY MODIFICATION. (APPROXIMATE IF GROUP NUMBERS ARE UNEQUAL.)

6.40000+01,
***** *****
7.40000+01, 7.81667+01,

SCHEFFE TECHNIQUE.

6.40000+01,
***** *****
7.40000+01, 7.81667+01,

TESTS FOR HOMOGENEITY OF VARIANCES.

COCHRAN'S C = MAX. VARIANCE/SUM(VARIANCES) = .4756, P = .439 (APPROX.)
BARTLETT-BOX F = .269, P = .764
MAXIMUM VARIANCE / MINIMUM VARIANCE = 1.996

MODEL II - COMPONENTS OF VARIANCE.

ESTIMATE OF BETWEEN COMPONENT 5.1147054+01

*****

SONEWAY for (C) group no in (C) put statistics in (C) and next three columns

SONEWAY for (C) with (C), put group in (C), number in (C), means (C), s.d. (C)

4.4 Analysis Of Two-way Table.

TWOWAY, STWOWAY

The TWOWAY instruction produces an automatic printing of an analysis of variance for a two-way crossed classification without replication; fixed effects, infinite model. One form of the instruction is for use in the balanced case (4 arguments) and one form for use in the unbalanced case (5 arguments). Before proceeding with a description of the two forms of the instruction, a very brief statistical background is given, which in turn will be related to an example.

Remarks which hold for both forms of the instruction follow the statistical background. Remarks which pertain to one form only are given in the description of the instruction.

Statistical Background.

Suppose we have a two-way table with three rows and four columns, as illustrated, for measurements y_{ij}:

	Column 1	Column 2	Column 3	Column 4
Row 1	y_{11}	y_{12}	y_{13}	y_{14}
Row 2	y_{21}	y_{22}	y_{23}	y_{24}
Row 3	y_{31}	y_{32}	y_{33}	y_{34}

A common statistical (additive) model in this situation is:

$$y_{ij} = \mu + \rho_i + \gamma_j + \varepsilon_{ij},$$

where i = 1, 2, ... ,r=3; j = 1, 2, ... ,c=4; the ε's are assumed to behave like independent, random, normal variables with mean zero and variance σ^2. Also, the constraints

$$\sum_{i=1}^{r}\rho_i = \sum_{j=1}^{c}\gamma_j = 0$$

are imposed, so that

$$\rho_3 = -\rho_1 - \rho_2 \quad \text{and} \quad \gamma_4 = -\gamma_1 - \gamma_2 - \gamma_3.$$

The parameter μ represents the overall mean, ρ_i represents the effect of the ith row, and the parameter γ_j represents the effect of the jth column.

In the least squares framework this model can be rewritten in the equivalent form

$$y_k = \beta_1 x_{1k} + \beta_2 x_{2k} + \beta_3 x_{3k} + \beta_4 x_{4k} + \beta_5 x_{5k} + \beta_6 x_{6k} + e_k$$

where k = 1, 2, ... , rc=12, and

$$\beta_1 = \mu, \quad \beta_2 = \rho_1, \quad \beta_3 = \rho_2, \quad \beta_4 = \gamma_1, \quad \beta_5 = \gamma_2 \quad \text{and} \quad \beta_6 = \gamma_3$$

are parameters to be estimated and the x's are fixed known constants. The constant x_{1k} is identically equal to one,

$x_{2k} = 1,$	$x_{3k} = 0,$	if y_k is in the first row;
$0,$	$1,$	if y_k is in the second row;
$-1,$	$-1,$	if y_k is in the third row;

the constants x_{4k}, x_{5k} and x_{6k} are defined in a similar manner. The constants x_{2k} and x_{3k} equal -1 for each measurement in the third row, because of the constraint imposed upon the row effects.

The above model can be conveniently expressed in matrix notation as $y = X\beta + e$. See pages 257 and 258 of Draper and Smith (1968). For clarity, the matrices are written out in full below.

MATRIX NOTATION

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{24} \\ y_{31} \\ y_{32} \\ y_{33} \\ y_{34} \end{bmatrix} = \begin{array}{c} \begin{matrix} \mu & \rho_1 & \rho_2 & \gamma_1 & \gamma_2 & \gamma_3 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & -1 & -1 & -1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} \end{array} \begin{bmatrix} \mu \\ \rho_1 \\ \rho 2 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} + \begin{bmatrix} e_{11} \\ e_{12} \\ e_{13} \\ e_{14} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{24} \\ e_{31} \\ e_{32} \\ e_{33} \\ e_{34} \end{bmatrix}$$

These matrices can be clearly generalized for arbitrary values of r and c. Note, the design matrix, X, is constructed with the constraints on row and column effects imposed.

Use of Instruction.

Let the first four arguments of the instruction be denoted by r, c, Y and X, all

integers. The first two arguments, r and c, specify the number of rows and columns in the two-way table. The third argument, Y, is the column containing the measurements. Data are entered into column Y one row at a time so that all the measurements in row 1 are entered first, then all the measurements in row 2, and so on; as indicated by the vector y in the MATRIX NOTATION table above. The fourth argument, X, designates the first column where automatic storage begins; see below.

Restrictions.

The following restrictions apply to both forms of the instruction. Additional restrictions on the use of weights are given in the description of the second form of TWOWAY.

(1) The product rc must equal NRMAX.
(2) Both r and c must exceed 1.
(3) The column number Y must be less than the column number X,
(4) (X+r+c+2) must not exceed the number of columns in the worksheet.
(5) 2(r+c+4) must not exceed the number of rows in the worksheet.
(6) For s = smaller of r and c, t = larger of r and c, the following table shows the largest permissible value of t for each possible value of s:

s	t	s	t	s	t
2	51	7	31*	12	22*
3	45	8	29*	13	20*
4	40	9	27*	14	19*
5	37	10	25*	15	18*
6	34*	11	23*	16	17*

The asterisk indicates that the worksheet has to be redimensioned for the largest tables.

If any of the above restrictions are violated, the following fatal errors will occur:

(1) *** ILLEGAL SIZE ROW NUMBER
(2) *** ILLEGAL ARGUMENT ON CARD
(3) *** ILLEGAL ARGUMENT ON CARD
(4) *** DEFINED MATRIX OVERFLOWS WORKSHEET
(5) *** DEFINED MATRIX OVERFLOWS WORKSHEET
(6) *** INSUFFICIENT SCRATCH AREA

Automatic Printing.

Different analyses of variance are printed for each form of the instruction. See the descriptions and examples below. The F-ratio is always the MEAN SQUARE on the same line divided by the RESIDUAL MEAN SQUARE with the corresponding degrees of freedom. The F PROB. values are the significance levels of the F-ratios. They give the probability that the observed F-ratio will be exceeded, if in fact there are no differences between row or column effects, as the case may be, i.e., if all row (or column) effects equal zero.

Each form prints a table containing the effects, the estimates of the effects and the standard deviations of the estimates. The standard deviations of the estimates are functions of the residual standard deviation. If the additive model is correct and if the assumptions concerning the measurement errors are correct, then the residual standard deviation is an estimate of σ. Note, the row (column) estimates are estimates of effects, not means. The estimate of a row (column) mean would be the sum of the estimate of the mean and the estimate of the row (column) effect.

The second page of the printing contains a table of standardized residuals; standardized by dividing each residual by its own standard deviation. See section 4.5 for further information on standardized residuals. Each residual is printed with two digits after the

decimal point. If the number of columns, c, exceeds 15, the printing of a row of residuals continues on the next line. A string of six asterisks indicates the residual is equal to or greater than 1000.00. (F6.2 format used.) When this happens, it is a sign of some difficulty; either there is an error in the set of instructions or there is some problem with the data. All printing is done using either "readable" or fixed formats.

Automatic Storage.

The instruction automatically stores results in (r+c+3) consecutive columns, starting with column X and continuing through column (X+r+c+2), as follows:

(1) The (rc) x (r+c-1) design matrix in columns X through (X+r+c-2), inclusive.
(2) Coefficients in 2(r+c+4) rows of column (X+r+c-1).
(3) Residuals in rc rows of column (X+r+c).
(4) Standard deviations of predicted values in rc rows of column (X+r+c+1).
(5) Fourier coefficients in 2(r+c) rows of column (X+r+c+2).

The design matrix is very useful for using a FIT instruction in further work; see examples below. The values in the last four columns are essentially the same as those which would be obtained using the appropriate FIT instruction. The coefficient column (X+r+c-1) contains 4 additional values; the estimates and standard deviations for the rth row and cth column. As a consequence of the constraints on the row and column effects, the estimates of the rth row effect and the cth column effect are the negative of the sum of the first (r-1) row estimates and the first (c-1) column estimates, respectively. Note, 2(r+c+4) or 2(r+c) may exceed NRMAX. Consequently, NRMAX may have to be reset in order to print the entire contents of the Coefficients and Fourier Coefficients columns.

The column of coefficients contains:

Rows	Description	No. of Rows
1	estimate (coefficient) of mean (effect)	1
2 to (r+1)	estimates of the r row effects	r
(r+2) to(r+c+1)	estimates of the c column effects	c
(r+c+2)	standard deviation of the mean	1
(r+c+3) to (2r+c+2)	standard deviations of the r row estimates	r
(2r+c+3) to 2(r+c+1)	standard deviations of the c column estimates	c
2(r+c)+3 to 2(r+c+4)	the six values: number of non-zero weights number of vectors in design matrix - (r+c-1) residual degrees of freedom residual standard deviation residual variance multiple correlation coefficient squared.	6

In the Fourier coefficients column (X+r+c+2), using SS for sum of squares, the following quantities are stored:

(i) Row 1 gives the SS due to the mean.
(ii) The sum of rows 2 to r gives the Between Rows SS.
(iii) The sum of rows (r+1) to (r+c-1) gives the Between Cols SS.
(iv) Row (r+c) contains the Residual SS.
(v) Row (r+c+1) contains the Total (uncorrected) SS.
(vi) Row (r+c+2) contains the first Fourier coefficient.
(vii) Rows (r+c+3) to (2r+c+1) contain the (r-1) row Fourier coefficients.
(viii) Rows (2r+c+2) to 2(r+c) contain the (c-1) column Fourier coefficients.

Note, the Total (uncorrected) SS is the (weighted) sum of the squared measurements. The TOTAL SUM OF SQUARES in the automatic printing is the corrected Total SS, i.e., the uncorrected Total sum of squares minus the sum of squares due to fitting the mean.

Computing Method.

A least squares approach is taken and the computing algorithm is essentially the same as that used in the FIT instruction (section 4.5). In this way, balanced tables and unbalanced tables are treated similarly, to the advantage of the user. In a restricted sense, TWOWAY is special case of FIT. A balanced two-way table (first form of TWOWAY) is a special case of an unbalanced table using all weights equal to one. The results are believed to be very accurate. In the event the sums of squares in the analysis of variance do not sum accurately, the following informative diagnostic is given:

```
* SUM OF SQRS DO NOT ADD UP-ABS. VALUE OF (TOTAL-ROW-COL-RES)/TOTAL EXCEEDS 5.E-7
```

```
:-------------------------------------------------------------------------:
: TWOWAY analysis for (r) by (c) table, data in col (C) put in (C) and succ. cols. :
:-------------------------------------------------------------------------:
```

Automatic Printing.

This form of TWOWAY also prints an analysis of variance to perform Tukey's test for non-additivity. The instruction TWOWAY assumes the measurements can be represented by an additive model. It is a well known, yet often overlooked, fact that additivity of effects depends on the scale of measurement used. Since the scale of measurement is often, at least in a certain sense, arbitrary, non-additivity is often present. Tukey's test enables the user to assess the adequacy of an additive model. Small values of the significance level, F PROB., indicate the presence of non-additivity. If the model is multiplicative, it may be appropriate to reanalyze the data using some form of transformation. Tukey's test is described in Graybill (1961), starting on page 324.

Example.

Pages 132 and 133 show the automatic printing and set of instructions to analyze the 3x4 table on page 331 of Graybill (1961). Notice that the data after the SET instruction look like a two-way table, but are entered into the worksheet as a single column.

After the TWOWAY instruction on page 133, the instruction

```
FIT 1, 1.0, 6, 11 *** 16
```

would provide additional information. In particular, it would give (i) the four per page plot of standardized residuals to supplement the table given by TWOWAY, (ii) print the predicted values, and (iii) print the variance-covariance matrix, which could be used to calculate the variance of linear combinations of the row or column estimates.

Automatic Storage in Example.

The fourth argument of the TWOWAY instruction on page 133 is X=11. Hence, the design matrix is stored in the 6=r+c-1 columns 11 (X) through 16 (X+r+c-2), the coefficients are in column 17, the residuals in column 18, the standard deviations of predicted values are in column 19, and the Fourier coefficients are in column 20. A print of columns 11 to 16 would reveal the same design matrix shown as the matrix X in the MATRIX NOTATION table on page 128. Note, column 11 has values in 22 (2(r+c)+8) rows and column 20 has values in 14 (2(r+c)) rows, whereas NRMAX=12.

The numbers under ESTIMATE on page 132 are stored in row 1, rows 2 to 4, and rows 5 to 8 of column 17. The numbers under STD. DEV. are stored in rows 9, 10 to 12, 13 to 16, and 20 of column 17. Rows 17, 18, 19, 21 and 22 contain the numbers 12., 6., 6., 3.8055554 and .58421852, which are the number of non-zero weights, the number of vectors in the design matrix, residual variance and squared multiple correlation coefficient.

ANALYSIS OF VARIANCE FOR TWO-WAY 3 X 4 TABLE

SOURCE	D.F.	SUM OF SQUARES	MEAN SQUARES	F RATIO	F PROB.
BETWEEN ROWS	2	7.1666665	3.5833333	.942	.441
BETWEEN COLS	3	24.916666	8.3055551	2.182	.191
RESIDUALS	6	22.833332	3.8055554		
TOTAL	11	54.916668			

TUKEY'S TEST FOR NON-ADDITIVITY

NON-ADDITIVITY	1	7.9069896	7.9069896	2.649	.165
BALANCE	5	14.926343	2.9852685		
RESIDUALS	6	22.833332	3.8055554		

COEFFICIENT		ESTIMATE	STD. DEV.
GRAND MEAN		2.4166667	.56314261
ROW	1	1.0833333	.79640393
ROW	2	-.41666666	.79640394
ROW	3	-.66666667	.79640394
COLUMN	1	2.2499999	.97539163
COLUMN	2	-1.4166666	.97539163
COLUMN	3	-1.0833333	.97539164
COLUMN	4	.25000001	.97539164
RESIDUAL			1.9507833

3 X 4 TABLE OF RESIDUALS, STANDARDIZED BY DIVIDING EACH RESIDUAL BY ITS STANDARD DEVIATION.

COLUMN / ROW	1	2	3	4
1	1.63	-.06	-1.03	-.54
2	-.18	-.42	1.51	-.91
3	-1.45	.48	-.48	1.45

LIST OF COMMANDS, DATA AND DIAGNOSTICS

```
SET DATA IN COLUMN 1 ROW BY ROW
   8   2   1   3
   4   0   3   1
   2   1   0   4
TWOWAY ANALYSIS FOR 3X4 TABLE IN COL 1 STORE STARTING IN COL 11
```

NATIONAL BUREAU OF STANDARDS, WASHINGTON, D. C. 20234, OMNITAB II VERSION 5.00 MAY 15, 1970

The numbers under SUM OF SQUARES on page 132 are obtained from the values in column 20. The Between Rows SS (7.1666665) is the sum of the values in rows 2 and 3. The Between Cols SS (24.916666) is the sum of the values in rows 4, 5 and 6. The Residual SS is in row 7 (r+c). The corrected Total SS (54.916668) equals the uncorrected Total SS (125.00000) in row 8 (r+c+1) minus the SS due to the mean (70.083333) in row 1.

Remark.

The following three instructions, after TWOWAY, would print a table of predicted values similar to the table of standardized residuals.

```
SUBTRACT col 18 from col 1 and put in col 2
MMATVEC column 2 into 1,31 size 3x4
MPRINT array in 1,31 of size 3x4
```

```
/------------------------------------------------------------------------\
/ TWOWAY anal. for (r) by (c) table, data in (C) store from (C) on, wts in col (C) /
/------------------------------------------------------------------------/
```

This form of TWOWAY has an additional argument, the last denoted by W, to specify a column of weights. It is useful for analyzing experiments with missing observations, balanced incomplete block designs, partially balanced incomplete block designs (intra-block analysis), etc. or even undesigned experiments. A useful procedure is to assign a weight of 1 to each measurement present and to assign a weight of 0 if a measurement is missing. If each cell in a table contains more than one measurement, TWOWAY can be used with the averages of each cell and using the number of measurements in each cell as the weight. Weights are entered into column W, one row below another, just as is done for the measurements in column Y.

Restrictions on Weights.

(1) The column number W must be less than the column number X.
(2) The column number W must not equal the column number Y.
(3) Weights can not be negative.
(4) There must be at least one positive weight in each row.
(5) There must be at least one positive weight in each column.
(6) The number of weights equal to zero must be less than (r-1)(c-1).

Violations of these restrictions result in the following error messages. In each case, the instruction is not executed.

```
(1)  *** ILLEGAL ARGUMENT ON CARD
(2)  *** ILLEGAL ARGUMENT ON CARD
(3)  *** NEGATIVE WEIGHTS MAY NOT BE USED
(4)    * ALL WEIGHTS ARE ZERO. COMMAND NOT EXECUTED
(5)    * ALL WEIGHTS ARE ZERO, COMMAND NOT EXECUTED
(6)    * DEGREE IS LARGER THAN NO. OF NON-ZERO WEIGHTS.
```

Caution. The above restrictions can be met and yet there exist patterns of weights which cause difficulty and no diagnostic is given. The matrix X'WX (X the design matrix and W the diagonal matrix with weights on the diagonal) can be singular. We know of no solution to this problem. These situations are not likely to occur, but they can happen. Unusually large or small values of the coefficients and/or standard deviations may indicate trouble. If the user is in doubt, the matrix X'WX can be examined (inverted) or the *ACC. DIGITS column can be checked in the automatic printing after using the appropriate FIT instruction. An example (provided by B. L. Joiner and J. R. Rosenblatt) is one where the weights for a 3x4 table are as follows:

Row/Column	1	2	3	4
1	1	0	0	0
2	0	1	1	1
3	0	1	1	1

Here, the singular X'WX matrix is

$$\begin{bmatrix} 7 & -2 & 0 & -1 & 0 & 0 \\ -2 & 4 & 3 & 2 & 0 & 0 \\ 0 & 3 & 6 & 0 & 0 & 0 \\ -1 & 2 & 0 & 3 & 2 & 2 \\ 0 & 0 & 0 & 2 & 4 & 2 \\ 0 & 0 & 0 & 2 & 2 & 4 \end{bmatrix}$$

This means that one of the row or column effects is not estimable. Clearly, the effect of row 1 plus column 1 is estimable, but the row 1 and column 1 effects, separated, are not estimable. (The rank of X'WX is 5 because column (6) is the linear combination of columns (1) through (5): 3(6) = -(1)-8(2)+4(3)+9(4)-3(5).) The instructions

```
TWOWAY 3x4, 11, 21, 12
FIT 11, 12, 6, 21 *** 26
```

would show -1.75 in every row in the column headed *ACC. DIGITS, which is a definite sign of trouble. If MINVERT were used to invert the (X'WX) matrix the error bound would be -.3+10; again, indicating trouble.

Automatic Printing.

The automatic printing shows two analyses of variance, rather than one, necessitated by the unbalance in the table. The first analysis shows Rows adjusted for columns and Columns unadjusted. The second analysis shows Rows unadjusted and Columns adjusted for Rows. After the printing of the analyses of variance, a line is printed which tells the number of non-zero weights used and the number of weights equal to zero. In this form of the instruction, Tukey's test for non-additivity is not performed. The printing of the coefficients and their standard deviations has the same form as that for the balanced case. In the table of standardized residuals, .00 appears in each cell for which the weight is zero.

Automatic Storage.

The automatic storage for this form of TWOWAY is the same as that described above. However, since the two-way table is unbalanced, the Fourier coefficients have little meaning. The Residual Sum of Squares, the Total (uncorrected) Sum of Squares, and the Sum of Squares due to the mean are in the three rows (r+c), (r+c+1) and 1. All other numbers in the column should be ignored.

Example.

Pages 136 and 137 show the automatic printing and set of instructions to give an intra-block analysis of the data from a partially balanced incomplete block design described on page 561 of Kempthorne (1952). There are 9 treatments and 9 blocks, with 3 treatments in each block. However, the data following the SET instruction are arranged like a 9x9 table with 9 treatments per block. The value zero is assigned to a treatment which does not appear in a block and correspondingly a weight of zero is assigned. Here, blocks are rows and treatments are columns. Hence, in the automatic printing, the first analysis is for Blocks (rows) eliminating treatments and Treatments (columns) ignoring blocks. This is the analysis on the right hand side of the analysis on page 562 of Kempthorne (1952). The second analysis is for Blocks ignoring treatments and Treatments eliminating blocks.

ANALYSIS OF VARIANCE FOR TWO-WAY 9 X 9 TABLE

SOURCE	D.F.	SUM OF SQUARES	MEAN SQUARES	F RATIO	F PROB.
BETWEEN ROWS	8	1473.2962	184.16203	13.808	.000
BETWEEN COLS	8	1145.1851	143.14814		
RESIDUALS	10	133.37037	13.337037		
TOTAL	26	2751.8519			
BETWEEN ROWS	8	2520.5183	315.06479		
BETWEEN COLS	8	97.962957	12.245370	.918	.539
RESIDUALS	10	133.37037	13.337037		
TOTAL	26	2751.8519			

A WEIGHTED LEAST SQUARES ANALYSIS USING 27 NON-ZERO WEIGHTS AND 54 ZERO WEIGHTS IN COLUMN 11

COEFFICIENT		ESTIMATE	STD. DEV.
GRAND MEAN		47.925926	.70282595
ROW	1	5.3518520	2.3310100
ROW	2	-9.2592589	2.3310100
ROW	3	-1.8148145	2.3310100
ROW	4	5.5740737	2.3310100
ROW	5	10.407407	2.3310100
ROW	6	2.5740739	2.3310100
ROW	7	9.2407402	2.3310100
ROW	8	-2.6481482	2.3310100
ROW	9	-19.425925	2.3310100
COLUMN	1	.40740731	2.3310100
COLUMN	2	-1.4814814	2.3310100
COLUMN	3	-.42592593	2.3310100
COLUMN	4	1.2407408	2.3310100
COLUMN	5	-.98148151	2.3310100
COLUMN	6	.74074067	2.3310100
COLUMN	7	-4.7592591	2.3310100
COLUMN	8	2.3518518	2.3310100
COLUMN	9	2.9074074	2.3310099
RESIDUAL			3.6519908

9 X 9 TABLE OF RESIDUALS, STANDARDIZED BY DIVIDING EACH RESIDUAL BY ITS STANDARD DEVIATION.

COLUMN / ROW	1	2	3	4	5	6	7	8	9
1	.00	.00	.52	-.68	.00	.00	.00	.17	.00
2	.00	-.98	.00	.04	.00	.00	.94	.00	.00
3	.67	.00	.00	.00	-.96	.00	.29	.00	.00
4	.00	.00	.00	.00	.00	.00	-1.23	.07	1.17
5	.00	.00	.00	.64	1.64	-2.28	.00	.00	.00
6	.00	.00	.87	.00	-.68	.00	.00	.00	-.18
7	-1.61	.00	.00	.00	.00	1.84	.00	-.23	.00
8	.00	.54	.00	.00	.00	.44	.00	.00	-.98
9	.94	.44	-1.38	.00	.00	.00	.00	.00	.00

LIST OF COMMANDS, DATA AND DIAGNOSTICS

```
    SET DATA IN COL 10 ROW BY ROW
       0   0  54  53   0   0   0  56   0
       0  35   0  40   0   0  36   0   0
      48   0   0   0  43   0  42   0   0
       0   0   0   0   0   0  46  56  59
       0   0   0  61  61  54   0   0   0
       0   0  52   0  48   0   0   0  53
      54   0   0   0   0  62   0  59   0
       0  45   0   0   0  47   0   0  46
      31  28  25   0   0   0   0   0   0
    DIVIDE COL 10 BY COL 10 AND PUT WEIGHTS IN COL 11

** ARITHMETIC FAULT IN ABOVE COMMAND, ZERO RETURNED  54 TIMES
** DIVISION BY ZERO, RESULT SET=0  54 TIMES

    TWOWAY ANALYSIS OF 9X9 TABLE IN COL 10 PUT RESULTS IN COL 20 WEIGHTS IN COL 11
```

Remark.

The variance of the difference between any two treatment estimates, X and Y, is

$$Var(X-Y) = Var(X) + Var(Y) - 2Cov(X,Y)$$

The variances and covariances can be obtained by using the sixth form of FIT (section 4.5). The following instructions after TWOWAY on page 137 would put, 3.4431299, the standard deviation of the difference between the 3rd and 5th treatment estimates (12th and 14th estimates) in column 62.

```
SFIT 10, 11, 17 vectors in 20 *** 36 and put in 37, 37, 37, 37 and 1,41
ADD the value *2,52* to the value *14,54* and put in column 60
MULTIPLY the value *12,54* by -2.0, mult 1.0, add col 60, put in col 61
SQRT of column 61 put in column 62
```

(Note, here we do not need to know whether treatments 3 and 5 are 1st or 2nd associates.)

*****

```
:-------------------------------------------------------------------------:
: STWOWAY analysis for (r) by (c) table, data in col (C) put in (C) and succ. cols :
:-------------------------------------------------------------------------:

/-------------------------------------------------------------------------/
/ STWOWAY anal. for (r) by (c) table, data in (C) store from (C) on, wts in col (C) /
/-------------------------------------------------------------------------/
```

4.5 Regression.

FIT, POLYFIT, SFIT, SPOLYFIT

There are six different forms of FIT and POLYFIT and five different forms of SFIT and SPOLYFIT. Brief descriptions of the different forms of FIT and POLYFIT are given on pages 152 to 154. These descriptions are preceded by a detailed discussion under the titles Weights, Restrictions, Example, Automatic Printing, Automatic Storage, Computing Method, and Remarks.

The instruction FIT performs a least squares analysis (regression) of a set of data. Measurements y_i, i=1, 2, ... ,n=NRMAX, are assumed to be functionally related to k vectors x_1, x_2, ... ,x_k of fixed, known constants. The statistical model

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_k x_{ki} + e_i,$$

is used, where the β's are parameters (coefficients) to be estimated and the e's are measurement errors. The measurement errors are assumed to be independent with mean zero and variance σ_i^2. In an unweighted analysis, $\sigma_i^2 = \sigma^2$, a constant. In the analysis of variance, it is further assumed that the errors are normally distributed. A summary of the formulas and notation used in this section is given on page 144.

Since POLYFIT is actually a special case of FIT, the two instructions are described jointly. In the polynomial of degree d

$$y_i = \alpha_o + \alpha_1 x + \alpha_2 x^2 + \ldots + \alpha_d x^d + e_i,$$

if x^r is set equal to x_{r+1} and α_r is set equal to β_{r+1}, r = 0, 1, ... ,d, the model is equivalent to the one above for FIT with d = k+1.

Weights.

Weighted least squares is performed. The second argument of the instruction designates the weights to be used. If an unweighted analysis is desired, i.e., equal weights all equal to 1.0, set the second argument equal to 1.0. Otherwise, use a column of weights.

It is assumed that the measurement errors are normally and independently distributed with variance given by

$$V(y_i) = \sigma_i^2 = k_i\sigma^2.$$

In an unweighted (all weights = 1.0) analysis, $k_i = 1.0$ for all measurements and the measurements have constant variance. In the presence of heterogeneous (unequal) variance, the "best" weights to use are $w_i = 1/k_i$. If the variances σ_i^2 (or k_i) are known, there is no problem. If the variances are unknown, the weights have to be estimated. If precise estimates are unavailable, the advantage of using weights is dubious. Note, an analysis with k non-zero weights is identical to the analysis obtained using only the measurements with non-zero weights. It is as though the measurements with zero weight were ignored.

Restrictions.

(1) Negative weights may not be used.
(2) The number of non-zero weights must be greater than or equal to the number of vectors fitted (k or d+1).
(3) Below are the maximum number of measurements, NRMAX, which can be used for a given number of vectors, k. In FIT, k is the third argument. In POLYFIT, k equals the degree (3rd argument) plus 1. (Values given are for NBS computer.)

k	NRMAX	k	NRMAX	k	NRMAX	k	NRMAX	k	NRMAX	k	NRMAX
1	2246	11	813	21	463	31	293	41	187	51	109
2	1922	12	761	22	441	32	281	42	178	52	103
3	1678	13	714	23	421	33	269	43	170	53	96
4	1487	14	672	24	402	34	257	44	161	54	90
5	1335	15	634	25	384	35	246	45	153	55	83
6	1209	16	599	26	367	36	235	46	146	56	77
7	1104	17	567	27	350	37	225	47	138	57	71
8	1015	18	538	28	335	38	215	48	131	58	65
9	939	19	511	29	321	39	205	49	123	59	59
10	872	20	486	30	307	40	196	50	116	60	***

Violations of these restrictions result in the following fatal errors, respectively:

(1) *** NEGATIVE WEIGHTS MAY NOT BE USED
(2) *** DEGREE IS LARGER THAN NO. OF NON-ZERO WEIGHTS
(3) *** INSUFFICIENT SCRATCH AREA

Example.

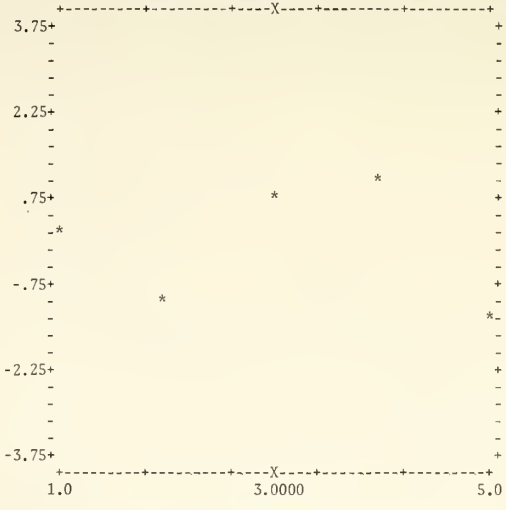
The following set of instructions produce the automatic printing on pages 140 to 143 to perform a straight-line regression using the data in section 6.2 of Snedecor and Cochran (1967).

```
OMNITAB 6/9/69  EXAMPLE OF STRAIGHT LINE FIT, SNEDECOR, COCHRAN PP 135-140
    SET Y = MEAN SYSTOLIC BLOOD PRESSURE IN COL 1
    114, 124, 143, 158, 166
    SET X = MIDPOINT OF AGE CLASS IN COLUMN 3
    35, 45, 55, 65, 75
    DEFINE 1.0 INTO COLUMN 2
    FIT COL 1 WTS 1.0, 2 VEC'S IN COLS 2 AND 3
```

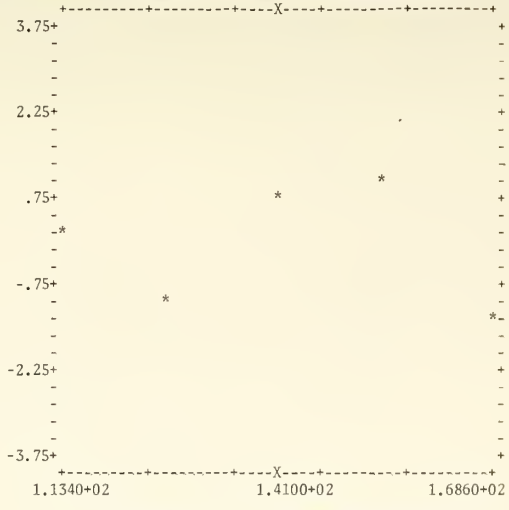
LEAST SQUARES FIT FOR DATA IN COLUMN 1
AS A LINEAR FUNCTION OF 2 PREDICTOR VARIABLES IN COLUMNS 2, 3
USING 5 NON-ZERO WEIGHTS = 1.0000000

ROW	VARIABLE X IN COLUMN 3	DATA IN COLUMN 1	PREDICTED VALUES	STD. DEV. OF PRED. VALUES	RESIDUALS	STD. RES.	WEIGHTS
1	35.000000	114.00000	113.40000	2.5139609	.59999935	.29	1.000
2	45.000000	124.00000	127.20000	1.7776388	-3.2000003	-1.18	1.000
3	55.000000	143.00000	141.00000	1.4514360	2.0000000	.69	1.000
4	65.000000	158.00000	154.80000	1.7776388	3.2000003	1.18	1.000
5	75.000000	166.00000	168.60000	2.5139609	-2.5999994	-1.27	1.000

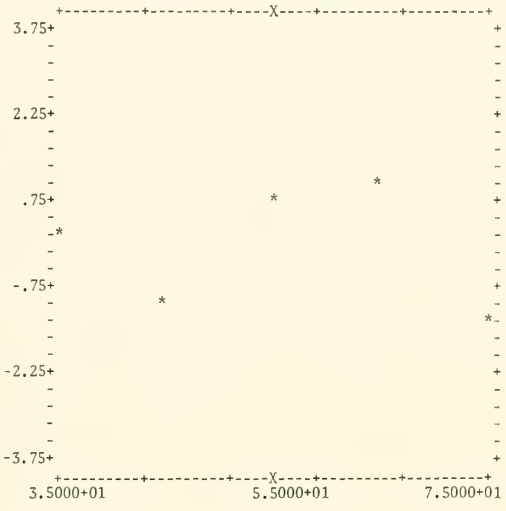
STANDARDIZED RESIDUALS VS ROW NUMBER



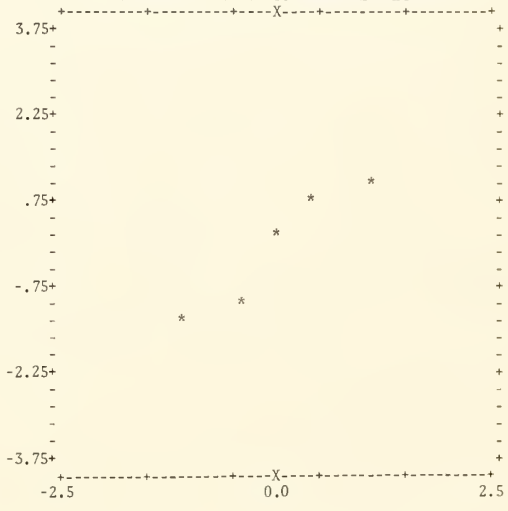
STANDARDIZED RESIDUALS VS PREDICTED VALUES



STANDARDIZED RESIDUALS VS VARIABLE X



PROBABILITY PLOT OF STANDARDIZED RESIDUALS



LEAST SQUARES FIT FOR DATA IN COLUMN 1
AS A LINEAR FUNCTION OF 2 PREDICTOR VARIABLES IN COLUMNS 2, 3
USING 5 NON-ZERO WEIGHTS = 1.0000000

VARIANCE-COVARIANCE MATRIX OF THE ESTIMATED COEFFICIENTS

COLUMN	2	3
2	33.969998	
3	-.57933331	.010533333

ANALYSIS OF VARIANCE
-DEPENDENT ON ORDER VARIABLES ARE ENTERED, UNLESS VECTORS ARE ORTHOGONAL-

COLUMN	SS=RED. DUE TO COEF.	CUM. MS REDUCTION	D.F.	CUM. RESIDUAL MS	D.F.	F(COEF=0)	P(F)	F(COEFS=0)	P(F)
2	99404.994	99404.994	1	487.99998	4	9437.183	.000	4808.990	.000
3	1904.3999	50654.697	2	10.533333	3	180.797	.001	180.797	.001
RESIDUAL	31.599998		3						
TOTAL	101341.00		5						

LEAST SQUARES FIT FOR DATA IN COLUMN 1
AS A LINEAR FUNCTION OF 2 PREDICTOR VARIABLES IN COLUMNS 2, 3
USING 5 NON-ZERO WEIGHTS = 1.0000000

	ESTIMATES FROM LEAST SQUARES FIT				FIT OMITTING LAST COLUMN		
COLUMN	COEFFICIENT	S.D. OF COEFF.	RATIO	*ACC. DIGITS	COEFFICIENT	S.D. OF COEFF.	RATIO
2	65.100002	5.8283787	11.17	8.00	141.00000	9.8386989	14.33
3	1.3800000	.10263203	13.45	8.00			
RESIDUAL STANDARD DEVIATION =		3.2455097			22.000000		
BASED ON DEGREES OF FREEDOM		5- 2 = 3			5- 1 = 4		

* THE NUMBER OF CORRECTLY COMPUTED DIGITS IN EACH COEFFICIENT USUALLY DIFFERS BY LESS THAN 1 FROM THE NUMBER GIVEN HERE

Notation and Formulas for FIT

1.	Number of measurements: NRMAX - number of zero weights	n
2.	Number of predictor variables (vectors):	k
3.	Measurements: y_i, i=1,2,...,n	y_{nx1}
4.	Predictor variables x_1, x_2 ... x_k	$X_{nxk} = (x_1, x_2, \ldots, x_k)$
5.	Measurement errors: e_1, e_2 ... e_n	e_{nx1}
6.	Error variance:	$\sigma_i^2 = k_i\sigma^2$
7.	Unknown parameters: β_1, β_2 ... β_k	β_{kx1}
8.	Statistical model: errors independent, normal, variance σ_i^2	$y = X\beta + e$
9.	Weights: "best" weights $w_i = 1/\sigma_i^2$ or $1/k_i$	$W = \text{diag}(w_i = 1/k_i)$
10.	Coefficients: estimates of β	$\hat{\beta} = (X'WX)^{-1}X'Wy$
11.	Predicted values:	$\hat{y} = X\hat{\beta}$
12.	Residuals: deviations from predicted values	$z = y - \hat{y}$
13.	Residual standard deviation: $s = \sqrt{\Sigma w_i z_i^2/(n-k)}$	$s = \sqrt{(y-\hat{y})'W(y-\hat{y})/(n-k)}$
14.	Variance-covariance matrix:	$V = s^2(X'WX)^{-1}$
15.	Standard deviation of predicted values:	$\sqrt{}$diagonals of XVX'
16.	Standard deviation of coefficients:	$\sqrt{}$diagonals of V
17.	Standardized residuals:	$z_i = \sqrt{(s^2/w_i - s_{\hat{y}}^2)}$
18.	Orthonormalization: X'WX = TT', T triangular, $A = (T')^{-1}$	$\Phi = XA$
19.	Fourier coefficients:	$\hat{\alpha} = \Phi Wy$
20.	Squared Fourier coefficients:	$\hat{\alpha}_i^2$
21.	Sum of squares: Reduction in total SS due to fitting	$S_i = \hat{\alpha}_i^2$
22.	Residual sum of squares: $(n-k)s^2 = \hat{\alpha}_{k+1}^2$	$S_{k+1} = (y-\hat{y})'W(y-\hat{y})$
23.	Total (uncorrected) sum of squares: $\Sigma w_i y_i^2$	y'Wy
24.	Cumulative mean square reduction: with r degrees of freedom	$\Sigma_1^r S_i/r$
25.	Cumulative residual mean square: with (n-r) deg. of freedom	$\sum_{r+1}^{k+1} S_i/(n-r)$
26.	F ratio for testing Ho: $\beta_r = 0$	S_r/Res. Mean Square
27.	F ratio for testing Ho: $\sum_r^k \beta_i = 0$	$[\sum_r^k S_i(k-r+1)]$/Res. MS
28.	Gram factors: $G = (x_i'Wx_1)^{\frac{1}{2}}$ $G_r = [(x_r'Wx_r) - (x_r'W\phi_1)^2 - \ldots - (x_r'W\phi_{r-1})^2]^{\frac{1}{2}}$	
29.	Vector norms:	$(x_r'Wx_r)^{\frac{1}{2}}$
30.	Gram determinants:	$\Pi G_i^2/(x_i'Wx_i)$
31.	Multiple correlation coefficient squared:	$R^2 = 1 - \text{Res. SS}/\sum w_i(y_i - \bar{y}_w)^2$
32.	Accurate digits: obtained from refit to predicted values	$-\log_{10}\lvert\hat{\beta} - \tilde{\beta}/\tilde{\beta}\rvert$

The data in Snedecor and Cochran (1967) were selected because the book is easily available and they show many of the calculations required to obtain the results.

The instructions on page 139 illustrate the use of FIT. Here, POLYFIT could be used more simply by replacing the DEFINE and FIT instructions above by

POLYFIT y in col 1, weights 1.0, degree 1, x in column 2

The automatic printing would be essentially the same as that shown on pages 140 to 143.

Automatic Printing.

The automatic printing consists of "four" pages containing a comprehensive set of results for analyzing data. Four is put in quotation marks because the printing of results on one page may be continued on subsequent pages if the number of measurements or number of predictor variables is large. Each "page" is described separately below.

A three line title appears at the top of pages one, three and four. The title in the example on pages 140, 142 and 143 is

```
          LEAST SQUARES FIT FOR     DATA     IN COLUMN    1
AS A LINEAR FUNCTION OF  2 PREDICTOR VARIABLES IN COLUMNS   2,   3
          USING    5 NON-ZERO WEIGHTS = 1.0000000
```

The following are variations in the title which can occur: (1) on the first line 12 spaces are allowed for the word DATA. If a HEAD instruction (section 1.4) has been used, the 12 characters after the slash (/) in the HEAD instruction are used, (2) if there are more than 11 predictor variables, the column numbers are continued on the next line, and (3) on line 3, the number of zero weights is given if the second argument is a column number.

In a POLYFIT instruction, the second line of the title has the form

```
AS A POLYNOMIAL OF DEGREE  1 IN  VARIABLE X  IN COLUMN   11
```

The 12 characters used for VARIABLE X, are replaced by those following the slash in a HEAD instruction, if one has been used.

Page One

The following 8 items are printed for each of the (NRMAX) measurements: (1) the row in the worksheet, (2) the predictor variable, (3) the data (measurements), (4) the predicted values, (5) the standard deviations of the predicted values, (6) the residuals, (7) the standardized residuals, and (8) the weights used. Items (3), (4), (5) and (6) are printed with 8 significant digits. The standardized residuals are printed with 2 decimal places and 4 significant digits are used to print the weights. The standardized residuals are the residuals divided by their standard deviation. Some remarks on (2) and (3) follow.

(2) (a) POLYFIT. The 12 characters in the first line of the column heading use either VARIABLE X or the 12 characters after the slash in the HEAD instruction, if one has been used. Eight significant digits are used to print the predictor variable.

(b) FIT. Either one, two or a maximum of three columns is printed. If a constant is the first term in the model (which usually is the case), the column of 1's is not printed (unless it is the only vector used). Otherwise, starting with the first vector not identically equal to one, as many predictor variables are printed as is possible with a maximum of three. Eight, six or four significant digits are used, depending upon whether 1, 2 or 3 columns are printed. If the values of either two or three predictor variables are printed, HEAD column is ignored and the word COLUMN is abbreviated to COL..

(3) The column heading DATA is replaced by the 12 characters after the slash in the HEAD instruction, if it has been used.

Page Two

This page contains four plots of the standardized residuals. The standardized residuals are used instead of the residuals to (i) have a common vertical scale for all plots going from -3.75 to +3.75 in steps of 0.3, and (ii) to avoid distortion, in some cases, due to differences in the standard deviations of the residuals. Item (i) avoids using a different scale for each set of measurements and makes it possible to present the results in a more compact form. Only residuals associated with non-zero weights are used in the plots. An examination of these plots can be very helpful in assessing the adequacy of the statistical model used. Non-random patterns and/or very large (small) values are evidence of one kind or another of failure of the model to represent the data. Good discussions of the examination of residuals may be found in Chapter 3 of Draper and Smith (1968) and Anscombe and Tukey (1963). All TITLE instructions are ignored in the printing of this page due to lack of space.

(1) Upper Left. The standardized residuals are plotted against the order in which the measurements are entered into the worksheet (row number). If this order corresponds to the order in which the measurements were taken, patterns of non-randomness indicate that time has an effect on the measurements.

(2) Upper Right. The standardized residuals are plotted against the predicted values. Non-randomness may indicate non-constant variance or that some important variable(s) has been excluded from the model. The former case may indicate the need to use weights, the weights used are improper or a transformation of the measurements is required.

(3) Lower Left. The standardized residuals are plotted against the predictor (independent) variable. In a POLYFIT, lack of randomness of the residuals often indicates the need for extra terms in the polynomial. In a FIT, this plot may or may not have much meaning, depending upon the order and character of the predictor variables. The predictor variable used for the plot is the first one which is not identically equal to one (unless the number of vectors is one).

(4) Lower Right. This is a probability plot of the standardized residuals against the expected value of the standardized residuals, assuming that the measurement errors are normally and independently distributed. The ordered standardized residuals are plotted against

$$x_i = \mathrm{Gau}^{-1}(p) \doteq 4.91(p^{.14} - (1-p)^{.14}),$$

where

$$p = (i - \pi/8)/(n + 1 - \pi/4), \quad \text{for } i \geq 2 \text{ or } n > 10$$

$$= (3i - 1)/(3n + 1), \quad \text{for } i = 1 \text{ and } n \leq 10$$

and $i=1,2,\ldots,n$ = the total number of points; x is the inverse (percentage point) of the normal probability integral and the Tukey approximation on the right is used to compute x. We thank James J. Filliben for providing the formula for p, which is somewhat more accurate than the more traditional formulas. If the statistical model adequately represents the data, the points should lie approximately on a straight line. Although the measurement errors may be independent, the residuals are not. This fact is of little consequence if the number of measurements is large compared to the number of predictor variables.

Page Three

There are two sections, which give the variance-covariance matrix and an analysis of variance. In both sections, on the extreme left, the heading COL is used in FIT and TERM in POLYFIT.

(a) Variance-covariance matrix of estimated coefficients. The matrix, $V = s^2(X'WX)$, is symmetric and only the lower triangular portion is printed. If the number of columns exceeds 7, the matrix is printed in blocks. All rows of the first 7 columns are printed in the first block and subsequent blocks are printed as required to complete the triangular matrix. The diagonal entries are the variances of the parameter estimates (coefficients) and the off-diagonal values are the covariances between pairs of estimates. The standard deviation of any linear combination of the estimates, a'b, is the square root of a'Va.

(b) Analysis of variance. This is not the most common analysis of variance, but rather several analyses combined into one. The results depend upon the order in which the vectors are entered (appear in the instruction); unless the vectors are orthogonal. The instructions

FIT 1, 1.0, 3, 11, 12, 13

and

FIT 1, 1.0, 3, 13, 12, 11

would yield the same results in other portions of the printing, but here the results would, in general, differ. The ten columns in the printing are described under (1) through (10) below.

(1) Shows the terms 0, 1, 2, ... , d in a POLYFIT or the column numbers of the vectors used in FIT. These numbers are followed by RESIDUAL and TOTAL.

(2) The second column shows a separation of the total sum of squares, $\sum_1^n w_i y_i^2$, into (k+1) parts

$$S_1 + S_2 + \ldots + S_k + S_{k+1} = S_n$$

with one degree of freedom for each of the k vectors fitted and (n-k) degrees of freedom for the residual. The second sum of squares is the sum of squares due to fitting the second vector after having fit the first. The third (if k exceeds 2) sum of squares is the reduction due to fitting the third vector after having fitted both the first and second vectors, and so on. The entries in this column are the squared Fourier coefficients and are identical to the numbers in the first (k+2) rows in the column of Fourier coefficients discussed under Automatic Storage.

(3) For the rth line of the printing, the cumulative mean square reduction equals

$$\sum_{i=1}^{r} S_i/r \ ,$$

where S_i is the ith sum of squares shown in the second column. In the example on page 142 50654.697 = (99404.994+1904.3999)/2.

(4) The cumulative degrees of freedom 1, 2, ... , k are printed and followed by the degrees of freedom for the residual, (n-k), and for the total, n. The number in the rth line, r, is the number in the denominator of the expression used to compute the cumulative mean square reduction above.

(5) The cumulative residual mean square on the rth line is

$$\sum_{i=r+1}^{k+1} S_i/(n-r)$$

The last number in the column is the residual mean square, which equals the residual sum of squares divided by the residual degrees of freedom. On page 142, 10.533333 = 31.599998/3 and 483.99998 = (1904.3999+31.599998)/4. The square root of the residual mean square (10.533333) equals the residual standard deviation (3.2455096) shown on the following page.

(6) Gives the degrees of freedom for the cumulative residual mean square, which equals (n-r) on the rth line.

(7) The F-ratio printed is the sum of squares (due to coefficient) divided by the residual mean square with 1 and (n-k) degrees of freedom. On page 142, 9437.183 = 99404.994/10.533333. The F-values are printed with 3 decimal places.

(8) Gives the significance level of the F-ratio on the same line in (7), under the hypothesis that the corresponding parameter in the model equals zero. These significance levels should be interpreted carefully. Three decimal places are shown. The value .001 on page 142 is the probability that a F-ratio will exceed 180.797 if there is no linear term in the model.

(9) The F-ratio on the rth line is

$$F = \frac{(\sum_{i=r}^{k+1} S_i - S_{k+1})/(k-r+1)}{\text{Residual Mean Square}}$$

$$= \frac{\sum_{i=r}^{k} S_i/(k-r+1)}{\text{Residual Mean Square}}$$

with (k-r+1) and (n-k) degrees of freedom. It is used to test the hypothesis that all of the parameters r, r+1, ... , k equal zero. On page 142

4808.990 = (99404.994+1904.3999)/2 divided by 10.533333

with 2 and 3 degrees of freedom.

(10) Gives the significance level for the F-ratio on the same line of (9).

Page Four

This page is divided into three parts: (a) estimates, (b) accuracy, and (c) estimates from refit omitting last term.

(a) Least squares estimates of the unknown parameters are printed on the left. On the extreme left, the column heading TERM appears with POLYFIT and the heading COL with FIT. This is followed by a listing of the estimates (coefficients), standard deviations of the estimates and the ratios of the estimates to their standard deviations.

The residual standard deviation and the associated degrees of freedom are printed at the bottom. The degrees of freedom equals the number of non-zero weights minus the number of vectors fitted, k, which is the third argument in a FIT instruction or the third argument plus one in a POLYFIT instruction. If the model is correct (satisfactory) the residual standard deviation is an estimate of σ.

The ratio can be used to perform t-tests and construct confidence intervals for the parameters in the model. However, see references in section 4.8 for a discussion of correct procedures.

(b) Next, a column headed *ACC. DIGITS is printed. The algorithm used in FIT (POLYFIT) is generally accurate, but no least squares fitting algorithm is fully accurate in all cases. (See Longley (1967), Wampler (1969) or Wampler (1970).) Computational accuracy is affected by several factors; in particular (i) the number of measurements and vectors, (ii) the scaling of the measurements and vectors, (iii) how closely the vectors are related (correlated), and (iv) the number of digits in the raw data. The values under *ACC. DIGITS provide an indication of how well the results have been computed. Loosely speaking, values

between 6. and 8. indicate computations are accurate. Values less than 4.0 indicate some source of computing difficulty. Negative values cry out for an investigation.

Mathematically, singular matrices cannot be inverted. However, in a computer, an algorithm sometimes fails to distinguish between a singular matrix and a non-singular matrix. This can happen here. Users will sometimes use a set of vectors which is not of full rank. For example, one of the columns may inadvertently contain all zeros. Any of the following may indicate trouble: (i) a coefficient exactly equal to zero, (ii) extremely large or small coefficients or standard deviations of coefficients, and (iii) small values under *ACC. DIGITS.

The values under *ACC. DIGITS are estimates based upon a refit of the predicted values. If the calculations are accurate, the two calculations of the coefficients should agree. (Mathematically, they are exactly equal.) The estimates of accuracy are reasonably reliable. The smallest and largest possible values are -8.0 and +8.0. If $\hat{\beta}$ is the coefficient and $\tilde{\beta}$ is a coefficient computed from the refit, the accuracy of the computations is given by

$$-\log_{10}|(\hat{\beta} - \tilde{\beta})/\tilde{\beta}|.$$

See, also, ACCURACY in section 2.6. Negative answers indicate the first digit in the coefficient is incorrect. In the example on page 143, the first coefficient has an error of 2 in the eighth digit. The second coefficient is correct to eight digits.

The results here apply to the coefficients and provide a general assessment of computational accuracy. They cannot be generalized to all other computations. For example, if a measurement equals 15923.648 and the predicted value equals 15923.635, the residual, 0.012681742, may have only 3 or 4 correct digits.

(c) On the right, the coefficients, their standard deviations and the ratio of the coefficients to the standard deviations are printed from a refit omitting the last term (vector). Again, the residual standard deviation and degrees of freedom are printed at the bottom. These results can be usefully compared with the results on the left; particularly in a POLYFIT. In a FIT, these results may or may not be useful, depending upon the order in which the vectors appear in the instruction and the importance of the last term. If only one vector is fitted (or POLYFIT of degree 0), only the residual standard deviation and degrees of freedom are printed.

Automatic Storage.

There are six different forms of each instruction. They differ with respect to the amount of information stored in the worksheet, which depends on the number of arguments in the instruction as follows:

Form	Arguments*	No. of Rows**	Storage
1	4	0	None
2	5	2k+6	C = Coefficients
3	6	n=NRMAX	C, R = Residuals
4	7	n=NRMAX	C, R, S = Standard deviations of predicted values
5	8	2k+2	C, R, S, F = Fourier coefficients
6	10	(k+3)x(k)	C, R, S, F, V = Variance-covariance matrix

* Add (k-1) for FIT instruction.
** In last named column (or matrix).

Note, (2k+6) and (2k+2) may exceed NRMAX. If so, NRMAX will have to be reset in order to print the entire contents of the column(s). If (2k+6) or (2k+2) exceeds the number of rows

in the worksheet, the spill will be lost. Only as many values as will fit in the column are stored and no diagnostic is printed.

The user must specify unique arguments for storing the required information. In

POLYFIT 1, 1.0, 3, 2 put in 11, 12, 12 and 13

the Coefficients would be put in column 11 and the Fourier Coefficients in column 13.

The Coefficient column contains:

Rows	Description	
1 to k	coefficients	
k+1 to 2k	standard deviations of coefficients	
2k+1 to 2k+6	six values:	n = number of non-zero weights
		k = number of vectors (degree+1 in POLYFIT)
		residual degrees of freedom
		residual standard deviation
		residual variance
		multiple correlation coefficient squared.

The residual variance is the last number under CUM. RESIDUAL MS on Page Three of the automatic printing. All the remaining values, except the squared multiple correlation coefficient, appear on Page Four of the automatic printing.

The squared multiple correlation coefficient is

$$R^2 = 1 - \text{Residual Sum of Squares}/\sum_{i=1}^{n} w_i (y_i - \bar{y}_w)^2, \text{ where } \bar{y}_w = \sum w_i y_i / \sum w_i.$$

The denominator in the fraction is the corrected Total Sum of Squares. If the first vector is a vector of 1's, the corrected Total Sum of Squares is the uncorrected Total Sum of Squares minus the sum of squares due to fitting the mean (constant term), i.e., the last number minus the first number under SS = RED. DUE TO COEFF. on Page Three of the automatic printing. If the first vector is not identically equal to 1, the R^2 has no meaning.

The Fourier coefficient column contains the k squared Fourier Coefficients, the Residual Sum of Squares, the Total Sum of Squares, and the k Fourier coefficients. The first (k+2) values are the same as those under SS = RED. DUE TO COEFF on Page Three of the automatic printing. The Fourier coefficients (see Computing Method below) are the estimates of the parameters after orthonormalization.

In the sixth form, the last two arguments give the row and column location of the number in the upper left-hand corner of a (k+3)x(k) matrix. The top (kxk) portion of the matrix is the (complete) variance-covariance matrix shown on Page Two of the automatic printing. The last three rows contain the Gram factors, the vector norms and the Gram determinants, which do not appear in the automatic printing.

The Gram factors are

$$G_1 = (x_1'Wx_1)^{\frac{1}{2}} = \left(\sum_{i=1}^{n} w_i x_i^2\right)^{\frac{1}{2}}$$

$$G_2 = (x_2'Wx_2 - (x_2'W\Phi_1)^2)^{\frac{1}{2}}$$

$$\vdots$$

$$G_k = (x_k'Wx_k - (x_k'W\Phi_1)^2 - (x_k'W\Phi_2)^2 - \ldots - (x_k'W\Phi_{k-1})^2)^{\frac{1}{2}}$$

where x_i is the ith vector of X and ϕ_i is the ith vector of $\Phi = X(T')^{-1}$ and X'WX = TT'.

The vector norms are the square roots of the diagonal terms of the matrix of normal equations,

$$(x_i'Wx_i)^{\frac{1}{2}},\ i = 1,2,\ldots,k$$

The Gram determinants are the partial products

$$\Pi_{i=1}^{r} G_i^2/(x_i'Wx_i),$$

where r = 1, 2, ... ,k. If the x vectors are orthogonal, the kth Gram determinant equals 1. If the X'WX matrix is singular, the kth Gram determinant equals zero. The square root of a quotient in the Gram determinants is the Gram factor divided by the vector norm.

If (R+k+3) exceeds the number of rows in the worksheet, or if (C+k) exceeds the number of columns in the worksheet, the following informative diagnostic is given

* PARTIAL STORAGE OF MATRIX

Computing Method.

The algorithm used for the solution of linear systems of equations involves a Gram-Schmidt orthonormaliztion of the predictor variables as described by Davis (1962) and Walsh (1962). The matrix X'WX of products and cross products is factored into the product TT', where T is a lower triangular matrix. A set of orthonormal vectors is then formed by computing $\Phi = XA$, where A is the inverse of T'. The matrix product ΦWy produces the Fourier coefficients (see Automatic Storage) used in a number of the calculations. Evidence indicates the algorithm is comparatively accurate; see Longley (1967), Wampler (1969) and Wampler (1970). Two slight modifications have been made to improve the accuracy. The sums of products and cross products are performed using double precision arithmetic. A constant, the mid-range of the measurements (with non-zero weights) is subtracted from all the measurements before computations begin and is added back when the computations are complete.

Remark 1.

Often it is desirable to perform several regressions using the same measurements, but different numbers of vectors. If the FIT instruction is stored, there is a question as to how to INCREMENT the instruction. It can only be done if triple asterisks are used. (See section B1.8.) The following three instructions would perform a FIT using 5, 4, 3, 2 and finally 1 vector.

```
1/ FIT                    1, 1.0,  5 in cols 11 *** 15
2/ INCREMENT instr 1 by 0, 0.0, -1,   and   0 *** -1
     PERFORM instrs 1 thru 2, 5 times
```

Remark 2.

The procedures described in this section pertain to linear least squares estimation. Algorithms for non-linear least squares estimation are more difficult and varied. One procedure which is relatively easy and sometimes works very satisfactorily is the Gauss-Newton (or Taylor series linearization) method described in section 10.3 of Draper and Smith (1968). The FIT (or rather SFIT) instruction can be used effectively in each iteration of this method.

Remark 3.

Considerable effort has been expended to develop the FIT (POLYFIT) instruction to print useful results in a readable form. This is not done without some cost to the user, both in

computing time and printing charges. We believe incorrect and/or misleading conclusions based upon an incomplete analysis of data are far more costly than the few additional cents it takes to print more detailed and easily interpreted results. Users who are particularly concerned about costs, for one reason or another, always have the option of using one of the forms of SFIT (SPOLYFIT).

Remark 4.

Joan R. Rosenblatt has used a FIT instruction effectively for cubic spline fitting with fixed knots. See De Boor and Rice (1968) for further details on cubic spline approximations. Three constraints are imposed: at each knot (1) the fitted cubics should agree, (2) the first derivatives should agree, and (3) the second derivatives should agree. Let θ_i be the knots, i = 1,2, ... ,r, and let

$$(x-\theta_i)_+^3 = (x-\theta_i)^3, \text{ if } x > \theta_i, \text{ and}$$

$$= 0, \qquad \text{if } x \leq \theta_i.$$

Then, the model used for a least squares fit is

$$E(y) = a_o + a_1x + a_2x^2 + a_3x^3 + b_1(x-\theta_1)_+^3 + b_2(x-\theta_2)_+^3 + \ldots + b_r(x-\theta_r)_+^3.$$

If the third constraint is relaxed, the following terms should be added to the model

$$c_1(x-\theta_1)_+^2 + c_2(x-\theta_2)_+^2 + \ldots + c_r(x-\theta_r)_+^2.$$

```
: FIT y in col (C), weights (E), (k) variables in columns (C), (C) ... (C)  :
```

Provides "four" page automatic printing, but no automatic storage. The instruction has (k+3) arguments.

```
/ FIT y in (C), weights (E), (k) variables in cols (C) ... (C), put coeffs in (C) /
```

Same as preceding form, but, in addition, the coefficients and standard deviations of the coefficients are stored in the column designated by the last argument. 2(k+3) values are stored. This form of the instruction has (k+4) arguments. If the FIT instruction in the example on page 139 is replaced by the instruction

```
FIT 1, 1.0, 2, 2, 3, 31
```

then column 31 would contain the 10 numbers 65.100002, 1.3800000, 5.8283787, .10263203, 5.0000000, 2.0000000, 3.0000000, 3.2455097, 10.533333 and .98367770.

```
/ FIT (C) wts (E) to (k) in (C)...(C), put coeffs in (C), residuals in (C)  /
```

Same as preceding form, but, in addition, the residuals are stored in the column designated by the last argument. NRMAX values are stored. This form of the instruction has (k+5) arguments.

```
/ FIT (C), (E), (k), (C)...(C), put coeffs in (C), res. in (C), sd of pv in (C) /
```

Same as preceding form, but, in addition, the standard deviations of the predicted values are stored in the column designated by the last argument. NRMAX values are stored. This form of the instruction has (k+6) arguments.

```
/------------------------------------------------------------------------/
/ FIT (C), (E), (k), (C)...(C), put in (C), (C) and (C), Fourier coeffs in (C) /
/------------------------------------------------------------------------/
```

Same as preceding form, but, in addition, the Fourier coefficients are stored in the column designated by the last argument. 2(k+1) values are stored. This form of the instruction has (k+7) arguments. If the FIT instruction in the example on page 139 is replaced by the instruction

FIT 1, 1.0, 2, 2, 3, 31, 32, 33, 34

then column 34 would contain the six values 9.9404994+04, 1.9043999+03, 31.599998, 1.0134100+05, 3.1528558+02 and 43.639431.

```
/------------------------------------------------------------------------/
/ FIT (C), (E), (k), (C)...(C) put in (C),(C),(C),(C) vc matrix in (R),(C) /
/------------------------------------------------------------------------/
```

Same as preceding form, but, in addition, the variance-covariance matrix, the Gram factors, vector norms and the Gram determinants are stored in a (k+3)x(k) matrix starting in row (R) of column (C), designated by the last two arguments. This form of the instruction has (k+9) arguments. If the FIT instruction in the example on page 139 is replaced by the instruction

FIT 1, 1.0, 2, 2, 3, 31, 32, 33, 34 and 1,41

then the variance-covariance matrix

```
33.969998      -.57933331
 -.57933331     .010533333
```

would be put in rows 1 and 2 of column 41 and 42, the Gram factors 2.2360680 and 31.622776 would be row 3 of columns 41 and 42, the vector norms 2.2360680 and 1.2698425+2 would be in row 4 of columns 41 and 42, and the Gram determinants 1.0000000 and .062015503 would be in row 5 of columns 41 and 42.

*****

```
:------------------------------------------------------------------------:
: POLYFIT y in col (C), using weights (E), of degree (d), predictor x in col (C) :
:------------------------------------------------------------------------:
```

Provides "four" page automatic printing of results, but does not provide any storage. This instruction has 4 arguments. The degree of the polynomial, d, may equal zero.

```
/------------------------------------------------------------------------/
/ POLYFIT y in col (C), wts (E), degree (d), x in (C), put coefficients in (C) /
/------------------------------------------------------------------------/
```

Same as preceding form, but, in addition, the coefficients and standard deviations of the coefficients are stored. A total of 2(d+4) values are stored. This form of the instruction has 5 arguments.

```
/------------------------------------------------------------------------/
/ POLYFIT y in (C), wts (E), deg (d), x in (C), put in (C), residuals in (C) /
/------------------------------------------------------------------------/
```

Same as preceding form, but, in addition, the residuals are stored in the column designated by the last argument. NRMAX values are stored. This form of the instruction has 6 arguments.

```
POLYFIT (C), (E), (d), (C), put coeffs in (C), res in (C) sd of pv in (C)
```

Same as preceding form, but, in addition, the standard deviations of the predicted values are stored in the column designated by the last argument. NRMAX values are stored. This form of the instruction has 7 arguments.

```
POLYFIT (C), (E), (d), (C), put in (C), (C), (C) and Fourier coeffs in (C)
```

Same as preceding form, but, in addition, the Fourier coefficients are stored in the column designated by the last argument. 2(d+2) values are stored (if there is sufficient room in the column). This form of the instruction has 8 arguments.

```
POLYFIT (C), (E), (d), (C), put in (C),(C),(C),(C) and vc matrix in (R),(C)
```

Same as preceding form, but, in addition, the variance-covariance matrix, the Gram factors, the vector norms and the Gram determinants are stored as a (d+4)x(d+1) matrix starting in row (R) of column (C), designated by the last two arguments. This form of the instruction has 10 arguments.

*****

```
SFIT y in (C), weights (E), (k) variables in cols (C) ... (C), put coeffs in (C)
```

```
SFIT (C) wts (E) to (k) in (C)...(C), put coeffs in (C), residuals in (C)
```

```
SFIT (C), (E), (k), (C)...(C), put coeff in (C), res. in (C), sd of pv in (C)
```

```
SFIT (C), (E), (k), (C)...(C), put in (C), (C) and (C), Fourier coeffs in (C)
```

```
SFIT (C), (E), (k), (C)...(C) put in (C),(C),(C),(C) vc matrix in (R),(C)
```

*****

```
SPOLYFIT y in col (C), wts (E), degree (d), x in (C), put coefficients in (C)
```

SPOLYFIT y in (C), wts (E), deg (d), x in (C), put in (C), residuals in (C)

SPOLYFIT (C), (E), (d), (C), put coeffs in (C), res in (C) sd of pv in (C)

SPOLYFIT (C), (E), (d), (C), put in (C), (C), (C) and Fourier coeffs in (C)

SPOLYFIT (C), (E), (d), (C), put in (C),(C),(C),(C) and vc matrix in (R),(C)

4.6 Correlation.

CORRELATION, SCORRELATION

Correlation describes the linear, statistical, relationship between two normally distributed variables. Correlation techniques can be used effectively in prediction and model building problems. Unfortunately, correlation coefficients are often used and interpreted incorrectly, because they are (i) used when a relationship is not linear, (ii) used to imply a causal rather than statistical relationship, (iii) used when the variables are not normally distributed, (iv) misconstrued as real when they are spurious, and (v) used blindly when outliers are present or when groups of data are not validly combined. See the references in section 4.8 for more details. To aid the user in the thoughtful use and interpretation of correlation coefficients, a CORRELATION instruction automatically prints three different types of correlation coefficients and four other tables for auxiliary use. An additional technique, which is not part of the instruction, is to plot pairs of variables (scatter diagrams) and also residuals. Liberal use of plots can be very helpful in extracting the relevant information in the data and in avoiding misinterpretations.

There are three different forms of CORRELATION which differ only with respect to the amount of information stored. In each case, the number of variables used (first argument) must be greater than 1 and less than 100. The arguments in each form are all integers and the number of arguments is equal to the first argument plus one, three or five, respectively, in the three forms.

Each form of the instruction provides an automatic printing of seven different tables. Pages 156 and 157 give the automatic printing and the set of instructions for some data taken from page 216 of Draper and Smith (1968). (Note, in their correlation matrix -0.615790 should read -0.615970; 0.615790 should read -0.615970; and 0.769950 should read 0.767950). The title at the top of the first page gives the number of variables used and the number of measurements (NRMAX). The seven tables are described below. Any reader who is in doubt about the meaning of a particular number would be well advised, on first use of the instruction, to calculate quantities directly by other means. Some methods for doing this are sketched in the descriptions which follow. Throughout, n=NRMAX will denote the number of measurements and p (first argument) the number of variables.

If the number of variables, p, is less than or equal to 10, all tables are printed in their entirety. If p exceeds 10, tables are printed in blocks with 10 or less columns in each block.

Unfortunately, it is difficult to give a single reference for all seven tables in the automatic printing. Non-statisticians are urged to consult a statistician for assistance in the interpretation of the results. A reference is given for each table, but a full

CORRELATION ANALYSIS FOR 4 VARIABLES WITH 9 OBSERVATIONS

SIMPLE CORRELATION COEFFICIENTS

COLUMN	11	12	13	14
11	1.0000			
12	.6837	1.0000		
13	-.6160	-.1725	1.0000	
14	.8018	.7680	-.6287	1.0000

SIGNIFICANCE LEVELS OF SIMPLE CORRELATION COEFFICIENTS (ASSUMING NORMALITY)

COLUMN	11	12	13	14
11	.0000			
12	.0423	.0000		
13	.0774	.6572	.0000	
14	.0094	.0157	.0697	.0000

PARTIAL CORRELATION COEFFICIENTS WITH 2 REMAINING VARIABLES FIXED

COLUMN	11	12	13	14
11	1.0000			
12	.4317	1.0000		
13	-.4566	.6972	1.0000	
14	.1054	.7268	-.6478	1.0000

SIGNIFICANCE LEVELS OF PARTIAL CORRELATION COEFFICIENTS (ASSUMING NORMALITY)

COLUMN	11	12	13	14
11	.0000			
12	.3334	.0000		
13	.3030	.0817	.0000	
14	.8221	.0642	.1157	.0000

SPEARMAN RANK CORRELATION COEFFICIENTS (ADJUSTED FOR TIES)

COLUMN	11	12	13	14
11	1.0000			
12	.6109	1.0000		
13	-.5667	-.1255	1.0000	
14	.6833	.6025	-.7167	1.0000

SIGNIFICANCE LEVEL OF QUADRATIC FIT OVER LINEAR FIT BASED ON F RATIO WITH 1 AND 6 DEGREES OF FREEDOM
(FOR EXAMPLE, .1703 IS THE SIGNIFICANCE LEVEL OF THE QUADRATIC TERM WHEN COLUMN 12 IS FITTED TO COLUMN 11)

COLUMN	11	12	13	14
11	1.0000	.4044	.9494	.8522
12	.1703	1.0000	.8099	.9377
13	.7165	.5676	1.0000	.8499
14	.1565	.5997	.3681	1.0000

CONFIDENCE INTERVALS FOR SIMPLE CORRELATION COEFFICIENTS (USING FISHER TRANSFORMATION)
95 PER CENT LIMITS BELOW DIAGONAL, 99 PER CENT LIMITS ABOVE DIAGONAL

COLUMN	11	12	13	14
11	99.0000	.9552	.3213	.9735
	95.0000	-.2122	-.9436	.0519
12	.9269	99.0000	.7051	.9685
	.0359	95.0000	-.8414	-.0362
13	.0815	.5552	99.0000	.3025
	-.9085	-.7506	95.0000	-.9459
14	.9565	.9484	.0607	99.0000
	.2944	.2119	-.9120	95.0000

LIST OF COMMANDS, DATA AND DIAGNOSTICS

```
READ DATA INTO COLUMNS 11, 12, 13 AND 14
42.2   11.2   31.9   167.1
48.6   10.6   13.2   174.4
42.6   10.6   28.7   160.8
39.0   10.4   26.1   162.0
34.7    9.3   30.1   140.8
44.5   10.8    8.5   174.6
39.1   10.7   24.3   163.7
40.1   10.0   18.6   174.5
45.9   12.0   20.4   185.7
CORRELATION WITH 4 VARIABLES IN COLUMNS 11, 12, 13 AND 14
```

understanding is best achieved by consulting several references. Kendall and Stuart (1961) is perhaps the best single reference.

(1) Simple Correlation Coefficient.

The simple (product moment) correlation coefficient for two variables x and y is given by

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}}$$

where $\bar{x}$ is the average of the x measurements and $\bar{y}$ the average of the y measurements. See, for example, section 1.6 (page 33) of Draper and Smith (1968) or Chapter 7 of Snedecor and Cochran (1967). Clearly, the correlation between any variable and itself is 1.0. Hence, the upper left to lower right diagonal entries in the table are always equal to 1.0. Also, it is clear that $r_{xy} = r_{yx}$, so only the lower half of the table is printed. The average of column 12 is 10.622222 and the average of column 13 is 22.422222. The sums of squared deviations about the averages are shown in the calculation of the correlation coefficient below:

$$r_{xy} = \frac{-8.1644448}{\sqrt{(4.4555554)(502.81554)}}$$

$$= \frac{-8.1644448}{47.332045}$$

$$= -.17249296$$

A single correlation coefficient can be obtained by using a FIT instruction and the formula

$$r_{xy} = \frac{\hat{\beta}}{\sqrt{\hat{\beta}^2 + (n-2)s_{\hat{\beta}}^2}}$$

For the above example

$$r_{xy} = \frac{-.016237455}{\sqrt{(-.016237455)^2 + 7(.035046006)^2}}$$

$$= -0.17249296$$

(2) Significance levels of simple correlation coefficients.

For each pair of variables (x and y), the F-statistic

$$F_0(1, n-2) = (n-2)r_{xy}^2 / (1 - r_{xy}^2)$$

with 1 and (n-2) degrees of freedom is computed. Here we are making use of the fact that F = t squared. From which the significance level

$$S = Pr(F \text{ exceeds } F_0)$$

is computed.

If the "true" correlation coefficient is equal to zero, then S is the probability that in a random sample (of the same size) the absolute value of a sample correlation coefficient will exceed the absolute value of the observed correlation coefficient. For columns 12 and 13

$$F_0 = 7(0.029753822)/(1-0.029753822) = 0.21466382 \quad \text{and} \quad S = 0.6572.$$

The value of S can be easily verified by using the appropriate F PROBABILITY instruction (see section 4.7). Small values (less than .05 for example) of S indicate that the correlation coefficient is significantly different from zero. In this case, S is very large and there is a lack of evidence that columns 12 and 13 are correlated. However, see the next two descriptions for contrary evidence.

(3) Partial Correlation Coefficients.

When more than two variables are under study, the simple correlation coefficient can be seriously distorted by the effect of other variables. The partial correlation coefficent may overcome this difficulty by measuring the correlation between two variables after eliminating the effect of the remaining variables under study (remaining variables held constant). The user should compare the simple correlation coefficients with the partial correlation coefficients. Any "large" discrepancy indicates that one or more of the remaining variables is having an important effect on the relationship. See Kendall and Stuart (1961), section 27.5, page 318.

Let $R = (r_{ij})$ be the matrix of simple correlations coefficients and let $C = R^{-1} = (r^{ij})$ be the inverse of R. Then, the partial correlation between any pair of variables, i and j, with the remaining variables fixed (held constant) is

$$r_{ij.} = \frac{-r^{ij}}{\sqrt{r^{ii}r^{jj}}} .$$

By using MINVERT (section 8.5), it can be verified that the partial correlation between columns 12 and 13 is

$$r_{ij.} = \frac{2.8514011}{\sqrt{(4.8973892)(3.4156598)}} = 0.69717001.$$

Note carefully that the partial correlation coefficient is 0.6972, whereas the simple correlation coefficient is -0.1725. The two coefficients not only differ in magnitude, but also in sign. Thus, the effect of column 11 and/or column 14 is distorting the value of the simple correlation coefficient. This can be seen further by examining the remaining coefficients.

(4) Significance Levels of Partial Correlation Coefficients.

For each pair of variables, the following F-statistic is computed from the partial correlation coefficient $r_{xy.}$:

$$F_0(1,n-p) = (n-p)r^2_{xy.}/(1 - r^2_{xy.})$$

with 1 and (n-p) degrees of freedom. From which the significance level, S, is computed from

$$S = Pr(F \text{ exceeds } F_0)$$

Under the hypothesis that the "true" partial correlation coefficient is zero, S is the probability that the absolute value of a partial correlation coefficient will exceed the absolute value of the observed partial correlation coefficient in a random sample (of the same size). See section 27.22 on page 333 of Kendall and Stuart (1961). For columns 12 and 13,

$$F_0 = (5)(0.48604604)/(1-0.48604604) = 4.7284977 \text{ and } S = 0.0817$$

Compare this value of S with the value 0.6572 in (2) above.

(5) Spearman Rank Correlation Coefficient.

A rank correlation coefficient is useful when variables can not be measured quantitatively, but only their ranks can be observed. The rank correlation coefficient can also be used to avoid the assumption of a bivariate normal distribution. A comparison should be made between the rank correlation coefficients and the corresponding simple and partial correlation coefficients. Again, a "large" discrepancy between two comparable coefficients is an indicator of some abnormality in the data. See Kendall (1948) for further details.

The Spearman rank correlation coefficient for any pair of variables x and y is computed from:

$$\rho_{xy} = \frac{A - D^2_{xy} - T_x - T_y}{\sqrt{(A - 2T_x)(A - 2T_y)}},$$

where

$$D^2_{xy} = \sum_{i=1}^{n} [\text{rank}(x_i) - \text{rank}(y_i)]^2,$$

$$A = (n-1)(n)(n+1)/6,$$

$$T_x = (1/12)\sum_x (t_x - 1)(t_x)(t_x + 1),$$

$$T_y = (1/12)\sum_y (t_y - 1)(t_y)(t_y + 1),$$

t_x = number of ties in a set of tied x's,

and

t_y = number of ties in a set of tied y's.

The quantities T_x and T_y are used to make adjustments for ties in the ranks; see section 4.1. If there are no ties T_x and T_y equal zero and

$$\rho_{xy} = 1 - D^2/A.$$

For columns 12 and 13, it can be easily verified using a RANKS instruction (see section 4.1) that the rank correlation coefficient equals

$$\rho_{xy} = \frac{120 - 134.5 - 0.5 - 0.0}{\sqrt{(120-1)(120-0)}}$$

$$= -15/119/119.49895 = -0.12552421,$$

which does not differ greatly from the product moment correlation coefficient -0.1725.

(6) Significance Level of Quadratic Fit over Linear Fit.

Underlying the use of a correlation coefficient is the assumption that the two variables are linearly related. The results in this table are useful in assessing the validity of this assumption of linearity. The variables are all assumed to be normally distributed. The numbers printed are the significance levels for a F-test of the hypothesis that the quadratic term in a quadratic model is zero. The F-statistic used is

$$F_o(1,n-3) = [\text{Residual sum of squares(linear model) - Residual SS (quadratic model)}]/\text{Residual variance(quadratic model)}$$

with 1 and (n-3) degrees of freedom. The values of 1 and (n-3), 1 and 6 in our example, are printed in the title. The significance level, S, is then computed from

$$S = Pr(F \text{ exceeds } F_o)$$

Small values of the significance level (less than .05, for example) indicate lack of linearity. The test results differ depending upon which variable of a pair is considered the dependent variable and which one is considered the independent (or predictor) variable. Hence, the entire table is printed, rather than just the lower half. The diagonal entries are always equal to one and have no particular relevance. Tests of hypotheses in linear regression are discussed in section 13.8 on page 441 of Brownlee (1965).

For a regression of column 12 on column 13, F = (4.3229856-4.2779137)/0.71298562 = 0.063215720 and S = 0.8099. The residual sums of squares can be obtained from using POLYFIT and the value of S can be verified using F PROBABILITY (see section 4.7). Actually, the values of F and S are the same as the last numbers in the columns headed F(COEF=0) and P(F) in the analysis of variance of the automatic printing of a POLYFIT of degree 2 of column 12 on column 13.

(7) Confidence Intervals For Simple Correlation Coefficients.

Both 95% and 99% confidence intervals for the simple correlation coefficients are printed in a two-way table, using four decimal places, with two entries in each cell of the table. The .95 and .99 confidence coefficients are printed along the upper left to lower right diagonal. The 95% confidence limits are printed below the diagonal and the 99% confidence limits are printed above the diagonal. The number in the lower left of each cell is the lower confidence limit and the number in the upper right is the upper confidence limit.

The confidence intervals are based on a normal approximation and are computed as follows:

$$\text{Lower confidence limit:} \quad \tanh[z - u/\sqrt{(n-3)}]$$

$$\text{Upper confidence limit:} \quad \tanh[z + u/\sqrt{(n-3)}],$$

where

$$z = \tanh^{-1}(r)$$
$$= \tfrac{1}{2}\log[(1+r)/(1-r)], \quad \text{(log to the base e)}$$

and

$$u = 1.9599640, \text{ for 95\% confidence interval}$$
$$= 2.5758295, \text{ for 99\% confidence interval.}$$

See Chapter 3, page 101, of Morrison (1967). For example, for the 95% upper confidence

limit for the correlation between columns 12 and 13, $z = \tanh^{-1}(-0.17249296) = -0.17423494$ and the upper confidence limit equals $\tanh(-0.17423494+1.959640/\sqrt{6}) = \tanh(0.6259176) = 0.5552$. Also, the lower 99% confidence limit equals $\tanh(-0.17423494-2.5758293/\sqrt{6}) = \tanh(-1.2258128) = -0.8414$. We are 95% confident that the "true" correlation coefficient lies between -0.7506 and +0.5552 and we are 99% confident that the "true" correlation coefficient lies between -0.8414 and +0.7051. (The confidence intervals are wide because the sample size (9) is small.)

CORRELATION between (p) variables in columns (C), (C) ... (C)

Provides the automatic printing described above with no storage of results.

CORRELATION (p) var's in (C), (C) ... (C), put array of simple coeffs in (R),(C)

Same as above, but, in addition, provides for the storage of the simple correlation coefficients as an array starting in row (R) of column (C). The instruction

CORRELATION with 4 variables in cols 11 *** 14, put coeffs in 1,41

would put the simple correlation coefficients in the worksheet as follows:

Row/Column	41	42	43	44
1	1.0000000	.68374211	-.61596990	.80175223
2	.68374211	1.0000000	-.17249296	.76795027
3	-.61596990	-.17249296	1.0000000	-.62874596
4	.80175223	.76795027	-.62874596	1.0000000

This form of the instruction is useful when one wants to perform additional calculations with the correlation coefficients or when more significant digits are required in the answers.

CORRELATION for (p) in (C) ... (C), put r coeffs in (R),(C), rho in (R),(C)

Same as above, but, in addition, the partial correlation coefficients are stored as an array starting in row (R) of column (C). The instruction

CORRELATION with 4 variables in columns 11 *** 14 store results in 1,41 and 6,41

would put the following partial correlation coefficients in the worksheet:

Row/Column	41	42	43	44
6	1.0000000	.43170931	-.45663585	.10539049
7	.43170931	1.0000000	.69717003	.72682013
8	-.45663585	.69717003	1.0000000	-.64778929
9	.10539049	.72682013	-.64778929	1.0000000

This form of the instruction is useful when one wants to perform further calculations using the partial correlation coefficients or when more significant digits are needed.

*****

```
SCORRELATION (p) var's in (C), (C)...(C), put array of simple coeffs in (R),(C)

SCORRELATION for (p) in (C) ... (C), put r coeffs in (R),(C), rho in (R),(C)
```

4.7 Probability.

F PROBABILITY, UNIFORM RANDOM

Both of the instructions in this section use two word commands.

```
F PROBABILITY with (E) and (E) degrees of freedom for (E) put in column (C)
```

Computes the right-tail area of an F-distribution with the specified number of degrees of freedom. Let the third argument be represented by F_0. The probability that a random variable, which follows the F-distribution with the specified degrees of freedom, exceeds F_0 is put in the column designated by the fourth argument. In the notation of AMS 55 (Abramowitz and Stegun (1964)), the instruction computes $Q(F_0)$. The instruction

F PROBABILITY with 3.0 and 5.0 degrees of freedom of 5.4095 put in column 46

would put the number 0.050 into column 46.

Numbers specified by the first two arguments should be integers. (If either argument is not a column number, it should be written with a decimal point.) If numbers are not integers, the following informative diagnostic is printed:

* NU1 OR NU2 TRUNCATED TO INTEGER

If any value specified by the 1st two arguments is less than 1.0, the value 1.0 is used and the following informative diagnostic is printed:

* NU1 OR NU2 LESS THAN 1

Any number specified by the third argument must be greater than or equal to zero. Otherwise, the following informative diagnostic is printed:

* F LESS THAN 0, SET=0

The probability integral is obtained by computing a finite series in double precision using the formulas on page 946 of Abramowitz and Stegun (1964).

```
UNIFORM RANDOM numbers starting with (K) put in column (C)
```

The instruction produces numbers pseudo randomly distributed between 0 and 1 in every row down to NRMAX. If the number (K) is equal to or greater than 1, the integral part of (K) is used as the starting value by the random number generator. If the value (K) is less than 1.0, it is assumed to be a random number and the corresponding starting integer is computed by the instruction. The value of (K) can be 1 to start with; or a random integer between 1.0 and 8192.0 can be chosen. All numbers greater than 8192. are reduced modulo 8192.

To use the instruction sequentially one can use the last random number generated as the starting value for the next sequence. For example, if NRMAX=50 and random numbers have been put into column 21, then the instruction

UNIFORM RANDOM numbers starting with *50,21* put in column 22

could be used to generate a new sequence of random numbers in column 22. The argument *50,21* designates the random number in row 50 of column 21.

The algorithm used is an adaption of the one given by Kruskal (1969). This particular generator was chosen because it is extremely portable. However, the portability is achieved at the expense of being optimum. The random number generator is not the most efficient nor of the highest quality. It should be used experimentally only and with caution. For this purpose it should be adequate. Because the generator is not optimum, the following informative diagnostic is always printed when the instruction is used:

* CAUTION, USE EXPERIMENTALLY ONLY. NOT OPTIMUM IN ORDER TO MAKE IT MACHINE INDPENDENT. REFERENCE - J.B. KRUSKAL, ACM, 12, 92. AND J. H. HALTON, SIAM REV., 12, 1.

4.8 References For Section 4.

Listed below are references to books and articles which discuss the statistical and computing techniques and concepts used by the OMNITAB instructions for statistical analysis. No attempt is made to be complete.

ABRAMOWITZ, MILTON and STEGUN, IRENE (1964). Handbook of Mathematical Functions, NBS Applied Mathematics Series 55, Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

ANSCOMBE, F. J. and TUKEY, J. W. (1963). The examination and analysis of residuals. Technometrics, 5, 141-160.

BRADLEY, J. V. (1968). Distribution-free Statistical Tests. Prentice-Hall.

BROWNLEE, K. A. (1965). Statistical Theory and Methodology in Science and Technology, 2nd Edition, John Wiley and Sons, Inc.

CROW, E. L. and SIDDIQUI, M. M. (1967). Robust estimation of location. J. Amer. Statist. Assoc., 62, 353-389.

DAVIS, P. J. (1962). Orthonormalization codes in numerical analysis. Survey of Numerical Analsis, Ed. J. Todd, McGraw-Hill. Chapter 10, 347-379.

DE BOOR, C. and RICE, J. R. (1968). Least squares cubic spline approximation I - fixed knots. Report CSD TR 20, Computer Sciences Department, Purdue University, Lafayette, Indiana

DIXON, W. J. and MASSEY, F. J., Jr. (1957). Introduction To Statistical Analysis. Second Edition. McGraw-Hill Book Company.

DRAPER, N. R. and SMITH, H. (1968). Applied Regression Analysis. John Wiley and Sons, Inc.

DUNCAN, A. J. (1965). Quality Control and Industrial Statistics, 3rd Ed., Richard D. Irwin.

EISENHART, C. (1947). Significance of the largest of a set of sample estimates of variance. Techniques of Statistical Analysis. Ed. by C. Eisenhart, M. W. Hastay and W. A. Wallis, Statistical Research Group, Columbia University. McGraw-Hill Book Company. pp 375-382.

FISHER, R. A. (1950). Statistical Methods for Research Workers, 11th Ed., Hafner Publishing Company.

FREUND, J. E. and WILLIAMS, F. J. (1958). Modern Business Statistics. Prentice-Hall.

GRAYBILL, F. A. (1961). An Introduction To Linear Statistical Models, Vol. 1. John Wiley & Sons, Inc.

HOGBEN, DAVID (1968). An algorithm for computing 95% confidence intervals for the mean and standard deviation. Unpublished.

HOGBEN, DAVID (1969). Selected references. Precision Measurement and Calibration, Statistical Concepts and Procedures. National Bureau of Standards Special Publication 300, Vol 1, H. H. Ku, Ed., 402-407. U.S. Government Printing Office.

KEMPTHORNE, O. (1952). The Design and Analysis of Experiments. John Wiley & Sons, Inc.

KENDALL, M. G. (1948). Rank Correlation Methods. Charles Griffin & Co.

KENDALL, M. G. and STUART, A. (1961). The Advanced Theory of Statistics, Vol. 2 Hafner Publishing Company, New York.

KRUSKAL, J. B. (1969). Extremely portable random number generator. Comm. ACM, 12, 93-94.

LONGLEY, J. W. (1967). An appraisal of least squares programs for the electronic computer from the point of view of the user. J. Amer. Statist. Assoc., 62, 819-841

MANDEL, JOHN (1964). The Statistical Analysis Of Experimental Data. Interscience.

MORRISON, D. F. (1967). Multivariate Statistical Methods. McGraw Hill Book Company.

NATRELLA, M. G. (1963). Experimental Statistics. National Bureau of Standards Handbook 91. U.S. Government Printing Office.

OWEN, D. B. (1962). Handbook of Statistical Tables, Addison-Wesley Publishing Co.

SNEDECOR, G. W. (1956). Statistical Methods, 5th Ed., Iowa State University Press.

SNEDECOR, G. W. and COCHRAN, W. G. (1967). Statistical Methods. Sixth Edition. Iowa State University Press.

WALSH, P. J. (1962). Algorithm 127: ORTHO. Comm. Assoc. Comp. Mach., 5, 511-513. See also: Barrodale, Ian (1970). Certification of Algorithm 127, ORTHO. Comm. Assoc. Comp. Mach., 13, 122.

WAMPLER, R. H. (1969). An evaluation of linear least squares computer programs. J. of Res., B, NBS, 73B, 59-90.

WAMPLER, R. H. (1970). A report on the accuracy of some widely used least squares computer programs. J. Amer. Statist. Assoc., 65, 549-565.

ZELEN, M. (1962). Linear estimation and related topics. Survey of Numerical Analysis, Ed. J. Todd, McGraw-Hill. Chapter 17, 558-584.

5. NUMERICAL ANALYSIS.

5.1 Special Integrals.

CERF, ELLIPTICAL FIRST, ELLIPTICAL SECOND, ERROR, STRUVE ONE, STRUVE ZERO

A full description of the complete elliptical integrals and Struve functions may be found in Abramowitz, M., and Stegun, I. A., "Handbook of Mathematical Functions", National Bureau of Standards Applied Mathematics Series 55, Superintendent of Documents, U. S. Government Printing Office, Washington, D. C. 20402. A full description of the computational methods used to calculate the error and complementary error function may be found in Stegun, I. A. and Zucker, R. "Automatic computing methods for special functions," J. of Res. NBS-B (Math. Sciences), 74B, 211-224 (1970).

CERF function of (E) put in column (C)

Computes values of the complementary error function

$$\text{erfc}(x) = 1 - \text{erf}(x), \text{ where}$$

$$\text{erf}(x) = (2/\sqrt{\pi})\int_0^x e^{-t^2}dt.$$

The range covered is limited by the capabilities of the computer. For the UNIVAC 1108, NBS computer, computations are performed for $|x| \leq 26.5$. For $|x| > 26.5$, erfc(x) =0. For the values .4, .8, 1.2 and 1.6 in column 1, the instruction

CERF of column 1 put in column 2

would put the numbers .57160765, .25789904, .089686025 and .023651617 into column 4.

ELLIPTICAL FIRST integral of (E) put in column (C)

ELLIPTICAL FIRST evaluates the complete elliptical integral of the first kind

$$K(x) = \int_0^{\pi/2} (1-x\sin^2\theta)^{-\frac{1}{2}}d\theta$$

for all positive values of the parameter x less than 1.0 (x=k , modulus k). If the value of x is greater than or equal to 1.0, the result is set equal to zero and the following arithmetic diagnostic is given:

** X FOR ELLIPTICAL INTEGRALS IS =1.0 OR GREATER. RESULT IS SET TO 0.0 (n) TIMES.

Let column 23 contain the following values .1, .3 and .5, then the instruction

ELLIPTICAL FIRST integral of col 23 and put results in col 3

will put the values 1.6124413, 1.7138894 and 1.8540747 in column 3.

ELLIPTICAL SECOND integral of (E) put in column (C)

ELLIPTICAL SECOND evaluates the complete integral of the second kind

$$E(x) = \int_0^{\pi/2} (1-x\sin^2\theta)d\theta$$

for all positive values of the parameter x less than or equal to 1.0 (x=k , modulus k). For x greater than 1.0, the result is set equal to zero, and the following arithmetic diagnostic is given:

** X FOR ELLIPTICAL INTEGRALS IS =1.0 OR GREATER. THE RESULT IS SET TO 0.0 (n) TIMES.

The instruction

ELLIPTICAL SECOND integral of column 23 and put results in col 4

will store 1.5307576, 1.4453631 and 1.3506439 in column 4 assuming column 23 contains the values .1, .3 and .5 (used above).

ERROR function of (E) put in column (C)

Computes values of the error function

$$\text{erf}(x) = (2/\sqrt{\pi})\int_0^x e^{-t^2}dt.$$

The range covered is limited by the capability of the computer used. For the NBS computer, UNIVAC 1108, the computations are performed for $|x| \leq 26.5$. For $|x| > 26.5$, $\text{erf}|x| = 1.0$. For the numbers .4, .8, 1.2 and 1.6 in column 1, the instruction

ERROR function of the values in column 1 put in column 2

would put the numbers .42839235, .74210096, .91031397 and .97634839 into column 2.

Remark.

Values of the normal probability integral

$$\text{Gau}(x) = (1/\sqrt{2\pi})\int_{-\infty}^x e^{-\frac{1}{2}t^2}dt$$

can be obtained from values of the error function using the relation

$$\text{Gau}(x) = \tfrac{1}{2}[1 + \text{erf}(x/\sqrt{2})].$$

STRUVE ONE integral of (E) put in column (C)

STRUVE ONE evaluates for real, positive values of x, the Struve function

$$H_1(x) = (2/\pi)[x^2/(1^2.3) - x^4/(1^2.3^2.5) + x^6/(1^2.3^2.5^2.7) - \ldots].$$

If column 1 contained the values 0, 2.5 and 5, then the instruction

STRUVE ONE integral of column 1 put results in col 2

will put the values 0.0, .8631542 and .80781195 in column 2.

STRUVE ZERO integral of (E) put in column (C)

STRUVE ZERO evaluates, for real, positive values of x, the Struve function

$$H_o(x) = (2/\pi)[x - x^3/(1^2.3^2) + x^5/(1^2.3^2.5^2) - \ldots].$$

If NRMAX = 2, the instruction

STRUVE ZERO integral of 4.5 and put results in col 3

will put the value -.058543316 into rows 1 and 2 of column 3.

5.2 Polynomials.

HERMITE, LAGUERRE, LEGENDRE, NORMLAGUERRE, TCHEBYSHEV, UCHEBYSHEV

Each of these instructions has exactly 3 arguments. The 1st argument is an integer (without a decimal point), greater than zero, indicating the order of the polynomial. (All polynomials of order zero are identically equal to one and are not calculated.) The 2nd and 3rd arguments specify column numbers. The results will be put in the column specified by the third argument and in successive columns. Assume C3 represents the third argument. Then, the polynomial results of order one will be put in column C3, order two in column C3+1, and order n in C3+n-1 column. If there are not enough successive columns to put the results of all the orders requested, the following fatal error message is printed:

*** COLUMN NUMBER TOO BIG OR LESS THAN 1

All the polynomials are computed using the appropriate recursion formulas. The equations for these recursion formulas may be found in Abramowitz, M., and Stegun, I. A., "Handbook of Mathematical Functions", National Bureau of Standards Applied Mathematics Series 55, Superintendent of Documents, U. S. Government Printing Office, Washington, D. C. 20402.

The examples below assume that column 11 contains .5, 1.0, 1.5 and 2.0.

HERMITE polynomial of order (n) of col (C) put in col (C) and succ. cols

HERMITE evaluates the polynomial

$$H_n(x) = n!\sum_{m=o}^{N}(-1)^m(2x)^{n-2m}/(m!(n-2m)!),$$

where N = [n/2]. The instruction

HERMITE polynomial of order 4 of x in col 11 put in col 2 and succ. cols

will put the following values in columns 2, 3, 4 and 5.

	$H_1(x)$	$H_2(x)$	$H_3(x)$	$H_4(x)$
Row/Column	2	3	4	5
1	1.0	-1.0	-5.0	1.0
2	2.0	2.0	-4.0	-20.0
3	3.0	7.0	9.0	-15.0
4	4.0	14.0	40.0	76.0

LAGUERRE	polynomial of order (n) of col (C) put in col (C) and succ. cols

LAGUERRE evaluates the polynomial

$$L_n(x) = \sum_{m=o}^{n} (-1)^m K x^m / m!,$$

where $K = \binom{n}{m}$. The instruction

LAGUERRE polynomial of order 3 of col 11 put in col 7 and succ. cols

will put the following results in columns 7, 8 and 9:

	$L_1(x)$	$L_2(x)$	$L_3(x)$
Row/Column	7	8	9
1	.5	.125	-.14583333
2	0.0	-.5	-.66666666
3	-.5	-.875	-.6875
4	-1.0	-1.0	-.33333333

LEGENDRE	polynomial of order (n) of col (C) put in col (C) and succ. cols

This instruction computes the polynomial

$$P_n(x) = (1/2^n) \sum_{m=o}^{N} (-1)^m K x^{n-2m} T,$$

where $N = [n/2]$, $K = \binom{n}{m}$ and $T = \binom{2n-2m}{n}$. The instruction

LEGENDRE polynomial of order 3 of x in col 11 start storing in col 15

will put the following results in columns 15, 16 and 17.

	$P_1(x)$	$P_2(x)$	$P_3(x)$
Row/Column	15	16	17
1	0.5	-.125	-.4375
2	1.0	1.0	1.0
3	1.5	2.875	6.1874999
4	2.0	5.5	17.0

NORMLAGUERRE	polynomial of order (n) of col (C) put in col (C) and succ. cols

NORMLAGUERRE scales the Laguerre polynomial with the factor n!, $NL_n(x) = n!L_n(x)$. Thus, the instruction

NORMLAGUERRE polynomial of order 3, x in col 11 start storing in col 2

will put the following results in columns 2, 3 and 4.

Row/Column	$NL_1(x)$ 2	$NL_2(x)$ 3	$NL_3(x)$ 4
1	.5	.25	-.875
2	0.0	-1.0	-4.0
3	-.5	-1.75	-4.125
4	-1.0	-2.0	-2.0

TCHEBYSHEV polynomial of order (n) of col (C) put in col (C) and succ. cols

TCHEBYSHEV computes the Chebyshev polynomial of the first kind

$$T_n(x) = (n/2)\sum_{m=o}^{N}(-1)^m(2x)^{n-2m}(n-m-1)!/(m!(n-2m)!),$$

where N = [n/2]. The instruction

TCHEBYSHEV polynomial of order 2 x in col 11 start storing in col 50

will store the following values in columns 50 and 51.

Row/Column	$T_1(x)$ 50	$T_2(x)$ 51
1	.5	-.5
2	1.0	1.0
3	1.5	3.5
4	2.0	7.0

UCHEBYSHEV polynomial of order (n) of col (C) put in col (C) and succ. cols

UCHEBYSHEV evaluates the Chebyshev polynomial of the second kind

$$U_n(x) = \sum_{m=o}^{N}(-1)^m(2x)^{n-2m}(n-m)!/(m!(n-2m)!),$$

where N = [n/2]. The instruction

UCHEBYSHEV polynomial of order 4 x in col 11 start storing in col 23

will put $U_1(x)$, $U_2(x)$, $U_3(x)$ and $U_4(x)$ in columns 23, 24, 25 and 26, respectively, as follows:

Row/Column	$U_1(x)$ 23	$U_2(x)$ 24	$U_3(x)$ 25	$U_4(x)$ 26
1	1.0	0.0	-1.0	-1.0
2	2.0	3.0	4.0	5.0
3	3.0	8.0	21.0	55.0
4	4.0	15.0	56.0	209.0

5.3 Iteration.

ISETUP, ISOLATE, ITERATE

These instructions provide the most powerful, general method for: finding approximations to the roots X of the function Y = f(X), finding values of an inverse function, or inverse interpolation. The instruction ISOLATE may be used to find multiple roots for a single function value, whereas ITERATE, in conjuction with ISETUP, may be used to find a single root for each of several function values.

The instructions ISETUP, ISOLATE and ITERATE locate the values of x, in the column designated by the 1st argument, such that the corresponding values of y = f(x), in the column designated by the 2nd argument, bracket the desired Y's in the column designated by the 3rd argument. Here, X denotes an exact root and x an approximation. Hence, the desired Y = f(X) and y = f(x). In ISOLATE, the 3rd argument is a constant, thus ISOLATE locates all sets of values. The other two instructions locate one set of values for each desired Y.

Each of the instructions ISETUP and ITERATE has 4 arguments; all column numbers. Let the arguments be C1, C2, C3 and C4. Column C1 contains values of x, column C2 corresponding values of y = f(x) and column C3 the desired values Y = f(X). Results are put in the 4 consecutive columns C4, (C4+1), (C4+2) and (C4+3). Each instruction looks for two adjacent values of y, y_j and y_{j+1}, in column C2 which bracket the desired value, Y_i, in column C3, i.e., $y_j < Y_i < y_{j+1}$, if the y's are monotonic increasing, or $y_{j+1} < Y_i < y_j$, if the y's are monotonic decreasing. The average of these two values, y_j and y_{j+1}, is put in column (C4+2). The desired value, Y_i, is put in column (C4+3). The average of the corresponding values of x in column C1, x_j and x_{j+1}, is put in column (C4+1). The instruction then inserts 3 values at equal intervals between x_j and x_{j+1} and puts all 5 values in column C4. The process is repeated for each desired Y. The results stored are summarized in the following table.

C4	C4+1	C4+2	C4+3
x_j	$(x_j+x_{j+1})/2$	$(y_j+y_{j+1})/2$	Y_i
$x_j+1\Delta$			
$x_j+2\Delta$			
$x_j+3\Delta$			
x_{j+1}			

where $\Delta = (x_{j+1}-x_j)/4$.

The last two arguments of ISOLATE indicate storage columns. The information stored in these two columns is the same as C4 and C4+1 in ISETUP and ITERATE.

NRMAX is set equal to 5 times the number of values located (or (p+2) times the number located by ISOLATE). Therefore, NRMAX may be either increased or decreased. If no values are located, the informative diagnostic

* ITERATION HAS FOUND NO VALUES

is printed and the value of NRMAX is not changed.

The following informative diagnostic will be given, if the number of results to be stored is greater than the number of rows in the worksheet (normally, 201):

* WORKSHEET IS TOO SHORT TO ACCOMMODATE ALL THE VALUES GENERATED BY THIS COMMAND.

If a root is found exactly, only one number is put in column C4 and NRMAX is reset to 1.

The values in the column specified by the first argument of ISOLATE or ISETUP should be monotonic. If this is not the case, the command will be executed but the following informative diagnostic will be given:

* 1ST COLUMN OF ISETUP OR ISOLATE IS NOT MONOTONIC OR IS A CONSTANT.

ISETUP x in (C), y in (C), desired y in (C), put in col (C) and next three cols

This instruction is used initially to set up values needed by the instruction ITERATE. The following set of instructions uses ISETUP and ITERATE to find the values of x, approximating X, for which Y = cos(X) = .6, .7 and .8. The results of using this set of instructions are shown on page 173.

```
OMNITAB EXAMPLE OF ISETUP AND ITERATE
    GENERATE X EQUAL TO 0 (.1) 1.0 IN COL 1
    GENERATE Y EQUAL TO .6 (.1) .8 IN COL 14
    COS OF X IN 1 PUT RESULT IN COL 12
    PRINT  1  12  14  WITH 6.0 SIGNIFICANT DIGITS
    ISETUP  X IN 1, Y IN 12, DESIRED Y IN 14 START STORING IN 1
    TITLE1 RESULTS OF INSTRUCTION ISETUP
    PRINT COLS 1***4 WITH 6.0 SIGNIFICANT DIGITS
    1/COS OF X IN 1 PUT Y IN 12
    2/ITERATE X IN COL 1, Y IN 12 DESIRED Y IN 14 PUT IN 1
    3/IFEQ COL 3 TO COL 4 TOLERANCE .0001
    PERFORM INSTRUCTIONS 1 THRU 3 100 TIMES
    TITLE1 SOLVE COS (X) =.6, .7, .8 FOR X BETWEEN 0 AND 1
    PRINT  2  3  4 WITH 6.0 SIGNIFICANT DIGITS
```

In the 1st set of printed results, column 1 initially contains generated values which are supposed to bracket the values of X. Column 12 contains the corresponding values of y = cos(x) and column 14 the desired cosines, Y = .6, .7 and .8. These are the values used by the instruction ISETUP.

The 2nd set of printed results shows the results of using ISETUP. For Y = .6, ISETUP determines that the bracketing values of y in column 2 are 0.621610 and 0.540302 and the corresponding values of x in column 1 are 0.9 and 1.0. The instruction then inserts 3 values of x (.925, .95 and .975) at equal intervals between .9 and 1.0. These 5 values were then put in the first 5 rows of column 1. The average of .90 and 1.0, .95, was put in row 1 of column 2, the average of the corresponding cosines was put in column 3 and the desired value .6 was put in row 1 of column 4. This process was repeated for Y = .7 and Y = .8. The numbers in rows 6 through 10 of column 1 were obtained for Y = .7 and the values in rows 11 through 15 were obtained for Y = .8. Before ISETUP was used, NRMAX was equal to 11. But, after ISETUP was executed NRMAX was set equal to 15 = 5x3.

ISOLATE x in (C), y in (C), desired y equal to (K), put in columns (C) and (C)

The following set of instructions uses ISOLATE to solve the equation sin(X) = 0.0, for all values of X between 1.0 and 5.0. The results of using this set of instructions are shown on page 174.

COLUMN 1	COLUMN 12	COLUMN 14
0.	1.00000	.600000
.100000	.995004	.700000
.200000	.980067	.800000
.300000	.955336	
.400000	.921061	
.500000	.877583	
.600000	.825336	
.700000	.764842	
.800000	.696707	
.900000	.621610	
1.00000	.540302	

RESULTS OF INSTRUCTION ISETUP

COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4
.900000	.950000	.580956	.600000
.925000	.750000	.730774	.700000
.950000	.650000	.795089	.800000
.975000			
1.00000			
.700000			
.725000			
.750000			
.775000			
.800000			
.600000			
.625000			
.650000			
.675000			
.700000			

SOLVE COS (X) =.6, .7, .8 FOR X BETWEEN 0 AND 1

COLUMN 2	COLUMN 3	COLUMN 4
.927295	.600000	.600000
.795361	.700027	.700000
.643506	.799997	.800000

SOLVE SIN(X)=0.0 FOR X BETWEEN 1 AND 5

COLUMN 1	COLUMN 2	COLUMN 31
1.0000000	.84147098	
1.5000000	.99749498	
2.0000000	.90929742	
2.5000000	.59847214	
3.0000000	.14112001	
3.5000000	-.35078323	
4.0000000	-.75680249	
4.5000000	-.97753011	
5.0000000	-.95892427	

COLUMN 1	COLUMN 2	COLUMN 31
3.0000000	.14112001	3.2500000
3.1250000	.016591892	
3.2500000	-.10819513	
3.3750000	-.23129381	
3.5000000	-.35078323	

COLUMN 1	COLUMN 2	COLUMN 31
3.1250000	.016591892	3.1875000
3.1562500	-.014656821	
3.1875000	-.045891223	
3.2187500	-.077080811	
3.2500000	-.10819513	

COLUMN 1	COLUMN 2	COLUMN 31
3.1250000	.016591892	3.1406250
3.1328125	.0087800407	
3.1406250	.00096765344	
3.1484375	-.0068447927	
3.1562500	-.014656821	

COLUMN 1	COLUMN 2	COLUMN 31
3.1406250	.00096765344	3.1445313
3.1425781	-.00098547115	
3.1445313	-.0029385920	
3.1464844	-.0048917017	
3.1484375	-.0068447927	

COLUMN 1	COLUMN 2	COLUMN 31
3.1406250	.00096765344	3.1416016
3.1411133	-.00098547115	
3.1416016	-.0029385920	
3.1420898	-.0048917017	
3.1425781	-.0068447927	

```
OMNITAB  EXAMPLE OF ISOLATE
    NOTE  SOLVE SIN(X)=0.0 FOR X BETWEEN 1 AND 5
    GENERATE TRIAL VALUES OF X  1. (.5) 5. IN COL 1
    NOTE1     COLUMN 1        COLUMN 2        COLUMN 31
    1/SIN OF X IN COL 1 PUT SIN(X) IN COL 2
    1.1/SPACE
    1.2/PRINT NOTE
    1.3/NPRINT COLS 1,2,31
    2/ISOLATE X IN COL 1 FOR Y IN COL 2 VALUE=0.0 PUT IN COL 1,31
    PERFORM INSTRUCTIONS 1 THRU 2 5 TIMES
    SPACE
    PRINT NOTE
    NPRINT COLS 1,2,31
```

The initial values of x in column 1 and y = sin(x) in column 2 are shown in the 1st printing of columns 1, 2 and 31. NRMAX equals 9. The ISOLATE instruction is repeated five times with results printed after each execution. Since there is only one value of X between 1.0 and 5.0 for which sin(X) = 0.0, column 31 has one result. The value in column 31 tends to come closer to the correct value, π = 3.1415927, each time ISOLATE is used. Column 1 contains the 5 nearest values of x for which sin(x) = 0.0 ($1.0 \leq x < 5$) and column 2 contains the corresponding values of y = sin(x). NRMAX is decreased to 5 after the first time ISOLATE is used.

ISOLATE x in (C), y in (C), for (K), use (p) points, put in cols (C) and (C)

This instruction is the same as the one above, however the (p+2) nearest values of x are located rather than five.

ITERATE x in (C) y in (C) desired y in col (C) put in (C) and next three cols

The ITERATE instruction is used in the repeat mode and performs exactly like ISETUP. In the example for ISETUP, ITERATE was used to refine the values of x until the specified absolute tolerance was met (e.g., .0001 in the IFEQ instruction). The 2nd PRINT shows the desired values of cos(x) = .6, .7 and .8 in column 4. Column 2 contains the values of x found by ITERATE and column 3 contains the corresponding values of cos(x), approximating the values in column 4. The 3rd PRINT shows the results after the tolerance has been satisfied. That is the values in column 3 must be equal to the values in column 4 within the tolerance of .0001.

Note, if y = f(x) approximates the desired Y = f(X) to within the specified tolerance, this does not necessarily mean that the value of x found by ITERATE will approximate X within the specifed tolerance.

5.4 Analysis.

HARMONIC, INTERPOLATE, MAXMIN, SOLVE

HARMONIC analysis of y in col (C) for (n) ordinates, put coefficients in col (C)

HARMONIC evaluates the coefficients A_k and B_k for the periodic function

$$y_i = A_o + \sum_{i=1}^{r} A_k \cos(k\theta_i) + \sum_{i=1}^{s} B_k \sin(k\theta_i),$$

where i=1,2,...,n and $2<n\leq NRMAX$; r=[n/2], s=[(n-1)/2] and n=1+r+s; $\theta = 2\pi x/T$, T is the period and $\theta_i = (i-1)2\pi/n$. If n is even, then s=r-1 and the coefficients are stored in the sequence

$$A_o, A_1, \ldots, A_{n/2}, B_1, B_2, \ldots, B_{(n-2)/2}.$$

If n is odd, r=s and the coefficients are stored in the sequence

$$A_o, A_1, \ldots, A_{(n-1)/2}, B_1, B_2, \ldots, B_{(n-1)/2}.$$

If NRMAX exceeds 1000, only the first 1000 coefficients will be stored. The integer (n) must be greater than 2 and must not exceed NRMAX. If (n) is less than NRMAX, only the first (n) values of y are used.

For the 12 values of y in column 1 shown below, the instruction

HARMONIC analysis of column 1 for 12 points, put coefficients in col 2

will put seven coefficients, A's, in rows one through seven of column 2 and five coefficients, B's, in rows eight through twelve of column 2 as follows:

	y	A and B coefficients
Row/Column	1	2
1	9.3	9.2166666
2	15.0	-6.9027767
3	17.4	3.45
4	23.0	3.3
5	37.0	-.61666666
6	31.0	.60277674
7	15.3	.24999998
8	4.0	21.089591
9	-8.0	-2.8001488
10	-13.2	1.9666667
11	-14.2	1.068098
12	-6.0	-1.0229243

The above example is from Scarborough, J. B., "Numerical Mathematical Analysis", Oxford University Press, 1950, Chapter XVII.

INTERPOLATE x in (C) y in (C), length (n) values (v) in (C), pts (p), put in (C)

INTERPOLATE provides (p)-point Lagrangian interpolation for tabulated values of y = f(x) in the two columns designated by the 1st and 2nd arguments. The length of the original table is designated by the 3rd argument, (n). Values of the independent variable x, in the column designated by the 1st argument, must be in either ascending or descending order, but need not be uniformly spaced. The 4th argument indicates the number of values in the column designated by the 5th argument which are to be interpolated. The (v) interpolated values of y are put in the column designated by the 7th (last) argument. Extrapolation will be done for values outside the table, but the following informative diagnostic will be given:

* EXTRAPOLATION DONE FOR MORE THAN ONE DELTA

If (p) is greater than (n), p is set equal to n and the following informative diagnostic is given:

* ORDER OF INTERPOLATION EQUALS LIST SIZE

When the value of (p) is very large, there may not be enough room to do the amount of interpolation requested. This will happen if (p^2+3p+v) exceeds the number of locations in the worksheet, 12,500 (for NBS computer). The value of p is reset to the largest value possible and the following informative diagnostic is given:

* ORDER OF INTERP WAS RESET TO (n) DUE TO SIZE OF SCRATCH AREA

If columns 1, 2 and 3 contain the following values:

Columns	1	2	3
	0.0	0.0	.25
	.15	.14943813	.35
	.30	.29552020	
	.45	.43496553	
	.60	.56464246	

then the instruction

INTERPOLATE x in col 1 y in col 2 length 5 for 2 values in col 3 4 pts. put in 4

will put the values .24740159 and .34289405 in column 4.

MAXMIN x in (C) y in (C) put max x. in (C) max y in (C) min x in (C) min y in (C)

MAXMIN finds all the maxima and minima of a function (y = f(x)) defined by its tabulated values. The validity depends on the adequacy of the interval of tabulation and monotonic arrangement of x for a single valued function. Each extreme value in a sequence of y values is identified and a parabola, $y = ax^2+bx+c$, is fitted through the extreme point and one on each side. The maxima and minima are determined by setting the derivative equal to zero to obtain $x = -b/2a$. The stored results may be used to determine an envelope for the y values.

If the y values are monotonic, the following fatal error message is printed:

* MAXMIN HAS FOUND NO EXTREMA

If an apparent extremum is found where the x values are identical, the extremum is ignored and the following informative diagnostic is given:

* MAXMIN HAS FOUND AND IGNORED A TRIAD OF X'S WITH AT LEAST TWO IDENTICAL VALUES

Assume column 10 contains the values of x in degrees -10, 0, 10, ... , 370 and column 11 contains the corresponding values of y = cos(x), then the instruction

MAXMIN x in col 10 y in col 11, put max x in 12 y in 13, put min x in 14 y in 15

finds two maxima whose coordinates are put in rows 1 and 2 of columns 12 and 13 and one minimum whose coordinates are put in row 1 of columns 14 and 15 as follows:

	max x	max y	min x	min y
Row/Column	12	13	14	15
1	3.8314035-08	1.0	180.0	-1.0
2	360.0	1.0000002		

The command EXTREMA is a synonym for MAXMIN.

SOLVE 1in eqs with coeffs in (R),(C) size (r)x(c) consts in (C), put sol in (C)

SOLVE provides the solution of a set of simultaneous linear equations, Ax = y, where A is a square matrix of coefficients defined by the first four arguments, y is a vector of constants and x is the solution vector. See sections 7 and 8 for the method of defining an array or matrix. The 3rd and 4th arguments should be equal.

If a solution is obtained and the list of commands is printed at the end of a set of OMNITAB instructions, a smallest error bound will be given below the command SOLVE in the following form:

++...+SMALLEST ERROR BOUND ON INVERTED MATRIX IS.....+...+

For a complete description of the meaning of the error bound, see section 8.5.

If r≠c, the following informative diagnostic will be given:

* NO. OF ROWS NOT = TO COLS. MATRIX USED IS LARGEST SQUARE

Furthermore, if r and c are inadvertently too large, the following fatal error message is given:

*** DEFINED MATRIX OVERFLOWS WORKSHEET

In order to solve a set of equations a large amount of scratch area is needed. If 2(r+2) is greater than the worksheet area, the following fatal error is given:

*** INSUFFICIENT SCRATCH AREA

Finally, if the matrix of coefficients, A, is singular, the following fatal error is given:

*** MATRIX IS (NEARLY) SINGULAR

If the matrix of coefficients, A, and the vector of constant, y, of a set of 10 simultaneous linear equations is:

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \qquad y = \begin{bmatrix} 0.5 \\ 1.0 \\ 1.5 \\ 2.0 \\ 2.5 \\ 3.0 \\ 3.5 \\ 4.0 \\ 4.5 \\ 5.0 \end{bmatrix}$$

Then, the instruction

SOLVE eqs. with coeff matrix A in 1,5 size 10x10, y in col 1, put solution in col 2

will put the solution (x vector) in column 2.

Column 2
5.4545431
10.090905
13.090903
14.636356
14.909083
14.090901
12.363629
9.9090850
6.9090868
3.5454524

5.5 Integration.

GAUSS QUADRATURE

GAUSS QUADRATURE with (K) points, from (K) to (K), put x in (C), weights in (C)

GAUSS QUADRATURE generates the (n) abscissas, x_i, and the corresponding weight coefficients, w_i, for the Gaussian quadrature formula

$$\int_b^a f(x)dx = \sum_{i=1}^{n} w_i f(x_i),$$

where n is the number of points specified by the first argument in the instruction. The region of integration is divided into n/4 intervals. In each of the intervals the 4-point Gaussian quadrature abscissas and weight coefficients are computed. The abscissas are put in the column specified by the 4th argument and the weight coefficients are put in the column specified by the 5th (last) argument. The limits of integration (a and b) are defined by the 2nd and 3rd arguments of the instruction. (The 1st, 2nd and 3rd arguments may be integers.)

The value of (n) must be greater than zero, some multiple of 4 and less than the current number of rows in the worksheet (normally, 201).

If the value of NRMAX, before execution of this instruction, is less than n, NRMAX will be set equal to n. Otherwise, NRMAX will remain unchanged. The instruction

GAUSS QUADRATURE using 8 pts from a = 0.0 to b = 1.0, put x in col 4 wts in col 5

would put the following results in the worksheet:

Row/Column	4	5
1	.034715922	.086963711
2	.16500474	.16303629
3	.33499526	.16303629
4	.46528408	.086963711
5	.53471592	.086963711
6	.66500474	.16303629
7	.83499526	.16303629
8	.96528408	.086963711

6. REPEAT MODE.

A discussion of how to use instructions repeatedly (in the repeat mode) was given in section B2. This section gives specific details on the use of individual instructions. The reader should refer to section B2 for the general discussion and further details.

Instructions in sections 6.1 and 6.2 use the argument (N), where (N) is an instruction number. See section B2.1. These are the only instructions in PART C which use this type of argument.

6.1 Repeated Execution.

BEGIN, FINISH, PERFORM

The PERFORM instruction executes numbered instructions which have been previously stored for later use. There are three forms of the instruction having 3, 2 and 1 arguments, respectively. The commands EXECUTE and REPEAT are commonly used synonyms for PERFORM.

Instructions may be stored for later use by numbering them, as described in section B2.1. An alternative (often less desirable) method for storing instructions is by the use of the BEGIN and FINISH commands which provide automatic numbering of the instructions between them. BEGIN and FINISH must not be numbered (stored), but a PERFORM instruction can be stored for later use.

BEGIN storing instructions for later use

The instructions which follow are stored for later use until a FINISH instruction is encountered. Each instruction is given a unique number, starting with 1 and continuing in steps of 1 until the FINISH instruction is found. The FINISH instruction is not stored. The instruction numbers are automatically printed in the LIST OF COMMANDS, DATA, AND DIAGNOSTICS on the left of the stored instructions.

BEGIN storing instructions, but start numbering with number (N)

Similar to above instruction; except the numbering of instructions, which follow, begins with the number (N) rather than one. Numbering of instructions continues in steps of one until a FINISH instruction is encountered. In this instruction (only), the instruction number (N) must be an integer, without a decimal point. The integer must be greater than zero and less than 1000. Warning, if the instruction number is written with a decimal point, it will not only create a fatal error here, but also will be indicated in some subsequent instructions. For example, when a reference is made to any of the stored instructions by another instruction (such as PERFORM or RESTORE), a fatal error will be indicated each time because the referenced instruction cannot be found.

FINISH storing instructions for later use

Instructs OMNITAB to terminate the storing and automatic numbering of instructions initiated by a BEGIN instruction. The FINISH instruction does not cause the stored instructions to be executed. This happens only when PERFORM is executed.

```
: PERFORM instructions numbered (N) through (N), (t) times :
```

Causes the execution of all instructions which have instruction numbers between the number designated by the first argument and the number designated by the second argument inclusive. The group of instructions is executed (t) times (unless countermanded by one of the branching instructions in sections 6.3 and 6.4). Prior to the use of a PERFORM instruction, there must be stored instructions with numbers corresponding to the first two arguments of the PERFORM instruction; otherwise a fatal error will occur. The 1st argument must be less than or equal to the 2nd argument.

```
/ PERFORM instructions numbered (N) through (N) once /
```

If the third argument is missing, it is assumed to be equal to one.

```
/ PERFORM instruction numbered (N) once /
```

If the second and third arguments are missing, the second argument is assumed to be equal to the first and the third argument is assumed to be equal to one.

6.2 Incrementing Instructions.

INCREMENT, RESTORE

The two instructions described in this section are used to change the arguments of a stored (numbered) instruction from one repeat of a set of instructions to another.

```
: INCREMENT instruction (N) by (E), (E) ... (E) :
```

Upon execution of an INCREMENT instruction, the arguments of the instruction numbered (N) are modified as indicated. The INCREMENT instruction always has exactly one more argument than the instruction which is being incremented. The arguments (after (N)) must be of the same kind as the corresponding arguments in the instruction (N). The 2nd argument of the INCREMENT instruction is added to the 1st argument of the indicated stored instruction, the 3rd argument of the INCREMENT instruction is added to the 2nd argument of the stored instruction, and so forth. If the instruction

```
41/ DEFINE the value          1.0 into column 54
```

is to be incremented, the INCREMENT instruction must have 3 (1+2) arguments. The second argument, corresponding to 1.0, must be written with a decimal point and the third argument, corresponding to 54, must be written without a decimal point. Hence, the instruction might be

```
42/ INCREMENT instruction 41 by 1.0      and      -1
```

After the execution of instruction 42, instruction 41 will, in effect, read DEFINE 2.0 into column 53.

An INCREMENT instruction can not be used to increment itself, i.e., the argument (N) must not be the same as the number of the INCREMENT instruction. See section B2.3 for further details.

The INCREMENT instruction does not have to be numbered (stored). The instructions

```
1/ DEFINE     the number 1.0 into column 1
2/ INCREMENT instr 1 by 1.0      and      0
   PERFORM instrs 1 thru 2, 9 times
   INCREMENT instr 1 by 8.6      and      0
   PERFORM instrs 1 thru 2, 9 times
```

would put 9.0 into column 1 after the execution of the first PERFORM instruction and would put 26.6 into column 1 after the execution of the second PERFORM instruction.

```
:-------------------------------------------------------------------------:
:  RESTORE instruction (N) to (E), (E) ... (E)                            :
:-------------------------------------------------------------------------:
```

When the instruction is executed, the arguments of the instruction numbered (N) will be replaced by the specified arguments. The number of arguments after (N) must be exactly the same as the number of arguments in the instruction numbered (N). Also, the arguments in the RESTORE instruction (after (N)) must be of the same kind as the corresponding arguments in the instruction (N). A RESTORE instruction does not have to be numbered.

6.3 Branching, Three Arguments.

COMPARE, IFEQ, IFNE

The instructions described in sections 6.3 and 6.4 may only be used as stored instructions in the repeat mode. They provide the capability of branching within a group of stored instructions by (1) terminating the execution of a subset of stored instructions or by (2) terminating the execution of a PERFORM instruction. The instructions in this section all have three arguments and use either a relative tolerance or an absolute tolerance specified by the last argument. The instructions in section 6.4 are similar but have only two arguments and do not use a tolerance.

Each instruction makes a comparison between two numbers or sets of numbers specified by the first two arguments, which may be either a constant or a column number. If the tolerance is not satisfied, or the condition is false, for any pair of values, no action is taken and, in effect, the instruction is ignored. If the designated tolerance is satisfied for all pairs of values, the action taken depends upon whether the instruction is (a) the last instruction in a subset of stored instructions, or (b) is not the last instruction in the subset.

If the tolerance is satisfied, or the condition is true, and the branching instruction is the last in the subset of stored instructions, the execution of the PERFORM instruction is terminated. For example, in the instructions

```
21/ DEFINE      the value 1.0 into column 1
22/ INCREMENT instr 21 by 1.0      and      0
23/ IFEQ 4.0 to column 1 within 0.5
24/ PERFORM instrs 21 thru 23, 7 times
```

the IFEQ instruction is the last in the subset of instructions and the PERFORM instruction will be terminated when every number in column 1 equals 4.0. In this case, the execution of instructions 21 through 23 will be terminated after the fourth time and the number 4.0 will be in column 1. The IFEQ instruction does not have to be the last stored instruction in the entire set, but only the last in the subset specified by the PERFORM instruction which executes the IFEQ instruction. In other words, the second argument of the PERFORM instruction, 23, must be the same as the instruction number, 23, of the IFEQ instruction.

If a tolerance is satisfied and the branching instruction is not the last instruction in the subset of stored instructions, the remaining instructions after the branching instruction are not executed. In the execution of the instructions

```
31/ DEFINE      the value 1.0 into column  1
32/ INCREMENT instr 31 by 1.0       and    0
33/ IFEQ 3.0 to column 1 within 0.5
34/ ADD the number 10.0 to col 1 and put in col 11
35/ INCREMENT 34 by 0.0        0     and        1
    PERFORM instrs 31 thru 35, 5 times
```

the number 5.0 will be put into column 1 and the numbers 11.0, 12.0, 14.0 and 15.0 into columns 11 through 14. In the comparison of 3.0 with column 1, in instruction 33, the equality condition is true after the 3rd execution of instruction 31 and instructions 34 and 35 are not executed this time only. The next two times that instruction 33 is executed the condition is not true and instructions 34 and 35 are executed.

COMPARE (E) to (E) using relative tolerance (E)

This is the only branching instruction which uses a relative tolerance. If the first argument is a constant, let ARG1 equal the constant. If the first argument is a column number, let ARG1 equal the number in any row of the column. Define ARG2 and ARG3 in a similar manner. For ARG1 $\neq$ 0 and ARG2 $\neq$ 0, the relative tolerance is satisfied if

$$|(ARG1-ARG2)/ARG2| < |ARG3|.$$

If ARG1=0 or ARG2=0, the tolerance is satisfied if

$$|ARG1-ARG2| < |ARG3|.$$

Also, if ARG1=0 or ARG2=0, the following arithmetic fault message is given:

** ONE OF THE VALUES COMPARED IS ZERO, ABSOLUTE TOLERANCE WAS USED (n) TIMES

For the numbers

Column 56	Column 57	Column 58
1.0	5.0	0.9
5.0	1.0	0.5
-2.0	0.0	1.0

the instruction

COMPARE column 56 with 57 using relative tolerance in column 58

would compare the numbers 0.8, 4.0 and 2.0 with 0.9, 0.5 and 1.0, respectively. The tolerance is met in row 1, but not in rows 2 or 3. Hence, the tolerance is not satisfied.

IFEQ (E) to (E) within the absolute tolerance (E)

The specified tolerance is satisfied, if, and only if, the absolute value of the difference between all the numbers designated by the first two arguments is equal to the number(s) designated by the third argument. For the numbers

Column 26	Column 27	Column 28
1.6	2.3	0.8
1.5	2.7	0.6
-5.3	1.8	7.9

the absolute tolerance in column 28 is met in rows 1 and 3, but not in row 2. Since the tolerance is not satisfied in all three rows, the tolerance in the instruction

IFEQ column 26 to column 27 within absolute tolerance in column 28

is not satisfied.

IFNE (E) to (E) within absolute tolerance (E)

This instruction operates exactly as the IFEQ instruction above with "not equal" replacing equal.

6.4 Branching, Two Arguments.

IFEQ, IFGE, IFGT, IFLE, IFLT, IFNE

These six instructions each have exactly two arguments. None of the instructions uses a tolerance, but in all other respects they operate like the IFEQ and IFNE instructions which are described in section 6.3. In fact, these two instructions have optional forms which are described here. The branching instructions with two arguments do not use a tolerance. Instead, the specified condition is either true or false.

IFEQ (E) to (E)

The condition is true, if, and only if, the number(s) specified by the first argument is (are all) equal to the number(s) specified by the second argument.

IFGE (E) to (E)

The condition is true, if, and only if, the number(s) specified by the first argument is (are all) greater than or equal to the number(s) specified by the second argument.

IFGT (E) to (E)

The condition is true, if, and only if, the number(s) specified by the first argument is (are all) greater than the number(s) specified by the second argument.

IFLE (E) to (E)

The condition is true, if, and only if, the number(s) specified by the first argument is (are all) less than or equal to the number(s) specified by the second argument.

IFLT (E) to (E)

The condition is true, if, and only if, the number(s) specified by the first argument is (are all) less than the number(s) specified by the second argument.

IFNE (E) to (E)

The condition is true, if, and only if, the number(s) specified by the first argument is (are all) not equal to the number(s) specified by the second argument.

7. ARRAY OPERATIONS.

In general, OMNITAB instructions perform operations on columns of data (not necessarily consecutive), working from the first row down to row NRMAX. In this section, instructions are described which perform operations on a rectangular (or square) array (of consecutive rows and columns), which may be located anywhere in the worksheet, including that portion below NRMAX. None of the instructions here have any effect on the value of NRMAX nor are they influenced in any way by NRMAX. Each command begins with the letter A, standing for array. The array operation instructions described herein can be used very effectively for data manipulation.

Four arguments designate an array in the worksheet:

(R) = the row number of the value in the upper left hand corner of the array
(C) = the column number of the value in the upper left hand corner of the array
(r) = the number of rows in the array
(c) = the number of columns in the array

The first four arguments (all integers) of each instruction in this section are always of the above form and designate an array. Some of the instructions have additional arguments which may designate a second or third array.

The descriptions of the instructions in sections 7.1 and 7.2 include an example based upon the following data in the worksheet:

Row/Column	11	12	13	Row/Column	26	27	28
7	2.0	0.0	8.0	14	2.0	5.0	2.0
8	6.0	4.0	-2.0	15	3.0	-2.0	4.0

and the numbers 1.0, 2.0, 3.0, 4.0 and 5.0 in column 41.

Care should be used with instructions for array operations to avoid designating an array which is partially outside the worksheet. In most instructions, if the arguments specify an array partially outside the worksheet, the following fatal error message appears:

*** DEFINED MATRIX OVERFLOWS WORKSHEET

7.1 Arithmetic.

AADD, ADIVIDE, AMULTIPLY, ARAISE, ASUBTRACT

Each of the instructions in this section has three forms having 10, 8 and 7 arguments respectively. All involve three arrays. The third array is designated simply by (R),(C) without using the size (r)x(c), since the size is always the same as the size of the first array. In the first form (10 arguments) the second array is completely designated by the four arguments (R),(C) and (r)x(c), even though the size of this array must be the same as the size of the first array. The second form (8 arguments) is the same as the first except the size (r)x(c) of the second array is omitted from the instruction. The third form (7 arguments) is used when the second array is a constant (scalar, with r=c=1) or column (vector, with R=1, r=(r) of first array and c=1).

The ADIVIDE, AMULTIPLY and ARAISE instructions in this section can produce the same type of arithmetic faults as the analagous instructions of section 2.1. See section 2.1 and section B3.3 for further details.

AADD the array (R),(C) size (r)x(c) to array (R),(C) size (r)x(c) put in (R),(C)

Two arrays with the same dimensions are added together element by element. The instruction

AADD the array in 7,11 size 2x3 to array in 14,26 size 2x3 put in 4,32

would give the following result:

Row/Column	32	33	34
4	4.0	5.0	10.0
5	9.0	2.0	2.0

AADD the array in (R),(C) size (r)x(c) to array (R),(C) and put in (R),(C)

Same as above; except the row and column size of the second array are ommitted. The instruction (equivalent to the one above)

AADD the array in 7,11 of size 2x3 to the array in 14,26 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	4.0	5.0	10.0
5	9.0	2.0	2.0

AADD the array in (R),(C) of size (r),(c) to (E) and put array in (R),(C)

Allows the addition of either a column or a constant to an array. When a column is added to an array, it is added to each column of the array. The first (r) rows of the column are used regardless of the value of NRMAX. The instruction

AADD the array in 7,11 of size 2x3 to column 41 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	3.0	1.0	9.0
5	8.0	6.0	0.0

When a constant is added to an array, it is added to each element in the array. The instruction

AADD the array in 7,11 of size 2x3 to -1.3 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	0.7	-1.3	6.7
5	4.7	2.7	-3.3

ADIVIDE array (R),(C) size (r)x(c) by array (R),(C) size (r)x(c) put in (R),(C)

The first array is divided, element by element, by the second array and stored as indicated. The command ADIV is an acceptable abbreviation of ADIVIDE in all three forms of the instruction. The instruction

ADIVIDE the array in 7,11 size 2x3 by the array in 14,26 size 2x3 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	1.0	0.0	4.0
5	2.0	-2.0	-0.5

ADIVIDE the array in (R),(C) size (r)x(c) by the array (R),(C) put in (R),(C)

Same as above; except the row and column size of the second array are omitted.

ADIVIDE the array in (R),(C) of size (r)x(c) by (E) and put in (R),(C)

Allows for the division of an array by either a column or a constant. When an array is divided by a column, each column of the array is divided by that column. The first (r) rows of the designated column are used regardless of the value of NRMAX. The instruction

ADIVIDE the array in 7,11 of size 2x3 by column 41 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	2.0	0.0	8.0
5	3.0	2.0	-1.0

When an array is divided by a constant, each number in the array is divided by the constant. The instruction

ADIVIDE the array in 7,11 of size 2x3 by 2.0 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	1.0	0.0	4.0
5	3.0	2.0	-1.0

AMULTIPLY array (R),(C) size (r)x(c) by (R),(C) size (r)x(c) put in (R),(C)

Element by element multiplication is performed on two arrays. The command AMULT is an acceptable abbreviation for AMULTIPLY in all three forms of the instruction. The instruction

AMULTIPLY the array in7,11 size 2x3 by the array in 14,26 size 2x3 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	4.0	0.0	16.0
5	18.0	-8.0	-8.0

AMULTIPLY array in (R),(C) size (r)x(c) by array in (R),(C) put in (R),(C)

Same as above; except the row and column size of the second array are omitted.

AMULTIPLY the array in (R),(C) of size (r)x(c) by (E) and put array in (R),(C)

Allows the multiplication of an array by either a column or a constant. When an array is multiplied by a column, each column of the array is multiplied by that column, element by element. The first (r) rows of the designated column are used regardless of the value of NRMAX. The instruction

AMULTIPLY the array in 7,11 of size 2x3 by column 41 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	2.0	0.0	8.0
5	12.0	8.0	-4.0

When an array is multiplied by a constant, each number in the array is multiplied by that constant. The instruction

AMULTIPLY the array in 7,11 of size 2x3 by 2.0 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	4.0	0.0	16.0
5	12.0	8.0	-4.0

ARAISE array (R),(C) size (r)x(c) to array (R),(C) size (r)x(c) put in (R),(C)

The first array is raised, element by element, to the power of the second array. The instruction

ARAISE the array in 7,11 size 2x3 to array in 14,26 size 2x3 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	4.0	0.0	64.0
5	216.0	.0625	16.0

ARAISE the array in (R),(C) size (r)x(c) to array (R),(C) put array in (R),(C)

Same as above; except the row and column size of the second array are omitted.

ARAISE the array in (R),(C) of size (r)x(c) to (E) and put array in (R),(C)

Allows the first array to be raised to a column or a constant. When an array is raised to a column, each column of the array is raised to the column, row by row. The first (r) rows of the column are used regardless of the value of NRMAX. The instruction

ARAISE the array in 7,11 of size 2x3 to column 41 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	2.0	0.0	8.0
5	36.0	16.0	4.0

When an array is raised to a constant, each number in the array is raised to the constant. The instruction

ARAISE the array in 7,11 of size 2x3 to 2.0 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	4.0	0.0	64.0
5	36.0	16.0	4.0

ASUBTRACT array (R),(C) size (r)x(c) minus (R),(C) size (r)x(c) put in (R),(C)

The second array is subtracted element by element from the first array. Note the difference in the order of subtraction from that in a SUBTRACT instruction (see section 2.1). The command ASUB is an acceptable abbreviation for ASUBTRACT in all three forms of the instruction. The instruction

ASUBTRACT the array in 7,11 size 2x3 minus array in 14,26 size 2x3 put in 4,32

would give the following result:

Row/Column	32	33	34
4	0.0	-5.0	6.0
5	3.0	6.0	-6.0

ASUBTRACT array (R),(C) of size (r)x(c) minus the array (R),(C) put in (R),(C)

Same as above; except the row and column size of the second array are omitted.

ASUBTRACT the array in (R),(C) of size (r)x(c) minus (E) and put array in (R),(C)

Allows the subtraction of either a column or a constant from an array. When a column is subtracted from an array, it is subtracted from each column of the array. The first (r) rows of the column are used regardless of the value of NRMAX. The instruction

ASUBTRACT the array in 7,11 of size 2x3 minus col 41 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	1.0	-1.0	7.0
5	4.0	2.0	-4.0

When a constant is subtracted from an array, it is subtracted from each number in the array. The instruction

ASUBTRACT the array in 7,11 of size 2x3 minus 2.0 and put in 4,32

would give the following result:

Row/Column	32	33	34
4	0.0	-2.0	6.0
5	4.0	2.0	-4.0

7.2 Data Manipulation.

ADEFINE, AERASE, AMOVE, ATRANSPOSE

ADEFINE the array in (R),(C) of size (r)x(c) to be equal to (K)

Every element in the designated array is set equal to the constant (K), The instruction

ADEFINE the array in 4,32 of size 2x3 to be equal to 7.5

would give the following result:

Row/Column	32	33	34
4	7.5	7.5	7.5
5	7.5	7.5	7.5

AERASE the array in (R),(C) of size (r)x(c)

Every entry in the array is set equal to zero. AZERO is a synonym for AERASE. Actually, an ADEFINE instruction with K = 0.0 performs the same operation as the AERASE instruction which has the same first four arguments. The instruction

AERASE the array in 4,32 of size 2x3

would give the following result:

Row/Column	32	33	34
4	0.0	0.0	0.0
5	0.0	0.0	0.0

AMOVE the array in (R),(C) of size (r)x(c) to (R),(C)

Moves an array from one part of the worksheet to another. The second named array may overlap the first named array. The command MOVE, described in section 3.2, is synonymous with the command AMOVE. The instruction

AMOVE the array in 7,11 of size 2x3 to the array in 4,32

would give the following result:

Row/Column	32	33	34
4	2.0	0.0	8.0
5	6.0	4.0	-2.0

ATRANSPOSE the array in (R),(C) of size (r)x(c) into (R),(C)

Rotates an array 90 degrees so that the first row of the array becomes the first column of the new array, the second row becomes the second column and so forth. Hence, if the first array has (m) rows and (n) columns, the transpose will have (n) rows and (m) columns. If c=1, the instruction will transpose a column into a row, whereas if r=1 (and the 2nd last argument is R=1), the instruction will transpose a row into a column. The instruction

ATRANSPOSE the array in 7,11 size 2x3 into 4,32

would give the following result:

Row/Column	32	33
4	2.0	6.0
5	0.0	4.0
6	8.0	-2.0

7.3 Summarization.

AAVERAGE, ACOALESCE

There are two forms of each instruction. The second form of each instruction searches for a particular value (K) in an array. If the number is not found, the following informative diagnostic is given

* VALUE REQUESTED IN SHORTEN, ACOALESCE OR AAVERAGE NOT FOUND.

The two instructions are similar. The instruction ACOALESCE computes certain sums as described, whereas the instruction AAVERAGE computes the corresponding averages. The description of ACOALESCE should be read before turning to the description of AAVERAGE. Examples in this section are based on the following array in the worksheet:

Row/Column	11	12	13	14
21	1.0	0.0	1.0	2.0
22	0.0	2.0	1.0	3.0
23	1.0	2.0	0.0	1.0
24	0.0	1.0	2.0	0.0
25	2.0	1.0	3.0	1.0

AAVERAGE on first col of array in (R),(C) size (r)x(c) put array in (R),(C)

Similar to ACOALESCE below; except averages are computed instead of sums. The instruction

AAVERAGE on first col of array in 21,11 size 5x4 and put in 41,31

would yield

Row/Column	31	32	33	34
41	1.0	1.0	0.5	1.5
42	0.0	1.5	1.5	1.5
43	2.0	1.0	3.0	1.0

Column 31 is the same as in the example of ACOALESCE below. The number 0.5 in row 41 of column 33 is the average of 1.0 and 0.0 in rows of 21 and 23 of column 13. The other numbers were obtained in a similar manner.

Remark.

If the first column of the array contains positive integers only and each integer from 1 to the largest appears in at least one row, then AAVERAGE, with R=1, computes the same averages as the instruction ONEWAY, described in section 4.3. For the following data in the worksheet:

Row/Column	11	12	13	14
1	1	0	4	9
2	2	9	8	2
3	1	4	3	7
4	2	3	7	6
5	3	2	7	5
6	2	3	9	7
7	4	4	0	3
8	3	4	6	1
9	5	2	0	2
10	4	7	4	8

the instruction

AAVERAGE 1,11 size 10x4 put in 1,21

puts the same results in columns 21, 22, 23 and 24 as the three instructions

SONEWAY 12, 11 put in 21, 20, 22, 20
SONEWAY 13, 11 put in 21, 20, 23, 20
SONEWAY 14, 11 put in 21, 20, 24, 20

namely

Row/Column	21	22	23	24
1	1.0	2.0	3.5	8.0
2	2.0	5.0	8.0	5.0
3	3.0	3.0	6.5	3.0
4	4.0	5.5	2.0	5.5
5	5.0	2.0	0.0	2.0

AAVERAGE on (K) in first col of array (R),(C) size (r)x(c) put row in (R),(C)

Bears the same relation to the form above that the second form of ACOALESCE bears to the first form. The instruction

AAVERAGE on 1.0 in first col of array in 21,11 size 5x4 and put row in 41,31

would put the numbers 1.0, 1.0, 0.5 and 1.5 into row 41 of columns 31 to 34.

ACOALESCE on first col of array (R),(C) size (r)x(c) put array in (R),(C)

First, the instruction finds the m different values in the first column of the first array and puts these values into the first column of the second array. For any set of two or more numbers in the first column which are equal, only the first number in each set is put into the second array. Hence, the number of rows in the second array is m, not r.

Let a, b, c, ... be the numbers in the first row of the second array. Then, b is the sum of all the numbers in the rows of column 2 which contain the number a in the same rows of column 1; c is the sum of all numbers in the rows of column 3 which have a in the same row of column 1; and so on. The remaining rows are constructed in a similar way. The instruction

ACOALESCE on first col of array in 21,11 size 5x4, put array in 41,31

would yield the results

Row/Column	31	32	33	34
41	1.0	2.0	1.0	3.0
42	0.0	3.0	3.0	3.0
43	2.0	1.0	3.0	1.0

Column 11 has three distinct numbers 1.0, 0.0 and 2.0. These numbers are put into rows 41, 42 and 43 of column 31. The number 1.0 in row 41 of column 31 appears in rows 21 and 23 of column 11. Hence, the numbers 2.0, 1.0 and 3.0 in row 41 of columns 32 to 34 are the sum, respectively, of: 0.0 and 2.0 in rows 21 and 23 of column 12; 1.0 and 0.0 in rows 21 and 23 of column 13; and 2.0 and 1.0 in rows 21 and 23 of column 14. The number 0.0 in row 42 of column 31 appears in rows 22 and 24 of column 11. Hence, the numbers 3.0, 3.0 and 3.0 in row 42 of columns 32 to 34 are the sum, respectively, of: 2.0 and 1.0 in rows 22 and 24 of column 12; 1.0 and 2.0 in rows 22 and 24 of column 13; and 3.0 and 0.0 in rows 22 and 24 of column 14. The numbers in row 43 were obtained in a similar way.

ACOALESCE on (K) in first col of array (R),(C) size (r)x(c) put row in (R),(C)

Similar to above instruction. The second array has only 1 row and (c) columns. The first column contains the constant (K). Each other column contains the sum of all numbers in the corresponding column of the first array in those rows which have the constant (K) in the same row of column 1. The instruction

ACOALESCE on 1.0 in first col of array in 21,11 of size 5x4 put array in in 41,31

would put the numbers 1.0, 2.0, 1.0 and 3.0 into row 41 of columns 31 through 34. The number 1.0, (K), appears in rows 21 and 23 of the first array. Hence, 2.0 in column 32 is the sum of 0.0 and 2.0 in rows 21 and 23 of column 12; 1.0 is the sum of 1.0 and 0.0 in rows 21 and 23 of column 13; and 3.0 is the sum of 2.0 and 1.0 in rows 21 and 23 of column 14.

7.4 Properties Of An Array.

APROPERTIES, SAPROPERTIES

There are six different forms of APROPERTIES. Each instruction automatically prints 18 different properties of the designated array. The six forms differ only in the amount of information which is stored. The first form does not provide any storage. The remaining forms provide storage of results as follows: (2) properties, (3) column averages, (4) properties and column averages, (5) column averages and row averages, and (6) properties, column averages and row averages.

An example of the APROPERTIES, using the sixth form, is given on page 197. The printing of the properties shows the row in which the property is stored, the description and the value of the property. The meaning of most of the descriptions should be clear, but a few may require a word of explanation. The trace of an array (1) is the sum of the numbers in the principal diagonal; that is the sum of the numbers in the diagonal starting with the number in the upper left-hand corner and moving down and to the right. If the array is not square, the number of values used in the sum is given in parentheses. Item 14, the sum of squares about the mean, is the same as the (corrected) total sum of squares printed by the instruction TWOWAY, described in section 4.4. Items 15 and 16 are the within sums of squares and should not be confused with the between sums of squares printed by TWOWAY.

All forms of the instruction, except the first, have an additional form which has the letter S at the beginning of the command. The letter S indicates that the automatic printing of the properties is to be suppressed and only the requested results are stored.

These forms are listed at the end of this section, but they are not described. If an attempt is made to put the letter S at the beginning of the command in the first form, the instruction will be ignored and the following informative diagnostic will be given:

```
* COMMAND IGNORED - S BEFORE COMMAND NAME MEANINGLESS IF NO STORAGE REQUESTED
```

```
APROPERTIES of the array in (R),(C) of size (r)x(c)
```

This form of the instruction prints the 18 properties of the specified array, but provides no storage of results.

```
APROPERTIES of the array in (R),(C) of size (r)x(c) put in column (C)
```

The properties of the specified array are both printed and put in the first 18 rows of the column designated by the last argument.

```
APROPERTIES of array in (R),(C) size (r)x(c), put column ave's in (R),(C)
```

The (c) column averages are put in the row designated by the last pair of arguments.

```
APROPERTIES of (R),(C) size (r)x(c), put prop in (C), col ave's in (R),(C)
```

Same as above form, but the properties are also stored in the worksheet. The properties of the array are put in the first 18 rows of the column designated by the 5th argument and the column averages are put in the row designated by the last two arguments.

```
APROPERTIES of (R),(C) size (r)x(c), put col ave's in (R),(C), row ave's (R),(C)
```

The column averages are put in the row indicated by the third pair of arguments and the row averages are put in the column designated by the fourth (last) pair of arguments.

```
APROPERTIES array (R),(C) size (r)x(c), in (C), ave's in (R),(C) and (R),(C)
```

Same as above, but the properties are also stored in the worksheet. The properties of the array are put in the first 18 rows of the designated column (5th argument). The (c) column averages are put in a row vector designated by the second last pair of arguments. The (r) row averages are put in a column vector designated by the last pair of arguments. The example on page 197 uses this form of the instruction.

*****

```
SAPROPERTIES of the array in (R),(C) of size (r)x(c) put in column (C)
```

```
SAPROPERTIES of array in (R),(C) size (r)x(c), put column ave's in (R),(C)
```

PROPERTIES OF 4 X 5 ARRAY STARTING LOCATION (3, 7)

GENERAL

R		
1	TRACE (4 VALUES USED)	0.000000
2	TRACE NO. 2	-1.000000+02
3	MAXIMUM ELEMENT	1.000000+01
4	MINUMUM ELEMENT	-9.000000+00
5	MAXIMUM ELEMENT IN ABS VALUE	1.000000+01
6	MINUMUM ELEMENT IN ABS VALUE	0.000000
7	MIN NON-ZERO ELEM IN ABS VAL	1.000000+00
8	NUMBER OF POSITIVE ELEMENTS	10
9	NUMBER OF ZERO ELEMENTS	1
10	NUMBER OF NEGATIVE ELEMENTS	9
11	SUM OF TERMS	1.000000+01
12	AVERAGE	5.000000-01
13	SUM OF SQUARES	6.700000+02
14	SUM OF SQUARES ABOUT MEAN	6.650000+02
15	WITHIN ROWS SUM OF SQUARES	4.000000+01
16	WITHIN COLS SUM OF SQUARES	6.250000+02
17	SUM OF ABSOLUTE VALUES	1.000000+02
18	AVERAGE OF ABSOLUTE VALUES	5.000000+00

-9.0000000	-8.0000000	-7.0000000	-6.0000000	-5.0000000	-7.0000000
-4.0000000	-3.0000000	-2.0000000	-1.0000000	0.	-2.0000000
1.0000000	2.0000000	3.0000000	4.0000000	5.0000000	3.0000000
6.0000000	7.0000000	8.0000000	9.0000000	10.000000	8.0000000
-1.5000000	-.50000000	.50000000	1.5000000	2.5000000	0.

LIST OF COMMANDS, DATA AND DIAGNOSTICS

```
READ DATA INTO COLUMNS 1, 2, 3, 4 AND 5
-9 -8 -7 -6 -5
-4 -3 -2 -1  0
 1  2  3  4  5
 6  7  8  9 10
AMOVE 1,1 4,5 TO 3,7
APROPERTIES OF ARRAY IN 3,7 4X5 PUT COL AVE'S IN 7,7 ROW AVE'S IN 3,12
SPACE 2
APRINT ARRAY IN 3,7 OF SIZE 5X6
```

SAPROPERTIES of (R),(C) size (r)x(c), put prop in (C), col ave's in (R),(C)

SAPROPERTIES of (R),(C) size (r)x(c), put col ave's in (R),(C), row ave's (R),(C)

SAPROPERTIES array (R),(C) size (r)x(c), in (C), ave's in (R),(C) and (R),(C)

7.5 Printing.

APRINT, APRINT "L"

These two instructions are listed here for completeness, but they are described in section 1.8.

APRINT the array in (R),(C) of size (r)x(c)

APRINT "L" format, the array in (R),(C) of size (r)x(c)

7.6 Matrix Synonyms.

No knowledge of matrix algebra is required to use the instructions described above. Although the terms array and matrix are sometimes used interchangeably, they are often used in different contexts in OMNITAB. In fact, some array instructions perform quite different operations from the counterpart matrix instruction. An AMULTIPLY instruction simply performs element by element multiplication, whereas the MMULTIPLY instruction operates in accordance with the rules of matrix algebra. Nevertheless, there are several array operation instructions which are synonymous with a matrix operation instruction described in section 8. Each matrix equivalent of an array instruction is listed below. (See section B1.13 for a complete list of synonyms.)

Array Command(s)	Matrix Command(s)
AADD	MADD
ADEFINE	MDEFINE
AERASE (AZERO)	MERASE (MZERO)
AMOVE	MMOVE (also MOVE)
APRINT "L"	MPRINT "L"
ASUBTRACT	MSUBTRACT
ATRANSPOSE	MTRANSPOSE

8. MATRIX OPERATIONS.

This section describes instructions which perform operations on matrices occupying any location in the worksheet. All of the instructions operate independently of NRMAX; particularly MDIAGONAL, MMATVEC, MVECMAT, and M(AD). Each command (except SMPROPERTIES) begins with the letter M to denote a matrix operation. Except in MMATVEC, the first four arguments always specify the beginning location of a matrix, (a row and column number) and the size of the matrix (number of rows and columns in the matrix). The beginning location of a matrix is the location of the number in the upper left-hand corner of the matrix. Diagnostics are given if the specified matrix does not fit in the worksheet.

8.1 Defining Operations.

MDEFINE, MDIAGONAL, MERASE, MIDENTITY

These commands are used to define the elements of a matrix.

MDEFINE the matrix in (R),(C) of size (r)x(c) to have all elements equal to (K)

All the (r)x(c) elements in the matrix beginning in row (R) of column (C) of the worksheet are set equal to the constant (K). The constant (K) must be written with a decimal point. This instruction produces the same results as MERASE (or MZERO), below, if the constant (K) = 0.0. The instruction

MDEFINE the matrix in row 6 of col 3 size 2x3 to be 12.245

would give the following result:

Row/Column	3	4	5
6	12.245	12.245	12.245
7	12.245	12.245	12.245

MDIAGONAL the matrix in (R),(C) of size (r)x(c) equal to (E) on the diagonal

The main diagonal elements of the (r)x(c) matrix beginning in row (R) of column (C) are set equal to the specified constant or equal to the 1st (r) numbers (independent of NRMAX) in the column specified by the fifth argument. Only the main diagonal elements are changed. The matrix must be square, i.e., the 3rd and 4th arguments must be equal. The instruction

MDIAGONAL matrix in row 6 col 3 size 3x3 equal to 2.2 on the diagonal

would give the following results:

Row/Column	3	4	5
6	2.2		
7		2.2	
8			2.2

MERASE the matrix in (R),(C) of size (r)x(c)

Every number in the (r)x(c) matrix beginning in row (R) of column (C) is set equal to zero. MZERO is a synonym for MERASE. The instruction

MERASE the matrix in row 6 col 3 of size 2x3

would give the following result:

Row/Column	3	4	5
6	0.0	0.0	0.0
7	0.0	0.0	0.0

MIDENTITY in (R),(C) of size (r)x(c)

All the non-diagonal elements of the specified matrix in row (R) of column (C) of size (r)x(c) are set equal to zero. The main diagonal elements are set equal to one. If the size (r)x(c) does not specify a square matrix, the main diagonal elements of the square matrix, whose length is the smaller of (r) and (c), are set equal to one. The instruction

MIDENTITY matrix in row 6 col 3 of size 2x3

would give the following result:

Row/Column	3	4	5
6	1.0	0.0	0.0
7	0.0	1.0	0.0

8.2 Moving Operations.

MMATVEC, MMOVE, MTRANSPOSE, MVECDIAGONAL, MVECMAT

These commands are very useful for data manipulation.

MMATVEC make by rows column (C) into the matrix in (R),(C) of size (r)x(c)

The column (vector), specified by the 1st argument, is transformed into the matrix specified by the last four arguments. The first (c) numbers in column (C) become the 1st row of the matrix, the next (c) numbers become the 2nd row of the matrix, and so forth. The first (r)x(c) numbers in column (C) are used to construct the matrix (regardless of the value of NRMAX). Assume the numbers one through six are in column 21, then the instruction

MMATVEC make by rows col 21 into a matrix in location 6,3 size 2x3

would give the following result:

Row/Column	3	4	5
6	1.0	2.0	3.0
7	4.0	5.0	6.0

MMATVEC col vector in (R),(C) into matrix in (R),(C) size (r)x(c)

Same as the above instruction; except the column vector starts in row (R), rather than row 1, of column (C). This form of MMATVEC is equivalent to the one above, if the 1st argument equals one.

MMOVE the matrix in (R),(C) of size (r)x(c) to matrix in (R),(C)

Moves a matrix from one part of the worksheet to another. The new matrix may overlap the first matrix. The command MOVE, described in section 3.2, is synonomous with the command MMOVE. Assume the following numbers are in the worksheet:

Row/Column	1	2	3	4
2	1.0	2.0	3.0	0.0
3	4.0	5.0	6.0	0.0
4	7.0	8.0	9.0	0.0

The instruction

MMOVE the matrix in row 2 col 1 size 2x3 to matrix in row 3 col 2

would give the following result:

Row/Column	1	2	3	4
2	1.0	2.0	3.0	0.0
3	4.0	1.0	2.0	3.0
4	7.0	4.0	5.0	6.0

MTRANSPOSE the matrix in (R),(C) size (r)x(c) into matrix in (R),(C)

The transpose of the matrix beginning in row (R) of column (C) of size (r)x(c) is stored in the location designated by the last two arguments in the instruction. The first row of the original matrix becomes the first column of the new matrix, the second row becomes the second column, and so forth. The size of the transposed matrix will be (c)x(r). This instruction may be used to transpose a column into a row by setting (c)=1 or a row into a column by setting (r)=1. The instruction

MTRANSPOSE the matrix in row 6 col 2 size 2x3 into row 6 col 3

would give the following result:

	original matrix			transposed matrix	
Row/Column	2	3	4	3	4
6	2.0	5.0	3.0	2.0	8.0
7	8.0	9.0	10.0	5.0	9.0
				3.0	10.0

MVECDIAGONAL the matrix in (R),(C) size (r)x(c), put diagonal in column (C)

The elements on the main diagonal of the matrix beginning in row (R) of column (C) of size (r)x(c) are stored in the column specified by the last argument. If r≠c, and the matrix is not square, the number of values put in the designated column will equal the smaller of (r) and (c). If the following matrix is in the worksheet

Row/Column	2	3	4
6	2.0	5.0	3.0
7	8.0	9.0	10.0

then, the instruction

MVECDIAGONAL the matrix in row 6 col 2 size 2x3 into col 5

will put the following two values in column 5:

Row/Column	5
1	2.0
2	9.0

MVECDIAGONAL matrix in (R),(C) of size (r)x(c), put column vector in (R),(C)

This instruction is similar to the one above; except the diagonal elements of the matrix are put in the starting location designated by the last two arguments of the instruction. Using the data from the previous example, the instruction

MVECDIAGONAL the matrix in row 6 col 2 size 2x3 put in row 6 col 5

would give the following result:

Row/Column	5
6	2.0
7	9.0

MVECMAT vectorize row by row matrix in (R),(C) size (r)x(c) into column (C)

The elements in the matrix beginning in row (R) of column (C) of size (r)x(c) are put, as a column vector, in the rows of the column designated by the last argument. The 1st row of the matrix is put in the first (c) rows of the column, the 2nd row is put in the next (c) rows of the column and so forth. The number of values stored is (r)x(c), and NRMAX is not changed. The instruction

MVECMAT vectorize by rows matrix in row 6 col 2 size 2x3 into col 5

would create from the matrix on the left below, the column shown on the right:

	Matrix				Column
Row/Column	2	3	4	Row	5
6	2.0	5.0	3.0	1	2.0
7	8.0	9.0	10.0	2	5.0
				3	3.0
				4	8.0
				5	9.0
				6	10.0

MVECMAT matrix in (R),(C) size (r)x(c), put vector into (R),(C) and below

Same as the above instruction; except storage begins in row (R), instead of row 1, of column (C), as indicated by the last two arguments of the instruction. Using the previous matrix, the instruction

MVECMAT vectorize by rows matrix in row 6 col 2 size 2x3 into row 2 col 5

would give the following result:

Row/Column	5
2	2.0
3	5.0
4	3.0
5	8.0
6	9.0
7	10.0

8.3 Matrix Arithmetic.

MADD, MKRONECKER, MMULTIPLY, MRAISE, MSCALAR, MSUBTRACT

For the instructions in this section, define the matrix $A = (a_{ij})$, where a_{ij} is the element in the ith row and jth column. Other matrices, such as B and C, are defined in a similar manner.

MADD the matrix in (R),(C) size (r)x(c) to (R),(C) size (r)x(c) put in (R),(C)

This instruction computes the matrix sum A+B = C, where A and B are the matrices specified by the first eight arguments and the result is put in the location determined by the 9th and 10th (last two) arguments. The size of A must be the same as the size of B, i.e., the 4th and 5th arguments should equal the 7th and 8th arguments, respectively. The elements of C are

$$c_{ij} = a_{ij} + b_{ij}, \text{ where } i=1,2,\ldots,r \text{ and } j=1,2,\ldots,c.$$

Using the matrices A and B on the left below, the instruction

MADD matrix A in row 4, col 2 size 3x2 to B in 3,7 size 3x2, put C in 6,12

will compute the matrix C = A+B shown on the right:

Matrix A			Matrix B			Matrix C = A+B		
Row/Column	2	3	Row/Column	7	8	Row/Column	12	13
4	2.0	3.0	3	2.0	-5.0	6	4.0	-2.0
5	5.0	9.0	4	6.0	8.0	7	11.0	17.0
6	8.0	-2.0	5	-10.0	12.0	8	-2.0	10.0

MADD the matrix in (R),(C) of size (r)x(c) to matrix in (R),(C) put in (R),(C)

Same as the above instruction; except the arguments which specify the size of the second matrix (7th and 8th arguments above) are omitted. The instruction

MADD 4,2 size 3x2 to 3,7, put in 6,12

is equivalent to the one in the above example of MADD.

MKRONECKER product (R),(C) size (r)x(c) by (R),(C) size (r)x(c) put in (R),(C)

The Kronecker product $A \otimes B = C$ is computed, where the matrices A and B are specified by the first eight arguments and the result, C, is put in the location determined by the last two arguments. Let the size of A be mxn, 3rd and 4th arguments, and the size of B be rxs, 7th and 8th arguments. Then, the size of C is mrxns. The matrix C may be partioned as follows:

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}$$

Using the matrices A and B on the left below, the instruction

MKRONECKER product of A in 4,3 size 2x2 by B in 2,10 size 3x2, put C in 3,12

will compute the matrix C shown on the right:

Matrix A			Matrix B			Matrix C = A⊗B			
Row/Column	3	4	Row/Column	10	11	Row/Column	12	13	14	15
4	2.0	4.0	2	1.0	7.0	3	2.0	14.0	4.0	28.0
5	-3.0	0.0	3	3.0	-2.0	4	6.0	-4.0	12.0	-8.0
			4	1.0	0.0	5	2.0	0.0	4.0	0.0
						6	-3.0	-21.0	0.0	0.0
						7	-9.0	6.0	0.0	0.0
						8	-3.0	0.0	0.0	0.0

MMULTIPLY matrix (R),(C) size (r)x(c) by (R),(C) size (r)x(c) put in (R),(C)

Computes the matrix product AB = C, where A and B are the matrices specified by the first 8 arguments and the matrix C is stored in the location determined by the last two arguments. The number of columns of A, the 4th argument (c), must equal the number of rows of B, the 7th argument (r). A fatal error occurs if the 4th and 7th arguments are not equal. This constraint is necessary because matrix multiplication is not commutative, i.e., $AB \neq BA$, in general. In fact, if the product AB is defined, the product BA may not be defined. Let n equal both the 4th and 7th arguments, then the elements of the matrix C are:

$$c_{ij} = \sum_{u=1}^{n} a_{iu} b_{uj}, \text{ where } i=1,2,\ldots,r \text{ (3rd argument) and } j=1,2,\ldots,c \text{ (8th argument).}$$

The command MMULT is an acceptable abbreviation for MMULTIPLY. Using the matrices A and B on the left below, the instruction

MMULTIPLY matrix A in 4,3 size 2x2 by B in 2,10 size 2x3, put C in 6,20

will compute the matrix C = AB shown on the right:

Matrix A			Matrix B				Matrix C = AB			
Row/Column	3	4	Row/Column	10	11	12	Row/Column	20	21	22
4	2.0	6.0	2	6.0	-1.0	7.0	6	30.0	-2.0	26.0
5	3.0	-2.0	3	3.0	0.0	2.0	7	12.0	-3.0	17.0

In computing C, each individual product is stored as a double precision number. The products are then sorted in ascending order. The summation, which is also performed in double precision arithmetic, begins with the mid-point value(s) and proceeds from there. For example, suppose the sorted products were -4, -2, 1, 2 and 4, then the summation would be performed as follows: 1 + (-2) + 2 + (-4) +4. This method of computation is used to provide greater accuracy. For large matrices it will increase the computing time appreciably.

MRAISE the matrix in (R),(C) of size (r)x(c) to power (K) put in (R),(C)

The matrix specified by the first four arguments is multiplied by itself (K) times, where (K) is the 5th argument and a constant. If (K) is not an integer, it is truncated to an integer and no diagnostic is printed. ((K) can be written without a decimal point.) If the value of (K) is zero, the indentity matrix is computed. The matrix must be square, i.e., the number of rows (r) must equal the number of columns (c). Using the matrix A on the left below, the instruction

MRAISE matrix A in row 4 col 3 of size 2x2 to 3.0 power, put in 6,20

will compute the result Y shown on the right:

Matrix A			Matrix $Y = A^k$		
Row/Column	3	4	Row/Column	20	21
4	2.0	6.0	6	44.0	132.0
5	3.0	-2.0	7	66.0	-44.0

MSCALAR matrix (R),(C) of size (r)x(c) by constant (K), put matrix in (R),(C)

The scalar product Y =sA is computed, where A is the first matrix and s is the scalar (constant) (K). The matrices Y and A have the same size, (r)x(c). The elements of Y are

$$y_{ij} = sa_{ij}, \text{ where } i=1,2,\ldots,r \text{ and } j=1,2,\ldots,c.$$

MSCALAR is actually a special case of AMULTIPLY (section 7.1). The instruction

MSCALAR matrix A in 4,3 size 2x2 by -2.0, put Y in 6,20

using the matrix A on the left below, will compute the result Y = sA shown on the right:

Matrix A			Matrix Y = sA		
Row/Column	3	4	Row/Column	20	21
4	2.0	6.0	6	-4.0	-12.0
5	3.0	-2.0	7	-6.0	4.0

MSUBTRACT mat (R),(C) size (r)x(c) minus mat (R),(C) size (r)x(c) into (R),(C)

The matrix subtraction A-B = C is computed, with A the 1st matrix, B the 2nd matrix and C the 3rd matrix. The size of B must agree with the size of A. The matrices A, B and C all have the same dimensions, (r)x(c). The elements of C are

$$c_{ij} = a_{ij} - b_{ij}, \text{ where } i=1,2,\ldots,r \text{ and } j=1,2,\ldots,c.$$

The command MSUB may be used as an abbreviation for MSUBTRACT. Note, the order of subtraction is the reverse of that in SUBTRACT described in section 2.1. Using the matrices A and B on the left below, the instruction

MSUBTRACT matrix A in 4,2 size 3x2, minus B in 3,7 size 3x2, put C in 6,12

will compute C = A-B shown on the right:

Matrix A			Matrix B			Matrix C = A-B		
Row/Column	2	3	Row/Column	7	8	Row/Column	12	13
4	2.0	3.0	3	2.0	-5.0	6	0.0	8.0
5	5.0	9.0	4	6.0	8.0	7	-1.0	1.0
6	8.0	-2.0	5	-10.0	12.0	8	18.0	-14.0

MSUBTRACT matrix (R),(C) size (r)x(c) minus mat (R),(C) put into (R),(C)

Same as the above instruction; except the size of the second matrix is not specified. The size of the second matrix is taken to be the same as the size of the first matrix.

8.4 Special Matrix Multiplication.

M(AD), M(AV), M(DA), M(V'A), M(X'X), M(XX'), M(X'AX), M(XAX')

Each of the commands in this section contain right and left parentheses. Several of the commands contain an apostrophe to denote the transpose of a matrix (or vector). These are the only commands in the OMNITAB II system which use either parentheses or an apostrophe.

Each instruction performs a special form of matrix multiplication. In each case, the result of the multiplication is a matrix (or vector) denoted by Y with elements y_{ij}. The notation $X = (x_{ij})$, $A = (a_{ij})$, etc. will be used to denote a matrix X with elements x_{ij} in

the ith row and jth column, a matrix A with elements a_{ij} in the ith row and jth column, etc.. The basic definition of matrix multiplication is given in section 8.3.

M(AD) mat (R),(C) size (r)x(c) times mat with (C) in diag, put mat in (R),(C)

The first matrix A is postmultiplied by a diagonal matrix, D, whose main diagonal elements are the 1st (c) numbers (independent of NRMAX) in the column (C), designated by the 5th argument. The matrix D is not actually stored in the worksheet. The result is the matrix Y = AD, with elements

$$y_{ij} = d_j a_{ij}, \text{ where } i=1,2,\ldots,r \text{ and } j=1,2,\ldots,c.$$

The matrices A and Y have the same dimensions. The instruction

M(AD) matrix A in 4,2 size 3x2 times diagonal in col 4, put Y in 3,7

used with the data on the left below, will compute the matrix Y = AD shown on the right:

Matrix A			Diagonal of D		Matrix Y = AD		
Row/Column	2	3	Row/Column	4	Row/Column	7	8
4	2.0	3.0	1	2.0	3	4.0	-9.0
5	5.0	9.0	2	-3.0	4	10.0	-27.0
6	8.0	-2.0			5	16.0	6.0

M(AV) mat (R),(C) size (r)x(c) by vector in col (C) put vector in col (C)

The first matrix, A, is postmultiplied by the vector V in column (C) to form the column vector Y = AV, with elements

$$y_i = \sum_{u=1}^{c} v_u a_{iu}, \text{ where } i=1,2,\ldots,r.$$

The column vector Y is put in the first (r) rows of column (C). The instruction

M(AV) matrix A in 4,2 size 3x2 by vector in col 4, put Y in column 7

using the matrix A and the vector V on the right below, will compute the result Y = AV shown on the right:

Matrix A			Vector V		Vector Y = AV	
Row/Column	2	3	Row/Column	4	Row/Column	7
4	2.0	3.0	1	2.0	1	-5.0
5	5.0	9.0	2	-3.0	2	-17.0
6	8.0	-2.0			3	22.0

M(AV) mat (R),(C) size (r)x(c) by column (C) put vector in row (R) of col (C)

Same as the above instruction; except the storage of the column vector Y = AV begins in the designated row (R) of column (C), rather than the 1st row of column (C), where (R) and (C) are the 6th and 7th (last two) arguments.

M(DA) mat (R),(C) size (r)x(c) premult by mat with (C) in diag, put in (R),(C)

The first matrix, A, is premultiplied by a diagonal matrix, D, whose main diagonal elements are in column (C), designated by the 5th argument. The matrix D is not actually stored in the worksheet. The result is Y = DA, with elements

$$y_{ij} = d_i a_{ij}, \text{ where } i=1,2,\ldots,r \text{ and } j=1,2,\ldots,c.$$

Y has the same dimensions as A. The instruction

M(DA) matrix A in 4,2 size 3x2, premultiplied by col 4, put Y in 3,7

used with the data on the left below, will compute the result shown on the right:

Diagonal of D		Matrix A			Matrix Y = DA		
Row/Column	4	Row/Column	2	3	Row/Column	7	8
1	2.0	4	2.0	3.0	3	4.0	6.0
2	-3.0	5	5.0	9.0	4	-15.0	-27.0
3	1.0	6	8.0	-2.0	5	8.0	-2.0

M(V'A) mat (R),(C) size (r)x(c), vector in col (C), put row vector in row (R)

The matrix A is premultiplied by the transpose of a column vector, V, to form the row vector Y = V'A, with elements

$$y_j = \sum_{u=1}^{r} v_u a_{uj}, \text{ where } j = 1,2,\ldots,c.$$

The result is put in row (R), 6th argument, of the first (c) columns. The instruction

M(V'A) matrix A in 4,2 size 3x2 by vector in col 4 put Y in row 7

would give the following results:

Column Vector V		Matrix A			Row Vector Y = V'A		
Row/Column	4	Row/Column	2	3	Row/Column	1	2
1	2.0	4	2.0	3.0	7	-3.0	-23.0
2	-3.0	5	5.0	9.0			
3	1.0	6	8.0	-2.0			

M(V'A) mat (R),(C) size (r)x(c), vector in col (C), put row vector in (R),(C)

Same as above instruction; except storage of the row vector begins in the designated column, rather than in column 1. Using the data from the example above, the instruction

M(V'A) matrix A in 4,2 size 3x2 by vector in col 4 put Y in row 7 of col 3 on

would give the following results:

Column Vector V		Matrix A			Row Vector Y = V'A		
Row/Column	4	Row/Column	2	3	Row/Column	3	4
1	2.0	4	2.0	3.0	7	-3.0	-23.0
2	-3.0	5	5.0	9.0			
3	1.0	6	8.0	-2.0			

M(X'X) x-matrix in (R),(C) of size (r)x(c) into (R),(C)

The matrix X is premultiplied by its transpose to produce Y = X'X, with elements

$$y_{ij} = \sum_{u=1}^{r} x_{ui} x_{uj}, \text{ where } i=1,2...c \text{ and } j=1,2..c.$$

The size of Y is (c)x(c). The instruction

M(X'X) matrix in 4,2 size 3x2 put Y in 3,7

applied to the matrix X on the left below, will compute the matrix Y = X'X shown on the right:

Matrix X			Matrix Y = X'X		
Row/Column	2	3	Row/Column	7	8
4	2.0	3.0	3	93.0	35.0
5	5.0	9.0	4	35.0	94.0
6	8.0	-2.0			

M(XX') x-matrix in (R),(C) of size (r)x(c) into (R),(C)

Similar to the above instruction; except the matrix X is postmultiplied by its transpose. The result is Y = XX' with elements

$$y_{ij} = \sum_{u=1}^{c} x_{iu} x_{ju}, \text{ where } i=1,2,...,r \text{ and } j=1,2,...,r.$$

The size of Y is (r)x(r). The instruction

M(XX') matrix in 4,2 size 3x2 put Y in 3,7

using the matrix X on the left below, will compute the result Y = XX' on the right:

Matrix X			Matrix Y = XX'			
Row/Column	2	3	Row/Column	7	8	9
4	2.0	3.0	3	13.0	37.0	10.0
5	5.0	9.0	4	37.0	106.0	22.0
6	8.0	-2.0	5	10.0	22.0	68.0

M(X'AX) a-mat in (R),(C) size (r)x(c), x-mat (R),(C) size (r)x(c), into (R),(C)

The transformation Y = X'AX is computed. A is the first matrix specified, X is the second matrix specified and the results are stored in the third matrix, Y. Matrix A is

square and of size nxn, specified by the 3rd and 4th arguments. Matrix X has size nxm, specified by the 7th and 8th arguments. The number of rows of X, 7th argument, must equal the number of rows and columns of A, 3rd and 4th arguments. The matrix Y is square of size mxm. Let a_{ij} be the elements of A, x_{ij} be the elements of X and y_{ij} be the elements of the result Y. Then,

$$y_{ij} = \sum_{u=1}^{n}\sum_{v=1}^{n} a_{uv}x_{ui}x_{vj}, \text{ where } i=1,2,\ldots,m \text{ and } j=1,2,\ldots,m \text{ (8th argument).}$$

If m=1, Y is a quadratic form. For the matrices A and X on the left below, the instruction

M(X'AX) matrix A in 2,3 size 2x2, X in 3,5 size 2x3, put Y in 2,8

will compute the result Y = X'AX shown on the right:

Matrix A			Matrix X				Matrix Y = X'AX			
Row/Column	3	4	Row/Column	5	6	7	Row/Column	8	9	10
2	1.0	3.0	3	-2.0	0.0	-2.0	2	10.0	-10.0	8.0
3	2.0	-4.0	4	-1.0	5.0	0.0	3	0.0	-100.0	-20.0
							4	10.0	-30.0	4.0

M(XAX') a-mat in (R),(C) size (r)x(c), x-mat (R),(C) size (r)x(c), into (R),(C)

The transformation Y = XAX' is computed. A is the first matrix specified, X is the second matrix specified and the results are stored in the third matrix, Y. Matrix A is square and of size nxn, specified by the 3rd and 4th arguments. Matrix X has size mxn, specified by the 7th and 8th arguments. The number of columns of X, 8th argument, must equal the number of rows and columns of A, 3rd and 4th arguments. The matrix Y is square of size mxm. Let a_{ij} be the elements of A, x_{ij} be the elements of X and y_{ij} be the elements of the result Y. Then,

$$y_{ij} = \sum_{u=1}^{n}\sum_{v=1}^{n} a_{uv}x_{iu}x_{jv}, \text{ where } i=1,2,\ldots,m \text{ and } j=1,2,\ldots,m \text{ (7th argument).}$$

For the matrices A and X on the left below, the instruction

M(XAX') matrix A in 2,3 size 2x2, X in 3,5 size 3x2, put Y in 2,8

will compute the matrix Y = XAX' shown on the right:

Matrix A			Matrix X			Matrix Y = XAX'			
Row/Column	3	4	Row/Column	5	6	Row/Column	8	9	10
2	1.0	3.0	3	-2.0	0.0	2	4.0	-28.0	-12.0
3	2.0	-4.0	4	-1.0	5.0	3	-18.0	-124.0	4.0
			5	3.0	1.0	4	-10.0	20.0	20.0

8.5 Matrix Analysis.

MEIGEN, MINVERT, MORTHO, MTRIANGULARIZE

The four instructions described here are useful in matrix analysis. There are three forms of MEIGEN which compute (i) only the eigenvalues, (ii) only the eigenvectors, and (iii) both the eigenvalues and eigenvectors. An example is given for the last form. The

error messages which can be printed by any of the three forms of MEIGEN are only listed under the first form. The computations performed by MORTHO and MTRIANGULARIZE are related to the computations performed by the FIT instruction described in section 4.5.

MEIGEN of the matrix in (R),(C) of size (r)x(c), put eigenvalues in column (C)

For the matrix A designated by the first four arguments, the instruction computes the eigenvalues (characteristic roots or latent roots) of the matrix A and puts them in descending order in the column designated by the 5th (last) argument. The matrix A must be square and symmetric. The eigenvalues, λ_i, are the roots of the characteristic polynomial $|A-\lambda I| = 0$. The Jacobi method is used to compute the eigenvalues.

If A is not square, 3rd and 4th arguments equal, the following informative diagnostic is printed:

* NO OF ROWS NOT = TO COLS. MATRIX USED LARGEST SQUARE

If (r)x(c) exceeds 6,750+2(r) (in NBS computer), the following fatal error message is printed:

*** INSUFFICIENT SCRATCH AREA

If the matrix A is not symmetric, the following fatal error mesage is printed:

*** MATRIX IS NOT SYMMETRIC

MEIGEN of matrix in (R),(C) size (r)x(c), put eigenvectors in (R),(C)

This form of MEIGEN computes the eigenvectors, but not the eigenvalues. For the matrix A and eigenvalues λ_i, the eigenvectors, x_i, satisfy the relation $Ax_i = \lambda_i x_i$, where i = 1,2,...,r=c. The eigenvectors are stored as a matrix as determined by the 5th and 6th (last two) arguments. The eigenvectors are stored as column vectors according to the descending order of the eigenvalues, λ_i. Each vector is normalized so that the sum of squares of the elements is unity. Thus, except for sign, each vector is unique. When two eigenvalues are equal, the corresponding eigenvectors will be orthogonal.

MEIGEN matrix in (R),(C) size (r)x(c), put values in (C) vectors in (R),(C)

Both the eigenvalues and eigenvectors of the symmetric matrix, designated by the first four arguments, are computed and stored. The (r)=(c) eigenvalues are put in the column designated by the 5th argument and the eigenvectors are stored as a matrix in the location determined by the 6th and 7th (last two) arguments. Using the following matrix, A:

Row/Column	3	4	5	6
2	4.0	4.0	-2.0	-6.0
3	-2.0	-6.0	4.0	4.0
4	-6.0	-2.0	4.0	4.0
5	4.0	4.0	-6.0	-2.0

the instruction

MEIGEN matrix A in row 2, col 3 size 4x4, put values in col 7, vectors in 2,8

would compute the four eigenvalues and eigenvectors of A and put them in the worksheet as follows:

Eigenvalues		Eigenvectors			
Row/Column 7		Row/Column 8	9	10	11
1	2.5600000+02	2	.50000000	-.53154530	-.46632566	.49999999
2	16.000000	3	.50000001	.53154529	.46632564	.50000002
3	16.000000	4	-.50000002	-.46632566	.53154528	.50000000
4	-1.1324343-06	5	-.50000001	.46632564	-.53154529	.49999999

MINVERT the matrix in (R),(C) of size (r)x(c), put inverse in (R),(C)

MINVERT computes the inverse, A^{-1}, of the matrix A, such that $AA^{-1} = I$. The 1st four arguments specify the location and size of A and the 5th and 6th (last two) arguments determine the location of the inverse of A, A^{-1}. The commands INVERT and MINVERT are synonomous. An example of the use of MINVERT to invert a (scaled) Hilbert matrix is shown on page 213. (The scaling factor 60.0 was used to make each element an integer.)

The matrix A must be square so that the 3rd and 4th arguments, (r) and (c), must be equal. If r≠c, the following informative diagnostic is printed:

* NO OF ROWS NOT = TO COLS. MATRIX USED IS LARGEST SQUARE.

If r and c are inadvertently too large, the following fatal error occurs:

*** DEFINED MATRIX OVERFLOWS WORKSHEET

In order to invert a matrix, a large amount of space (scratch area) is required in the computer. If (r) exceeds 77, or $2(r+2)^2$ is greater than the size of the worksheet, 12,500, the following fatal error message is printed:

*** INSUFFICIENT SCRATCH AREA

Finally, if the matrix, A, is singular, the following fatal error message is printed:

*** MATRIX IS (NEARLY) SINGULAR

The inversion algorithm used was written by A. R. Sadaka and submitted to the SHARE library. Elementary row operations on the input matrix, A, transform it into the unit matrix of the same magnitude. The matrix of these transformations is the inverse. Selective pivoting is used to minimize truncation error accumulation. Further description of the SHARE subroutines may be obtained in SHARE write-ups 3180, and 3181, Catalog of Programs for IBM 704-709-7040-7090 and 7094 Data Processing Systems, IBM, File No. 7040/7090-20. This algorithm enables error bounds to be computed.

In the LIST OF COMMANDS, DATA AND DIAGNOSTICS, an error bound is printed immediately below the command MINVERT, if an inverse has been obtained. The form of the error bound, as seen on page 213, is

++++++++++++++++++++ SMALLEST ERROR BOUND ON INVERTED MATRIX IS .1-04 ++++

OMNITAB EXAMPLE OF MINVERT USING 3X3 HILBERT MATRIX

3X3 HILBERT MATRIX

ROW/COL	1	2	3
1	60.000000	30.000000	20.000000
2	30.000000	20.000000	15.000000
3	20.000000	15.000000	12.000000

INVERSE OF HILBERT MATRIX

ROW/COL	4	5	6
1	.15000004	-.60000020	.50000018
2	-.60000024	3.2000012	-3.0000011
3	.50000025	-3.0000013	3.0000012

EXACT INVERSE OF HILBERT MATRIX

7	8	9
.15000000	-.59999999	.50000000
-.59999999	3.2000000	-3.0000000
.50000000	-3.0000000	3.0000000

OMNITAB EXAMPLE OF MINVERT USING 3X3 HILBERT MATRIX

LIST OF COMMANDS, DATA AND DIAGNOSTICS

```
READ   1      2      3
      60     30     20
      30     20     15
      20     15     12
       9    -36     30
     -36    192   -180
      30   -180    180
MINVERT 1,1 SIZE 3X3 PUT IN 1,4
  ++++++++++++++++++++ SMALLEST ERROR BOUND ON INVERTED MATRIX IS   .1-04   ++++
ADIVIDE 4,1 SIZE 3X3 BY 60. PUT IN 1,7
NOTE  3X3 HILBERT MATRIX
MPRINT 1,1 SIZE 3X3
SPACE
NOTE1  INVERSE OF HILBERT MATRIX
NOTE2  EXACT INVERSE OF HILBERT MATRIX
PRINT NOTE
MPRINT 1,4 SIZE 3X6
```

NATIONAL BUREAU OF STANDARDS, WASHINGTON, D. C. 20234, OMNITAB II VERSION 5.00 MAY 15, 1970

Let A be the matrix to be inverted and X be the result of the inversion. Define Y = I-AX. Furthermore, for any square matrix Z = (z_{ij}), size nxn, let N(Z) be a norm of Z defined in any of the following three ways:

$$(1) \quad N(Z) = [\sum_{i,j}^{n} |z_{ij}|^2]^{\frac{1}{2}}$$

$$(2) \quad N(Z) = n \max_{i,j} |z_{ij}|$$

$$(3) \quad N(Z) = \max_{i} \sum_{j=1}^{n} |z_{ij}|.$$

In order to guarantee that X be a good approximation to A^{-1}, it is necessary to have ERR = N(X)N(Y)/[1-N(Y)] be positive and small. The instruction MINVERT computes ERR for each of the norms defined above and the smallest one is printed as the error bound. A description of error checking methods may be found in Chapter 6 of "Survey of Numerical Analysis," J. Todd, Ed., McGraw-Hill (1962). Note, AX being close to I does not guarantee that X is close to A^{-1}.

The error bound is a very conservative estimate of the error in computing the inverse. In the example on page 213, the error bound indicates that the results of the inversion are accurate to about 5 significant digits, whereas, in reality, the results are correct to 6 significant digits.

MORTHO mat (R),(C) size (r)x(c) weights (E) put orthonormal vectors in (R),(C)

The Gram-Schmidt orthonormalization process is applied to the matrix, X, designated by the first four arguments, using the weights in the first (r) rows of the column designated by the 5th argument. The orthonormal vectors are stored as a (r)x(c) matrix as indicated by the 6th and 7th (last two) arguments. See "Orthonormalizing codes in numerical analysis.", P. J. Davis, Chapter 10 of "Survey of Numerical Analysis," J. Todd, Ed., McGraw-Hill (1962).

All weights must be positive. Otherwise, a fatal error occurs. If r<c, the instruction is not executed and a fatal error message is given. If the last two arguments do not specify enough area in the worksheet to store the orthonormal vectors, the following informative diagnostic is given:

* PARTIAL STORAGE OF MATRIX

For the following matrix, X, in 1,1 and weights in column 10:

Row/Column	1	2	3	4	Column 10
1	1.0	1.0	2.0	2.0	2.0
2	1.0	2.0	2.0	3.0	2.0
3	0.0	3.0	3.0	3.0	1.0
4	0.0	2.0	1.0	1.0	1.0
5	0.0	1.0	-3.0	0.0	1.0
6	0.0	1.0	0.0	0.0	1.0
7	0.0	0.0	1.0	0.0	1.0

the instruction

MORTHO matrix X in 1,1 size 7x4 with weights in col 10, put in 11,41

would compute the transformation Φ = XA, where A is an upper triangular matrix. Shown below is the matrix 16.Φ, rather than Φ, for ease in reading:

Row/Column	41	42	43	44
11	8.0	-2.0	1.0	-3.0
12	8.0	2.0	-1.0	3.0
13	0.0	12.0	6.0	6.0
14	0.0	8.0	0.0	-8.0
15	0.0	4.0	-14.0	2.0
16	0.0	4.0	-2.0	-10.0
17	0.0	0.0	4.0	-4.0

In a FIT instruction (section 4.5), $\Phi = XA$ is computed from the set of vectors X. The matrices A and X are related by $X'WX = (A^{-1})'(A^{-1})$, where W is a diagonal matrix with the column of weights on the diagonal.

```
/------------------------------------------------------------------------\
/ MORTHO mat (R),(C) size (r)x(c) weights (E) put in (R),(C) trans mat (R),(C) /
/------------------------------------------------------------------------/
```

This form of MORTHO is the same as the one above; except in the transformation $\Phi = XA$, the transformation matrix, A, is also stored in the worksheet. The matrix X is designated by the first four arguments, the weights by the 5th argument, the location of the orthornormal matrix, Φ, by the 6th and 7th arguments and the location of the transformation matrix, A, by the 8th and 9th (last two) arguments. The matrix A is upper triangular. Using the matrix X defined in the example above, the instruction

MORTHO X in 1,1 size 7x4 wts in col 10, put orthonormal mat in 11,41 and trans in 21,41

would locate the transformation matrix, A, in the worksheet as shown below. For ease in reading, the matrix shown is actually 16.A.

Row/Column	41	42	43	44
11	8.0	-6.0	-5.0	-17.0
12	0.0	4.0	-2.0	-10.0
13	0.0	0.0	4.0	-4.0
14	0.0	0.0	0.0	16.0

See the description of the second form of MTRIANGULARIZE, below, for further remarks.

```
:------------------------------------------------------------------------:
: MTRIANGULARIZE the matrix in (R),(C) size (r)x(c) into matrix in (R),(C) :
:------------------------------------------------------------------------:
```

A lower triangular matrix, T, is computed for the matrix, A, such that $A = TT'$. The triangularization is performed only for symmetric matrices of full rank with leading submatrices nonsingular. The matrix, A, to be triangularized is designated by the first four arguments. A should be square, so the 3rd and 4th arguments should be equal. The triangular matrix, T, is located in the worksheet as determined by the 5th and 6th (last two) arguments. The size of T is (r)x(r) and all the elements above the principal diagonal are equal to zero. Page 216 contains the results and a set of instructions to illustrate the use of both forms of the instruction MTRIANGULARIZE.

The fatal error

*** ILLEGAL *STATEMENT*

will occur if any of the following three conditions exist:

(1) $r \neq c$
(2) Any leading submatrix is singular.
(3) A is not symmetric, $|\,|a_{ij}/a_{ji}|-1| > 10^{-6}$.

THE FOLLOWING IS AN EXAMPLE OF THE MTRIAN COMMAND.

MATRIX A IS THE ORIGINAL MATRIX.

ROW/COL	2	3	4	5
3	4.0000	6.0000	8.0000	10.0000
4	6.0000	25.0000	20.0000	27.0000
5	8.0000	20.0000	36.0000	30.0000
6	10.0000	27.0000	30.0000	36.0000

MATRIX T IS THE TRIANGULAR OF MATRIX A (T X T-TRANSPOSE = A)

ROW/COL	7	8	9	10
3	2.0000	.0000	.0000	.0000
4	3.0000	4.0000	.0000	.0000
5	4.0000	2.0000	4.0000	.0000
6	5.0000	3.0000	1.0000	1.0000

MATRIX C IS THE INVERSE OF MATRIX T

ROW/COL	11	12	13	14
3	.5000	.0000	.0000	.0000
4	-.3750	.2500	.0000	.0000
5	-.3125	-.1250	.2500	.0000
6	-1.0625	-.6250	-.2500	1.0000

LIST OF COMMANDS, DATA AND DIAGNOSTICS

```
READ THE FOLLOWING DATA INTO COLUMNS 2***5
 0.0     0.0      0.0    0.0
 0.0     0.0      0.0    0.0
 4.0     6.0      8.0   10.0
 6.0    25.0     20.0   27.0
 8.0    20.0     36.0   30.0
10.0    27.0     30.0   36.0
MTRIAN MATRIX IN R=3 C=2 SIZE=4X4 PUT TRIANGULAR IN R=3 C=7
TITLE1 THE FOLLOWING IS AN EXAMPLE OF THE MTRIAN COMMAND.
NEW PAGE
SPACE
NOTE MATRIX A IS THE ORIGINAL MATRIX.
FIXED 4
MPRINT MATRIX A IN R=3 C=2  SIZE=4X4
SPACE
NOTE MATRIX T IS THE TRIANGULAR OF MATRIX A (T X T-TRANSPOSE = A)
MPRINT MATRIX B IN R=3 C=7 SIZE=4X4
MTRIAN MATRIX IN 3,2 SIZE=4X4 PUT TRIANGULAR IN 3,7 INVERSE IN 3,11
SPACE
NOTE MATRIX C IS THE INVERSE OF MATRIX T
MPRINT MATRIX C IN R=3 C=11 SIZE=4X4
STOP
```

Formulas for performing the triangularization are described, starting on page 6-38, in "Experimental Statistics," Natrella, M. G., National Bureau of Standards Handbook 91, U. S. Government Printing Office.

```
MTRIANGULARIZE matrix (R),(C) size (r)x(c) in (R),(C) inverse in (R),(C)
```

This form of the instruction is the same as the one above; except the inverse of the triangular matrix is also computed. The inverse is put in the location in the worksheet determined by the 7th and 8th (last two) arguments.

In order to invert a matrix, a large amount of scratch area (internal space) is needed for the computations. If (r) exceeds 77, or $2(r+2)^2$ is greater than the size of the worksheet, the following fatal error occurs:

*** INSUFFICIENT SCRATCH AREA

An example of the use of this form of MTRIANGULARIZE is also shown on page 216. The matrix A in 3,2 is equal to the matrix X'WX, where the matrices X and W are defined under MORTHO on page 214. Hence, X'WX = TT'. W is a diagonal matrix with weights on the principal diagonal. The transpose of the matrix C, equal to T inverse, in 3,11 on page 216, is equal to the transpose of the transformation matrix, A, where 16.A is in 11,41 shown on page 215.

Remark

In a weighted least squares analysis, see section 4.5, estimates of β are obtained for the model $y = X\beta + e$. The estimates are computed from $\hat{\beta} = A\Phi'Wy$. The matrix A is the inverse of T transpose when MTRIANGULARIZE is applied to X'WX. The matrix A is also the transformation matrix computed when MORTHO is applied to the matrix X. The matrix Φ can be obtained by applying MORTHO to the design matrix X. The Fourier coefficients, described in section 4.5, are given by $\Phi'Wy$. It is instructive to verify the above statements using the commands FIT, MORTHO and MTRIANGULARIZE. This can be done using the measurements y = 13.0, 17.0, 18.2, 8.8, -3.0, 2.8 and 2.1 and the X matrix and weights given on page 214.

8.6 Properties.

MPROPERTIES, SMPROPERTIES

The instructions MPROPERTIES and SMPROPERTIES evaluate properties of the matrix specified by the first four arguments. If the matrix is square (e.g., r=c), 31 different properties of the matrix will be computed, while 20 properties are computed if the matrix is not square. An MPROPERTIES instruction automatically prints the different properties of the designated matrix, whereas SMPROPERTIES stores results, but suppresses the automatic printing. The first 18 properties computed by MPROPERTIES are the same as those computed by APROPERTIES, described in section 7.4.

There are six different forms of MPROPERTIES. The six forms differ only in the amount of information which is stored. The first form does not provide any storage. The remaining forms provide storage of results as follows: (2) properties, (3) column averages, (4) properties and column averages, (5) column averages and row averages, and (6) properties, column averages and row averages.

The printing of the properties shows the row in which the property is stored, if storage is requested, the description of the property and the value of the property. The meaning of most of the descriptions should be clear, but some may require a word of explanation, which is given below. Items 1 through 18, 28 and 29 are always printed, for both square and non-square matrices.

Pages 219 and 220 show the automatic printing of MPROPERTIES for both a square matrix and a non-square matrix using the following set of instructions. (A few inconsequential changes were made in the actual computer printing in order to make it fit on a page.)

```
OMNITAB- EXAMPLE OF MPROPERTIES
    READ 1 2 3 4
      4  6  8 10
      6 25 20 27
      8 20 36 30
     10 27 30 36
    MPROPERTIES  MATRIX IN 1,1 SIZE 4X4  STORE PROPERTIES IN 10
    MPROPERTIES MATRIX IN 1,1 SIZE 3X4
    STOP
```

The trace of a matrix, item 1, is the sum of the numbers in the principal diagonal; that is the sum of the numbers in the diagonal starting with the number in the upper left-hand corner and moving down and to the right. The number of values used in the sum is printed in parentheses. Item 2, trace no. 2, is evaluated using the formula

$$\sum_{k=2}^{n}\sum_{i=1}^{k-1}(a_{ii}a_{kk} - a_{ki}a_{ik}), \text{ where n is the smaller of r and c.}$$

Trace no. 2 is the second elementary symmetric function of the eigenvalues and also the second coefficient in the characteristic polynomial.

Item 14, the sum of squares about the mean, is the same as the (corrected) total sum of squares printed by the instruction TWOWAY, described in section 4.4. Items 15 and 16 are the within sums of squares and should not be confused with the between sums of squares printed by TWOWAY.

A square matrix is orthogonal if the matrix premultiplied by its transpose is equal to the identity matrix (e.g., A'A = I). For a complete discussion of matrices, see "Theory of Matrices," S. Perlis, Addison-Wesley (1952). MPROPERTIES also checks the orthogonality of non-square matrices by using the largest square matrix contained in the specified rectangular matrix. If r> c, the condition to be satisfied is A'A = I, otherwise, for r<c, the condition is AA' = I. In the automatic printing, item 28, the answer will be no, if the matrix is not orthogonal and yes, if the matrix is orthogonal. Enclosed in parentheses, after the answer, is the value stored (if storage is requested) in row 28. The value will be zero if the matrix is non-orthogonal, one if the matrix is exactly orthogonal (A'A (or AA') = I, exactly) and two if A'A = I within a tolerance. For the tolerance to be satisfied, each number in A'A must be zero or one plus or minus $1x10^{-7}$. For non-square matrices, the automatic printing will indicate row or column wise orthogonality. Item 29 is similar to item 28; except the product A'A (or AA') is examined to see if it is a diagonal matrix, rather than the identity matrix.

For square matrices, items 19 through 27, 30 and 31 are also computed. The norms, items 21, 22 and 23, are error bounds on the inverse of the matrix. See the discussion of MINVERT in section 8.5 for further details.

A matrix will be a normal matrix, item 24, if it is symmetric, item 25, skew symmetric, item 26, or diagonal, item 27. A matrix, A, is symmetric if A = A'. A skew symmetric matrix is one where A = -A' and all the principal diagonal elements equal zero. A matrix is diagonal, if all the off-diagonal elements equal zero (e.g., $a_{ij} = 0$, if $i \neq j$). The automatic printing will show YES if condition is true and NO if the condition is not true. The value inside the parentheses is the value stored, if storage has been requested. Zero indicates the condition has not been met, one indicates the condition has been met exactly and two indicates the condition is true within the tolerance $1x10^{-7}$.

PROPERTIES OF 4 X 4 MATRIX STARTING LOCATION (1, 1)

COL	10
	GENERAL
R	
1 TRACE (4 VALUES USED)	1.010000+02
2 TRACE NO. 2	1.255000+03
3 MAXIMUM ELEMENT	3.600000+01
4 MINUMUM ELEMENT	4.000000+00
5 MAXIMUM ELEMENT IN ABS VALUE	3.600000+01
6 MINUMUM ELEMENT IN ABS VALUE	4.000000+00
7 MIN NON-ZERO ELEM IN ABS VAL	4.000000+00
8 NUMBER OF POSITIVE ELEMENTS	16
9 NUMBER OF ZERO ELEMENTS	0
10 NUMBER OF NEGATIVE ELEMENTS	0
11 SUM OF TERMS	3.030000+02
12 AVERAGE	1.893750+01
13 SUM OF SQUARES	7.691000+03
14 SUM OF SQUARES ABOUT MEAN	1.952937+03
15 WITHIN ROWS SUM OF SQUARES	1.112750+03
16 WITHIN COLS SUM OF SQUARES	1.112750+03
17 SUM OF ABSOLUTE VALUES	3.030000+02
18 AVERAGE OF ABSOLUTE VALUES	1.893750+01
SPECIFIC	
19 DETERMINANT	1.024000+03
20 RANK	4
NORMS	
21 SQ ROOT OF SUM OF B(I,J)**2	2.7-06
22 N*MAX(B(I,J))	1.2-05
23 MAX VAL OF ROW SUM	2.5-06
24 NORMALITY	YES*(1)
25 SYMMETRY	YES*(1)
26 SKEW SYMMETRY	NO*(0)
27 DIAGONALITY	NO*(0)
28 ORTHOGONALITY	NO*(0)
29 A'A=DIAGONAL MATRIX	NO*(0)
30 TRIANGULAR	NO**(0)
31 STOCHASTIC (R AND/OR C SUMS=1)	NO***(0)

* IF ANSWER IS YES, (R,C)=1 OR 2. (1, IF EXACT; 2, IF TOLERANCE IS SATISFIED.)
IF ANSWER IS NO, (R,C)=0.

TRIANGULAR
** (R,C)=0,IF ANSWER IS NO; (R,C)=1, IF UPPER PART OF MATRIX IS ZERO;
(R,C)=2, IF LOWER PART IS ZERO; (R,C)=3, IF ALL OFF DIAGONAL ELEMENTS = 0.

STOCHASTIC
*** (R,C)=0, IF MATRIX IS NOT STOCHASTIC; (R,C)=1, IF SUM OF EACH ROW = 1;
(R,C)=2, IF SUM OF EACH COLUMN=1; (R,C)=3, IF SUM OF EACH ROW AND COLUMN=1.

PROPERTIES OF 3 X 4 MATRIX STARTING LOCATION (1, 1)

R	GENERAL	
1	TRACE (3 VALUES USED)	6.500000+01
2	TRACE NO. 2	6.440000+02
3	MAXIMUM ELEMENT	3.600000+01
4	MINUMUM ELEMENT	4.000000+00
5	MAXIMUM ELEMENT IN ABS VALUE	3.600000+01
6	MINUMUM ELEMENT IN ABS VALUE	4.000000+00
7	MIN NON-ZERO ELEM IN ABS VAL	4.000000+00
8	NUMBER OF POSITIVE ELEMENTS	12
9	NUMBER OF ZERO ELEMENTS	0
10	NUMBER OF NEGATIVE ELEMENTS	0
11	SUM OF TERMS	2.000000+02
12	AVERAGE	1.666667+01
13	SUM OF SQUARES	4.666000+03
14	SUM OF SQUARES ABOUT MEAN	1.332667+03
15	WITHIN ROWS SUM OF SQUARES	7.400000+02
16	WITHIN COLS SUM OF SQUARES	8.293333+02
17	SUM OF ABSOLUTE VALUES	2.000000+02
18	AVERAGE OF ABSOLUTE VALUES	1.666667+01
	SPECIFIC	
28	ORTHOGONALITY	NO*(0)
29	A'A=DIAGONAL MATRIX	NO*(0)

* (R,C)=0, IF MATRIX IS NOT ORTHOGONAL; (R,C)=1 OR 2 IF MATRIX IS ORTHOGONAL ROW WISE; (R,C)=3 OR 4, IF MATRIX IS ORTHOGONAL COL WISE. ((R,C)=I, IF I=1,3 ORTHOGONALITY EXACT; FOR I=2 OR 4 RELATIVE WITHIN ERROR BOUND OF .1E-6)

A matrix is triangular, item 30, if all the elements below (or above) the principal exactly equal to zero. Enclosed in the parentheses is the value stored (if storage has been requested) in row 30. The value will be zero, one, two or three depending upon whether the matrix is not triangular, lower triangular (values above diagonal equal zero), upper triangular (values below diagonal equal zero) or all off-diagonal elements equal zero.

A matrix is defined to be stochastic, item 31, if all its elements are non-negative and all the row (or column) sums are one. The value in parentheses is zero, one, two or three depending upon whether the matrix is non-stochastic, all rows sum to unity, all columns sum to unity or both all row and columns sums are unity, respectively. For a description of stochastic matrices, see "The Theory of Stochastic Processes," Cox, D. R. and Miller, H. D., John Wiley and Sons, Inc. (1965).

If the worksheet has been redimensioned and the number of rows is less than that needed to store all the properties of a matrix, then only as many properties will be stored as is possible.

```
MPROPERTIES of the matrix in (R),(C) of size (r)x(c)
```

This form of the instruction prints the 31 properties for a square matrix, or 20 properties for a non-square matrix, but provides no storage of results.

```
MPROPERTIES of matrix (R),(C) size (r)x(c) put properties in col (C)
```

The properties of the matrix are printed and also put in the column designated by the 5th (last) argument.

```
MPROPERTIES of (R),(C) size (r)x(c) put row of column averages in (R),(C)
```

The (c) column averages are put in the row designated by the last pair of arguments. If the instruction

MPROPERTIES of matrix in 1,1 size 3x4, put row of column averages in 1,21

had been used in the set of instructions on page 218, then the numbers 6.0000000, 17.000000, 21.333333 and 22.333333 would have been put in row 1 of columns 21, 22, 23 and 24.

```
MPROPERTIES of (R),(C) size (r)x(c) put prop's in (C) col ave's in (R),(C)
```

Same as the above form, but the properties are also stored in the worksheet. The properties of the matrix are put in the column designated by the 5th argument. The location of the row of column averages is determined by the 6th and 7th (last two) arguments.

```
MPROPERTIES of (R),(C) size (r)x(c) put col ave's (R),(C) row ave's (R),(C)
```

This form of the instruction stores both the row and column averages of the matrix. The column averages are put in the row indicated by the 3rd pair of arguments. The row averages are put in the column indicated by the 4th (last) pair of arguments. If the instruction

MPROPERTIES matrix in 1,1 size 3x4 put col ave's in 1,21 and row ave's in 1,31

had been used in the set of instructions on page 218, then the numbers 7.0, 19.5 and 23.5 would have been put in the first three rows of column 31. The instruction

MPROPERTIES matrix in 1,1 size 3x4, put averages in 4,1 and 1,5

would border the original 3x4 matrix with row and column averages to form a 4x5 matrix.

/ MPROPERTIES of (R),(C) size (r)x(c) put in (C) ave's in (R),(C) and (R),(C) /

Same as the above form, but the properties are also stored in the worksheet. The properties are put in the column designated by the 5th argument. The location in the worksheet of the row of column averages is determined by the 6th and 7th arguments. The location of the column of row averages is determined by the 8th and 9th (last two) arguments.

All forms of the instruction, except the first, have an additional form which has the letter S at the beginning of the command. The letter S indicates that the automatic printing of the properties is to be suppressed and only the requested results are stored. These forms are listed below, but they are not described. If an attempt is made to put the letter S at the beginning of the command in the first form, above, the instruction will be ignored and the following informative diagnostic will be given:

* COMMAND IGNORED - S BEFORE COMMAND NAME MEANINGLESS IF NO STORAGE REQUESTED

*****

: SMPROPERTIES of matrix (R),(C) size (r)x(c) put properties in col (C) :

/ SMPROPERTIES of (R),(C) size (r)x(c) put row of column averages in (R),(C) /

/ SMPROPERTIES of (R),(C) size (r)x(c) put prop's in (C) col ave's in (R),(C) /

/ SMPROPERTIES of (R),(C) size (r)x(c) put col ave'ss (R),(C) row ave's (R),(C) /

/ SMPROPERTIES of (R),(C) size (r)x(c) put in (C) ave's in (R),(C) and (R),(C) /

8.7 Printing.

MPRINT, MPRINT "L"

These two instructions are listed here for completeness, but they are described in section 1.8.

: MPRINT the matrix in (R),(C) of size (r)x(c) :

MPRINT "L" format, the matrix in (R),(C) of size (r)x(c)

9. BESSEL FUNCTIONS.

This section describes 34 instructions which may be used to evaluate Bessel functions. Included are: Bessel functions and modified Bessel functions of order zero, one and n (integer) for both real and complex argument. Also included, are instructions for computing zeros of Bessel functions and a definite integral.

9.1 First and Second Functions of Order Zero and One.

BJONE, BJZERO, BYONE, BYZERO.

All the Bessel functions of order zero and one are evaluated using standard series and asymptotic expansions, and the calculations are performed in double precision arithmetic but stored as single precision.

$$J_n(x) = (1/\pi)\int_0^{\pi}\cos[x\sin(t)-nt]dt,$$

$$Y_n(x) = (1/\pi)\int_0^{\pi}\sin[x\sin(t)-nt]dt,$$

for n = 0 and 1. See Abramowitz, M., and Stegun, I. A., "Handbook of Mathematical Functions", AMS 55, chapter 9.

BJONE of (E) put results in column (C)

BJONE evaluates the Bessel function of the first kind of first order, $J_1(x)$.

BJZERO of (E) and put results in column (C)

BJZERO evaluates the Bessel function of the first kind of zero order, $J_0(x)$.

BYONE of (E) and put results in column (C)

BYONE evaluates the Bessel function of the second kind of first order, $Y_1(x)$. Whenever x=0.0, the result will be set equal to zero and the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

BYZERO of (E) and put results in column (C)

BYZERO evaluates the Bessel function of the second kind of zero order $Y_0(x)$. If x=0.0, the result is set equal to zero and the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

9.2 Modified Functions.

BIONE, BIZERO, BKONE, BKZERO.

These instructions evaluate the following modified Bessel functions:

$$I_n(x) = (1/\pi)\int_0^{\pi}\exp[x\cos(t)]\cos(nt)dt$$

$$K_n(x) = \csc(n\pi/2)\int_0^{\infty}\sin[x\sinh(t)]\sinh(nt)dt,$$

for n = 0 and 1.

The results of the modified Bessel functions will be scaled for arguments whose absolute values are equal to or greater than K = 88.0 (for the National Bureau of Standards computer), since that is the largest value of exp(K) that can be evaluated in single precision floating point arithmetic without overflow. The scaling factor used is exp(K). Whenever scaling does occur, the following arithmetic diagnostic will be given:

** BESSEL ARGUMENTS SCALED TO AVOID OVER/UNDER FLOW. RETURNED (n) TIMES

BIONE of (E) and put results in column (C)

BIONE evaluates the modified Bessel function of the first kind of first order, $I_1(x)$. The result will be set equal to $\exp(-x)I_1(x)$, for the absolute value of x equal or greater than K. If column 38 contains the values 1.0, 2.5 and 5.0, then the instruction

BIONE of x in col 38 and put $I_1(x)$ in col 2

will put the following values in column 2:

Row/Column	2
1	.56515910
2	2.5167162
3	24.335642

BIZERO of (E) and put results in column (C)

BIZERO evaluates the modified Bessel function of the first kind of zero order, $I_0(x)$. For $|x| \geq K$, $I_0(x)$ will be multiplied by exp(-x) so that the result will stay within the bounds of the computer capability.

BKONE of (E) and put results in column (C)

BKONE evaluates the modified Bessel function of the second kind of first order, $K_1(x)$. The result will be scaled by exp(x), if $|x| \geq K$. If x=0.0, the result is set equal to zero and the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

BKZERO of (E) and put results in column (C)

BKZERO evaluates the modified Bessel function of the second kind of zero order, $K_0(x)$, and the results will be multiplied by exp(x), if $|x| \geq K$. If x=0.0, the result is set equal to zero and the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

9.3 Modified Functions with Extreme Valued Argument.

EXIONE, EXIZERO, EXKONE, EXKZERO

The formulas for evaluating these functions are the same as those in section 9.2; except the scale factor, exp(x), is part of the formulas and is not optional.

EXIONE of (E) and put results in column (C)

EXIONE evaluates the modified Bessel function of the first kind of first order. Assuming column 38 contains the numbers 1.0, 2.5 and 5.0, the instruction

EXIONE of x in col 38 and put $\exp(-x)I_1(x)$ in col 3

will put the following numbers in column 3:

Row/Column	3
1	.20791041
2	.20658465
3	.16397227

Note, if the numbers in column 2 of the example of BIONE are multiplied by exp(-x), the results will equal those in column 3 in the above example.

EXIZERO of (E) and put results in column (C)

EXIZERO evaluates the modified Bessel function of the first kind of zero order, $\exp(-x)I_0(x)$.

EXKONE of (E) and put results in column (C)

EXKONE evaluates the modified Bessel function of the second kind of first order, $\exp(x)K_1(x)$. The result is set equal to zero for x=0.0, and the following arithmetic diagnostic is given;

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

EXKZERO of (E) and put results in column (C)

EXKZERO evaluates the modified Bessel function of the second kind of zero order, $\exp(x)K_0(x)$. If x=0.0, the result is set equal to zero and the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

9.4 Complex Functions; Angle=$\pi/4$ (Kelvin Functions).

KBIONE, KBIZERO, KBKONE, KBKZERO

The Bessel functions of complex argument, Rexp($i\pi/4$) , computed for the above commands are related to the Kelvin functions as follows:

$$I_0(Re^{i\pi/4}) = \text{ber}(R) + i\ \text{bei}(R)$$

$$K_0(Re^{i\pi/4}) = \text{ker}(R) + i\ \text{kei}(R)$$

$$I_1(Re^{i\pi/4}) = \text{bei}_1(R) - i\ \text{ber}_1(R)$$

$$K_1(Re^{i\pi/4}) = \text{kei}_1(R) + i\ \text{ker}_1(R)$$

The results will be scaled if the absolute value of $R/\sqrt{2}$ is equal to or greater than K = 88.0 (for NBS computer), and the following arithmetic diagnostic will be given:

** BESSEL ARGUMENTS SCALED TO AVOID OVER/UNDER FLOW. RETURNED (n) TIMES

KBIONE of R=(E) put real part in column (C) and imaginary part in column (C)

KBIONE evaluates the Bessel function of complex arguments of order one $I_1(Rexp(i\pi/4))$. The results will be scaled by the factor $\exp(-R/\sqrt{2})$, if necessary. If NRMAX is equal to 3, then the instruction

KBIONE of R = 3.0 put real part in col 7, imaginary part in col 8

will put the number -.48745418 in rows one, two and three of column 7 and the number 1.7326442 in column 8.

KBIZERO of R=(E) put real part in column (C) and imaginary part in column (C)

KBIZERO evaluates the Bessel function of complex arguments of order zero, $I_0(Rexp(i\pi/4))$. The results will be scaled by the factor $\exp(-R/\sqrt{2})$, if necessary.

KBKONE of R=(E) put real part in column (C) and imaginary part in column (C)

KBKONE evaluates the Bessel function of complex arguments of order one, for $K_1(Rexp(i\pi/4))$. Whenever necessary, the scale factor used is $\exp(R/\sqrt{2})$.

KBKZERO of R=(E) put real part in column (C) and imaginary part in column (C)

KBKZERO evaluates the Bessel function of complex arguments of order zero, for $K_0(Rexp(i\pi/4))$. Whenever necessary, the scale factor used is $exp(R/\sqrt{2})$.

9.5 Complex Functions with Extreme Valued Real Argument (Kelvin Functions).

KEXIONE, KEXIZERO, KEXKONE, KEXKZERO

These instructions are similar to the instructions in section 9.4. They evaluate the same Bessel functions of complex argument; except the values are scaled before the results are put in the designated columns.

KEXIONE of R=(E) put real part in column (C) and imaginary part in column (C)

KEXIONE evaluates the Bessel function of complex argumeuts of order one with a scale factor, $exp(-R/\sqrt{2})I_1(Rexp(i\pi/4))$.

KEXIZERO of R=(E) put real part in column (C) and imaginary part in column (C)

KEXIZERO evaluates the Bessel function of complex arguments of order zero with a scale factor, $exp(-R/\sqrt{2})I_0(Rexp(i\pi/4))$.

KEXKONE of R=(E) put real part in column (C) and imaginary part in column (C)

KEXKONE evaluates the Bessel function of complex arguments of order one with a scale factor, $exp(R/\sqrt{2})K_1(Rexp(i\pi/4))$. If R is less than or equal to zero, the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

KEXKZERO of R=(E) put real part in column (C) and imaginary part in column (C)

KEXKZERO evaluates the Bessel function of complex arguments of order zero with a scale factor, $exp(R/\sqrt{2})K_0(Rexp(i\pi/4))$. If R is less than or equal to zero, the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

9.6 Complex Functions with Arbitrary Angle, $0 \leq A \leq \pi/2$.

CIONE, CIZERO, CKONE, CKZERO

These instructions are similar to the instructions in section 9.4. Whereas in section 9.4 the angle is assumed to be $\pi/4$, these instructions permit the user to specify the angle in radians. The angle or angles designated must be equal to or greater than zero and less than or equal to $\pi/2$.

The results will be scaled, if the absolute value of $Rcos(A)$ is greater than or equal to K = 88.0 (for NBS computer), and the following arithmetic diagnostic will be given:

** BESSEL ARGUMENTS SCALED TO AVOID OVER/UNDER FLOW. RETURNED (n) TIMES

CIONE of R=(E), A=(E) put real part in column (C) and imaginary part in col (C)

CIONE evaluates the Bessel function of complex arguments of order one, $I_1(Rexp(iA))$, where the angle A is in radians. If necessary the result will be scaled by the factor exp(-Rcos A). If column 54 contains the values 2, 4, 6 and 8, the instruction

CIONE of R in col 54, angle=.523598775, put real part in col 2, imaginary in col 3

will place the following results in columns 2 and 3:

	Real Part	Imaginary
Row/Column	2	3
1	.78785283	1.0378416
2	-1.3246755	5.7005536
3	-25.943990	9.9659038
4	-112.10820	-80.535762

Note, the angle in this example (.523598775 radians) is 30 degrees.

CIZERO of R=(E), A=(E) put real part in col (C) and imaginary part in col (C)

CIZERO evaluates the Bessel function of complex arguments of order zero, $I_0(Rexp(iA))$, and the scale factor used is exp(-Rcos A), if necessary.

CKONE of R=(E), A=(E) put real part in column (C) and imaginary part in col (C)

CKONE evaluates the Bessel function of complex arguments of order one, $K_1(Rexp(iA))$, and the scale factor exp(Rcos A)is used, if necessary.

CKZERO of R=(E), A=(E) put real part in col (C) and imaginary part in col (C)

CKZERO evaluates the Bessel function of complex arguments of order zero, $K_0(Rexp(iA))$, and the scale factor exp(Rcos A) is used, if necessary.

9.7 Complex Functions with Extreme Real Argument.

CEIONE, CEIZERO, CEKONE, CEKZERO

These commands are the same as those in 9.6; except each of the functions is always multiplied by a scale factor.

CEIONE of R=(E), A=(E) put real part in col (C) and imaginary part in col (C)

CEIONE evaluates the Bessel function of complex arguments of order one with a scale factor, $exp(-Rcos\ A)I_1(Rexp(iA))$.

CEIZERO of R=(E), A=(E) put real part in col (C) and imaginary part in col (C)

CEIZERO evaluates the Bessel function of complex arguments of order zero with a scale factor, $\exp(-R\cos A)I_0(R\exp(iA))$.

CEKONE of R=(E), A=(E) put real part in col (C) and imaginary part in col (C)

CEKONE evaluates the Bessel function of complex arguments of order one with a scale factor, $\exp(R\cos A)K_1(R\exp(iA))$, and if R is less than or equal to zero, the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

CEKZERO of R=(E), A=(E) put real part in col (C) and imaginary part in col (C)

CEKZERO evaluates the Bessel function of complex arguments of order zero with a scale factor, $\exp(R\cos A)K_0(R\exp(iA))$, and if R is less than or equal to zero, the following arithmetic diagnostic is given:

** NEGATIVE ARGUMENT TO SQRT, LOG, OR RAISE

9.8 Zeros of Bessel Functions.

ZEROS BJONE, ZEROS BJZERO

If NRMAX is greater than 1000, only the first 1000 positive roots are computed.

ZEROS BJONE put x in col (C) and $J_0(x)$ in col (C)

ZEROS BJONE computes the positive roots for the Bessel function $J_1(x_s)=0$, s=1,2,...,NRMAX and the values of $J_0(x_s)$. For NRMAX=2, the instruction

ZEROS BJONE put x in col 17 and $J_0(x_s)$ in col 24

will give the following results in columns 17 and 24;

s	Row/column	x_s 17	$J_0(x_s)$ 24
1	1	3.831706	-.4027594
2	2	7.0155866	.30011575

ZEROS BJZERO put x in col (C) and $J_1(x)$ in col (C)

ZEROS BJZERO computes the positive roots for the Bessel function $J_0(x_s)=0$, s=1,2,...,NRMAX and the values of $J_1(x_s)$. The instruction

ZEROS BJZERO put x in col 1 and $J_1(x_s)$ in col 2

for NRMAX=2, will produce the following results:

		x_s	$J_1(x_s)$
s	Row/Column	1	2
1	1	2.4048256	.51914749
2	2	5.200781	-.34026480

9.9 Bessel Fuctions of Order n.

BESIN, BESJN, BESKN

These commands compute the integral orders of the Bessel functions of first and second kind for n=0,1,...,NRMAX. If NRMAX is greater than 99, only the first 100 values are computed. The first argument of these commands must be a constant with a decimal point. If the constant is too large or too small, the result is set equal to zero and the following informative diagnostic is given:

* ARG FOR BESIN, BESJN, BESKN GIVES A RESULT TOO LARGE/SMALL. COMMAND NOT EXECUTED

BESIN of x=(K) and put results in column (C)

BESIN computes $I_n(x)$ for order n=0,1,...,NRMAX. For NRMAX=3, the instruction

BESIN for x=5.0 put results in column 4

will put the following values in column 4:

	Row/Column	4
$I_0(5)$	1	-.17759677
$I_1(5)$	2	-.32757914
$I_2(5)$	3	.046565116

BESJN of x=(K) put in column (C)

BESJN computes $J_n(x)$ for order n=0,1,...,NRMAX.

BESKN of x=(K) put in column (C)

BESKN computes $K_n(x)$ for order n=0,1,...,NRMAX.

9.10 Integral.

INTJO

INTJO of x=(E) put in column (C)

INTJO evaluates the definite integral of the Bessel function of order zero:

$$f(x)= \int_0^x J_0(t)dt.$$

10. THERMODYNAMICS.

This section contains 11 instructions which are useful for thermodynamic calculations. The instructions in section 10.1 have more general use.

10.1 Temperature Scale Conversion.

CTOF, FTOC

The two instructions in this section enable one to convert temperature from degrees Celsius (centigrade) to degrees Fahrenheit and to convert from degrees Fahrenheit to degrees Celsius, respectively. Whenever degrees in Celsius is less than -273.15 (less than Kelvin zero), the following informative diagnostic is given:

* NEGATIVE ABSOLUTE TEMPERATURES CONVERTED

CTOF for Celsius (E) put Fahrenheit equivalent in column (C)

The instruction converts degrees Fahrenheit to degrees Celsius using the relation

$$°F = 32.0 + 1.8°C$$

FTOC Fahrenheit is (E) and put Celsius equivalent in column (C)

The instruction converts degrees Celsius to degrees Fahrenheit using the relation

$$°C = -32.0 + °F/1.8$$

10.2 Systems Of Units.

CGS, SI

Values of the fundamental physical constants are given and discussed in section B1.14. The fundamental physical constants (except π and e) listed in the table in section B1.14 are in the OMNITAB II system to the full significance given in Abramowitz, M. and Stegun, I. A., Handbook of Mathematical Functions, National Bureau of Standards, AMS 55. The physical constants may be referenced in either centimeter-gram-second (CGS) units or in the Système International (SI) units. If the system of units is not specified, then SI units will be used.

CGS system of fundamental physical constants, centimeter-gram-second

After this instruction is executed, the fundamental physical constants will be in the centimeter-gram-second units. In the example

```
CGS system
DEFINE *CTWO* into column 5
```

the value entered into all rows of column 5 is 1.43879.

SI system of fundamental physical constants, Système International

This instruction resets the units of the fundamental physical constants in the SI system, or Système International d'Unités.

10.3 Molecular Weight.

ATOMIC, MOLWT

ATOMIC mass table put in column (C)

The atomic weights of the elements from atomic number 1, hydrogen, to atomic number 103, Lawrencium, will be put in the specified column. If the number of rows in the worksheet is less than 103, only enough values will be stored to fill the column and the following informative diagnostic will be given:

* COLUMN NOT LONG ENOUGH TO STORE ALL ELEMENTS. ONLY NROW WILL BE STORED

If NRMAX is less than 103 or the number of rows in the worksheet, NRMAX will be reset to the number of atomic weights stored. The values for the atomic weights were obtained from Comptes Rendus XXV Conference, International Union of Pure and Applied Chemistry, 1970.

MOLWT of compunds Z=(k), N=(k); Z=(k), N=(k); ... etc. put in column (C)

MOLWT evaluates the molecular weight of compounds. The last argument of the instruction specifies the column number where the molecular weight of the compound is to be put. All the rows through NRMAX will contain the same value. The other arguments are integers and are used as pairs. Therefore, this instruction always has an odd number of arguments. The first constant in each pair specifies the atomic number of the element and the second argument indicates what multiplying factor is to be used. If one wants to evaluate the molecular weight of water (2 parts hydrogen, 1 part oxygen) the instruction to use is

MOLWT 1 atomic no. hydrogen 2 parts, 8 at. no. oxygen 1 part put in col 31

If NRMAX=4, then column 31 contains 18.0154, 18.0154, 18.0154 and 18.0154.

10.4 Properties Of State.

BOLDISTRIBUTION, EINSTEIN, PARTFUNCTION, PFATOMIC, PFTRANSLATIONAL

These instructions calculate and store thermodynamic tables of properties of state. Temperatures are specified in Kelvin degrees. The last argument of each instruction specifies the starting storage location of the table computed. If there are not enough columns in the worksheet to store the table, the following fatal error occurs:

*** COLUMN NUMBER TOO BIG OR LESS THAN 1

Negative temperatures and wave numbers are not permitted and cause the following fatal error:

*** ILLEGAL ARGUMENT ON CARD

BOLDISTRIBUTION for temp (E), wave nos in (C), degens (C), put table in (C) on

This instruction (Bolzmann distribution) produces a table giving the fraction of molecules in each of the n energy levels, having the specified wave numbers, at the given temperatures, using the formula

$$P_i = g_i \exp(-hcE_i/kT) / \sum_{j=1}^{n} g_j \exp(-hcE_j/kT)$$

to find the fractional population of the ith energy level. The ordered pairs of energy levels and degeneracies are read from parallel columns indicated as the second and third arguments of the instruction. These columns are read down to the last nonzero degeneracy to obtain n pairs. Note, this may be above or below NRMAX. The table generated has NRMAX rows and n columns. If fewer than n columns lie to the right of the column in which the table begins, the following fatal error occurs:

*** DEFINED MATRIX OVERFLOWS WORKSHEET

If only one nonzero degeneracy is given, ones will be vectorized to NRMAX, indicating that all molecules are in the given energy level.

Negative Kelvin temperatures or degeneracies in the worksheet will result in an informative diagnostic. (If a negative Kelvin temperature is entered in the instruction, the fatal error *** ILLEGAL ARGUMENT ON CARD occurs.) If no nonzero degeneracies are found in the specified column, a fatal error occurs.

EINSTEIN of temperatures (E), wave numbers (E), put table in (C) and succ. cols

EINSTEIN evaluates the contributions to the thermodynamic properties of a harmonic oscillator in one degree of freedom for desired temperatures designated by the first argument and the vibrational frequencies defined in wave numbers by the second argument. The information is stored as a 7 column table starting with the column designated by the third argument through the succeeding six columns C+1, C+2, ..., C+6 as follows:

Column	Function
C	E - wave numbers specified by second argument
C+1	T - temperatures specified by first argument
C+2	$-(F^\circ-E_o^\circ)/RT = -\ln(1-e^{-x})$
C+3	$(H^\circ-E_o^\circ)/RT = xe^{-x}/(1-e^{-x})$
C+4	$S^\circ/R = -(F^\circ-E_o^\circ)/RT + (H^\circ-E_o^\circ)/RT$
C+5	$C_p^\circ/R = x^2e^{-x}/(1-e^{-x})^2$
C+6	$(H^\circ-E_o^\circ)/R$,

where $x = hcE/kT$ and $hc/k = 1.43879$.

If NRMAX=2 and column 1 had the values 1.25 and 1.50, then the instruction

EINSTEIN 1.43879, column 2 put in column 13

would put the following table in the worksheet in rows 1 and 2 of columns 13 through 19:

Row/Col	13	14	15	16	17	18	19
1	1.25	1.43879	.33757957	.50193889	.83951846	.87936628	.72218466
2	1.50	1.43879	.25248246	.43082537	.68330783	.83184856	.61986723

EINSTEIN temps (E), wave nos (E), R=(K), put table in (C) and succ. cols

This instruction is similar to the preceding one. Each of the stored thermal functions, with the exception of temperatures and wave numbers, is multiplied by R, the third argument in the instruction.

PARTFUNCTION temp is (E), wave nos in (C), degens in (C), put table in (C) on

Evaluates the following three equations and stores them as a table:

$$Q^0 = \sum_{i=1}^{n} g_i \exp(-hcE_i/kT)$$

$$Q^1 = \sum_{i=1}^{n} g_i (hcE_i/kT) \exp(-hcE_i/kT)$$

$$Q^2 = \sum_{i=1}^{n} g_i (hcE_i/kT)^2 \exp(-hcE_i/kT),$$

where $hc/k = 1.43879$, E_i is the wave number of the ith energy level, g_i are the degeneracies (weights) of the energy levels and n is the number of energy levels with non-zero degeneracies.

If column 2 contains 78, 15868 and 33792 and column 3 contains 9, 5 and 1, then the instruction

PARTFUNCTION 3000. col 2, col 3, put in column 11

would put the numbers 8.0445831, .9028034 and .10131854 into row 1 of columns 11, 12 and 13 (NRMAX = 1).

PFATOMIC temp is (E) mol wt (E) wave nos in (C) degens (C), put table in (C) on

PFATOMIC evaluates a table of the contributions to the following thermal functions and stores them in six consecutive columns starting with the column specified by the last argument.

Column	Function
C	T - temperatures
C+1	$-(F^\circ-E_o^\circ)/RT = 2.5x\ln(T) + 1.5x\ln(M) - 3.66495 + \ln(Q^0)$
C+2	$(H^\circ-E_o^\circ)/RT = 2.5 + Q^1/Q^0$
C+3	$S^\circ/R = (H^\circ-E_o^\circ)/RT - (F^\circ-E_o^\circ)/RT$
C+4	$C_p^\circ/R = 2.5 + (Q^2/Q^0) - (Q^1/Q^0)^2$
C+5	$(H^\circ-E_o^\circ)/R$,

where R = 1.98717 and Q^0, Q^1 and Q^2 are the formulas listed under PARTFUNCTION, above. Assume columns 2 and 3 contain the values given under PARTFUNCTION, then the instruction

PFATOMIC temp 3000. mol wt 31.9988 wave nos 2, G = col 3 put in col 41

would put the numbers 3000.0, 21.549516, 2.5, 24.049516, 2.5 and 7500.0 into row 1 of columns 41 to 46.

PFTRANSLATIONAL temp is (E) mol wt (E) put table in (C) and successive cols

A table of the translational contributions to the thermal functions is computed and stored in six consecutive columns starting with the column specified by the last argument. The contents of the six columns are as follows:

Column	Function
C	T - temperatures
C+1	$-(F^\circ-E_o^\circ)/RT = 2.5x\ln(T) + 1.5x\ln(M) - 3.66495$
C+2	$(H^\circ-E_o^\circ)/RT = 2.5$
C+3	$(H^\circ-E_o^\circ)/RT - (F^\circ-E_o^\circ)/RT$
C+4	$C_p^\circ/R = 2.5$
C+5	$(H^\circ-E_o^\circ)/R$

11. INDEX TO COMMANDS DESCRIBED IN PART C.

Page number shown is the page on which the first form of the instruction appears. Abbreviations and synonyms are not listed.

PART D

LIST OF INSTRUCTIONS

1. Instructions are listed alphabetically; except all the array operation instructions are listed separately after the A's and all the matrix operation instructions are listed separately after the M's. All array operation instructions begin with the letter A and all the matrix operation instructions begin with the letter M. The exception is the letter S, which is used in SAPROP and SMPROP to indicate suppression of automatic printing. All forms of an instruction, synonyms and abbreviations are listed.

2. The command name is given in capital letters. Non-essential, descriptive words which clarify the meaning of an instruction are printed in lower case letters. The dollar sign, $, precedes additional comments and information. No numerals are used in a command name except in TITLE1, TITLE2, TITLE3, TITLE4, NOTE1 and NOTE2. The number must immediately follow TITLE or NOTE and is part of the command name. A blank space must precede any descriptive words used immediately after a command name. When an instruction has more than one form, the alternative form(s) is given just below and indented.

3. Some command names have a qualifier denoted by "L" where "L" indicates either the letter A, B, C, D, E or F. The qualifier (without quotation marks) is part of the command name. One blank space, at least, must precede and follow the qualifier without any additional characters. Some of the TAPE instructions have two qualifiers. One instruction, RESET "V", has the qualifier "V" where "V" denotes the letter V, W, X, Y or Z. Instructions with a qualifier are considered distinct from the similar instruction without a qualifier and are not indented.

4. Parentheses enclosing a letter indicate the type of argument (number) allowed. To make the the type of argument used explicit, descriptive words are used as much as possible. Lower case letters always represent integers without a decimal point. Examples are (r) = the number of rows and (c) = the number of columns. Capital letters are used as follows:

(C) = a COLUMN number, which must not have a decimal point
(E) = EITHER a column number or a constant
(K) = a CONSTANT, which must have a decimal point
(N) = an instruction NUMBER with or without a decimal point
(R) = a ROW number, which must not have a decimal point

5. On the extreme right, auxiliary information is given. Use of this information can prevent errors. Notes are defined as follows. Note on number of arguments is always first.

A = an abbreviation exists
B = this instruction may store values below NRMAX
C = this instruction cannot be stored for repeated execution
D = number of arguments in the instruction must be odd, but cannot exceed 100
E = number of arguments in the instruction must be even, but cannot exceed 100
M = this instruction must be stored for repeated execution
N = execution of this instruction may or will affect the value of NRMAX
P = this instruction produces printing
S = one or more synonyms exist or this is an abbreviation of another instruction
V = the number of arguments is variable, but cannot exceed 100
W = this instruction will work anywhere in the worksheet, i.e. below NRMAX
X = this particular form of the instruction does not provide storage of results

If V, D or E is not applicable, the exact number of arguments allowed is given.

This information is summarized in the footnote which appears below and on all other pages.

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. nulber
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

***** A *****

ABRIDGE row (R) of columns (C), (C), ... (C)	V,P,W
ABRIDGE row (R) of columns (C), (C) ... (C) with (K) significant digits	V,P,W
ABRIDGE row (R) of (C)...(C) with (K) s. digits, (C)...(C) with (K), etc	V,P,W
ABRIDGE row (R), (K) cols, (C) (s) s.d., (C) (s) ... $ max width 22, 3 blanks	V,P,W
ABRIDGE row (R), (K) cols, (C) (s) (m) max width, (C) (s) (m), ... $ 3 blanks	V,P,W
ABRIDGE row (R) of (K) cols, (C) (s) (m) (b) blanks, (C) (s) (m) (b), ...	V,P,W
ABRIDGE "L" format, row (R) of columns (C), (C) ... (C)	V,P,W
ABS value of (E) put in column (C)	2,S
ABS value of (E), multiply by (E), add to (E), put in column (C)	4,S
ABSOLUTE value of (E) put in column (C)	2,A
ABSOLUTE value of (E), multiply by (E), add to (E), put in column (C)	4,A
ACCURACY of (E) compared to (E) put in column (C)	3
ACOS of (E) put in column (C)	2
ACOS of (E), multiply by (E), add to (E), put in column (C)	4
ACOSD of (E), put in column (C)	2
ACOSD of (E), multiply by (E), add to (E), put in column (C)	4
ACOSH of (E), put in column (C)	2
ACOSH of (E), multiply by (E), add to (E), put in column (C)	4
ACOT of (E), put in column (C)	2
ACOT of (E), multiply by (E), add to (E), put in column (C)	4
ACOTD of (E), put in colum (C)	2
ACOTD of (E), multiply by (E), add to (E), put in column (C)	4
ACOTH of (E), put in column (C)	2
ACOTH of (E), multiply by (E), add to (E), put in column (C)	4
ADD (E) to (E) and put in column (C)	3
ADD (E) to (E), multiply by (E), add to (E), put in column (C)	5
ANTILOG of (E), put in column (C)	2
ANTILOG of (E), multiply by (E), add to (E), put in column (C)	4
ASIN of (E), put in column (C)	2
ASIN of (E), multiply by (E), add to (E), put in column (C)	4
ASIND of (E), put in column (C)	2
ASIND of (E), multiply by (E), add to (E), put in column (C)	4
ASINH of (E), put in column (C)	2
ASINH of (E), multiply by (E), add to (E), put in column (C)	4
ATAN of (E), put in column (C)	2
ATAN of (E), multiply by (E), add to (E), put in column (C)	4
ATAND of (E), put in column (C)	2
ATAND of (E), multiply by (E), add to (E), put in column (C)	4
ATANH of (E), put in column (C)	2
ATANH of (E), multiply by (E), add to (E), put in column (C)	4
ATOMIC mass table put in column (C)	1
AVERAGE of column (C) put in column (C)	2

***** ARRAY OPERATIONS *****

AADD the array (R),(C) size (r)x(c) to array (R),(C) size (r)x(c) put in (R),(C)	10,S,W
AADD the array in (R),(C) size (r)x(c), to array (R),(C), put in (R),(C)	8,S,W
AADD the array in (R),(C) of size (r)x(c) to (E), put array in (R),(C)	7,W
AAVERAGE on first col of array in (R),(C) size (r)x(c), put array in (R),(C)	6,W

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

AAVERAGE on (K) in first col of array (R),(C) size (r)x(c) put row in (R),(C)	7,W
ACOALESCE on first col of array in (R),(C) size (r)x(c), put array in (R),(C)	6,W
ACOALESCE on (K) in first col of (R),(C) size (r)x(c) put row in (R),(C)	7,W
ADEFINE the array in (R),(C) of size (r)x(c) to be equal to (K)	5,S,W
ADIV the array (R),(C) size (r)x(c) by array (R),(C) size (r)x(c) put in (R),(C)	10,S,W
ADIV the array in (R),(C) size (r)x(c), by the array (R),(C), put in (R),(C)	8,S,W
ADIV the array in (R),(C) of size (r)x(c), by (E), put array in (R),(C)	7,S,W
ADIVIDE array (R),(C) size (r)x(c) by array (R),(C) size (r)x(c) put in (R),(C)	10,A,W
ADIVIDE the array in (R),(C) size (r)x(c) by array (R),(C), put in (R),(C)	8,A,W
ADIVIDE the array (R),(C), size (r)x(c) by (E) put array in (R),(C)	7,A,W
AERASE the array in (R),(C) of size (r)x(c)	4,S,W
AMOVE the array in (R),(C) of size (r)x(c) to (R),(C)	6,S,W
AMULT array in (R),(C) size (r)x(c) by array (R),(C) size (r)x(c) put in (R),(C)	10,S,W
AMULT the array in (R),(C) size (r)x(c) by the array (R),(C), put in (R),(C)	8,S,W
AMULT the array in (R),(C) of size (r)x(c) by (E), put array in (R),(C)	7,S,W
AMULTIPLY array (R),(C) size (r)x(c) by (R),(C) size (r)x(c) put in (R),(C)	10,A,W
AMULTIPLY array in (R),(C) size (r)x(c) by array in (R),(C) put in (R),(C)	8,A,W
AMULTIPLY the array in (R),(C) size (r)x(c) by (E) put array in (R),(C)	7,A,W
APRINT the array in (R),(C) of size (r)x(c)	4,W,P
APRINT "L" format, the array in (R),(C) of size (r)x(c)	4,W,P
APROPERTIES of the array in (R),(C) of size (r)x(c)	4,W,X,P
APROPERTIES of the array in (R),(C) of size (r)x(c), put in column (C)	5,W,P,B
APROPERTIES of array in (R),(C) of size (r)x(c), put column ave's in (R),(C)	6,W,P
APROPERTIES of (R),(C) size (r)x(c), put prop in (C), col ave's in (R),(C)	7,W,P,B
APROPERTIES of (R),(C) size (r)x(c) col ave's in (R),(C) row ave's in (R),(C)	8,W,P
APROPERTIES array (R),(C) size (r)x(c), in (C), ave's in (R),(C) and (R),(C)	9,W,P,B
ARAISE array (R),(C) size (r)x(c) to array (R),(C) size (r)x(c) put in (R),(C)	10,W
ARAISE the array in (R),(C) of size (r)x(c) to array (R),(C), put in (R),(C)	8,W
ARAISE the array in (R),(C) of size (r)x(c) to (E), put array in (R),(C)	7,W
ASUB array (R),(C) size (r)x(c) minus array (R),(C) size (r)x(c) put in (R),(C)	10,S,W
ASUB array in (R),(C) size (r)x(c) minus the array (R),(C), put in (R),(C)	8,S,W
ASUB the array in (R),(C) of size (r)x(c) minus (E), put array in (R),(C)	7,S,W
ASUBTRACT array (R),(C) size (r)x(c) minus array (R),(C) size (r)x(c) in (R),(C)	10,A,W
ASUBTRACT array (R),(C) size (r)x(c) minus array (R),(C) put in (R),(C)	8,A,W
ASUBTRACT the array in (R),(C) size (r)x(c) minus (E), put array in (R),(C)	7,A,W
ATRANSPOSE the array in (R),(C) of size (r)x(c) into (R),(C)	6,S,W
AZERO the array in (R),(C) of size (r)x(c)	4,S,W
SAPROPERTIES of array in (R),(C) of size (r)x(c) put properties in column (C)	5,W,B
SAPROPERTIES of (R),(C) size (r)x(c) put column ave's in (R),(C)	6,W
SAPROP of (R),(C) size (r)x(c) put properties in col (C) col ave's in (R),(C)	7,W,B
SAPROP of (R),(C) size (r)x(c) put col ave's in (R),(C) row ave's in (R),(C)	8,W
SAPROP of (R),(C) size (r)x(c) put in (C) col ave's (R),(C) row ave's (R),(C)	9,W,B

***** B *****

BACKSPACE TAPE "L" unit, (n) records	1
BEGIN storing instructions for later use	0,C
BEGIN storing instructions starting with instr no. (N) $ no. less than 1000	1,C
BESIN of (K) put in column (C) $ if nrmax exceeds 99, only 1st 100 computed	2
BESJN of (K) put in column (C) $ if nrmax exceeds 99, only 1st 100 computed	2
BESKN of (K) put in column (C) $ if nrmax exceeds 99, only 1st 100 computed	2
BIONE of (E) put in column (C)	2

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

BIZERO of (E) put in column (C) 2
BJONE of (E) put in column (C) 2
BJZERO of (E) put in column (C) 2
BKONE of (E) put in column (C) 2
BKZERO of (E) put in column (C) 2
BOLDISTRIBUTION for temp (E), wave nos in (C), degens (C), put table in (C) on 4,B
BYONE of (E) put in column (C) 2
BYZERO of (E) put in column (C) 2

***** C *****

CADD real (E) imag (E) to real (E) imag (E) put real in col (C) imag in col (C) 6
CDIVIDE real (E) imag (E) by real (E) imag (E) put real in col (C) imag in (C) 6
CEIONE of R equal to (E), A equal to (E), put x in col (C) and y in col (C) 4
CEIZERO of R equal to (E), A equal to (E), put x in col (C) and y in col(C) 4
CEKONE of R equal to (E), A equal to (E), put x in col (C) and y in col (C) 4
CEKZERO of R equal to (E), A equal to (E), put x in col (C) and y in col (C) 4
CENSOR col (C) for values less than or equal to (E), replace by (E), put in (C) 4
CERF of (E) put in column (C) 2
CGS system of fundamental physical constants, centimeter-gram-second 0
CHANGE sign of values in columns (C), (C) ... (C) V
CIONE of R equal to (E), A equal to (E), put x in col (C) and y in col (C) 4
CIZERO of R equal to (E), A equal to (E), put x in col (C) and y in col (C) 4
CKONE of R equal to (E), A equal to (E), put x in col (C) and y in col (C) 4
CKZERO of R equal to (E) A equal to (E), put x in col (C) and y in col (C) 4
CLOSE UP rows with (K) in columns (C), (C) ... (C) $ puts zeros at bottom V
CMULTIPLY real (E) imag (E) by real (E) imag (E) put real in (C) imag in (C) 6
COMPARE (E) to (E) using relative tolerance (E) 3,M
CORRELATION between (p) variables in columns (C), (C) ... (C) V,P,X
  CORRELATION (p) var's in (C) ... (C), put array of simple coeffs in (R),(C) V,P,B
  CORRELATION for (p) in (C)...(C), put r coeffs in (R),(C), rho in (R),(C) V,P,B
COS of (E) put in column (C) 2
  COS of (E), multiply by (E), add to (E), put in column (C) 4
COSD of (E) put in column (C) 2
  COSD of (E), multiply by (E), add to (E), put in column (C) 4
COSH of (E) put in column (C) 2
  COSH of (E), multiply by (E), add to (E), put in column (C) 4
COT of (E) put in column (C) 2
  COT of (E), multiply by (E), add to (E), put in column (C) 4
COTD of (E) put in column (C) 2
  COTD of (E), multiply by (E), add to (E), put in column (C) 4
COTH of (E) put in column (C) 2
  COTH of (E), multiply by (E), add to (E), put in column (C) 4
COUNT length of col (C) put in (C) $ searches from below for first nozero number 2
CPOLAR for x = (E), y = (E) put rho in col (C), theta in col (C) 4
CREAD TAPE "L" unit, using (n) records into columns (C), (C) ... (C) V,N
CREAD TAPE "L" "L" unit and format, using (n) records, into cols (C) ... (C) V,N
CRECTANGULAR for rho = (E), theta = (E) put x in col (C), y in col (C) 4
CSET TAPE "L" unit, using (n) records, into column (C) 2,N
  CSET TAPE "L" unit, using (n) records, into row (R) of column (C) 3,N
CSUBTRACT real (E) imag (E) from real (E) imag (E), put real in (C) imag in (C) 6
CTOF for centigrade (E) put Fahrenheit equivalent in column (C) 2

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

***** D *****

DEFINE (E) into column (C)	2
DEFINE the constant (K) into row (R) of column (C)	3
DEFINE the value in row (R) of column (C) into column (C)	3
DEFINE the value in row (R) of column (C) into row (R) of column (C)	4
DEMOTE by (r) rows, col (C) into col (C), col (C) into col (C), etc.	D,N
DEMOTE all values in the worksheet by (r) rows	1,N
DIM the worksheet to be (r) rows by (c) columns $ r x c at most 12,500	2,S
DIMENSION the worksheet to be (r) rows by (c) columns $ r x c at most 12,500	2,A
DIV (E) by (E) and put in column (C)	3,S
DIV (E) by (E), multiply by (E), add to (E), put in column (C)	5,S
DIVIDE (E) by (E) and put in column (C)	3,A
DIVIDE (E) by (E), multiply by (E), add to (E), put in column (C)	5,A
DUMMY "L" $ available for using one's own Fortran subroutines	V
DUPLICATE (t) times, the array in (R),(C) of size (r)x(c), put in (R),(C)	7,N

***** E *****

EINSTEIN of temperatures (E), wave numbers (E), put table in (C) and succ. cols	3
EINSTEIN temps (E), wave nos (E), R=(K), put table in (C) and succ. cols	4
ELLIPTICAL FIRST of (E) put in column (C)	2
ELLIPTICAL SECOND of (E) put in column (C)	2
ENDFILE TAPE "L" unit	0
ERASE columns (C), (C) ... (C)	V
ERASE the entire worksheet and reset nrmax to zero	0,N
ERROR of (E) put in column (C)	2
EXCHANGE col (C) with col (C), col (C) with col (C), etc.	E
EXECUTE instructions numbered (N) through (N), (t) times	3,S
EXECUTE instructions numbered (N) through (N) once	2,S
EXECUTE instruction numbered (N) once	1,S
EXIONE of (E) put in column (C)	2
EXIZERO of (E) put in column (C)	2
EXKONE of (E) put in column (C)	2
EXKZERO of (E) put in column (C)	2
EXP of (E) put in column (C)	2,S
EXP of (E), multiply by (E), add to (E), put in column (C)	4,S
EXPAND (E) to power (p) in increments of (i), put in col (C) and successive cols	4
EXPAND (E) to power (K) in increments of (K), put in (C) and succ. columns	4
EXPONENTIAL of (E) put in column (C)	2,A
EXPONENTIAL of (E), multiply by (E), add to (E), put in column (C)	4,A
EXTREMA for x in (C) y in (C) put max x in (C) max y in (C) min x (C) min y (C)	6,S

***** F *****

FINISH storing instructions for later use	0,C
FIT y in col (C), weights (E), (k) variables in columns (C), (C) ... (C)	V,P,X
FIT y in (C), weights (E), (k) var's in cols (C)...(C), put coeffs in col (C)	V,P,B
FIT (C), wts (E), to (k) in (C)...(C), put coeffs in (C), residuals in (C)	V,P,B

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

FIT (C), (E), (k), (C)...(C), put coeff in (C), res. in (C), sd of pv in (C)	V,P,B
FIT (C), (E), (k), (C)...(C), put in (C), (C), (C) and Fourier coef in (C)	V,P,B
FIT (C), (E), (k), (C)...(C), put in (C),(C),(C),(C) vc matrix in (R),(C)	V,P,B
FIXED with (d) digits after decimal point	1
FLEXIBLE to return to readable printing	0
FLIP column (C) into column (C), column (C) into column (C), etc	E
FLIP the entire worksheet upside down	0
FLOATING with (s) significant digits	1
FLOATING with eight significant digits	0
FORMAT "L" (_) $ put regular Fortran format inside parentheses	0,C
F PROBABILTY with (E) and (E) degrees of freedom, for (E), put tail-area in (C)	4
FRACTIONAL part of (E) put in column (C)	2
FRACTIONAL part of (E), multiply by (E), add to (E), put in column (C)	4
FREQUENCY distribution of column (C) put in column (C)	2,N
FREQUENCY distribution of column (C), use (k) classes, put in column (C)	3,N
FREQUENCY dist'n of col (C), use (k) classes of length (K), put in col (C)	4,N
FREQUENCY of (C), use (k) classes of length (K), start at (K), put in col (C)	5,N
FREQUENCY of (C), put lower boundaries in (C), upper in (C), freq's in (C)	4,N
FREQUENCY of (C) using (k) classes, put in columns (C), (C) and (C)	5,N
FREQUENCY of (C) using (k) classes of length (K) put in cols (C),(C) and (C)	6,N
FREQUENCY of (C), classes (k), length (K), start at (K), put in (C),(C) and (C)	7,N
FTOC fahrenheit is (E) put centigrade equivalent in column (C)	2

***** G *****

GAUSS QUADRATURE with (K) points, from (K) to (K), put x in (C), weights in (C) (an integer can be used instead of a constant)	5,N
GENERATE from (K) in steps of (K) to (K) steps (K) to (K), ... , put in col (C) (an integer can be used instead of a constant)	E,N

***** H *****

HARMONIC analysis of column (C) for (n) ordinates, put coefficients in col (C)	3
HEAD column (C)/ $ 12 characters after / used as column heading	1,C
HERMITE polynomial order (n) of col (C), put in column (C) and successive cols	3
HIERARCHY of column (C), put locations of smallest thru largest in column (C)	2
HISTOGRAM using mid-points in column (C) and frequencies in column (C)	2,P

***** I *****

IFEQ (E) to (E) within the absolute tolerance (E)	3,M
IFEQ (E) to (E)	2,M
IFGE (E) to (E)	2,M
IFGT (E) than (E)	2,M
IFLE (E) to (E)	2,M
IFLT (E) than (E)	2,M
IFNE (E) to (E) with absolute tolerance (E)	3,M
IFNE (E) to (E)	2,M

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number (R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F. A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store; N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

INCREMENT instruction number (N) by (E), (E)...(E) $ no. of args in inst (N) + 1 V,M
INSERT into col (C) from (C) at every (i) row, start with row (R), put in (C) 5,N
INTEGER part of (E) put in column (C) 2
INTEGER part of (E), multiply by (E), add to (E), put in column (C) 4
INTERPOLATE x in (C) y in (C) length (n), values (v) in (C), pts (p), put in (C) 7
INTJO of (E) put in column (C) 2
INVERT the matrix in (R),(C) of size (r)x(c) and put in (R),(C) 6,S,W
ISETUP x in (C), y in (C), desired y in (C), put in col (C) and next three cols 4,N
ISOLATE x in (C), y in (C) desired y equal to (K), put in columns (C) and (C) 5,N
ISOLATE x in (C), y in (C), for (K), use (p) points, put in cols (C) and (C) 6,N
ITERATE x in (C) y in (C) desired y in col (C) put in (C) and next three cols 4,N

***** K *****

KBIONE of (E) put real part of result in col (C) imaginary part in col (C) 3
KBIZERO of (E) put real part in col (C) and imaginary part in col (C) 3
KBKONE of (E) put real part in col (C) and imaginary part in col (C) 3
KBKZERO of (E) put real part in col (C) and imaginary part in col (C) 3
KEXIONE of (E) put real part in col (C) and imaginary part in col (C) 3
KEXIZERO of (E) put real part in col (C) and imaginary part in col (C) 3
KEXKONE of (E) put real part in col (C) and imaginary part in col (C) 3
KEXKZERO of (E) put real part in col (C) and imaginary part in col (C) 3

***** L *****

LAGUERRE polynomial order (n) of col (C), put in col (C) and successive cols 3
LEGENDRE polynomial order (n) of col (C), put in col (C) and successive cols 3
LIST (n) $ controls listing of instructions, n is zero, one, two, three or four 1,S,P
LIST instructions and diagnostics 0,S,P
LOG of (E) put in column (C) $ log to the base e 2,S
LOG of (E), multiply by (E), add to (E), put in column (C) 4,S
LOGE of (E) put in column (C) 2,A
LOGE of (E), multiply by (E), add to (E), put in column (C) 4,A
LOGTEN of (E) put in column (C) 2
LOGTEN of (E), multiply by (E), add to (E), put in column (C) 4

***** M *****

MATCH column (C) with (E), extract from (E) and put in column (C) 4
MAX value of column (C) put in column (C) 2,S
MAX of column (C) put in col (C), corresponding value of (C) into col (C) ... E,S
MAXIMUM value of column (C) put in column (C) 2,A
MAXIMUM of col (C) put in col (C), corresp value of (C) in (C), ... E,A
MAXMIN x in col (C) y in col (C), put max x in (C) max y (C) min x (C) min y (C) 6,S
MIN value of column (C) put in column (C) 2,S
MIN of col (C) put in col (C), corresp value of col (C) in (C), ... E,S
MINIMUM value of column (C) put in column (C) 2,A
MINIMUM of col (C) put in col (C) corresp value of col (C) in col (C) ... E,A

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

MOLWT of compounds Z=(k), N=(k); Z=(k), N=(k); ... etc put in column (C) D
MOVE the array in (R),(C) of size (r)x(c) to (R),(C) 6,S,W
MULT (E) by (E) and put in column (C) 3,S
MULT (E) by (E), multiply by (E), add to (E), put in column (C) 5,S
MULTIPLY (E) by (E) and put in column (C) 3,A
MULTIPLY (E) by (E), multiply by (E), add to (E), put in column (C) 5,A

***** MATRIX OPERATIONS *****

MADD the matrix in (R),(C) size (r)x(c) to (R),(C) size (r)x(c) put in (R),(C) 10,S,W
MADD the matrix in (R),(C) size (r)x(c) to matrix in (R),(C) put in (R),(C) 8,S,W
MDEFINE the matrix in (R),(C) of size (r)x(c) to have all elements equal to (K) 5,S,W
MDIAGONAL the matrix in (R),(C) of size (r)x(c) equal to (E) on the diagonal 5,W
MEIGEN of the matrix in (R),(C) of size (r)x(c), put eigenvalues in column (C) 5,W
MEIGEN of matrix in (R),(C) size (r)x(c) put eigenvectors in (R),(C) 6,W
MEIGEN matrix in (R),(C) size (r)x(c), put values in (C), vectors in (R),(C) 7,W
MERASE the matrix in (R),(C) of size (r)x(c) $ sets every element in matrix = 0 4,S,W
MIDENTITY in (R),(C) of size (r)x(c) 4,W
MINVERT the matrix in (R),(C) of size (r)x(c), put inverse in (R),(C) 6,A,W
MKRONECKER product (R),(C) size (r)x(c) by (R),(C) size (r)x(c) put in (R),(C) 10,W
MMATVEC make by rows column (C) into the matrix in (R),(C) of size (r)x(c) 5,W
MMATVEC col vector in (R),(C) into matrix in (R),(C) of size (r)x(c) 6,W
MMOVE the matrix in (R),(C) of size (r)x(c) to (R),(C) 6,S,W
MMULT matrix (R),(C) size (r)x(c) by matrix (R),(C) size (r)x(c) put in (R),(C) 10,S,W
MMULTIPLY matrix (R),(C) size (r)x(c) by (R),(C) size (r)x(c) put in (R),(C) 10,A,W
MORTHO mat (R),(C) size (r)x(c) weights (E) put orthnormal vectors in (R),(C) 7,W
MORTHO mat (R),(C) size (r)x(c) weights (E) put in (R),(C) trans mat (R),(C) 9,W
MPRINT the matrix in (R),(C) of size (r)x(c) 4,W,P
MPRINT "L" format, the matrix in (R),(C) of size (r)x(c) 4,W,P
MPROPERTIES of the matrix in (R),(C) of size (r)x(c) 4,W,P,X
MPROPERTIES of matrix in (R),(C) size (r)x(c) put properties in column (C) 5,W,P,B
MPROPERTIES of (R),(C) size (r)x(c) put column ave's in (R),(C) 6,W,P
MPROPERTIES of (R),(C) size (r)x(c) put prop's in (C) col ave's in (R),(C) 7,W,P,B
MPROPERTIES of (R),(C) size (r)x(c) put col ave's (R),(C) row ave's (R),(C) 8,W,P
MPROPERTIES of (R),(C) size (r)x(c) put in (C), ave's in (R),(C) and (R),(C) 9,W,P,B
MRAISE the matrix in (R),(C) of size (r)x(c) to power (K), put in (R),(C) 7,S,W
(an integer can be used instead of a constant)
MSCALAR matrix (R),(C) of size (r)x(c) by constant (K), put matrix in (R),(C) 7,S,W
MSUB mat (R),(C) size (r)x(c) minus mat (R),(C) size (r)x(c) into (R),(C) 10,S,W
MSUB matrix in (R),(C) size (r)x(c) minus matrix (R),(C) put in (R),(C) 8,S,W
MSUBTRACT mat (R),(C) size (r)x(c) minus mat (R),(C) size (r)x(c) into (R),(C) 10,A,W
MSUBTRACT matrix (R),(C) size (r)x(c) minus mat (R),(C) put in (R),(C) 8,A,W
MTRANSPOSE the matrix (R),(C) size (r)x(c) into matrix in (R),(C) 6,S,W
MTRIANGULARIZE the matrix in (R),(C) size (r)x(c) into matrix in (R),(C) 6,W
MTRIANGULARIZE matrix (R),(C) size (r)x(c) in (R),(C) and inverse in (R),(C) 8,W
MVECDIAGONAL the matrix in (R),(C) of size (r)x(c), put diagonal in column (C) 5,W
MVECDIAGONAL matrix (R),(C) size (r)x(c), put col vector in (R),(C) 6,W
MVECMAT vectorize row by row matrix in (R),(C) size (r)x(c) into column (C) 5,W
MVECMAT matrix (R),(C) size (r)x(c), put vector into (R),(C) and below 6,W
MZERO the matrix in (R),(C) of size (r)x(c) 4,W,S
M(AD) mat (R),(C) size (r)x(c) times mat with (C) in diag, put mat in (R),(C) 7,W
M(AV) matrix in (R),(C) size (r)x(c) by vector in col (C) put vector in col (C) 6,W

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

M(AV) matrix in (R),(C) size (r)x(c) by column (C) put vector in (R),(C)	7,W
M(DA) mat (R),(C) size (r)x(c) premult by mat with (C) in diag, put in (R),(C)	7,W
M(V'A) matrix (R),(C) size (r)x(c), vector in col (C) put vector in row (R)	6,W
M(V'A) mat (R),(C) size (r)x(c), vector in col (C),put row vector in (R),(C)	7,W
M(XAX') a-mat in (R),(C) size (r)x(c), x-mat (R),(C) size (r)x(c), into (R),(C)	10,W
M(XX') matrix in (R),(C) of size (r)x(c) into (R),(C)	6,W
M(X'AX) a-mat in (R),(C) size (r)x(c), x-mat in (R),(C) size (r)x(c), in (R),(C)	10,W
M(X'X) matrix in (R),(C) of size (r)x(c) into (R),(C)	6,W
SMPROPERTIES of matrix in (R),(C) of size (r)x(c) put properties in column (C)	5,W,B
SMPROPERTIES of (R),(C) size (r)x(c) put column ave's in (R),(C)	6,W
SMPROPERTIES of (R),(C) size (r)x(c) put prop's in (C) col ave's in (R),(C)	7,W,B
SMPROP of (R),(C) size (r)x(c) put col ave's in (R),(C) row ave's in (R),(C)	8,W
SMPROP of (R),(C) size (r)x(c) put in (C) col ave's (R),(C) row ave's (R),(C)	9,W,B

***** N *****

NEGEXPONENTIAL of (E) put in column (C)	2
NEGEXPONENTIAL of (E), multiply by (E), add to (E), put in column (C)	4
NEW PAGE $ assures next printing will start on a new page	0
NHISTOGRAM using midpoints in col (C) and frequencies in col (C) $ no new page	2,P
NO LIST $ suppresses listing of instructions	0
NORMLAGUERRE polynomial of order (n) of col (C) put in col (C) and succ. cols.	3
NOTE $ information in hollerith card columns 7-80 is printed immediately	0,C,P
NOTE1 $ next sixty characters stored for printing first half of note	0,C
NOTE2 $ next sixty characters stored for printing second half of note	0,C
NPRINT columns (C), (C), ... (C) $ no new page, col headings or titles	V,P
NPRINT columns (C), (C), ... (C) with (K) significant digits	V,P
NPRINT cols (C)..(C) with (K) s. digits, (C)...(C) with (K) s. digits, etc.	V,P
NPRINT (K) cols, (C) with (s) s.d., (C) with (s), etc $ max width 22,3 blanks	V,P
NPRINT (K) cols, (C) with (s) s.d. and (m) max width, (C),(s),(m) ...	V,P
NPRINT (K) cols, (C) with (s) s.d. (m) max w (b) blanks, (C),(s),(m),(b) ...	V,P
NPRINT "L" format, columns (C), (C) ... (C)	V,P
NULL $ this instruction does nothing	0

***** O *****

OMNITAB $ information on card is printed as title at the top of each page	0,N,C
ONEWAY analysis for data in column (C) with group number in column (C)	2,P,X
ONEWAY for (C) with tag in (C) put statistics in (C) and next three cols	3,P
ONEWAY for (C) with (C), put tag in (C), number (C), means (C) s.d. in (C)	6,P
ORDER independently columns (C), (C) ... (C) smallest to largest	V

***** P *****

PAGE PLOT columns (C), (C) ... (C) against column (C) $ max of 6 arguments	V,P
PAGE PLOT cols (C) ... (C) vertical scale from (K) to (K) against col (C)	V,P
PAGE PLOT cols (C)...(C) against col (C), horizontal scale from (K) to (K)	V,P
PAGE PLOT cols (C)...(C) vertical (K) to (K) vs col (C) horizontal (K) to (K)	V,P

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

```
   PAGE PLOT cols (C)...(C) vs (C) horizontally (K) to (K) vertically (K) to (K)   V,P
PARPRODUCT of column (C), put partial products in column (C)                       2
PARSUM column (C), put partial sums in column (C)                                  2
PARTFUNCTION temp is (E) wave nos in (C) degens in (C) put table in (C) on         4
PERFORM instructions numbered (N) through (N), (t) times                           3,S
   PERFORM instructions numbered (N) through (N) once                              2,S
   PERFORM instruction numbered (N) once                                           1,S
PFATOMIC temp is (E) mol wt (E) wave nos in (C) degens (C) put table in (C) on     5
PFTRANSLATIONAL temp is (E) mol wt (E) put table in (C) and succ. cols             3
PLOT columns (C), (C) ... (C) against column (C)   $ max of 6 arguments           V,P
   PLOT cols (C) ... (C) vertical scale from (K) to (K) against col (C)            V,P
   PLOT cols (C)...(C) against col (C), horizontal scale from (K) to (K)           V,P
   PLOT cols (C)...(C) vertical (K) to (K), vs col (C) horizontal (K) to (K)       V,P
   PLOT cols (C)...(C) vs (C) horizontally (K) to (K) vertically (K) to (K)        V,P
POLYFIT y in col (C), using weights (E), of degree (d), predictor x in col (C)     4,P,X
   POLYFIT y in col (C), wts (E), degree (d), x in (C), put coefficients in (C)    5,P,B
   POLYFIT y in (C), wts (E), deg (d), x in (C), put coeffs in (C), residuals (C)  6,P,B
   POLYFIT (C), (E), (d), (C) put coeffs in (C), res in (C) sd of pv in (C)        7,P,B
   POLYFIT (C), (E), (d), (C) put in (C), (C), (C) and fourier coeffs in (C)       8,P,B
   POLYFIT (C), (E), (d), (C) put in (C), (C), (C), (C) vc matrix in (R),(C)      10,P,B
PRINT columns (C), (C) ... (C)                                                     V,P
   PRINT columns (C) ... (C) with (K) significant digits                           V,P
   PRINT cols (C)...(C) with (K) s. digits, (C)...(C) with (K) s. digits, etc.     V,P
   PRINT (K) cols, (C) with (s) s.d., (C) with (s), etc. $ max width 22,3 blanks   V,P
   PRINT (K) cols, (C) with (s) and max width (m), (C),(s),(m) etc.                V,P
   PRINT (K) cols, (C) with (s) s.d. (m) max w (b) blanks, (C),(s),(m),(b) etc.    V,P
PRINT "L" format, columns (C), (C) ... (C)                                         V,P
PRINT NOTE  $ causes information from note1 and note2 to be printed immediately    0,P
PRODUCT row by row of cols (C), (C), ... (C) put in col (C) $ at least 4 col nos   V
   PRODUCT of columns (C) through (C) put in column (C)                            3
PROMOTE by (r) rows, col (C) into col (C), col (C) into col (C), etc.              D
   PROMOTE all values in the worksheet by (r) rows                                 1
PUNCH data in columns (C), (C) ... (C) on hollerith cards $ 4 column limit         V
PUNCH "L" format, data in cols (C) ... (C) on hollerith cards                      V
```

***** R *****

```
RAISE (E) to power (E) and put in column (C)                                       3
   RAISE (E) to power (E), multiply by (E), add to (E), put in column (C)          5
RANKS of column (C) put in column (C)   $ smallest has rank 1                      2,B
READ data on following cards into columns (C), (C) ... (C) one card for each row   V,N,C
READ "L" format, (n) cards or rows, into columns (C), (C) ... (C)                  V,N
READ TAPE "L" unit into columns (C), (C) ... (C)                                   V,N
READ TAPE "L", "L" unit and format, into columns (C), (C) ... (C)                  V,N
REPEAT instructions numbered (N) through (N), (t) times                            3,S
   REPEAT instructions numbered (N) through (N) once                               2,S
   REPEAT instruction numbered (N) once                                            1,S
RESET nrmax to equal (r) rows   $ establishes new working length of worksheet      1,N
RESET "V" equal to (K)    $ "V" is variable V, W, X, Y or Z                        1
      (an integer can be used instead of a constant)
RESTORE instruction (N) to (E), (E) ... (E) $ no of args in instr(N) + 1           V
REWIND TAPE "L" unit                                                               0
```

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

RMS of column (C) put root mean square in column (C) 2
ROUND the numbers in column (C) to (n) significant digits and put in column (C) 3
ROW SUM columns (C), (C) ... (C) put in column (C) $ use at least 4 col nos V,S
ROW SUM columns (C) through (C) and put in column (C) 3,S
ROW SUM the entire worksheet and put in column (C) 1,S
ROWSUM columns (C), (C) ... (C) put in column (C) $ use at least 4 col nos V,S
ROWSUM columns (C) through (C) and put in column (C) 3,S
ROWSUM the entire worksheet and put in column (C) 1,S

***** S *****

SCAN only the first (c) card columns on each of the following hollerith cards 1
SCORRELATION (p) variables in (C) ... (C) put array of simple coeffs in (R),(C) V,B
SCORRELATION (p) var's in (C) ... (C) put r coeffs in (R),(C) rho in (R),(C) V,B
SEARCH col (C) equal col (C), move corresp nos in col (C) to (C), (C) to (C) etc E
SELECT in (C) nos approximating col (C) within abs tolerance (K) put in col (C) 4
SELECT in (C) nos approx col (C) to within abs tol (K) put in cols (C) to (C) 5
SELECT in (C) nos approx (C) within abs tol (K) put in (C) to (C) count (C) 6
SEPARATE from column (C) every (r) th row, start with row (R), put in column (C) 4
SET in one column (C), data on following hollerith cards 1,N,C
SET starting with row (R) of column (C) data on following hollerith cards 2,N,C
SET TAPE "L" unit into column (C) 1,N,C
SET TAPE "L" unit starting with row (R) of column (C) 2,N,C
SFIT y in (C), weights (E), (k) vars in cols (C)...(C), put coefficients in (C) V,B
SFIT (C), wts (E), to (k) in (C)...(C), put coeffs in (C) residuals in (C) V,B
SFIT (C), (E), (k), (C)...(C), put coeff in (C) res in (C) sd of pv in (C) V,B
SFIT (C), (E), (k), (C)...(C), put in (C), (C), (C) and Fourier coeff in (C) V,B
SFIT (C), (E), (k), (C)...(C), put in (C),(C),(C),(C) vc matrix in (R),(C) V,B
SHORTEN column (C) for column (C) equal to (K) put shortened cols in (C) and (C) 5,N
SI system of fundamental physical constants $ formerly mksa 0
SIN of (E) put in column (C) 2
SIN of (E), multiply by (E), add to (E), put in column (C) 4
SIND of (E) put in column (C) 2
SIND of (E), multiply by (E), add to (E), put in column (C) 4
SINH of (E) put in column (C) 2
SINH of (E), multiply by (E), add to (E), put in column (C) 4
SKIP TAPE "L" unit, forward (n) records 1
SOLVE lin eqs with coeffs in (R),(C) size (r)x(c) consts in (C), put sol in (C) 6
SONEWAY analysis for (C) with group number (C) put in (C) and next three cols 3
SONEWAY for data (C) group no (C) put in cols (C), (C), (C) and (C) 6
SORT column (C) min to max, carry along corresp values in cols (C) ... (C) V
SORT column (C) 1,S
SPACE (p) lines on printed page 1
SPACE one line on printed page 0
SPOLYFIT y in col (C), weights (E), degree (d), x in (C) put coeffs in col (C) 5,B
SPOLYFIT y in (C), wts (E), deg (d), x in (C), put coeffs in (C) res in (C) 6,B
SPOLYFIT (C), (E), (d), (C) put coeffs in (C), res in (C), sd of pv in (C) 7,B
SPOLYFIT (C), (E), (d), (C) put in (C), (C), (C) and Fourier coeffs in (C) 8,B
SPOLYFIT (C), (E), (d), (C) put in (C),(C),(C),(C) and vc matrix in (R),(C) 10,B
SQRT of (E) put in column (C) 2
SQRT of (E), multiply by (E), add to (E), put in column (C) 4
SQUARE (E) and put in column (C) 2

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

SQUARE (E), multiply by (E), add to (E), put in column (C) 4
SSTATISTICAL analysis of column (C) put statistics in (C) and next three cols 2,B
SSTATISTICAL analysis of (C), weights in (C), put in (C) and next three cols 3,B
SSTATISTICAL analysis of col (C), put statis in cols (C), (C), (C) and (C) 5,B
SSTATISTICAL analysis of (C), weights (C), put in col (C), (C), (C) and (C) 6,B
STATISTICAL analysis of column (C) 1,P,X
STATISTICAL analysis of column (C), put statis in (C) and next three cols 2,P,B
STATISTICAL anal. of (C), wts in (C), put statis in (C) and next three cols 3,P,B
STATISTICAL anal of (C) wts in (C) don't put in (-C) $ - col no. = no storage 3,P
STATISTICAL analysis of col (C), put statis in cols (C), (C), (C) and (C) 5,P,B
STATISTICAL anal. of (C) wts in (C), put statis in cols (C), (C), (C) and (C) 6,P,B
STOP $ this is last card of last set of instructions 0,C
STRUVE ONE of (E) put in column (C) 2
STRUVE ZERO of (E) put in column (C) 2
STWOWAY analysis for (r)x(c) table, data in (C) store in (C) and succ. cols 4,B
STWOWAY analysis for (r)x(c) table, data (C), store from (C) on, wts in (C) 5,B
SUB (E) from (E) and put in column (C) 3,S
SUB (E) from (E), multiply by (E), add to (E), put in column (C) 5,S
SUBTRACT (E) from (E) and put in column (C) 3,A
SUBTRACT (E) from (E), multiply by (E), add to (E), put in column (C) 5,A
SUM rows of column (C) and put sum in column (C) 2
SUM column (C), rows (R) through (R), put sum in column (C) 4
SUM col (C), rows (R), (R) ... (R), put sum in col (C) $ at least 5 args V

***** T *****

TAN of (E) put in column (C) 2
TAN of (E), multiply by (E), add to (E), put in column (C) 4
TAND of (E) put in column (C) 2
TAND of (E), multiply by (E), add to (E), put in column (C) 4
TANH of (E) put in column (C) 2
TANH of (E), multiply by (E), add to (E), put in column (C) 4
TCHEBYSHEV polynomial of order (n) of col (C) put col (C) and successive cols 3
TITLE1 $ next 60 characters printed on first half of second line 0,C
TITLE2 $ next 60 characters printed on second half of second line 0,C
TITLE3 $ next 60 characters printed on first half of third line 0,C
TITLE4 $ next 60 characters printed on second half of third line 0,C
TITLEX $ 60 characters after 2nd space are printed on horizontal axis of plot 0,C
TITLEY $ 51 characters after 2nd space are printed on vertical axis of plot 0,C
TWOWAY analysis for (r) by (c) table, data in (C), store in (C) and succ. cols 4,P,B
TWOWAY anal. for (r) by (c) table, data in (C), store from (C) on, wts in (C) 5,P,B

***** U *****

UCHEBYSHEV polynomial order (n) of col (C) put in (C) and successive cols 3
UNIFORM RANDOM numbers starting with (K) put in column (C) 2
(an integer can be used instead of a constant)

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

***** W *****

WRITE TAPE "L" unit from columns (C), (C) ... (C) V
WRITE TAPE "L" "L" unit and format, from columns (C), (C) ... (C) V

***** Z *****

ZEROS BJONE put in (C) and (C) $ if nrmax exceeds 1000 only 1st 1000 computed 2
ZEROS BJZERO put in (C) and (C) $ if nrmax exceeds 1000 only 1st 1000 computed 2

***** END *****

(C)=COLUMN number; (E)=EITHER col number or constant; (K)=CONSTANT; (N)=instr. number
(R)=ROW number; (small letter) = always integer; qualifier "L"=LETTER A,B,C,D,E or F.
A=abbreviation; B=below NRMAX; C=cannot store; D=args odd; E=args even; M=must store;
N=affect NRMAX; P=print; S=synonym; V=args variable; W=work below NRMAX; X=no storage

FORM NBS-114A (1-71)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS-TN-552	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

OMNITAB II USER'S REFERENCE MANUAL

5. Publication Date

October 1971

6. Performing Organization Code

7. AUTHOR(S)

David Hogben, Sally T. Peavy and Ruth N. Varner

8. Performing Organization

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No.

Proj. 2050131

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

Same as No. 9 above.

13. Type of Report & Period Covered

Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)

OMNITAB II, a highly user-oriented system for a large computer, is designed to make computing easy, accurate and effective, particularly for persons who are not programmers. It is a general-purpose program, which can be learned quickly, for both simple and complex numerical, statistical and data analysis. OMNITAB executes instructions written in the form of simple English sentences. Problem-solving is further enhanced by the natural structure of the system and its many features. OMNITAB has been used successfully in government, industry and universities across the country and in several centers abroad. The system has been implemented on large computers of at least seven different manufacturers.

The original version of OMNITAB has been completely rewritten to make it as machine independent as possible and to implement many improvements. This manual describes Version 5.0. Details are presented so that the user can easily find the specific information needed in any particular instance. PART A is a simple, compact introduction to OMNITAB for people who have had no experience using a large computer. PART B describes the general and special features of the OMNITAB system. PART C gives explanations, with short examples, for the use of specific instructions. PART D is a complete list of the instructions which are in the system.

17. KEY WORDS (Alphabetical order, separated by semicolons) Automatic printing; Bessel functions; Data analysis; Data manipulation; Easy and effective programming in English; List of instructions; Matrix operations; Numerical analysis; OMNITAB II user oriented computing system; Self-teaching; Statistical analysis.

18. AVAILABILITY STATEMENT

[X] UNLIMITED.

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NTIS.

19. SECURITY CLASS (THIS REPORT)

UNCLASSIFIED

21. NO. OF PAGES

264

20. SECURITY CLASS (THIS PAGE)

UNCLASSIFIED

22. Price

$2.00

USCOMM-DC 66244-P71

☆ U. S. GOVERNMENT PRINTING OFFICE : 1971 O - 446-649

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH reports National Bureau of Standards research and development in physics, mathematics, chemistry, and engineering. Comprehensive scientific papers give complete details of the work, including laboratory data, experimental procedures, and theoretical and mathematical analyses. Illustrated with photographs, drawings, and charts.

Published in three sections, available separately:

- **Physics and Chemistry**

Papers of interest primarily to scientists working in these fields. This section covers a broad range of physical and chemical research, with major emphasis on standards of physical measurement, fundamental constants, and properties of matter. Issued six times a year. Annual subscription: Domestic, $9.50; $2.25 additional for foreign mailing.

- **Mathematical Sciences**

Studies and compilations designed mainly for the mathematician and theoretical physicist. Topics in mathematical statistics, theory of experiment design, numerical analysis, theoretical physics and chemistry, logical design and programming of computers and computer systems. Short numerical tables. Issued quarterly. Annual subscription: Domestic, $5.00; $1.25 additional for foreign mailing.

- **Engineering and Instrumentation**

Reporting results of interest chiefly to the engineer and the applied scientist. This section includes many of the new developments in instrumentation resulting from the Bureau's work in physical measurement, data processing, and development of test methods. It will also cover some of the work in acoustics, applied mechanics, building research, and cryogenic engineering. Issued quarterly. Annual subscription: Domestic, $5.00; $1.25 additional for foreign mailing.

TECHNICAL NEWS BULLETIN

The best single source of information concerning the Bureau's research, developmental, cooperative, and publication activities, this monthly publication is designed for the industry-oriented individual whose daily work involves intimate contact with science and technology—*for engineers, chemists, physicists, research managers, product-development managers, and company executives.* Annual subscription: Domestic, $3.00; $1.00 additional for foreign mailing.

NONPERIODICALS

Applied Mathematics Series. Mathematical tables, manuals, and studies.

Building Science Series. Research results, test methods, and performance criteria of building materials, components, systems, and structures.

Handbooks. Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications. Proceedings of NBS conferences, bibliographies, annual reports, wall charts, pamphlets, etc.

Monographs. Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

National Standard Reference Data Series. NSRDS provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated.

Product Standards. Provide requirements for sizes, types, quality, and methods for testing various industrial products. These standards are developed cooperatively with interested Government and industry groups and provide the basis for common understanding of product characteristics for both buyers and sellers. Their use is voluntary.

Technical Notes. This series consists of communications and reports (covering both other agency and NBS-sponsored work) of limited or transitory interest.

Federal Information Processing Standards Publications. This series is the official publication within the Federal Government for information on standards adopted and promulgated under the Public Law 89–306, and Bureau of the Budget Circular A–86 entitled, Standardization of Data Elements and Codes in Data Systems.

Consumer Information Series. Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

NBS Special Publication 305, Supplement 1, Publications of the NBS, 1968-1969. When ordering, include Catalog No. C13.10:305. Price $4.50; $1.25 additional for foreign mailing.

Order NBS publications from:

Superintendent of Documents
Government Printing Office
Washington, D.C. 20402

EDGE INDEX