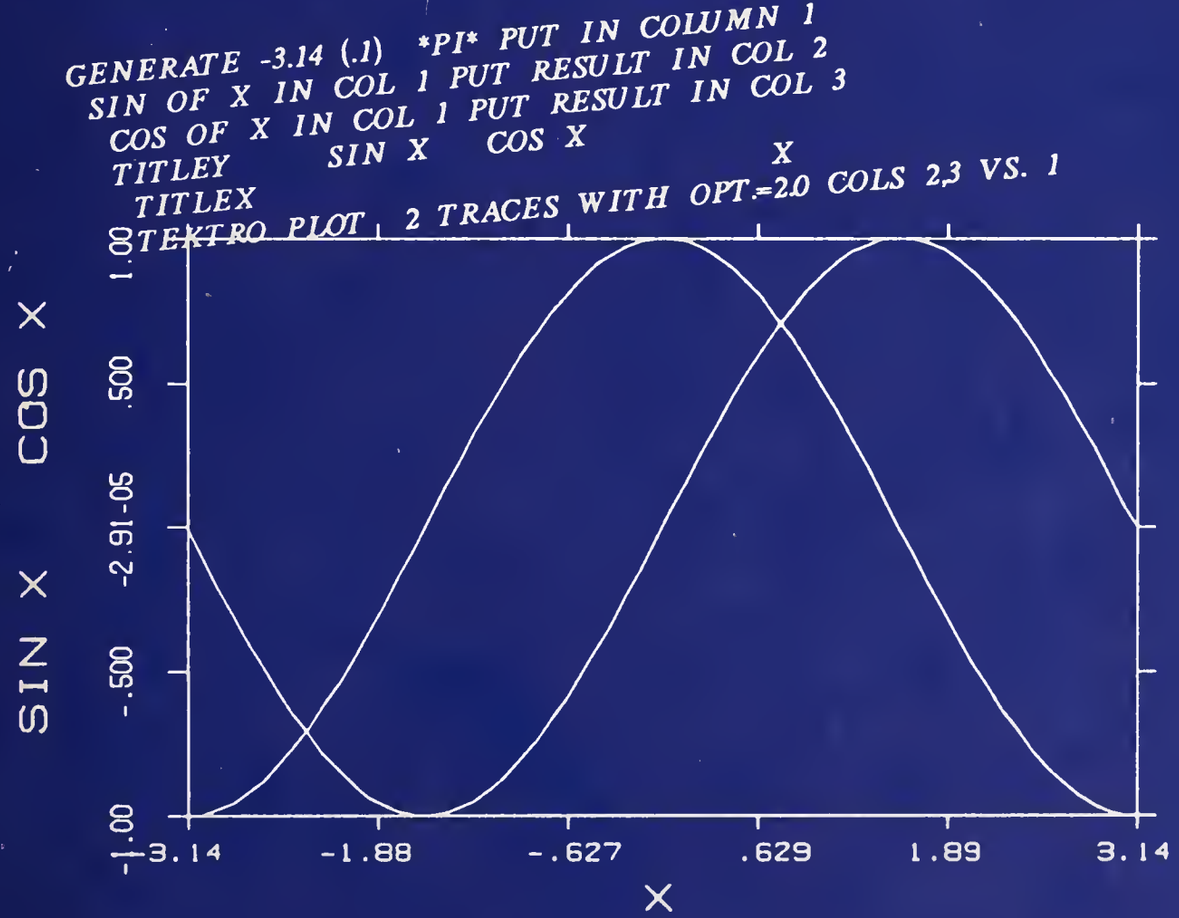
OMNITAB 80 : An Interpretive System For Statistical and Numerical Analysis / by Peavy, Bremer, Varner, Hogben.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

NBS Special Publication 701

OMNITAB 80:
An Interpretive System for Statistical and Numerical Data Analysis

Sally T. Peavy, Shirley G. Bremer, Ruth N. Varner, David Hogben



The National Bureau of Standards[1] was established by an act of Congress on 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's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Institute for Materials Science and Engineering.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[2]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

- Programming Science and Technology
- Computer Systems Engineering

The Institute for Materials Science and Engineering

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-country scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Institute consists of the following Divisions:

- Ceramics
- Fracture and Deformation [3]
- Polymers
- Metallurgy
- Reactor Radiation

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.
[2]Some divisions within the center are located at Boulder, CO 80303.
[3]Located at Boulder, CO, with some elements at Gaithersburg, MD.

NBS Special Publication 701

OMNITAB 80: An Interpretive System for Statistical and Numerical Data Analysis

Sally T. Peavy, Shirley G. Bremer, Ruth N. Varner, David Hogben*

Statistical Engineering Division
Center for Applied Mathematics
National Engineering Laboratory
National Bureau of Standards
Gaithersburg, MD 20899

*present address: 13000 Glen Mill Road
Rockville, MD 20850

Supersedes NBS Technical Note 552 and NBSIR 77-1276

Issued November 1986

U.S. Department of Commerce
Malcolm Baldrige, Secretary
National Bureau of Standards
Ernest Ambler, Director

Library of Congress
Catalog Card Number: 86-600589
National Bureau of Standards
Special Publication 701
Natl. Bur. Stand. (U.S.),
Spec. Publ. 701,
353 pages (Nov. 1986)
CODEN: XNBSAV

U.S. Government Printing Office
Washington: 1986

For sale by the Superintendent
of Documents,
U.S. Government Printing Office,
Washington, DC 20402

CONTENTS

OMNITAB 80: An Interpretive System for Statistical and Numerical Data Analysis

Sally T. Peavy, Shirley G. Bremer, Ruth N. Varner, and David Hogben

Statistical Engineering Division, National Bureau of Standards, Gaithersburg, MD 20899

OMNITAB 80 is a highly integrated general purpose programming language and statistical software computing system. The system enables the user to use a digital computer to perform statistical and numerical data analysis without having any prior knowledge of computers or programming languages. The system responds to simple instructions to obtain accurate results since reliable, varied and sophisticated algorithms for data analysis and manipulation are referenced. It may be used either interactively or in batch mode. OMNITAB 80 has been installed nationally and internationally.

OMNITAB has been completely written to make it as machine independent as possible. This document describes Version 6.0. 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. 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: Array and matrix operations; Besse[l] functions; computer language; data analysis; data manipulation; linear least squares fit; mathematical and statistical software system; numerical analysis; plotting; portable software system; probability functions; statistical analysis; tabulation; thermodynamic properties.

Introduction

OMNITAB 80 is an interpretative computing system developed and maintained by the National Bureau of Standards. OMNITAB 80 permits one to perform simple arithmetic, complex arithmetic, trigonometric calculations, data manipulation, special function calculations, statistical analysis and operations on matrices and arrays. Provision for free-field input and flexible automatic output alleviates the tedious task of formatting data for input and output. Because OMNITAB 80 is an integrated system, data and subsets of data can be analyzed in many ways within the same set of instructions. The system has extensive plotting, numerical analysis and matrix analysis capabilities.

The statistical capability of OMNITAB 80 includes: elementary analysis (oneway and twoway analysis of variance), regression (least absolute residuals estimation and selection of best subsets), correlation analysis, cross tabulation of any 14 statistics, contingency table analysis, and over 100 instructions for probability densities, cumulatives, percentiles, probability plots and random samples.

OMNITAB was conceived by Joseph Hilsenrath in the early 1960's, and its development was supported by the Thermodynamics Section in the Heat and Power Division. The heart of the OMNITAB design was the universal worksheet concept and the use of very simple English-like instructions to perform scientific calculations on data in columns of the worksheet. With the help of numerous colleagues, the language was developed to the point where it became a very effective and easy to use tool for the analysis of data. A complete description of the first Version of OMNITAB, which was programmed for the IBM 7090, is given in Hilsenrath, J.; Ziegler, G. G.; Messina, C. G.; Walsh P. J.; Herbold, R. J. *OMNITAB: A Computer Program for Statistical and Numerical Analysis*, Natl. Bur. Stand. (U.S.) Handb. 101; 1966.

The need to reprogram the system for a new computer provided the opportunity to make the system more portable and to extend its repertory of statistical instructions. The responsibility for its further development was assumed by Walter Gilbert of the Computer Services Division. Two years later the responsibility for the development and maintenance of the OMNITAB system was assumed by David Hogben and Sally T. Peavy of the Statistical Engineering Laboratory of the Applied Mathematics Division (now Statistical Engineering Division of the Center of Applied Mathematics). In the course of this revision enough additions and improvements were made to necessitate a new documentation. A new user's manual (Hogben, David; Peavy, Sally T.; Varner, Ruth N. *OMNITAB II User's Reference Manual,* Natl. Bur. Stand. (U.S.) Tech. Note 522; 1971) was completed. In 1977 a supplement, documenting additional improvements, was issued by Hogben and Peavy entitled OMNITAB II Reference Manual 1977 Supplement (NBSIR 77-1276, 1977).

In 1980 a new version of OMNITAB II was implemented within the National Bureau of Standards as Version 6.0 and OMNITAB II was renamed OMNITAB 80.

This new documentation is a composite of NBS Technical Note 552, the Supplement NBSIR 77-1276 and internal NBS OMNITAB newsletters describing new instructions and improvements to the OMNITAB 80 system.

This manual describes Version 6.0 of OMNITAB 80 which is in operation at the Gaithersburg, MD site of the National Bureau of Standards on the mainframe. Users of OMNITAB 80 Version 6.0 at other computer centers may find some minor differences. Certainly, the system control statements 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 peripheral devices (sec. C2.8) may be different. When operating in the interactive mode, the action taken when fatal errors occur is different (sec. B3.2).

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, concentration is on 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 80.

The manual is divided into four parts. Part A is a simple, compact introduction to OMNITAB 80 for people who have had no experience using OMNITAB. Part B describes the general features of the OMNITAB 80 computing system. Part C gives detailed explanations, with short examples, for the use of specific instructions. Part D is a complete alphabetical list and index 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.

This is a reference manual, not a textbook. The style is not conducive from cover to cover reading. Rather, it is designed so that the user can easily and quickly locate the information required in a specific application.

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 in the introduction. In section 6 of Part C, all the references for the statistical instructions are collected together and put in section C6.10. Considerable effort has been exerted to supply a liberal amount of cross referencing. 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, one can quickly grasp the scope of OMNITAB 80 and can then proceed at one's own pace to learn as much or as little as required. Particular attention should be given to the discussion of self-teaching in section B4.1. Sections 1.1, 1.2 and 1.3 of Part B and sections 1.1, 2.1 and 2.2 of Part C need to be studied before attempting any computing. Sections such as B1.7 and B4.4 are best read after the user has gained some experience using OMNITAB 80.

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 experienced user will primarily use Part D. Section B4.1 should also be useful.

If OMNITAB is easy to learn and use, why is this manual so long? There are several reasons. First, the manual is 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.

OMNITAB 80 is a very large system and its great success is the result of contributions of many National Bureau of Standards mathematicians, statisticians and scientists who were acknowledged in NBS Handbook 101 and NBS Technical Note 552. Credit is due to Walter J. Gilbert for converting OMNITAB to a computer independent system. David Hogben, Sally T. Peavy and Shirley G. Bremer, Statistical Engineering Division, Center For Applied Mathematics, directed the development and maintenance of the OMNITAB II and OMNITAB 80 systems. Major algorithms were contributed by James J. Filliben, Roy W. Wampler, Irene Stegun, Wesley L. Nicholson and George Marsaglia. Other members of the Statistical Engineering Division have made contributions, particularly to this manual and to the statistical instructions with automatic printing. A special thanks to Mary Carroll Croarkin, Charles P. Reeve and James R. Crichton for reading the manuscript and for their comments and corrections.

Part A: BEGINNER'S OMNITAB

The National Bureau of Standards OMNITAB 80 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 the system. 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 calculator plus a number that can not be done easily with a 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. The introduction of technical material which is not germane to the explanation of OMNITAB is unavoidable. 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, *On the distribution of first significant digits.* Annals of Mathematical Statistics, 32; 1223-1230: 1961. This provides the background for an interesting example to illustrate the basic features of the OMNITAB 80 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. The law log(n+1) and cumulative distribution imply 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.)

One may wish to 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 can be done 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. These observed frequencies can then be compared 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 familiar form. It is common practice in laboratories to write down the results of hand calculations on a 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.×(6)	diff. (2)–(7)	(8)×(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

One 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 one should expect them to appear closer to two thirds of the time.

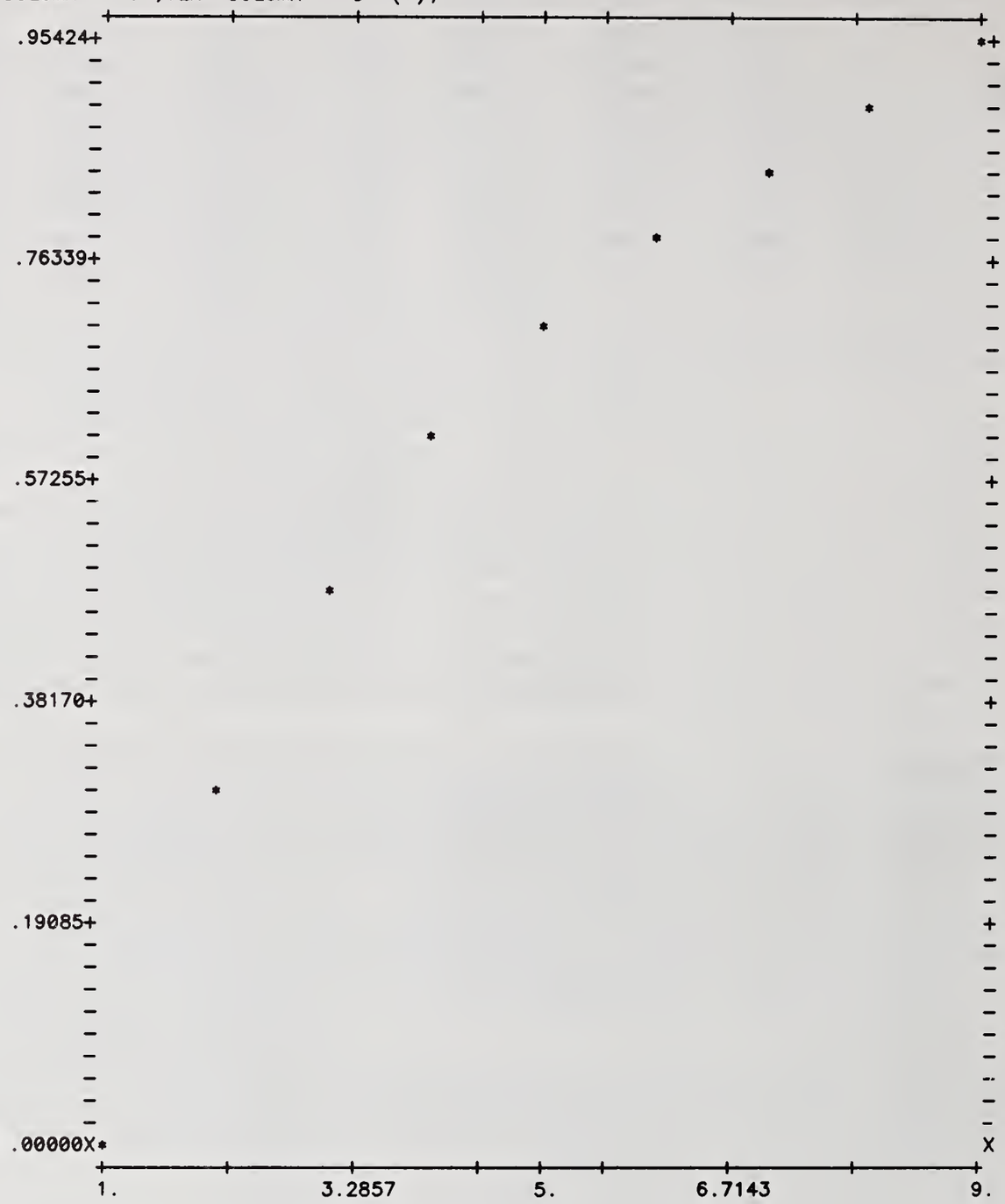
If one were to describe how to perform the above calculations using a calculator, the step-by-step instructions would be some verbalization of the headings that appear at the top of the columns. The same method is used in writing a set of instructions in OMNITAB to do the calculations on a computer.

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, writing out a set of step-by-step instructions to perform the above computations and now examine the following set of OMNITAB instructions to perform these calculations.

```
OMNITAB 80 distribution of initial digit of physical constants
WIDTH of printed page is set to 90 characters per line
GENERATE n from 1.0 in steps of 1.0 thru 9.0, put in column 1
SET initial digit of fifty 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
FREQUENCY dist. for column 21 using 9 cells, put 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 coumn 6 by 50.0 and put expected frequencies in column 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
CPLOT rel cumulative freq. in column 5 with symbol value 41. = * vs n col 1
PRINT columns 1, 2, 3, 4, 5 and 6
SPACE 2
NOTE COLUMN 7 COLUMN 8 COLUMN 9 COLUMN 10 COLUMN 11
SPACE 2
NPRINT columns 7, 8, 9, 10 and 11
STOP
```

The results obtained by using this set of instructions are shown on pages 6 - 7 . The OMNITAB results differ in a few respects from the hand-calculated results. The frequency distribution was computed and a plot of the theoretical cumulative distribution is given. The set of instructions are examined in some detail.

HOR- COLUMN 1 ,VER- COLUMN 5 (*),



COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4	COLUMN 5	COLUMN 6
1.0000000	16.000000	2.0000000	.30103000	0.	.30103000
2.0000000	11.000000	3.0000000	.47712126	.30103000	.17609126
3.0000000	2.0000000	4.0000000	.60205999	.47712126	.12493873
4.0000000	5.0000000	5.0000000	.69897000	.60205999	.096910015
5.0000000	6.0000000	6.0000000	.77815125	.69897000	.079181246
6.0000000	4.0000000	7.0000000	.84509804	.77815125	.066946790
7.0000000	2.0000000	8.0000000	.90308999	.84509804	.057991944
8.0000000	1.0000000	9.0000000	.95424251	.90308999	.051152527
9.0000000	3.0000000	10.000000	1.0000000	.95424251	.045757487

COLUMN 7	COLUMN 8	COLUMN 9	COLUMN 10	COLUMN 11
15.051500	.94850039	.89965300	.059771651	6.1281347
8.8045630	2.1954370	4.8199434	.54743697	6.1281347
6.2469367	-4.2469367	18.036471	2.8872505	6.1281347
4.8455007	.15449929	.023870031	.0049262259	6.1281347
3.9590623	2.0409377	4.1654267	1.0521246	6.1281347
3.3473395	.65266055	.42596579	.12725503	6.1281347
2.8995972	-.89959720	.80927511	.27909915	6.1281347
2.5576264	-1.5576264	2.4261999	.94861388	6.1281347
2.2878744	.71212563	.50712291	.22165680	6.1281347

LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS

```
WIDTH OF PRINTED PAGE IS SET TO 90 CHARACTERS PER LINE
GENERATE N FROM 1.0 IN STEPS OF 1.0 THRU 9.0, PUT IN COLUMN 1

     * INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION -
       NRMAX HAS BEEN RESET FROM    0 TO    9.

SET INTIAL DIGIT OF FIFTY 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

     * INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION -
       NRMAX HAS BEEN RESET FROM    9 TO   50.

FREQUENCY DIST. FOR COLUMN 21 USING 9 CELLS, PUT FREQ. IN COLUMN 2

     * INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION -
       NRMAX HAS BEEN RESET FROM   50 TO    9.

     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
CPLOT REL CUMULATIVE FREQ. IN COLUMN 5 WITH SYMBOL VALUE 41. = * VS N COL 1
PRINT COLUMNS 1, 2, 3, 4, 5 AND 6
SPACE 2
NOTE   COLUMN    7    COLUMN    8    COLUMN    9    COLUMN   10    COLUMN   11
SPACE 2
NPRINT COLUMNS 7, 8, 9, 10 AND 11
STOP

          NATIONAL BUREAU OF STANDARDS. WASHINGTON, D. C.  20234
                 OMNITAB 80  VERSION 6.03   DECEMBER 30, 1982
```

2. DISCUSSION OF EXAMPLE

Each instruction must be a single line or record with no more than 80 characters, including blanks. 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 boldface capital letters. On the remaining portion of the record, only the numbers are important and required. The words in lower case letters are descriptive words of text which help the user understand the meaning of the instruction, but are ignored by the OMNITAB system for computing purposes. Each instruction is interpreted and executed by the OMNITAB system as it is encountered. Each of the instructions is examined in some detail.

OMNITAB is the first instruction of any set. The date and other information on the record serves as a title which appears at the top of each page that is printed, along with the page number, as shown on pages 6–7. The OMNITAB instruction does a certain amount of initialization; in particular, it sets every entry in the 20162 worksheet equal to zero.

WIDTH is an instruction which specifies the maximum number of characters (including blanks) permitted per line on a printed page or per record on an output unit. Unless this is set with the WIDTH or INTERACTIVE instruction, the default value or number of characters per record is 120.

GENERATE is an instruction 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. The constants in this instruction must be written with a decimal point.

One of the instructions for entering data, such as GENERATE or SET, must be used before any arithmetic is done. As data is entered or generated into each row of the specified column, a count is made and upon completion of the instruction a scalar variable NRMAX is set equal to the count. NRMAX always contains the number of the current entries in a row in the worksheet. There are a number of instructions which may cause NRMAX to be altered. Whenever this happens, the previous value of NRMAX and current value of NRMAX are printed under the listing of the instruction modifying NRMAX.

SET is another instruction to 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. There is no indication anywhere that 50 numbers have to be entered. The OMNITAB system automatically determines the number 50. Considerable freedom is allowed in entering data. At the end of the second data record 6.0 is written, whereas all the other numbers do not have a decimal point. It is obvious when one reads 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 in the record as comments, e.g. the word "and" at the end of the second data record. Also, data may start and appear anywhere in the record. All that is required is a comma, space or word to separate the numbers.

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 A1. It gives the number of times (frequency) the digit 1 appears, the number of times the digit 2 appears and so on. This is another instruction which may change the value of NRMAX.

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 character position of the line. Instructions can appear anywhere on lines as long as one and only one instruction is on one line. Two lines 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. The two LOGTEN instructions differ in that the first LOGTEN instruction contains descriptive text. 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 instruction listed on page 8 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 text 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 6.1281347 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.

CPLOT provides a convenient graphical display of the data via the printer with the option to select the plotting symbol. The scales along the axes of the plot are automatically determined by the computer. If the command CPLOT were replaced by PLOT and the constant 41. removed, the OMNITAB system would select the plotting character. For complete detail of plotting characters see section C3.2.

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.

SPACE instruction prints blank lines. The integer number indicates the number of blank lines to be printed or number of lines to be skipped. This instruction provides flexibility in organizing printed information.

NOTE provides a way to print a line of information. The information following the command NOTE is printed as soon as the instruction is executed.

NPRINT is similar to the PRINT instruction, except printing does not start on a new page.

STOP is always the last instruction. The OMNITAB system relinquishes control of the computer back to the operating system of the computer. At the same time it causes a LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS to be printed for reference purposes.

3. A FEW SIMPLE RULES

(1) One and only one instruction of 80 characters or less is permitted per line or record.
(2) The first word of any instruction must be a valid command name in the OMNITAB vocabulary.
(3) Column numbers must be written 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 written 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 with a decimal point
k = an integer, other than a column number, without a decimal point.

SET the data on the following line(s) into column **C**
READ data into columns **C**, **C**, ..., **C** one line or record 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**

FIT Y=**C** wts=**E** with **k** var's in 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 80

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 also 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 line goes into row 1 of the designated columns, the data on the second line into the second row and so on. For the READ instruction a line of data is used for each row of data, whereas the SET instruction uses only as many lines as needed for 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; 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 C6.3.

The instruction FIT performs a linear least squares analysis on a set of data and also gives an automatic printing on a comprehensive set of results. A complete description of this instruction is given in section C6.4.

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 unless the INTERACTIVE instruction has been used. A message indicating the type of error is given in the LIST OF DATA, INSTRUCTIONS 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 B3.

In addition to the OMNITAB control commands OMNITAB and STOP, a few control statements in regards to the operating system of a specific computer are necessary for accounting and administrative purposes. These statements differ from one computer center to another. Specific details should be obtained from one's computer center. Information on how to access the computer and receive printed results should also be obtained from one's computer center.

The material provided in this part of the manual will enable one to solve a number of different problems easily and quickly. However, only the minimum has been shown of what can be done using the OMNITAB 80 system. 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 B: THE OMNITAB 80 SYSTEM

1. HOW TO USE OMNITAB 80

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 consists of one line of information with 80 or less characters per line including blanks. Each instruction is executed (except as in sec. B2) 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 consists of a line of the information 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

COMMAND DOES NOT EXIST.

A complete discussion of error messages is given in section B3. All commands, except those listed in sections B1.10 and B1.15 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 OMNITAB system except in the printing of the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS which is given at the end of the execution of a set of instructions. The use of comments is discussed in section B1.4.

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 B2.

In addition to instructions, the two other types of statements used with OMNITAB are data records and operating system control statements.

Throughout this manual commands are written in capital letters. Non-essential descriptive words in an instruction are written in lowercase letters. Descriptive words or text must not contain numerals. 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 B1.15. These qualifiers are used to distinguish between formats and units. 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 unit operation commands have either one or two qualifiers. One command, RESET "V", has the qualifier "V", where "V" denotes the letter V, W, X, Y or Z; see section B1.6.

Boldfaced letters indicate the types of arguments 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 with a decimal point
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 in scientific 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.

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. For example:

(a) An instruction can appear anywhere on the line. It does not have to start in the first position of a line as long as it is the first non-blank word.
(b) Data can be spaced in any way and descriptive words or phrases (without numerals) may appear anywhere. If a descriptive word is used at the beginning of a line, it must not be a valid command 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. For example,

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 B1.5.
(f) The OMNITAB system 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 contained in a single OMNITAB statement.

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

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

(4) Constants which are part of an instruction must have a decimal point.

(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 B1.7, 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 line must not be a stored instruction.

(12) Any data value less than -8191 must have a decimal point, e.g., -9328 must be typed 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 real numbers should have an exponent less than 10^{38}.

1.4 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 OMNITAB system to ignore all information which follows it. The dollar sign can be used to make an entire statement a comment statement by putting the dollar sign as the first non-blank character, as in:

$ the following group of instructions computes the geometric mean.

If numbers in comments occur in many statements, a simpler procedure is to use the instruction SCAN. (See sec. C1.2.) The instruction

SCAN only the first **c** characters in the following records

instructs the computer to ignore all information beyond the c-th character of each record.

1.5 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:

CHOOSE	CREAD UNIT "L"	CREAD UNIT "L" "L"	CSET UNIT "L"
DELETE	DEMOTE	DUPLICATE	ERASE
FREQUENCY	GAUSS QUADRATURE	GENERATE	INSERT
ISETUP	ISOLATE	ITERATE	OMIT
OMNITAB	READ	READ "L"	READ UNIT "L"
READ UNIT "L" "L"	RESET	RETAIN	SAMPLE WITHOUTR
SAMPLE WITHR	SET	SET UNIT "L"	SHORTEN

1.6 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 B1.7. 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 five constants are needed one can put all the constants in a column and then use asterisks as described in section B1.7. 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.

1.7 Use of Asterisks

Asterisks used in an OMNITAB instructions have a special meaning as described.

(a) Three asterisks can be used to designate an implied "through". In PRINT 1 *** 8 the three asterisks indicate that all the integers between 1 and 8 are to be used. 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.

(b) Single asterisks enable one 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.

1.8 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.

1.9 Automatic Printing

A number of instructions automatically produce the printing of a comprehensive set of results. In contrast to other packages, the user is not asked to choose between many different options. Everything is printed that the instruction produces. Invariably, the exercise of an option implies an a priori judgment 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 one 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 the statistical model. Careful use of the printed results can be educational.

All instructions with either the words PLOT, NTABLE, or TABLE as part of the command produce automatic printing. Furthermore the following instructions also provide automatic printing:

APROPERTIES	BESTCP	CONTENTS	CONTINGENCY
CORRELATION	DESCRIBE	DIFFERENCES	DIVDIFFERENCES
FIT	HISTOGRAM	MPROPERTIES	ONEWAY
POLYFIT	STATISTICAL	STEM LEAF	TWOWAY
VOCABULARY			

Some of these instructions have options which 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. A complete description of the automatic output for each of the above instructions is given in Part C.

1.10 Multiple-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. Other commands have two, three or four words and each word is separated by at least one space. In multiple word commands all the words must be used. Except, the command ROWSUM is allowed as a synonym for the command ROW SUM.

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.

1.11 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*
ABSOLUTE	ABS	ADIVIDE	ADIV
DIMENSION	DIM	AMULTIPLY	AMULT
DIVIDE	DIV	ASUBTRACT	ASUB
EXPONENTIAL	EXP	MMULTIPLY	MMULT
LOGE	LOG	MSUBTRACT	MSUB
MAXIMUM	MAX		
MINIMUM	MIN		
MULTIPLY	MULT		
SUBTRACT	SUB		

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

1.12 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)*
AADD	MADD
ADEFINE	MDEFINE
AERASE	MERASE, AZERO, MZERO
APRINT "L"	MPRINT "L"
ASUBTRACT	MSUBTRACT
INVERT	MINVERT
MAXMIN	EXTREMA
MOVE	AMOVE, MMOVE
ORDER	SORT C, only one argument
PERFORM	REPEAT, EXECUTE

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

1.13 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 = 3.1415926, 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 7.3890560 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 (Systéme 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 (Systéme Internationale)

The OMNITAB symbols for the 18 fundamental physical constants are:

ALPHA, C, CONE, CTWO, F, G, GAMMA, H, K, ME, MP, MUB, N, Q, QME, R, RINI, SIGMA.

A complete description of the physical constants with their equivalent OMNITAB names and values for both the SI and CGS units are given in section C13.2.

1.14 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)	/	(	)	'	$

Lower case alphabetic characters are interpreted as upper case. All other characters are made equivalent to $.

1.15 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 units. 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 unit 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"	READ UNIT "L"	READ UNIT "L" "L"
PRINT "L"	CREAD UNIT "L"	CREAD UNIT "L" "L"
PUNCH "L"	WRITE UNIT "L"	WRITE UNIT "L" "L"
READ "L"	SET UNIT "L"	
ABRIDGE "L"	CSET UNIT "L"	
NPRINT "L"	ENDFILE UNIT "L"	
APRINT "L"	SKIP UNIT "L"	
MPRINT "L"	BACKSPACE UNIT "L"	
RESET "V"	REWIND UNIT "L"	

2. REPEATED USE OF COMMANDS

Ordinarily, each instruction is executed as soon as it is encountered by the OMNITAB system. 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 one were 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 the above 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, as shown:

```
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 not executed. To execute the instructions one must use the instruction PERFORM, indicating which instructions are to be executed and how often. Thus, the 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 B3. 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 EXAMPLE OF INFORMATIVE DIAGNOSTIC IN PERFORM MODE

GENERATE FROM 1. IN STEPS OF 1. THRU 100. PUT IN COLUMN 1.

      * INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION
        NRMAX HAS BEEN RESET FROM        0 TO 100.

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 DIAGNOSTICS FOR THE ABOVE INSTRUCTION
        ATTEMPT TO PROMOTE FROM BELOW NRMAX -      100.
          FIRST ARGUMENT IS RESET TO -      100.
        IN INSTRUCTION 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. For example, 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

INCORRECT 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 record. If one wants to put numbered instructions right after data statements, 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 instructions that follows rather than to use a new number. See, also, section B2.3. 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 deletes 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 10 numbered instructions with numbers 1, 2, ..., 10 were rearranged in any order and then put back in the set of instructions, the results would be exactly the same as those obtained had the numbered instructions been supplied in order.

There is a limit to the number of instructions which can be stored by the OMNITAB system. The limit is rarely exceeded, but when it is the following fatal error results

TOO MANY NUMBERED INSTRUCTIONS.

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 synonymous with PERFORM.

PERFORM usually has three arguments, but may have only two or one. The first argument must be less than or equal to the second 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

The instruction numbers in a PERFORM instruction can be any valid instruction number provided a stored instruction with that number exists. 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 must exist.

PERFORM can be a stored instruction to be executed by another PERFORM instruction. When PERFORM is a numbered instruction it is referred as a PERFORM within a PERFORM. The maximum number of PERFORM commands allowed within a PERFORM is eight. An instruction cannot repeat itself.

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 stored instruction that is to be modified. The extra argument is the first one, which is the instruction number of the arguments of the modified instruction. The remaining arguments must agree in kind with the arguments of the modified instruction. 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. If asterisks are used to define an argument (see sec. B1.7) the INCREMENT instruction must also contain asterisks.

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**.

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

Some thought should be given 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 6, if the instructions were numbered as shown,

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

the results will be wrong. 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 be desirable, 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 deletes the old instruction 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 one were to read into the worksheet eight columns of data to find the averages and to repeat this 15 times. This could be done using 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 records )
PERFORM instrs 3 to 5
READ data into columns 11 *** 18
     ( data records )
PERFORM instrs 3 thru 5
     ..... etc. ( repeat last three lines 12 times )
```

See section C9.2 for further details about the INCREMENT instruction.

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

INSTRUCTION MUST BE STORED.

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

2.5 Instructions Which Cannot Be Stored

Most instructions may be stored for repeated execution. Because of their unique form, a few instructions must not be stored.

If an attempt is made to store the instructions BEGIN, EVALUATE, FINISH, OMNITAB, READ, SCAN, SET, TAPE and/or UNIT, the following fatal error is given:

INSTRUCTION IS NOT ALLOWED IN THE REPEAT MODE.

ALABEL, CONTENTS, DESCRIBE, INTERACTIVE, LABEL and/or MLABEL instructions will be ignored if an effort is made to store them and the following informative diagnostic is given:

THE INSTRUCTION WAS IGNORED BECAUSE. . .
INSTRUCTION CANNOT BE STORED.

If the instructions FORMAT "L", HEAD, NOTE, NOTE1, NOTE2, TITLE1, TITLE2, TITLE3, TITLE4, TITLEX and/or TITLEY are included in a set of instructions to be repeated, the instructions will be executed as soon as encountered, not stored, and the following informative diagnostic is printed:

THE INSTRUCTION WAS EXECUTED, BUT NOT 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. Even though 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 records, 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 starting with 1 and proceeding in steps of 1 until the last instruction, before FINISH, has been numbered. In the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS the instruction number assigned is listed on the extreme left.

The instructions may be numbered starting with some number other than 1, with 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 erased. The instructions

BEGIN

and

BEGIN 1

are equivalent.

Manually numbered instructions can be used in conjunction with BEGIN and FINISH. If they are placed between BEGIN and FINISH, the fatal error printed is

INSTRUCTIONS BETWEEN BEGIN AND FINISH CANNOT BE NUMBERED.

Any numbered instruction erases the instruction which has been given the same number by the OMNITAB 80 system. 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 sections C9.3 and C9.4. Of these instructions, COMPARE is probably the one most often used. 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:

STDDEV	RETAIN	CLOSE UP	PROMOTE
CHANGE	CHOOSE	CLOSE UP	DELETE
DEMOTE	ERASE	EXCHANGE	FLIP
OMIT	ORDER	PRODUCT	PROMOTE
RANGE	RETAIN	ROW SUM	SORT
STDDEV			

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, the following set of instructions may be used:

MDEFINE matrix in row **51** of column **1** size **1x50** is set = **1.0**
MMULTIPLY matrix in row **51**, col **1** size **1x50** by matrix in row **1**, col **11** size **50x8** put in **1,21**
ADIVIDE the array in **1,21** size **1x8 by 50.0 and put** in **1,21**
DUPLICATE 50 times the array in **1,21** size **1,8** 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 *NRMAX* for 50.0 (see sec. 1.7).

A good use of the PERFORM instruction is in the generation of ad hoc procedures. It is not uncommon to write a set of instructions which can be used in different sets of instructions at different times by different people. Often the procedure 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, the following instructions would do the job.

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 instructions could be used with

PERFORM instructions **501** thru **503**

whenever the harmonic mean of data in column 1 was needed and results stored in column 2.

The procedure 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 procedure is used. This method is particular useful if the set of instructions is lengthy or complex. With a little extra effort and cooperation among colleagues, a small local procedure library can be maintained for special purposes.

3. DIAGNOSTIC FEATURES AND ACCURACY

3.1 Diagnostic Features

After all the instructions in a set have been executed, the OMNITAB 80 system prints the

LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS

subject to certain restraints which may be imposed by the user. See LIST, NO LIST and READ "L" in sections C1.2 and C2.5. Each instruction processed is listed and any errors 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 system.

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 EXAMPLES OF ERROR DIAGNOSTICS


                    LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS


TITLE7  PLOT OF LOG OF X

     * INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION -
       TITLE NUMBER MUST BE 1, 2, 3 OR 4 AND 1 WAS USED.

GENERATE 0.0 IN STEPS OF .1 THRU 1.0 AND PUT IN COLUMN 1

     * INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION -
       NRMAX HAS BEEN RESET FROM    0 TO   11.

LOGTEN OF COLUMN 1 PUT IN COLUMN 2

    ** ARITHMETIC FAULT IN ABOVE INSTRUCTION. ZERO RETURNED   1 TIMES
       SQRT, LOG OR RAISE OF NEGATIVE NUMBER.

PLOT COLUMN 2 VS COLUMN 1
SQUARE COLUMN 2 PUT RESULTS IN COLUMN 3
INVERT MATRIX IN ROW 2 COL 1 SIZE 3 BY 3 STORE IN ROW 1 OF COL 4
 ++++ SMALLEST ERROR BOUND ON INVERTED MATRIX IS  .7-005
PRIMT COLUMNS 4, 5 AND 6
   *** FATAL ERROR IN ABOVE INSTRUCTION -
       COMMAND DOES NOT EXIST.

STOP


             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 four +'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 unless the INTERACTIVE instruction had been previously envoked. 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.

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. One exception exists where instructions will be executed after a fatal error message. If the instruction INTERACTIVE was executed prior to a fatal error, subsequent instructions will be executed after a fatal error. See section C1.4 for details.

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 DATA, INSTRUCTIONS AND DIAGNOSTICS as in the example of section B3.1. 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:

COMMAND DOES NOT EXIST.
INCORRECT ARGUMENT IN INSTRUCTION.
NRMAX=0.
NUMBER OF ARGUMENTS SHOULD BE (n).
COLUMN NUMBER(S) OUTSIDE (n) COLUMN WORKSHEET.
IMPROPER TYPE OF ARGUMENT.

The following fatal errors can occur in the use of stored instructions. See section C9 for further details.

INCORRECT STATEMENT NUMBER.
INSTRUCTION IS NOT ALLOWED IN REPEAT MODE.
INSTRUCTIONS BETWEEN BEGIN AND FINISH CANNOT BE NUMBERED.
TOO MANY NUMBERED INSTRUCTIONS.
INSTRUCTION NUMBER NOT FOUND.
NUMBERED PERFORM INSTRUCTION EXECUTES ITSELF.
INSTRUCTION MUST BE STORED.
AN INCREMENT INSTRUCTION CAN NOT INCREMENT ITSELF.

The following fatal errors can only occur in a particular instruction or group of instructions:

FIRST COMMAND MUST BE OMNITAB.
FUNCTION NAME NOT FOLLOWED BY A LEFT PARENTHESIS.
LAST TERM IN EXPRESSION IS EITHER AN OPERATOR OR A FUNCTION.
(n) IS AN INCORRECT NUMBER OF ARGUMENTS.
ALL NUMBERS IN THE COLUMN ARE THE SAME.
DIMENSIONED WORKSHEET EXCEEDS LIMIT OF (n).
ROW NUMBER(S) OUTSIDE (n) ROW WORKSHEET.
ARRAY OR MATRIX OUTSIDE (n) ROW X (n) COLUMN WORKSHEET.
INTEGER ARGUMENT IS LESS THAN −8191.
MATRIX IS (NEARLY) SINGULAR.
INSUFFICIENT SCRATCH AREA.
DEGREE IS GREATER THAN NUMBER OF NON-ZERO WEIGHTS.
NEGATIVE WEIGHTS MAY NOT BE USED.
INCONSISTENT ROW AND COLUMN NUMBERS.
MISSING OR INCORRECT FORMAT OR QUALIFIER.
EACH AND EVERY GROUP HAS ONLY ONE MEASUREMENT.
MATRIX IS NOT SYMMETRIC.
STORAGE COLUMN NUMBERS CANNOT EQUAL OTHER COLUMN NUMBERS.
ITERATIVE REFINEMENT FAILED TO CONVERGE TO A SOLUTION.
NRMAX IS (n) AND MUST BE GREATER THAN OR EQUAL TO (n).
VALUE OF SOME MEASUREMENT IS NOT ACCEPTABLE.
I, J, K AND/OR L ARE NOT DEFINED CORRECTLY, OR RULE
CANNOT DETERMINE PROPER PARAMETER VALUES FOR THESE DATA.
THE ARGUMENTS *NRMAX*, **NRMAX**, *V*, OR **V**
CAN ONLY BE INCREMENTED WITH 0.0.
FUNCTION NOT DEFINED FOR SPECIFIED PARAMETER VALUE.
LABEL OF A COLUMN (ARRAY, MATRIX) CANNOT BE A NUMBER.
LABEL MUST HAVE AT LEAST 1 AND LESS THAN 13 CHARACTERS.
NUMBER OF LABELS EXCEEDS (n) COLUMNS IN THE WORKSHEET.
LABEL IN ALABEL OR MLABEL MUST BE FOLLOWED BY 4 NUMBERS.
THE CHARACTER = IS MISSING.
THE NUMBER OF LEFT AND RIGHT PARENTHESES ARE NOT EQUAL.
ILLEGAL OPERATOR, EXPRESSION OR FUNCTION.
ALL WEIGHTS ARE ZERO.

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 15 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 sec. B4.6), 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.

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 ARITH. DIAGNOSTICS HAVE BEEN ENCOUNTERED
ANY SUCH ADDITIONAL DIAGNOSTICS FOR THIS COMMAND OR REPEAT MODE
  ARE DISREGARDED.
```

The most commmon 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 following are the 15 possible arithmetic faults which can occur:

```
SQRT, LOG OR RAISE OF NEGATIVE NUMBER.
EXPONENT TOO SMALL OR TOO LARGE.
VALUE OUT OF RANGE AND INVERSE FUNCTION CANNOT BE EVALUATED.
X TOO LARGE FOR SIN(X) OR COS(X).
VALUE SCALED TO AVOID OVERFLOW OR UNDERFLOW.
DIVISION BY ZERO.
TRIGONOMETRIC FUNCTION NOT DEFINED.
ONE OF THE VALUES COMPARED IS ZERO. ABSOLUTE TOLERANCE USED.
X FOR ELLIPTICAL INTEGRALS IS GREATER THAN OR EQUAL TO ONE.
OVERFLOW FROM USE OF THE SUM ALGORITHM.
FUNCTION NOT DEFINED FOR NEGATIVE OR ZERO VALUES.
FUNCTION NOT DEFINED FOR NEGATIVE VALUES.
FUNCTION NOT DEFINED FOR ZERO VALUES.
Y = F(X) IS NOT DEFINED FOR A SPECIFIED VALUE OF X.
FUNCTION NOT DEFINED FOR SPECIFIED PARAMETER VALUE.
```

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 OMNITAB 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.
ALL DATA WERE LOST AFTER ROW (n).
```

Either the user forgot to put in a DIMENSION instruction or the 3 was entered incorrectly.

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 B3.3, after the first 100 informative diagnostics occur, the following message is printed:

```
100 INFORMATIVE AND ARITH. 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 AFTER SET, READ OR GENERATE.
  ALL DATA WERE LOST AFTER ROW (n).
THE INSTRUCTION WAS EXECUTED, BUT CANNOT BE STORED.
VALUE REQUESTED WAS NOT FOUND.
COLUMN NUMBER INCORRECT OR NOT FOUND.
THE INSTRUCTION WAS IGNORED BECAUSE...
  ITS MEANING IS NOT CLEAR.
FUNCTION NOT DEFINED FOR SPECIFIED PARAMETER VALUE.
A VALUE OF DEGREES OF FREEDOM LESS THAN 1 WAS RESET TO 1.
A VALUE OF DEGREES OF FREEDOM WAS TRUNCATED TO AN INTEGER.
TITLE NUMBER MUST BE 1, 2, 3 OR 4 AND 1 WAS USED.
NO. ROWS NOT = TO NO. COLS. LARGEST SQUARE MATRIX WAS USED.
AN INCORRECT ASTERISK STRING IMPLYING THROUGH WAS IGNORED.
UNNECESSARY ARGUMENTS IN INSTRUCTION WERE IGNORED.
MATRIX EXTENDS BEYOND (n) ROW BY (n) COLUMN WORKSHEET.
  ONLY PART OF THE MATRIX IS STORED IN THE WORKSHEET.
INSUFFICIENT SCRATCH AREA.
NRMAX = (n) IS NOT LARGE ENOUGH TO ALLOW ITERATION.
1ST COLUMN OF ISETUP OR ISOLATE IS NOT MONOTONIC,
  OR IS CONSTANT.
ITERATION DID NOT FIND ANY ROOTS.
(n) ROW WORKSHEET IS TOO SHORT TO ACCOMMODATE ALL THE VALUES
  GENERATED BY THIS INSTRUCTION.
NO EXTREMA WERE FOUND.
A TRIAD OF X'S WITH AT LEAST TWO IDENTICAL VALUES.
  WAS FOUND AND IGNORED.
ONLY THE FIRST ARGUMENT IN THE INSTRUCTION WAS USED.
FORMAT WAS NOT FOUND. READABLE FORMAT WAS USED.
THE INSTRUCTION WAS IGNORED BECAUSE...
  ONE, SOME OR ALL WEIGHTS ARE NEGATIVE.
THE INSTRUCTION WAS IGNORED BECAUSE...
  ALL WEIGHTS ARE ZERO.
THE INSTRUCTION WAS IGNORED BECAUSE...
  VALUE OF FUNCTION IS TOO LARGE OR TOO SMALL.
COLUMN NOT LONG ENOUGH TO STORE ALL NUMBERS.
  FIRST (n) NUMBERS WERE STORED.
NOT ENOUGH DATA IN COLUMN TO RESTORE MATRIX/ARRAY.
  DATA AVAILABLE WERE USED.
THE OPTIMAL SOLUTION IS PROBABLY NOT UNIQUE.
MORE THAN 50 HEAD COLUMN INSTRUCTIONS AND/OR LABELS
  HAVE BEEN USED. ONLY THE LAST 50 HAVE BEEN RETAINED.
ATTEMPT TO PROMOTE FROM BELOW NRMAX = (n).
  FIRST ARGUMENT IS RESET TO = (n).
ATTEMPT TO DEMOTE BELOW THE (n) ROW WORKSHEET.
  DATA BELOW ROW (n) IS LOST.
X FOR ELLIPTICAL INTEGRALS IS 1.0 OR GREATER.
  THE RESULT IS SET EQUAL TO 0.0.
NEGATIVE VALUE(S) WERE ENCOUNTERED BY PARTITION FUNCTION.
  ZEROES STORED.
POSITIVE, INSTEAD OF NEGATIVE, TEMPERATURES USED.
FOR Y= F(X, THETA), X OR THETA WAS TRUNCATED TO AN INTEGER.
THE INSTRUCTION WAS IGNORED BECAUSE...
  COMMAND BEGINS WITH S AND STORAGE MUST BE REQUESTED.
NUMBER OF SIGNIFICANT DIGITS AFTER DECIMAL POINT
  HAS BEEN SET TO (n).

THE INSTRUCTION WAS IGNORED BECAUSE...
  ALL POINTS ARE OUTSIDE SPECIFIED LIMITS.
FIRST OF EQUAL ROW OR COLUMN TOTALS USED TO COMPUTE LAMBDA.
PARTIAL CORRELATION COEFFICIENTS ARE NOT DEFINED.
  MEASUREMENTS MUST EXCEED (n) VARIABLES.
THE INSTRUCTION WAS IGNORED BECAUSE...
  LOWER LIMIT OF AXIS EQUALS UPPER LIMIT.
PRINTING OF STEM AND LEAF DISPLAY IS NOT POSSIBLE.
  INSTRUCTION WAS TREATED AS SSTEM LEAF.
(N) ROWS IN WORKSHEET IS NOT ENOUGH FOR COMPLETE STORAGE.
DISPLAY IS (n) LINES, ONLY FIRST 99 DISPLAYED.
THE INSTRUCTION WAS IGNORED BECAUSE...
  WIDTH = (n) IS TOO SMALL OR TOO LARGE.
THE INSTRUCTION WAS IGNORED BECAUSE...
  WIDTH = (n) IS TOO SMALL FOR A PLOT.
WIDTH = (n) IS INSUFFICIENT FOR PAGE PLOT. BEST PLOT GIVEN.
THE INSTRUCTION WAS IGNORED BECAUSE...
  ALL NUMBERS IN THE COLUMN ARE THE SAME.
COMPUTING PROBLEMS ENCOUNTERED. RESULTS MAY BE MEANINGLESS.
THE INSTRUCTION WAS IGNORED BECAUSE...
  COLUMN WAS HEADED BY LABEL INSTRUCTION.
NRMAX HAS BEEN RESET FROM (n) TO (n).
IMPLIED THROUGH FOR LABELS MUST BE IN ALPHABETICAL ORDER.
LENGTH = (n) IS TOO LARGE AND IS RESET FOR NORMAL USE.
THE INSTRUCTION WAS IGNORED BECAUSE...
  LENGTH = (n) IS TOO SMALL FOR A PLOT.
ITERATION FAILED TO FIND AN EIGENVALUE (OR EIGENVECTOR).
  (n) UNORDERED VALUES FOUND.
NRMAX IS (n) AND MUST BE GREATER THAN OR EQUAL TO (n).
VALUE OUTSIDE ALLOWABLE RANGE. RESULT WAS SET EQUAL to 1.0.
PRINTING PROBLEM. PLEASE CALL SALLY PEAVY, 301-921-3651.
COLUMN (ARRAY, MATRIX) HAS BEEN PREVIOUSLY LABELED.
EXTRAPOLATION DONE FOR MORE THAN ONE DELTA.
ORDER OF INTERPOLATION EQUALS LIST SIZE.
ORDER OF INTERP WAS RESET DUE TO SIZE OF SCRATCH AREA.
INSTRUCTION WAS IGNORED BECAUSE...
  INSTRUCTION CANNOT BE STORED.

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 eight 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 eight significant digits and yields a result which is only accurate to two significant digits.

Small errors should not be dismissed lightly. The essential characteristic of a computer is its ability to perform millions of calculations in seconds. The small errors can accumulate into very large errors quickly. This is dramatically illustrated in the following example taken from Anscombe, F.J. *Topics in the investigation of linear relations fitted by the method of least squares.* 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

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

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 eight 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 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.

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 OMNITAB system. At times some loss in efficiency results in an attempt to provide greater accuracy. It is small comfort to know that there is a savings of one or two seconds of 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 is printed. Notable examples appear in the instructions FIT and INVERT. The FIT instruction indicates the computing accuracy of the least-squares coefficients in the automatic printing. (See sec. C6.4 for further details.) Under rather general conditions, a matrix can be inverted exactly. (See Newman, Morris *Solving equations exactly*, J. Res. Natl. Bur. Stand. (U.S.). 71B(4): 171–179; 1967.) 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 DATA, INSTRUCTIONS AND DIAGNOSTICS immediately after the listing of the command INVERT (or MINVERT).

In order to make OMNITAB as machine independent as possible, intrinsic procedures of the FORTRAN compiler of the particular computer system are used. The accuracy of system procedures varies from one computer to another. Some are quite accurate. Others are surprisingly inaccurate as was explained excellently by Cody, W.J. *Software for the Elementary Functions*, Mathematical Software Symposium, Purdue University, April 1–3; 1970. The user should not blithely assume that all eight digits are correct (or that full machine accuracy is obtained).

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 80

4.1 Self-Teaching

Despite the simplicity of OMNITAB, users may have many questions. What then? 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, the user may wish to examine the LIST OF INSTRUCTIONS in Part D to find an instruction (or a few) which (a) has not been used before, (b) may be used 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:

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 one knows that RMS stands for something like root mean square then one is 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:

GENERATE 1.(1.)10. in column **1**
FLIP column **1** into column **2**
CENSOR LEcolumn **1** for values less than or equal **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 one uses 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 sec. B1.9).

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 Peavy, Sally T.; Bremer, Shirley, G. *Test Problems and Results for OMNITAB II*. Natl. Bur. Stand. (U.S.) Tech. Note 1147; 1981 should also be of help.

The instructions CONTENTS and DESCRIBE are also useful when the OMNITAB system is used in the interactive mode. The DESCRIBE command followed by the command word of the instruction, in which one is interested, will print the instruction as given in Part D. For a complete description of CONTENTS and DESCRIBE refer to section C1.4.

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 written instruction against the number of arguments given in the column of notes 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 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 records 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. 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 the first command of each set of instructions. The OMNITAB command (see sec. C1.1) 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 UNIT "L", "L"
PRINT "L"	MPRINT "L"	READ UNIT "L", "L"
NPRINT "L"		WRITE UNIT "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 auxiliary unit operations.

The qualifier "L" represents anyone of the first six letters A, B, C, D, E or F without quotation marks. Thus, as many as six different formats can be used at one time. Note, all auxiliary unit 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 auxiliary unit 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 OMNITAB system 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 lines are not allowed. The regular OMNITAB rule that an instruction must be on a single line 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 *American National Standard Programming Language FORTRAN*, American National Standard Institute; 1978 and Meissner, Loran P.; Organick, E.I. *FORTRAN 77*, Addison-Wesley; 1980. The FORTRAN format specifications I and A 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 format specification A or I: CHOOSE, DELETE, DEMOTE, DUPLICATE, EXCHANGE, FLIP, INSERT, MMATVEC, MOVE (and AMOVE or MMOVE), MTRANSPOSE (and ATRANSPOSE), MVECMAT, OMIT, ORDER, PROMOTE, RECODE, REPLACE, SEPARATE and SORT. However, caution should be exercised with the use of the optional forms of the commands CHOOSE, DELETE, OMIT and RECODE.

4.5 Organizing a Set of Instructions

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 assistance is needed in interpreting a set of instructions some months later. Sometimes it helps to divide the columns into multiples of ten and use a separate multiple for each different logical unit of computations.

Blank lines 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 DATA, INSTRUCTIONS AND DIAGNOSTICS the presence of a blank line causes a blank line to be printed. In a lengthy set of instructions it is sometimes helpful to insert blank lines 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. Writing 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 DATA, INSTRUCTIONS AND DIAGNOSTICS in a written report, particularly if descriptive words have been used liberally.

Although an instruction can be anywhere on a line it is usually easiest to start with the first character position. If this practice is generally followed, the indenting of instructions from the left, say to character position 6, helps to offset one or more instructions from the rest of the instructions. This can be done for intermediate calculations.

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 (sec. C1.2). This will not be enough if one has more than 12,500 measurements. NBS users have the option of using SUPEROMNITAB if a large worksheet is desired. The size of the worksheet for SUPEROMNITAB is 52,500. For further details about SUPEROMNITAB call the consultant at the consultant desk. 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 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 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 lines of data and 20 numbers on each line. 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. 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 compute means and std. dev's. for 20 sets of 1000 values
DIMENSION worksheet to 1000 rows by 12 columns
1/AVERAGE column 1 put in column 11
$ AVERAGE instruction computes mean of the data in a column and puts
$   it in every row of column 11.
2/MOVE mean value in row 1 of col 11 size 1 by 1 put in row 1 col 12
$   The mean is moved to the odd row of column 12.
3/STDDEV of column 1 put in column 11
$ STDDEV instruction computes standard deviation of the data in a column and
$   puts it in every row of column 11.
4/MOVE std. dev. value in row 1 of col 11 size 1 by 1 put in row 2 col 12
$ The standard deviation is moved to the even row of column 12.
5/INCREMENT instruction 1 by 1 and 0
6/INCREMENT instruction 3 by 1 and 0
$ Modify instructions 1 and 3 so that the average and standard
$   deviation of next column can be computed.
7/INCREMENT instruction 2 by 0,0 0 by 0 2,0
8/INCREMENT instruction 4 by 0, 0 0 by 0 2,0
$ Modify instructions 2 and 4 so that mean and std. dev. may be stored
$   in the proper rows of column 12.
$
$ Instructions 1 thru 8 are stored instructions.
READ into columns 1***10

    (followed with data)

REPEAT stored instructions 1 thru 8 10 times
RESTORE arguments of instruction 1 to original arg. 1 and 11
RESTORE arguments of instruction 3 to original arguments 1 and 11
$   It is necessary to restore instructions 1 and 3 to their original form,
$   because as the instructions are modified by the INCREMENT instruction,
$   they are stored as such.If the instructions were not restored,
$   then, when the next repeat instruction is executed, the average and
$   std. dev. of col 11 and beyond will be computed instead starting with column 1.
READ into columns 1***10

    (followed with data)

REPEAT stored instructions 1 thru 8, 10 times
RESET nrmax = 40
$ Total number of data in column 12 is 40 (20 mean values, 20 std. dev. values).
SEPARATE from col 12 all mean values from every 2nd row start with row 1 put 1
SEPARATE from col 12 all std. devs. from every 2nd row start with row 2 put in 2
$ Put mean values in column 1 and std. dev. values in column 2.
RESET NRMAX = 20
PRINT columns 1 and 2
```

(b) *Row Titles.* Row titles can be obtained by reading in and printing the titles using format specification A, see section B4.4. One column should be allotted for each three characters in the title, including blank spaces. 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.

If one wishes 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 less than or = **0.0**, replace by **0.0**, store in col **15**
SUM col **15** put in column **15**

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 accuracy of data that has been entered into the worksheet.

(d) *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 sections C9.3 and C9.4 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 one wants 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, the logarithm is to be replaced 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**

(e) *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 C1.2.

PART C: DESCRIPTION OF INSTRUCTIONS

Part C is divided into 13 sections. Each section consists of related OMNITAB 80 instructions, such as plotting and statistical analysis. Sections are further divided into subsections. Within the subsection, instructions are listed alphabetically except for C6.2, C6.3, C6.4 and C6.6. Thus, anyone interested in printing, for example, can immediately turn to the appropriate subsection(s) to find which commands are available.

OMNITAB 80 instructions consist of a command name, arguments and descriptive text. Instructions are enclosed in a rectangular box. If an instruction has optional forms, each optional form is enclosed in a trapezoidal box.

The command name is given in bold face capital letters. Some command names have one or two qualifiers denoted by "**L**", where "**L**" indicates either the letter A, B, C, D, E or F. The qualifiers (without quotation marks) are part of the command name and one (or more) space character(s) must precede and follow each qualifier without any additional characters. The RESET "**V**" instruction has the qualifier "**V**", where "**V**" denotes the letter V, W, X, Y or Z.

A bold face letter indicates the type of argument (number) allowed. Lower case letters always represent integers without a decimal point. Examples are **r**=the number of rows and **c**=the number of columns. Capital bold face letters are used as follows:

C = a COLUMN number without a decimal point,
E = Either a column number or a constant,
K = a CONSTANT with a decimal point,
N = an instruction NUMBER with or without a decimal point and
R = a ROW number without a decimal point.

Descriptive text which clarifies the meaning of the instruction, command name and type of argument is printed in lower case letters. Descriptive text is non-essential and used for understanding the meaning of the instruction.

Beneath each instruction an explanation is given on how to use the instruction. For each optional form, the description highlights the differences between the primary instruction and the optional form. Comments pertaining to a group of instructions appear at the beginning of a section or subsection. Synonyms and abbreviations are not treated separately, but listed under the principal instructions.

Examples are given to further illustrate the use of a particular instruction or group of instructions. The examples are tutorial and are not necessarily given to solve a particular problem.

Diagnostics (sec. B5) 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.

1. CONTROL INSTRUCTIONS

Control instructions described in this section are used for a variety of purposes. The instructions OMNITAB and STOP are necessary to initialize parameters and to terminate execution of a set of OMNITAB instructions. Other instructions permit the user the flexibility to control the size of the worksheet, the width of input and/or printed line, the amount of automatic printing and the use of labels. CONTENTS and DESCRIBE instructions assist the user when help is needed in the interactive mode.

1.1 Essential Control Instructions

OMNITAB, STOP

These two instructions are indispensable to any set or groups of sets of OMNITAB instructions. The OMNITAB instruction must be the first in each set of instructions and initializes certain parameters. The STOP command is the last instruction in the set or group of sets.

OMNITAB descriptive identification

OMNITAB must be the first instruction in any set of instructions. The OMNITAB instruction with descriptive identification, the word PAGE and the page number are printed as a heading for each page of output. This feature permits the user to provide an identification or a title for the output generated. The amount of information printed varies if instructions INTERACTIVE (sec. C1.4) and/or WIDTH (sec. C1.5) have been invoked. More extensive titles may be printed as described in section C2.3.

The OMNITAB instruction initializes all the conditions listed below.

(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 DATA, INSTRUCTIONS, AND DIAGNOSTICS to be printed.
(7) The values of the fundamental physical constants are set in SI units.
(8) The argument of LIST **n** is set equal to 3. (See sec. C1.2.)
(9) The argument of SCAN **c** is set equal to 80. (See sec. C1.2.)
(10) All FORMATs are removed. (See sec. C2.5.)
(11) The command PRINT is set in normal mode (readable printing). (See sec. C2.4.)
(12) All TITLEs are erased. (See sec. C2.3.)
(13) All NOTEs are erased. (See sec. C2.3.)
(14) All HEADings and LABELs are erased. (See sec. C1.3 and C2.3.)
(15) WIDTH or number of characters printed per line is set to 120.
(16) LENGTH or maximum number of lines printed per page is set to 50.
(17) Previous fatal errors are nullified.

If OMNITAB is not the first instruction, the fatal error message printed is:

FIRST COMMAND MUST BE OMNITAB.

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, FIT, PLOT or MINVERT). Nonexecutable 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:

NRMAX=0.

STOP

This must be the command at the end of the last set of instructions. It signals the end of the use of the OMNITAB system and returns control of the computer to its operating system. It causes the last LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS to be printed.

1.2 Control Instructions for Flexibility

DIMENSION, LIST, NO LIST, NULL, SCAN

The instructions described below permit the user to override some of the default parameters set by the OMNITAB instruction.

DIMENSION the worksheet to have **r** rows and **c** columns

The DIMENSION instruction changes the dimension of the worksheet from 201 rows and 62 columns to the specified number of rows and columns, but it can not change the size of the worksheet which is constant. The instruction must follow immediately after the OMNITAB instruction and should never be used anywhere within 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. The product of the number of rows and number of columns must not exceed 12,500, the standard size of the worksheet. If the product **r**×**c** exceeds 12,500, the following fatal error will occur:

DIMENSIONED AREA EXCEEDS LIMIT.

The command DIM is allowed as an abbreviation for DIMENSION.

LIST n

The instruction LIST with an argument **n** controls the printing of instructions, arithmetic faults and informative diagnostics in the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS according to the value of the integer **n** as denoted below.

Action taken by OMNITAB

0. Print nothing. LIST 0 is synonymous with NO LIST (see below).
1. Suppress the printing of arithmetic faults 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 faults and informative diagnostics.

LIST 1 or LIST 2 is particularly useful when it is known that a large number of diagnostics will occur and one does not wish 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 faults will result which may be numerous if instructions are stored. The printing of these faults can be suppressed by using LIST 1 before the DIVIDE command. LIST could be used right after DIVIDE, if further diagnostics are needed.

As soon as a fatal error occurs, the effect of using LIST **n** is changed to LIST for the remainder of the set of instructions.

LIST all diagnostics

The command LIST without an argument is used to negate LIST **n** or NO LIST, see below, and all subsequent instructions, arithmetic faults and informative diagnostics are printed. The instruction LIST itself is not printed.

NO LIST

All instructions and data which appear after this instruction are not printed in the LIST OF DATA, INSTRUCTIONS, AND DIAGNOSTICS until the instruction is countermanded by LIST, see above, or until a fatal error occurs. The instruction affects only the listing of instructions, arithmetic faults and informative diagnostics. 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.

SCAN the first **c** characters of this and all following instructions

Normally, all 80 characters, including blanks, of an instruction are examined by the OMNITAB system. However, if certain information, particularly numbers, at the end of an instruction is to be ignored then the SCAN **c** instruction can be used to do this. For example, if data characters beyond the 72nd character are to be ignored use SCAN 72. The information, although not scanned, will be printed in the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS. If the SCAN instruction has more than one argument, the following informative diagnostic will be given:

ONLY THE FIRST ARGUMENT IN THE INSTRUCTION WAS USED.

1.3 Use of Labels

ALABEL, LABEL, MLABEL

The LABEL instruction allows the use of labels in instructions instead of column numbers. With the ALABEL and MLABEL instructions, it is easy to use a label instead of four (or two) arguments to specify an array or matrix.

A label may have anywhere from 1 to 12 characters. A blank (space) counts as a character. The first character of a label must be either a letter or a number. If a label has only one character, that character must be a letter (from A to Z).

The characters comma (,) and currency symbol ($) cannot be used in a label. A label cannot consist entirely of numbers. As indicated in the description of LABEL below, a minus sign (−) can be used to imply through as in X−Z. Hence, a minus sign should not be the second character of a label.

Correct labels	*Incorrect labels*
TEMPERATURE	THERMODYNAMICS
METHOD ONE	X, Y AND Z
A	+
X+1	X−1
X(1)	1976

In the ALABEL, LABEL and MLABEL instructions, labels must be separated by commas. No descriptor words are allowed in any of these instructions. The instructions may appear anywhere in a set of instructions after the OMNITAB instruction. The ALABEL, LABEL or MLABEL instruction cannot be stored.

An informative diagnostic is always printed which shows the column, array or matrix associated with each label. For arrays and matrices, four numbers are given. Two numbers are given for the position of the number in the upper left hand corner, or pivotal point, and two numbers are printed for the size. Labels, column numbers, pivotal points and sizes of arrays and matrices are enclosed in parentheses. For example:

```
LABEL X, TEMP,17, METHOD, R-T

* INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION -
     (X) IS IN COL (1), (TEMP) IS IN COL (17)
     (METHOD) IS IN COL (2), (R) IS IN COL (3)
     (S) IS IN COL (4), (T) IS IN COL (5)

ALABEL  U TIMES V, 2, 3, 4x6, X+Y, 13, 21, 8x9

* INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION -
     (U TIMES V) IS IN ROW (2) COL (3) SIZE (4x6)
     (X+Y) IS IN ROW (13) COL (21) SIZE (8x9)
```

When a label is used in any other instruction it *must* be *preceded* by one or more *blanks* and *followed* by a comma or one or more blanks. Labels cannot be erased. A label can be reassigned to another column, array or matrix by using a new ALABEL, LABEL or MLABEL instruction.

When a LABEL (or ALABEL or MLABEL) instruction is used, it is not necessary to use the label in any other instruction. The column number assigned to a label by OMNITAB, either indirectly or directly, and the label can be used interchangeably. For example, the set of instructions:

```
LABEL TEMPERATURE, PRESSURE, YIELD
READ 1 *** 3
  (data)
MULTIPLY column 2 by 100.0 and put in col 4
```

is equivalent to the instructions:

```
LABEL TEMPERATURE, PRESSURE, YIELD
READ TEMPERATURE *** YIELD
  (data)
MULTIPLY PRESSURE by 100.0 and put in col 4
```

When many column headings are desired, one or a few LABEL instructions can be used instead of a lot of HEAD instructions.

If a HEAD instruction is used for a column which is assigned to a label, the following informative diagnostic is given:

```
THE INSTRUCTION WAS IGNORED BECAUSE ...
  COLUMN WAS HEADED BY LABEL INSTRUCTION.
```

If more than 50 labels are used, the following informative diagnostic is given:

```
MORE THAN 50 HEAD COLUMN INSTRUCTIONS AND/OR LABELS
HAVE BEEN USED. ONLY THE LAST 50 HAVE BEEN RETAINED.
```

The first labels (or headings) in are the first out.

If a label is a number, the following fatal error message is printed:

```
LABEL OF A COLUMN (ARRAY, MATRIX) CANNOT BE A NUMBER.
```

If a label has more than 12 characters, the following fatal error message is printed:

A LABEL MUST HAVE AT LEAST 1 AND LESS THAN 13 CHARACTERS.

If two labels are assigned to the same column or if two arrays (matrices) are assigned the same pivotal point, the following fatal error message is printed:

COLUMN (ARRAY, MATRIX) HAS BEEN BEEN PREVIOUSLY LABELED.

No check is made to determine whether a label has been used more than once. If a label is used more than once, the last one will determine the column number or pivotal point in use.

If the column number assigned to a label is greater than the largest column number in the worksheet, the following fatal error message is printed:

NUMBER OF LABELS EXCEEDS (n) COLUMNS IN THE WORKSHEET.

where (n) is the number of columns in the worksheet.

If a label in ALABEL or MLABEL is followed by less than four arguments, the following fatal error message is printed:

LABEL IN ALABEL OR MLABEL MUST BE FOLLOWED BY 4 NUMBERS.

This error can occur if the commas between the arguments are accidently omitted or if a comma is not put after any label.

Descriptors as well as labels are allowed in all other OMNITAB instructions. However, a label must never be used as a descriptor word. Care should be exercised in using a single letter as a label, because the label might be interpreted as a format, unit or variable qualifier. In

LABEL A, Y
READ A, Y

the A in the second instruction is considered a qualifier, not a label. If the letter A in the READ instruction is meant to be a label, one or more descriptors should be inserted before it, as in

LABEL A, Y
READ data into **A** and **Y**

ALABEL label, R, C, r, c, label, R, C, r, c, ...

Each label in the instruction is followed by exactly four arguments:

R = the row locating the number in the upper left hand corner of the array,
C = the column locating the number in the upper left hand corner of the array,
r = the number of rows in the array and
c = the number of columns in the array.

These four arguments must be separated by commas, except an **x** may be used instead of a comma between the third and fourth arguments. A comma should not be placed at the end of the instruction.

When writing array (or matrix) operation instructions, it is common to put the descriptor "size" before **r x c**. Be sure not to do this in an ALABEL (or MLABEL) instruction, as descriptors are not allowed.

LABEL label, label, label, ...

Column numbers are automatically assigned to the labels. The first label is associated with column 1, the second label is associated with column 2, and so on.

Expressions such as A—Z or S—W can be used to create labels. The minus sign is used to denote through. Hence,

LABEL A—E

is equivalent to

LABEL A, B, C, D, E

Only a single letter is allowed. For example, D—G is correct, but DD—GG is interpreted as a single label. The minus sign must be the second character and should not be followed by a blank space.

The letter following the minus sign must be in alphabetical order, otherwise the following informative diagnostic is printed:

IMPLIED THROUGH FOR LABELS MUST BE IN ALPHABETICAL ORDER.

A LABEL instruction automatically uses labels for column headings. The label is centered. If the label has an odd number of characters, there will be one more blank on the left than on the right. The (first LABEL) instruction

LABEL METER

automatically invokes

HEAD column **1/WWWWMETERWWW**

where W is used to denote a blank space.

An example of the use of labels is as follows. Degrees Celsius are computed from degrees Fahrenheit using the relation C = (F—32)/1.8. If the Fahrenheit degrees are in column 1, then a set of OMNITAB instructions to perform this calculation could be

HEAD column **1/ FAHRENHEIT**
HEAD column **3/ CELSIUS**
SUBTRACT 32.0 from column **1** and put in column **2**
DIVIDE column **2** by **1.8** and put in column **3**
PRINT columns **1** and **3**

These five instructions are now exactly equivalent to

LABEL FAHRENHEIT, STORE, CELSIUS
SUBTRACT 32.0 from **FAHRENHEIT** and put in **STORE**
DIVIDE STORE by **1.8** and put in **CELSIUS**
PRINT FAHRENHEIT and **CELSIUS**

Column 1 is automatically used for FAHRENHEIT, column 2 for STORE, and column 3 for CELSIUS. The words FAHRENHEIT and CELSIUS are automatically used for column headings when the data are printed. It is also possible to use the column number associated with a label.

LABEL label, C, label, C, label, C, ...

In this form of LABEL, the column numbers are assigned by the user instead of being automatically assigned by OMNITAB. Any argument C may be omitted and the column number will be assigned by OMNITAB, using the next consecutive number after the last automatically assigned column number.

If a minus sign is used to denote through, as in A—E, the second letter may be followed by one or more arguments, separated by commas. If a single argument is used, consecutive column numbers will be assigned to the labels, starting with the specified column number. When more than one argument is used, the arguments will be used to assign column numbers to the labels. If the number of arguments is less than the number of labels, the smallest column numbers available will be assigned to the remaining labels by OMNITAB.

In the first LABEL instruction

LABEL ALPHA, 30, S—V, 25, DATA SET 2, B—D, 10, 15, 13, 2X, J—M

column numbers will be assigned to labels as follows:

label	*column number*
ALPHA	30
S	25
T	26
U	27
V	28
DATA SET 2	1
B	10
C	15
D	13
2X	2
J	3
K	4
L	5
M	6

MLABEL label, R, C, r, c, label, R, C, r, c, ...

The MLABEL instruction is synonymous with the ALABEL instruction described above.

1.4 Use of OMNITAB With Terminals

CONTENTS, CRT, DESCRIBE, INTERACTIVE, LOCAL, REMOTE, TERMINAL

The instructions described in this section are for interactive use of the OMNITAB system from terminals. The instructions LOCAL and REMOTE control the location of printing. The instructions INTERACTIVE and TERMINAL control the number of characters printed per line. CONTENTS and DESCRIBE instructions are designed to provide assistance to the interactive user. A CRT instruction temporarily halts the display on a CRT terminal at the end of each page.

CONTENTS

A CONTENTS instruction without any arguments prints a list of the major section titles of PART C of this reference manual. The following line is printed before the list:

MAJOR SECTIONS OF DESCRIPTION OF INSTRUCTIONS.

The following line is printed after the list:

TO OBTAIN SUBSECTION TITLES, TYPE CONTENTS AND SECTION NUMBER.

The CONTENTS instruction cannot be stored.

CONTENTS of section **n**

A CONTENTS instruction with an argument without a decimal point prints the subsection titles of the designated section. A title is printed before the list and a remark explaining how to obtain a list of commands is printed after the list. The instruction

CONTENTS of section **1**

would print:

SUBSECTION TITLES IN SECTION 1.

1.1 ESSENTIAL CONTROL INSTRUCTIONS.
1.2 CONTROL INSTRUCTIONS FOR FLEXIBILITY.
1.3 USE OF LABELS.
1.4 USE OF OMNITAB WITH TERMINALS.
1.5 CONTROLLING SIZE OF PAGE.
1.6 CONTROLLING AMOUNT OF PRINTING.
1.7 MULTILINGUAL OMNITAB.

TO OBTAIN LIST OF COMMANDS, TYPE CONTENTS AND SUBSECTION NO.

CONTENTS of subsection **K**

When the argument of a CONTENTS instruction has a decimal point, the instruction prints the commands of the instructions described in the specified subsection. A title is printed before the list and a remark explaining the use of a DESCRIBE instruction (see below) is printed after the list. The instruction:

CONTENTS of subsection **1.3**

would print

LIST OF COMMANDS IN SECTION 1.3 ...
LABEL, ALABEL, MLABEL

TO OBTAIN FORM OF INSTRUCTION TYPE DESCRIBE AND COMMAND.

CRT to temporarily halt printing at the end of each page

A CRT instruction temporarily halts printing at the end of each page so that the page can be reviewed, hard copied, etc. The instruction prints three periods at the end of each page. Printing will continue, if and only if, the period and the carriage return keys are depressed.

The instruction should only be used if an INTERACTIVE instruction has been used and a CRT terminal is being used.

The printing of PRINT, STATISTICAL analysis, FIT, etc. has been revised to control overflow from one page to another. This allows the effective use of the CRT instruction with a CRT display device to pause between the display of pages.

DESCRIBE XXXXXX $print the form of an instruction with command XXXXXX

The instruction prints the information in PART D of this manual for the specified command (XXXXXX). The information on the extreme right in PART D is not printed. The instruction

DESCRIBE RECIPROCAL

would print

RECIPROCAL OF E PUT IN COLUMN C
RECIPROCAL OF E, MULTIPLY BY E, ADD E, AND PUT IN COLUMN C

All optional forms of the instruction are printed. No descriptors are allowed between DESCRIBE and the command of the instruction to be described. Unfortunately, some confusion may result because lower case letters and script letters, or Greek symbols cannot be printed. A DESCRIBE instruction cannot be stored for repeated use.

INTERACTIVE use from a terminal

This instruction should *only* be used if instructions are entered from a terminal. If an error occurs, the error message will be printed immediately after the instruction is typed, in addition to being printed in the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS. The instruction can then be corrected and retyped. All instructions following an instruction with a fatal error *will* be executed, which is not the case if an INTERACTIVE instruction has not been used.

An INTERACTIVE instruction can appear at the beginning of a set of instructions and should not be stored for repeated use.

The maximum number of characters which can be printed on a line is changed from 120 to 72.

INTERACTIVE use with **c** characters per line

This instruction is the same as the one above, except it also allows one to specify the maximum number of characters printed on a line. It has the same effect of using both INTERACTIVE with no arguments and the WIDTH instruction described in section C1.5. If OMNITAB is used with a terminal or CRT with more than 72 characters per line, the instruction

INTERACTIVE 120

can be used to take full advantage of this.

The argument **n** of an INTERACTIVE **n** instruction only refers to the number of characters per line to be printed. It does not affect the number of characters per line which are read.

LOCAL printing at terminal $ no arguments

A LOCAL instruction is used to countermand a REMOTE instruction, described below, and force all printing to be done at the terminal.

REMOTE printing on remote printer $ no arguments

When using OMNITAB from a terminal, all printing performed by OMNITAB is done at the terminal. Sometimes, a set of instructions will produce a large amount of printing, in which case it may be advantageous to use a terminal to write and correct, if necessary, a set of instructions, but have the printing done on a remote printer. This is done simply by using a REMOTE instruction. The effect of using a REMOTE instruction can be negated by using a LOCAL instruction.

TERMINAL	$	no arguments

A TERMINAL instruction is equivalent to the use of

INTERACTIVE 72
BRIEF

and in addition suppresses the printing of the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS and the MESSAGE after the complete set of instructions has been executed.

The TERMINAL instruction is intended for interactive use with a terminal or similar peripheral device.

A TERMINAL instruction countermands any previous WIDTH instructions.

1.5 Controlling Size of Page

LENGTH, WIDTH

These two instructions are designed primarily for interactive use. However, they can be effectively used to control the length and/or width of any printed page.

LENGTH equal to **n** lines printed per page

A LENGTH instruction controls the number of lines per page and the size of plots. It only affects the PRINT, PLOT, PAGE PLOT, NPLOT, CPLOT and NCPLOT instructions.

For any PRINT instruction, the maximum number of printed lines per page is normally 50. The number 50 does not include two lines available for the printing of the OMNITAB instruction, two lines of titles, a one-line NOTE at the bottom of the page, or blank lines between groups of numbers. The argument of a LENGTH instruction can be set to a number smaller than 50 for short pages or to a larger number for longer pages.

Normally, a plot has 50 intervals, or 51 divisions (lines), on the vertical axis. Smaller or larger plots can be obtained by changing the value of LENGTH. The number of intervals on the vertical axis is always some multiple of ten as indicated by the formula:

$$\text{number of intervals} = 10 \text{ times the integral part of } [(n-5)/10].$$

The following table gives a value of the argument **n** of LENGTH for different plot sizes:

*Interval*s	*LENGTH* **n**
10	15-24
20	25-34
30	35-44
40	45-54
50*	55-65
60*	65-74
70*	75-84
80*	85-94
90*	95-104
100*	105-114
150*	155-164
200*	205-214
300*	305-314
400*	405-414
500*	505-514

* These sizes are only available if an INTERACTIVE instruction has been used previously.

A LENGTH instruction can be used very effectively to produce plots of different sizes. Some examples of the use of LENGTH are given in section C3.2. With a terminal, long plots can be obtained by setting the argument of a LENGTH instruction to a large number.

The argument **n** must be positive. If it is zero, negative or not specified, the following informative diagnostic is printed:

THE INSTRUCTION WAS IGNORED BECAUSE ...
ITS MEANING IS NOT CLEAR.

If the argument **n** is greater than 50 and an INTERACTIVE or TERMINAL instruction has not been used, the following informative diagnostic is printed:

LENGTH = (n) IS TOO LARGE AND IS RESET FOR NORMAL USE.

If the argument **n** of LENGTH is less than 15 and a CPLOT, NCPLOT, NPLOT, PAGE PLOT or PLOT instruction is used, the following informative diagnostic message is printed:

THE INSTRUCTION WAS IGNORED BECAUSE ...
LENGTH = (n) IS TOO SMALL FOR A PLOT.

WIDTH set to a maximum of **c** characters per line

The maximum number of characters (including blanks) which is printed on a line is **c**. This instruction is normally used when output is on a terminal, but it can also be used to adjust the maximum number of columns printed on a page and the size of plots.

The OMNITAB instruction sets **c** = 120. An INTERACTIVE instruction, without any arguments, and a TERMINAL instruction sets **c** = 72. The INTERACTIVE instruction with the argument **c** puts WIDTH **c** into effect.

A WIDTH **c** instruction determines the maximum number of columns printed on a page by the PRINT instruction as follows:

c	*Number of columns*
15 to 29	1
30 to 44	2
45 to 59	3
60 to 74	4
75 to 89	5
90 to 104	6
105 to 119	7
120	8

Thus, if **c** is 72, up to four columns per page will be printed.

The instruction is negated by an INTERACTIVE instruction and hence should be used after the INTERACTIVE instruction which normally follows the OMNITAB instruction. It has no effect on print instructions which have a formal qualifier such as PRINT "L", ABRIDGE "L", etc., nor does it control the line width for any of the WRITE TAPE or WRITE UNIT instructions. Also, it has no effect on the printing of error messages or the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS. CalComp and Tektronic plots are not affected by the use of a WIDTH or LENGTH instructions since the size of plots is determined directly by the instructions used.

The minimum value of **c** allowed is 15 and the maximum value allowed is 120. If a value of **c** smaller than the allowed minimum is used or a value larger than the allowed maximum is used, the informative diagnostic:

THE INSTRUCTION WAS IGNORED BECAUSE ...
WIDTH = (n) IS TOO SMALL OR TOO LARGE.

is printed and the number of characters per line will be 120 if the INTERACTIVE *has not* been used and 72 if the INTERACTIVE instruction *has* been used.

The size of graphs produced by CPLOT, NCPLOT, NPLOT, PAGE PLOT or PLOT instruction is changed in the horizontal direction by a WIDTH or an INTERACTIVE instruction. The number of divisions of the horizontal scale is a suitable multiple of ten, plus one for an end point, so that the plot fits in the specified width. The WIDTH instruction can thus be used very effectively to obtain plots of different sizes. A table of the number of plotting positions in the horizontal direction is given below.

c *of WIDTH* **c**	*Number of plotting positions, including end points*
59 to 68	41
69 to 78	51
79 to 88	61
89 to 98	71
99 to 108	81
109 to 118	91
119 to 120	101

If **c** is less than 59 none of the plot instructions will be executed and the following informative diagnostic will be printed:

THE INSTRUCTION WAS IGNORED BECAUSE ...
WIDTH = (n) IS TOO SMALL OR TOO LARGE.

The size of the plot produced by the PAGE PLOT instruction may or may not be affected by the WIDTH command. For **c** greater than 78 the plot printed is the same as described in section C3.1 under PAGE PLOT. If **c** is greater than 58 and less than 79, the number of plotting positions is adjusted according to the above chart, and the following informative diagnostic is printed:

WIDTH = (n) IS INSUFFICIENT FOR PAGE PLOT. BEST PLOT GIVEN.

1.6 Controlling Amount of Printing

BRIEF, FULL

BRIEF $ no arguments

An S before some commands, as in SFIT, suppresses all the automatic printing of the instruction. The instruction BRIEF suppresses only part of the automatic printing and affects only the STATISTICAL analysis, FIT, POLYFIT and CALCOMP PLOT instructions.

BRIEF suppresses all pages after the first page of the automatic printing of the STATISTICAL analysis instruction. For the FIT and POLYFIT instructions, it suppresses the printing of the first page, the last two plots of the standardized residuals, the variance-covariance matrix and the second half of the analysis of variance.

If a BRIEF instruction has been used before a CALCOMP PLOT instruction, the information on the OMNITAB instruction and the title "GRAPH XX" (where XX is the appropriate number) will not appear at the top of the CALCOMP PLOT.

BRIEF is in effect for all subsequent instructions until it is countermanded by a FULL instruction described below.

FULL	$	no arguments

A FULL instruction is used to countermand any and all BRIEF instructions which precede it.

1.7 Multilingual OMNITAB

DANSK, DEUTSCH, ENGLISH, ESPANOL, FRANCAIS,
ITALIANO, JAPANESE, NEDERLANDS, NORSK, PORTUGESE,
SLOVENE, SVENSKA, YUGOSLAV, VOCABULARY

There are 12 commands which permit instructions to be written in languages other than English. It is possible to switch from one language to another by simply inserting the appropriate command. Hence, the command ENGLISH is also given to return the system to its normal mode of operation. An additional command, VOCABULARY, prints the first six letters of each word of every command in the system for both English and the last language specified by one of these commands.

Translations for some of the languages have not been made. For other languages, the translation is tentative and subject to revision. Updating of translations will not, in general, keep up with the addition of new instructions. In all cases, translations are not given for the UNIT instructions or the Bessel function instructions.

Fatal errors and arithmetic and informative diagnostics will still be printed in English regardless of the multilingual command used.

2. ENTERING AND PRINTING DATA

Information must be entered into the worksheet before any executable instructions (such as ADD, FIT, TABLE AVERAGES, etc.) are executed. The instructions in this section describe the different options available in the OMNITAB system for entering information into the worksheet. Also discussed are the many features to output information from the worksheet onto a printed page or into auxiliary storage.

2.1 Entering Data Into the Worksheet

GENERATE, READ, SET

Data on records following the READ or SET instructions may appear anywhere on the record and either with or without a decimal point. A record may be a card or a line entered from a terminal. Data must be separated by a space, comma or word (non-numeric characters). Integers less than -8191 must have a decimal point.

Each of these commands may affect the value of NRMAX. NRMAX is reset to the number of values entered in the worksheet and, if the value of NRMAX differs from the previous value, the following informative diagnostic is printed:

NRMAX HAS BEEN RESET FROM (old value) TO (new value).

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

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 AFTER SET, READ OR GENERATE.
ALL DATA WERE LOST AFTER ROW (n).

Stored instructions should not immediately follow data records used by a READ or SET instruction because instruction numbers will be interpreted as data.

Data records entered from a terminal may consist of more than 80 characters. Only the first 80 characters including blanks of the data records are used by the READ or SET commands. If data records consist of more than 80 characters, the auxiliary unit commands must be used. (See sec. C2.8 for details.)

Other forms of READ and SET are described in sections C2.5 and C2.8.

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 upper and lower bounds specified by the odd arguments and increments specified by the even arguments. 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. 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 from following records into columns **C, C** ,..., **C** row by row

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

If any record is partially complete, zeros are entered in the remaining columns. Blank records are ignored. The value of NRMAX is changed, if necessary, to agree with the number of records read. Any extra numbers in a record are ignored. Stored instructions should not immediately follow the data records.

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 from the second record, 24, is ignored. Since there are only two numbers in the third record, the third number is set equal to zero. If NRMAX had been 5 originally, it would be reset to 3.

SET data from following records into column **C**

The numbers from the following record(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 in a separate record but can follow the previous number in the same record. Consequently, many numbers can be entered using a few records and it is often preferable to use SET when entering data into just a few columns. As in READ, the value of NRMAX is changed if necessary. Stored instructions should not immediately follow the data.

SET the data from the following records, 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. If **R** = 1, this instruction is equivalent to the SET instruction with only one argument.

2.2 Common Printing Instructions.

ABRIDGE, FIXED, FLEXIBLE, FLOATING, NPRINT, PRINT

The basic command for printing data from the worksheet is PRINT, which simply prints columns of data in "readable form", a feature unique to OMNITAB. 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 C2.3. Optional forms of the printing commands are described in section C2.4.

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 the PRINT instruction.

FIXED with **d** digits after the decimal point

The instruction forces any of the commands ABRIDGE, APRINT, MPRINT, NPRINT, or PRINT, to print numbers in a column with exactly **d** digits after the decimal point. The instruction remains in effect until countermanded by either FLEXIBLE or FLOATING (or another FIXED). A FIXED instruction with argument zero prints integers without a decimal point. The single argument **d** must be an integer between 0 and 8, inclusive. If not, **d** is set equal to 8 and the following informative diagnostic is printed:

NUMBER OF SIGNIFICANT DIGITS AFTER DECIMAL POINT
HAS BEEN SET TO 8.

FLEXIBLE to return to readable printing

Removes the effect of using either FIXED or FLOATING. It returns the output 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 than 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 or 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) three 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

bbb−7.62894+002

Some computer software may print two digits for the power ten and/or put the letter E before the sign of power ten as in

bbbb−7.62894+02

or

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 does not start printing on a new page and ignores all HEAD column commands (see sec. C2.3).

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**

Printing always starts on a new page and all NRMAX values in the specified columns are printed in "readable form" unless FIXED or FLOATING is in effect. All numbers are printed with 8 significant digits. (See sec. C2.4 for other forms of the instruction.)

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.

The word COLUMN and the column number are printed two lines above each column. For an alternative, see discussion of LABEL and/or HEAD in sections C1.3 and C2.3. Fifteen positions (1 1/2″) are needed for each column. At most 5 columns will fit on 8 1/2″ x 11″ paper. The word COLUMN starts in position 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 provide extra spacing between columns if desired.

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

If the number of arguments exceeds 8 and the WIDTH instruction has not been used, 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**

If the WIDTH command has been used prior to the PRINT command, the number of columns per block may vary. (For complete details see sec. C1.5.)

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".

2.3 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 and NOTE2 are associated and are described jointly. Similarly, the commands TITLE1, TITLE2, TITLE3 and 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 ($) (see sec. B1.4) is any one of the characters in any of the instructions in this section, it does not stop scanning of the rest of the record. It is treated like any other character. Normally, a dollar sign ($) 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 B1.14 are counted. Any characters after the 12th character, following the slash, will be ignored. The instruction can be used to provide headings for any of the commands FIT, FOURPLOTS, MPROP, ONEWAY, PLOT, POLYFIT, PRINT, STEM LEAF, STATIS, TABLE, TWOPLOTS and TWOWAY. 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 instruction, the column heading will follow VER- and HOR-. The column heading will also be used by the commands FIT and POLYFIT in the title and data headings.

The HEAD instruction may be used in conjuction with the first two optional forms of PRINT described in section C2.4. It can not be used with a PRINT "L" instruction. The column heading will be ignored and no diagnostic will be given.

Labels defined by the LABEL instruction are also used as headings. See section C1.3 for complete details.

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

MORE THAN 50 HEAD COLUMN INSTRUCTIONS AND/OR LABELS
HAVE BEEN USED. ONLY THE LAST 50 HAVE BEEN RETAINED.

The instruction 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:

THE INSTRUCTION WAS EXECUTED, BUT CANNOT BE STORED.

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

COLUMN NUMBER INCORRECT OR NOT FOUND.

NEW PAGE

This instruction assures that 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 on this line is printed immediately

One blank space should follow the command NOTE. The characters following NOTE through the 80th character, 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 sec. B1.9).

The instruction NOTE cannot 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

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

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, the blank after NOTE1 and the next 59 characters of NOTE1 are printed; 60 of NOTE2.

Since the information contained in NOTE1 and NOTE2 are automatically stored, the instructions cannot be in a repeat mode.

PRINT NOTE

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. The number of characters printed may be less than 120 if the instructions TERMINAL or WIDTH (see sec. C1.4 and C1.5) have been used prior to the PRINT NOTE instruction.

SPACE **p** lines on printed page

The specified number **p** of blank lines appear on a printed page. This instruction 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 instructions 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. The number of characters printed on line 2 and 3 may be less than 120 if the instructions TERMINAL or WIDTH have been evoked. (See sec. C1.4 and C1.5.)

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

TITLE NUMBER MUST BE 1, 2, 3 OR 4 AND 1 WAS USED.

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

THE INSTRUCTION WAS EXECUTED, BUT CANNOT BE STORED.

2.4 Optional Forms of Readable Printing

ABRIDGE, NPRINT, PRINT

The basic forms of ABRIDGE, NPRINT and PRINT were described in section C2.2. 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.

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 and/or LABEL instructions are 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 INSTRUCTION WAS IGNORED BECAUSE ...
ITS MEANING IS NOT CLEAR.

Each option has an argument which specifies the number of digits to be printed. If the argument is less than 1 or exceeds 8, it is automatically reset to 1 or 8 and 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 for each value of all the columns printed 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 that instruction is (4**K**+1). The arguments in each set, all integers without a decimal point, 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 3. The total number of arguments is (3**K**+1). 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 10 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**=11, the actual width would be 10. On the other hand, if **m** were less than 10, 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.

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 instructions with example of the fourth option:

PRINT 4.0 cols **31 8 0** floating **32 4 −13** fixed **33 8 22** readable **34 0 −9** integer
NOTE 234567801234567890123456789012345678901234567890123456789012345678901234567

would give the following results. The NOTE instruction prints digits under the results so that the reader can see how many blank spaces are printed and where they are located.

```
 COLUMN 31  COLUMN 32          33          34
 -1.2345678+00          12.3457  123.45678        1235
234567890123456789012345678901234567890123456789012345678901234567
```

The argument **b** was not defined explicitly, so **b**=3. The total column widths, including blanks, are 16, 16, 13, and 12. (The total width is 57.) The word COLUMN does not appear above columns 33 and 34 because the widths, excluding blanks, are only 9 and 10 respectively.

2.5 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 C2.6, C2.7 and C2.8.

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":

MISSING OR INCORRECT FORMAT OR QUALIFIER.

The following informative diagnostic appears with any PRINT "L" instruction:

FORMAT WAS NOT FOUND. READABLE FORMAT WAS USED.

The LABEL and HEAD instructions described in sections C1.3 and C2.3 are ignored when printing according to a specified format. The options described in section C2.4 are unavailable.

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 record or line. For example, for the instructions

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

the data on record 1 would go into row 1 of columns 1–20, the data on record 2 would go into row 1 of columns 21–40 and the data on record 3 would go into row 1 of columns 41–45. The remaining 24=3x8 records 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 specified columns according to the indicated format. (See sec. B1.15 for description of "L".)

FORMAT "L" ( )

The FORTRAN format specifications are placed inside the parentheses. The format specifications must be contained in one input record. Also, the format should specify only one line of printing.

NPRINT "L" format, columns **C, C ,..., C**

This instruction bears the same relation to PRINT "L" 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. None of the special features of PRINT are available.

READ "L" format, **n** rows into columns **C, C ,..., C**

This instruction is similar to the READ instruction except for each row of the indicated columns one or more records are read in accordance with the specified format. Note carefully, this instruction has one more argument than READ. The first argument determines how many rows of data are to be entered into the columns. If this argument is zero (0), then reading of data will continue until either a record of zeros or all blanks is read. The data records after READ "L", unlike READ, are not listed in the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS, instead the following message is printed:

(n) DATA CARD(s) READ BUT NOT LISTED

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 sec. B2.5) when READ "L" is stored for repeated use.

2.6 Printing Arrays and Matrices.

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

The commands in this section are used to print data from any part of the worksheet as long as the rows and columns are consecutive. Printing does not have to start with row 1 nor is it limited to data above NRMAX. All other commands pertaining to array and matrix operations are described in sections C10 and C11.

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 instructions 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 C2.5 for further discussion on the use of formats. Headings and labels are ignored by all four of the commands described here.

If the commands INTERACTIVE, TERMINAL or WIDTH are used prior to the APRINT or MPRINT commands, the maximum number of columns per page will be less than or equal to 8. For the maximum number of columns printed per page for different widths see section C1.5 under the WIDTH command.

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:

ARRAY OR MATRIX OUTSIDE (n) ROW x (n) COLUMN WORKSHEET.

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

APRINT the array in **R**, **C** of size **rxc**

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 **rxc**

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

MPRINT the matrix in **R**, **C** of size **rxc**

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 7 (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 **rxc**

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

2.7 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 cards

This instruction produces 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 consists of three blanks, a minus sign if the number is negative, otherwise an additional blank, the first significant digit, the decimal point, the last six significant digits, a plus or minus sign denoting sign of the exponent of 10, and 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 are punched according to the indicated format. Unlike the instruction PUNCH, this instruction allows the punching of more than four columns.

2.8 Use of Peripheral Devices

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

Data is normally entered into the worksheet either from punched cards or from lines typed from a terminal. The instructions described in this section enable one to use magnetic tapes, drums, discs, or any other medium. The average user of OMNITAB should seek additional assistance from his or her system programmer when working with magnetic tapes or discs. Throughout this section the word UNIT in an instruction will denote a tape, disc or drum, etc. The word TAPE can be used instead of UNIT as indicated at the end of this section under synonyms.

Except for UNIT, each command has two words and at least one qualifier. The second word is always the word UNIT or TAPE. The qualifier, see section B1.15, is always one of the six letters A, B, C, D, E, or F (without quotation marks) and refers to the unit being used. If there are two qualifiers, the first qualifier refers to the unit and the second qualifier refers to the corresponding FORMAT "L".

Information on any unit or tape is grouped into logical records within blocks. Logical records may be thought of as lines on a page and blocks as pages. Unless a UNIT instruction is used, the number of characters per logical record is 80 and the number of records per block is 1. If the data is in block form or if more than 80 characters per record are to be read, a UNIT instruction must be used to indicate the number of characters per logical record and, if necessary, the number of logical records per block.

Input instructions (CREAD UNIT, CSET UNIT, READ UNIT and SET UNIT) may affect the value of NRMAX.

BACKSPACE UNIT "L" r logical records

This instruction backspaces unit "L" by r logical records. If a UNIT instruction has not been used, the instruction backspaces r blocks. However, if a UNIT instruction was used, the number of blocks backspaced is:

$$r/r' + m,$$

where r′ is the number of logical records per block and:

$m = 1$, if r/r′ has a remainder
$m = 0$, if r/r′ has no remainder.

The unit is always positioned at the beginning of a block.

CREAD UNIT "L", using r logical records, into columns C, C ,..., C

This instruction is similar to the READ UNIT "L" command, except r (r>0) specifies the number of logical records to be read instead of terminating input with a logical record of zeros as in READ UNIT "L". The data within a block, not read, are lost.

Data will not be printed with the LIST OF DATA, INSTRUCTIONS and DIAGNOSTICS. Instead the following informative diagnostic is printed:

(n) DATA RECORDS READ BUT NOT PRINTED.

CREAD UNIT "L" "L" format, using n blocks, into cols C ,..., C

Similar to the above instruction, except a FORMAT "L" is referenced and similar to the READ UNIT "L" "L" instructions, except n (n>0) blocks of data are read.

If the instruction FORMAT "L" with the same qualifier has not been used before, then the following fatal error occurs:

MISSING OR INCORRECT FORMAT OR QUALIFIER.

CSET UNIT "L", using r logical records, into column C

This instruction is similar to the SET UNIT instruction. Here, the indicated number of logical records r (r>0) is read, whereas the SET UNIT command reads data until a logical record of zeros is encountered. If all the data from a block are not read, the rest of the information will be lost.

Data read will not be printed with the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS. Instead the following informative diagnostic is printed:

(n) DATA VALUES READ BUT NOT PRINTED.

CSET UNIT "L", using r logical records, into row R of column C

Similar to above instruction, except data is entered into row R and below.

ENDFILE UNIT "L"

This instruction writes an end-of-file mark on the specified unit, "L". It is necessary to ENDFILE a unit after all the data have been written on that unit. Otherwise, when the data are read at some future time, extraneous information may be entered into the worksheet.

READ UNIT "L" into columns C, C ,..., C

This instruction is similar to the READ command, except data are read from the specified unit. Reading of data continues until a logical record of zeros is encountered (not a blank logical record). The execution of this instruction always starts at the beginning of a new block. Data for any particular row of all the columns specified must be completely contained within a logical record. If all the data from a block are not used, they are lost. Subsequent READ UNIT or CREAD UNIT instructions start with a new block.

Data read will not be printed with the LIST OF DATA, INSTRUCTIONS and DIAGNOSTICS. Instead the following informative diagnostic is printed:

(n) DATA RECORDS READ BUT NOT PRINTED.

READ UNIT "L" "L" format into columns C, C ,..., C

Same as above instruction, except unit "L" is read using the specified format. Reading of data is under format control and not under the control of OMNITAB. Data will be entered into the specified columns until some row contains all zeros (i.e., a block of zeros). This instruction always starts at the beginning of a new block for each row of data entered. Data for any particular row may be in one or more blocks.

One must be aware of exactly how logical records are blocked and must set up the format accordingly. Generally information should be read with the same format as it was written.

If the instruction FORMAT "L" with the same qualifier has not been used before, then the following fatal error occurs:

MISSING OR INCORRECT FORMAT OR QUALIFIER.

REWIND UNIT "L"

The unit specified by "L" is positioned to the beginning of the unit.

SET UNIT "L" into column C

This instruction is similar to a SET instruction, except data are read from the specified unit. Reading of data continues until a logical record of zeros is encountered, not a blank logical record. Extra data, if any, in the block are lost.

Data read will not be printed with the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS. Instead the following informative diagnostic is printed:

(n) DATA VALUES READ BUT NOT PRINTED.

SET UNIT "L" starting with row R of column C

Same as above, except first datum is entered into row R instead of row 1.

SKIP UNIT "L", forward r logical records

The specified unit "L" is moved forward **r** logical records. The instruction is just like a BACKSPACE UNIT instruction, except the unit is moved forward instead of backwards. The unit is positioned at the beginning of a block.

UNIT with c characters per logical record

The number of characters per logical record is set equal to **c**. The number of logical records per block is assumed to be one.

UNIT with c characters logical record and r logical records block

The number of characters per logical record is set equal to **c** and the number of logical records per block is set equal to **r**. This instruction should be used with blocked data. The instruction controls the number of characters per logical record and the number of logical records per block for all qualifiers. Thus, if two media are used with differing values of **c** and **r**, a different UNIT instruction must be used before the use of each medium.

Theoretically, **cxr** could be less than or equal to the size of the worksheet, 12500. However, since FORTRAN input/output procedures are used, a systems programmer should be consulted to determine how large **cxr** can be. For NBS users **cxr** < 133 and no check is made to see that this number is not exceeded.

WRITE UNIT "L" from columns C, C ,..., C

A WRITE UNIT instruction is to tapes, discs or drums what PUNCH is to cards. Data is written on the unit "L" with **c** characters per logical record with **r** logical records per block. See the description of UNIT instruction for a definition of **c** and **r**. The default values of **c** and **r** are 80 and 1, respectively. Each row of data starts a new logical record. Let i=the number of columns specified in the instruction and j=integral part of **c**/15. Then the number of logical records written for each row is the integral part of i/j + k, where k=0, if i/j has no remainder and k=1, if i/j remainder is not zero. If the last logical record contains less than **c** characters, it will be packed with blank characters. A row of zeros is written after all the rows of data.

If the last block has only m logical records (m<**r**), then the last **r**−m logical records will contain zeros. The instructions FIXED and FLOATING have no effect.

WRITE UNIT "L" "L" format, from columns C, C ,..., C

Same as above, except blocks are written according to the referenced format. The output from this instruction is under format control and is not controlled by the OMNITAB system. Each slash (/) in the format specification or each scan of the format from the beginning will generate a new block. Furthermore, a row of zeros will be written under format control after all the data are written.

Synonyms

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", TAPE, WRITE TAPE "L", WRITE TAPE "L" "L"

3. PLOTTING DATA

The instructions in this section describe the extensive plotting capability of OMNITAB 80, which includes plots with easy to read scales, plots of varying sizes, multiple plots per page, CALCOMP plots and Tektronix plots. Histograms, stem-and-leaf displays, probability plots and four statistical plots on a page are routinely used for statistical analysis.

3.1 Basic Plotting Instructions

NPLOT, PAGE PLOT, PLOT

The three instructions, NPLOT, PAGE PLOT and PLOT enable the user to plot data using the on-line printer. The description of these instructions assumes that the instructions INTERACTIVE, LENGTH and/or WIDTH have not been used (i.e., width is 120 characters per line and length is 52 lines printed per page). If the instructions INTERACTIVE, LENGTH and/or WIDTH have been envoked prior to the plot instructions, the size of the plot will vary. For complete details in regard to the size of the plot, see sections C1.4 and C1.5. The last two instructions 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 8 1/2x11″ piece of paper.

The four different options for each of these instructions control the scale of the vertical and/or horizontal axes. Each plot has 51 positions vertically and 101 horizontal positions for NPLOT and PLOT and 61 horizontal positions for PAGE PLOT. On each scale, plus signs are printed at every tenth position (left to right, top to bottom). Minus signs are printed in the other positions. Six values are printed along each axis at equal intervals. 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 HOR (horizontal) and VER (vertical) axes, either "COLUMN C" if HEAD or LABEL instructions have not been used or the heading if HEAD or LABEL instructions have been used. In addition, the plotting symbol(s) used are printed.

Any number of columns (functions) can be plotted against the single abscissa. Plotting symbols are assigned as follows:

Columns	*Plotting Symbols*
1st thru 5th	. * + , −
6th thru 15th	0 thru 9
16th thru 41st	A thru Z, respectively

and the pattern is repeated if the plot instructions contain more than 41 columns. Where two or more symbols 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 main form, 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 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.

Titles on the horizontal and vertical axes may be added to the plots at the user's discretion with the use of the TITLEX and TITLEY instructions. See section C3.4 for details on the use of these instructions.

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

NPLOT columns **C**, **C** ,..., **C** versus column **C**

A NPLOT instruction has the same relation to a PLOT instruction as a NPRINT instruction has to a PRINT instruction. A NPLOT instruction does not print the plot on a new page. It has no use unless a LENGTH instruction is used with an argument less than 50. If a LENGTH instruction is used it is possible to get more than one plot on the same page. The optional forms of PLOT exist for NPLOT. See PLOT instruction below for complete description of main and optional forms of the instruction.

NPLOT columns **C**, **C** ,..., **C** vertical scale from **K** to **K** versus column **C**

NPLOT cols **C** ,..., **C** versus column **C** with horizontal scale from **K** to **K**

NPLOT columns **C** ,..., **C** vertical **K** to **K** vs col **C** horizontal **K** to **K**

NPLOT columns **C** ,..., **C** vs col **C** horizontal **K** to **K** vertical **K** to **K**

PAGE PLOT columns **C**, **C** ,..., **C** against column **C**

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. The instructions TITLE2, TITLE3 and TITLE4 are ignored by the instruction PAGE PLOT.

PAGE PLOT columns **C** ,..., **C**, vertical scale **K** to **K**, against column **C**

PAGE PLOT columns **C** ,..., **C** against column **C**, horizontal scale **K** to **K**

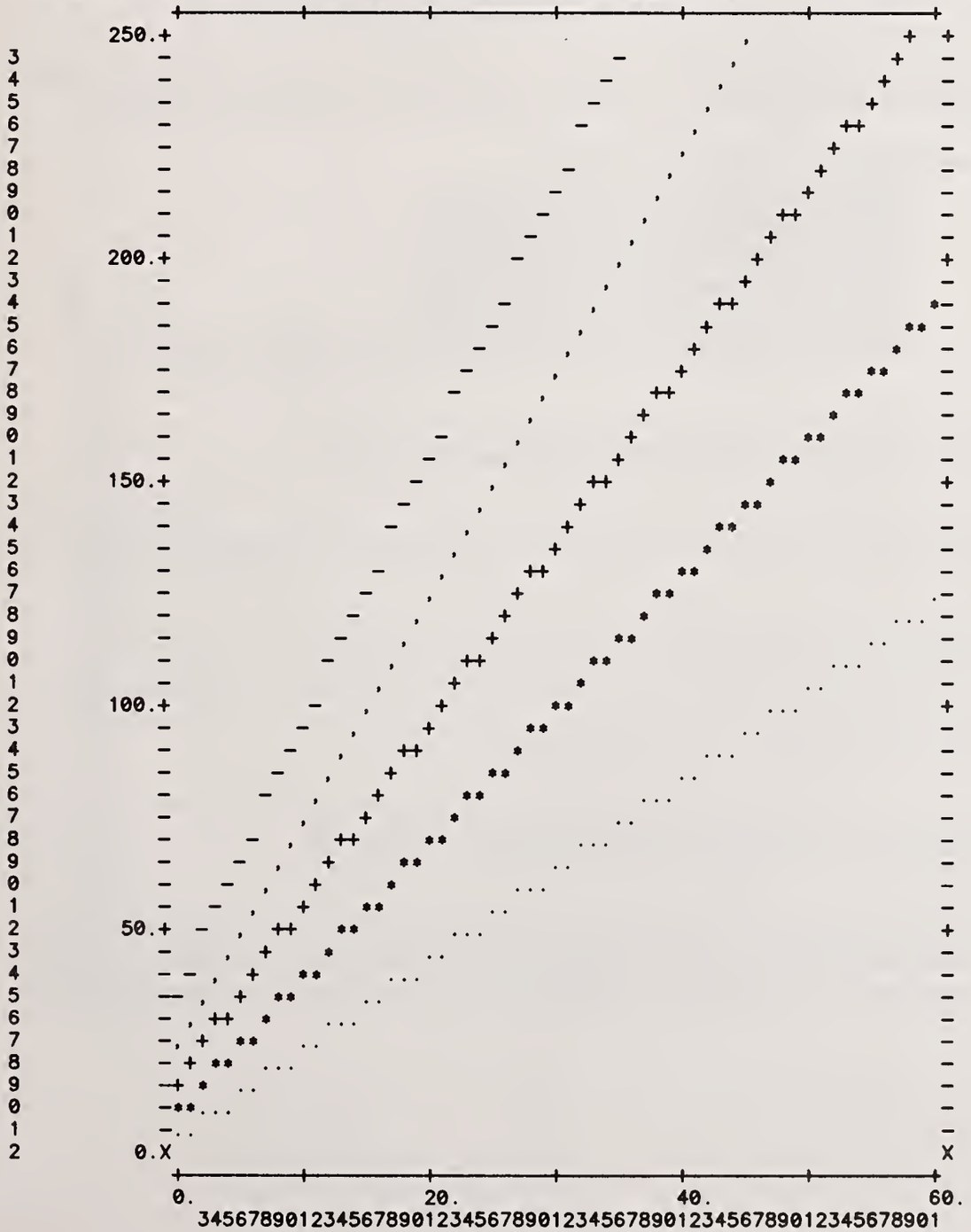
PAGE PLOT cols **C** ,..., **C** vertical **K** to **K** vs col **C** horizontal **K** to **K**

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

Page 75 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 $x=0.0(1.0)100.0$. 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 80 EXAMPLE OF PAGE PLOT
     GENERATE 0. (1.0) 100.0 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 4TH. 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 COL 11 12 13 14 15 VS COL 1 HOR FROM 0. TO 60., VER FROM 0. TO 250.
```

```
OMNITAB 80 EXAMPLE OF PAGE PLOT                                                    PAGE    1
 PAGE PLOT OF FIVE STRAIGHT LINES USING 5TH. OPTION
HOR- COLUMN    1
VER- COLUMN   11  (.), COLUMN   12  (*), COLUMN   13  (+), COLUMN   14  (,), COLUMN   15  (-),
POINTS PLOTTED  263, POINTS NOT PLOTTED - OUT OF BOUND  242
```



PLOT columns **C**, **C** ,..., **C** against column **C**

Instruction plots columns C,C ,..., C against a single column specified by the last argument. Both the horizontal and vertical scales are determined by OMNITAB. The instruction is easy to use and requires no thought. The options below provide more flexibility.

PLOT columns **C** ,..., **C**, with vertical scale **K** to **K**, against column **C**

This option is identical to the first, except the range of the vertical scale is chosen by the user rather than determined by the OMNITAB system. 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 77 shows an example of the use of the second form of the PLOT instruction. The functions sine, cosine, tangent and cotangent are plotted for x = −1.0(0.1)1.0. The actual set of instructions used was:

```
OMNITAB 80    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
```

PLOT cols **C** ,..., **C** against column **C** with horizontal scale from **K** to **K**

The range of the horizontal scale is selected by the user and the vertical scale is automatically determined.

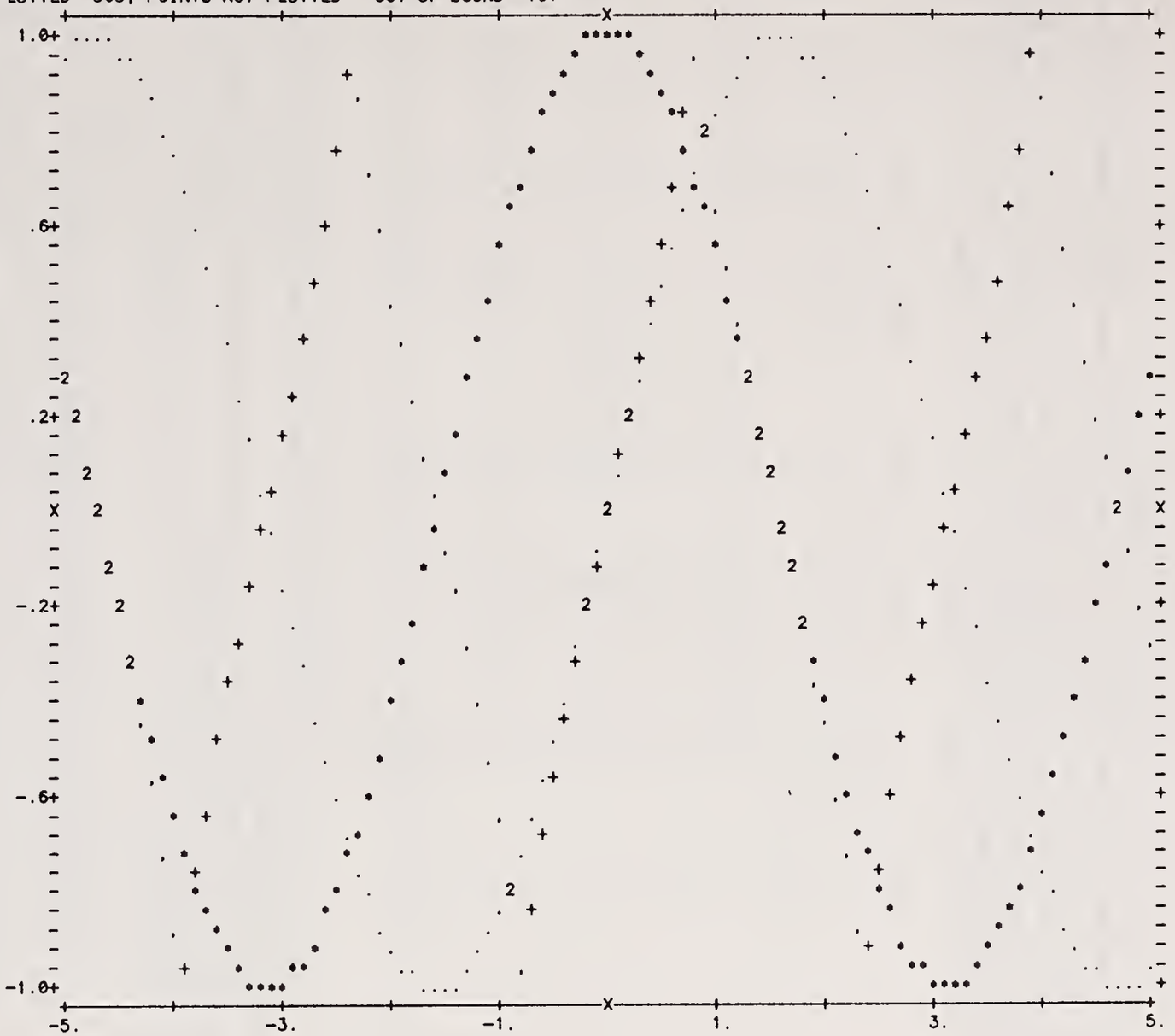
PLOT columns **C** ,..., **C** vertical **K** to **K** vs column **C** horizontal **K** to **K**

Both the vertical scale and the horizontal scale are selected by the user.

PLOT columns **C** ,..., **C** vs column **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. Unlike the previous instruction, the two arguments which specify the horizontal scale come *before* the two arguments which specify the vertical scale.

HOR- COLUMN 10 ,VER- COLUMN 1 (.), COLUMN 2 (*), COLUMN 3 (+), COLUMN 4 (,).
POINTS PLOTTED 303, POINTS NOT PLOTTED - OUT OF BOUND 101



3.2 Character Plots

CPLOT, NCPLOT

These two instructions are similar to NPLOT and PLOT instructions except the plotting symbol is specified instead of being automatically assigned. Much greater flexibility is available in the use of plotting symbols.

The argument which specifies the plotting symbol always is placed immediately after the column number designating the vertical axis. The argument can be either a constant or a column number. If it is a constant with a decimal point, the specified symbol is used for each point on the curve. If the argument is a column number, the symbol specified in row one is used to plot the point in row one, the symbol specified in row two is used to plot the point in row two, and so on. This use of CPLOT can be very powerful. If the points fall into logical subsets such as for different laboratories, different test methods, etc., then a different plotting symbol can be used for each subset.

Numbers are used to indicate the required plotting symbol in accordance with the following table.

Symbol	*Constant*	*Symbol*	*Constant*	*Symbol*	*Constant*
0	0	G	17	W	33
1	1	H	18	X	34
2	2	I	19	Y	35
3	3	J	20	Z	36
4	4	K	21	/	37
5	5	L	22	.	38
6	6	M	23	-	39
7	7	N	24	+	40
8	8	O	25	*	41
9	9	P	26	(	42
0	10	Q	27	)	43
A	11	R	28	,	44
B	12	S	29	blank	45
C	13	T	30	=	46
D	14	U	31	$	47
E	15	V	32	'	48
F	16				

Note, 0.0 or 10.0 can be used for the symbol zero. Whenever more than one point falls on the same plotting position, the last symbol is used.

If the number specifying a plotting symbol is less than zero or greater than 48, a CPLOT or NCPLOT instruction will be executed as a PLOT instruction.

The instructions INTERACTIVE, LENGTH and WIDTH may affect the size of the plots (see sec. C1.4 and C1.5 for details).

CPLOT column **C**, symbol **E** ,..., column **C**, symbol **E** versus column **C**

The CPLOT instruction and its optional forms are analogous to the PLOT instruction and its optional forms.

CPLOT col **C**, symbol **E** ,..., **C E** vertical scale from **K** to **K** vs col **C**

CPLOT col **C**, symbol **E** ,..., **C E** vs col **C** horizontal scale **K** to **K**

CPLOT C, E ,..., C E vertical K to K vs C horizontal K to K

CPLOT C, E ,..., C E vs C horizontal K to K vertical K to K

NCPLOT C, E ,..., C, E versus column C

A NCPLOT instruction has exactly the same relation to a CPLOT instruction as a NPRINT instruction has to a PRINT instruction. In other words, a NCPLOT instruction does not plot on a new page. Clearly, it should be used in conjunction with a LENGTH instruction. The optional forms of PLOT exist for NCPLOT as shown below.

NCPLOT column C symbol E ,..., C, E vertical scale K to K vs column C

NCPLOT C, E ,..., C, E versus column C horizontal scale K to K

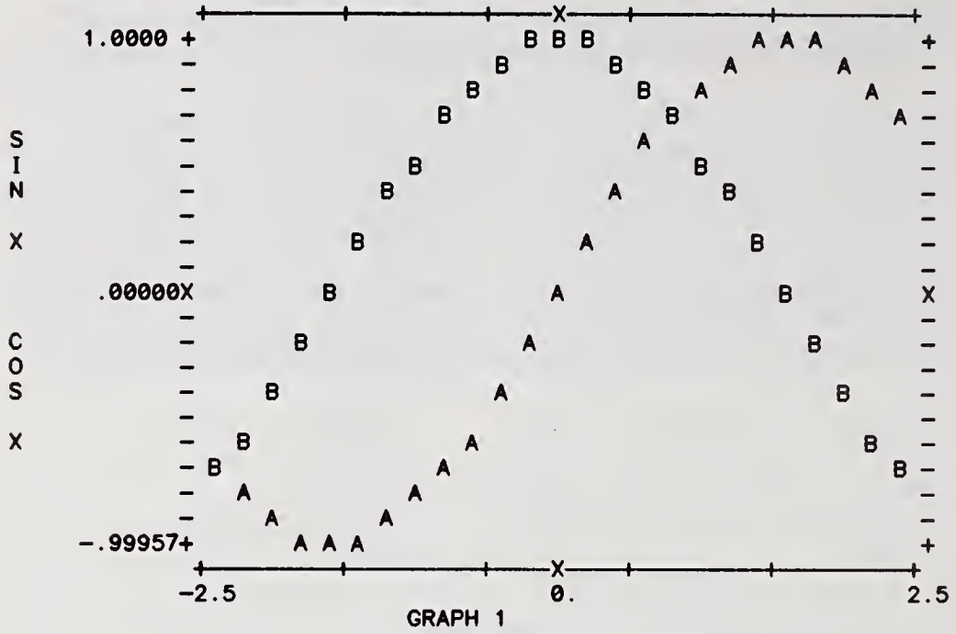
NCPLOT C, E ,..., C, E vertical K to K versus C horizontal K to K

NCPLOT C, E ,..., C, E versus C horizontal K to K vertical K to K

The following set of instructions is used to illustrate the use of CPLOT, NCPLOT (described above), WIDTH and LENGTH (described in sec. C1.5.). The results are shown on page 80.

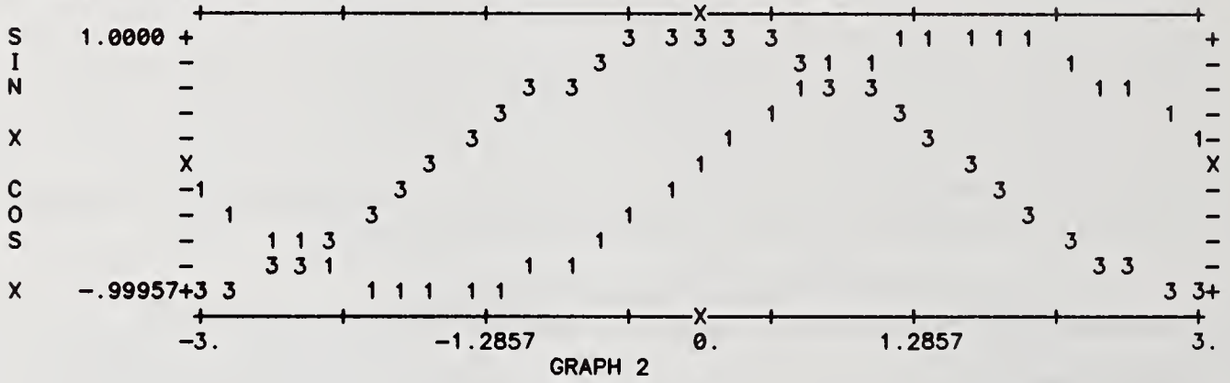
```
OMNITAB     80     EXAMPLE OF USE OF CPLOT AND NCPLOT
LABEL  X, SINE, COSINE
GENERATE    -4.  (.2) 4.0 IN X
SIN OF X  PUT IN SINE
COS OF X PUT IN COSINE
TITLEY SIN X       COS X
TITLEX                    GRAPH 1
WIDTH 75 CHARACTERS
LENGTH 30 LINES
CPLOT SINE WITH SYMBOL 11. AND COSINE WITH SYMBOL 12. VS X FROM -2.5 TO +2.5
SPACE 5
WIDTH 90 CHARACTERS
LENGTH 20 LINES

TITLEY SIN X COS X
TITLEX                         GRAPH 2
NCPLOT SINE WITH SYMBOL 1. AND COSINE WITH SYMBOL 3. VS X FROM -3.0 TO 3.0
STOP
```

```
HOR-       X     ,VER-     SINE     (A),    COSINE    (B),
POINTS PLOTTED   50, POINTS NOT PLOTTED - OUT OF BOUND   32
```



GRAPH 1

```
HOR-       X     ,VER-     SINE     (1),    COSINE    (3),
POINTS PLOTTED   62, POINTS NOT PLOTTED - OUT OF BOUND   20
```



GRAPH 2

3.3 Plots With Nice Scales

NICE CPLOT, NICE NCPLOT, NICE NPLOT, NICE PLOT

The instructions NICE CPLOT, NICE NCPLOT, NICE NPLOT, and NICE PLOT perform the same functions as the instructions CPLOT, NCPLOT, NPLOT and PLOT, except scale limits are chosen so that plotted curves occupy roughly two-thirds of the plot and the numbers on the scales are easy to read. Optional forms in which the user specifies the limits are unavailable. Although the instructions do not necessarily produce optimum results, they do eliminate the need to carefully specify limits in order to get scales which are easy to read.

NICE CPLOT column **C** symbol **E** ,..., column **C** symbol **E** versus column **C**

Similar to a CPLOT instruction described in section C3.2, except instead of using the maximum and minimum to determine scale limits, limits are chosen to make the scale easy to read with convenient intervals.

NICE NCPLOT column C symbol E ,..., column C symbol E versus column C

Similar to a NICE CPLOT instruction, except printing does not start on a new page. The instruction is particularly useful when used with a LENGTH instruction described in section C1.5.

NICE NPLOT of columns C, C ,..., C against column C

Similar to a NICE PLOT instruction, except printing does not start on a new page. The instruction is particularly useful when used with a LENGTH instruction described in section C1.5.

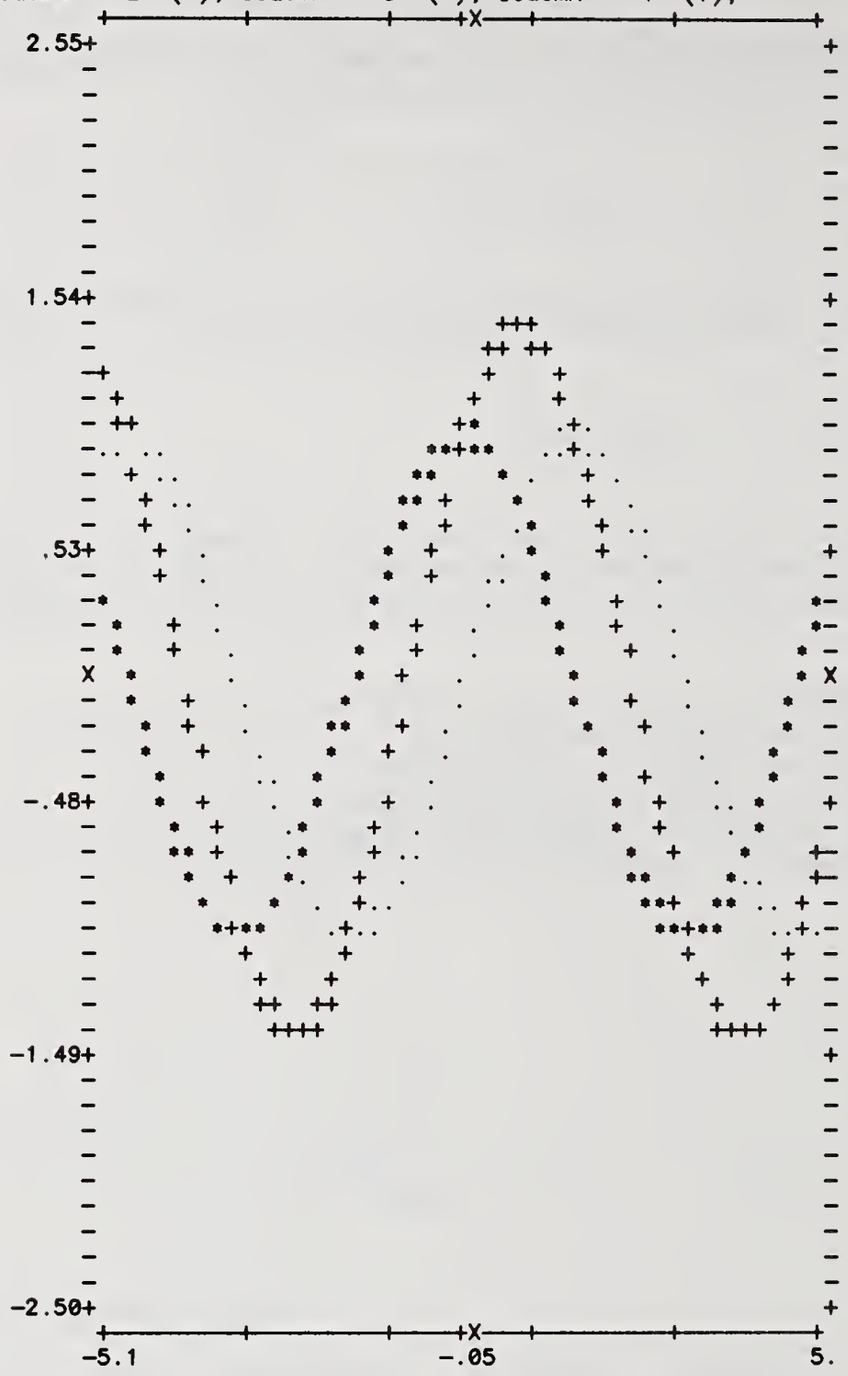
NICE PLOT columns C, C ,..., C against column C

Similar to PLOT instruction described in section C3.1, except scale limits are chosen to make the scale easy to read with convenient intervals instead of using the maximum and minimum to determine limits.

The following set of instructions is similar to the set in section C3.1 for the PLOT instructions and produces the plot shown on page 82.

GENERATE NOS -5.0 IN STEPS OF 0.1 THRU 5.0 IN COLUMN 1
WIDTH 70 CHARACTERS PER LINE
SIN OF COLUMN 1 PUT IN COLUMN 2
COS OF COLUMN 1 PUT IN COLUMN 3
ADD COLUMN 2 TO 3 PUT IN COLUMN 4
NICE PLOT COLUMNS 2, 3, AND 4 VERSUS COLUMN 1

HOR- COLUMN 1
VER- COLUMN 2 (.), COLUMN 3 (*), COLUMN 4 (+),



3.4 Titles for Plots

TITLEX, TITLEY

These instructions enable the user to add titles on the horizontal (TITLEX) and vertical (TITLEY) axes. They are described jointly.

TITLEX $ 60 characters following command are printed on horizontal axis

TITLEY $ 51 characters following command are printed on the vertical axis

One space 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 first space after the X of TITLEX will be printed, centered, below the horizontal axis (x-axis or abscissa) on every subsequent plot. The 51 characters following the first space after Y of TITLEY will be printed vertically to the left of the vertical axis (y-axis or ordinate) of any subsequent plot. 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 instructions LENGTH and/or WIDTH may affect the number of characters printed on the horizontal axis and/or vertical axis.

The TITLEX and TITLEY instructions can 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 (sec. C2.3), an informative diagnostic is given.

3.5 Control of Size of Plot

LENGTH, WIDTH

A detailed description of these instructions is given in section C1.5. They are included in this section for completeness.

LENGTH equal to **n** lines printed per page

WIDTH set to a maximum of **c** characters per line

3.6 Multiple Plots per Page

FOURPLOTS, TWOPLOTS

These instructions allow multiple plots to be printed on a single page instead of a page for each plot. Each plot has 21 by 51 divisions or 20x50 intervals. A title

PLOT OF xxxxxxxxxxxx VERSUS xxxxxxxxxxxx

appears at the top of each plot, where xxxxxxxxxxxx is either the word COLUMN followed by the appropriate column number or the column heading if a HEAD or LABEL instruction has been used. The number of points plotted and the number of points not plotted because they are out of bounds are printed at the bottom of each plot. The plotting symbol is a period (.). If two or more points coincide, the plotting symbol is an asterisk (*).

The first column number, C, in any pair is always used for the vertical axis variable and the second column number, C, in any pair is always used for the horizontal axis variable.

FOURPLOTS of **C** versus **C**, **C** versus **C**, **C** versus **C**, **C** versus **C**

The instruction allows four plots to be printed on a single page, instead of on four separate pages. This makes a visual comparison much easier and conserves paper.

If the value of width, as determined by a WIDTH, INTERACTIVE or TERMINAL instruction, is less than 120, the first two plots are printed, one below the other, on one page and the last two plots are printed, one below the other, on the next page.

FOURPLOTS C, K, K, C, K, K, C, K, K, C, K, K, C, K, K, C, K, K

This form of FOURPLOTS is the same as the one above, except the upper and lower limits of the vertical and horizontal axis can be specified by a pair of constants **K, K**. Any of the eight pairs of limits, **K K**, may be omitted or inserted. The instructions:

```
OMNITAB 80 EXAMPLE OF FOURPLOTS.
READ DATA INTO COLUMNS 1, 2, 3, AND 4
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
LABEL REL. INCOME,2
FOURPLOTS 1 VS 2, 1 (30.,40.) VS 3, 1 VS 4 (140.,190.) 2 (9.,11.5) VS 3 (10.,35.)
```

produce the plots shown on page 85.

TWOPLOTS of column **C** versus column **C** and column **C** versus column **C**

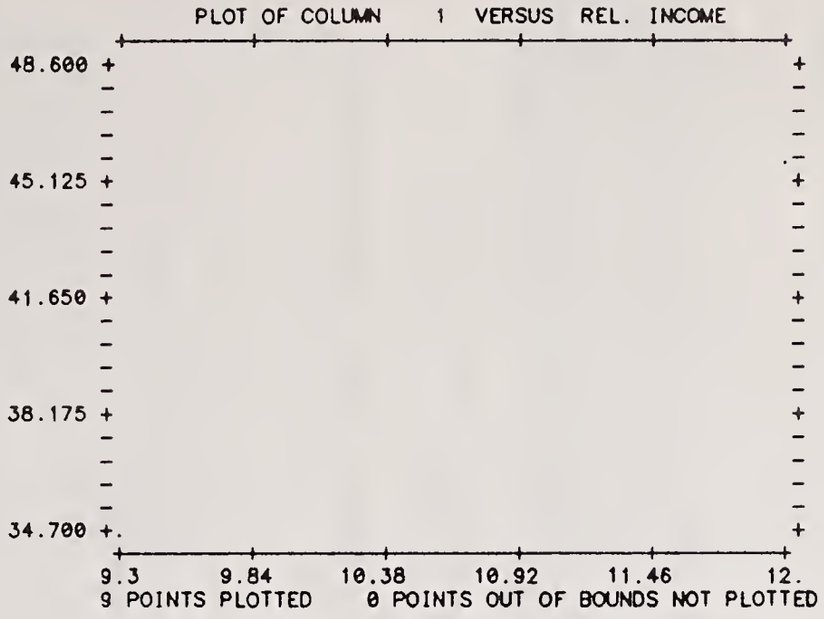
A TWOPLOTS instruction prints two plots in the upper half of a page, instead of on two separate pages.

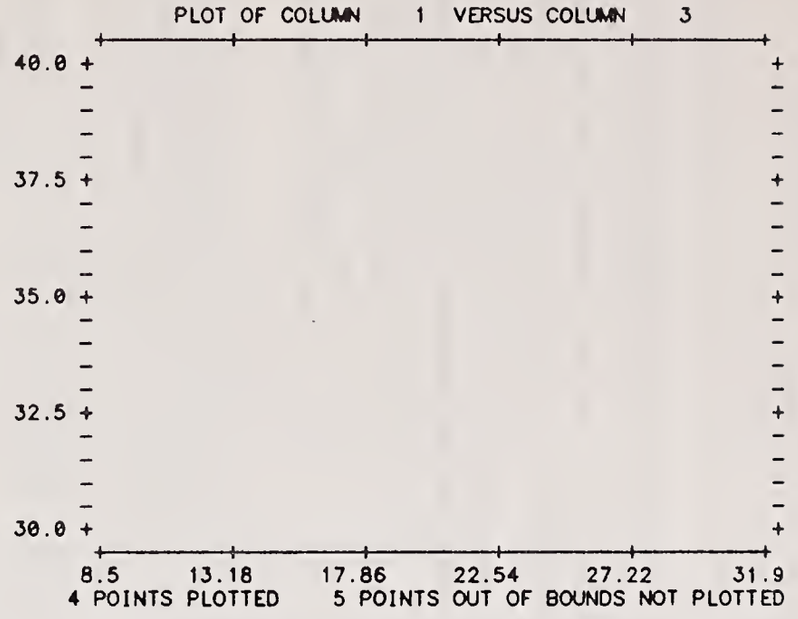
If the value of width, as determined by a WIDTH, INTERACTIVE or TERMINAL instruction, is less than 120, the two plots are printed one below the other on the same page.

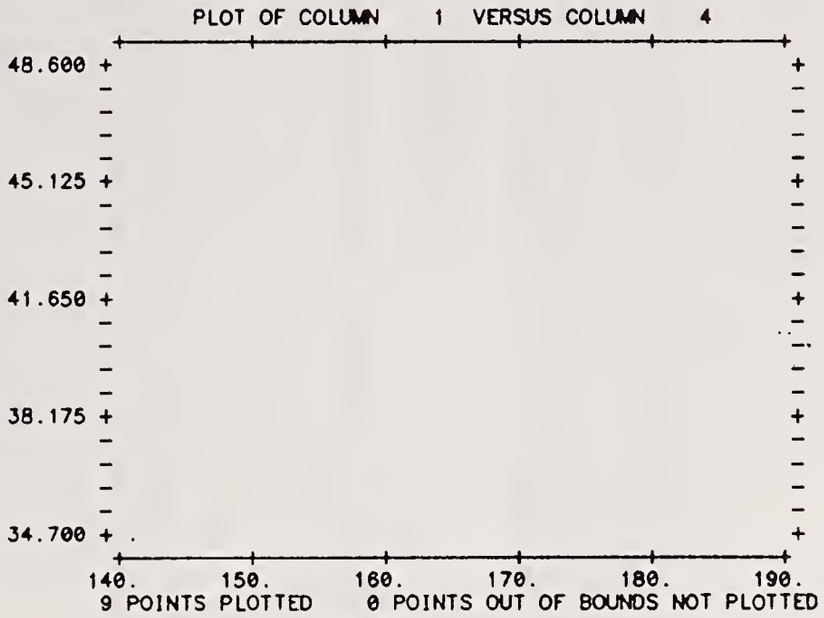
TWOPLOTS col **C**, from **K** to **K** vs **C**, from **K** to **K**, **C**, from **K** to **K** vs **C**, from **K** to **K**

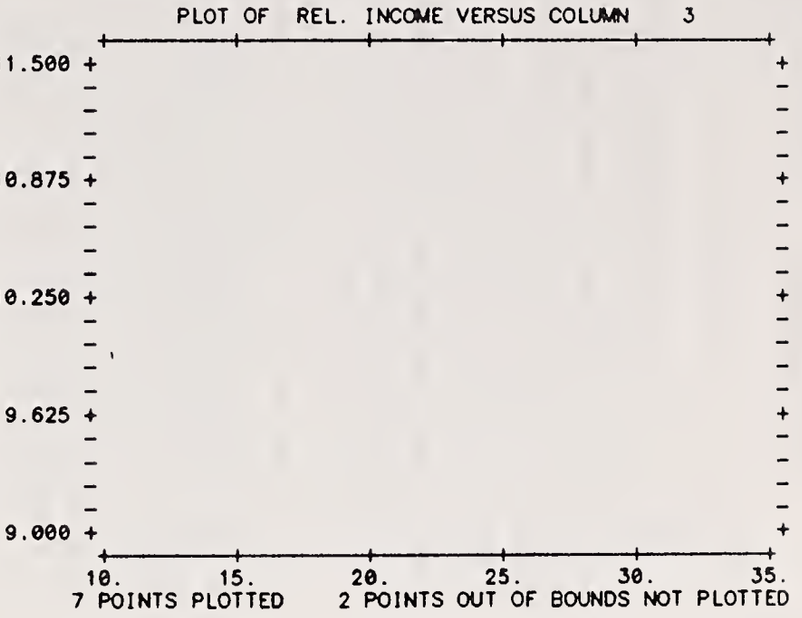
This form of TWOPLOTS is the same as the one above, except the upper and lower limits of the vertical and horizontal scales can be specified by a pair of constants. Any of the four pairs of constants may be omitted or included.

OMNITAB 80 EXAMPLE OF FOURPLOTS.









3.7 Use of Calcomp Plotter

CALCOMP AXIS, CALCOMP FAST, CALCOMP PAPER, CALCOMP PLOT,
CALCOMP SIZE, CALCOMP SLOW, CALCOMP SPEED, CALCOMP TAPE

This section is only applicable if a Calcomp Digital Incremental Plotter and associated software are available.

The eight Calcomp instructions allow one to plot data using the Calcomp Digital Incremental Plotter. The instructions for plotting have two word commands. The first word is always CALCOMP and the second word specifies plotting conditions. The instruction CALCOMP PLOT is the basic instruction and the remaining seven instructions provide additional flexibility by taking advantage of the special features of the Calcomp Plotter.

The CALCOMP PLOT instruction and the seven options generate the data to plot the graph. The different forms allow the user (i) to specify the scale of the vertical and/or horizontal axes, (ii) to plot a number of curves against a single horizontal axis, and (iii) to plot curves with independent horizontal axes (i.e., each curve defined by two arguments, a vertical and a horizontal axis). A new graph is generated with each CALCOMP PLOT instruction.

The three instructions, CALCOMP AXIS, CALCOMP PAPER and CALCOMP SIZE, change the standard height and width of the axes of a graph and the height of the plotting paper.

The normal plotting speed is fast, but the speed at which the plotting is done can be controlled by using any one of the three instructions CALCOMP FAST, CALCOMP SLOW or CALCOMP SPEED. If one wishes to use ink pens, the plotting should be done in slow mode to permit the ink to dry before the paper or plotting arm moves. However, plotting time on the Calcomp Plotter will be longer with slow mode.

The CALCOMP TAPE "L" instruction permits the user to direct the output for plotting to any of the logical tape units A through F. If the instruction is not used, logical tape A is assumed.

Additional information is given in "Programming for Calcomp Digital Incremental Plotters", Bulletin No. 170-E, California Computer Products, Inc., 305 N. Muller St., Anaheim, CA 92803, November 1966.

CALCOMP AXIS K height of y-axis, **K** width of x-axis

The height of the y-axis and width of the x-axis are defined in inches by the two arguments. If the height and/or width is given in fraction of inches, the values will be rounded to the nearest inch. The standard size of the graphs is 9 inches in height by 6 inches in width for a 12 inch paper height and 27 by 18 for 30 inch paper height. Labels and headings add another 1 1/2 inches to the height and 1/2 inch to the width. TITLEX and/or TITLEY add 1/2 inch more in either direction. The maximum height of the y-axis must be less than or equal to the height of the paper minus 2 inches. Otherwise the following informative diagnostic is given:

CALCOMP PAPER WIDTH IS NOT 12 OR 30 INCHES,
OR Y HEIGHT IS TOO LARGE.

and the axes are redefined to the standard size (9x6 or 27x18 for 12 or 30 inch paper, respectively).

CALCOMP FAST mode

Plotting mode is set to fast speed. This mode is the default unless CALCOMP SLOW or CALCOMP SPEED 0 has been used. For ball point pen, fast mode is sufficient.

CALCOMP PAPER n height of paper

Chart paper roll of 120 feet in length with a plotting height (vertical axis) of 12 or 30 inches is available on the Calcomp Plotter. Twelve inch height is assumed unless otherwise specified. This instruction should only

be used once and before any CALCOMP PLOT instruction. If **n** is not 12 or 30, 12 inch height paper is assumed and the following informative diagnostic is given:

CALCOMP PAPER WIDTH IS NOT 12 OR 30 INCHES,
OR Y HEIGHT IS TOO LARGE.

CALCOMP PLOT n curves, **E** options, columns **C, C ,..., C** against **C**

Generates a graph of **n** curves with the same horizontal axis, defined by the last column. Both the horizontal and vertical scales are determined by OMNITAB. The maximum number of curves per graph is 97. The first argument, an integer number, indicates the number of curves to be plotted for that graph. The second is either a column number or a constant with a decimal point. This argument describes the options to be used for plotting the curves. If the argument is a constant, then the option applies to all the curves on that graph. However, if the argument is a column number, then the option in row 1 of that column applies to the first curve, option in row 2 of that column to the second curve, and the option in row n to the nth curve.

The general form of the option is

X.YYZZ

where X is any value 1 through 9, YY takes on values 00 through 85, and ZZ values are 00 through 99.

The value of X indicates whether symbols are to be used and how the points are to be joined (if at all) as indicated in the following table:

X	*Use Symbols*	*Join Points*
1	yes	no
2	no	yes
3	yes	yes
4	no	dash lines
5	yes	dash lines
6	no	smooth curve
7	yes	smooth curve
8	no	closed smooth curve
9	yes	closed smooth curve

If X is 2 or 3, straight lines are drawn from point to point. For X=6,7,8 or 9, a curve is fitted through the points (using a modified spline fitting technique). Furthermore, the first and last points are joined for X=8 or 9.

YY defines the particular symbol of the 85 available to be used for labeling points on the curve. If X is odd and YY is 00, then the first curve that requires a label will be labeled with the first symbol in the table below, the nth curve with the nth symbol. The first 15 symbols (01–15) are centered symbols, the rest (16–85) are lower left centered symbols. The values of YY and corresponding symbols are:

Symbol	*YY*	*Symbol*	*YY*	*Symbol*	*YY*
A	16	0	42	→	52
B	17	1	43	↑	53
C	18	2	44	Σ	54
D	19	3	45	⩾	55
E	20	4	46	⩽	56
F	21	5	47	-	57
G	22	6	48	$\overline{\overline{\Lambda}}$	58
H	23	7	49	↑	59
I	24	8	50	↓	60
J	25	9	51	[	61
K	26			]	62
L	27			#	63
M	28			Δ	64
N	29	*Centered*		)	65
O	30				66
P	31		01	$\bar{+}$	67
Q	32		02	<	68
R	33		03	>	69
S	34	+	04	@	70
T	35	X	05	$	71
U	36		06	*	72
V	37	↑	07	(	73
W	38	X	08	%	74
X	39	Z	09	:	75
Y	40	Y	10	?	76
Z	41		11	!	77
		*	12	,	78
			13	\	79
		|	14	'	80
			15	;	81
				/	82
				.	83
				⌑	84
				≠	85

Every ZZth point on the curve will be labeled with the symbol requested. If ZZ is not defined (i.e., ZZ=00) and labeling is requested, every point on the curve is labeled. Thus to label every 5th point on the curve, 05 should be the value of ZZ. If column 6 has the values 1.1200, 2.0000, 3.0402 and 4.0000, then the instruction

CALCOMP PLOT 4 curves, option in **6**, curves in cols **2**, **3**, **4** and **5** vs **1**

would plot 4 curves on a graph. The option 1.1200 indicates that the first curve is to be plotted with no points joined (1), the symbol to be used is * (12) and every point is to be labeled since the last two digits are zero. The second curve is drawn with points joined and no labeling. The points of the third curve are to be joined and every second point to be labeled with the symbol + (04). The last curve is drawn using dash lines.

Graphs are enclosed on four sides and ticked at one inch intervals. Values are printed at the tick marks along the left vertical and bottom horizontal boundaries. The numbers are in readable format. Three significant digits are used if the absolute value of the numbers is less than or equal to 100. If the absolute values exceed 100, the number of significant digits varies from 3 to 6.

If the CALCOMP PLOT instruction is successfully executed, the following information will be printed immediately below the instruction in the LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS:

.....GRAPH m WAS PLOTTED.....

where m is an integer indicating the graph number. Because the plotting is done after the OMNITAB program is executed, this number along with the word GRAPH will be printed on the top of the graph to identify the CALCOMP PLOT instruction which generated that output for the Plotter. The user can also determine the number of graphs produced on the magnetic tape if the OMNITAB program terminates through a fatal error.

At the top of every graph drawn by the Calcomp Plotter, the OMNITAB instruction is printed. On the next line, GRAPH (m) is printed, where (m) is the number printed after the CALCOMP PLOT instruction. The instructions TITLEX and TITLEY (sec. C3.4) may be used to label the horizontal and vertical axes. The information provided by TITLEY is printed on the left of the y-axis (vertical), if this instruction is used. The information supplied by TITLEX, if any, appears at the bottom of the x-axis (horizontal).

If a BRIEF instruction has been used before a CALCOMP PLOT instruction, the information on the OMNITAB instruction and the title "GRAPH XX" (where XX is the appropriate number) will not appear at the top of the CALCOMP plot.

Whenever the information in a TITLEX or TITLEY is too long for a graph, the following informative diagnostic is given:

TITLEX OR TITLEY IS TOO LONG FOR GRAPH. TITLE IS OMITTED.

If the option(s) specified in the CALCOMP PLOT instruction is (are) incorrect, the following informative diagnostic is given:

EITHER SYMBOL OR JOINING SPECIFIED IS INCORRECT.
SYMBOL . OR NO JOINING IS USED.

CALCOMP PLOT n, E vertical scales **K** to **K, C, C** ,..., **C** versus **C**

This form is the same as the first except the user defines the range of the vertical axis. The scale need not be increasing, i.e., the first **K** which specifies the value at the bottom of the graph can be greater than the second **K** which specifies the value at the top of the graph. The horizontal axis is determined by the instruction. Any points outside the range are not plotted.

CALCOMP PLOT n, E, C ,..., **C** versus **C** horizontal scales from **K** to **K**

The user selects the horizontal scale while the instruction determines the vertical scale.

CALCOMP PLOT n, E vertical **K, K, C** ,..., **C** versus **C** horizontal **K, K**

The user selects the horizontal and vertical scales.

CALCOMP PLOT n curves, **E** options **C** vs **C, C** vs **C** ,..., **C** vs **C**

This form of the CALCOMP PLOT instruction is the same as the first form, except every curve is defined by a pair of arguments (column numbers). For each curve that is to be plotted, the vertical and horizontal coordinates are given. This form makes it possible to plot curves with unrelated horizontal coordinates on the same graph. The maximum number of curves per graph is 49 if this option is used. The number of points for each curve must be the same, otherwise extraneous points will be plotted for some curves.

CALCOMP PLOT n, E, vertical scale K, K, C vs C ,..., C vs C

The range of the vertical axis is defined by the user. The horizontal axis is determined by the instruction. Any points outside the range are not plotted.

CALCOMP PLOT n, E, C versus C ,..., C vs C horizontal scale K, K

The user selects the horizontal scale while the instruction determines the vertical scale.

CALCOMP PLOT n, E vertical K, K, C vs C ,..., C vs C horizontal K, K

The user selects the horizontal and vertical scales.

CALCOMP SIZE n height of paper

The argument **n** is used to set the height of the paper to either 12 or 30 inches. This form is synonymous with CALCOMP PAPER.

The maximum height of the y-axis must be less than or equal to the height of the paper minus 2 inches. Otherwise the following informative diagnostic is given:

CALCOMP PAPER WIDTH IS NOT 12 OR 30 INCHES,
OR Y HEIGHT IS TOO LARGE.

and the axes are redefined to the standard size (9x6 or 27x18 for 12 or 30 inch paper, respectively).

CALCOMP SIZE K height of vertical axis (y-axis)

The maximum value of **K** is the height of the paper minus 2 inches.

CALCOMP SIZE n height of paper K height of vertical axis

Both the height of paper and vertical axis are defined.

CALCOMP SIZE K height of vertical axis, K length of horizontal axis

The height and length of the graph are given. This form is the same as CALCOMP AXIS.

CALCOMP SIZE n height of paper, K vertical height, K horizontal width

The user defines the height of the paper and vertical axis and length of the horizontal axis.

CALCOMP SLOW mode

The plotting arm of the Calcomp Plotter will operate in slow mode to allow the ink to dry.

CALCOMP SPEED n $ n=0 for slow mode, n=1 for fast mode

The speed of the Calcomp Plotter is defined as either slow if n=0, or fast if n=1. The instruction

CALCOMP SPEED 1

is synonymous with the instruction

CALCOMP FAST

and the instruction

CALCOMP SPEED 0

is synonymous with the instruction

CALCOMP SLOW

The following informative diagnostic is printed if the argument in the instruction is incorrect:

CALCOMP PLOTTING SPEED SPECIFIED NOT CORRECT,
ZIP CODE OR HIGH SPEED ASSUMED.

CALCOMP UNIT "L" unit

The graphs generated by the CALCOMP PLOT instruction are outputed on tape unit "L". The qualifier "L" is any letter A through F.

3.8 Statistical Plots

HISTOGRAM, NHISTOGRAM, STATPLOTS, STEM LEAF, SSTEM LEAF

These instructions are useful for displaying large sets of data in a graphical representation for exploratory analysis of the data. The HISTOGRAM and NHISTOGRAM instructions generate graphs of the frequencies.

A STATPLOTS instruction produces four plots which can be very helpful for detecting any anomalies in the data such as nonrandomness, lack of independence, presence of outliers, etc.

A STEM LEAF instruction produces a stem-and-leaf graphical display of data, an innovation from J. W. Tukey, Exploratory Data Analysis, Addison Wesley, 1977. The display is a concise representation of a set of measurements which is similar to a histogram, but has certain advantages. The stem-and-leaf display preserves more of the fine structure of the data and obviates the problem of choosing an appropriate class width for the histogram. Additional information is given on order statistics, such as the median, which makes it a better technique for studying the frequency distribution of measurement data in order to formulate mathematical models and to evaluate "outlying" values and other anomalies, so that summary measures (average, standard deviation) can be correctly interpreted. Very simply, a stem-and-leaf display replaces the mark (X) of a histogram by a digit, thereby preserving the graphical features of the histogram and at the same time providing more information. The four forms of SSTEM LEAF suppress the automatic display.

HISTOGRAM using mid-points in column **C** and frequencies in column **C**

Automatically prints a histogram using one line for each class and representing measurements by a plus sign. Each plus sign may represent more than one measurement as indicated by the scale at the top. The zero of the number printed in the scale is printed at the 0th, 10th, 20th, etc. marks on the scale.

The instruction automatically prints a title, HISTOGRAM WITH MIDPOINTS IN COLUMN XX, FREQUENCIES IN COLUMN XX, where XX is a column number, and column headings MID-POINTS, RCF (relative cumulative frequencies), CF (cumulative frequencies), RF (relative frequencies) and FREQUENCY. The scale is printed below the column headings.

Using the following instructions:

GENERATE LOWER BOUNDARIES FROM **11.5** IN STEPS OF **2.0** TO **31.5** PUT IN COL **11**
GENERATE UPPER BOUNDARIES FROM **13.5** IN STEPS OF **2.0** TO **33.5** PUT IN COL **12**
SET FREQUENCIES IN COL **13**
40 82 51 47 54 31 25 25 18 2 5
ADD COL **11** TO COL **12** MULT BY **0.5** ADD **0.0** PUT IN COL **10** MID-POINTS
WIDTH 72
HISTOGRAM USING MID-PTS IN COL **10** AND FREQUENCIES IN COL **13**

would produce the following histogram.

```
HISTOGRAM WITH MID-POINTS IN COLUMN   10 , FREQUENCIES IN COLUMN   13

  MID-POINTS     RCF     CF     RF     FREQUENCY
                                       0---------30---------60---------90
  12.500000     .105     40   .105    40  ++++++++++++++
  14.500000     .321    122   .216    82  ++++++++++++++++++++++++++++++
  16.500000     .455    173   .134    51  ++++++++++++++++++
  18.500000     .579    220   .124    47  ++++++++++++++++
  20.500000     .721    274   .142    54  ++++++++++++++++++++
  22.500000     .803    305   .082    31  ++++++++++++
  24.500000     .868    330   .066    25  +++++++++
  26.500000     .934    355   .066    25  +++++++++
  28.500000     .982    373   .047    18  ++++++
  30.500000     .987    375   .005     2  +
  32.500000    1.000    380   .013     5  ++

RCF = RELATIVE CUMULATIVE FREQUENCY
 CF = CUMULATIVE FREQUENCY
 RF = RELATIVE FREQUENCY
```

Each tenth mark on the scale is 0, 30, 60 and 90. Therefore each plus represents three measurements, except the last plus may represent only one or two measurements.

HISTOGRAM for class boundaries in columns **C** and **C** frequencies in column **C**

Optional form of HISTOGRAM specifies the class boundaries of the frequency distribution instead of the mid-points. However, the mid-points are printed rather than the class boundaries.

NHISTOGRAM using mid-points in column **C** and frequencies in column **C**

Same as the HISTOGRAM instruction, except printing does not start on a new page nor are titles and column headings printed.

NHISTOGRAM for class boundaries in columns **C** and **C** frequencies in column **C**

Same as the optional form of HISTOGRAM except printing does not start on a new page.

STATPLOTS of column **C**

Measurements in the specified column are automatically displayed by four statistical plots per page.

Plot 1, in the upper left-hand corner, is a simple plot of the measurements X(i) versus the row number i, or the order the measurements are entered into the worksheet. This plot may be used to detect many different patterns of nonrandomness such as trends, outliers, etc.

Plot 2, in the upper right-hand corner, is an autoregressive plot of X(i) versus X(i-1). The plot is particularly useful for assessing lack of independence in the data. If the measurements are well behaved, the plot should not show any recognizable pattern.

Plot 3, in the lower left-hand corner, is a histogram. The histogram, designed by J. J. Filliben, is somewhat different from a conventional histogram. The interval for the horizontal axis is 0.2 standard deviations so that the horizontal axis goes from -5 to $+5$ standard deviations. Two scales are shown: one for the values of the measurements and one below it for multiples of standard deviations from the mean. Frequency is shown on the vertical axis. Information is printed below the histogram which gives the number of measurements, the value of the 0.2 standard deviation class width and the number of observations in excess of the mean plus or minus 5 standard deviations.

Plot 4, in the lower right-hand corner, is a normal probability plot. Section C3.9 gives a description of the advantages and uses of a probability plot. An assumption of a normal distribution of measurements is not satisfied, if the points do not lie approximately on a straight line. Any marked curvature indicated by the points in the plot is a sign of non-normality. Values of the probability plot correlation coefficient (discussed in sec. C3.9), the scrawl (discussed below in STEM LEAF), the mean and the standard deviation are printed at the bottom of the plot.

The vertical and horizontal scales for all four plots are determined by the minimum and maximum values plotted. The plotting symbol for plots 1, 2 and 4 is a period (.). If two or more points coincide, the plotting symbol becomes an asterisk (*). The plotting symbol for plot 3 (histogram) is an X. Depending upon the frequency indicated on the vertical axis, any X may represent one or more measurements. The maximum number of measurements represented by an X is given in the last line of information at the bottom of the histogram.

A LENGTH instruction has no effect on a STATPLOTS instruction. If the width, as determined by a WIDTH, INTERACTIVE or TERMINAL instruction, is less than 120 characters per line, plots 1 and 2 are printed on one page and plots 3 and 4 are printed on the next page. The information printed below the normal probability plot appears below plot 2 on the first page.

The number of measurements must be at least five and less than or equal to one half the size of the worksheet (usually 6250).

Four plots on one page are far more effective than four plots on separate pages. By having all the plots on one page the eye can survey all plots simultaneously. A plot may display evidence not obvious in the other plots. Sometimes one plot can be used to support evidence found in another plot. Occasionally, the evidence in two plots may seem to conflict. This frequently suggests that further analysis or experimentation is needed.

The set of instructions listed below is used to show the use of STATPLOTS to analyze a set of 200 steel beam deflection measurements. The results are shown on page 95.

```
SET BEAM DEFLECTION DATA IN COLUMN 1
−213 −564 −035 −015 +141 +115 −420 −360 +203 −338
−431 +194 −220 −513 +154 −125 −559 +092 −021 −579
−052 +099 −543 −175 +162 −457 −346 +204 −300 −474
+164 −107 −572 −008 +083 −541 −224 +180 −420 −374
+201 −236 −531 +083 +027 −564 −112 +131 −507 −254
+199 −311 −495 +143 −046 −579 −090 +136 −472 −338
+202 −287 −477 +169 −124 −568 +017 +048 −568 −135
+162 −430 −422 +172 −074 −577 −013 +092 −534 −243
+194 −355 −465 +156 −081 −578 −064 +139 −449 −384
+193 −198 −538 +110 −044 −577 −006 +066 −552 −164
+161 −460 −344 +205 −281 −504 +134 −028 −576 −118
+156 −437 −381 +200 −220 −540 +083 +011 −568 −160
+172 −414 −408 +188 −125 −572 −032 +139 −492 −321
+205 −262 −504 +142 −083 −574 −000 +048 −571 −106
+137 −501 −266 +190 −391 −406 +194 −186 −553 +083
−013 −577 −049 +103 −515 −280 +201 +300 −506 +131
−045 −578 −080 +138 −462 −361 +201 −211 −554 +032
+074 −533 −235 +187 −372 −442 +182 −147 −566 +025
+068 −535 −244 +194 −351 −463 +174 −125 −570 +015
+072 −550 −190 +172 −424 −385 +198 −218 −536 +096
STATPLOTS OF COLUMN 1
```

STEM LEAF of column C

The main instruction uses an automatic rule to produce three, or sometimes two, displays without any storage. Three, rather than one, displays are given partly because it is not possible to automatically determine the best display and partly to allow one to look at the data in different ways.

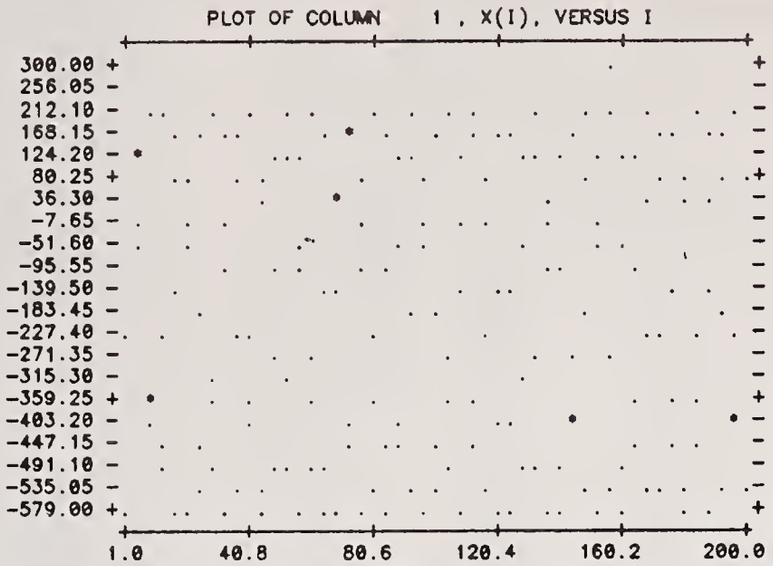
Suppose column 7 contains the 114 measurements:

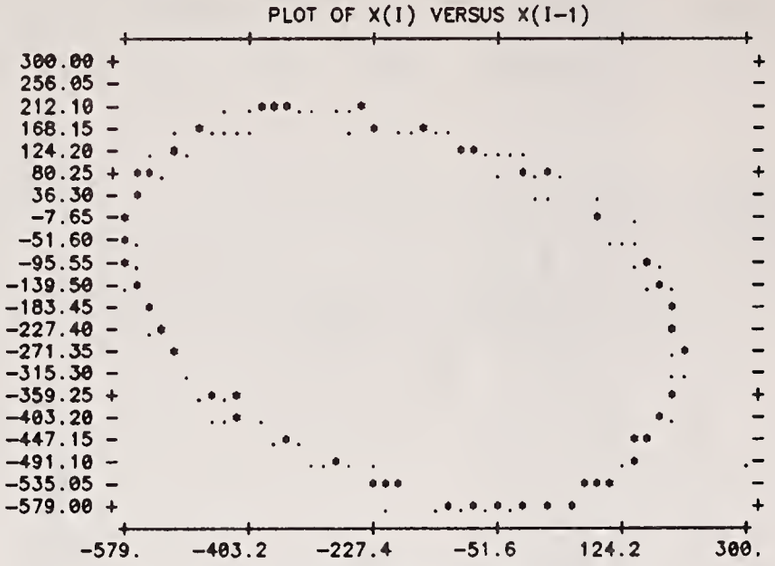
```
116.  82. 127. 100.  94. 117.  90. 121.  93. 120.  83.  95.  88. 123.
 81.  75. 100.  98.  88.  86.  80. 105. 117.  92.  81.  85.  90. 100.
106.  98.  93.  96.  90. 108. 138. 133.  83. 100.  85.  95.  98. 123.
109. 108. 119. 100. 101. 129.  80.  92. 119.  94. 104.  91.  96. 103.
131. 115. 104. 103. 116. 103. 133.  77.  65.  80. 134. 103. 104.  86.
 81.  89. 115. 130. 149. 126.  85.  93. 126.  93.  93.  82.  95.  89.
 83. 124.  95.  84.  82.  75.  97. 100.  83.  83.  77.  78. 125. 113.
104. 121. 105.  80.  92. 103. 116.  68.  86.  76. 102.  94. 141.  86.
126. 119.
```

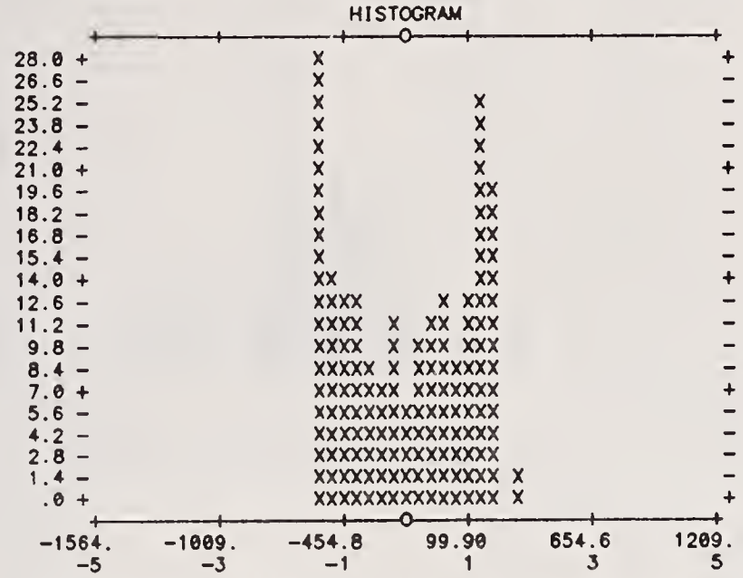
The instructions

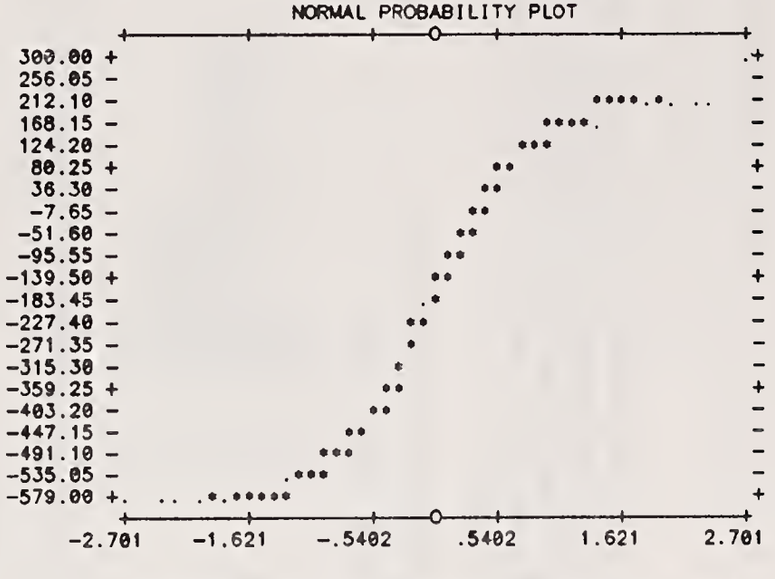
WIDTH 72 characters per line
STEM LEAF of column 7

will cause the three displays shown on pages 96 and 97 to be printed. An explanation of a display follows.









NUMBER OF MEASUREMENTS = 200
CLASS WIDTH = .2 STANDARD DEVIATIONS = 55.466433
0 OBSERVATIONS WERE IN EXCESS OF 5 STANDARD DEVIATIONS
ABOUT THE SAMPLE MEAN AND WERE NOT PRINTED IN THE HISTOGRAM

PROBABILITY PLOT CORRELATION COEFFICIENT = .9540
SCRAWL (N= 200, *= -579.00000, H= -453.00000,
M= -162.00000, H= 94.000000, *= 300.00000)
MEAN = -177.43500, STD. DEV. = 277.33217

A careful examination of the stem-and-leaf display shows that some digits seem to occur more frequently than expected and some less frequently. In the third line of the first display, there are several 0's, 3's and 6's, but only one 4 and no 7's. Further investigation would be needed to determine whether the differences in expected frequencies are real or simply due to chance, but in any case a histogram would not reveal any difference.

Each display starts on a new page which begins with a title

STEM AND LEAF DISPLAY FOR COLUMN XX

where XX is a column number. If LABEL or HEAD command has been used, COLUMN XX is replaced by the appropriate label or heading. The display has four columns for the depth, scrawl, stem and leaves. The additional information printed at the bottom of the display consists of numerical values for the scrawl, units of measurement for the stem and leaves with examples and values of the four parameters used to determine the form of the display.

```
 STEM AND LEAF DISPLAY FOR COLUMN   7

DEPTH SCRAWL STEM     LEAF

   2   *       06  (  58
   8           07  )  556778
  35   H       08  (  00001111222333334555666688 99
       M       09  )  000122233333444555566788 8
  54           10  (  00000012333334444556889
  31   H       11  )  35566677999
  20           12  (  011334566679
   8           13  )  013348
   2   *       14  (  19

SCRAWL (N= 114, *= 65.000000, H= 86.000000, M= 97.500000,
                H= 115.00000, *= 149.00000)

    EACH STEM UNIT IS     10
    EACH LEAF UNIT IS      1
        06  (  5 READS AS   65
        14  (  9 READS AS  149

FOR THE ABOVE STEM AND LEAF DISPLAY, THE FOLLOWING VARIABLES WERE USED
    I =   1, J =   2, K =   2, L =   1

 STEM AND LEAF DISPLAY FOR COLUMN   7

DEPTH SCRAWL STEM     LEAF

   2   *       06  (  58
   2           07  )
   8           07  (  556778
  24           08  )  0000111222333334
  35   H       08  (  55566668899
  50           09  )  000122233333444
       M       09  (  5555667888
  54           10  )  00000012333334444
  37           10  (  556889
  31           11  )  3
  30   H       11  (  5566677999
  20           12  )  011334
  14           12  (  566679
   8           13  )  01334
   3           13  (  8
   2           14  )  1
   1   *       14  (  9

SCRAWL (N= 114, *= 65.000000, H= 86.000000, M= 97.500000,
                H= 115.00000, *= 149.00000)

    EACH STEM UNIT IS     10
    EACH LEAF UNIT IS      1
        06  (  5 READS AS   65
        14  (  9 READS AS  149

FOR THE ABOVE STEM AND LEAF DISPLAY, THE FOLLOWING VARIABLES WERE USED
    I =   1, J =   2, K =   4, L =   1
```

```
 STEM AND LEAF DISPLAY FOR COLUMN   7

DEPTH SCRAWL STEM     LEAF

   1   *       06  (  5
   1           06  )
   2           06  (  8
   2           07  )
   2           07  (
   4           07  )  55
   7           07  (  677
   8           07  )  8
  15           08  (  0000111
  23           08  )  22233333
  27           08  (  4555
  31   H       08  )  6666
  35           08  (  8899
  39           09  )  0001
  47           09  (  22233333
  54           09  )  4445555
  57           09  (  667
       M       09  )  888
  54           10  (  0000001
  47           10  )  233333
  41           10  (  444455
  35           10  )  6
  34           10  (  889
  31           11  )
  31           11  (  3
  30   H       11  )  55
  28           11  (  66677
  23           11  )  999
  20           12  (  011
  17           12  )  33
  15           12  (  45
  13           12  )  6667
   9           12  (  9
   8           13  )  01
   6           13  (  33
   4           13  )  4
   3           13  (
   3           13  )  8
   2           14  (  1
   1           14  )
   1           14  (
   1           14  )
   1   *       14  (  9

SCRAWL (N= 114, *= 65.000000, H= 86.000000, M= 97.500000,
                H= 115.00000, *= 149.00000)

    EACH STEM UNIT IS      10
    EACH LEAF UNIT IS       1
        06  (  5 READS AS   65
        14  (  9 READS AS  149

FOR THE ABOVE STEM AND LEAF DISPLAY, THE FOLLOWING VARIABLES WERE USED
    I =   1, J =   2, K =  10, L =   1
```

Depth column: The depth is basically a cumulative frequency count, but the counting works both from the minimum down to the middle and from the maximum up to the middle of the display. From the minimum to the median (top half of the display), the depth for any stem is the total number of measurements less than or equal to the measurement corresponding to the largest leaf. On the second line of the first display in the above example, the stem is 7 and the largest leaf is 8. This represents the measurement 78. The depth, 8, is the total number of measurements less than or equal to 78. From the maximum to the median (bottom half of display), the depth for any stem is the total number of measurements equal to or greater than the measurement corresponding to the smallest leaf. On the seventh line of the first display in the above example, the stem is 12 and the smallest leaf is 0. Thus, there are 20 measurements equal to or greater than 120. A value for the depth is not given for the stem where the median lies. In some cases, it will take more than one line to print all the leaves for a particular stem. When this occurs, the depth is printed on the last line for that particular stem.

Peter Nemenyi, Sylvia I. Dixon and Nathaniel B. White, Jr., Statistics from Scratch, Holden Day, 1975, give an approximate 95 percent confidence interval for the population median of a set of measurements which can be easily determined from the depth of a stem-and-leaf display. Let a be the integral part of $(n+1)/2 - \sqrt{n}$, where n is the number of measurements. In the example n = 114 and a is the integral part of 57.5 − 10.7 = 46.8 or a = 46. Now find the measurements corresponding to the depth a = 46 to obtain the 95 percent confidence interval. Thus, we can be 95 percent confident the median of our distribution is between 93 and 102.

Scrawl column: The five symbols used are:

*	for the minimum, or smallest measurement,
H	for the lower hinge, or lower quartile,
M	for the median,
H	for the upper hinge, or upper quartile and
*	for the maximum, or largest measurement.

The lower hinge is that value such that 25 percent of the measurements fall below it and 75 percent of the measurements fall above it. Similarly, the upper hinge is that value such that 75 percent of the measurements fall below it and 25 percent of the measurements fall above it. The median, as usual, is the value for which 50 percent of the measurements fall below it and 50 percent of the measurements fall above it.

More precisely, let $m' = ([(n+1)/2]+1)/2$ where [...] is the integral portion of the quotient and n is the number of measurements. Define m to be the largest integer less than or equal to m'. Then, if m=m', the lower hinge will be the $X_{(m)}$ measurement and the upper hinge will be the $X_{(n+1-m)}$ measurement. For m not equal to m', the lower hinge is the mean of the adjacent measurements (i.e. $(X_{(m)}+X_{(m+1)})/2$) and the upper hinge is $(X_{(n-m)}+X_{(n+1-m)})/2$. Let n'=n/2 and m be the largest integer less than or equal to n'. Then if n is odd, the median is the $X_{(2m)}$ measurement, otherwise the median is the mean of the adjacent measurements, $(X_{(2m)}+X_{(2m+1)})/2$.

Each symbol is printed on the line in which the appropriate leaf appears. In the above example, the median is 97.5. For this measurement, the stem is 9 and the leaf 8. Thus, M is printed on the fourth line of the first display opposite the 23d leaf, 8.

Stem column: The stem gives the leading significant digits of a measurement.

In the first display of the above example, each stem is different (k=2).

In the second display, two lines have the same stem (k = 4). The first is used for leaves with values 0, 1, 2, 3 or 4 and the second for leaves with the values 5, 6, 7, 8 or 9. Thus, the six numbers in the 130's, namely, 130, 131, 133, 133, 134 and 138, which are represented on one line in the first display are represented on two lines in the second display.

In the third display, five lines have the same stem (k=10). The first is used for leaves with values 0 or 1, the second for leaves with values 2 or 3, the third for 4 or 5, the fourth for 6 or 7 and the fifth for 8 or 9. Thus, five lines are used to represent the six numbers in the 130's.

If there is more than one stem on a line, the stems are separated by a slash (/) and the corresponding sets of leaves are separated by a slash. Thus

4/5/6

could be used to display all numbers from 40 to 69 and

4/5/6 (//3

would represent 63.

Leaf column: The leaves are separated from their stems by alternating left and right parentheses. The leaf is the next significant digit after the last significant digit in the stem. If a leaf has more than one digit (see below), the leaves are separated by a comma. This is also true for mixed displays when any of the leading digits are actually part of the stem (see below).

Scrawl: Values of the smallest measurement, lower hinge, median, upper hinge and largest measurement are printed after giving the total number of measurements. In the above example, the smallest of the 114 measurements is 65 and the largest is 149. The lower hinge is 86.0, the median is 97.5, and the upper hinge is 115.0.

Scale: Below the scrawl, the units of measurement for the stems and leaves are printed and examples are given for the smallest and largest measurements.

Parameters: The parameters **i, j, k** and **l** described below are determined by the instruction. The value 1 is assigned to **i** and **l** and the best pair of **j** and **k** will be determined by one of the following rules.

Let A=number of display lines from lower hinge to upper hinge divided by the total number of lines in display.

Let B=total number of lines in display divided by the number of measurements.

Then choose **j** and **k** such that $.2 \leqslant A \leqslant .5$, $B \leqslant .15$ and $k \geqslant 2$. Otherwise stop whenever (i) $j + l + 1$ is greater than the number of significant digits in the data; (ii) $B \leqslant .03$; or (iii) total number of display lines is greater than $50 (\log_{10} n)$.

If the above rule is not satisfied, choose the first pair of **j, k** such that **k** is greater than one and the total number of display lines is greater than or equal to 30.

Whenever the two previous rules fail, the best **j, k** pair will be selected which satisfies the conditions **k**=4 and the number of lines in the display is maximum but less than 30.

If NRMAX is less than 5 or all the data values are the same, the following fatal error will be given:

ALL NUMBERS IN THE COLUMN ARE THE SAME.

Whenever the range of data is too large, the fatal error printed is:

VALUE OF SOME MEASUREMENT IS NOT ACCEPTABLE.

If the rules for determining the parameters **i, j, k** and/or **l** are not satisfied, the fatal error message printed is

I, J, K AND/OR L NOT DEFINED CORRECTLY, OR RULE
CANNOT DETERMINE PROPER PARAMETER VALUES FOR THESE DATA.

If the stem **j** is very large and the defined printer line is small, the following informative diagnostic is printed:

PRINTING OF STEM AND LEAF DISPLAY IS NOT POSSIBLE.
INSTRUCTION WAS TREATED AS SSTEM LEAF.

STEM LEAF of column **C** and put scrawl in column **C**

The scrawl will be stored in the second named column. The eight values stored are:

Row	*Description*
1	NRMAX,
2	minimum value of data,
3	the next value after the minimum,
4	the lower hinge,
5	the median,
6	the upper hinge,
7	the value preceding the maximum and
8	maximum value of the data.

STEM LEAF of column **C**, put scrawl in column **C** and depth in column **C**

Both the scrawl and the depth are stored in the specified columns. The first row of the column, where the depth information is stored, contains the total number of stems. The depth of the first stem line is in row 2, depth of the second stem line in row 3 and so forth. If the number of stems is greater than the number of rows, only the number of rows minus one depths will be stored.

STEM LEAF of column C for parameter values i, j, k and l

This instruction is similar to the STEM LEAF, except the user defines the parameters **i, j, k** and **l**.

The parameter **i** determines whether a simple display is given, $i > 0$, or a mixed display is given, $i=0$.

Mixed stem-and-leaf displays, which are necessarily more complicated, can be used to provide a more concise display when the data vary over several orders of magnitude.

The numbers 78, 85, 96, 102 and 113 can be easily shown with a simple stem-and-leaf display as:

```
 7  (  8
 8  )  5
 9  (  6
10  )  2
11  (  3
```

The numbers 78, 85, 96, 198, 276 could not be displayed so easily without using a lot of stems (10, 11,..., 18 and 20, 21,..., 26) which had no leaves. But a mixed stem-and-leaf display would look like:

```
 7  (  8
 8  )  5
 9  (  6
1*  )  98
2*  (  76
```

The asterisk after the 1 (or 2) means that the first digit of the leaf is really part of the stem. Thus, 1* (98 reads as 198, whereas 1 (98 reads as 19 and 18. If two asterisks are used, then the first two digits of the leaf are actually part of the stem, etc.

The parameter **j** is the number of significant digits in the largest stem. It is increased by one, if the decimal point falls within the stem. The value of **j** must be greater than zero. In the first display of the original example, the smallest stem is 6 and the largest is 14 with two digits and hence $j=2$.

The parameter **k** controls the length of each stem. If the display is simple ($i > 0$), then $k=1$, 2, 4 or 10. If $k=1$, two stems are on a line and each stem and the leaves of each stem are separated by a slash. If $k=2$, each stem is on a different line. If $k=4$, a stem will appear twice. If $k=10$, a stem will appear five times. Thus, the numbers 61, 68, 69, 73 and 74 (or 610, 680, 690, 730 and 740) would be displayed

for $k=1$, as

```
6/7  (  189/34
```

for $k=2$, as

```
6  (  189
7  )  34
```

for $k=4$, as

```
6  (  1
6  )  89
7  (  34
```

and for $k=10$, as

```
6  (  1
6  )
6  (
6  )
6  (  89
7  )
7  (  3
7  )  4
```

In other words, the unit of measurement is 2, 1, .5 and .2 times some power of ten depending upon whether $k=1$, 2, 4 or 10.

If the display is mixed, **i**=0, then **k**=1 or 3. If **k**=1, there will be three stems per line as in:

```
4/5/6  (  //189
7/8/9  )  34//
```

If **k**=3, each stem will be on a new line as in:

```
6  (  189
7  )  34
```

The value of the parameter **l** determines the number of digits in any leaf. For the instructions

WIDTH 72 characters per line
STEM LEAF of col 7 for i=**1**, j=**2**, k=**1**, and l=**1**

only one display form is printed. This display contains two stem values per stem line.

```
 STEM AND LEAF DISPLAY FOR COLUMN   7

DEPTH SCRAWL STEM     LEAF

   8   *    06/07  (  58/556778
       H    08/09  )  0000111222333334555666688899/0001222333334445555
       M           )  667888
  54   H    10/11  (  0000001233333444455688 9/35566677999
  20        12/13  )  011334566679/013348
   2   *    14/15  (  19/

SCRAWL (N= 114, *= 65.000000, H= 86.000000, M= 97.500000,
               H= 115.00000, *= 149.00000)

    EACH STEM UNIT IS     10
    EACH LEAF UNIT IS      1
        06/07  (  5 READS AS   65
        14/15  (  9 READS AS  149

FOR THE ABOVE STEM AND LEAF DISPLAY, THE FOLLOWING VARIABLES WERE USED
    I =   1, J =   2, K =   1, L =   1
```

STEM LEAF of column **C** for **i**, **j**, **k** and **l** put scrawl in column **C**

Parameters are defined by the user and the scrawl is stored in specified column.

STEM LEAF of column **C** for **i**, **j**, **k**, **l** put scrawl in **C** and depth in **C**

Similar to above instruction, except depth is also stored in specified column. If the parameters **i**, **j**, **k** and **l** are not defined carefully, the total number of stems may be huge and the following informative diagnostics are printed.

(n) ROWS IN WORKSHEET ARE NOT ENOUGH FOR COMPLETE STORAGE.

and

DISPLAY IS (n) LINES, ONLY FIRST 99 DISPLAYED.

SSTEM LEAF of column **C** put scrawl in column **C**

This instruction is the same as the second form of the STEM LEAF instruction except the automatic display is suppressed.

SSTEM LEAF of column **C** put scrawl in column **C** and depth in column **C**

Similar to the third form of the STEM LEAF instruction except automatic printing is omitted.

SSTEM LEAF column **C** for **i**, **j**, **k** and **l**, put scrawl in column **C**

This instruction is the same as the fifth form of the STEM LEAF instruction except the automatic printing is deleted.

SSTEM LEAF of **C** for **i**, **j**, **k** and **l**, put scrawl in **C** and depth in **C**

Similar to the sixth form of the STEM LEAF instruction except automatic display is depressed.

3.9 Probability Plotting

CAUCHY PLOT,	**DEXPONENTIAL PLOT,**	**EXPONENTIAL PLOT,**	**EXTREME PLOT,**
GAMMA PLOT,	**HALFNORMAL PLOT,**	**LAMBDA PLOT,**	**LOGISTIC PLOT,**
LOGNORMAL PLOT,	**NORMAL PLOT,**	**PARETO PLOT,**	**POISSON PLOT,**
UNIFORM PLOT,	**WEIBULL PLOT**		

A probability plot is a graphical tool for assessing the goodness of fit of some hypothesized statistical distribution (e.g., normal, Poisson, uniform, etc.) to some observed data. It is a plot of the ordered observations, x_i, vertically, versus the theoretical "typical values", M_i, for the ordered observations of the hypothesized distribution, horizontally. The x_i values plotted along the vertical axis are only a function of the observed data. The M_i values plotted along the horizontal axis are only a function of the number of observations, n, and the hypothesized distribution. The discriminatory power of a probability plot stems from the fact that the horizontal axis values, M_i, are strongly affected by the hypothesized distribution. If the user's choice of a distribution is correct, the x_i and M_i will be (approximately) linearly related and the resulting probability plot will be close to a straight line. However, if the user's choice of a distribution is not in accord with the true underlying distribution of the data, then the x_i and M_i will not have a simple linear relation and the resulting probability plot will be non-linear.

A probability plot is a simple, yet powerful technique for assessing the form of the underlying distribution of an observed set of data. An analysis of a probability plot will help answer the following questions.

(1) Are the data normal (or exponential, or ...)?
(2) Are the data longer-tailed or shorter-tailed than normal?
(3) Are certain observations outliers?
(4) Is the experiment under "statistical control?"
(5) Is the assumed regression model correct (an incorrect model will frequently result in gross non-normality of the residuals)?
(6) Are the t and F tests commonly employed in regression and ANOVA valid (they are fully valid only if the underlying distribution of residuals is normal)?
(7) What are the optimal (minimum variance) estimators for location and dispersion (the optimal estimators are dependent on the actual underlying distribution)?

There are several advantages of using a probability plot as opposed to other techniques for checking distributional assumptions.

(1) It is easy to use and interpret.
(2) It is a graphical technique and thus it makes use of and presents information about every element in the data set (outliers, for example, stand out). In this respect, probability plots are superior to methods which reduce all of the data to a single statistic (like chi-squared).
(3) It needs no *a priori* estimate of the unknown location and dispersion parameters of the distribution (as does, e.g., chi-squared).
(4) It uses individual data points and therefore avoids class width and number-of-classes problems intrinsic to, e.g., histogram methods.
(5) It provides "free" estimates of location and dispersion parameters via the intercept and slope of the probability plot of the fitted line to the probability plot.
(6) It is generally applicable to a wide gamut of distributions, distributional families, and distributional types (continuous and discrete).
(7) When the hypothesized distribution is inappropriate, the plot provides "feedback" information for a better hypothesized distribution.

The dominant feature to look for in a probability plot is linearity. If the plot approximates a straight line (necessarily from the lower left corner to the upper right corner), then the hypothesized distribution yields a good fit to the observed data. On the other hand, the more non-linear the probability plot, the worse the distributional fit.

As alluded to previously, under probability plot advantages, an important property of a probability plot is the existence of feedback information. If the choice of a distribution results in a non-linear plot, then the type of non-linearity can be used to determine a better-fitting choice of a distribution. To be specific, suppose the choice was a particular symmetric distribution with "moderate" tails (like, say, the normal distribution) and the resulting probability plot was non-linear. If the shape of the plot is vertical, then horizontal, then vertical (VHV), then this suggests that the distribution is not long-tailed enough, and a distribution which is longer-tailed will probably give a better fit. Conversely, if we had a horizontal, then vertical, then horizontal (HVH) type probability plot, then this suggests that a distribution which is shorter-tailed will fit the data better. Finally, if a probability plot has a non-symmetric cross-over (inflection) point (or no cross-over (inflection) point at all), then this suggests that a non-symmetric (skewed) distribution would be an improved choice for fitting the data.

If the hypothesized distribution adequately describes the data, then the probability plot may show considerable variation about the straight line when the number of measurements is small. Such variation will, of course, tend to decrease as the number of measurements increases.

The probability plot correlation coefficient is a measure of the linearity of the probability plot and hence the adequacy of the distributional fit. See Filliben, James J., "The probability plot correlation coefficient test for normality," Technometrics, 17, No. 1, 1975 pp. 111–117, for percentage points of the PPCC for the normal case.

The intercept and slope of the probability plot serve as estimators of the location parameter μ and the scale parameter σ of the distribution, respectively. It is to be noted that in general μ and σ do not always correspond to the population mean and standard deviation of the distribution; however, in the important normal case, such equivalence does exist.

A statistician should be consulted for help in interpreting the probability plot correlation coefficient, intercept or slope.

It is appropriate to apply probability plots not only to "raw" data, but also to residuals from a regression or an analysis of variance. In using OMNITAB probability plots, the user need not order the observations (or residuals) in the column of interest. Such ordering is done automatically and internally. After a probability plot has been formed, the column being operated on remains unchanged. The column is exactly as it was before the probability plot instruction was executed.

Three different sizes of probability plots are possible. The vertical axis always has the same length and the usual number of 51 plotting positions. The size of the horizontal axis is determined by the use or non-use of a WIDTH instruction. (See sec. C1.5.) The three possibilities are:

Case	*(n) of WIDTH (n)*	*Number of Plotting Positions*
1	120	51x101
2	72–119	51x61
3	59–71	51x41

If a WIDTH instruction has not been used, n = 120 and the plot is the normal size, i.e., 51x101. If n of WIDTH n is less than 59, a plot is not given and the following informative diagnostic is printed:

THE INSTRUCTION WAS IGNORED BECAUSE ...
WIDTH = (n) IS TOO SMALL OR TOO LARGE.

The one line title at the top of the plot and the one line of information at the bottom of the plot are somewhat different for each of the three different sized plots. For a Weibull probability plot with parameter 2.0 and 250 measurements in column 45, the three possible headings at the top of the plot are:

WEIBULL PROBABILITY PLOT OF 250 MEASUREMENTS IN COLUMN 45 WITH PARAMETER=2.0000000

WEIBULL PR.. PLOT OF COLUMN 45 N = 250 PARAMETER 2.0000000

WEIBULL PR PLOT OF COLUMN 45 PARAM. 2.0000000

The line of information at the bottom of the plot, in each case, is:

PROB.. PLOT CORR.. COEFF.. = .9952, ESTIMATES * INTERCEPT = -1.95270000, SLOPE 2.2013169

PLOT COR COEF = .9952, EST* INT. = -1.9527000, SLOPE = 2.2013169

PROBABILITY PLOT CORRELATION COEFF. = .9952

CAUCHY PLOT of column **C**

Plots the Cauchy probability plot for the data in the specified column. The following set of instructions is used to produce the Cauchy probability plot of 200 random normal deviates which appears on page 109.

```
SET TEST DATA IN COLUMN 1
-1.276  -1.218  -0.453  -0.350   0.723   0.676  -1.099  -0.314  -0.394  -0.633
-0.318  -0.799  -1.664   1.391   0.382   0.733   0.653   0.219  -0.681   1.129
-1.377  -1.257   0.495  -0.139  -0.854   0.428  -1.322  -0.315  -0.732  -1.348
 2.334  -0.337  -1.955  -0.636  -1.318  -0.433   0.545   0.428  -0.297   0.276
-1.136   0.642   3.436  -1.667   0.847  -1.173  -0.355   0.035   0.359   0.930
 0.414  -0.011   0.666  -1.132  -0.410  -1.077   0.734   1.484  -0.340   0.789
-0.494   0.364  -1.237  -0.044  -0.111  -0.210   0.931   0.616  -0.377  -0.433
 1.048   0.037   0.759   0.609  -2.043  -0.290   0.404  -0.543   0.486   0.869
 0.347   2.816  -0.464  -0.632  -1.614   0.372  -0.074  -0.916   1.314  -0.038
 0.637   0.563  -0.107   0.131  -1.808  -1.126   0.379   0.610  -0.364  -2.626
 2.176   0.393  -0.924   1.911  -1.040  -1.168   0.485   0.076  -0.769   1.607
-1.185  -0.944  -1.604   0.185  -0.258  -0.300  -0.591  -0.545   0.018  -0.485
 0.972   1.710   2.682   2.813  -1.531  -0.490   2.071   1.444  -1.092   0.478
 1.210   0.294  -0.248   0.719   1.103   1.090   0.212  -1.185  -0.338  -1.134
 2.647   0.777   0.450   2.247   1.151   1.676   0.384   1.133   1.393   0.814
 0.398   0.318  -0.928   2.416  -0.936   1.036   0.024  -0.560   0.203  -0.871
 0.846  -0.699  -0.368   0.344  -0.926  -0.797  -1.404  -1.472  -0.118   1.456
 0.654  -0.955   2.907   1.688   0.752  -0.434   0.746   0.149  -0.170  -0.479
 0.522   0.231  -0.619  -0.265   0.419   0.558  -0.549   0.192  -0.334   1.373
-1.288  -0.539  -0.824   0.244  -1.070   0.010   0.482  -0.469  -0.090   1.171
CAUCHY PLOT OF COLUMN 1
```

The tails of a Cauchy distribution are longer than the tails of a normal distribution and hence the probability plot has a horizontal-vertical-horizontal appearance.

DEXPONENTIAL PLOT of column **C**

Plots the double exponential probability plot for the data in the specifed column.

EXPONENTIAL PLOT of column **C**

Plots the exponential probability plot for the data in the specified column.

EXTREME PLOT of column **C**

Plots the extreme value type 1 probability plot for the data in the specifed column.

EXTREME PLOT with parameter **K** of column **C**

Plots the extreme value type 2 probability plot with the value of the parameter **K** for the data in the specified column. The value of the parameter must be positive. If the parameter value is zero or negative, the following informative diagnostic is printed:

FUNCTION NOT DEFINED FOR SPECIFIED PARAMETER VALUE.

GAMMA PLOT with parameter **K** of column **C**

Plots the gamma probability plot with the value of the parameter **K** for the data in the specified column. The parameter value must be positive.

HALFNORMAL PLOT of column **C**

Plots the half-normal probability plot for the data in the specified column.

LAMBDA PLOT with parameter **K** of column **C**

Plots the lamda probability plot with the parameter value **K** for the data in the specified column. The parameter value may be negative, zero or positive.

LOGISTIC PLOT of column **C**

Plots the logistic probability plot for the data in the specified column.

LOGNORMAL PLOT of column **C**

Plots the log-normal probability plot for the data in the specified column.

NORMAL PLOT of column **C**

Plots the normal probability plot for the data in the specified column. The normal probability plot on page 107 was produced with the instruction

NORMAL PLOT of random normal deviates in column **1**

and data used for the Cauchy plot. Since the data are random normal deviates, the normal probability plot is approximately a straight line.

PARETO PLOT with parameter **K** of column **C**

Plots the Pareto probability plot with the parameter value **K** for the data in the specified column. The value of the parameter must be positive.

POISSON PLOT with parameter **K** of column **C**

Plots the Poisson probability plot with the parameter value **K** for the data in the specifed column. The value of the parameter must be positive.

UNIFORM PLOT of column **C**

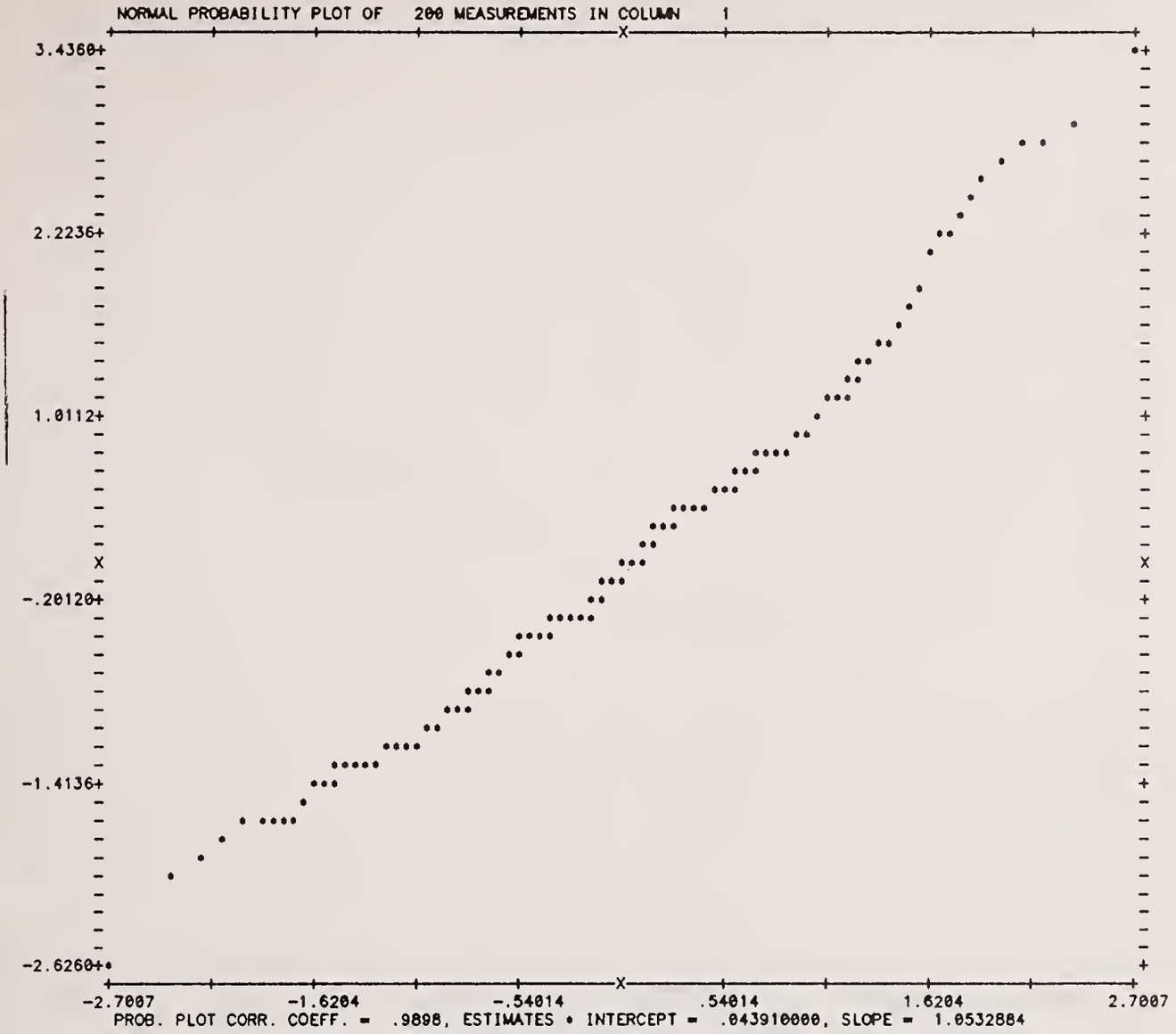
Plots the uniform probability plot for the data in the specified column. The instruction

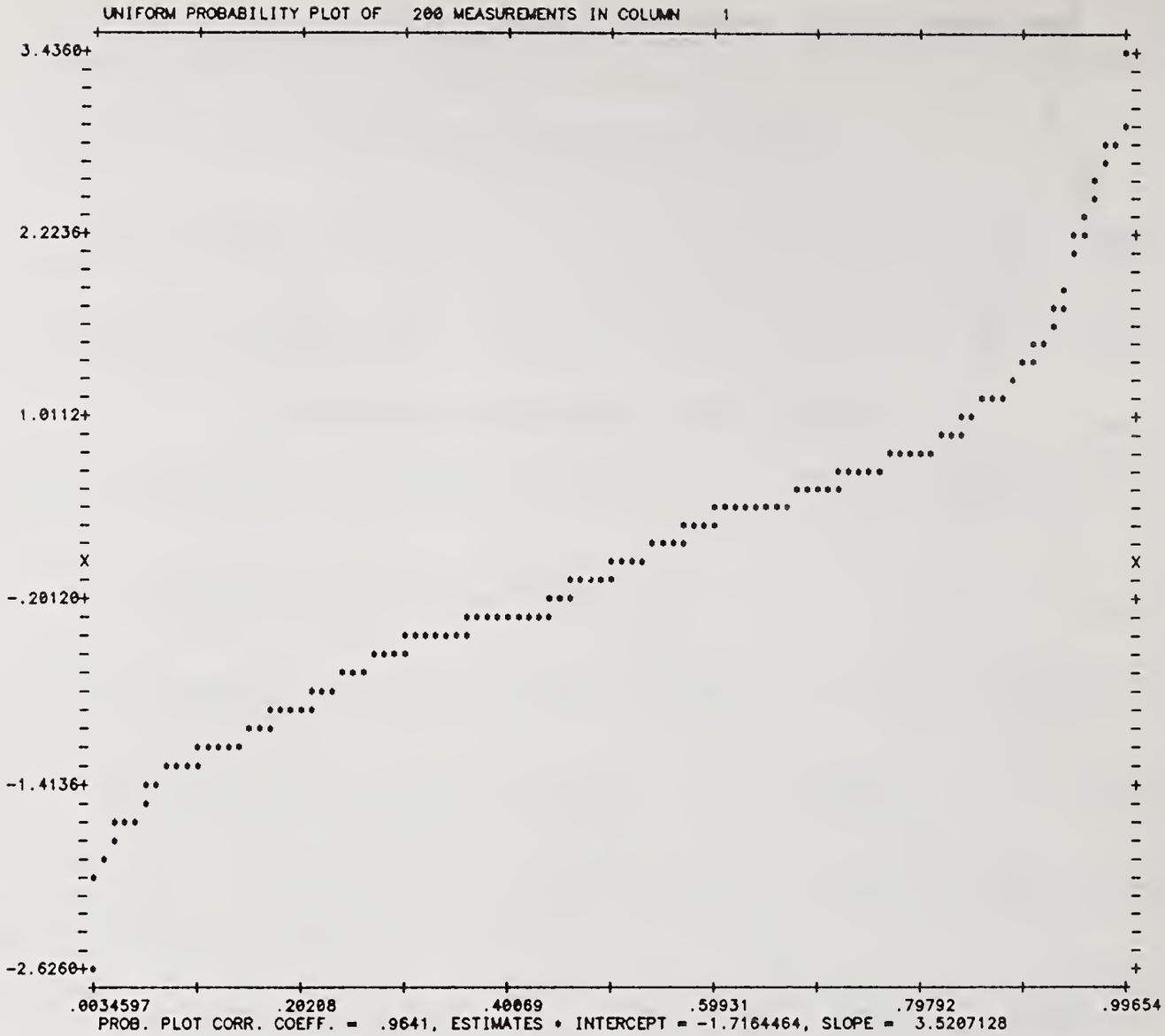
UNIFORM PLOT OF COLUMN 1

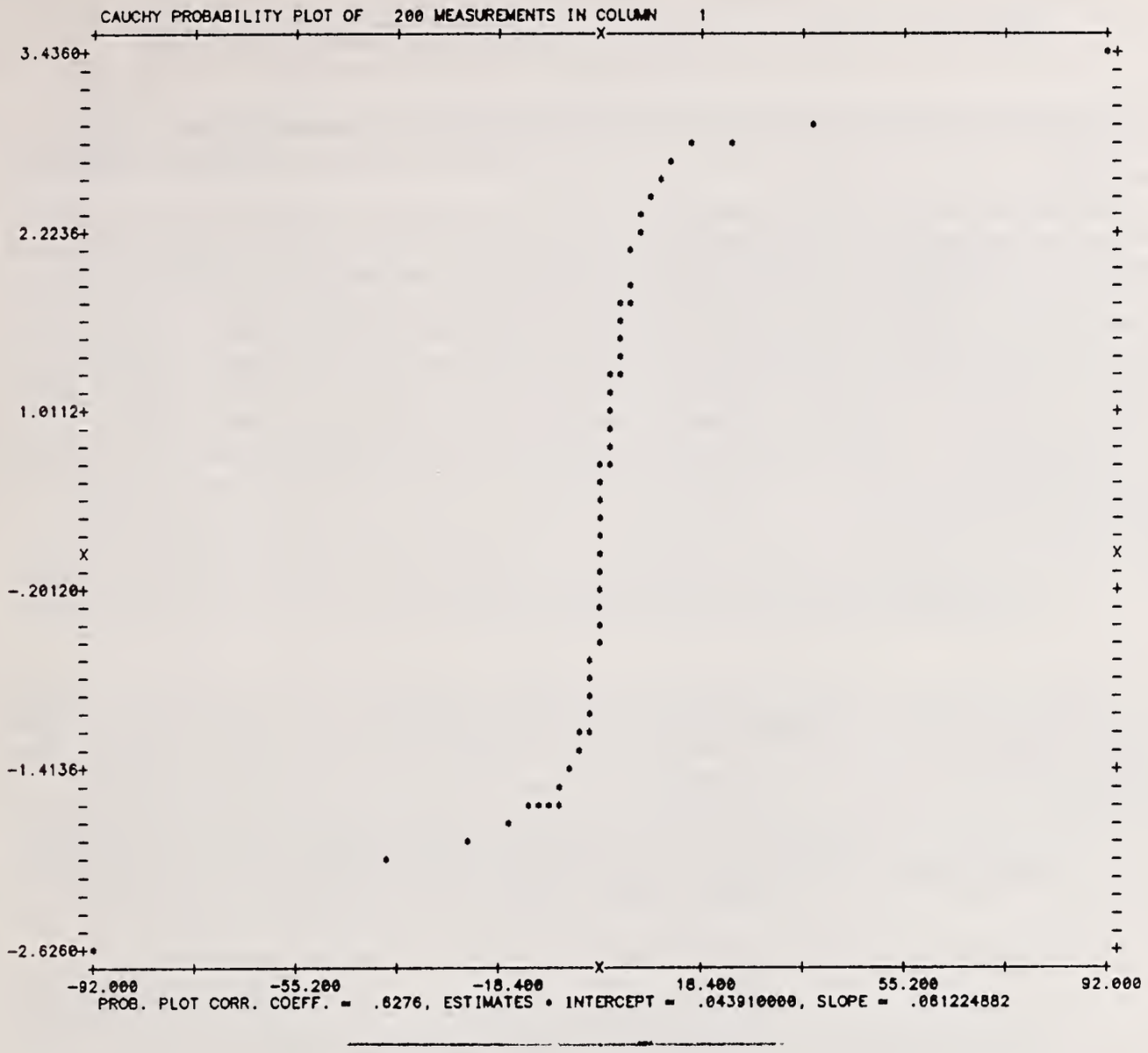
using the data described under the CAUCHY PLOT instruction gives the plot shown on page 108. The uniform distribution has shorter tails, and hence the probability plot does not resemble a straight line and has a vertical-horizontal-vertical appearance.

WEIBULL PLOT with parameter **K** of column **C**

Plots the Weibull probability plot for the parameter value **K** for the data in the specified column. The value of the parameter must be positive.


NORMAL PROBABILITY PLOT OF 200 MEASUREMENTS IN COLUMN 1
3.4360
2.2236
1.0112
-.20120
-1.4136
-2.6260
-2.7007
-1.6204
-.54014
.54014
1.6204
2.7007
PROB. PLOT CORR. COEFF. = .9898, ESTIMATES : INTERCEPT = .043910000, SLOPE = 1.0532884


UNIFORM PROBABILITY PLOT OF 200 MEASUREMENTS IN COLUMN 1
3.4360+
2.2236+
1.0112+
X
-.20120+
-1.4136+
-2.6260+
.0034597
.20208
.40069
.59931
.79792
.99654
PROB. PLOT CORR. COEFF. = .9641, ESTIMATES : INTERCEPT = -1.7164464, SLOPE = 3.5207128



3.10 Use of TEKTRONIX Plotter

TEKTRONIX AXIS, TEKTRONIX PLOT, TEKTRONIX TERMINAL

The instructions in this section generate plots to be displayed on a TEKTRONIX terminal. The TEKTRONIX AXIS and TEKTRONIX PLOT instructions are exactly the same as the CALCOMP AXIS and CALCOMP PLOT instructions described in section C3.7.

TEKTRONIX AXIS K height of y-axis, **K** width of x-axis

The default values of the height and width are 9 inches. The two arguments allow for specifying different values for the height of the y-axis and the width of the x-axis in inches. If the height and/or width are given in fraction of inches, the values will be rounded to the nearest inch. If either argument is greater than 9, that argument is set to 9 and an informative diagnostic is printed. The graph on the hard copy is reduced to 5.25 by 5.25 inches. If the TEKTRONIX screen is smaller than 19 inches, the graph on the screen is automatically scaled to the size of the screen.

TEKTRONIX PLOT n curves, E options, columns C, C ,..., C against C

Generates a graph of **n** curves. The horizontal and vertical scales are automatically determined.

The first argument, an integer number between 1 and 97, indicates the number of curves to be plotted for that graph.

The second argument is either a column number or a constant with a decimal point. This argument describes the options to be used for plotting the curves. If the argument is a constant, the option applies to all curves on the graph. However, if the argument is a column number, the option in row 1 of that column applies to the first curve, the option in row 2 of that column to the second curve, and the option in row n to the nth curve. The general form of the option is X.YYZZ, where X indicates if symbols are to be used and how the points are to be joined (if at all), YY defines the particular symbol to be used, and ZZ indicates that every ZZth point will be labeled with a symbol. Refer to section C3.7 under the description of CALCOMP PLOT for complete details of the values of X, the symbols available and corresponding values for YY. Note that, if the second argument is 1.0, curves will be plotted with symbols at each point and points will not be joined. Unique symbols will be used for each curve. If the second argument is 2.0, the points will be joined by a straight line and no symbols will be used.

TEKTRONIX PLOT n, E, vertical scales K to K, C, C ,..., C vs C

This form is the same as the first except the user defines the range of the vertical scale. The scale need not be increasing, i.e., the first **K** which specifies the value at the bottom of the graph can be greater than the second **K** which specifies the value at the top of the graph. The horizontal scale is automatically determined. Points which fall outside of the range are not plotted.

TEKTRONIX PLOT n, E, C ,..., C vs C horizontal scales from K to K

The user selects the horizontal scales while the vertical scale is automatically determined.

TEKTRONIX PLOT n, E, vertical K, K, C ,..., C vs C horizontal K, K

The user selects the horizontal and vertical scales.

TEKTRONIX PLOT n curves, E options, C vs C, C vs C ,..., C vs C

This form of the TEKTRONIX PLOT instruction is the same as the first form, except every curve is defined by a pair of arguments (column numbers). This form makes it possible to plot curves with unrelated abscissas (x-axes) on the graph. The maximum number of curves per graph is 49. The number of points for each curve must be the same, otherwise extraneous points will be plotted for some curves.

TEKTRONIX PLOT n, E, vertical scales K, K, C vs C ,..., C vs C

The range of the vertical scale is defined by the user. The horizontal scale is automatically determined. Points which fall outside of the range are not plotted.

TEKTRONIX PLOT n, E, C vs C ,..., C vs C horizontal scale K, K

The user selects the horizontal scale while the vertical scale is automatically determined. Points outside the range are not plotted.

TEKTRONIX PLOT n, E vertical from K, K, C vs C ,..., C vs C horizontal from K, K

The user selects the horizontal and vertical scales. Points outside the range are not plotted.

TEKTRONIX TERMINAL n terminal number, n baud rate

The first argument of this instruction indicates the TEKTRONIX terminal number (i.e., 4010, 4012, 4014, etc.). The second argument specifies the number of characters per second transmitted (i.e., 300, 1200, 9600, etc.). If the terminal number is either 4010 or 4012, or the baud rate is other than 9600, the instruction TEKTRONIX TERMINAL must be used before any TEKTRONIX PLOT instruction.

4. ARITHMETIC OPERATIONS

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

None of the instructions in this section change the value of NRMAX. Be sure that data has been entered into the worksheet by using one of the instructions described in section C2.1.

4.1 Simple Arithmetic

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 fault is given:

DIVISION BY ZERO.

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 if not a column number 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 *result is set equal to one*, and the following arithmetic fault is given:

SQRT, LOG OR RAISE OF NEGATIVE NUMBER.

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.

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 large (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. Otherwise overflow will occur. If E2 is not an integer and E1 is positive, overflow will also occur when E2*log (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.

4.2 More Simple Arithmetic

ABSOLUTE, CHANGE, RECIPROCAL, SQRT, SQUARE

The instructions in this section do not have three arguments. ABSOLUTE, RECIPROCAL, 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 instruction has the effect of multiplying each designated column by -1.0. All arguments in this instruction are column numbers. The results are put back into the designated column and not into new columns. 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

RECIPROCAL OF E put in column **C**

Divides 1.0 by a constant or a column as indicated by the first argument and puts results into the column specified by the second argument.

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 fault is given:

SQRT, LOG OR RAISE OF NEGATIVE NUMBER.

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 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 either of the instructions

MULTIPLY column **1** by column **1**, put in column **2**

or

RAISE column **1** by **2.0**, put in column **2**

4.3 Logarithms and 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(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 result of the instruction is larger than the largest real number contained in the computer, an overflow occurs, the result is set equal to zero and the following arithmetic fault is given:

EXPONENT TOO SMALL OR TOO LARGE.

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 fault given:

SQRT, LOG OR RAISE OF NEGATIVE NUMBER.

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 fault given:

SQRT, LOG OR RAISE OF NEGATIVE NUMBER.

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.

4.4 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:

VALUE OUT OF RANGE AND INVERSE FUNCTION CANNOT BE EVALUATED.
X TOO LARGE FOR SIN(X) OR COS(X).
TRIGONOMETRIC FUNCTION NOT DEFINED.

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. The instructions are described in alphabetical order.

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**; $y=\log[(x+1)/(x-1)]/2$.

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**; $y=\log[(x+1)/(x-1)]/2$.

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**; $y=(e^x+e^{-x})/2$.

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**; $y=(e^x-e^{-x})/2$.

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**; $y=(e^x-e^{-x})/(e^x+e^{-x})$.

4.5 Triple Operations

All the commands in sections C4.1, C4.2 (except CHANGE), C4.3 and C4.4 have an additional form with two additional arguments for triple operations. (Hence, commands in section C4.1 will have 5 arguments, whereas those in sections C4.2, C4.3 and C4.4 will have 4 arguments.) Also, there are two commands in section C4.7, FRACTIONAL and INTEGER, which have this additional form. The three operations are: (1) the operation specified by command (e.g., DIVIDE), (2) multiplication, and (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

If the number of arguments in an instruction is incorrect (should be 3 or 5 for those in section C4.1 and 2 or 4 for those in sections C4.2, C4.3 and C4.4), the following fatal error occurs:

(n) IS AN INCORRECT 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 (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 **E**

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**
RECIPROCAL	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**

4.6 Evaluation of FORTRAN Arithmetic Expressions

EVALUATE

This instruction evaluates FORTRAN like arithmetic expressions. It can be conveniently used to replace several instructions for evaluating a complex expression by a single instruction.

EVALUATE column **C** = a FORTRAN arithmetic expression

An EVALUATE instruction is governed by both FORTRAN rules and OMNITAB rules. The result, as in FORTRAN, appears on the left hand side of the equation, rather than on the right. Comments are not allowed except after a currency symbol sign ($). Labels are allowed, but care should be exercised to avoid the use of a label which is the same as the name of a basic external or intrinsic function such as SIN. The number

0.0176 can be written 1.76-2 after a SET or READ instruction, but in an EVALUATE instruction, it must be written 0.0176 or 1.76E-2. The OMNITAB rule of using integers for column numbers and numbers with a decimal point to indicate a constant still holds. Hence, the number 714 must be written 714. or 714.0. The equal sign must appear after the first argument. The FORTRAN rule which requires that an expression begin in character position 7 is not in force. The EVALUATE instruction can not be stored.

The instruction

EVALUATE 8 = 3.4 * (11−6.0) + ALOG(5)

is equivalent to the instructions

SUBTRACT 6.0 from column **11** and put in column **8**
MULTIPLY column **8** by **3.4** and put in column **8**
LOG of column **5,** mult by **1.0,** add column **8,** put in column **8**

The FORTRAN operators which can be used are:

\+
−
/
*
**

The FORTRAN intrinsic functions which are available are:

ABS	AINT

The FORTRAN basic external functions which are available are:

ALOG	ALOG10	ATAN	COS
EXP	SIN	SQRT	TANH

The OMNITAB command names LOG and LOGTEN may be used in place of the FORTRAN basic external functions ALOG and ALOG10, respectively.

If any other operator, function or expression is used, the following fatal error message is printed:

ILLEGAL OPERATOR, EXPRESSION OR FUNCTION.

4.7 Data Summarization

ACCURACY,	**DAYS,**	**EXPAND,**	**FRACTIONAL,**
INTEGER,	**PARPRODUCT,**	**PARSUM,**	**PERCENTAGES,**
PRODUCT,	**PROPORTIONS,**	**RMS,**	**ROUND,**
ROW SUM,	**SUM**		

Frequently a summarization of the data is required in order to ascertain particular information about the data. The instructions in this section enable the user to obtain the sum both by column and row, to find partial products and product of a column, to round the data or extract the integer part or fractional part, to check agreement between two sets of data and to determine certain properties of the data.

ACCURACY of **E** compared with **E** put in column **C**

This instruction is used to check how closely two (sets of) numbers agree. In ACCURACY X, Y, Z

$$
\begin{array}{lll}
Z = -\text{LOGTEN} & (X-Y)/Y, & \text{if } X \neq Y \text{ and } Y \neq 0 \\
Z = -\text{LOGTEN} & X-Y, & \text{if } X \neq Y \text{ and } Y = 0 \\
Z = 8, & & \text{if } X = Y
\end{array}
$$

But Z is never less than -8.0 or greater than $+8.0$.

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, 4.6144510, .096367392, 8.0000000, 7.0731967, −.30103000, −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, 7.0731968, −.30103000, .0070906729, .30103000 and −.091514942. 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 −.30103000.

DAYS for month **E** day **E** and year **E** put in column **C**

The instruction DAYS converts a date like 8/5/76 into a unique number. For individual dates, this number is of minor importance, unless one is coding data for a statistical analysis. However, it can be very useful for finding the number of days between any two dates. The instruction automatically adjusts for the differing numbers of days in the months and years. The definition of a leap year is somewhat complicated and not widely known, but the DAYS instruction uses it appropriately. No month, day or year may be negative or zero.

For the years in the 20th century (1900 to 1999), it is not necessary to include the first two digits of the year. Thus, 76 may be used instead of 1976.

To compute the number of days between January 17, 1976 and May 8, 1976, inclusive, we could use the instructions

READ date into columns **1**, **2** and **3**
1/17/76
5/ 8/76
DAYS for columns **1,2,** and **3** put in col **4**
PROMOTE 1 row column **4** into col **5**
SUBTRACT col **4** from col **5** and put in col **6**
ADD 1.0 to column **6** and put in column **6**
ABRIDGE row **1** of col **6**

to get the answer 113.

For reference purposes, the number computed for 1/1/100 is 1. Because of the convention described above, of being able to use only the last two digits for years between 1900 and 1999, inclusive, years between 1 AD and 99 AD cannot be used. Note, according to American, but not European, tradition, the month comes before the day rather than after it.

EXPAND E to power **p** in increments **i** and put in column **C** and successive columns

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**.

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.

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 increments of **K** put in column **C** and successive columns

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.

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

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 to the right of the decimal point is chopped and the integer 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.

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.

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.

PERCENTAGES of numbers in column **C** put in column **C**

Each number in the first named column is divided by the sum of the numbers, multiplied by 100.0 and put into the second column. If x_i represents values in the column of the first argument and y_i, values in the column of the second argument, then $y=[x_i/(\sum_{i=1}^{NRMAX} x_i)]*100$. If the numbers 8, 11, 14, 5 and 1 are in column 8, the instruction

PERCENTAGES of column **8** put in column **23**

would put the numbers 20.512820, 28.205128, 35.897435, 12.820513 and 2.5641025 in column 23. The sum of the values in the column of the second argument is always 100.

PERCENTAGES of columns **C, C** ,..., **C** put in column **C, C** ,..., **C**

The optional form of the PERCENTAGES instruction makes it possible to compute the percentages of numbers in several columns.

PRODUCT row by row of columns **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 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.

PROPORTIONS of numbers in column **C** put in column **C**

A PROPORTIONS instruction is similar to the PERCENTAGES instruction, described above, except the numbers in the second column are not multiplied by 100. If the numbers 8, 11, 14, 5 and 1 are in column 36, the instruction

PROPORTIONS of column **36** put in column **52**

would put the numbers 0.2051282, 0.28205128, 0.35897436, 0.12820513 and 0.025641026 in column 52. The sum of the numbers in the second column is 1.0.

PROPORTIONS of columns **C, C ,..., C** put in columns **C, C ,..., C**

The optional form of the PROPORTIONS instruction makes it possible to compute the proportions of numbers in several columns.

RMS of column **C** put in column **C**

For the n=NRMAX values x in the 1st named column, the instruction puts the root mean square (RMS)

$$\sqrt{\sum_{i=1}^{n} x_i^2/n}$$

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.

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.

If the (**n**+1)st digit is less than 5, the portion beyond the **n**th digit is dropped and the **n**th digit is unchanged. E.g., 1234.5678 rounded to 3 digits is 1230.

If the (**n**+1)st digit is greater than 5, the **n**th 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.

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.

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 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 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**

4.8 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 C4.1. 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).

CADD real **E**, imaginary **E**, to real **E**, imaginary **E**, put real in **C** imaginary in **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**, imaginary **E**, by real **E**, imaginary **E**, put real in **C**, imaginary 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 fault is given.

CMULTIPLY real **E**, imaginary **E**, by real **E**, imaginary **E**, put real in **C**, imaginary in **C**

Performs the indicated complex multiplication. For $z=z_1z_2$,

$$x=x_1x_2-y_1y_2 \text{ and } y=x_1y_2+x_2y_1.$$

CPOLAR for x=**E**, y=**E** put rho in column **C**, theta in column **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 arithemtic fault is given.

CRECTANGULAR for rho=**E**, theta=**E** put x in column **C**, y in column **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**, imaginary **E**, from real **E**, imaginary **E**, put real in **C**, imaginary 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.

5. DATA MANIPULATION

This section contains six types of instructions for data manipulation which enhance problem solving considerably. Several of the instructions in this section affect the value of NRMAX. Also, in some of the instructions an argument may inadvertently 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 difficult to comprehend. A liberal amount of self-teaching can help immensely, see section B4.1.

5.1 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 exceeding 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 sec. C5.2)

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 sec. C5.2) 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 through row NRMAX 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. See, also, section B1.5.

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 B1.6 and B1.15 for further details.

5.2 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 C10 along with the regular array operation commands. See section C10 for further details.

DEMOTE by **r** rows, column **C** into column **C** ,..., column **C** into column **C**

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. 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

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 PROMOTE **r** and NRMAX is not affected.

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 BELOW THE (n) ROW WORKSHEET.
DATA BELOW ROW (n) 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. The new value of NRMAX is NRMAX plus **r**. The numbers in the first **r** rows of the worksheet remain unchanged.

DUPLICATE t times the array starting in **R, C** of size **rxc** 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 **txr** rows and **c** columns. 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 three MOVE instructions below:

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 **txr** exceeds the number of rows in the worksheet, the following fatal error occurs:

ROW NUMBER(S) OUTSIDE (n) ROW WORKSHEET.

EXCHANGE column **C** with column **C**, column **C** with column **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 **rxc** to the array in **R, C**

The first two pairs of 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 and the array may be moved below NRMAX. NRMAX is not changed. 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 C10 and the description of the synonym AMOVE in section C10.2.

If an attempt is made to move a rectangular array outside the worksheet, the following fatal error will result.

ROW NUMBER(S) OUTSIDE (n) ROW WORKSHEET.

PROMOTE by **r** rows, column **C** into column **C** ,..., column **C** into column **C**

This instruction is essentially the reverse of DEMOTE, which is described above, except NRMAX is not affected. 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.

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 instruction DEMOTE **r** and NRMAX is affected.

If the argument **r** exceeds NRMAX, the following informative diagnostic is given

ATTEMPT TO PROMOTE FROM BELOW NRMAX = (n).
FIRST ARGUMENT IS RESET = (n).

PROMOTE all values in the worksheet by **r** rows

Similar to the above instruction, but the entire worksheet is promoted. Each column is promoted and 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.

5.3 Manipulative Operations

CENSOR, CENSOR EQ, CENSOR GE, CENSOR GT,
CENSOR LE, CENSOR LT, CENSOR NE, CLOSE UP,
FLIP, INSERT, SEPARATE, SHORTEN

These instructions enhance problem solving considerably. The CENSOR instructions are related to the instructions described in section C5.5.

CENSOR column **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. 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 C5.5. 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.

The command CENSOR is synonymous with the command CENSOR LE.

CENSOR EQ column **C** for values equal to **E**, replace by **E**, put in column **C**

CENSOR GE C for values greater than or equal to **E**, replace by **E**, put in **C**

CENSOR GT column **C** for values greater than **E**, replace by **E**, put in **C**

CENSOR LE C for values less than or equal to **E**, replace by **E**, put in **C**

CENSOR LT column **C** for values less than **E**, replace by **E**, put in column **C**

CENSOR NE column **C** for values not equal to **E**, replace by **E**, put in **C**

The above six instructions are modifications of the CENSOR instruction. Each command consists of two words. The first word is always CENSOR and the second word indicates the comparison used to CENSOR the values of the first argument. Each number, X, in the first named column is compared with the value(s), Y=**E** of the second argument. X is replaced by the number designated by the third argument and put in the corresponding row of the column designated by the last argument, only if the comparison is satisfied. Otherwise, X is moved to the designated last argument.

A CENSOR LE instruction is synonymous with a CENSOR instruction. For the numbers

column 11:	2.3,	7.6,	5.2,	8.3,	4.2
column 12:	3.4,	5.8,	5.2,	4.7,	6.3

the instructions

CENSOR EQ	column	**11** for	**7.6,**	replace by	**0.0,**	put in column	**21**
CENSOR GE	column	**11** for column	**12,**	replace by	**−1.0,**	put in column	**22**
CENSOR GT	column	**11** for	**4.5,**	replace by column	**12,**	put in column	**23**
CENSOR LE	column	**11** for column	**12,**	replace by	**−1.0,**	put in column	**24**
CENSOR LT	column	**11** for	**6.0,**	replace by column	**12,**	put in column	**25**
CENSOR NE	column	**11** for column	**12,**	replace by	**−1.0,**	put in column	**26**

would leave columns 11 and 12 unchanged and put the following numbers in columns 21 to 26:

21	*22*	*23*	*24*	*25*	*26*
2.3	2.3	2.3	−1.0	3.4	−1.0
0.0	−1.0	5.8	7.6	7.6	−1.0
5.2	−1.0	5.2	−1.0	5.2	5.2
8.3	−1.0	4.7	8.3	8.3	−1.0
4.2	4.2	4.2	−1.0	6.3	−1.0

CLOSE UP rows with **K** in columns **C, C ,..., C**

Each row in a column is searched for the number **K**. When a match is found the value in that row is set equal to zero, moved to the bottom of the column and all rows in the column below that row are 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 column **C** ,..., column **C** into column **C**

In each pair of columns, the second named column becomes 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 column **C** from **C** at every **i**th row, start at 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 in the last named column. The first two columns remain unchanged. The value of NRMAX is increased by

$$1+(\text{NRMAX}-\mathbf{R}+1)/(\mathbf{i}-1)$$

NRMAX never exceeds the number of rows in the worksheet (normally 201). The beginning value of NRMAX must be at least 2. 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 **2**nd. 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. 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 **2**nd 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 six rows intact. The instruction performs virtually the reverse operation of INSERT, above.

SHORTEN column **C** for column **C** equal to **K**, put shortened columns in **C**, **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 of the second column 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**. 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

In this case 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 WAS NOT FOUND.

NRMAX remains unchanged and the first two columns are put in the columns designated by the last two arguments.

5.4 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 C6.1. 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** from 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** from minimum to maximum, 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

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 instruction

SORT column **23** and carry along column **11**

would yield the results:

Column 23:	2.15,	4.29,	6.47,	7.38	and	8.54
Column 11:	3.00,	2.00,	5.00,	1.00	and	4.00

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.

5.5 Search Operations

MATCH, SEARCH, SELECT

These three instructions perform three different kinds of searching. The instruction MATCH is particularly useful in treating subsets of data. The instruction ONEWAY, described in section C6.6, 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, 5,385.00 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, 5,385.00 and 0.00 in column 31 after the execution of MATCH and would put the desired result 10,597.00 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 C6.6) could be used.

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 specify an unused column or use an ERASE instruction. 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 column **C** equal column **C**, move corresponding numbers in **C** into **C**, **C** to **C** etc.

A search is made in the first named column until the first number equal to the number in row 1 of the second named column is found. If the number is found in the kth row, the number in row k of the 3rd named column is transferred 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 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 column **C** numbers approx=in **C** within absolute tolerance **K** and put in **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** numbers approximately= in **C** within absolute tolerance **K** put in **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** numbers approx=in **C** within absolute tolerance **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.

5.6 Editing Data

CHOOSE, CODE, DELETE, OMIT, RECODE, REPLACE, RETAIN

The instructions described in this section are useful for editing data or for data manipulation in the following problem areas.

1. *Deletion*. Certain data are to be deleted for any one of a number of reasons:
   (a) coded missing observations,
   (b) outliers,
   (c) possible outliers,
   (d) analysis of subsets of data, etc.

2. *Retention*. This is the complement of deletion above and is done for similar reasons. A person's height equal to -10cm should be deleted as an outlier. At the same time, we would want to retain all positive heights, knowing that all negative heights are outliers by definition.

3. *Coding*. There are a number of reasons for coding data. For example, to
   (a) specify subsets for further analysis,
   (b) simplify data,
   (c) aggregate values to be treated alike, e.g., all persons aged 10 to 14,
   (d) and/or identify missing observations, etc.

Missing observations have to be identified. A blank is unsatisfactory. Sometimes zero can be used, but at other times it cannot be used. Missing observations must be coded and subsequently deleted from further analysis. The coded value must be outside the known range of permissible values.

The instructions OMIT and DELETE are useful for deleting data, CHOOSE and RETAIN for retaining data, and CODE, RECODE, and REPLACE for coding data. DELETE is similar to OMIT, except it can operate on several columns simultaneously. Also, RETAIN is similar to CHOOSE, except it can operate on several columns simultaneously.

Because of rounding, truncation or conversion errors, a number in the computer may be slightly different from what one thinks it should be. For example, a number believed to be 8.5 may actually be something like 8.4999999 or 8.5000001. If an instruction is looking for numbers equal to 8.5, it may not succeed when it should. This is not a serious problem. However, when it does (or might) occur, a ROUND instruction can be used to avoid any difficulty.

CHOOSE rows with **K** in column **C** and put in column **C**

All numbers in the first column equal to **K** are stored in the second column. Zeros are put at the bottom of the second column. NRMAX is reset to the number of rows in the first column which have the value **K**.

CHOOSE rows with numbers between **K** and **K** in column **C** and put in column **C**

All numbers in the first column between **K** and **K**, inclusive, are stored in the second column. Zeros are put at the bottom of the second column and NRMAX is reset accordingly.

CHOOSE rows with **K** in **C**, corresponding rows of **C** ,..., **C** put in **C** ,..., **C**

The number of arguments in this form of CHOOSE is 1 + 2n, where n = 1, or 2, or 3, ..., up to 49. There are two sets of n column numbers. If **K** is found in any row of the first named column, that row of every column in the first set is included in the second set of columns. The other rows are omitted. Zeros are put at the bottom of the second set of columns and NRMAX is reset.

CHOOSE from **K** to **K** in **C**, corresponding rows of **C** ,..., **C** put in **C** ,..., **C**

This fourth form of CHOOSE bears the same relation to the second form as the third form does to the first. The instruction has 2 + 2n arguments, where n can be any number from 1 to 48. For the data in columns 1 to 4 shown below, the instructions

CHOOSE rows with **6.0** in column **1** and put in column **5**
RESET 15
CHOOSE rows with numbers between **5.0** and **8.0** in column **1** and put in column **6**
RESET 15
CHOOSE rows with **6.0** in col **1** corr row of col **2** put in cols **7** and **8**
RESET 15
CHOOSE rows between **5.0** and **8.0** in col **1** corr rows of col **2** put in cols **9** and **10**

would put data in columns 5 to 10 as shown in the following table:

Row/Column	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	6	25	40	101	6	6	6	25	6	25
2	7	23	48	−1	6	7	6	25	7	23
3	7	25	40	104	6	7	6	−3	7	25
4	6	25	49	109	6	6	6	25	6	25
5	−2	26	40	106	6	7	6	23	7	24
6	7	24	48	102	6	6	6	25	6	−3
7	6	−3	40	103		6			6	25
8	6	25	49	108		6			6	23
9	6	23	−2	105		7			7	24
10	7	24	48	108		6			6	25
11	−1	24	48	102		7			7	26
12	6	25	49	101		7			7	25
13	7	26	48	−1		7			7	26
14	7	25	48	103		0			0	0
15	7	26	49	107		0			0	0

For the first CHOOSE instruction, the number 6 occurs in the six rows 1, 4, 7, 8, 9, and 12 of column 1. The numbers in the other rows of column 1 are not put into column 5. NRMAX is reset to 6=15−9. For the second CHOOSE instruction, all the numbers in column 1 are between 5 and 8, except the numbers in rows 5 and 11. The numbers in rows 5 and 11 of column 1 are not put into column 6. NRMAX is reset to 13=15−2. The third CHOOSE instruction behaves just like the first and in addition for each row that has a six in column 1 the corresponding row of column 2 is put into column 8. NRMAX is reset to 6=15−9. The fourth CHOOSE instruction behaves just like the second, but whenever a row from column 1 is put into column 9 the same row of column 2 is put into column 10.

CODE column **C** using length **K** put in column **C**

The CODE instruction divides the numbers in the first column into intervals of length **K**; starting with the smallest number in the column. All numbers in the first interval are assigned the value 1, all numbers in the second interval are assigned the value 2, etc. and the results are put in the column designated by the third argument. The instruction is useful for aggregating values to be treated alike. For example, all persons aged 10 to 14 could be put into one class. If the numbers 12.6, 36.4, 27.1, 6.2 and 3.7 are in column 1, the instruction

CODE column **1** using length **5.0** put in column **4**

would put the numbers 2, 7, 5, 1, and 1 into column 4.

CODE column **C** starting at **K** using length **K** put in column **C**

This form of the CODE instruction is the same as the one above, except the starting value, indicated by the second argument, is specified instead of being assigned automatically the minimum value in the column. All numbers in the first column which are less than the specified starting value are assigned the coded value 0.0.

For the numbers 12.6, 36.4, 27.1, 6.2, and 3.7 in column 1, the instruction

CODE column **1** starting at **0.0** using length **5.0** put in column **2**

would put the numbers 3, 8, 6, 2, and 1 into column 2. On the other hand, the instruction

CODE column **1**, starting at **10.0**, using length **5.0**, put in column **3**

would put the numbers 1, 6, 4, 0, and 0 into column 3. In the last instruction, the starting value 10.0 exceeds the minimum value 3.7. The values 6.2 and 3.7, which are below the starting value, are coded 0.0.

DELETE rows having **K** in any columns **C** ,..., **C** put in columns **C** ,..., **C**

If the number **K** is in a row of any column in the first set of columns, that row is not put into the second set of columns. The other rows in the first set of columns are retained in the second set of columns. Zeros are put at the bottom of the second set of columns and NRMAX is reset to the number of rows retained. The instruction

DELETE all rows having **−1.0** in any of cols **1,2,3**, and **4** and put in cols **11,12,13**, and **14**

would produce the following results for the numbers shown in columns 1 to 4:

Row/Column	*1*	*2*	*3*	*4*	*11*	*12*	*13*	*14*
1	6	25	40	101	6	25	40	101
2	7	23	48	−1	7	25	40	104
3	7	25	40	104	6	25	49	109
4	6	25	49	109	−2	26	40	106
5	−2	26	40	106	7	24	48	102
6	7	24	48	102	6	−3	40	103
7	6	−3	40	103	6	25	49	108
8	6	25	49	108	6	23	−2	105
9	6	23	−2	105	7	24	48	108
10	7	24	48	108	6	25	49	101
11	−1	24	48	102	7	25	48	103
12	6	25	49	101	7	26	49	107
13	7	26	48	−1				
14	7	25	48	103				
15	7	26	49	107				

The number −1 occurs in row 2 of column 4, row 11 of column 1, and row 13 of column 4. Hence, rows 2, 11, and 13 are deleted when the results are stored in columns 11 to 14. NRMAX is reset to 15−3=12.

DELETE rows with numbers between **K** and **K** in **C**, ..., **C** put in **C**, ..., **C**

All rows in the first set of columns with a number between **K** and **K**, inclusive, are not included in the second set of columns. The other rows in the first set of columns are retained in the second set of columns. Zeros are put at the bottom of the second set of columns and NRMAX is reset to the number of rows retained.

The instruction

DELETE all numbers between **−10.0** and **0.0** in cols **1,2,3**,and **4** put in cols **15,,16,17**, and **18**

would produce the following results for the numbers shown in above columns 1 to 4:

Row/Column	*1*	*2*	*3*	*4*	*15*	*16*	*17*	*18*
1	6	25	40	101	6	25	40	101
2	7	23	48	−1	7	25	40	104
3	7	25	40	104	6	25	49	109
4	6	25	49	109	7	24	48	102
5	−2	26	40	106	6	25	49	108
6	7	24	48	102	7	24	48	108
7	6	−3	40	103	6	25	49	101
8	6	25	49	108	7	25	48	103
9	6	23	−2	105	7	26	49	107
10	7	24	48	108				
11	−1	24	48	102				
12	6	25	49	101				
13	7	26	48	−1				
14	7	25	48	103				
15	7	26	49	107				

In this example, the six rows 2, 5, 7, 9, 11, and 13 contain numbers in columns 1 to 4 which are in the range −10.0 to 0.0 and are deleted. All the numbers in the remaining rows of columns 1 to 4 are outside the range −10.0 to 0.0 and these rows are retained. NRMAX is reset to 15−6=9.

OMIT rows with **K** in column **C** and put in column **C**

If a number in the first column is not equal to **K**, it is put in the second column. If a number in the first column is equal to **K**, zero is put at the bottom of the second column. NRMAX is reset to the number of rows retained in the second column.

OMIT rows with numbers between **K** and **K** in column **C** and put in column **C**

If a number in the first column is not between **K** and **K**, inclusive, it is put in the second column. If a number in the first column is in the interval **K** to **K**, zero is put at the bottom of the second column. NRMAX is reset to the number of rows retained in the second column.

OMIT rows with **K** in column **C**, corresponding rows of **C** ,..., **C** put in **C** ,..., **C**

The number of arguments in this form of OMIT is 1+2n, where n=1, or 2, or 3, ..., up to 49. There are two sets of n column numbers. If **K** is found in any row of the first column of the first set of columns, that row of every column in the first set is not included in the second set of columns. Zeros are put at the bottom of the second set of columns and NRMAX is reset.

OMIT from **K** to **K** in **C**, corresponding rows of **C** ,..., **C** put in **C** ,..., **C**

This fourth form of OMIT bears the same relation to the second form as the the third form does to the first. The instruction has 2+2n arguments, where n can be any number from 1 to 48.

For the data in columns 1 to 4 shown below, the instructions

OMIT rows with **−2.0** in column **1** and put in column **5**
RESET 15
OMIT rows with numbers between **−10.0** and **0.0** in column **1** and put in column **6**
RESET 15
OMIT rows with **−2.0** in col **1** corr row of col **2** put in cols 7 and **8**
RESET 15
OMIT rows between **−10.0** and **0.0** in col **1** corr rows of col **2** put in cols **9** and **10**

would put data in columns 5 to 10 as shown in the following table:

Row/Column	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	6	25	40	101	6	6	6	25	6	25
2	7	23	48	−1	7	7	7	23	7	23
3	7	25	40	104	7	7	7	25	7	25
4	6	25	49	109	6	6	6	25	6	25
5	−2	26	40	106	7	7	7	24	7	24
6	7	24	48	102	6	6	6	−3	6	−3
7	6	−3	40	103	6	6	6	25	6	25
8	6	25	49	108	6	6	6	23	6	23
9	6	23	−2	105	7	7	7	24	7	24
10	7	24	48	108	−1	6	−1	24	6	25
11	−1	24	48	102	6	7	6	25	7	26
12	6	25	49	101	7	7	7	26	7	25
13	7	26	48	−1	7	7	7	25	7	26
14	7	25	48	103	7		7	26		
15	7	26	49	107						

For the first OMIT instruction, there is a −2 in row 5 of column 1 and this row is omitted in column 5. NRMAX is reset to 15−1=14. For the second OMIT instruction, the numbers in rows 5 and 11 in column 1 are between −10.0 and 0.0 and these rows are omitted in column 6. NRMAX is reset to 15−2=13. For the third OMIT instruction, there is a −2 in row 5 of column 1 and hence row 5 is omitted from columns 7 and 8. NRMAX is reset to 15−1=14. For the fourth (and last) OMIT instruction, the numbers in rows 5 and 11 in column 1 are between −10.0 and 0.0 and hence rows 5 and 11 of columns 1 and 2 are omitted in columns 9 and 10. NRMAX is reset to 15−2=13.

RECODE column **C** into column **C**

The number 1 is assigned to the smallest number in the first column, the number 2 to the second smallest number, and so on and the result is stored in the column designated by the second argument. If the numbers 40, 49, 48, 40, 49, and 48 are in column 2, the instruction

RECODE column **2** into column **5**

would put the numbers 1, 3, 2, 1, 3, and 2 into column 5.

REPLACE the number **K** in column **C** by the number **K** and put in column **C**

The replace instruction is another coding instruction which simply replaces one number by another. It can be used for the treatment of missing or outlying data. For the numbers −1, −9, 2, 5, and −9 in column 5, the instruction

REPLACE −**9.0** in column **5** by **3.0** and put in column **7**

would put the numbers −1, 3, 2, 5, and 3 into column 7.

REPLACE the numbers from **K** to **K** in column **C** by **K**, put in column **C**

This form of REPLACE is the same as the one above, except a range of numbers, instead of just one, is replaced by a single number. For the numbers −1, −9, 2, 5, and −9 in column 5, the instruction

REPLACE numbers from −**3.0** to +**3.0** in col **5** by **7.0** and put in col **9**

would put the numbers 7, −9, 7, 5, and −9 into column 9.

RETAIN numbers between **K** and **K** in columns **C** ,..., **C** put in **C** ,..., **C**

Any row in the first set of columns with all numbers between **K** and **K**, inclusive, is put at the top of the second set of columns. Any row in the first set of columns which has a number outside the range **K** to **K** is deleted. Zeros are put at the bottom of the second set of columns and NRMAX is reset to the number of rows retained. The instruction

RETAIN all numbers between **0.0** and **200.0** in cols **1**,**2**,**3**,and **4** put in cols **11**,**12**,**13** and **14**

would produce the following results for the numbers shown in columns 1 to 4:

Row/Column	*1*	*2*	*3*	*4*	*11*	*12*	*13*	*14*
1	6	25	40	101	6	25	40	101
2	7	23	48	−1	7	25	40	104
3	7	25	40	104	6	25	49	109
4	6	25	49	109	7	24	48	102
5	−2	26	40	106	6	25	49	108
6	7	24	48	102	7	24	48	108
7	6	−3	40	103	6	25	49	101
8	6	25	49	108	7	25	48	103
9	6	23	−2	105	7	26	49	107
10	7	24	48	108				
11	−1	24	48	102				
12	6	25	49	101				
13	7	26	−1	−1				
14	7	25	48	103				
15	7	26	49	107				

In this example, the six rows 2, 5, 7, 9, 11, and 13 contain numbers in columns 1 to 4 which are outside the range 0.0 to 200.0 and are deleted. All the numbers in the remaining rows of columns 1 through 4 are within the range 0.0 to 200.0 and these rows are retained. NRMAX is reset to 6=15−9.

6. 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 C6.2 through C6.9 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. To supplement the material here and to aid the non-statistician, a set of references is provided in section C6.10.

Several sections 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 B1.9.) 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

THE INSTRUCTION WAS IGNORED BECAUSE ...
COMMAND BEGINS WITH S AND STORAGE MUST BE REQUESTED.

The instructions in sections 6.3, 6.4, 6.5 and 6.6 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

THE INSTRUCTION WAS IGNORED BECAUSE ...
ALL WEIGHTS ARE ZERO.

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 NRMAX is not changed.

6.1 Basic Statistics

AVERAGE, FREQUENCY, MAXIMUM, MEDIAN, MINIMUM, RANGE, RANKS, STDDEV

These instructions enable the user to gather basic statistics in the analysis of one or more data sets. Except for the FREQUENCY instruction, the same basic statistics can also be obtained with the STATISTICAL analysis instruction (C6.3).

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.

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

$$\bar{x} = \sum_{i=1}^{n} x_i/n$$

and puts the result into each row of the 2nd named column.

AVERAGE of columns **C**, **C** ,..., **C** put in columns **C**, **C** ,..., **C**

With this optional form of AVERAGE one can compute the averages of several columns with a single instruction.

FREQUENCY distribution of column **C** put in column **C**

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.

The number of classes, the class length, and the lower boundary of the first class, are all determined by the instruction. The number of classes is computed to be the integral part of 1.0 + 3.3log (NRMAX), but never less than 5. The class length 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 is equal to the smallest datum minus one half the class length.

NRMAX is reset to agree with the number of classes. The following data, which are the actual 39 measurements of the velocity of light given by Mandel (1964) (coded by subtracting 299,799.0 from each measurement), are entered into column 1 of the worksheet.

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

The instruction, using the above data,

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.

The instruction STATISTICAL analysis (sec. C6.3) automatically prints a frequency distribution using 10 classes. The computational method is different from that described here.

FREQUENCY distribution of column **C**, using **k** classes, put in column **C**

The number of classes, **k** an integer, is specified by the user. The class length and lower boundary are determined by the instruction as described in the above instruction.

The instruction, using the above data,

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.

If the number of classes exceeds the number of rows in the worksheet, the following informative diagnostic is printed:

COLUMN NOT LONG ENOUGH TO STORE ALL NUMBERS.
FIRST (n) NUMBERS WERE STORED.

FREQUENCY distribution of column **C**, using **k** classes of length **K**, put in column **C**

Both the number of classes, **k** an integer, and the class length, **K** a constant, are specified by the user. The lower boundary is determined by the instruction in the same manner as in the first form of the instruction.

The instruction, using the above data,

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, using k classes of length K starting at K put in C

The number of classes, k an integer, the class length, **K** a constant, and the lower boundary of the first class, **K** a constant, are specified by the user.

The instruction, using the Mandel data,

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.

If the specified starting point **K** of a FREQUENCY distribution instruction exceeds the smallest number in the data column, data below the starting point are ignored. If the numbers 1.0 to 10.0 are in column 2, the instruction:

FREQUENCY of col **2**, using **3** classes of length **2.0**, starting at **4.5**, put in col **7**

will store the frequency 2.0 in the first three rows of column 7 for the three classes:

4.5 to 6.5
6.5 to 8.5
8.5 to 10.5

The four measurements 1.0, 2.0, 3.0 and 4.0 are ignored and the total frequency count is 6 rather than 10. Furthermore, no diagnostic is given. One can deliberately set the starting point above the minimum to advantage, but care should be exercised in doing it.

A similar situation exists if the specified number of classes k is not large enough to include all the data. For the above example, the instruction:

FREQUENCY of col 2, using **3** classes of length **2.0**, starting at **0.5**, put in col **8**

will store the frequency 2.0 in the first three rows of column 8 for the three classes:

0.5 to 2.5
2.5 to 4.5
4.5 to 6.5

The total frequency count is six rather than ten, and the numbers 7.0, 8.0, 9.0 and 10.0 are ignored. Again, no diagnostic is given.

FREQUENCY of C, put lower boundaries in C upper in C and frequencies in C

This instruction is the same as the first form of FREQUENCY, except the lower and upper boundaries of each class are stored in the columns specified by the second and third arguments.

The instruction, using the Mandel data,

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 $1+3.3\times\log(39)=1+3.3(1.59016)=6.2475$ 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** using **k** classes, put bounds and frequencies in **C**, **C** and **C**

This instruction is the same as the second form of FREQUENCY, except the lower and upper boundaries of each class are stored in the columns specified by the third and fourth arguments.

The instruction, using the Mandel data,

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 columns **C**, **C** and **C**

This instruction is the same as the third form of FREQUENCY, except the lower and upper boundaries of each class are stored in the column specified by the fourth and fifth arguments.

The instruction, using the Mandel data,

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**

This instruction is the same as the fourth form of FREQUENCY, except the lower and upper boundaries of each class are stored in the columns specified by the fifth and sixth arguments.

The instruction, using the Mandel data,

FREQUENCY of col **1** use **5** classes of length **0.5** start at **−0.5** put in **41,42,43**

would reset NRMAX to 5 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**

This instruction is the same as the fourth form of FREQUENCY, except the lower and upper boundaries of each class are stored in the columns specified by the fifth and sixth arguments.

The instruction, using the Mandel data,

FREQUENCY of col **1** use **5** classes of length **0.5** start at **−0.5** put in **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

MAXIMUM value of the numbers in column **C** put in column **C**

The largest value in the 1st named column is put 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 column **C** put in col **C**, corresponding 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.

MEDIAN of column **C** put in column **C**

The mid-value of the ordered data in the first named column is put into the second named column. Let k=NRMAX, number of data entries. For k odd, the median is the $(k+1)/2$ ordered datum value. For k even, the median is the average of the $k/2$ ordered value and $(k/2)+1$ ordered value.

If the numbers 3.1, 17.6, 11.6, 9.2 and 8.3 are in column 17, the instruction

MEDIAN of column **17** put in column **18**

would put the median 9.2 into every row of column 18.

MEDIAN of columns **C**, **C** ,..., **C** put in columns **C**, **C** ,..., **C**

The optional form of the MEDIAN instruction makes it possible to compute the median for any number of columns with a single instruction.

MINIMUM value of the numbers in column **C** put in column **C**

The smallest value in the first named column is put 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 column **C**, put in **C**, corresponding 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.

RANGE of numbers in column **C** put in column **C**

The difference between largest and smallest numbers in the first named column is put in the second named column. If the numbers 3.1, 17.6, 9.2, 11.6 and 8.3 are in column 31, the instruction

RANGE of column **31** put in column **32**

would put the number 14.5 in every row of column 32.

RANGE of columns **C**, **C** ,..., **C**, put in columns **C**, **C** ,..., **C**

The optional form of the RANGE instruction makes it possible to compute the range for several columns of data.

RANKS of column **C** put in column **C**

The ranks, as defined in Kendall (1948), of the numbers in the first named column are put in the second named column. In the event of a tie (two or more numbers equal), an average rank is assigned to the numbers in the tie, e.g. the numbers 3, 3, 7, 7 and 7 would have ranks 1.5, 1.5, 4, 4 and 4. A value T, which is useful for making adjustments for ties in statistics such as the rank correlation coefficient, is put into the first row below row NRMAX (unless NRMAX equals the number of rows in the worksheet). If there are no ties, $T=0$. If there are k ties, each of size $t_i (i=1,...,k)$, then

$$T=\frac{1}{12}\sum_{i=1}^{k}(t_i-1)t_i(t_i+1)$$

In the above example, T=(1/12)x(1x2x3+2x3x4)=(1/12)x30=2.5 would be stored in row 6.
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.

STDDEV of numbers in column **C** put in column **C**

The standard deviation, as defined in section C6.3, of the numbers in the first named column is put in the second named column. If the numbers 9000, 9001 and 9003 are in column 6, the instruction

STDDEV of column **6** put in column **7**

would put the number 1.5275252 in every row of column 7.

STDDEV of columns **C**, **C** ,..., **C** put in columns **C**, **C** ,..., **C**

The optional form of the STDDEV instruction makes it possible to compute the standard deviations of numbers in several columns.

6.2 Statistical Plots

HISTOGRAM, NHISTOGRAM, STATPLOTS, STEM LEAF, SSTEM LEAF

A complete description of these commands is given in section C3.8. They are included in this section for completeness.

HISTOGRAM using mid-points in column **C** and frequencies in column **C**

HISTOGRAM for class boundaries in columns **C** and **C** and frequencies in column **C**

NHISTOGRAM using mid-points in column **C** and frequencies in column **C**

NHISTOGRAM for class boundaries in columns **C**, **C** and frequencies in column **C**

STATPLOTS of column **C**

STEM LEAF of column **C**

STEM LEAF of column **C** and put scrawl in column **C**

STEM LEAF of column **C**, put scrawl in column **C** and depth in column **C**

STEM LEAF of column **C** for parameter values **i**, **j**, **k** and **l**

STEM LEAF of column **C** for **i**, **j**, **k** and **l** put scrawl in column **C**

STEM LEAF of column **C** for **i**, **j**, **k**, **l** put scrawl in **C** and depth in **C**

SSTEM LEAF of column **C** put scrawl in column **C**

SSTEM LEAF of column **C** put scrawl in column **C** and depth in column **C**

SSTEM LEAF of column **C** for **i**, **j**, **k** and **l**, put scrawl in column **C**

SSTEM LEAF of **C** for **i**, **j**, **k** and **l**, put scrawl in **C** and depth in **C**

6.3 Analysis of One Column of Data

STATISTICAL, SSTATISTICAL

A STATISTICAL analysis instruction automatically prints (unless suppressed by using SSTATISTICAL), on the first page, statistics for a single column of data, the frequency distribution of the data with ten classes and the frequency distribution of the least significant digit of the data. This page is followed by the printing of the data, the ranks of the data, the deviations (residuals) about the mean (average), the weights if specified, the ordered observations and the differences between adjacent ordered observations.

STATIS is a shortened form of STATISTICAL analysis. There are 6 different forms of STATIS, plus 4 more (SSTATIS) which suppress the automatic printing. 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

A user's guide by Ku (1973) explains many, but not all, of the statistics that are produced by this instruction.

A BRIEF instruction suppresses all pages after the first page of the automatic printing. See section C1.6.

The user of a STATISTICAL analysis would be well advised to use a STAPLOTS (see C3.8) instruction in conjunction with a STATISTICAL analysis in most cases.

STATISTICAL analysis of column **C**

The STATISTICAL instruction with one argument prints the comprehensive set of statistics using weights equal to one and does not store any results in the worksheet. The following set of instructions with 39 actual measurements of the velocity of light given in Mandel (1964)

```
SET VELOCITY IN COL 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
STATISTICAL ANALYSIS OF COLUMN 1
```

produces the following automatic output.

```
STATISTICAL ANALYSIS OF COLUMN     1

  NUMBER OF MEASUREMENTS =     39, NO. OF DISTINCT MEASUREMENTS =     17

      FREQUENCY DISTRIBUTION WITH 10 CLASSES OF LENGTH .24
         5     3     8     3     7     7     5     0     0     1
      FREQUENCY DISTRIBUTION OF LEAST SIGNIFICANT DIGIT (0,1,...,9)
         6     4     6     3     4     4     4     4     3     1
MEASURES OF LOCATION
   ARITHMETIC MEAN . . . . . .        .41025640
   MEDIAN . . . . . .                 .50000000
   MID-RANGE . . . . . .              .70000000
   MID-MEAN (25 PERCENT TRIMMED MEAN) . .        .42380952

MEASURES OF DISPERSION
   STANDARD DEVIATION . . . . . .     .50668939
     AS PERCENT OF MEAN (COEF. OF VAR.) .    123.50554
   RANGE . . . . . .                 2.4000000
   MEAN DEVIATION . . . . . .         .40486522
   INTER-QUARTILE RANGE . . . . . .   .70000000
   VARIANCE . . . . . .               .25673414

STANDARD DEVIATION OF MEAN. . . . . .        .081135237
TREND STATISTICS
   SLOPE, SIGNIFICANCE LEVEL . . . . . .      -.0040080961        .585
   QUADRATIC COEFF., SIGNIFICANCE LEVEL .       .0026019551       .000
 OTHER TESTS FOR NON-RANDOMNESS
   NUMBER OF RUNS UP AND DOWN, Z VALUE  .     23                -1.037
   MEAN SQUARE SUCCESSIVE DIFFERENCE  . .       .28289473
      MS SUCC DIFF/2(VARIANCE), Z VALUE .       .55094880       -2.888
   DEVIATIONS FROM ARITHMETIC MEAN
      NUMBER OF - SIGNS, + SIGNS  . . . .     19                20
      NUMBER OF RUNS, Z VALUE . . . . . .      8                -4.056
   AUTOCORRELATION COEFFICIENT  . . . . .       .43572843
A TWO-SIDED 95 PERCENT CONFIDENCE INTERVAL FOR  THE
   MEAN IS     .24600670 TO  .57450610
   MEDIAN IS   .20000000 TO  .59999999          (DISTRIBUTION-FREE)
   S.D. IS     .41070743 TO  .64762166
```

```
STATISTICAL TOLERANCE INTERVAL WITH 95 PERCENT CONFIDENCE FOR
   50 PCT NORMAL COVERAGE IS  -.017515361 TO   .83802817
   95 PCT NORMAL COVERAGE IS  -.83254328  TO  1.6530561
   99 PCT NORMAL COVERAGE IS -1.2228149   TO  2.0433277
   INTERVAL FROM MIN TO MAX HAS DIST.-FREE COVERAGE 88.40
OTHER STATISTICS
   MINIMUM . . . . . .        -.50000000
      SECOND MINIMUM . . . .  -.40000000
   MAXIMUM . . . . . .        1.9000000
      SECOND MAXIMUM . . . .  1.1000000
   (MEAN-MINIMUM)/STANDARD DEVIATION . .  1.7964781
   (MAXIMUM-MEAN)/STANDARD DEVIATION . .  2.9401515
   SQRT(B1), SKEWNESS COEFFICIENT . . . .   .31064667
   B2, KURTOSIS COEFFICIENT . . . . . .   3.3379327
   LOWER QUARTILE . . . . . .             0.
   UPPER QUARTILE . . . . . .              .70000000


            OBSERVATIONS                          ORDERED OBSERVATIONS
  I      X(I)        RANK   X(I)-MEAN        NO.      X(J)     X(J+1)-X(J)
  1    .40000000     18.5  -.01025641        24   -.50000000   .1000000
  2    .59999999     24.5   .1897436         16   -.40000000   0.
  3   1.0000000      35.5   .5897436         19   -.40000000   .1000000
  4   1.0000000      35.5   .5897436         17   -.30000000   0.
  5   1.0000000      35.5   .5897436         20   -.30000000   .1000000
  6    .50000000     21.0   .08974360        28   -.20000000   .1000000
  7    .59999999     24.5   .1897436         22   -.10000000   0.
  8    .70000000     28.5   .2897436         26   -.10000000   .1000000
  9   1.0000000      35.5   .5897436         15   0.           0.
 10    .59999999     24.5   .1897436         18   0.           .1000000

 11    .20000000     14.0  -.2102564         21    .10000000   .1000000
 12   1.9000000      39.0  1.489744          11    .20000000   0.
 13    .20000000     14.0  -.2102564         13    .20000000   0.
 14    .40000000     18.5  -.01025641        23    .20000000   0.
 15   0.             9.5   -.4102564         27    .20000000   0.
 16   -.40000000     2.5   -.8102564         35    .20000000   .1000000
 17   -.30000000     4.5   -.7102564         25    .30000000   .1000000
 18   0.             9.5   -.4102564          1    .40000000   0.
 19   -.40000000     2.5   -.8102564         14    .40000000   .1000000
 20   -.30000000     4.5   -.7102564          6    .50000000   0.

 21    .10000000     11.0  -.3102564         30    .50000000   0.
 22   -.10000000     7.5   -.5102564         36    .50000000   .09999999
 23    .20000000     14.0  -.2102564          2    .59999999   0.
 24   -.50000000     1.0   -.9102564          7    .59999999   0.
 25    .30000000     17.0  -.1102564         10    .59999999   0.
 26   -.10000000     7.5   -.5102564         31    .59999999   .1000000
 27    .20000000     14.0  -.2102564          8    .70000000   0.
 28   -.20000000     6.0   -.6102564         33    .70000000   0.
 29    .80000000     32.0   .3897436         34    .70000000   0.
 30    .50000000     21.0   .08974360        37    .70000000   .1000000

 31    .59999999     24.5   .1897436         29    .80000000   0.
 32    .80000000     32.0   .3897436         32    .80000000   0.
 33    .70000000     28.5   .2897436         38    .80000000   .2000000
 34    .70000000     28.5   .2897436          3   1.0000000    0.
 35    .20000000     14.0  -.2102564          4   1.0000000    0.
 36    .50000000     21.0   .08974360         5   1.0000000    0.
 37    .70000000     28.5   .2897436          9   1.0000000    .09999999
 38    .80000000     32.0   .3897436         39   1.1000000    .8000000
 39   1.1000000      38.0   .6897436         12   1.9000000
```

STATISTICAL analysis of column **C**, put statistics in **C** and next three columns

The STATISTICAL instruction with two arguments performs an unweighted analysis and puts results in the column designated by the second argument and the three successive columns which follow. The statistics in the automatic printing are stored in the column designated by the second argument, the ranks of the

measurements in the next consecutive column, the ordered measurements in the second consecutive column and the residuals (deviations from the mean) in the third consecutive column.

The order in which the statistics are stored, formulas and references for most statistics are given below. The user should be aware that the order of automatic printing is not the same as the order for the stored statistics, and that most of the references do not discuss the weighted case. Furthermore, there are some differences between the statistics that are printed automatically and those that are stored in the worksheet: (i) the value stored in row 27 of the worksheet is D^2/s^2,whereas the corresponding value from the automatic printing is $D^2/2s^2$, and (ii) the statisics stored in rows 38 through 43 are not printed.

Let $y_j (j=1,...,n)$ denote the total set of measurements with weights $w_j (j=1,...,n)$, some but not all of which can be zero. Let x_i denote the subset of the y_j's which have non-zero weights, and let $x_{[i]}$ denote the corresponding ordered measurements with weights $w_{[wi]}$ for $i=1,...,k$. Furthermore, assume that the index i for x denotes some sequence in which the measurements were taken, such as time.

Row	*Formula*	*Name of Statistic*	*Reference*
1.	$n=NRMAX$	N=LENGTH OF COLUMN, NRMAX	Section B1.5
2.	k	NUMBER OF NON-ZERO WEIGHTS	Section C6.3
	MEASURES OF LOCATION		
3.	$\bar{y}=\sum_{i=1}^{n} y_i/n$	UNWEIGHTED MEAN	Dixon and Massey (1957), 14
4.	$\bar{x}=\left[\sum_{i-1}^{k} w_i x_i\right] / \left[\sum_{i=1}^{k} w_i\right]$	WEIGHTED MEAN	Brownlee (1965), 95–97
5.	$x_{[(k+1)/2]}$, k odd, or $(x_{[k/2]}+x_{[(k+1)/2]})/2$, k even	MEDIAN	Dixon and Massey (1957), 70
6.	$(x_{[1]}+x_{[k]})/2$	MID-RANGE	Dixon and Massey (1957), 71
7.	$\sum_{i=1+[n/4]}^{n-[n/4]} x_{[i]}/(n-2[n/4])$	25 PCT UNWTD TRIMMED MEAN	Crow and Siddiqui (1967)
8.	$\dfrac{\sum_{i=1+[k/4]}^{k-[k/4]} w_{[i]}x_{[i]}}{\sum_{i=1+[k/4]}^{k-[k/4]} w_{[i]}}$	25 PCT WTD TRIMMED MEAN	
	MEASURES OF DISPERSION		
9.	$S=\left[\frac{1}{k-1}\sum_{i=1}^{k} w_i(x_i-\bar{x})^2\right]^{1/2}$	WTD STANDARD DEVIATION	Snedecor and Cochran (1967), 44
10.	$\bar{s}_{\bar{x}}=s/\left[\sum_{i=1}^{k} w_i\right]^{1/2}$	S.D. OF WTD MEAN	Brownlee (1965), 97
11.	$R=x_{[k]}-x_{[1]}$	RANGE	Snedecor and Cochran (1967), 39

Row	Formula	Name of Statistic	Reference
12.	$\sum_{i=1}^{k} \left\lvert x_i - \bar{x} \right\rvert / k$	MEAN DEVIATION	Duncan (1965), 50
13.	$s^2 = \sum_{i=1}^{k} w_i (x_i - \bar{x})^2 / (k-1)$	VARIANCE	Snedecor and Cochran (1967), 44
14.	ABS $(100s/\bar{x})$	COEFFICENT OF VARIATION	Snedecor and Cochran (1967), 62
	TWO-SIDED 95 PCT CONFIDENCE INTERVALS		
15.	$\bar{x} - t_{.025} s_{\bar{x}}$, using $(k-1)$ d.f.	LOWER CONFIDENCE LIMIT, MEAN	Natrella (1963), 2-2, 2-3
16.	$\bar{x} + t_{.25} s_{\bar{x}}$, using $(k-1)$ d.f.	UPPER CONFIDENCE LIMIT, MEAN	Natrella (1963), 2-2, 2-3
17.	$s[(k-1)/(x^2_{.975})]^{1/2}$ using $(k-1)$ d.f.	LOWER CONFIDENCE LIMIT, S.D.	Natrella (1963), 2-7
18.	$s[(k-1)/(x^2_{.025})]^{1/2}$ using $(k-1)$ d.f.	UPPER CONFIDENCE LIMIT, S.D.	Natrella (1963), 2-7
19.	$B = 12 \sum_{i=1}^{k} i(x_i - \bar{x}) / k(k^2-1)$	SLOPE	Fisher (1950), 136–140
20.	$s_B = \frac{[12\Sigma(x_i - \bar{x})^2/k(k^2-1) - B^2]^{1/2}}{\sqrt{(k-2)}}$	S.D. OF SLOPE	Fisher (1950), 136–140
21.	$t_0 = B/s_B$	STUDENT'S T WITH $(k-2)$ D.F.	
22.	$Pr[t < -\lvert t_0 \rvert$ and $t > +\lvert t_0 \rvert]$	PROB EXCEEDING ABS VALUE OF t_0	Brownlee (1965), 344
	STATISTICS FOR TESTING NON-RANDOMNESS OF RAW DATA		
23.	r	NO. OF RUNS UP AND DOWN	Brownlee (1965), 223
24.	$(2k-1)/3$	EXPECTED NO. OF RUNS	Bradley (1968), 279
25.	$\sqrt{(16k-29)/90}$	S.D. OF NO. OF RUNS	Bradley (1968), 279
26.	$D^2 = \sum_{i=1}^{k-1} (x_{[i+1]} - x_{[i]})^2 / (k-1)$	MEAN SQ SUCCESSIVE DIFF	Brownlee (1965), 222
27.	(D^2/s^2)	STATISTIC FOR TESTING RANDOMNESS	Young (1941), 294 & Brownlee (1965), 221

Row	Formula	Name of Statistic	Reference
	STATISTICS FOR TESTING NON-RANDOMNESS OF DEVIATIONS FROM WEIGHTED MEAN		
28.	U	NO. OF + SIGNS	
29.	v	NO. OF − SIGNS	
30.	1 + no. of sign changes of $(x_i-\bar{x})$	NO. OF RUNS	Brownlee (1965), 224
31.	$1 + (2uv)/k$	EXPECTED NO. OF RUNS	Brownlee (1965), 227
32.	$[2uv(2uv-u-v)/(u+v)^2(k-1)]^{1/2}$	S.D. OF NO. OF RUNS	Brownlee (1965), 230
33.	[(30)−(31)]/(32)	T STATISTIC FOR NO. OF RUNS	Brownlee (1965), 230
	OTHER STATISTICS		
34.	$x_{[1]}$, smallest value	MINIMUM	Natrella (1963), 19-1
35.	$x_{[k]}$, largest value	MAXIMUM	Natrella (1963), 19-3
36.	$\beta_1=k\left[\sum_{i=1}^{k}(x_i-\bar{x})^3\right]^2/\left[(k-1)s^2\right]^3$	BETA ONE	Snedecor and Cochran (1967), 86
37.	$\beta_2=k\sum_{i=1}^{k}(x_i-\bar{x})^4/[(k-1)s^2]^2$	BETA TWO	Snedecor and Cochran (1967), 87
38.	$\sum_{i=1}^{k} w_i x_i$	WTD SUM OF VALUES	
39.	$\sum_{i=1}^{k} w_i x_i^2$	WTD SUM OF SQUARES	
40.	$\sum_{i=1}^{k} w_i(x-\bar{x})^2$	WTD SUM OF DEVS SQUARED	
41.	$t=\left[\sum_{i=1}^{k} w_i\right]^{1/2}\bar{x}/s$	STUDENT'S T WITH (k−1) D.F. FOR TESTING MEAN=0	Brownlee (1965), 296

Row	Formula	Name of Statistic	Reference
42.	$\sum_{i=1}^{k} w_i \lvert x_i \rvert$	WTD SUM ABSOLUTE VALUES	
43.	$\left[\sum_{i=1}^{k} w_i \lvert x_i \rvert\right] / \left[\sum_{i=1}^{k} w_i\right]$	WTD AVE ABSOLUTE VALUES	
44.	$x_{[2]}$	SECOND MINIMUM	Natrella (1963), 17-4
45.	$x_{[k-1]}$	SECOND MAXIMUM	Natrella (1963), 17-4
46.	$(\bar{x}-x_{[1]})/s$	(MEAN-MINIMUM)/S	Natrella (1963), 17-5
47.	$(\bar{x}-x_{[k]})/s$	(MEAN-MAXIMUM)/S	Natrella (1963), 17-5
48.	$\sqrt{\beta_1}$	SQRT (β_1)	Snedecor and Cochran (1967), 86

Let r be the integral part of m. Let $p=m-r$ be the fractional part of m and let $q=1-p$.

49.	$Q_1=qx_{[r]}+px_{[r+1]}$ where $m=(k+1)/4$	LOWER QUARTILE	Snedecor (1956), 111
50.	$Q_3=px_{[r]}+qx_{[r+1]}$, where $m=3(n+1)/4$	UPPER QUARTILE	Snedecor (1956), 111
51.-60.		FREQUENCY DISTRIBUTION	Freund and Williams (1958), 17

MEASURES OF DISPERSION

61.	Q_3-Q_1	INTER—QUARTILE RANGE	Snedecor (1956), 111

QUADRATIC TREND STATISTICS

62.		QUADRATIC COEFFICIENT	Snedecor and Cochran (1967), 453
63.		QUADRATIC SIGNIFICANCE LEVEL	Snedecor and Cochran (1967), 453

Row	Formula	Name of Statistic	Reference
	OTHER TESTS FOR NON-RANDOMNESS		
64.	r_{uv}, where $u=x_1, x_2,\ldots,x_{k-1}$ and $v=x_2, x_3,\ldots, x_k$.		
	The correlation coefficient, r_{xy}, is defined on page 204		
	TWO-SIDED 95 PCT CONFIDENCE INTERVALS ON MEDIAN		
65.	$x_{[a]}$ where a is the value of c which makes $\sum_{i=0}^{c} \binom{k}{i} 2^{-k}$ closest to 0.025	LOWER CONFIDENCE LIMIT, MEDIAN	Brownlee (1965), 183
66.	$x_{[k+1-a]}$	UPPER CONFIDENCE LIMIT, MEDIAN	Brownlee (1965), 183
	95 PCT TOLERANCE INTERVALS		
	Gau(x) is the cumulative normal distribution defined on page 235		
67.	$T_L=\bar{x}-Ks$ where $Pr(Gau(T_U)-Gau(T_L)=0.50)=0.95$	LOWER LIMIT, 50% COVERAGE	Natrella (1963) & Gardner&Hull (1966)
68.	$T_U=\bar{x}+Ks$ where $Pr(Gau(T_U)-Gau(T_L)=0.50)=0.95$	UPPER LIMIT, 50% COVERAGE	Natrella (1963) & Gardner&Hull (1966)
69.	$T_L=\bar{x}-Ka$ where $Pr(Gau(T_U)-Gau(T_L)=0.95)=0.95$	LOWER LIMIT, 95% COVERAGE	Natrella (1963) & Gardner&Hull (1966)
70.	$T_U=\bar{x}+Ks$ where $Pr(Gau(T_U)-Gau(T_L)=0.95)=0.95$	UPPER LIMIT, 95% COVERAGE	Natrella (1963) & Gardner&Hull (1966)
71.	$T_L=\bar{x}-Ks$ where $Pr(Gau(T_L)-Gau(T_L)=0.99)=0.95$	LOWER LIMIT, 99% COVERAGE	Natrella (1963) & Gardner&Hull (1966)
72.	$T_U=\bar{x}+Ks$ where $Pr(Gau(T_L)-Gau(T_L)=0.99)=0.95$	UPPER LIMIT, 99% COVERAGE	Natrella (1963) & Gardner&Hull (1966)
73.	The largest value of p such that $kp^{k-1}-(k-1)p^k\leqslant 0.05$	95% DISTRIBUTION FREE COVERAGE	Natrella (1963), 2-15

STATISTICAL analysis of **C**, weights in **C**, put in **C** and next three columns

The STATISTICAL 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.

When weights equal to zero are used, the analysis is exactly the same as if the corresponding measurements were not used. 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.

STATISTICAL analysisof **C**, weights in **C**, do not put in **−C**

This form of the STATISTICAL 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 previous one.

STATISTICAL analysis of **C**, put statistics in columns **C**, **C**, **C** and **C**

The STATISTICAL instruction with 5 arguments does not use weights and stores statistics in the four columns designated by the last four arguments. Statistics of the automatic printing are stored in the column designated by the second argument, ranks are stored in the column designated by the third argument, ordered measurements are stored in the column designated by the fourth argument and results are stored in the column designated by the last argument.

STATISTICAL analysis of **C**, weights in **C**, put statistics in **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.

*****

The next four instructions are similar to the second, third, fifth and sixth forms of the STATISTICAL analysis instructions except automatic printing is suppressed.

SSTATISTICAL analysis of column **C**, put statistics in **C** and next three columns

SSTATISTICAL analysis of **C**, weights in **C**, put in **C** and next three columns

SSTATISTICAL analysis of **C**, put statistics in columns **C**, **C**, **C** and **C**

SSTATISTICAL analysis of **C**, weights in **C**, put statistics in **C**, **C**, **C** and **C**

6.4 Regression

FIT, POLYFIT, SFIT, SPOLYFIT, LARFIT

The instruction FIT performs a least squares analysis (regression) of a set of data. A measurement (response), y, is assumed to be functionally related to k fixed, known constants, $x_1, x_2, \ldots, x_k$, which are frequently referred to as the independent or predictor variables. For the ith (i=1,2,...,n=NRMAX) measurement, the statistical model is:

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_k x_{ki} + e_i,$$

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.

In the POLYFIT instruction a polynomial of degree d

$$y_i = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \ldots + \alpha_d x^d + e_i,$$

is equivalent to the first model with d=k+1, if x^r is set equal to x_{r+1} and α_r is set equal to β_{r+1}, r=0, 1, ... ,d.

Since POLYFIT and SPOLYFIT are actually special cases of FIT and SFIT respectively, the instructions are discussed jointly. The LARFIT instruction for least absolute residuals estimation is described separately.

The classical least squares estimation procedure for linear models has well known optimal properties when the usual assumptions concerning measurement errors are tenable. Least squares estimation is used by the FIT and POLYFIT instructions. If the underlying error assumptions are not valid, other estimation procedures may be superior. L estimates, otherwise known as LAR (least absolute residuals) or LAD (least absolute deviations) estimates, are obtained by minimizing the sum of the absolute value of the deviations of the model from the observed data. LAR estimates may be appropriate to use in the presence of outliers (unusually large or small measurements).

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 176 to 178. These descriptions are preceded by a detailed discussion under the titles Weights; Restrictions; Example; Notation and Formulas for FIT; Automatic Printing; Brief, Interactive and Width; Storage; Computing Method; and Remarks.

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, 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. An analysis with m non-zero weights is identical to the analysis obtained using only the m measurements with non-zero weights. It is as though the measurements with zero weight were ignored. Standardized residuals with zero weights are always set equal to zero.

Restrictions

Negative weights may not be used. The number of non-zero weights must be greater than or equal to the number of parameters that are estimated, (k or d+1). Below are the maximum number of measurements, NRMAX, which can be used for a given number of parameters, k. In FIT, k is the third argument. In POLYFIT, k equals the degree (3rd argument) plus 1. (Values given are for a standard size worksheet.)

k	*NRMAX*	*k*	*NRMAX*	*k*	*NRMAX*	*k*	*NRMAX*	*k*	*NRMAX*
1	2697*	21	525*	41	275*	61	172	81	113
2	2246*	22	504*	42	268*	62	169	82	111
3	1924*	23	484*	43	261*	63	165	83	109
4	1682*	24	466*	44	254*	64	162	84	107
5	1494*	25	448*	45	248*	65	158	85	104
6	1343*	26	432*	46	242*	66	155	86	102
7	1220*	27	417*	47	236*	67	152	87	100
8	1117*	28	403*	48	231*	68	149	88	98
9	1030*	29	390*	49	225*	69	146	89	96
10	955*	30	377*	50	220*	70	143	90	94
11	890*	31	366*	51	215*	71	140	91	92
12	833*	32	354*	52	210*	72	137	92	***
13	783*	33	344*	53	206*	73	134		
14	739*	34	334*	54	201	74	131		
15	699*	35	324*	55	197	75	129		
16	663*	36	315*	56	192	76	126		
17	630*	37	306*	57	188	77	123		
18	600*	38	298*	58	184	78	121		
19	573*	39	290*	59	180	79	118		
20	548*	40	282*	60	176	80	116		

The asterisk indicates the worksheet must be redimensioned for the larger values of NRMAX. Violations of these restrictions result in the following fatal errors, respectively:

NEGATIVE WEIGHTS MAY NOT BE USED.
DEGREE IS GREATER THAN NUMBER OF NON-ZERO WEIGHTS.
INSUFFICIENT SCRATCH AREA.

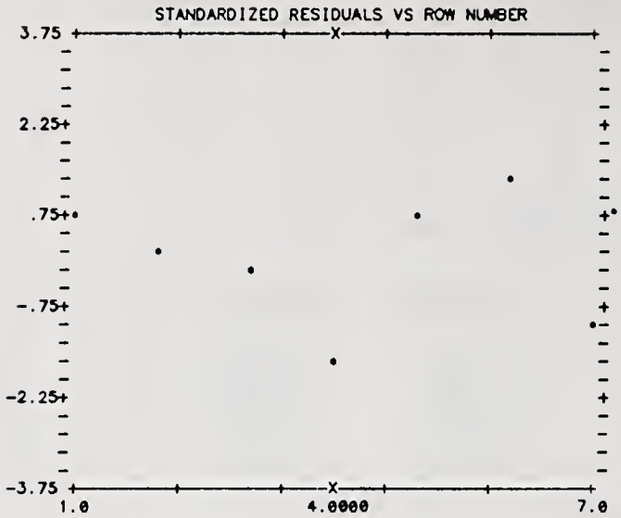
Example

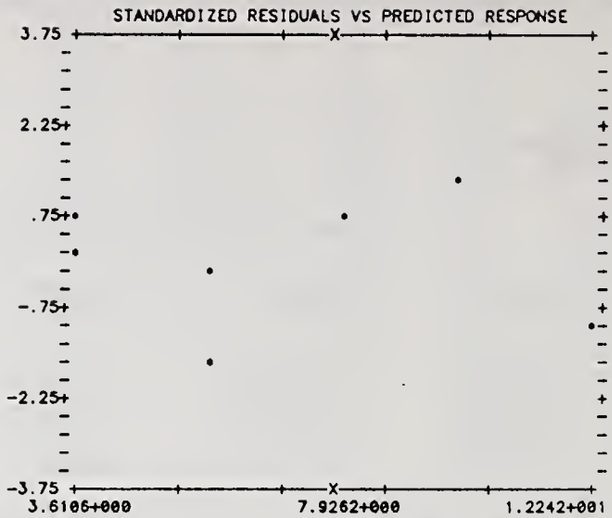
The following instructions illustrate the automatic printing of FIT using data taken from page 314 of Brownlee (1965).

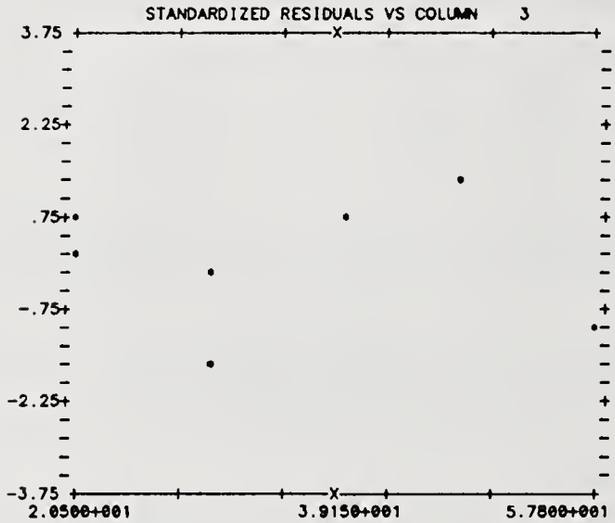
```
OMNITAB 80      BROWNLEE 11.18 LACK OF FIT ANOVA
SET STOPPING DISTANCE OF AUTO IN COLUMN 1
3.92    3.65    5.82    5.20    8.55    10.63    11.94
SET MILES PER HOUR IN COLUMN 3
20.5    20.5    30.5    30.5    40.5    48.8    57.8
DEFINE 1.0 IN COLUMN 2
FIT COLUMN 1 WEIGHTS=1.0, 2 INDEPENDENT VARIABLES IN COLUMNS 2 and 3
POLYFIT COLUMN 1, WEIGHTS 1.0, DEGREE 1, INDEPENDENT VARIABLE IN COLUMN 3
STOP
```

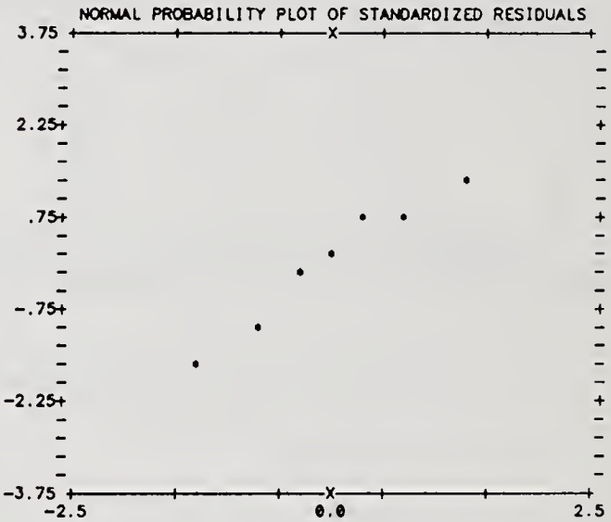
The automatic printing of POLYFIT is essentially the same as that of FIT except a plot of 95% confidence ellipse is generated and is shown on page 168. The results are:

```
OMNITAB 80 BROWNLEE 11.18 LACK OF FIT ANOVA                              PAGE  1

                    LEAST SQUARES FIT OF RESPONSE, COLUMN    1,
              AS A LINEAR FUNCTION OF  2 INDEPENDENT VARIABLES IN COLUMNS    2,  3
                          USING    7 NON-ZERO WEIGHTS = 1.0000000

 ROW   INDEP VAR.  COLUMN  3   RESPONSE  COLUMN  1   PREDICTED RESPONSE   STD. DEV. OF PRED. RESPONSE   RESIDUALS      STD. RES.   WEIGHTS
   1      20.500000              3.9200000              3.6105717              .26341465              .30942826          .83      1.000
   2      20.500000              3.6500000              3.6105717              .26341465              .039428249         .11      1.000
   3      30.500000              5.8200000              5.9245867              .18519791             -.10458669         -.25      1.000
   4      30.500000              5.2000000              5.9245867              .18519791             -.72458670        -1.74      1.000
   5      40.500000              8.5500000              8.2386017              .18439105              .31139833          .75      1.000
   6      48.800000             10.630000              10.159234               .24530475              .47076603         1.22      1.000
   7      57.800000             11.940000              12.241847               .34006014             -.30184748         -.99      1.000

   DIAGNOSTIC INFORMATION FOR IDENTIFYING INFLUENTIAL MEASUREMENTS.
     I  = ROW, FOR  7 LARGEST VALUES. T(I) = STANDARDIZED RESIDUAL.
   H(I) = DIAGONAL OF HAT MATRIX,     D(I) = COOK STATISTIC.
 WSSD(I) = DANIEL-WOOD STATISTIC,     V(I) = VAR(YHAT) / VAR(RESIDUAL).

    I   T(I)      I   H(I)      I   D(I)      I   WSSD(I)      I   V(I)
    4  -1.74      7   .554      7   .61       7   126.68       7   1.24
    6   1.22      1   .333      6   .30       1   58.42        1   .50
    7   -.99      2   .333      4   .30       2   58.42        2   .50
    1    .83      6   .288      1   .17       6   44.82        6   .41
    5    .75      3   .164      5   .05       3   6.64         3   .20
    3   -.25      4   .164      3   .01       4   6.64         4   .20
    2    .11      5   .163      2   .00       5   6.20         5   .19

   THE DURBIN-WATSON STATISTIC IS D =  2.0840896
```


STANDARDIZED RESIDUALS VS ROW NUMBER
3.75
2.25
.75
-.75
-2.25
-3.75
1.0
4.0000
7.0


STANDARDIZED RESIDUALS VS PREDICTED RESPONSE
3.75
2.25
.75
-.75
-2.25
-3.75
3.6106+000
7.9262+000
1.2242+001


STANDARDIZED RESIDUALS VS COLUMN 3
3.75
2.25
.75
-.75
-2.25
-3.75
2.0500+001
3.9150+001
5.7800+001


NORMAL PROBABILITY PLOT OF STANDARDIZED RESIDUALS
3.75
2.25
.75
-.75
-2.25
-3.75
-2.5
0.0
2.5

LEAST SQUARES FIT OF RESPONSE, COLUMN 1,
AS A LINEAR FUNCTION OF 2 INDEPENDENT VARIABLES IN COLUMNS 2, 3
USING 7 NON-ZERO WEIGHTS = 1.0000000

SAMPLE VARIANCE-COVARIANCE MATRIX OF THE ESTIMATED COEFFICIENTS WITH CORRELATIONS ABOVE THE DIAGONAL.

COLUMN	2	3
2	.25008450	-.9385
3	-.0061902779	1.7395402-04

LACK OF FIT ANALYSIS OF VARIANCE
EVIDENCE OF LACK OF FIT HERE MAY INVALIDATE OTHER RESULTS

SOURCE	D.F.	SUM OF SQUARES	MEAN SQUARES	F-VALUE	P(F)
REGRESSION	2	417.22133	208.61066		
RESIDUAL	5	1.0429662	.20859323		
LACK OF FIT	3	.81431615	.27143872	2.374	.310
ERROR	2	.22865000	.11432500		
TOTAL	7	418.26430			

ANALYSIS OF VARIANCE
-DEPENDENT ON ORDER INDEPENDENT VARIABLES ARE ENTERED, UNLESS VECTORS ARE ORTHOGONAL-

INDEP VAR.		SS-RED. DUE TO COEF.	CUM. RESIDUAL MS	D.F.	F(COEF=0)	P(F)	F(COEFS=0)	P(F)
COLUMN	2	353.01201	10.875381	6	1692.346	.000	1000.084	.000
COLUMN	3	64.209318	.20859323	5	307.821	.000	307.821	.000
RESIDUAL		1.0429662		5				
TOTAL		418.26430		7				

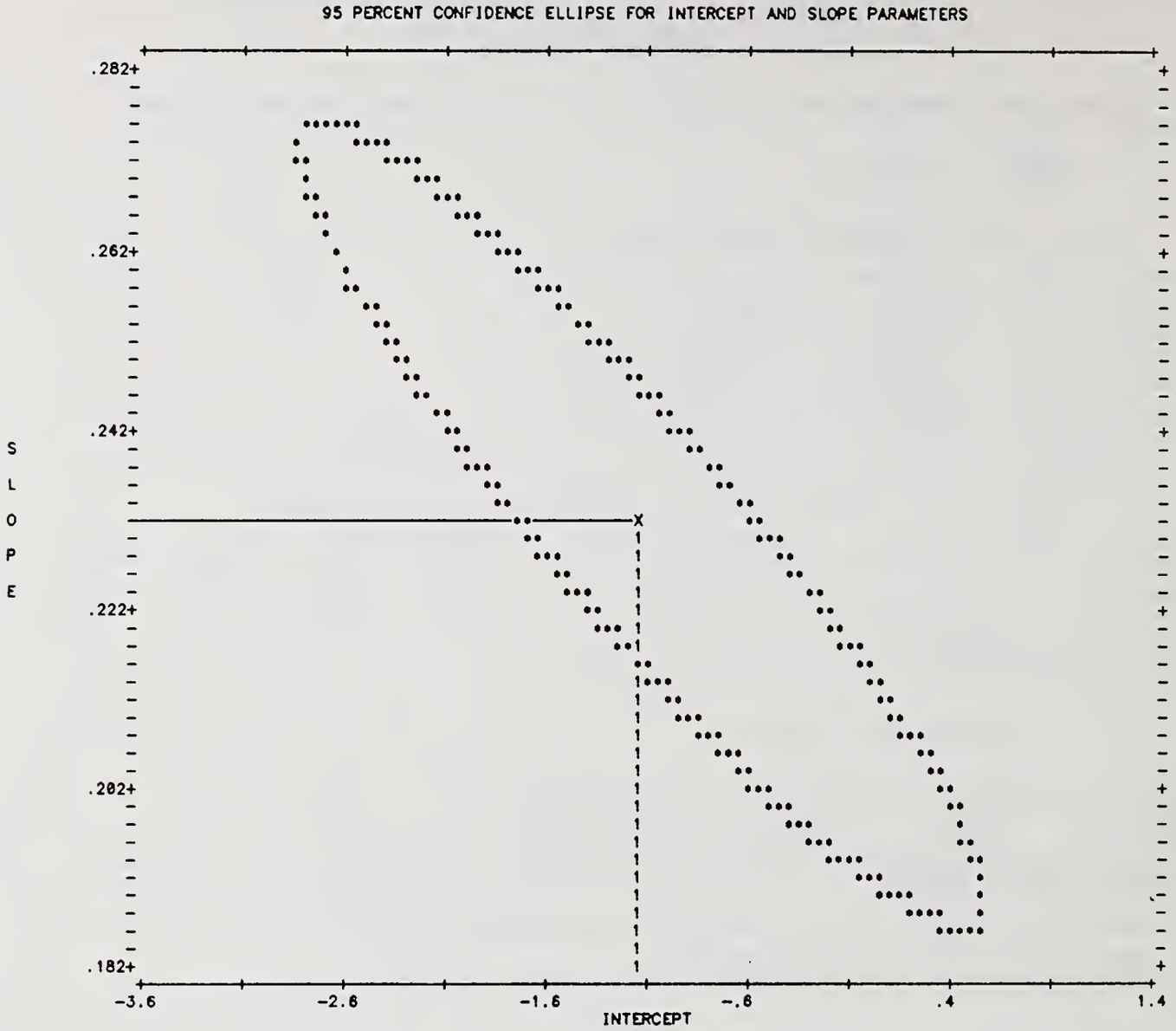
ESTIMATES FROM LEAST SQUARES FIT

INDEP VAR.		COEFFICIENT	S.D. OF COEFF.	RATIO	ACCURACY*
COLUMN	2	-1.1331589	.50008449	-2.27	8.00
COLUMN	3	.23140149	.013189163	17.54	8.00

RESIDUAL STANDARD DEVIATION = .45672008
BASED ON DEGREES OF FREEDOM 7 - 2 = 5

*THE NUMBER OF CORRECTLY COMPUTED DIGITS IN EACH COEFFICIENT USUALLY DIFFERS BY LESS THAN 1 FROM THE NUMBER GIVEN HERE.

THE NUMBER OF ITERATIONS WAS 2. SCALING WAS NOT USED.
THE AVERAGE NUMBER OF DIGITS IN AGREEMENT BETWEEN INITIAL SOLUTION AND 1ST ITERATION IS 7.61.



Notation and Formulas for FIT

The following conventions are adopted for notational ease and are utilized in the ensuing discussions. A capital letter denotes the matrix or vector equivalent of the corresponding lower case letter; subscripts denote the dimensions of the matrix or vector.
For example Y_{nx1} is the (nx1) vector of measurements y_i, $i=1,...,n$.

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_{nxl}
4.	Predictor variables x_1, x_2,...,x_k	X_{nxk}
5.	Measurement errors: e_1, e_2,...,e_n	E_{nxl}
6.	Error variance: σ_i^2, $i=1,...,n$	$\sigma_i^2=k_i\sigma^2$
7.	Unknown parameters: β_1, β_2,...,β_k	B_{kxl}
8.	Statistical model: errors independent, normal, variance σ_i^2	$Y=XB+E$
9.	Weights: "best" weights $w_i=1/\sigma_i^2$ or $1/k_i$, $i=1,...,n$	$W_{nxn}=\text{diag}(w_i=1/k)$
10.	Coefficients: estimates of $\hat{\beta}_1$, $\hat{\beta}_2$,...,$\hat{\beta}_k$	$\hat{B}_{kxl}=(X'WX)^{-1}X'WY$
11.	Predicted values: $\hat{y}_i$, $i=1,...,n$	$\hat{Y}_{nxl}=X\hat{B}$
12.	Residuals: deviations from predicted values z_i, $i=1,...,n$	$Z_{nxl}=Y-\hat{Y}$
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	$V_{kxk}=s^2(X'WX)^{-1}$
15.	Standard deviation of predicted values: $s_{\hat{y}i}$, $i=1,....,n$	$\sqrt{\text{diagonals of } XVX'}$
16.	Standard deviations of coefficients	$\sqrt{\text{diagonals of } V}$
17.	Standardized residuals	$z_i/\sqrt{(s^2/w_i-s_{\hat{y}}^2)}$
18.	Orthonormalization: $X'WX=TT'$, T triangular, $A=(T')^{-1}$	$\phi_{nxk}=XA$
19.	Fourier coefficients: δ_1, δ_2,...,δ_k	$\phi'WY$
20.	Reduction in total SS due to fitting: δ_1^2, δ_2^2,...,δ_k^2	
21.	Residual sum of squares: δ_{k+1}^2	$\delta_{k+1}^2=s^2$
22.	Total (uncorrected) sum of squares: $\sum_{i=1}^{n} w_i y_i^2$	$Y'WY$
23.	Cumulative mean square with r degrees of freedom	$\sum_{i=1}^{r} \delta_i^2/r$
24.	Cumulative residual mean square with $(n-r)$ degrees of freedom: RMS	$\text{RMS}=\sum_{i=r+1}^{k+1} \delta_i^2/(n-r)$
25.	F ratio for testing Ho: $\beta_r=0$	δ_r^2/RMS
26.	F ratio for testing Ho: $\sum_{i=r}^{k} \beta_i=0$	$\left[\sum_{i=r}^{k} \delta_i^2(k-r+1)\right]/\text{RMS}$
27.	Number of accurate digits obtained from refit to predicted values: $\tilde{\beta}_1$, $\tilde{\beta}_2$,..., $\tilde{\beta}_k$	$-\log_{10}\left\|\hat{\beta}_i-\tilde{\beta}_i/\tilde{\beta}_i\right\|$

In the discussion of the automatic printout it is assumed that instructions BRIEF, INTERACTIVE and/or WIDTH of less than 120 characters per line have not been used. The effect of these instructions on the automatic printout is outlined at the end of this section and in sections C1.4, C1.5 and C1.6.

The automatic printing consists of "three" or "more" pages containing a comprehensive set of results for analyzing data. The number 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 and three. The title in the example on pages 165 and 167 is

LEAST SQUARES FIT OF RESPONSE, COLUMN 1
AS A LINEAR FUNCTION OF 2 INDEPENDENT VARIABLES IN COLUMNS 2, 3
USING 7 NON-ZERO WEIGHTS = 1.0000000

In a POLYFIT instruction, the second line of the title has the form

AS A POLYNOMIAL OF DEGREE 1, INDEPENDENT VARIABLE IS COLUMN 3

The following are variations in the title which can occur: (1) on the first line 12 spaces are allowed for the word COLUMN and column number which may be replaced by the information provided by the HEAD or LABEL instruction (sec. C1.3 and C2.3); (2) only column numbers (never the information contained in HEAD and/or LABEL instructions) are printed on line 2 if there is more than one independent variable and, furthermore, column numbers are continued on the next line if there are more than 11 independent variables; and (3) the number of zero weights is given on line 3 if the second argument is a column number.

Page One

Eight items are printed for each of the (NRMAX) measurements: (1) the row in the worksheet, (2) the independent variable, (3) the response (measurement), (4) the predicted response, (5) the standard deviations of the predicted response, (6) the residuals, (7) the standardized residuals and (8) the weights. Items (3), (4), (5) and (6) are printed with eight significant digits. The standardized residuals are printed with two decimal places and four significant digits are used to print the weights. The standardized residuals are the residuals divided by their standard deviation.

If the instruction is FIT, either one, two or a maximum of three columns of independent variables are 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 variable). Otherwise, starting with the first variable not identically equal to one, as many independent 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.

The identification of influential points in linear regression follows for the purpose of detecting outliers and/or points which may have an unusually large influence on the estimation of the unknown coefficients.

Values of the five statistics T, H, D, WSSD, and V are printed in decreasing order starting with the largest. If NRMAX is less than 10, all NRMAX values are printed. If NRMAX is greater than 9 and less than 100, 10 values are printed. If NRMAX is greater than 99 and less than 1000, 15 values are printed. If NRMAX is 1000 or larger, 20 values are printed.

The T statistic is the standardized residual. The standardized residuals are printed in decreasing order of absolute value.

The H statistic is the diagonal of the hat matrix. Its use and interpretation are discussed in Hoaglin and Welsch (1978).

The D statistic, is the Cook statistic proposed and discussed in Cook (1977).

The WSSD, or weighted standardized squared distance, statistic is proposed and discussed in Daniel and Wood (1971).

The V statistic is the ratio of the sample variance of the predicted value divided by the variance of the residual. It is discussed in Cook (1977).

The above references can assist in interpreting the importance, or lack of importance, of individual values. In the analysis of data, the interpretation of any result should be based on a consideration of the particular

problem being studied and all other relevant information. This is particularly true when interpreting the statistics printed for the identification of influential measurements.

If there is evidence that one or more measurements is contributing an undue influence on the model estimates, several courses of action are possible. One discussion of what to do when there are bad points is given in Hettmansperger and McKean (1977).

The Durbin-Watson statistic is printed just below diagnostics for identifying influential measurements. It is used to test for interdependence or non-randomness of the residuals. A discussion of the Durbin-Watson statistic starts on page 243 of Goldberger (1964).

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 (1981) and Anscombe and Tukey (1963). All TITLE instructions are ignored in the printing of this page due to lack of space.

Upper Left Plot. 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.

Upper Right Plot. The standardized residuals are plotted against the predicted responses. Non-randomness may indicate non-constant variance or that some important variable(s) has been excluded from the model. The former case may indicate that there is a need for weights, or that the weights are improper, or that a transformation of the measurements is required.

Lower Left Plot. The standardized residuals are plotted against the response (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 response variables. The response variable used for the plot is the first one which is not identically equal to one (unless the number of vectors is one).

Lower Right Plot. 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

$$x_i = Gau^{-1}(p) = 4.91(p^{.14} - (1-p)^{.14}),$$

where

$$\begin{aligned} p &= (i-\pi/8)/(n+1-\pi/4), && \text{for } i \geq 2 \text{ or } n > 10 \\ &= (3i-1)/(3n+1), && \text{for } i = 1 \text{ and } n \leq 10 \end{aligned}$$

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. James J. Filliben (NBS) provided 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 response variables.

Page Three

The printout on this page consists of four sections: (1) sample variance-covariance matrix, (2) lack of fit analysis of variance, (3) analysis of variance and (4) estimates from least squares fit.

Sample Variance-Covariance Matrix of Estimated Coefficients with Correlations Above The Diagonal. The matrix, $V = s^2(X'WX)^{-1}$, 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, $A'\beta$ where A is a (kx1) vector, is the square root of A'VA. The estimated correlations of the parameter estimates are printed above the diagonal.

Lack of Fit Analysis of Variance. If two or more rows of the independent variables are the same, a replication is said to exist. When there are replications, it may be possible to separate the residual sum of squares into two components with one component due to measurement error and the other component due to lack of fit of of the assumed model to the observed data. In the presence of replications, an analysis of variance test is printed to test the adequacy of the assumed model. The lack of fit information is printed just below the printing of the sample variance- covariance matrix.

If the analysis of variance test shows evidence that the assumed model does not adequately represent the data, as indicated by small (i.e., less than 0.05) values of P(F), other information printed by FIT or POLYFIT may be meaningless.

If there are no replications, the following line is printed:

THERE ARE NO REPLICATIONS FOR A LACK OF FIT ANALYSIS OF VARIANCE.

If the residual degrees of freedom is due entirely to replications, the following information is printed:

THE RESIDUAL SUM OF SQUARES IS DUE ENTIRELY TO REPLICATIONS.
LACK OF FIT ANALYSIS OF VARIANCE NOT POSSIBLE.

Page 167 shows an example of the lack of fit analysis of variance printing.

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 response in col **1**, all wts.=**1.0**, for **3** ind. vectors in cols **11**, **12**, and **13**

and

FIT response in col **1**, all wts.=**1.0** for **3** ind. vectors in cols **13**, **12** at **11**

would yield the same results in other portions of the printing, but here the results would, in general, differ. The eight columns in the printing are described under (1) through (8) below.

(1) The column under INDEP VAR 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 partition of the total sum of squares, $\Sigma_1^n w_i y_i^2$, 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 Storage.

(3) 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 167, .20859323=1.0429662/5 and 10.875381=(64.209318+1.0429662)/6. The square root of the residual mean square (.20859323) equals the residual standard deviation (.45672008) shown as the estimates from least squares fit.

(4) The fourth column gives the degrees of freedom (n-r) for the cumulative residual mean squares. These are followed by the degrees of freedom (n-k) for the residual sum of squares and the degrees of freedom for the total sum of squares.

(5) 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 167, 1692.346 = 353.01201/.20859323. The F-values are printed with 3 decimal places.

(6) The significance level of the F-ratio on the same line in (5) is computed 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 .000 on page 167 is the probability that a F-ratio will exceed 307.821 if there is no linear term in the model.

(7) The F-ratio on the rth line is

$$F=\frac{\left(\sum_{i=r}^{k+1} S_i - S_{k+1}\right)/(k-r+1)}{\text{Residual Mean Square}}$$

$$=\frac{\sum_{i=r}^{k} S_i/(k-r+1)}{\text{Residual Mean Square}}$$

with (k-r+1) and (n-k) degrees of freedom. It is used to test the hypothesis that all of the parameters β_i, $\beta_{r+1},\ldots,\beta_k$ equal zero. On page 167

$$1000.084 = .5(353.01201+64.209318)/.20859323$$

with 2 and 3 degrees of freedom.

(8) The significance level is given for the F-ratio on the same line of (7).

Estimates From Least Squares Fit. This is divided into two parts, estimates and accuracy.

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 INDEP VAR. 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 ratios can be used to perform t-tests and construct confidence intervals for the parameters in the model. However, see references in section C6.10 for a discussion of correct procedures.

Next, a column headed ACCURACY* 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 ACCURACY* 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.

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.

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 ACCURACY*.

Since the accurate digits are less precise, additional information is printed to help assess the accuracy of the least squares calculations. At the end, after

* THE NUMBER OF CORRECTLY COMPUTED DIGITS IN EACH COEFFICIENT
USUALLY DIFFERS BY LESS THAN 1 FROM THE NUMBER GIVEN HERE.

is printed, the following information is printed:

THE NUMBER OF ITERATIONS WAS X. SCALING WAS (or WAS NOT) USED.
THE AVERAGE NUMBER OF DIGITS IN AGREEMENT BETWEEN INITIAL SOLUTION AND 1ST ITERATION IS XX.

If the number of iterations is one or two, the results are probably accurate. If the number of iterations is three or four, the accuracy should be questioned. If the number of iterations is greater than four, the accuracy is in doubt. Scaling is used when accurate results are difficult to obtain. A small number of digits in agreement between the coefficients in the initial solution and those of the first iteration is indicative of computational problems.

For the special case, POLYFIT (not FIT) of degree one, a 95% confidence ellipse is printed for the intercept and slope parameters of the unknown straight line. See the example on page 168. See Draper and Smith (1971), chapters 1 and 2.

Brief, Interactive and Width

A BRIEF instruction limits the automatic printout to the following:
(1) diagnostic information for identifying influential measurements,
(2) lack of fit analysis of variance,
(3) analysis of variance,
(4) estimates from least squares fit,
(5) and 95% confidence ellipse for the POLYFIT instruction of degree one.

If the instruction INTERACTIVE or WIDTH is used with an argument less than 120, 72 characters per line will be printed. Values of the independent variable are printed on the right where the weights are normally printed, unless the second argument is a column number or a constant not equal to one. Two pages are used instead of one for the four plots of standardized residuals. The analysis of variance is split into two sections and one is printed below the other.

Storage

There are six different forms of each instruction. They differ with respect to the amount of information stored in the worksheet depending 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 = *C*oefficients
3	6	n=NRMAX	C, R = *R*esiduals
4	7	n=NRMAX	C, R, S = *S*tandard deviations of predicted responses
5	8	k+2	C, R, S, F = *F*ourier coefficients
6	10	kxk	C, R, S, F, V = *V*ariance-covariance matrix

* Add (k−1) for FIT instruction.
** In last named column (or matrix).

Note, (2k+6) and (k+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 (k+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.

In the instruction

POLYFIT 1, 1.0, 3, 2 put in **11, 12, 12** and **13**

the coefficients would be put in column 11, standard deviations of predicted responses in column 12 and the Fourier coefficients in column 13. Since the sixth and seventh argument indicate the same storage column, the residuals are over-written by the predicted responses.

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, also appear on Page Three of the automatic printing.

The squared multiple correlation coefficient is

$$R^2 = 1 - \text{Residual Sum of Squares} / \sum_{i=1}^{n} w_i(y_i - \bar{y}_w)^2, \text{ where } \bar{y}_w = \Sigma w_i y_i / \Sigma w_i.$$

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 of FIT or SFIT is not identically equal to 1, the R has no meaning and the multiple correlation coefficient is not stored.

The Fourier coefficient column contains the k squared Fourier Coefficients, the Residual Sum of Squares and the Total Sum of Squares. The (k+2) values are the same as those under SS=RED. DUE TO COEFF on Page Three of the automatic printing.

In the sixth form, the last two arguments give the row and column location of the number in the upper left-hand corner element of a kxk matrix. The matrix is the (complete) variance-covariance matrix shown on Page Two of the automatic printing.

If (C+k) exceeds the number of columns in the worksheet, the following informative diagnostic is given:

MATRIX EXTENDS BEYOND (n) ROW BY (n) COLUMN WORKSHEET.
ONLY PART OF THE MATRIX IS STORED IN THE WORKSHEET.

If the following instructions

```
FIT COL1 WGHTS=1.0, 2 IND VARS IN COL 2 AND 3 PUT IN 4***7 1,8
RESET NRMAX TO 10
PRINT COLS 4***9
```

replace the last three instructions for the example on page 164, the results stored are:

```
OMNITAB 80 BROWNLEE 11.18 LACK OF FIT ANOVA                                PAGE    4

  COLUMN    4     COLUMN    5     COLUMN    6     COLUMN    7     COLUMN    8     COLUMN    9

-1.1331589         .30942826         .26341465    353.01201         .25008450    -.0061902779
  .23140149        .039428249        .26341465     64.209318    -.0061902779     .00017395402
  .50008449       -.10458669         .18519791      1.0429662
  .013189163      -.72458670         .18519791    418.26430
 7.0000000          .31139833         .18439105

 2.0000000          .47076603         .24530475
 5.0000000         -.30184748         .34006014
  .45672008
  .20859323
  .98401642
```

The information stored in column 4 consists of 2 coefficients and their standard deviations in the first 4 rows. Row 5 of column 4 contains the number of non-zero weights and row 6 the number of independent vectors. The residual degrees of freedom, residual standard deviation, residual variance and multiple correlation coefficient squared are in rows 7 through 10 in column 4. Columns 5 and 6 contain the seven values of the residuals and standard deviation of the predicted responses, respectively. The two values of the Fourier coefficients, the residual sum of squares and total sum of squares are stored in column 7. The variance-covariance matrix is stored in the first two rows of columns 8 and 9.

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 $\psi = A$, where A is the inverse of T'. The matrix product WY produces the Fourier coefficients 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.

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 B1.7.) 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
```

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 10.2 of Draper and Smith (1981). The FIT (or rather SFIT) instruction can be used effectively in each iteration of this method.

FIT y in column **C**, weights **E**, **k** variables in columns **C**, **C** ,..., **C**

Provides automative printing, but no storage. The instruction has (k+3) arguments.

FIT y in **C**, weights **E**, **k** variables in columns **C** ,..., **C**, put coefficients 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.

FIT C weights **E**, **k** variables in **C** ,..., **C** put coefficients 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, s. d. of response values in C

Same as preceding form, but, in addition, the standard deviations of the response 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 coefficients in C

Same as preceding form, but, in addition, the Fourier coefficients are stored in the column designated by the last argument. (k+2) values are stored. This form of the instruction has (k+7) arguments.

FIT C, E, k, C ,..., C, put in C, C, C, C, variance-covarience matrix in R, C

Same as preceding form, but, in addition, the variance-covariance matrix is stored in a kxk matrix starting in row R of column C, designated by the last two arguments. This form of the instruction has (k+9) arguments.

POLYFIT y in column C, weigths = E, of degree d, predictor x in column C

Provides 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 C, weights = E, degree of 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, weights E, degree 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 coefficients in C, residuals in C, s. d. of response values in C

Same as preceding form, but, in addition, the standard deviations of the response 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 coefficients in C

Same as preceding form, but, in addition, the Fourier coefficients are stored in the column designated by the last argument. (d+3) values are stored. This form of the instruction has 8 arguments.

POLYFIT C, E, d, C, put in **C, C, C, C** and variance-covariance matrix in **R, C**

Same as preceding form, but, in addition, the variance-covariance matrix is stored as a $(d+1)x(d+1)$ 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 columns **C** ,..., **C,** put coefficients in **C**

Similar to the second form of FIT instruction except automatic printing is suppressed.

SFIT C weights **E, k** variables in **C** ,..., **C,** put coefficients in **C,** residuals in **C**

Similar to the third form of FIT except automatic printing is suppressed.

SFIT C, E, k, C ,..., **C,** put coefficients in **C,** residuals in **C,** s. d. of response values in **C**

Similar to the fourth form of FIT except automatic printing is suppressed.

SFIT C, E, k, C ,..., **C,** put in **C, C** and **C,** Fourier coefficients in **C**

Similar to the fifth form of FIT except automatic printing is suppressed.

SFIT C, E, k, C ,..., **C** put in **C, C, C, C** variance-covariance matrix in **R, C**

Similar to the last form of FIT except automatic print is suppressed.

SPOLYFIT y in column **C,** weights **E,** degree **d,** x in **C,** put coefficients in **C**

Similar to the second form of POLYFIT except automatic printing is suppressed.

SPOLYFIT y in **C,** weights **E,** deg **d,** x in **C,** put coefficients in **C** residuals in **C**

Similar to the third form of POLYFIT except automatic printing is suppressed.

SPOLYFIT C, E, d, C, put coefficients in **C,** residuals in **C** s. d. of response values in **C**

Similar to the fourth form of POLYfit except automatic printing is suppressed.

SPOLYFIT C, E, *d*, C, put in **C, C, C** and Fourier coefficients in **C**

Similar to the fifth form of POLYFIT except automatic printing is suppressed.

SPOLYFIT C, E, d, C, put in **C, C, C, C** and variance-covariance matrix in **R, C**

Similar to the last form of POLYFIT except automatic print is suppressed.

LARFIT C, weights **1.0, k** vectors **C** ,..., **C,** put coefficients in **C,** residuals in **C**

This instruction has been added for LAR (least absolute residuals) estimation. The argument structure is similar to that of FIT, except, only one form of the instruction exists. The number of arguments must equal **k** plus five. The minimum sum of absolute residuals is stored in the (**k**+ 1)st row of the column designated by the next to last argument. The sample standard deviations of the estimated coefficients are not stored. The second argument must be 1.0 to indicate that equal weights are to be used.

If the instruction

LARFIT of col **1,** wts **1.0, 2** vectors in **2** and **3,** put coeffs in **21,** res. in **22**

is used instead of FIT in the example on page 164, then the numbers -.9062, .2223 and 2.1403 are put in the first three rows of column 21 and the residuals .27, 0.0, -.05252, -.67252, .45496, .69027 and 0.0 are put in the first seven rows of column 22. The value -.9062 is the LAR estimate of the intercept and .2223 is the LAR estimate of the slope of the unknown straight line. The corresponding least squares estimates are 1.1331589 and .23140149. The value 2.1403 is the minimum sum of absolute residuals. The solution is not unique, and the residuals do not sum to zero as is the case with least squares residuals.

Although LAR estimates have advantages over least squares estimates in certain situations, the estimates have two disadvantages. First, LAR solutions are not necessarily unique. If more than one solution exists, different computer programs may easily produce different solutions. Minor modifications in a single algorithm may lead to different solutions. OMNITAB uses a code developed by Barrodale and Roberts. After initial parameter estimates have been obtained, values of the dependent and independent variables with non-zero residuals are used in a reinversion to improve accuracy. When the possibility of multiple solutions is detected, the following informative diagnostic is printed

THE OPTIMAL SOLUTION IS PROBABLY NOT UNIQUE.

Note the use of PROBABLY in the diagnostic and the use of "detected" in the preceding phrase. Nonunique solutions are not necessarily detected. There is no known computer code which will automatically determine all possible solutions. The second disadvantage of LAR estimates is the lack of theoretical knowledge concerning the uncertaintities of the estimates.

A robust or resistive estimation procedure is one which is not very sensitive to departures from the underlying assumptions. Robust regression estimation is receiving considerable attention in the literature. See, for example, Hill and Holland (1977). Many of the robust regression procedures that have been proposed use LAR estimation in some iterative scheme, with or without weights. A set of instructions using LARFIT could easily be written to compute the estimates described by Hill and Holland (1977).

6.5 Selection of Variables in Linear Regression

BESTCP

Frequently an experimenter starts with **k** predictors (independent variables) in a linear regression problem, but wants to use a model with only (p) predictors, where (p) is often much smaller than **k**. The objective is to obtain a parsimonious model which describes the data satisfactorily and is subject to little or no bias.

The problem is often referred to as one of selecting the "best" subset of the **k** predictor variables. It is not a straightforward problem. The meaning of "best" may vary from one problem to another and may not be clear in any one problem. Hocking (1976) gives a comprehensive review of the literature.

Stepwise regression is often used in variable subset selection. The method has a number of disadvantages. It yields a single answer, which may not be the "best." Both forward selection and backward selection procedures can be used. They do not necessarily give the same answer and when they do, the answer is not necessarily the "best" answer. The method normally yields a single answer when several might be preferable or needed.

Recently, a number of approaches have been taken based on computing regressions for all possible subsets. Some of these techniques depend heavily on the use of the statistic:

$$C(p)=\frac{RSS(p)}{\hat{\sigma}^2}+2p-n$$

where

n = number of measurements,
p = number of variables in the subset,
RSS(p) = residual sum of squares using p variables and
$\hat{\sigma}^2$ = an independent estimate of the error variance.

Usually, an independent estimate of $\hat{\sigma}^2$ is not available, and the residual variance from the full model with **k** variables is used for $\hat{\sigma}^2$.

If the model is satisfactory, C(p) will be approximately equal to p, subject to sampling variability.

This section describes the BESTCP instruction which computes the ten smallest C(p) statistics among all possible C(p) statistics with **k** variables in the full model.

BESTCP of y in column **C** weights **E** with **k** variables in columns **C** ,..., **C**

The structure of a BESTCP instruction is the same as the FIT instruction without storage. The instruction uses a very efficient algorithm to compute only as many regressions as are needed in order to produce the ten smallest C(p) statistics. The total number of arguments must equal the third argument plus three. The BESTCP instruction can be used to compute C(p) statistics for models without a constant term.

The number of variables (3rd argument of instruction) must be greater than or equal to 3 and less than or equal to 28. If the number of variables is 3, the instruction only finds the seven smallest C(p) statistics.

Weights can only be equal to 0.0 or 1.0. If the second argument is a column number, the numbers in the column should either be equal to zeros or ones. All other non-zero weights are treated as ones.

For a given number of variables in a regression (subset size) a maximum of ten C(p) statistics are printed.

The results in the automatic printout may be thought of as being in four sections.

First, a two- line title gives the column number (or column heading) for the independent variable (lst argument), the number of predictor variables and the number of measurements with non-zero weights. The predictor variables are numbered consecutively from 1 to **k** if the first predictor variable is not a constant term and from 1 to (**k**-1) if the first predictor variable is a constant term. Immediately below the title, a table is given relating the variables to the column numbers (or column headings). The expression A CONSTANT AND at the end of the first line of title is omitted if the first predictor variable is not a constant term (identically equal to one).

Second, under titles REGRESSION WITH 1 VARIABLE, REGRESSIONS WITH 2 VARIABLES, etc., the ten smallest C(p) statistics are printed. There are three columns under each of these C(p) statistics titled VARIABLE, COEFFICIENT, and F RATIO. The first column gives the specific variables in the regression

and the second column gives the least squares estimates of the regression coefficients. The F-ratios have 1 and (n-p-1) degrees of freedom. The value of the F-ratio is influenced by the order of the variables in the regression.

Third, all additional C(p) statistics, obtained as a byproduct of computing the ten smallest, are printed with titles C(P) STATISTIC and VARIABLES for REGRESSION WITH 1 VARIABLE, REGRESSIONS WITH 2 VARIABLES, etc

Fourth, at the bottom, the number of regressions needed to find the ten smallest C(p) statistics is printed along with the number of operations.

C(p) statistics which are approximately equal to p should be looked for. Remember, p equals the number of variables. Among subsets with comparable C(p) statistics, the subsets with the smallest number of variables are usually the most desirable. In a particular problem, there may be practical considerations that should force a variable to be included in the chosen subset. A good analyst will look for patterns among the low and high C(p) statistics. In the example below it should be noticed that variable 1 does not appear in any of the subsets with C(p) statistics greater than 7.

The subset with the smallest C(p) statistic should not necessarily be selected. The small value may be the result of sampling variability. A plot of C(p) versus p can be very helpful in subset selection.

The following instructions produce C(p) statistics for Hald's data discussed on pages 89 and 90 of Daniel and Wood (1971).

```
READ DATA INTO COLUMNS 11 12 13 14 AND 1
 7  26  6  60   78.5
 1  29 15  52   74.3
11  56  8  20  104.3
11  31  8  47   87.6
 7  52  6  33   95.9
11  55  9  22  109.2
 3  71 17   6  102.7
 1  31 22  44   72.5
 2  54 18  22   93.1
21  47  4  26  115.9
 1  40 23  34   83.8
11  66  9  12  113.3
10  68  8  12  109.4
HEAD COL 14/ PCT. C2S
DEFINE 1.0 IN COLUMN 10
BESTCP of Y IN COL 1 WHTS=1.0, 5 VARIABLES IN 10, 11, 12, 13 AND 14
```

The results are:

```
C(P) STATISTICS FOR COLUMN    1  AS A LINEAR FUNCTION OF A CONSTANT AND
 FROM 1 UP TO  4 VARIABLES WITH  13 MEASUREMENTS WITH NONZERO WEIGHTS

VARIABLE  1 IS COLUMN   11 ,   VARIABLE  2 IS COLUMN   12 ,
VARIABLE  3 IS COLUMN   13 ,   VARIABLE  4 IS   PCT. C2S

   REGRESSION WITH 1 VARIABLE

       C(P) STATISTIC  VARIABLES
              138.731     4
              142.486     2
              202.549     1
              315.154     3
   REGRESSIONS WITH  2 VARIABLES
                C(P) =   2.678
      VARIABLE      COEFFICIENT       F RATIO
          1          1.4683058       146.523
          2           .66225047      208.582
                C(P) =   5.496
      VARIABLE      COEFFICIENT       F RATIO
          1          1.4399583       108.224
          4          -.61395362      159.295
                C(P) =  22.373
      VARIABLE      COEFFICIENT       F RATIO
          3         -1.1998512        40.295
          4          -.72460013      100.357
                C(P) =  62.438
```

```
VARIABLE      COEFFICIENT       F RATIO
    2           .73132958        36.683
    3         -1.0083862         11.816
            C(P) = 138.226
VARIABLE      COEFFICIENT       F RATIO
    2           .31090473          .172
    4         -.45694190           .431

  C(P) STATISTIC  VARIABLES
        198.094     1  3
REGRESSIONS WITH  3 VARIABLES
            C(P) =   3.018
VARIABLE      COEFFICIENT       F RATIO
    1          1.4519380        154.008
    2           .41610947         5.026
    4          -.23654049         1.863
            C(P) =   3.041
VARIABLE      COEFFICIENT       F RATIO
    1          1.6958901         68.716
    2           .65691487       220.547
    3           .25001752         1.832


            C(P) =   3.497
VARIABLE      COEFFICIENT       F RATIO
    1          1.0518541         22.113
    3          -.41004334         4.236
    4          -.64279614       208.240
            C(P) =   7.337
VARIABLE      COEFFICIENT       F RATIO
    2          -.92341541         12.427
    3         -1.4479710          96.940
    4         -1.5570444          41.654
REGRESSIONS WITH  4 VARIABLES
            C(P) =   5.000
VARIABLE      COEFFICIENT       F RATIO
    1          1.5510969          4.337
    2           .51016177          .497
    3           .10190346          .018
    4          -.14406672          .041
  14 REGRESSIONS          56 OPERATIONS
```

The coefficients printed below C(P) = 5.496 are the 2nd and 4th coefficients that would be obtained by using the instruction

FIT OF Y IN COL 1 WHTS=1.0, 3 IND. VAR. IN COLS 10, 11 AND 14

The F ratios are the same as the square of F ratios printed by FIT in the second column from the right in the table of estimates from least square fit given on the third page.

If k is very large, it takes a lot of computing time to compute all possible regressions. The algorithm used by a BESTCP instruction is believed to be the most efficient one available. If the number of variables is ten or less, the computing time should not be very important. However, if the number of variables is large, the computing time could run into minutes depending upon the number of variables, the data and the computer. The number of operations at the end of the printing may be helpful in assessing time requirements.

The BESTCP instruction is an exception to the rule in OMNITAB development that accuracy comes before speed. The results should not be expected to be as accurate as results obtained from an excellent program designed to perform calculations for a single regression. The accuracy of the results is more than adequate for their purpose and have no evidence of numerical instability. As a precaution, a FIT instruction could be used for the subset under final consideration. This should be done anyway, because further analysis, including an examination of the residuals, should be done before the final selection is made.

In the above example, all possible C(p) statistics were printed (excluding the one for the regression with no variables). This will not always be the case, particularly if k is large. If m C(p) statistics are printed in addition to the ten smallest, these m statistics will not necessarily be the remaining C(p) statistics. They are merely the C(p) statistics which were obtained as a by-product in computing the regression necessary to obtain the ten smallest.

6.6 Analysis of Designed Experiments

ONEWAY, SONEWAY, SPLIT PLOT, TWOWAY, STWOWAY

The instructions ONEWAY, SPLIT PLOT and TWOWAY automatically print a comprehensive set of results for analysis of one-way classification; on analysis of variance for split-plot experimental design; and analysis of variance for a two-way crossed classification without replication, fixed effects and infinite model. The instructions SONEWAY and STWOWAY suppress the automatic printing.

ONEWAY analysis for data in column C with group number in column C

The ONEWAY instruction with two arguments provides a comprehensive automatic printing but does not store any of the results. Both arguments are column numbers. The first named column contains the data or measurements. The second named column contains corresponding group numbers (identifications or tags). 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, could be 1, 1, 2, 3, 3 and 3. The group numbers can be any signed values and need not be in increasing order.

If the number of measurements is greater than 2700, or if the number of unique groups is less than 2 or greater than 540, or if the number of groups is greater than the number of measurements with positive tags, the following fatal error message is printed

COLUMN NUMBER(S) OUTSIDE (n) COLUMN WORKSHEET.

The automatic printing consists of a title and six sections: analysis of variance, estimates, pairwise multiple comparison of means, tests for homogeneity of variances, Model II - components of variance, and box plots.

A two-line title gives the number of measurements and column heading on the first line and the number of groups and the column heading for the column containing the group numbers on the second line. If a LABEL or HEAD instruction has been used, its effect is reflected in the column heading used in this two-line title.

Each of the six sections of the automatic printing is described below under the headings which appear on the printed page as shown in the example on pages 186 and 187.

I ANALYSIS OF VARIANCE I

The traditional analysis of variance for a one-way classification 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 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 for

F = Between Groups Mean Square/Within Groups Mean Square

is less than 0.10 and the number of groups exceeds 2, the between groups (means) sum of squares is separated into two components: (i) the slope with 1 degree of freedom and (ii) the balance representing deviations about the straight line regression of group averages on group numbers. 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.

The significance level for the Kruskal-Wallis non-parametric H-test for difference between means (averages) is printed. 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.

I ESTIMATES I

The estimates section consists of the following statistics for each group: the group number, number of measurements (N0.), mean (average), minimum, maximum, sum of the ranks of measurements, within standard deviation, standard deviation of the mean and 95% confidence interval for the mean. 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.

The high (largest) and low (smallest) values of the means (averages), minimums, maximums, within standard deviations and standard deviations of the means are marked with the letters H and L. 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, *** ESTIMATES ARE NOT AVAILABLE BECAUSE SAMPLE SIZE IS 1 *** is printed under WITHIN S.D., S.D. OF MEAN and 95 PCT CONF INT FOR MEAN.

The number of measurements, mean, minimum, and maximum are also printed for the entire data set. 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, the random effects model, and the ungrouped model which assumes that all measurements were taken from a single group.

Let

n = the total number of measurements,
k = the number of groups,
x_i = the individual measurements for $i=1,...,n$,
$\bar{x}_j$ = the group means for $j=1,...,k$ and
$\bar{\bar{x}}$ = the grand mean.

Then the values printed are:

MODEL	WITHIN S.D.	S.D. OF MEAN	95% CONF INT FOR MEAN
FIXED	s_w = (within groups mean square from ANOVA table)	$s_w/\sqrt{n}$	$\bar{\bar{X}} \pm [s_w t_{.975}(n-k)]/\sqrt{n}$
RANDOM	$s_w = \left(\sum_{j=1}^{k} (\bar{X}_j - \bar{\bar{X}})^2/(k-1)\right)^{1/2}$	$s_w/\sqrt{k}$	$\bar{\bar{X}} \pm s_w t_{.975}(k-1)/\sqrt{k}$
UNGROUPED	$s_w = \left(\sum_{i=1}^{k} (x_i - \bar{\bar{x}})^2/(n-1)\right)^{1/2}$	$s_w/\sqrt{n}$	$\bar{\bar{X}} \pm s_w t_{.975}(n-1)/\sqrt{n}$

where $t_{.975}(\nu)$ is the percent point of Student's t distribution with ν degrees of freedom that is exceeded with probability .025.

I PAIRWISE MULTIPLE COMPARISION OF MEANS I

This section only appears if the significance level (value under 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 averages are divided into groups so 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). An approximation for comparing the ith and jth means, having n_i and n_j measurements repectively, use an effective number n where

$$1/n = (1/n_i + 1/n_j)/2$$

The Scheffé method is discussed in section 10.3 of Brownlee (1965). Groups are separated by a string of 3 asterisks. If adjacent groups have no average in common, the two groups are separated by a string of 5 asterisks.

I TEST FOR HOMOGENEITY OF VARIANCES I

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

ASSUMING HETEROGENEOUS VARIANCE, APPROX. BETWEEN MEANS F-TEST SL=(value)

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 187, because both P values (significance levels) exceed 0.10.

I MODEL II—COMPONENTS OF VARIANCE I

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).

I BOX PLOTS I

A box plot is a graphical display of the scrawl of a set of measurements, where the scrawl consists of the five statistics: the minimum, the lower hinge, the median, the upper hinge, and the maximum (sec. C3.8). The lower hinge and the upper hinge are essentially equivalent to the lower and upper quartiles.

An examination of a box plot enables one to get a quick assessment of the shape of the distribution of the numbers; including an assessment of location, variation and skewness. By examining all of the box plots simultaneously, the relative shapes of the distributions for the different groups can be assessed. There are several variations of box plots which are described in McGill et al. (1977). The simplest form is given by ONEWAY.

A box plot has a line starting with an asterisk to represent the minimum and ending with an asterisk to represent the maximum. In between, H's are appropriately placed to represent the hinges and M is appropriately placed inside to represent the median. A box is erected above and below the line from the lower hinge to the upper hinge.

In small samples, a hinge, the median, or an extreme value may coincide. In such cases, only one symbol is printed. The M, for median, takes precedence over an H, for hinge, or an *, for an extreme value, and an * takes precedence over an H. Obviously, at least five measurements are needed to print a box plot. If a group has less than five measurements, an X is used to represent each measurement in the group. If two or more points coincide, a single X is printed.

The box plot section starts on a new page. A title and axis are printed at the top of the page and on any subsequent pages. Three numbers are printed above the 0's of the axis to show the scale; namely, the minimum, the midrange and the maximum of all the measurements. The group number and the number of measurements in the group are printed on the left.

The following set of instructions of the data on page 315 of Brownlee (1965)

```
      OMNITAB 80 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 numbers 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 group nos. in col 2
```

produces the following automatic output:

```
OMNITAB 80 EXAMPLE OF ONEWAY - BROWNLEE DATA PAGE 251

      ONEWAY ANALYSIS OF   16 MEASUREMENTS IN COLUMN    1
   CLASSIFIED INTO    3 GROUPS WITH NUMBERS IN COLUMN    2

                    I  ANALYSIS OF VARIANCE  I

SOURCE          D.F.   SUMS OF SQUARES    MEAN SQUARES   F RATIO  PROB.

BETWEEN GROUPS    2        565.10413        282.55206   26.082    .000
  SLOPE             1         64.508909       64.508909    1.408  .257
  ABOUT LINE        1        500.59522       500.59522    46.209  .000
WITHIN GROUPS    13        140.83333         10.833333
TOTAL            15        705.93750
        KRUSKAL-WALLIS RANK TEST FOR DIFFERENCE BETWEEN MEANS
              SIGNIFICANCE LEVEL IS APPROXIMATELY  .000

                    I  ESTIMATES  I

  GROUP    NO.       MEAN          MINIMUM          MAXIMUM      SUM RANKS

    1        6    78.166666  H    72.000000  H    83.000000  H     76.0H
    2        5    64.000000  L    61.000000  L    67.000000  L     15.0L
    3        5    74.000000       71.000000       78.000000        45.0
 -------   ----  ------------    ------------    ------------    --------
  TOTAL     16    72.437500       61.000000       83.000000

  GROUP       WITHIN S.D.    S.D. OF MEAN     95 PCT CONF INT FOR MEAN

    1         3.8686776  H    1.5793810  H     74.106800 TO  82.226532
    2         3.0000000       1.3416408       60.275069 TO  67.724931
    3         2.7386128  L    1.2247449  L     70.599619 TO  77.400381
MODEL        ----------      ----------      ------------------------
 FIXED       3.2914029         .82285073      70.659840 TO  74.215160
 RANDOM      7.2807455        4.2035403       53.968983 TO  90.142128
 UNGROUPED   6.8602114        1.7150528       68.781952 TO  76.093048
```

I PAIRWISE MULTIPLE COMPARISION OF MEANS I

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 THE NUMBERS OF MEASUREMENTS IN THE GROUPS DIFFER)

64.000000
*****
74.000000, 78.166666

- SCHEFFE TECHNIQUE -

64.000000
*****
74.000000, 78.166666

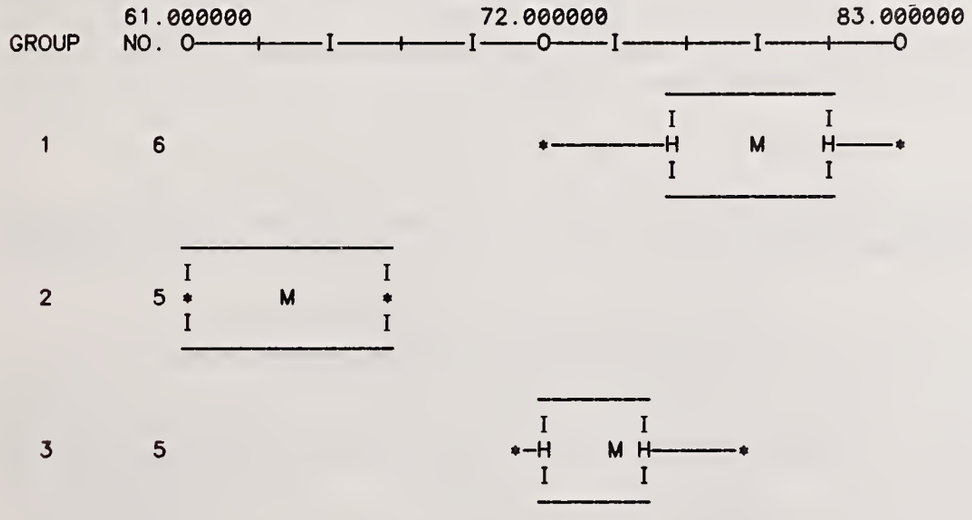
I TESTS FOR HOMOGENEITY OF VARIANCES I

COCHRAN'S C = MAX VARIANCE/SUM(VARIANCES) = .4756, APPROX SL = .439
BARTLETT-BOX F = .269, SIGNIFICANCE LEVEL = .764
MAXIMUM VARIANCE / MINIMUM VARIANCE = 1.9955555

I MODEL II - COMPONENTS OF VARIANCE I

ESTIMATE OF BETWEEN COMPONENT OF VARIATION IS 51.147055

I BOX PLOTS I



ONEWAY for **C** group numbers in **C**, put statistics in **C** and next three columns

This form of ONEWAY, with an additional argument, provides 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 numbers,
X+1	number of measurements in each group,
X+2	group means (averages) and
X+3	group standard deviations.

NRMAX is unchanged.

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 numbers in **C**, number in **C**, means **C**, s.d. in **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 consecutive.

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 number of measurements in each group in column 11 and the standard deviations in column 12. The group number and means would not be stored.

SONEWAY for **C** group numbers in **C**, put statistics in **C** and next three columns

This instruction is similar to the second form of ONEWAY, except automatic printing is omitted.

SONEWAY for **C** with **C**, put group numbers in **C**, number in **C**, means in **C**, s.d. in **C**

No automatic printing is produced. Otherwise, instruction is the same as the last form of ONEWAY.

SPLIT PLOT analysis of **C**, replicate nos **C**, whole plot nos **C**, split plot nos **C**

The instruction has a two-word command (without a hyphen). An analysis of variance is automatically printed on one page. The four arguments must be column numbers.

The experimental response measurements are in the column designated by the first argument. The replicate numbers must be 1, 2, ..., r=number of replicates. An analysis is not possible unless r is at least two. Similarly, the whole plot treatments are 1, 2, ..., w=the number of whole plots, and the split-plot treatments are 1, 2, ..., s=the number of split-plots. The product r*w*s must equal NRMAX.

Although not common, split-plot experimental designs occur more frequently than is realized. To quote Hicks (1964), "There are still many practical situations in which it is not at all feasible to even randomize within a block. Under certain conditions these restrictions will lead to a split-plot design." The analysis of split-plot experimental design data poses additional problems because there are two error terms for tests of significance. The design and analysis of split-plot experiments is discussed on pages 190 to 200 of Hicks (1964).

John Barnes, NBS, designed a split-plot experiment to calibrate three thermocouples. Three thermocouples were placed in an oven in three positions. Thermocouples were split-plot treatments and positions were whole plot treatments. The experiment was replicated twice. The set of OMNITAB instructions, data and the one page of automatic printing follows. (The word thermocouples is spelled thermocples because the number of characters in a label must not exceed 12.)

```
OMNITAB 80 ANALYSIS OF J. BARNES OVEN DATA
$
$     RESPONSE IS PERCENT OF FULL SCALE WHICH IS
$        PROPORTIONAL TO MILLIVOLTS.
$
LABEL MILLIVOLTS, REPLICATE, POSITIONS, THERMOCPLES
SET MILLIVOLTS DATA
   77.9   77.0   73.3   79.6   79.2   79.5   88.5   82.1   83.0
   88.1   89.0   86.3   73.2   72.3   72.9   70.6   71.6   67.5
SET REPLICATE NUMBER
   1   2   1   2   1   2   1   2   1   2   1   2   1   2   1   2   1   2
SET WHOLE PLOT TREATMENT NUMBER EQUAL TO POSITIONS
   1   1   1   1   1   1   2   2   2   2   2   2   3   3   3   3   3   3
SET SPLITPLOT TREATEMENT NUMBER EQUAL TO THERMOCPLES
   1   1   2   2   3   3   1   1   2   2   3   3   1   1   2   2   3   3
SPLIT PLOT ANALYSIS OF MILLIVOLTS REPLICATE POSITIONS THERMOCPLES
STOP
```

```
OMNITAB 80 ANALYSIS OF J. BARNES OVEN DATA                    PAGE    1

        ANALYSIS OF VARIANCE FOR SPLIT-PLOT EXPERIMENTAL DESIGN
 MODEL IS Y(IJK) = MU + R(I) + W(J) + ETA(IJ) + S(K) +WS(JK) + E(IJK),
   WHERE ETA AND E ARE RANDOM AND MU, W, S, AND WS ARE FIXED EFFECTS.

                  18 MEASUREMENTS (Y) IN  MILLIVOLTS
                   2 REPLICATES   (R) IN   REPLICATE
                   3 WHOLE-PLOTS  (W) IN   POSITIONS
                   3 SPLIT-PLOTS  (S) IN  THERMOCPLES
SOURCE             D.F.  SUM OF SQUARES    MEAN SQUARES  F-RATIO   P(F)
-------------------------------------------------------------------------
(R)   REPLICATE       1       1.7422231       1.7422231
(W)   POSITIONS       2     662.66779       331.33389        .005   .995
R X W INTERACTION     2      15.221115        7.6105574
(S)  THERMOCPLES      2       2.6677791       1.3338895      .154   .860
W X S INTERACTION     4      23.395555        5.8488889      .676   .633
RESIDUAL              6      51.916665        8.6527774
-------------------------------------------------------------------------
TOTAL                17     757.61112
```

TWOWAY analysis for **r** by **c** table, data in column **C**, put in **C** and successive columns

The instruction is used for the balanced case to produce an automatic printing of an analysis of variance for two-way crossed classification without replication, fixed effects and infinite model. For the unbalanced case use the optional form of TWOWAY discussed below. The instruction has four arguments, all integers. The first two arguments, r and c, specifiy the number of rows and columns in the two-way table. The third argument is the column containing the measurements. Data are entered into this column 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. The fourth argument designates the column where storage begins. See below.

For example, suppose a two-way table with three rows and four columns, for measurements y_{ij} (i=1,2,3; j=1,2,3,4) is given as follows:

	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:

$$y_{ij}=\mu+\rho_i+\gamma_j+\epsilon_{ij},$$

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. The parameter μ represents the grand mean, ρ_i represents the effect of the ith row, and the parameter γ_j represents the effect of the jth column.

The constraints

$$\sum_{i=1}^{r}\rho_i=\sum_{j=1}^{c}\gamma_j=0$$

reduce the paramters to be estimated to μ, ρ_1, ρ_2, γ_1, γ_2 and γ_3 because

$$\rho_3=-\rho_1-\rho_2 \text{ and } \gamma_4=-\gamma_1-\gamma_2-\gamma_3.$$

The model can be conveniently expressed in matrix notation as $Y=X\beta+e$, where

MATRIX NOTATION

$$\begin{bmatrix} y_{11}\\ y_{12}\\ y_{13}\\ y_{14}\\ y_{21}\\ y_{22}\\ y_{23}\\ y_{24}\\ y_{31}\\ y_{32}\\ y_{33}\\ y_{34}\end{bmatrix}=\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0\\ 1 & 1 & 0 & 0 & 1 & 0\\ 1 & 1 & 0 & 0 & 0 & 1\\ 1 & 1 & 0 & -1 & -1 & -1\\ 1 & 0 & 1 & 1 & 0 & 0\\ 1 & 0 & 1 & 0 & 1 & 0\\ 1 & 0 & 1 & 0 & 0 & 1\\ 1 & 0 & 1 & -1 & -1 & -1\\ 1 & -1 & -1 & 1 & 0 & 0\\ 1 & -1 & -1 & 0 & 1 & 0\\ 1 & -1 & -1 & 0 & 0 & 1\\ 1 & -1 & -1 & -1 & -1 & -1\end{bmatrix}\begin{bmatrix}\mu\\ \rho_1\\ \rho_2\\ \gamma_1\\ \gamma_2\\ \gamma_3\end{bmatrix}+\begin{bmatrix} e_{11}\\ e_{12}\\ e_{13}\\ e_{14}\\ e_{21}\\ e_{22}\\ e_{23}\\ e_{24}\\ e_{31}\\ e_{32}\\ e_{33}\\ e_{34}\end{bmatrix}$$

The ijth row of design matrix X consists of signed ones and zeros indicating how the parameters of interest appears in the model for the corresponding measurement y_{ij}. See pages 450 and 451 of Draper and Smith (1981). Note, the design matrix, X, is constructed with the constraints on row and column effects imposed.

NRMAX must be set equal to the product **rc**, otherwise the following fatal error is printed:

ROW NUMBER(S) OUTSIDE (n) ROW WORKSHEET.

Both **r** and **c** must be greater than 1 and the third argument number must be less than the fourth argument number, otherwise the following fatal error is printed:

INCORRECT ARGUMENT IN INSTRUCTION.

If $2(r+c+4)$ is greater than the number of rows in worksheet or the sum of the last argument number plus **r** plus **c** plus 2 is greater than the number of columns in the worksheet, the fatal error printed is:

ARRAY OR MATRIX OUTSIDE (n) ROW X (n) COLUMN WORKSHEET.

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*	*s*	*t*
2	67	6	38*	10	27*	14	21*
3	56	7	35*	11	25*	15	19*
4	48	8	32*	12	24*	16	18*
5	43*	9	29*	13	22*	17	17*

The asterisk indicates that the worksheet must be redimensioned for the largest tables.

If the above restriction is violated, the fatal error given is:

INSUFFICIENT SCRATCH AREA.

The automatic printing consists of a title, indicating the size of the table and the column containing the data, and five sections. The first three sections are separated by a line of dashes.

The first section contains the source, degrees of freedom, sums of squares, mean squares, F-ratio and F PROB. 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.

An analysis of variance to perform Tukey's test for non-additivity is in the second section. 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. Because 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 levels, 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.

The third section consists of a table containing estimates of the grand mean μ, row effects ρ_i for $i=1,...,r$, column effects γ_j for $j=1,...,c$ and rank sums. Row and column means are estimated by $\mu+\rho_i$ and $\mu+\gamma_j$, respectively. Nonparametric rank sum statistics for a twoway table are discussed in Chapter 13 of Gibbons (1971) and section 5.7 of Conover (1971). The rank sum statistics are properly adjusted for ties. Following the procedure in ONEWAY, the symbols H for high and L for low are used to designate the largest and smallest effects and rank sums.

The fourth section is divided into two parts. The first part, on the left, gives the sample standard deviations of the grand mean, row effects, column effects followed by the residual standard deviation. The second part, on the right, gives values of the Kendall's W and Friedman's χ^2 rank sum test statistics for both rows and columns.

The last section contains a table of standardized residuals; standardized by dividing each residual by its own standard deviation. See section C6.4 for further information on standardized residuals. Each residual is printed with two digits after the decimal point. If the number of columns, **c**, exceeds 8, 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.

The instruction stores results in (**r**+**c**+3) consecutive columns, starting with column number (X) of last argument 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) Sum of squares in (**r**+**c**+1) rows of column (X+**r**+**c**+2).

NRMAX may have to be reset in order to print the entire contents of the coefficients and sum of squares columns, since 2(**r**+**c**+4) may exceed NRMAX.

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 (2**r**+**c**+2)	standard deviations of the **r** row estimates	**r**
(2**r**+**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:	6
	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.	

In the sum of squares 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.

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.

A least squares approach is taken and the computing algorithm is essentially the same as that used in the FIT instruction (sec. C6.4). In a restricted sense, TWOWAY is special case of FIT. A balanced two-way table is a special case of an unbalanced table (optional TWOWAY) 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:

THE OPTIMAL SOLUTION IS PROBABLY NOT UNIQUE.

For the following set of instructions to analyze the 3x4 table on page 331 of Graybill (1961),

```
SET DATA IN COLUM 1 ROW BY ROW
     8     2     1     3
     4     0     3     1
     2     1     0     4
TWOWAY ANALYSIS FOR 3X4 TABLE IN COLUMN 1, START STORING IN COLUMN 11
RESET NRMAX = 22
PRINT 11***20
```

the automatic printing is as follows:

ANALYSIS OF VARIANCE FOR TWOWAY 3 X 4 TABLE OF COLUMN 1

SOURCE	D.F.	SUMS OF SQUARES	MEAN SQUARES	F RATIO	F PROB.
BETWEEN ROWS	2	7.1666666	3.5833333	.942	.528
BETWEEN COLS	3	24.916667	8.3055555	2.182	.191
RESIDUALS	6	22.833333	3.8055556		
TOTAL	11	54.916667			
		TUKEY'S TEST FOR NON-ADDITIVITY			
NON-ADDITIVITY	1	7.9069896	7.9069896	2.649	.165
BALANCE	5	14.926344	2.9852688		
RESIDUALS	6	22.833333	3.8055556		

COEFFICIENT		ESTIMATE	RANK SUM
GRAND MEAN		2.4166667	
ROW	1	1.0833333 H	10.0H
ROW	2	-.41666666	7.0L
ROW	3	-.66666666L	7.0L
COLUMN	1	2.2500000 H	11.0H
COLUMN	2	-1.4166667 L	5.0L
COLUMN	3	-1.0833333	5.0L
COLUMN	4	.25000000	9.0

SAMPLE STANDARD DEVIATIONS ...

GRAND MEAN =	.56314263
ROW EFFECT =	.79640396
COL EFFECT =	.97539166
RESIDUAL =	1.9507833

RANK SUM TEST STATISTICS ...

ROWS - KENDALL W	=	.188
FRIEDMAN CHISQ	=	1.500
COLS - KENDALL W	=	.600
FRIEDMAN CHISQ	=	5.400

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

COLUMN 11	COLUMN 12	COLUMN 13	COLUMN 14	COLUMN 15	COLUMN 16	COLUMN 17	COLUMN 18
1.0000000	1.0000000	0.	1.0000000	0.	0.	2.4166667	2.2500000
1.0000000	1.0000000	0.	0.	1.0000000	0.	1.0833333	-.083333333
1.0000000	1.0000000	0.	0.	0.	1.0000000	-.41666666	-1.4166667
1.0000000	1.0000000	0.	-1.0000000	-1.0000000	-1.0000000	-.66666666	-.75000000
1.0000000	0.	1.0000000	1.0000000	0.	0.	2.2500000	-.25000000
1.0000000	0.	1.0000000	0.	1.0000000	0.	-1.4166667	-.58333334
1.0000000	0.	1.0000000	0.	0.	1.0000000	-1.0833333	2.0833333
1.0000000	0.	1.0000000	-1.0000000	-1.0000000	-1.0000000	.25000000	-1.2500000
1.0000000	-1.0000000	-1.0000000	1.0000000	0.	0.	.56314263	-2.0000000
1.0000000	-1.0000000	-1.0000000	0.	1.0000000	0.	.79640396	.66666667
1.0000000	-1.0000000	-1.0000000	0.	0.	1.0000000	.79640396	-.66666666
1.0000000	-1.0000000	-1.0000000	-1.0000000	-1.0000000	-1.0000000	.79640396	2.0000000
						.97539166	
						.97539166	
						.97539166	
						.97539166	
						12.000000	
						6.0000000	
						6.0000000	
						1.9507833	
						3.8055556	
						.58421852	

COLUMN 19	COLUMN 20
1.3794121	70.083332
1.3794121	6.1250000
1.3794121	1.0416666
1.3794121	6.0000000
1.3794121	14.222222
1.3794121	4.6944445
1.3794121	22.833333
1.3794121	125.00000
1.3794121	
1.3794121	
1.3794121	
1.3794121	

The design matrix is stored in the $6=\mathbf{r}+\mathbf{c}-1$ columns 11 through 16 ; the coefficients are in column 17; the residuals in column 18; the standard deviations of predicted values are in column 19; and the sum of squares are in column 20. Columns 11 through 16 contain the same design matrix as the matrix X in the MATRIX NOTATION table on page 190. Column 17 has values in 22 $(2(\mathbf{r}+\mathbf{c})+8)$ rows and column 20 has values in 8 $(r+c+1)$ rows, whereas NRMAX=12.

The grand mean, row estimates and column estimates are stored in rows 1 through 8 of column 17. The corresponding sample standard deviations are stored in rows 9 through 16 of column 17. Rows 17 through 22 of column 17 are the number of non-zero weights, number of vectors in the design matrix, residual degrees of freedom, residual standard deviation, residual variance and squared multiple correlation coefficient.

The SUMS OF SQUARES in the ANOVA table can be obtained from the values in column 20. The Between Rows SS (7.1666666) is the sum of the values in rows (2 to $\mathbf{r}-1$) or 2 and 3. The Between Cols SS is the sum of the values in rows ($\mathbf{r}$ to $\mathbf{r}+\mathbf{c}-1$) or 4, 5 and 6. The Residual SS is in row 7 $(\mathbf{r}+\mathbf{c})$. The corrected Total SS equals the uncorrected Total SS in row 8 $(\mathbf{r}+\mathbf{c}+1)$ minus the SS due to the mean in row 1.

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**

If the instruction

FIT 1, 1.0, 6, 11 * 16**

is added after the TWOWAY instruction, it 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) the predicted values, and (iii) the variance-covariance matrix, which could be used to calculate the variance of linear combinations of the row or column estimates.

TWOWAY anal. for **r** by **c** table, data in **C**, store from **C** on, weights in column **C**

This form of TWOWAY has an additional argument to specify a column of weights and is used in the unbalanced case. 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 to each missing measurement. If each cell in a table contains more than one measurement, TWOWAY can be used on the cell averages using the number of measurements in each cell as weights. Weights w_{ij} are entered one row below another in the same order as their corresponding measurements y_{ij}.

There may exist patterns of weights which cause difficulty for which 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 *ACCURACY* column can be checked in the automatic printing after using the appropriate FIT instruction. An example is provided by B. L. Joiner and J. R. Rosenblatt where weights for the 3x4 table of the last example causes problems with the estimation procedure. The weights w ($i=1,...,r$; $j=1,...,c$) corresponding to the measurements y_{ij} are:

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

$$\begin{bmatrix} 7 & -2 & 0 & -1 & 0 & 0 \\ -2 & 4 & 3 & 2 & 0 & 0 \\ 0 & 3 & 6 & 0 & 0 & 0 \\ -1 & 2 & 0 & 3 & 2 & 2 \\ 0 & 0 & 0 & 2 & 2 & 4 \end{bmatrix}$$

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 are not estimable separately. (The rank of X'WX is 5 because column (6) is the linear combination of columns (1) through (5): $3(6)=-(1)-8(2)+4(3)+9(4)-3(5)$.) If MINVERT were used to invert the (X'WX) matrix the error bound would be $-3+10$, indicating trouble.

The automatic printing consists of three sections. The first section 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, two lines are printed indicating the number of non-zero weights and the number of weights equal to zero.

The second section 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. 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 third section is similar to the last section of the first form of the TWOWAY instruction, except .00 appears in each cell for which the weight is zero.

If the last column number of the instruction is greater or equal to the column number of the fourth argument and less than or equal to the column number of the fourth argument plus **r** plus **c** plus two, the following fatal error message is printed:

STORAGE COLUMN NUMBERS CANNOT EQUAL OTHER COLUMN NUMBERS.

Weights must not be negative, otherwise the fatal error message given is:

NEGATIVE WEIGHTS MAY NOT BE USED.

If the number of weights equal to zero is equal to or greater than (**r**−1)*(**c**−1), the following fatal error is printed:

DEGREE IS GREATER THAN NUMBER OF NON-ZERO WEIGHTS.

A TWOWAY instruction can be used to analyze data from a row by column classification with one or no measurements in each cell. If there is more than one measurement in any cell, a TWOWAY instruction cannot be used directly. However, it can be used indirectly, as is explained below.

The analysis of variance table for a two-way classification with replication is:

Source	*d.f.*	*Sums of Squares*	*Mean Squares*	*F Ratio*	*Significance Level*
Rows	DR	SR	MR	FR	PR
Columns	DC	SC	MC	FC	PC
Row x Column Interaction	DI	SI	MI	FI	PI
Residual	DE	SE	ME		
Total	DT	ST			

The above table can be easily constructed by hand after using a set of OMNITAB instructions to obtain the necessary sums of squares and mean squares.

The degrees of freedom are calculated as follows:

DR = **r**−1, where **r** = number of rows
DC = **c**−1, where **c** = number of columns
DI = (**r**−1)(**c**−1)

$$DE = \sum_{i=1}^{rc}(n_i-1),$$ where n_i is the number of measurements in the ith cell and i=1, 2, ..., **rc**

$$DT = \sum_{i=1}^{rc}(n_i)-1$$

The sums of squares can be calculated in a number of ways. The following procedure is suggested. Enter the measurements into the worksheet using a SET instruction. Using another SET instruction, enter group numbers for a ONEWAY analysis. Set the group number equal to 1 for the measurements in the first row of the first column of the **rxc** table, 2 for the measurements in the first row of the second column, etc., until numbers for the first row have been assigned. Continue, using consecutive integers, for the second row, third row, etc. Do a ONEWAY analysis using the form of ONEWAY with six arguments for storing the group

number, number of measurements in each group, the means and the standard deviations. Then do a TWOWAY analysis using the means from ONEWAY as measurements and the numbers in the groups as the weights.

The sums of squares for the analysis of variance are obtained as follows:

SR = row sum of squares from TWOWAY
SC = column sum of squares from TWOWAY
SI = residual sum of squares from TWOWAY
SE = within sum of squares from ONEWAY
ST = total sum of squares from ONEWAY

The row and column sums of squares from TWOWAY should be chosen from the line showing a value for F PROB. Note that SR+SC+SI+SE will not equal ST, unless there is an equal number of measurements in each cell or the cell frequencies are proportional.

The mean squares are obtained, as usual, by dividing the sums of squares by the corresponding degrees of freedom. For a fixed effects model, the F-ratios FR, FC and FI are obtained by dividing MR, MC and MI by ME.

An F-test should be done to assess whether there is a significant row by column interaction. If there is a significant interaction, it makes little sense to test for row or column effects. Instead, a graphical study of the means should be done to determine the reason for the row by column interaction. A significant interaction may be caused by an outlier or an inappropriate scale for the response measurements. If an outlier is present, an attempt should be made to determine the cause. A reanalysis without using the outlier may be appropriate. If an outlier is not present, a transformation (such as log or square root) of the response variable may remove the interaction. Probability plots of residuals may be helpful.

The BETWEEN GROUPS sum of squares from the ONEWAY analysis should equal the TOTAL sum of squares from the TWOWAY analysis.

The following data has been drawn from a medical thermometry experiment. The data consisted of measurements of bead thermistor drift rates for five manufacturers and three resistances, low, medium and high. A set of OMNITAB instructions to analyze the data and the results follow.

```
SET DRIFT RATE IN COLUMN 1
 -.00146271   .02112775   .00282039 -.02015703 -.00576177   .00087166   .00195146
  .00486606   .00023100 -.03231597 -.05563992   .01034201 -.00423480 -.00661010
 -.01153926   .00618470   .00389811   .01307638   .00128444   .00779145   .01144474
  .00788975 -.02578416   .00356302   .01287440 -.05037265 -.00086367 -.00673810
  .00921573   .00604309   .00219059   .01001163   .01505623   .00875096   .00931758
 -.00713043   .02330194 -.00255854   .02775204   .01250641   .00207295 -.00672891
  .00517192 -.00761442   .03754005   .00952288   .00981737   .00480323   .02078291
  .01174765 -.22716029 -.01801598   .01118366   .00146129   .02934898   .01127789
 -.05363813   .00084837
SET ROW-COLUMN CELL NUMBER IN COLUMN 2
   1   1   1   1   1   2   2   2   2   3   3   4   4   4
   4   4   5   5   5   5   5   6   6   6   6   6   7   7
   7   7   8   8   8   8   8   9   9   9   9   9  10  10
  10  10  10  11  11  11  11  11  12  12  12  15  15  15  15  15
ONEWAY ANALYSIS OF COLUMN 1 WITH IDENTIFICATION IN COLUMN 2 STORE 11 12 13 14
DEFINE 13,12 INTO 15,12
DEFINE 13,13 INTO 15,13
ADEFINE 13,12 SIZE 2X2 EQUAL TO 0.0
RESET 15
TWOWAY ANALYSIS FOR 5X3 TABLE OF MEANS IN COL 13 STORE 21 WEIGHTS 12
```

ONEWAY ANALYSIS OF 58 MEASUREMENTS IN COLUMN 1
CLASSIFIED INTO 13 GROUPS WITH NUMBERS IN COLUMN 2

I ANALYSIS OF VARIANCE I

SOURCE	D.F.	SUMS OF SQUARES	MEAN SQUARES	F RATIO	PROB.
BETWEEN GROUPS	12	.024407991	.0020339992	2.041	.042
SLOPE	1	.00022141700	.00022141700	.180	.674
ABOUT LINE	11	.024186574	.0021987795	2.207	.027
WITHIN GROUPS	45	.044841114	.00099646919		
TOTAL	57	.069249106			

KRUSKAL-WALLIS RANK TEST FOR DIFFERENCE BETWEEN MEANS
SIGNIFICANCE LEVEL IS APPROXIMATELY .205

I ESTIMATES I

GROUP	NO.	MEAN	MINIMUM	MAXIMUM	SUM RANKS
1	5	-.00068667396	-.020157030	.021127750	122.0
2	4	.0019800450	.00023100000	.0048660600	99.0
3	2	-.043977945	-.055639920	-.032315970 L	7.0L
4	5	-.0011714900	-.011539260	.010342010	118.0
5	5	.0074990239	.0012844400	.013076380	187.0
6	5	-.010365928	-.050372650	.012874400	126.0
7	4	.0019142625	-.0067381000	.0092157299	104.0
8	5	.0090653979	.0021905900	.015056230	200.0
9	5	.010774284	-.0071304300	.027752040	188.0
10	5	.0060883179	-.0076144200	.037540050 H	140.0
11	5	.011334808 H	.0048032300 H	.020782910	215.0H
12	3	-.077997536 L	-.22716029 L	.011183660	54.0
15	5	-.0021403202	-.053638130	.029348980	151.0
TOTAL	58	-.0026617961	-.22716029	.037540050	

GROUP	WITHIN S.D.	S.D. OF MEAN	95 PCT CONF INT FOR MEAN
1	.014946757	.0066843929	-.019245222 TO .017871874
2	.0020508190 L	.0010254095 L	-.0012832181 TO .0052433081
3	.016492523	.011661975	-.19215732 TO .10420144
4	.0091258864	.0040812204	-.012502590 TO .010159610
5	.0049585463	.0022175293	.0013422756 TO .013655772
6	.026952077	.012053335	-.043830808 TO .023098952
7	.0071400029	.0035700015	-.0094469101 TO .013275435
8	.0045883242	.0020519609	.0033683336 TO .014762462
9	.015382860	.0068794243	-.0083257495 TO .029874317
10	.018427023	.0082408153	-.016791482 TO .028968117
11	.0058669472	.0026237785	.0040501492 TO .018619466
12	.13000115 H	.075056201 H	-.40094185 TO .24494678
15	.031006260	.013866421	-.040639051 TO .036358411
MODEL			
FIXED	.031566900	.0041449370	-.011010128 TO .0056865356
RANDOM	.025767820	.0071467073	-.022316240 TO .0088264316
UNGROUPED	.034855367	.0045767339	-.011826547 TO .0065029552

I PAIRWISE MULTIPLE COMPARISION OF MEANS I

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 THE NUMBERS OF MEASUREMENTS IN THE GROUPS DIFFER)

```
-.077997536, -.043977945
    ***
-.043977945, -.010365928, -.0021403202, -.0011714900, -.00068667396,
 .0019142625,  .0019800450,  .0060883179,  .0074990239,  .0090653979,
 .010774284,  .011334808
                    -   SCHEFFE TECHNIQUE   -

-.077997536, -.043977945, -.010365928, -.0021403202, -.0011714900,
-.00068667396,  .0019142625,  .0019800450,  .0060883179,  .0074990239,
 .0090653979,  .010774284,  .011334808
```

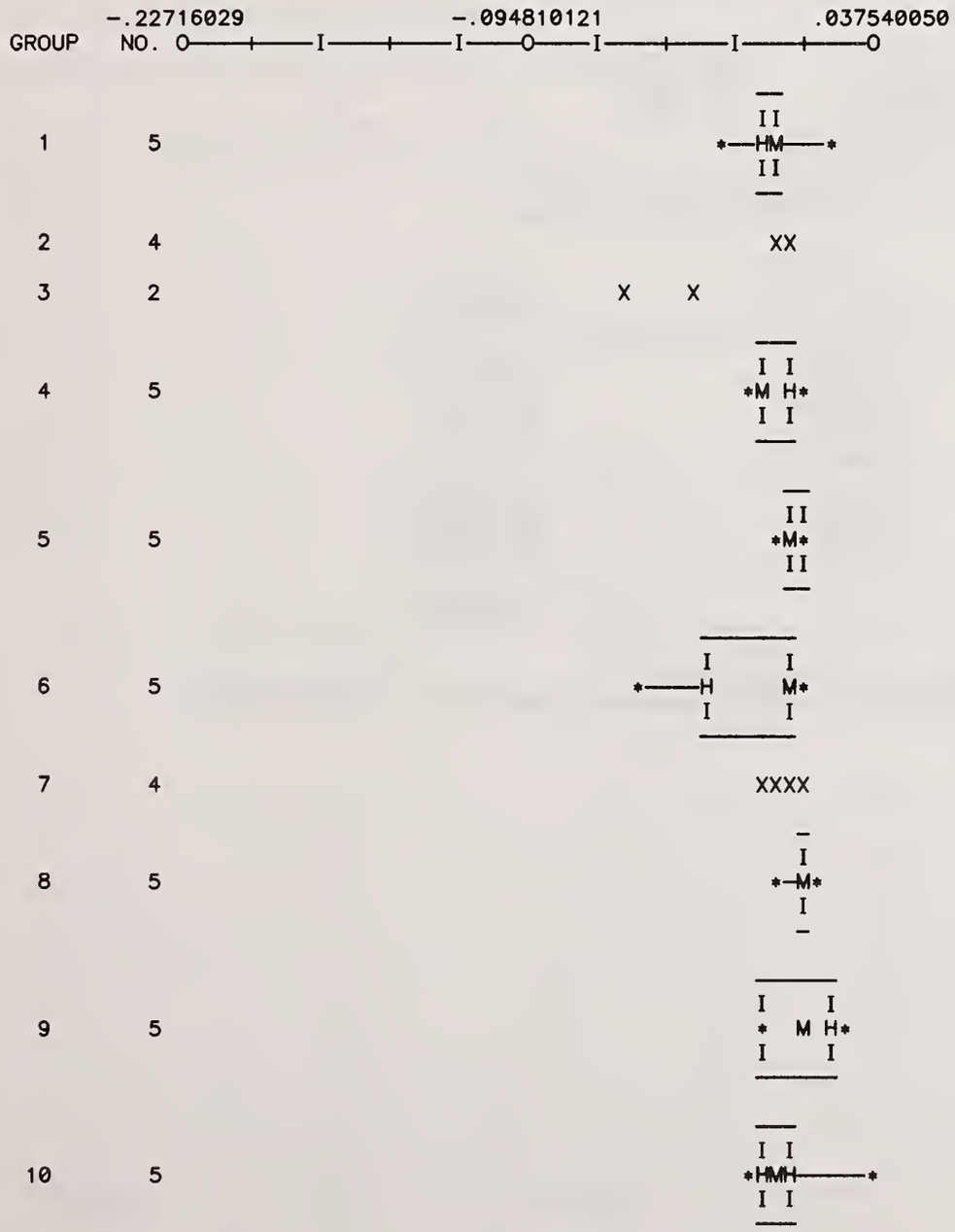
I TESTS FOR HOMOGENEITY OF VARIANCES I

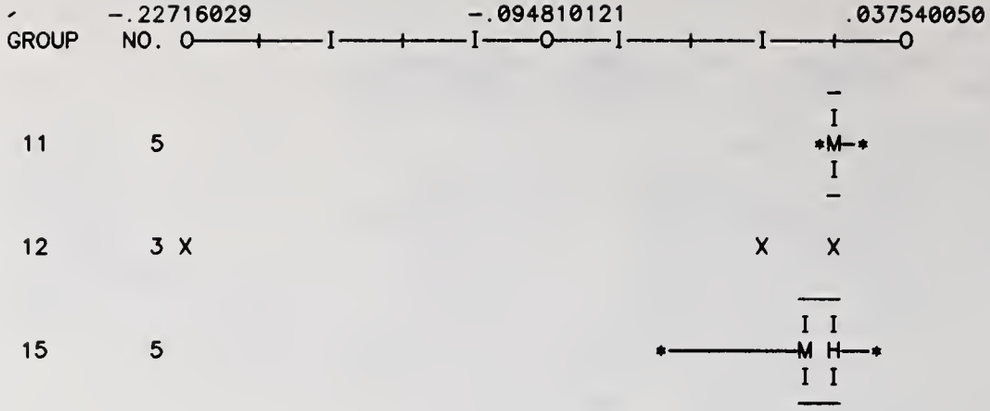
COCHRAN'S C = MAX VARIANCE/SUM(VARIANCES) = .8502, APPROX SL = .000
BARTLETT-BOX F = 7.386, SIGNIFICANCE LEVEL = .000
MAXIMUM VARIANCE / MINIMUM VARIANCE = 4018.2760
ASSUMING HETROGENEOUS VARIANCE, APPROX. BETWEEN MEANS F-TEST SL = .061

I MODEL II - COMPONENTS OF VARIANCE I

ESTIMATE OF BETWEEN COMPONENT OF VARIATION IS .00023339397

I BOX PLOTS I





ANALYSIS OF VARIANCE FOR TWOWAY 5 X 3 TABLE OF COLUMN 13

SOURCE	D.F.	SUMS OF SQUARES	MEAN SQUARES	F RATIO	F PROB.
BETWEEN ROWS	4	.0054699488	.0013674872	.646	.650
BETWEEN COLS	2	.0062415993	.0031207997		
RESIDUALS	6	.012696443	.0021160739		
TOTAL	12	.024407991			
BETWEEN ROWS	4	.0027475458	.00068688645		
BETWEEN COLS	2	.0089640021	.0044820011	2.118	.201
RESIDUALS	6	.012696443	.0021160739		
TOTAL	12	.024407991			

A WEIGHTED LEAST SQUARES ANALYSIS USING
13 NON-ZERO WEIGHTS AND 2 ZERO WEIGHTS IN COLUMN 12

COEFFICIENT		ESTIMATE	STD. DEV.
GRAND MEAN		-.00042717216	.0066603258
ROW	1	-.011283252	.013046439 H
ROW	2	-.00091895923	.011357944
ROW	3	.0085589784	.011603147
ROW	4	-.013821500 L	.012155542
ROW	5	.017464733 H	.011357944 L
COLUMN	1	.0069898284	.0088861861H
COLUMN	2	.012188053 H	.0088157889
COLUMN	3	-.019177882 L	.0088157888L
RESIDUAL			.046000803

5 X 3 TABLE OF RESIDUALS, STANDARDIZED BY
DIVIDING EACH RESIDUAL BY ITS STANDARD DEVIATION.

COLUMN / ROW	1	2	3
1	.30	.09	-.48
2	-.48	-.23	.73
3	-.78	-.80	1.59
4	.96	.96	-2.17
5	.00	.00	.00

An analysis of variance for these data is:

Source	*d.f.*	*Sums of Squares*	*Mean Squares*	*F Ratio*
Manufacturers (Rows)	4	.0054699488	.0013674872	
Resistance (Cols)	2	.0062415993	.0031207997	
Interaction	6	.012696443	.0021160739	2.12
Residual	45	.044841114	.00099646919	
Total	57	.069249106		

No measurements were available for the fourth manufacturer for low and medium resistance. Hence, two degrees of freedom are lost for estimating the manufacturers x resistance interaction.

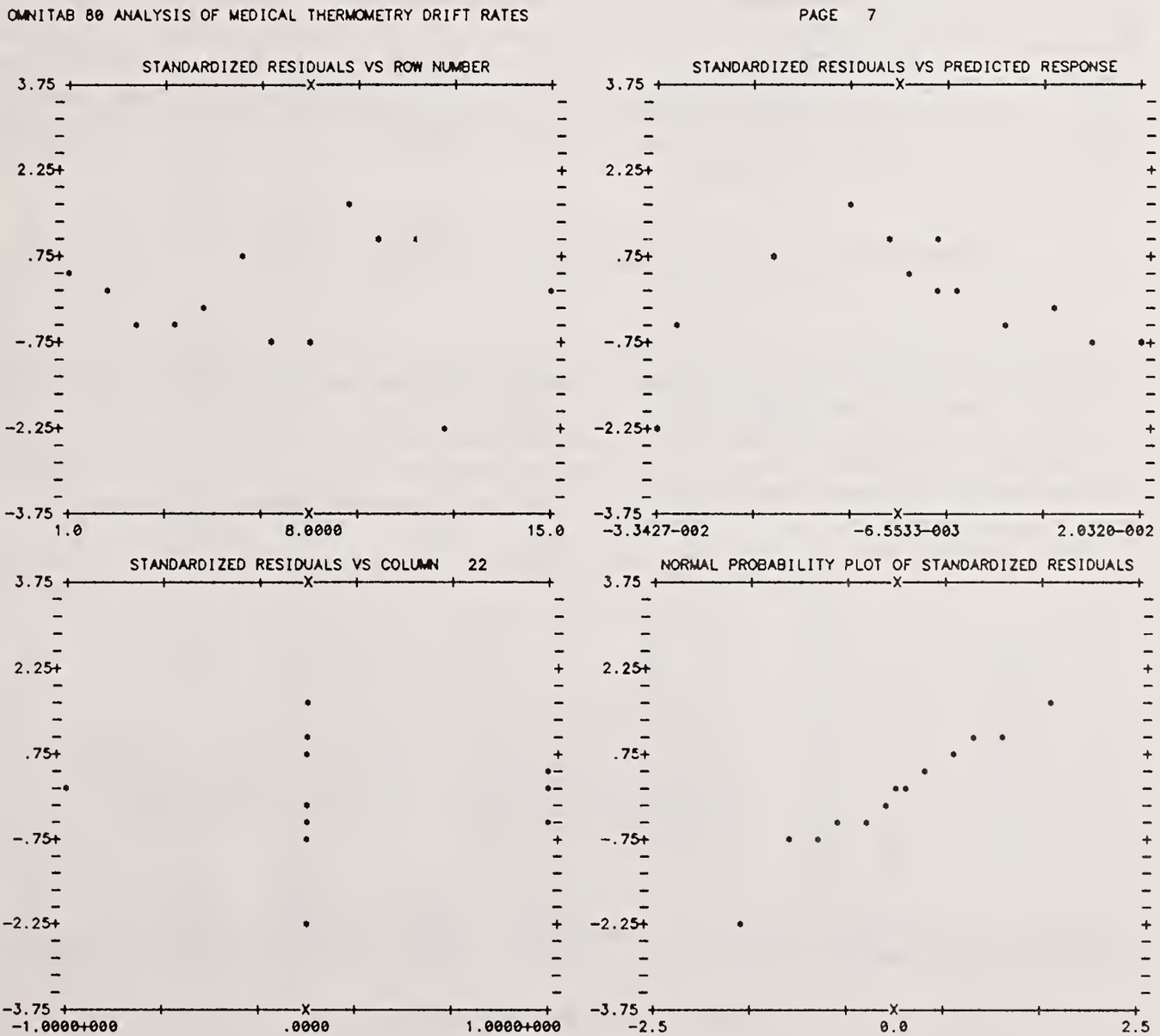
After an examination of the ONEWAY analysis, in this particular example, one might decide not to proceed with a TWOWAY analysis. It is clear from the box plots that one of the high resistance measurements for the fourth manufacturer is substantially smaller than any of the other measurements. An attempt should be made to determine the cause of this low measurement before proceeding with any further analysis.

The interaction F-ratio is almost significant at the 0.05 level. No sophisticated analysis is needed, as the interaction is undoubtedly due to the outlier detected by an examination of the box plots.

It is a good idea to examine plots of the residuals. The instruction:

FIT 13, 12,7, 21*27**

will produce the four plots of residuals.



STWOWAY analysis for **r** by **c** table, data in column **C**, put in **C** and successive columns

This instruction is identical to the first form of the TWOWAY instruction, except the automatic printing is omitted.

STWOWAY analysis for **r** by **c** table, data in **C**, store from **C** on, weights in column **C**

No automatic printing is produced. Otherwise the instruction is similar to the optional form of TWOWAY.

6.7 Correlation Analysis

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 C6.10 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.

CORRELATION between **p** variables in columns **C**, **C** ,..., **C**

Provides the automatic printing described below with no storage of results. The number of variables used (first argument) must be greater than 1 and less than 100. The arguments are all integers and the number of arguments is equal to the first argument plus one.

For the number of variables, **p**, the maximum number of measurements, n (NRMAX), allowed is given in the table below. The maximum value of **p** is 51 and the worksheet would have to be redimensioned to accommodate the maximum number of measurements. The table assumes the standard worksheet size.

p	*n*	*p*	*n*	*p*	*n*	*p*	*n*	*p*	*n*
		11	1106	21	589	31	388	41	279
2	3372	12	1019	22	561	32	375	42	270
3	3347	13	945	23	536	33	362	43	262
4	2676	14	880	24	512	34	349	44	254
5	2229	15	823	25	491	35	338	45	247
6	1909	16	773	26	471	36	327	46	240
7	1668	17	728	27	452	37	316	47	233
8	1481	18	688	28	435	38	306	48	226
9	1331	19	651	29	418	39	296	49	219
10	1209	20	619	30	403	40	287	50	213
								51	207

If the number of measurements exceeds the number in the table given above for a particular value of **p**, the following fatal error message is printed:

INSUFFICIENT SCRATCH AREA.

The following set of instructions using data taken from page 373 of Draper and Smith (1981)

```
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
```

produces the automatic printing of the following tables:

```
CORRELATION WITH 4 VARIABLES IN COLUMNS 11, 12, 13 AND 14

         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 COEFF'S (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 COEF'S (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 COEFF'S (FISHER APPROX.)
  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
```

The instruction provides an automatic printing of seven different tables. If the number of variables, **p**, is less than or equal to 6, all tables are printed in their entirety. If **p** exceeds 6, tables are printed in blocks with 6 or less columns in each block. 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.

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 understanding is best achieved by consulting several references. Kendall and Stuart (1961) is perhaps the best single reference.

The first table contains simple correlation coefficients. The title before the table gives the number of variables and the number of measurements (NRMAX). 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 n measurements, x_i and y_i (i=1,...,n), have been made on variables x and y respectively and where $\bar{x}$ is the average of x_i and $\bar{y}$ is the average of y_i. See, for example, section 1.6 (pages 43-45) of Draper and Smith (1981) 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 calculation of the correlation between columns 12 and 13 for the Draper and Smith data gives the coefficient

$$r_{xy}=\frac{-8.1644448}{\sqrt{(4.4555554)(502.81554)}}$$

$$=\frac{-.81644448}{47.332045}$$

$$=-.17249296\,.$$

The single correlation coefficient can also be obtained by using a FIT instruction to fit

$$y_i=\alpha+\beta x_i$$

and the formula

$$r_{xy}=\frac{\hat{\beta}}{\sqrt{\hat{\beta}^2+(n-2)s_{\hat{\beta}}^2}}$$

For the example, r_{xy} is computed from the equation above to be:

$$r_{xy} = \frac{-.016237455}{\sqrt{(-.016237455)+7\,(.035046006)}}$$

$$= -0.17249296$$

which agrees with the value in table I.

The second table consists of significance levels of simple correlation coefficients. For each pair of variables (x and y), the F-statistic with 1 and (n-2) degrees of freedom for testing whether or not the correlation coefficient is significantly different from zero is

$$F_0\,(1,n-2) = (n-2)\,r_{xy}^2/(1-r_{xy}^2)$$

The significance level for the test statistic is

$$S = \Pr(F \text{ exceeds } F_0).$$

Small values (less than .05 for example) of S indicate that the correlation coefficient is significantly different from zero. For columns 12 and 13

$$F_0 = 7(0.029753822)/(1-0.029753822) = 0.21466382 \text{ and } S = 0.6572.$$

The value of S can be easily verified by using the appropriate F PROBABILITY instruction (see sec. C7.2). 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.

Partial correlation coefficients are printed in the third table. 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

$$r_{ij\cdot} = \frac{-r^{ij}}{\sqrt{r^{ii}r^{jj}}} \quad .$$

By using MINVERT (sec. C11.5), it can be verified that the partial correlation between columns 12 and 13 is

$$r_{ij\cdot} = \frac{2.8514007}{\sqrt{(4.8973885)(3.4156597)}} = 0.69716998\,.$$

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.

The fourth table contains significance levels of partial correlation coefficients. For each pair of variables, the following F-statistic with 1 and $(n-p)$ degrees of freedom is computed from the partial correlation coefficient r_{xy}:

$$F_0(1,n-p) = (n-p)r_{xy\cdot}^2/(1-r_{xy\cdot}^2).$$

The significance level, S, is computed as

$$S = \Pr(\,F \text{ exceeds } F_0).$$

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,

$$F_0=(5)(0.48604604)/(1-0.48604604)=4.7284967 \text{ and } S=0.0817.$$

Compare this value of S with the value 0.6572 in the second table.

Spearman rank correlation coefficients are printed in the fifth table. 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:

$$\int xy=\frac{A-D_{xy}^2-T_x-T_y}{\sqrt{(A-2T_x)(A-2T_y)}},$$

where

$$D_{xy}^2 = \sum_{i=1}^{n}[\text{rank}(x_i)-\text{rank}(y_i)]^2,$$

$$A = (n-1)(n)(n+1)/6,$$

$$T_x = (1/12)\Sigma_x(t_x-1)(t_x)(t_x+1),$$

$$T_y = (1/12)\Sigma_y(t_y-1)(t_y)(t_y+1),$$

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 C6.1. If there are no ties T_x and T_y equal zero and

$$\int xy=1-D^2/A.$$

For columns 12 and 13, it can be easily verified using a RANKS instruction (see sec. C6.1) that the rank correlation coefficient equals

$$\int xy=\frac{120-134.5-0.5-0.0}{\sqrt{(120-1)(120-0)}}$$

$$= -15/119.49895=-0.12552412,$$

which does not differ greatly from the product moment correlation coefficient -0.1725 in the first table.

The sixth table contains significance levels for a test of a quadratic fit over a 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 F-tests of the hypothesis that the quadratic term in a quadratic model is zero. The F-statistic with 1 and (n-3) degrees of freedom is

$F_0(1,n-3)=$[Residual sum of squares (linear model)$-$Residual SS (quadratic model)]/
Residual variance (quadratic model)

The significance level, S, is then computed from

$$S = \Pr(F \text{ exceeds } F_0).$$

Small values of a 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,

$$F_0 = (4.3229856\text{-}4.2779137)/0.71298562 = 0.063215720 \text{ and } S = 0.8099.$$

The residual sums of squares can be obtained from POLYFIT and the value of S can be verified using F PROBABILITY (see sec. C7.2). 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.

The seventh table consists of the 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 interval for a correlation coefficient r is based on a normal approximation and 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

$$z = \tanh^{-1}(r) = \log_e[(1+r)/(1-r)]/2,$$

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

$$\tanh(-0.17423494 + 1.959964/\sqrt{6}) = \tanh(0.625917) = 0.5552.$$

Also, the lower 99% confidence limit equals

$$\tanh(-0.17423494\text{-}2.5758295/\sqrt{6}) = \tanh(-1.2258128) = -0.8414.$$

It is 95% confident that the "true" correlation coefficient lies between -0.7506 and $+0.5552$ and it is 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.)

An accuracy check of matrix inversion is made and the following fatal error is printed whenever appropriate:

MATRIX IS (NEARLY) SINGULAR.

CORRELATION p variables in **C, C** ,..., **C** put array of simple coefficients 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	.68374210	−.61596992	.80175221
2	.68374210	1.0000000	−.17249296	.76795024
3	−.61596992	−.17249296	1.0000000	−.62874595
4	.80175221	.76795024	−.62874595	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 p, C ,..., **C** put simple in **R, C** and partial coefficients 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	.43170938	−.45663594	.10539044
7	.43170938	1.0000000	.69716997	.72682004
8	−.45663594	.69716997	1.0000000	−.64778922
9	.10539044	.72682004	−.64778922	1.0000000

SCORRELATION p variables in **C, C** ,..., **C,** put array of simple coefficients in **R, C**

The instruction is the same as the second form of correlation except the automatic printing is suppressed.

SCORRELATION p, C ,..., **C** put simple in **R, C** and partial coefficients in **R, C**

Similar to the last form of CORRELATION without the automatic printing.

6.8 Analysis of Twoway Contingency Table

CONTINGENCY

Consider the case where subjects (individuals or experimental units) can be classified in two ways, for example by sex and marital status, and where each classification can take on two or more values. For example,

sex is either female or male and marital status is single, married or divorced. For each subject a count of one can be put into one and only one cell of a twoway table in which rows represent possible values of one classification and columns represent possible values of the other classification. A table which contains the total number of counts for n subjects is known as a twoway contingency table. The table can be used to assess whether the two classifications are independent and to answer other questions concerning the association between the two classifications.

CONTINGENCY table analysis of **r** x **c** table starting in column **C**

The table to be analyzed is located in the first **r** rows of the **c** consecutive columns starting with column C. The instruction automatically prints two (or more) pages of statistics and references for a comprehensive analysis of the specified twoway contingency table.

The size of the table which can be analyzed by a CONTINGENCY instruction is limited by the size of the worksheet. Let s be the smaller of **r** and **c** and let t be the larger of **r** and **c**. For each possible value of s from 1 to 40, the following table gives the maximum value of t which can be used, assuming a standard size worksheet.

s	*max t*	*s*	*max t*	*s*	*max t*	*s*	*max t*
1	1222*	11	147	21	78	31	53
2	707*	12	135	22	74	32	51
3	497*	13	125	23	71	33	49
4	383*	14	116	24	68	34	48
5	312*	15	108	25	65	35	47
6	263*	16	102	26	63	36	45
7	227*	17	96	27	60	37	44
8	200	18	91	28	58	38	43
9	178	19	86	29	56	39	42
10	161	20	82	30	54	40	41

The asterisk indicates that the worksheet has to be redimensioned for the largest values of t.

If two or more row totals, or column totals, are the same, the following informative diagnostic message is printed:

FIRST OF EQUAL ROW OR COLUMN TOTALS USED TO COMPUTE LAMBDA.

The frequency counts in a contingency table may be determined by using a TABLE FREQUENCY instruction. See section C6.9.

In a study of child behavior related to refrigerator door safety release devices, measurements were made on 201 children. Each child was assigned one of six safety release devices. The parents' combined education was classified into four groups 16-25 years, 26-30 years, 31-35 years, and 36-40 years. The experiment and data are described in Kullback et al. (1962). Using the instructions,

```
READ TABLE OF COUNTS IN COLUMNS 11***14
    4   13   13    1
    6   14    8    4
    4   16   16    5
   15   14   18    3
   10    6    0    0
    6   13   12    0
CONTINGENCY TABLE ANALYSIS OF 6 X 4 STARTING IN COLUMN 11
```

produces the following automatic printing.

CONTINGENCY TABLE ANALYSIS OF 6 BY 4 TABLE. * OBSERVED COUNT
OBSERVED COUNTS IN COLUMNS 11 TO 14. SHOWING * EXPECTED COUNT
HYPOTHESIS IS P(IJ) = P(I.)P(.J). * ———————
* DIFFERENCE

COL / ROW	11	12	13	14	TOTALS
1	4	13	13	1	31
	6.94	11.72	10.33	2.00	
	-2.94	1.28	2.67	-1.00	
2	6	14	8	4	32
	7.16	12.10	10.67	2.07	
	-1.16	1.90	-2.67	1.93	
3	4	16	16	5	41
	9.18	15.50	13.67	2.65	
	-5.18	.50	2.33	2.35	
4	15	14	18	3	50
	11.19	18.91	16.67	3.23	
	3.81	-4.91	1.33	-.23	
5	10	6	0	0	16
	3.58	6.05	5.33	1.03	
	6.42	-.05	-5.33	-1.03	
6	6	13	12	0	31
	6.94	11.72	10.33	2.00	
	-.94	1.28	1.67	-2.00	
TOTALS	45	76	67	13	201

FREEMAN-TUKEY DEVIATES

COL / ROW	11	12	13	14
1	-1.13	.43	.84	-.59
2	-.35	.59	-.78	1.19
3	-1.91	.19	.66	1.28
4	1.11	-1.14	.38	.00
5	2.56	.08	-3.73	-1.27
6	-.27	.43	.56	-2.00

REFERENCE. FREEMAN, M. F. AND TUKEY, J. W. (1950). TRANSFORMATIONS RELATED TO THE ANGULAR AND THE SQUARE ROOT. ANN. MATH. STATIST.,21,607.

ANALYSIS OF INFORMATION TABLES

INDEPENDENCE

COMPONENT DUE TO HYPOTHESIS	LIKELIHOOD CHI-SQUARED INFORMATION VALUE	D.F.	CHI-SQUARED PROBABILITY
P(IJ) = 1/24	115.664	23	.000
P(I.) = 1/6	20.418	5	.001
P(.J) = 1/4	56.350	3	.000
P(IJ) = P(I.)P(.J)	38.895	15	.001
(PEARSON CHI-SQUARED	34.046	15	.003)

HOMOGENEITY OF R SAMPLES

P(RJ) = 1/4 EACH SAMPLE	95.246	18	.000
P(J) = 1/4 ALL SAMPLES	56.350	3	.000
HOMOGENEITY OF R SAMPLES	38.895	15	.001
(PEARSON CHI-SQUARED	34.046	15	.003)

REFERENCE ...
KU, H. H., VARNER, R. N., AND KULLBACK, S. (1971). ON THE ANALYSIS OF MULTIDIMENSIONAL CONTINGENCY TABLES, J. AMER. STATIST. ASSOC.,66,55.

MEASURES OF ASSOCIATION	VALUE	ASYMPTOTIC STANDARD ERROR	RATIO
BASED ON CHI-SQUARED STATISTIC			
PHI (CRAMER)	.238		
CONTINGENCY COEFFICIENT	.381		
BASED ON RANK ORDER CORRELATION			
KENDALL'S TAU	-.131	.047	-2.753
SPEARMAN'S RHO	-.163	.070	-2.329
KRUSKAL'S GAMMA	-.172	.070	-2.479
BASED ON PROBABILITY OF PREDICTION			
LAMBDA A	.026	.040	.660
LAMBDA B	.064	.053	1.206
LAMBDA	.043	.038	1.144

NOTE ... DISCUSSIONS OF THESE MEASURES AND FORMULAS FOR COMPUTING STANDARD ERRORS MAY BE FOUND IN ...

GOODMAN, L. A. AND KRUSKAL, W. H., MEASURES OF ASSOCIATION FOR CROSS CLASSIFICATIONS. PART I, II, AND III, J. OF AMER. STATIST. ASSOC. VOL. 49,732. (1954). VOL. 54,123. (1959), VOL. 58,310 (1963).

KENDALL, M. AND STUART, A. (1973). THE ADVANCED THEORY OF STAT., V 2.

HAYS, W. L. (1973). STATISTICS FOR THE SOCIAL SCIENCES. 2ND ED.

It was expected that there was a relation between success in using safety devices and the combined years of education of the children's parents. The analysis displayed by the CONTINGENCY instruction shows that the distribution of parents' education among the six devices is not homogeneous and that further experimentation is necessary to conclude that the responses are dependent on parents' education.

6.9 Table Making or Cross Tabulation

NTABLE AVERAGE, NTABLE CPERCENTAGE, NTABLE CPROPORTION, NTABLE FREQUENCY, NTABLE MAXIMUM, NTABLE MEDIAN, NTABLE MINIMUM, NTABLE PERCENTAGE, NTABLE PROPORTION, NTABLE RANGE, NTABLE RPERCENTAGE, NTABLE RPROPORTION, NTABLE STDDEV, NTABLE SUM, TABLE AVERAGE, TABLE CPERCENTAGE, TABLE CPROPORTION, TABLE FREQUENCY, TABLE MAXIMUM, TABLE MEDIAN, TABLE MINIMUM, TABLE PERCENTAGE, TABLE PROPORTION, TABLE RANGE, TABLE RPERCENTAGE, TABLE RPROPORTION, TABLE STDDEV, TABLE SUM

A collection of data is often classified according to several variables, each of which can assume discrete values. For example, people can be classified by sex, marital status, education, etc. Sex, marital status, education, etc. will be called variables. The values that a variable can assume will be called levels. The levels must have numerical values, such as the numbers 0 and 1 to denote the two levels of sex. The interpretation of the data is aided considerably by cross tabulation or the construction of a table. The most common table is one which shows the frequency (or count) for every possible combination in the classification. The table will show, for example, how many of the persons in a study are male, divorced and have a college education. There may be a response variable, such as salary, for which a table of statistics is needed. For example, a table showing the average salary for divorced males with a college education, single females with an advanced degree, etc., may be needed. A large and powerful set of instructions for constructing and printing tables (cross tabulation) is included in OMNITAB.

Each command consists of two words. The first word is either TABLE or NTABLE. Commands with TABLE as the first word will start the automatic printing on a new page with titles, whereas commands with the first word NTABLE will not start printing on a new page and no titles are printed. The second word in the command may be any of the fourteen words AVERAGE, CPERCENTAGE, CPROPORTION, FREQUENCY, MAXIMUM, MEDIAN, MINIMUM, PERCENTAGE, PROPORTION, RANGE, RPERCENTAGE, RPROPORTION, STDDEV, or SUM.

The first argument in any of the instructions is always, **n**, the number of variables in the classification and must be greater than zero. The remaining arguments are always column numbers.

Each instruction automatically prints a table. The first line at the top of the table gives (i) the number of variables in the classification, (ii) the statistic tabulated, and (iii) the number of measurements. The coding scheme for the variables is shown below the title and includes the column number where the levels of the variable are stored and the number of levels for each of the variables. If a LABEL or HEAD instruction has been used, the heading is used instead of the column number.

A distinction is made between zero and nothing for all the tables except those for frequencies, percentages and proportions. When no data exists for a specific combination of levels of variables, an asterisk is printed in the table. In the table of standard deviations a zero is printed when there is only one measurement (or all the measurements are the same), but an asterisk is printed when the frequency is zero. A note is printed at the bottom of the table showing the number of cells in the table which are empty.

Row and column totals are printed for the TABLE (or NTABLE) FREQUENCY, MAXIMUM, MINIMUM, PERCENTAGE, PROPORTION, RANGE, and SUM instructions. The totals are not printed for the AVERAGE, MEDIAN or STDDEV instructions. Only row totals are printed for the RPERCENTAGE and RPROPORTION and column totals for the CPERCENTAGE and CPROPORTION instructions. The row totals are not printed for 1-way tables and no totals are printed for 1-way tables when the variable has a single level. For 2-way tables the row totals can be used directly. For 3-way and higher tables, the row totals can be summed by hand to obtain the different marginal totals.

Two decimal places are used to print the entries in the table produced by a TABLE (or NTABLE) CPERCENTAGE, PERCENTAGE and RPERCENTAGE instructions and four decimal places are used for a TABLE (or NTABLE) CPROPORTION, PROPORTION and RPROPORTION instructions. For all the other instructions and the levels of the variables, the maximum number of digits printed is five. The number of digits needed to print a number is computed. If a number can be printed with less than five digits, it is printed with the minimum number of digits required. The tables are printed compactly without any waste. The actual size will vary depending upon the number of digits needed to print the numbers. The WIDTH or INTERACTIVE instruction controls the width of each line. If neither instruction has been used, each line is printed with a maximum of 120 characters. If the second variable has a large number of levels, the page may not be wide enough to print all the columns on one page and some columns will be continued on the next page.

Each of the fourteen TABLE and the fourteen NTABLE instructions has an optional form with one additional argument to specify storage. Data is stored in (**n**+1+**r**) columns, starting with the last column number (last argument), where **r** is the number of levels of the second variable.

Items stored in the first column are:

ROW	*Description*
1	length of table,
2	number of columns in table,
3	total size of table,
4	number of columns of stored data.
5	starting column for storing levels and
6	last column of stored data.

The table is stored in the next **r** columns. The last **n** columns contain the number of levels in the first row and the values of the levels for each of the **n** variables in rows below the first row.

Examples are given below which use the following instructions and data:

```
READ DATA INTO COLUMNS   7    5   27   50
                         6   25   40  101
                         7   23   48  107
                         7   25   40  104
                         6   25   49  109
                         6   26   40  106
                         7   24   48  102
                         6   26   40  103
                         6   25   49  108
                         6   23   49  105
                         7   24   48  108
                         7   24   48  102
                         6   25   49  101
                         7   26   48  108
                         7   25   48  103
                         7   26   49  107
HEAD COL 5/ AGE
ADD 0.98765 TO COLUMN 27 AND PUT IN COLUMN 28
```

NTABLE AVERAGE n way with levels in columns **C, C ,..., C** for data in **C**

The value of **n** must be greater than zero and less than eight. A table of averages is printed for **n** levels.

NTABLE AVERAGE n way, levels in **C ,..., C**, data in **C**, start storing in **C**

Same as the above instruction except data are stored as described in the introduction.

NTABLE CPERCENTAGE n way with levels in columns **C, C ,..., C**

The value of **n** must be greater than zero and less than nine. Only column totals are printed. The TABLE PERCENTAGE instruction computes the frequency for each entry in the table, divides each entry by the total frequency and then multiplies by 100.0. Hence, the sum of all entries in the table is 100.0. In the NTABLE CPERCENTAGE instruction each entry is divided by the corresponding column total frequency so that the sum of all entries in a column is 100.0.

The result of using the instructions

```
SPACE 5
NTABLE CPERCENTAGE 3 way for data in columns 7, 5 and 27
```

is shown at the bottom of page 228.

NTABLE CPERCENTAGE n way, levels in columns **C ,..., C**, start storing in column **C**

Same as the above instruction except data are stored as described in the introduction.

NTABLE CPROPORTION n way with levels in columns C, C ,..., C

The value of **n** must be less than nine. Only column totals are printed. The TABLE PROPORTION instruction computes the frequency for each entry in the table and divides each entry by the total frequency. Hence, the sum of all entries in the table is 1.0. In the NTABLE CPROPORTION instruction each entry is divided by the corresponding column total frequency so that the sum of all entries in a column is 1.0.

The result of using the instructions

SPACE 5
NTABLE CPROPORTION 3 way for data in columns **7, 5** and **27**

is shown in the middle of page 229.

NTABLE CPROPORTION n way, levels in columns C ,..., C, start storing in column C

Same as the above instruction except data are stored as described in the introduction.

NTABLE FREQUENCY n way with levels in columns C, C ,..., C

The value of **n** must be less than nine. Row and column totals are printed for the table.

NTABLE FREQUENCY n way, levels in columns C, C ,..., C, start storing in C

Same as the above instruction except data are stored as described in the introduction.

NTABLE MAXIMUM n way, levels in columns C, C ,..., C for data in column C

The value of **n** must be less than eight. Values entered in row and column totals are the maximum values of each row or column, respectively.

The result of using the instructions

SPACE 5
NTABLE MAXIMUM 3 WAY FOR DATA IN COLUMNS **7, 5, 28 and 50**

is shown at the bottom of page 224.

NTABLE MAXIMUM n way, levels in C ,..., C data in C, start storing in C

Same as above instruction except data are stored as described in the introduction.

NTABLE MEDIAN n way, levels in columns C, C ,..., C, data in column C

The value of **n** must be less than eight. A table of medians is printed. For a definition of a median see section C6.1.

The result of using the instructions

SPACE 5
NTABLE MEDIAN 3 WAY FOR DATA IN COLUMNS **7, 5, 28 and 50**

is shown at the bottom of page 225.

NTABLE MEDIAN n way, levels in **C** ,..., **C**, data in **C**, start storing in **C**

Same as above instruction except data are stored as described in the introduction.

NTABLE MINIMUM n way, levels in columns **C**, **C** ,..., **C**, data in column **C**

The value of **n** must be less than eight. The values entered in row and column totals are the minimum values of each row or column, respectively.

NTABLE MINIMUM n way, levels in **C** ,..., **C**, data in **C**, start storing in **C**

Same as above instruction except data are stored as described in the introduction.

NTABLE PERCENTAGE n way with levels in columns **C**, **C** ,..., **C**

The value of **n** must be less than nine. Row and column totals are printed with the table.

NTABLE PERCENTAGE n way, levels in **C**, **C** ,..., **C**, start storing in **C**

Same as above instruction except data are stored as described in the introduction.

NTABLE PROPORTION n way with levels in columns **C**, **C** ,..., **C**

The value of **n** must be less than nine. Row and column totals are printed with the table.
The result of using the instructions

SPACE 5
NTABLE PROPORTION 3 WAY FOR DATA IN COLUMNS **7, 5, 28**

is shown at the bottom of page 226.

NTABLE PROPORTION n way, levels in **C**, **C** ,..., **C**, start storing in **C**

Same as the above instruction except data are stored as described in the introduction.

NTABLE RANGE n way, levels in columns C, C ,..., C, data in column C

The value of n must be less than eight. The values printed in the column and row totals are the range from minimum to the maximum for each row and column, respectively.

The meaning of a frequency total is obvious. The meaning of any of the totals in a table of ranges may not be quite so obvious. In the table of ranges on page 225, the number 5 in the first row of the row totals is the difference between the largest number in column 50 corresponding to level 40 in column 27, 106, and the smallest, 101.

NTABLE RANGE n way, levels in C ,..., C, data in C, start storing in C

Same as the above instruction except data are stored as described in the introduction.

NTABLE RPERCENTAGE n way with levels in columns C, C ,..., C

The value of n must be less than nine. Only row totals are printed. The TABLE PERCENTAGE instruction computes the frequency for each entry in the table, divides each entry by the total frequency and then multiplies by 100.0. Hence, the sum of all entries in the table is 100.0. In the NTABLE RPERCENTAGE instruction each entry is divided by the corresponding row total frequency so that the sum of all entries in a row is 100.0.

NTABLE RPERCENTAGE n way, levels in C, C ,..., C, start storing in C

Same as the above instruction except data are stored as described in the introduction.

NTABLE RPROPORTION n way with levels in columns C, C ,..., C

The value of n must be less than nine. Only row totals are printed. The TABLE PROPORTION instruction computes the frequency for each entry in the table and divides each entry by the total frequency. Hence, the sum of all entries in the table is 1.0. In the NTABLE RPROPORTION instruction each entry is divided by the corresponding row total frequency so that the sum of all entries in a row is 1.0.

NTABLE RPROPORTION n way, levels in C, C ,..., C, start storing in C

Same as the above instruction except data are stored as described in the introduction.

NTABLE STDDEV n way, levels in columns C, C ,..., C, data in column C

The value of n must be less than eight. No column or row totals are printed.

The result of using the instructions

SPACE 5
NTABLE STDDEV 3 WAY FOR DATA IN COLUMNS **7, 5, 28 and 50**

is shown at the bottom of page 223.

NTABLE STDDEV n way, levels in **C** ,..., **C**, data in **C**, start storing in **C**

Same as the above instruction except data are stored as described in the introduction.

NTABLE SUM n way, levels in columns **C**, **C** ,..., **C**, data in column **C**

The value of **n** must be less than eight. No column or row totals are printed.
The result of using the instructions

SPACE 5
NTABLE SUM 3 WAY FOR DATA IN COLUMNS **7, 5, 27 and 50**

is shown at the bottom of page 222.

NTABLE SUM n way, levels in **C** ,..., **C**, data in **C**, start storing in **C**

Same as the above instruction except data are stored as described in the introduction.

xxx

With the exception that each instruction starts the automatic printing on a new page, the next 28 instructions are exactly like the previous 28 instructions.

TABLE AVERAGE n way, levels in columns **C**, **C** ,..., **C**, data in column **C**

The result of using the instruction

TABLE AVERAGE 3 WAY FOR DATA IN COLUMNS **7, 5, 27** and **50**

is shown at the top of page 223.

TABLE AVERAGE n way, levels in **C** ,..., **C**, data in **C**, start storing in **C**

TABLE CPERCENTAGE n way with levels in columns **C**, **C** ,..., **C**

TABLE CPERCENTAGE n way, levels in **C**, **C** ,..., **C**, start storing in **C**

TABLE CPROPORTION n way with levels in columns **C**, **C** ,..., **C**

TABLE CPROPORTION n way, levels in **C**, **C** ,..., **C**, start storing in **C**

TABLE FREQUENCY n way with levels in columns **C, C ,..., C**

The result of using the instruction

TABLE FREQUENCY 4 WAY FOR DATA IN COLUMNS **7, 5, 27** and **50**

is shown on pages 221 and 222.

TABLE FREQUENCY n way, levels in **C, C ,..., C**, start storing in **C**

Page 227 shows the results of using the instructions:

TABLE FREQUENCY 3 WAY FOR DATA IN COLUMNS **7, 5** and **28 STORE 31**
SPACE 3
NPRINT COLUMNS **31 *** 35**
NPRINT COLUMNS **36 *** 38**

TABLE MAXIMUM n way, levels in columns **C, C ,..., C**, data in column **C**

TABLE MAXIMUM n way, levels in **C ,..., C**, data in **C**, start storing in **C**

TABLE MEDIAN n way, levels in columns **C, C ,..., C**, data in column **C**

TABLE MEDIAN n way, levels in **C ,..., C**, data in **C**, start storing in **C**

TABLE MINIMUM n way, levels in columns **C, C ,..., C**, data in column **C**

The result of using the instruction

TABLE MINIMUM 3 WAY FOR DATA IN COLUMNS **7, 5, 27** and **50**

is shown at the top of page 224.

TABLE MINIMUM n way, levels in **C ,..., C**, data in **C**, start storing in **C**

TABLE PERCENTAGE n way with levels in columns **C, C ,..., C**

The result of using the instruction

TABLE PERCENTAGE 3 WAY FOR DATA IN COLUMNS 7, 5, and 27

is shown at the top of page 226.

TABLE PERCENTAGE n way, levels in C, C ,..., C, start storing in C

TABLE PROPORTION n way with levels in columns C, C ,..., C

TABLE PROPORTION n way, levels in C, C ,..., C start storing in C

TABLE RANGE n way, levels in columns C, C ,..., C, data in column C

The result of using the instruction

TABLE RANGE 3 WAY FOR DATA IN COLUMNS 7, 5,, 27 and 50

is shown at the top of page 225.

TABLE RANGE n way, levels in C ,..., C, data in C, start storing in C

TABLE RPERCENTAGE n way with levels in columns C, C ,..., C

The result of using the instruction

TABLE RPERCENTAGE 3 WAY FOR DATA IN COLUMNS 7, 5 and 27

is shown at the top of page 228.

TABLE RPERCENTAGE n way, levels in C, C ,..., C, start storing in C

TABLE RPROPORTION n way with levels in columns C, C ,..., C

The result of using the instruction

TABLE RPROPORTION 3 WAY FOR DATA IN COLUMNS 7, 5 and 27

is shown at the top of page 229.

TABLE RPROPORTION n way, levels in **C**, **C** ,..., **C**, start storing in **C**

TABLE STDDEV n way, levels in columns **C**, **C** ,..., **C**, data in column **C**

TABLE STDDEV n way, levels in **C** ,..., **C**, data in **C**, start storing in **C**

TABLE SUM n way, levels in columns **C**, **C** ,..., **C**, data in column **C**

TABLE SUM n way, levels in **C** ,..., **C**, data in **C**, start storing in **C**

4-WAY TABLE OF FREQUENCIES OF 15 MEASUREMENTS

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS
(4) = COLUMN 50 , 9 LEVELS

(4)	(3)	(2) / (1)	23	24	25	26	TOTALS
101	40	6	0	0	1	0	1
		7	0	0	0	0	0
	48	6	0	0	0	0	0
		7	0	0	0	0	0
	49	6	0	0	1	0	1
		7	0	0	0	0	0
102	40	6	0	0	0	0	0
		7	0	0	0	0	0
	48	6	0	0	0	0	0
		7	0	2	0	0	2
	49	6	0	0	0	0	0
		7	0	0	0	0	0
103	40	6	0	0	0	1	1
		7	0	0	0	0	0
	48	6	0	0	0	0	0
		7	0	0	1	0	1
	49	6	0	0	0	0	0
		7	0	0	0	0	0
104	40	6	0	0	0	0	0
		7	0	0	1	0	1
	48	6	0	0	0	0	0
		7	0	0	0	0	0
	49	6	0	0	0	0	0
		7	0	0	0	0	0
105	40	6	0	0	0	0	0
		7	0	0	0	0	0
	48	6	0	0	0	0	0
		7	0	0	0	0	0
	49	6	1	0	0	0	1
		7	0	0	0	0	0

CONTINUATION

(4)	(3)	(2) (1)	23	24	25	26	TOTALS
106	40	6	0	0	0	1	1
		7	0	0	0	0	0
	48	6	0	0	0	0	0
		7	0	0	0	0	0
	49	6	0	0	0	0	0
		7	0	0	0	0	0
107	40	6	0	0	0	0	0
		7	0	0	0	0	0
	48	6	0	0	0	0	0
		7	1	0	0	0	1
	49	6	0	0	0	0	0
		7	0	0	0	1	1
108	40	6	0	0	0	0	0
		7	0	0	0	0	0
	48	6	0	0	0	0	0
		7	0	1	0	1	2
	49	6	0	0	1	0	1
		7	0	0	0	0	0
109	40	6	0	0	0	0	0
		7	0	0	0	0	0
	48	6	0	0	0	0	0
		7	0	0	0	0	0
	49	6	0	0	1	0	1
		7	0	0	0	0	0
	TOTALS		2	3	6	4	15

3-WAY TABLE OF SUMS OF 15 MEASUREMENTS IN COLUMN 50

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) (1)	23	24	25	26	TOTALS
40	6	*	*	101	209	310
	7	*	*	104	*	104
48	6	*	*	*	*	*
	7	107	312	103	108	630
49	6	105	*	318	*	423
	7	*	*	*	107	107
TOTALS		212	312	626	424	1574

* 14 OF THE 24 CELLS IN THE TABLE ARE EMPTY.

3-WAY TABLE OF AVERAGES OF 15 MEASUREMENTS IN COLUMN 50

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26
40	6	*	*	101.0	104.5
	7	*	*	104.0	*
48	6	*	*	*	*
	7	107.0	104.0	103.0	108.0
49	6	105.0	*	106.0	*
	7	*	*	*	107.0

* 14 OF THE 24 CELLS IN THE TABLE ARE EMPTY.

3-WAY TABLE OF STANDARD DEVIATIONS OF 15 MEASUREMENTS IN COLUMN 50

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 28 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26
40.988	6	*	*	.0000	2.1213
	7	*	*	.0000	*
48.988	6	*	*	*	*
	7	.0000	3.4641	.0000	.0000
49.988	6	.0000	*	4.3589	*
	7	*	*	*	.0000

* 14 OF THE 24 CELLS IN THE TABLE ARE EMPTY.

3-WAY TABLE OF MINIMUMS OF 15 MEASUREMENTS IN COLUMN 50

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) (1)	23	24	25	26	TOTALS
40	6	*	*	101	103	101
	7	*	*	104	*	104
48	6	*	*	*	*	*
	7	107	102	103	108	102
49	6	105	*	101	*	101
	7	*	*	*	107	107
TOTALS		105	102	101	103	101

* 14 OF THE 24 CELLS IN THE TABLE ARE EMPTY.

3-WAY TABLE OF MAXIMUMS OF 15 MEASUREMENTS IN COLUMN 50

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 28 , 3 LEVELS

(3)	(2) (1)	23	24	25	26	TOTALS
40.988	6	*	*	101	106	106
	7	*	*	104	*	104
48.988	6	*	*	*	*	*
	7	107	108	103	108	108
49.988	6	105	*	109	*	109
	7	*	*	*	107	107
TOTALS		107	108	109	108	109

* 14 OF THE 24 CELLS IN THE TABLE ARE EMPTY.

3-WAY TABLE OF RANGES OF 15 MEASUREMENTS IN COLUMN 50

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26	TOTALS
40	6	*	*	0	3	5
	7	*	*	0	*	0
48	6	*	*	*	*	*
	7	0	6	0	0	6
49	6	0	*	8	*	8
	7	*	*	*	0	0
	TOTALS	2	6	8	5	8

* 14 OF THE 24 CELLS IN THE TABLE ARE EMPTY.

3-WAY TABLE OF MEDIANS OF 15 MEASUREMENTS IN COLUMN 50

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 28 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26
40.988	6	*	*	101.0	104.5
	7	*	*	104.0	*
48.988	6	*	*	*	*
	7	107.0	102.0	103.0	108.0
49.988	6	105.0	*	108.0	*
	7	*	*	*	107.0

* 14 OF THE 24 CELLS IN THE TABLE ARE EMPTY.

3-WAY TABLE OF PERCENTAGES OF 15 MEASUREMENTS

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26	TOTALS
40	6	.00	.00	6.67	13.33	20.00
	7	.00	.00	6.67	.00	6.67
48	6	.00	.00	.00	.00	.00
	7	6.67	20.00	6.67	6.67	40.00
49	6	6.67	.00	20.00	.00	26.67
	7	.00	.00	.00	6.67	6.67
TOTALS		13.33	20.00	40.00	26.67	100.00

3-WAY TABLE OF PROPORTIONS OF 15 MEASUREMENTS

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 28 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26	TOTALS
40.988	6	.0000	.0000	.0667	.1333	.2000
	7	.0000	.0000	.0667	.0000	.0667
48.988	6	.0000	.0000	.0000	.0000	.0000
	7	.0667	.2000	.0667	.0667	.4000
49.988	6	.0667	.0000	.2000	.0000	.2667
	7	.0000	.0000	.0000	.0667	.0667
TOTALS		.1333	.2000	.4000	.2667	1.0000

3-WAY TABLE OF FREQUENCIES OF 15 MEASUREMENTS

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 28 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26	TOTALS
40.988	6	0	0	1	2	3
	7	0	0	1	0	1
48.988	6	0	0	0	0	0
	7	1	3	1	1	6
49.988	6	1	0	3	0	4
	7	0	0	0	1	1
TOTALS		2	3	6	4	15

6.0000000	0.	0.	1.0000000	2.0000000
4.0000000	0.	0.	1.0000000	0.
24.000000	0.	0.	0.	0.
8.0000000	1.0000000	3.0000000	1.0000000	1.0000000
36.000000	1.0000000		3.0000000	0.
38.000000				1.0000000

2.0000000	4.0000000	3.0000000
6.0000000	23.000000	40.987650
7.0000000	24.000000	48.987650
	25.000000	49.987650
	26.000000	

3-WAY TABLE OF PERCENTAGES IN ROWS OF 15 MEASUREMENTS

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26	TOTALS
40	6	.00	.00	33.33	66.67	100.00
	7	.00	.00	100.00	.00	100.00
48	6	.00	.00	.00	.00	100.00
	7	16.67	50.00	16.67	16.67	100.00
49	6	25.00	.00	75.00	.00	100.00
	7	.00	.00	.00	100.00	100.00

3-WAY TABLE OF PERCENTAGES IN COLUMNS OF 15 MEASUREMENTS

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) / (1)	23	24	25	26
40	6	.00	.00	16.67	50.00
	7	.00	.00	16.67	.00
48	6	.00	.00	.00	.00
	7	50.00	100.00	16.67	25.00
49	6	50.00	.00	50.00	.00
	7	.00	.00	.00	25.00
TOTALS		100.00	100.00	100.00	100.00

3-WAY TABLE OF PROPORTIONS IN ROWS OF 15 MEASUREMENTS

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) (1)	23	24	25	26	TOTALS
40	6	.0000	.0000	.3333	.6667	1.0000
	7	.0000	.0000	1.0000	.0000	1.0000
48	6	.0000	.0000	.0000	.0000	1.0000
	7	.1667	.5000	.1667	.1667	1.0000
49	6	.2500	.0000	.7500	.0000	1.0000
	7	.0000	.0000	.0000	1.0000	1.0000

3-WAY TABLE OF PROPORTIONS IN COLUMNS OF 15 MEASUREMENTS

(1) = COLUMN 7 , 2 LEVELS
(2) = AGE , 4 LEVELS
(3) = COLUMN 27 , 3 LEVELS

(3)	(2) (1)	23	24	25	26
40	6	.0000	.0000	.1667	.5000
	7	.0000	.0000	.1667	.0000
48	6	.0000	.0000	.0000	.0000
	7	.5000	1.0000	.1667	.2500
49	6	.5000	.0000	.5000	.0000
	7	.0000	.0000	.0000	.2500
TOTALS		1.0000	1.0000	1.0000	1.0000

6.10 References for Section 6

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, M.; Stegun, I. Handbook of Mathematical Functions, Natl. Bur. Stand. (U.S.) Appl. Math. Ser. 55; 1964.

Anscombe, F. J.; Tukey, J. W. The examination and analysis of residuals. Technometrics 5: 141-160; 1963.

Bradley, J. V. Distribution-free Statistical Tests. Englewood Cliffs, N.J.: Prentice-Hall Inc.; 1968.

Brownlee, K. A. Statistical Theory and Methodology in Science and Technology. 2nd Ed., New York, NY: John Wiley and Sons, Inc.; 1965.

Conover, W. J. Practical Nonparametric Statistics. New York, NY: John Wiley and Sons, Inc.; 1971.

Cook, Dennis R. Detection of influential observations in linear regression. Technometrics 19; 1977. 15 p.

Crow, E. L.; Siddiqui, M. M. Robust estimation of location. J. Amer. Statist. Assoc. 62: 353-389; 1967.

Daniel, Cuthbert; Wood, Fred S. Fitting Equations to Data. New York: Wiley-Interscience; 1971.

Davis, P. J. Orthonormalization codes in numerical analysis, chapter 10 in Survey of Numerical Analysis. Todd, J., Ed., New York: McGraw-Hill Book Company; 1962.

De Boor, C.; Rice, J. R. Least squares cubic spline approximation I—fixed knots. Report CSD TR 20, Computer Sciences Department, Purdue University, Lafayette, Indiana; 1968.

Dixon, W. J.; Massey, F. J., Jr. Introduction To Statistical Analysis. 2nd Ed., New York: McGraw-Hill Book Company; 1957.

Draper, N. R.; Smith, H. Applied Regression Analysis. 2nd Ed., New York: John Wiley and Sons, Inc.; 1981.
Duncan, A. J. Quality Control and Industrial Statistics. 3rd Ed., Homewood, Illinois: Richard D. Irwin, Inc.; 1965.
Eisenhart, C. Significance of the largest of a set of sample estimates of variance. Techniques of Statistical Analysis. Eisenhart, C.; Hastay, M. W.; Wallis, W. A. Eds. Statistical Research Group, Columbia University. New York: McGraw-Hill Book Company; 375-382; 1947.
Fisher, R. A. Statistical Methods for Research Workers, 11th Ed., New York: Hafner Publishing Company; 1950.
Freund, J. E.; Williams, F. J. Modern Business Statistics. Englewood Cliff, N.J.: Prentice-Hall, Inc.; 1958.
Gardner, D. A.; Hull, N. C. An approximation to two-sided tolerance limits for normal populations. Technometrics, 8: 115- 122; 1966.
Gibbons, J. D. Nonparametric Statistical Inference. New York: McGraw-Hill Book Company; 1971.
Goldberger, A. S. Econometric Theory. New York: John Wiley & Sons, Inc.; 1964.
Graybill, F. A. An Introduction to Linear Statistical Models, Vol. 1. New York: John Wiley & Sons, Inc.; 1961.
Hettmansperger, T. P.; McKean, J. W. A robust alternative based on ranks to least squares in analyzing linear models. Technometrics, 19; 1977. 275 p.
Hicks, Charles R. Fundamental Concepts in the Design of Experiments. New York: Holt, Rinehard and Winston; 1964.
Hill, R. W.; Holland, P. W. Two robust alternatives to least-squares regression. J. Am. Statist. Assoc., 72: 828-833; 1977.
Hoaglin, David C.; Welsch, Roy E. The hat matrix in regression and ANOVA. Amer. Statistician, 32; 1978. 17 p.
Hocking, R. R. The analysis and selection of variables in linear regression. Biometrics, 32; 1-49; 1976.
Hogben, David. Selected references. Precision Measurement and Calibration, Statistical Concepts and Procedures. Natl. Bur. Stand. (U.S.) Spec. Publ. 300, Vol. 1, Ku, H. H. Ed.; 402-407; 1969.
Jenkins, G. M.; Watts, D. G. Spectral Analysis and its Applications. San Francisco, California: Holden-Day; 1968.
Kempthorne, O. The Design and Analysis of Experiments. New York: John Wiley & Sons, Inc.; 1952.
Kendall, M. G. Rank Correlation Methods. London: Charles Griffin & Co.; 1948.
Kendall, M. G.; Stuart, A. The Advanced Theory of Statistics, Vol. 2; New York: Hafner Publishinag Company; 1961.
Ku, H. H. A User's Guide to the OMNITAB Command "Statistical Analysis". NBS Tech Note 756; 1973.
Kullback, J. S.; Kupperman, M.; Ku, H. H. An application of information theory to the analysis of contingency tables, with a table of 2n ln n, n=1(1)10,000. J. Res. Natl. Bur. Stand. (U.S.). B, 217-243; 1962.
Longley, J. W. 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; 1967.
Mandel, John. The Statistical Analysis of Experimental Data. New York: Interscience; 1964.
McGill, R.; Tukey, J. W.; Larson, W. A. Variations of box plots. American Statistician: 12-16; 1977.
Morrison, D. F. Multivariate Statistical Methods. New York: McGraw Hill Book Company; 1967.
Natrella, M. G. Experimental Statistics. Natl. Bur. Stand. (U.S.). Handb. 91; 1963.
Owen, D. B. Handbook of Statistical Tables, Reading, Mass: Addison-Wesley Publishing Co; 1962.
Snedecor, G. W. Statistical Methods, 5th Ed., Ames, Iowa: Iowa State University Press; 1956.
Snedecor, G. W.; Cochran, W. G. Statistical Methods. 6th Ed., Ames, Iowa: Iowa State University Press; 1967.
Wampler, R. H.; An evaluation of linear least squares computer programs. J. Res. Natl. Bur. Stand. (U.S.). 73B, 59-90; 1969.
Wampler, R. H.; A report on the accuracy of some widely used least squares computer programs. J. Amer. Statist. Assoc., 65, 549-565; 1970.
Young, L. C. On randomness in ordered sequences. Annals of Mathematical Statistics, 12: 293-300; 1941.
Zelen, M.; Linear estimation and related topicas. Chapter 17 in Survey of Numerical Analysis, Todd, J. Ed. New York: McGraw-Hill Book Company; 558-584; 1962.

7. PROBABILITY

Ninety-six instructions for evaluating probability density functions, cumulative distribution functions and percent point functions, for generating random samples and for producing probability plots are described in this section. Each instruction has a two word command. The first word of the command specifies the distribution as in BINOMIAL, NORMAL, etc. The second word describes the property of the distribution as in CUMULATIVE, DENSITY, etc.

7.1 Probability Density Functions

BETA DENSITY, BINOMIAL DENSITY, CAUCHY DENSITY,
DEXPONENTIAL DENSITY, EXPONENTIAL DENSITY, EXTREME DENSITY,
GEOMETRIC DENSITY, HALFNORMAL DENSITY, LAMBDA DENSITY,
LOGISTIC DENSITY, LOGNORMAL DENSITY, NEGBINOMIAL DENSITY,
NORMAL DENSITY, PARETO DENSITY, POISSON DENSITY,
UNIFORM DENSITY, WEIBULL DENSITY

Care should be taken to use only values for which the probability density function is defined and to use proper values for the parameter(s), if any. In each case, it is assumed that the probability density function is zero outside the specified range. An arithmetic fault occurs if an improper value is used.

Formulas for the probability density functions may be found in Norman Johnson and Samuel Kotz, Continuous Univariate Distributions, Vol. 1 and 2, Houghton Mifflin Co., 1970.

BETA DENSITY of x = **E** with a = **E** and b = **E** put in column **C**

Compute the beta probability density function

$$f(x)=\frac{1}{B(a,b)}x^{a-1}(1-x)^{b-1};\ 0\leqslant x\leqslant 1 \text{ and } a>0,\ b>0,\ B(a,b)=\Gamma(a)\Gamma(b)/\Gamma(a+b)$$

and put results in the designated column. The values of a and b (with a decimal point) must be positive integers or half-integers. The instruction

BETA DENSITY of x=**0.5** with a=**3.0** and b=**5.0** put in column **35**

would put 1.6406250 into every row of column 35.

The informative diagnostic message

A VALUE OF DEGREES OF FREEDOM WAS TRUNCATED TO AN INTEGER.

will be printed, if the value of a or b is not a half-integer. The word INTEGER should be read INTEGER OR HALF-INTEGER and DEGREES OF FREEDOM should be interpreted as a or b. A separate error message for half-integers does not exist.

BINOMIAL DENSITY of x = **E** with n = **E** and p = **E** put in column **C**

Compute the binomial probability density function

$$f(x)=\binom{n}{x}p^{x}(1-p)^{n-x};\ x=0,1,\ldots,n \text{ and } 0<p<1$$

and put results in the designated column. The value(s) of x must be non-negative integer(s) with a decimal point, n must be a positive integer(s) with a decimal point and p must be between 0 and 1 exclusive. The instruction

BINOMIAL DENSITY of x=**5.0** with n = **20.0** and p = **0.2** put in column **35**

would put .17455952 into every row of column 35. The value .17455952 is the probability of obtaining 5 successes in 20 Bernoulli trials in which the probability of success in a single trial is 0.2.

CAUCHY DENSITY of x = **E** put in column **C**

Compute the Cauchy probability density function

$$f(x)=\frac{1}{\pi(1+x^2)};\quad -\infty<x<\infty$$

and put in the designated column. The instruction

CAUCHY DENSITY of x=**0.5** put in column **35**

would put .25464791 into every row of column 35.

DEXPONENTIAL DENSITY of x = **E** put in column **C**

Compute the double exponential probability density function

$$f(x)=e^{-|x|}/2;\quad -\infty<x<\infty$$

and put in the designated column. The instruction

DEXPONENTIAL DENSITY of x=**0.5** put in column **35**

would put .30326533 into every row of column 35.

EXPONENTIAL DENSITY of x = **E** put in column **C**

Compute the exponential probability density function

$$f(x)=e^{-x};\quad x\geqslant 0$$

and put in the designated column. The instruction

EXPONENTIAL DENSITY of x=**0.5** put in column **35**

would put .60653066 into every row of column 35.

EXTREME DENSITY of x = **E** put in column **C**

Compute the extreme value type 1 probability density function

$$f(x)=e^{-x}\,e^{-e^{-x}};\quad -\infty<x<\infty$$

and put in the designated column. The instruction

EXTREME DENSITY of x=**0.5** put in column **35**

would put .33070430 into every row of column 35.

EXTREME DENSITY of x = **E** with γ = **E** put in column **C**

Compute the extreme value type 2 probability density function

$$f(x)=\gamma x^{-\gamma-1}e^{-x^{-\gamma}};\quad x>0$$

and put in the designated column. The values of x and γ must be positive.

If x is zero or negative, the following arithmetic fault is printed

Y=F(X) IS NOT DEFINED FOR SPECIFIED VALUE OF X.

If γ is zero or negative, the arithmetic fault printed is

FUNCTION NOT DEFINED FOR SPECIFIED PARAMETER VALUE.

The instruction

EXTREME DENSITY of x=**0.5** with γ=**0.6** put in column **35**

would put .39951390 into every row of column 35.

GEOMETRIC DENSITY of x = **E** with p = **E** put in column **C**

Compute the geometric probability density function

$$f(x)=p(1-p)^x;\quad x=0,1,2,\ldots \text{ and } 0<p<1$$

and put in the designated column. The value(s) of x must be a non-negative integer with a decimal point and p must be between 0 and 1 exclusive. The instruction

GEOMETRIC DENSITY of x=**3.0** with p = **0.2** put in column **35**

would put .10240000 into every row of column 35. The value .10240000 is the probability of obtaining 3 failures before 1 success in an indefinite sequence of Bernoulli trials in which the probability of a success in a single trial is 0.2.

HALFNORMAL DENSITY of x = **E** put in column **C**

Compute the half-normal probability density function

$$f(x)=\sqrt{2/\pi}\,e^{-x^2/2};\quad x>0$$

and put in the designated column. The value(s) of x must be non-negative. The instruction

HALFNORMAL DENSITY of x=**0.5** put in column **35**

would put .70413066 into every row of column 35.

LAMBDA DENSITY of x = **E** with θ = **E** put in column **C**

The lambda probability density function of the specified value(s) with θ is put in the designated column. The Tukey lambda variate does not have a simple probability density function. However, its percent point function is given by

$$x(p)=\frac{1}{\theta}(p^{\theta}-(1-p)^{\theta}); \quad 0<p<1 \qquad \begin{array}{l}\text{for } \theta>0; \quad -\frac{1}{\theta}\leqslant x\leqslant\frac{1}{\theta} \\ \text{for } \theta\leqslant 0; \quad -\infty<x<\infty\end{array}$$

The instruction

LAMBDA DENSITY of **0.5** with θ=**0.6** put in column **35**

would put .36320394 into every row of column 35.

LOGISTIC DENSITY of x = **E** put in column **C**

Compute the logistic probability density function

$$f(x)=\frac{e^{x}}{(1+e^{x})^{2}}; \quad -\infty<x<\infty$$

and put in the designated column. The instruction

LOGISTIC DENSITY of x=**0.5** put in column **35**

would put .23500371 into every row of column 35.

LOGNORMAL DENSITY of x = **E** put in column **C**

Compute the log-normal probability density function

$$f(x)=\frac{e^{-(\log x)^{2}/2}}{x\sqrt{2\pi}}; \quad x>0$$

and put in designated column. The value(s) of x must be positive.

LOGNORMAL DENSITY of x=**0.5** put in column **35**

would put .62749608 into every row of column 35.

NEGBINOMIAL DENSITY of x = **E** with n = **E** and p = **E** put in column **C**

Compute the negative binomial probability density function

$$f(x)=\binom{n+x-1}{x}p^{n}(1-p)^{x}; \quad x=0,1,2,\ldots, \quad 0<p<1 \quad \text{and} \quad n>0$$

and put in the designated column. The value(s) of x must be non-negative integer(s) with a decimal point. The value(s) of n must be positive integer(s) with a decimal point and p must be between 0 and 1 exclusive. The instruction

NEGBINOMIAL DENSITY of x=**5.0** with n = **20.0** and p = **0.8** put in column **35**

would put .15681208 into every row of column 35. The value .15681208 is the probability of obtaining exactly 5 failures before the 20th success in an indefinite Bernoulli sequence when the probability of success in a single trial is 0.8.

NORMAL DENSITY of x = **E** put in column **C**

Compute normal probability density function

$$f(x)=\frac{1}{\sqrt{2\pi}}e^{-x^2/2};\quad -\infty<x<\infty$$

and put in the designated column. The instruction

NORMAL DENSITY of x=**0.5** put in column **35**

would put .35206533 into every row of column 35.

PARETO DENSITY of x = **E** with θ = **E** put in column **C**

Compute Pareto probability density function

$$f(x)=\frac{\theta}{x^{\theta+1}};\quad x\geqslant 1,\quad \theta>0$$

and put in the designated column. The value(s) of x must be greater than or equal to one and value(s) of θ must be positive. The instruction

PARETO DENSITY of x=**1.5** with θ=**0.6** put in column **35**

would put .31362107 into every row of column 35.

POISSON DENSITY of x = **E** with mean λ = **E** put in column **C**

Compute the Poisson probability density function

$$f(x)=\frac{\lambda^x e^{-\lambda}}{x!};\quad x=0,1,2,\ldots,\text{ and }\lambda>0$$

and put in the designated column. The value(s) of x must be non-negative integer(s) with a decimal point and value(s) of λ must be greater than zero.

POISSON DENSITY of x=**0.5** with mean λ=**6.0** put in column **35**

would put .16062314 into every row of column 35. The value .16062314 is the probability of obtaining 5 counts in a Poisson process with mean 6.0.

UNIFORM DENSITY of x = **E** put in column **C**

Compute the uniform probability density function

$$f(x)=1; \quad 0 \leqslant x \leqslant 1$$

and put in the designated column. The value(s) of x must be between 0 and 1 inclusive. The instruction

UNIFORM DENSITY of x=**0.5** put in column **35**

would put 1.0000000 into every row of column 35.

WEIBULL DENSITY of x = **E** with θ = **E** put in column **C**

Compute the Weibull probability density function

$$f(x)=\theta x^{\theta-1} e^{-x^{\theta}}; \quad x>0, \quad \theta>0$$

and put in the designated column. The values of x and θ must be positive. The instruction

WEIBULL DENSITY of x=**0.5** with θ=**0.6** put in column **35**

would put .40929434 into every row of column 35.

7.2 Cumulative Distribution Functions

BETA CUMULATIVE, BINOMIAL CUMULATIVE, CAUCHY CUMULATIVE, CHISQUARED CUMULATIVE, DEXPONENTIAL CUMULATIVE, EXPONENTIAL CUMULATIVE, EXTREME CUMULATIVE, F CUMULATIVE, F PROBABILITY, GAMMA CUMULATIVE, GEOMETRIC CUMULATIVE, HALFNORMAL CUMULATIVE, LAMBDA CUMULATIVE, LOGISTIC CUMULATIVE, LOGNORMAL CUMULATIVE, NEGBINOMIAL CUMULATIVE, NORMAL CUMULATIVE, PARETO CUMULATIVE, POISSON CUMULATIVE, T CUMULATIVE, UNIFORM CUMULATIVE, WEIBULL CUMULATIVE

Twenty-one commands evaluate the cumulative distribution functions and one command computes the right-tail area of an F-distribution. All instructions are two word commands. Care must be exercized to use values for which the cumulative distribution function is defined. An arithmetic fault occurs if an improper parameter value is used. For formulas of the probability density functions see section C7.1.

BETA CUMULATIVE of **E** with a = **E** and b = **E** put in column **C**

The beta cumulative distribution function of the specified value(s) with parameters a and b is put in the designated column. The parameters (with a decimal point) must be positive integers or half-integers. The instruction

BETA CUMULATIVE of x=**0.5** with a=**3.0** and b=**6.0** put in column **36**

would put .85546875 into every row of column 36.

The informative diagnostic message

A VALUE OF DEGREES OF FREEDOM WAS TRUNCATED TO AN INTEGER.

will be printed, if the values of a and/or b are not an integer or a half-integer. The word INTEGER should be read INTEGER OR HALF-INTEGER and DEGREES OF FREEDOM should be interpreted as PARAMETER. A separate error message for half-integers does not exist.

BINOMIAL CUMULATIVE of x = **E** with n = **E** and p = **E** put in column **C**

The binomial cumulative distribution function of the specified value(s) is put in the designated column. The value(s) specified by the first argument must be non-negative integers. If the first argument is a constant, it must contain a decimal point. The value of n (second argument) must be a positive integer with a decimal point, if a constant, and p (third argument) must be greater than zero and less than one. The instruction

BINOMIAL CUMULATIVE of x=**5.0** with n = **20.0** and p = **0.2** put in column **36**

would put .80420779 into every row of column 36. The value .80420779 is the probability of obtaining 5 or less successes in 20 Bernoulli trials in which the probability of success in a single trial is 0.2.

CAUCHY CUMULATIVE of x = **E** put in column **C**

The Cauchy cumulative distribution function of the specified value(s) is put in the designated column. The instruction

CAUCHY CUMULATIVE of x=**0.5** put in column **36**

would put .64758362 into every row of column 36.

CHISQUARED CUMULATIVE of x = **E** with **E** degrees of freedom put in column **C**

The chisquared cumulative distribution function of the specified value(s) with **E** degrees of freedom is put in the designated column. The formula for the chisquared probability density function is

$$f(x)=\frac{x^{(k-2)/2}}{2^{k/2}\,e^{x/2}\,\Gamma\left(\frac{k}{2}\right)};\quad k>0,\quad x\geqslant 0$$

where k is degrees of freedom. The instruction

CHISQUARED CUMULATIVE of x=**4.5** with **6.0** degrees of freedom put in column **36**

would put .39066073 into every row of column 36.

DEXPONENTIAL CUMULATIVE of x = E put in column C

The double exponential cumulative distribution function of the specified value(s) is put in the designated column. The instruction

DEXPONENTIAL CUMULATIVE of x=**0.5** put in column **36**

would put .69673467 into every row of column 36.

EXPONENTIAL CUMULATIVE of x = E put in column C

The exponential cumulative distribution function of the specified value(s) is put in the designated column. The instruction

EXPONENTIAL CUMULATIVE of x=**0.5** put in column **36**

would put .39346934 into every row of column 36.

EXTREME CUMULATIVE of x = E put in column C

The extreme value type 1 cumulative distribution function of the specified value(s) is put in the designated column. The instruction

EXTREME CUMULATIVE of x=**0.5** put in column **36**

would put .45476079 into every row of column 36.

EXTREME CUMULATIVE of x = E with γ = E put in column C

The extreme value type 2 cumulative distribution function of the specified value(s) with the specified γ is put in the designated column. The second argument γ must be positive. The instruction

EXTREME CUMULATIVE of x=**0.5** with γ=**0.6** put in column **36**

would put .21965074 into every row of column 36.

F CUMULATIVE of x = E with m = E and n = E put in column C

Snedecor's F cumulative distribution function of the specified value(s) with m=E and n=E is put in the designated column. The formula for Snedecor's F probability density function is

$$f(x)=\frac{\Gamma\left(\frac{m+n}{2}\right)\left(\frac{m}{n}\right)^{m/2}x^{(m-2)/2}}{\Gamma\left(\frac{m}{2}\right)\Gamma\left(\frac{n}{2}\right)(1+mx/n)^{(m+n)/2}} \quad ; \quad x>0 \quad \text{and} \quad m, \quad n>0$$

The instruction

F CUMULATIVE of x=**0.5** with m=**6.0** and n=**7.0** put in column **36**

would put .20830941 into every row of column 36. An F CUMULATIVE instruction gives the complement of an F PROBABILITY instruction described below. I.e., the instruction

F PROBABILITY with **6.0** and **7.0** degrees of freedom of **0.5** put in column **46**

would put .79169059 into every row of column 46. *Caution:* the degrees of freedom are specified by the 2nd and 3rd arguments of a F CUMULATIVE instruction, but by the 1st and 2nd arguments of a F PROBABILITY instruction.

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. The probability that a random variable, which follows the F-distribution with the specified degrees of freedom, exceeds F is put in the column designated by the fourth argument. The instruction computes Q(F), the notation used by Abramowitz, M.; Stegun, I.A. Handbook of Mathematical Functions, Natl. Bur. Stand. (U.S.) Applied Mathematics Series 55; 1964 June. The instruction

F PROBABILITY with **3.0** and **5.0** degrees of freedom of **5.4095** put in column **46**

would put the number 0.049999177 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:

A VALUE OF DEGREES OF FREEDOM WAS TRUNCATED TO AN 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:

A VALUE OF DEGREES OF FREEDOM LESS THAN 1 WAS RESET TO 1.

Values specified by the third argument must be greater than or equal to zero. Otherwise, the following informative diagnostic is printed:

VALUE OUTSIDE ALLOWABLE RANGE. RESULT WAS SET EQUAL TO 1.0.

GAMMA CUMULATIVE of x=**E** with p=**E** put in column **C**

The gamma cumulative distribution function of the specified value(s) with parameter p is put in the designated column. The formula for the gamma probability density function is

$$f(x)=\frac{1}{\Gamma(p)}x^{p-1}e^{-x};\quad p>0\quad\text{and}\quad x>0.$$

The instruction

GAMMA CUMULATIVE of x=**0.5** with p=**0.6** put in column **36**

would put .61890102 into every row of column 36.

GEOMETRIC CUMULATIVE of x = **E** with p = **E** put in column **C**

The geometric cumulative distribution function of the specified value(s) with p is put in the designated column. The value(s) specified by the first argument must be non-negative integers. If the argument is not a column number, it should be written with a decimal point. The parameter p (second argument) must be between 0 and 1, exclusive. The instruction

GEOMETRIC CUMULATIVE of **5.0** with p = **0.2** put in column **36**

would put .73785600 into every row of column 36. The value .73785600 is the probability of obtaining 5 or less successes before 1 failure in an indefinite sequence of Bernoulli trials in which the probability of a success in a single trial is 0.2.

HALFNORMAL CUMULATIVE of x = **E** put in column **C**

The half-normal cumulative distribution function of the specified value(s) is put in the designated column. The instruction

HALFNORMAL CUMULATIVE of x=**0.5** put in column **36**

would put .38292494 into every row of column 36.

LAMBDA CUMULATIVE of x=**E** with θ=**E** put in column **C**

The lambda cumulative distribution function of the specified value(s) with θ is put in the designated column. The value specified by the second argument must be positive. The instruction

LAMBDA CUMULATIVE of x=**0.5** with θ=**0.6** put in column **36**

would put .68687725 into every row of column 36.

LOGISTIC CUMULATIVE of x = **E** put in column **C**

The logistic cumulative distribution function of the specified value(s) is put in the designated column. The instruction

LOGISTIC CUMULATIVE of x=**0.5** put in column **36**

would put .62245933 into every row of column 36.

LOGNORMAL CUMULATIVE of x = **E** put in column **C**

The log-normal cumulative distribution function of the specified value(s) is put in the designated column. The instruction

LOGNORMAL CUMULATIVE of x=**0.5** put in column **36**

would put .24410851 into every row of column 36.

NEGBINOMIAL CUMULATIVE of x = **E** with n = **E** and p = **E** put in column **C**

The negative binomial cumulative distribution function of the specified value(s) is put in the designated column. The value(s) specified by the first argument must be non-negative integers. If the first argument is a constant, it must contain a decimal point. The value of n (second argument) must be a positive integer with a decimal point if a constant and p (third argument) must be greater than zero and less than one. The instruction

NEGBINOMIAL CUMULATIVE of x = **5.0** with n = **6.0** and p = **0.8** put in column **36**

would put .98834579 into every row of column 36. The value .98834579 is the probability of obtaining 5 or fewer failures before the 6th success in an indefinite Bernoulli sequence when the probability of success in a single trial is 0.8.

NORMAL CUMULATIVE of x = **E** put in column **C**

The normal cumulative distribution function of the specified value(s) is put in the designated column. The instruction

NORMAL CUMULATIVE of x=**0.5** put in column **36**

would put .69146247 into every row of column 36.

PARETO CUMULATIVE of x = **E** with θ = **E** put in column **C**

The Pareto cumulative distribution function of the specified value(s) with θ is put in the designated column. The second argument, θ must be positive. The instruction

PARETO CUMULATIVE of x=**1.5** with θ=**0.6** put in column **36**

would put .21594732 into every row of column 36.

POISSON CUMULATIVE of x = **E** with mean λ = **E** put in column **C**

The Poisson cumulative distribution function of the specified value(s) with mean lambda is put in the designated column. The value(s) specified by the first argument must be non-negative integers. If the argument is not a column number, it should contain a decimal point. The second argument must be positive. The instruction

POISSON CUMULATIVE ofx=**5.0** with mean λ=**6.0** put in column **36**

would put .44567964 into every row of column 36. The value .44567964 is the probability of obtaining 5 or less counts in a Poisson process with mean 6.0.

T CUMULATIVE of x = **E** with k = **E** degrees of freedom put in column **C**

Student's t cumulative distribution function of the specified value(s) with k degrees of freedom is put in the designated column. The formula for Student's t probability density function is

$$f(x)=\frac{\Gamma\left(\frac{k+1}{2}\right)(1+x^2/k)^{-(k+1)/2}}{\sqrt{\pi k}\,\Gamma\left(\frac{k}{2}\right)} \quad ; \quad -\infty<x<\infty \quad \text{and} \quad k>0$$

where k is degrees of freedom.

The instruction

T CUMULATIVE of x=**0.5** with **6.0** degrees of freedom put in column **36**

would put .6825600 into every row of column 36.

UNIFORM CUMULATIVE of x = **E** put in column **C**

The uniform cumulative distribution function of the specified value(s) is put in the designated column. The instruction

UNIFORM CUMULATIVE of x=**0.5** put in column **36**

would put 0.50000000 into every row of column 36.

WEIBULL CUMULATIVE of x = **E** with θ = **E** put in column **C**

The Weibull cumulative distribution function of the specified value(s) with θ is put in the designated column. The second argument, θ must be positive. The instruction

WEIBULL CUMULATIVE of x=**0.5 with** θ=0.6 put in column **36**

would put .48302148 into every row of column 36.

7.3 Percent Point Functions

BINOMIAL PERCENTILE, CAUCHY PERCENTILE, CHISQUARED PERCENTILE,
DEXPONENTIAL PERCENTILE, EXPONENTIAL PERCENTILE, EXTREME PERCENTILE,
F PERCENTILE, GAMMA PERCENTILE, GEOMETRIC PERCENTILE,
HALFNORMAL PERCENTILE, LAMBDA PERCENTILE, LOGISTIC PERCENTILE,
LOGNORMAL PERCENTILE, NEGBINOMIAL PERCENTILE, NORMAL PERCENTILE,
PARETO PERCENTILE, POISSON PERCENTILE, T PERCENTILE,
UNIFORM PERCENTILE, WEIBULL PERCENTILE

When the second word of a command is **PERCENTILE**, the instruction evaluates the percent point function. Statistical terminology for percent point functions is certainly not standard. In OMNITAB, the percent point function is the inverse of the cumulative distribution function. For example, the instruction

NORMAL CUMULATIVE of x=**0.5** put in column **56**

would put .69146247 into column 56 and the instruction

NORMAL PERCENTILE of x=**0.69146247** put in column **57**

would put 0.5 into column 57. Although the word PERCENTILE is used, the result is actually a proportion: the result is 0.50 rather than 50.0%. Some authors use fractile or quantile instead of percentile.

Percentiles are often needed for a significance level. If such is the case, the value corresponding to the complement of the cumulative distribution function is needed. Hence, the percentile of 1.0 minus the significance level should be specified. Or, in the case of two-sided tests, 1.0 minus one half the significance level. For example, to obtain the 5% critical value for a two-sided t-test with 6.0 degrees of freedom in column 61, use the instruction

T PERCENTILE of x=**0.975** with **6.0** degrees of freedom put in column **61**

Discrete probability distributions, BINOMIAL PERCENTILE, GEOMETRIC PERCENTILE, NEGBINOMIAL PERCENTILE, AND POISSON PERCENTILE, do not necessarily have an exact percentile. The result computed is the integer whose cumulative distribution function value is closest to the specified value. The instruction

POISSON PERCENTILE of x=**0.95** with mean **6.0** put in col **53**

would put 10.0 into column 53. The instructions

POISSON CUMULATIVE of x=**9.0** with mean **6.0** put in col **51**
POISSON CUMULATIVE of x=**10.0** with mean **6.0** put in col **52**

would put .91607598 and .95737907 into columns 51 and 52. Since .95737907 is closer to .95 than .91607598, the percentile is 10.0.

The value(s) specified by the first argument should always be between 0.0 and 1.0. Care should be taken to use only values for which the percent point function is defined and to use proper values for the parameter(s), if any. An arithmetic fault occurs if an improper parameter value is used. The formulas for the probability density functions are given in sections C7.1 and C7.2.

BINOMIAL PERCENTILE of x = **E** with n = **E** and p = **E** put in column **C**

The binomial percent point function of the specified value(s) is put in the designated column. The second argument, n, must be a positive integer and if it is a constant it must contain a decimal point. The third argument, p, must be greater than zero and less than one. The instruction

BINOMIAL PERCENTILE of x = **0.95** with n = **20.0** and p = **0.2** put in column **37**

would put 7.0000000 into every row of column 37. The value 0.95 is approximately the probability of obtaining 7 or less successes in 20 Bernoulli trials in which the probability of success in a single trial is 0.2.

CAUCHY PERCENTILE of x = **E** put in column **C**

The Cauchy percent point function of the specified value(s) is put in the designated column. The instruction

CAUCHY PERCENTILE of x=**0.95** put in column **37**

would put 6.3137509 into every row of column 37.

CHISQUARED PERCENTILE of x = **E** with **E** degrees of freedom put in column **C**

The chisquared percent point function of the specified value(s) with **E** degrees of freedom is put in the designated column. The instruction

CHISQUARED PERCENTILE of x=**0.95** with **6.0** degrees of freedom put in column **37**

would put 12.591587 into every row of column 37.

DEXPONENTIAL PERCENTILE of x = **E** put in column **C**

The double exponential percent point function of the specified value(s) is put in the designated column. The instruction

DEXPONENTIAL PERCENTILE of x=**0.95** put in column **37**

would put 2.3025850 into every row of column 37.

EXPONENTIAL PERCENTILE of x = **E** put in column **C**

The exponential percent point function of the specified value(s) is put in the designated column. The instruction

EXPONENTIAL PERCENTILE of x=**0.95** put in column **37**

would put 2.9957322 into every row of column 37.

EXTREME PERCENTILE of x = **E** put in column **C**

The extreme value type 1 percent point function of the specified value(s) is put in the designated column. The instruction

EXTREME PERCENTILE of x=**0.95** put in column **37**

would put 2.9701951 into every row of column 37.

EXTREME PERCENTILE of x = **E** with γ = **E** put in column **C**

The extreme value type 2 percent point function of the specified value(s) with the specified γ value(s) is put in the designated column. The value(s) specified by the second argument must be positive. The instruction

EXTREME PERCENTILE of x=**0.95** with γ=**0.6** put in column **37**

would put 141.22090 into every row of column 37.

F PERCENTILE of x = **E** with degrees of freedom m = **E**, n = **E** put in column **C**

Snedecor's F percent point function of the specified value(s) with parameters m and n is put in the designated column. The parameters (with a decimal point) must be positive integers. The instruction

F PERCENTILE of x=**0.5** with m=**6.0** and n=**7.0** put in column **37**

would put .98333868 into every row of column 37.

The informative diagnostic message

A VALUE OF DEGREES OF FREEDOM WAS TRUNCATED TO AN INTEGER.

will be printed, if the values of m and/or n are not integers.

GAMMA PERCENTILE of x = **E** with p = **E** put in column **C**

The gamma percent point function of the specified value(s) with p is put in the designated column. The instruction

GAMMA PERCENTILE of x=**0.95** with p=**0.6** put in column **37**

would put 2.15920 into every row of column 37.

GEOMETRIC PERCENTILE of x = **E** with p = **E** put in column **C**

The geometric percent point function of the specified value(s) with p is put in the designated column. The value(s) of p must be between 0.0 and 1.0, exclusively. The instruction

GEOMETRIC PERCENTILE of x=**0.95** with p = **0.2** put in column **37**

would put 12.000000 into every row of column 37. The value 0.95 is the approximate probability of obtaining 12 or less failures before the first success in an indefinite sequence of Bernoulli trials in which the probability of a success in a single trial is 0.2.

HALFNORMAL PERCENTILE of x = **E** put in column **C**

The half-normal percent point function of the specified value(s) is put in the designated column. The instruction

HALFNORMAL PERCENTILE of x=**0.95** put in column **37**

would put 1.9599639 into every row of column 37.

LAMBDA PERCENTILE of x = **E** with θ = **E** put in column **C**

The lambda percent point function of the specified value(s) with θ is put in the designated column. The value(s) of the second argument must be positive. The instruction

LAMBDA PERCENTILE of x=**0.95** with θ=**0.6** put in column **37**

would put 1.3399501 into every row of column 37.

LOGISTIC PERCENTILE of x = **E** put in column **C**

The logistic percent point function of the specified value(s) is put in the designated column. The instruction

LOGISTIC PERCENTILE of x=**0.95** put in column **37**

would put 2.9444389 into every row of column 37.

LOGNORMAL PERCENTILE of x = **E** put in column **C**

The log-normal percent point function of the specified value(s) is put in the designated column. The instruction

LOGNORMAL PERCENTILE of x=**0.95** put in column **37**

would put 5.1802509 into every row of column 37.

NEGBINOMIAL PERCENTILE of x = **E** with n = **E** and p = **E** put in column **C**

The negative binomial percent point function of the specified value(s) is put in the designated column. The value(s) of n must be a positive integer and if it is a constant, it must contain a decimal point. The value(s) of p must be greater than zero and less than one. The instruction

NEGBINOMIAL PERCENTILE of x = **0.95** with n = **6.0** and p = **0.8** put in column **37**

would put 4.0000000 into every row of column 37. The value 0.95 is the approximate probability of obtaining 4 or less failures before 6th success in an indefinite Bernoulli sequence when the probability of success in a single trial is 0.8.

NORMAL PERCENTILE of x = **E** put in column **C**

The normal percent point function of the specified value(s) is put in the designated column. The instruction

NORMAL PERCENTILE of x=**0.95** put in column **37**

would put 1.6448535 into every row of column 37.

PARETO PERCENTILE of x = **E** with θ = **E** put in column **C**

The Pareto percent point function of the specified value(s) with θ is put in the designated column The value(s) of θ must be positive. The instruction

PARETO PERCENTILE of x=**0.95** with θ=**0.6** put in column **37**

would put 147.36125 into every row of column 37.

POISSON PERCENTILE of x = **E** with mean λ = **E** put in column **C**

The Poisson percent point function of the specified value(s) with mean λ is put in the designated column. The value(s) of the mean must be positive. The instruction

POISSON PERCENTILE of x = **0.95** with mean λ= **6.0** put in column **37**

would put 10.000000 into every row of column 37. The value 0.95 is the approximate probability of obtaining 10 or less counts in a Poisson process with mean 6.0.

T PERCENTILE of x = **E** with k = **E** degrees of freedom put in column **C**

Student's t percent point function of the specified value(s) with k degrees of freedom is put in the designated column. The instruction

T PERCENTILE of x=**0.95** with **6.0** degrees of freedom put in column **37**

would put 1.9431802 into every row of column 37.

UNIFORM PERCENTILE of x = **E** put in column **C**

The uniform percent point function of the specified value(s) is put in the designated column. The instruction

UNIFORM PERCENTILE of x=**0.95** put in column **37**

would put 0.950000000 into every row of column 37.

WEIBULL PERCENTILE of x = **E** with θ = **E** put in column **C**

The Weibull percent point function of the specified value(s) with θ is put in the designated column. The value(s) of θ must be positive. The instruction

WEIBULL PERCENTILE of x=**0.95** with θ=**0.6** put in column **37**

would put 6.2254629 into every row of column 37.

7.4 Random Numbers

BETA RANDOM,	**BINOMIAL RANDOM,**	**CAUCHY RANDOM,**
CHISQUARED RANDOM,	**DEXPONENTIAL RANDOM,**	**EXPONENTIAL RANDOM,**
EXTREME RANDOM,	**F RANDOM,**	**GAMMA RANDOM,**
GEOMETRIC RANDOM,	**HALFNORMAL RANDOM,**	**LAMBDA RANDOM,**
LOGISTIC RANDOM,	**LOGNORMAL RANDOM,**	**NEGBINOMIAL RANDOM,**
NORMAL RANDOM,	**PARETO RANDOM,**	**POISSON RANDOM,**
T RANDOM,	**UNIFORM RANDOM,**	**WEIBULL RANDOM**

Instructions are available for generating random numbers from any of 21 different probability distributions. The formulas for the probability density functions are included in sections C7.1 and C7.2.

Each distribution has an optional form with an additional integer argument at the beginning which allows the user to specify the starting point in the random number sequence. The argument **n** specifies that the first (**n**-1) random numbers from the particular probability distribution should be discarded. If NRMAX is 50 and the first argument in the optional form is 21, then the instruction computes the first 70 random numbers from the specified distribution, discards the first 20, and puts the remaining 50 random numbers in the designated column. The additional argument in the optional forms is an integer without a decimal point and not a constant with a decimal point.

BETA RANDOM numbers with parameters a = **K** and b = **K** put in column **C**

BETA RANDOM numbers start number **n** with a = **K** and b = **K** put in column **C**

BINOMIAL RANDOM numbers with n = **K** and p = **K** put in column **C**

BINOMIAL RANDOM numbers, start number **n** with n = **K**, p = **K** put in column **C**

CAUCHY RANDOM numbers put in column **C**

CAUCHY RANDOM numbers starting with number **n** put in column **C**

CHISQUARED RANDOM numbers with degrees of freedom k = **K** put in column **C**

CHISQUARED RANDOM nos. start with no. **n**, degrees of freedom k=**K** put in col. **C**

DEXPONENTIAL RANDOM numbers put in column **C**

DEXPONENTIAL RANDOM numbers start with number **n** put in column **C**

EXPONENTIAL RANDOM numbers put in column **C**

EXPONENTIAL RANDOM numbers start with number **n** put in column **C**

EXTREME RANDOM numbers put in column **C**

The extreme random numbers of type 1 are put in the designated column.

EXTREME RANDOM numbers starting with number **n** put in column **C**

EXTREME RANDOM numbers with parameter **K** put in column **C**

The extreme random numbers of type 2 of the specified parameter are put in the designated column.

EXTREME RANDOM numbers, start with number **n**, parameter **K**, put in column **C**

F RANDOM numbers with degrees of freedom m = **K** and n = **K** put in column **C**

F RANDOM numbers start with number **n**, d.f. m=**K** and n=**K** put in column **C**

GAMMA RANDOM numbers with p = **K** put in column **C**

GAMMA RANDOM numbers starting with number **n** with p = **K** put in column **C**

GEOMETRIC RANDOM numbers with θ = **K** put in column **C**

GEOMETRIC RANDOM numbers starting with number **n** with θ = **K** put in column **C**

HALFNORMAL RANDOM numbers put in column **C**

HALFNORMAL RANDOM numbers starting with number **n** put in column **C**

LAMBDA RANDOM numbers with θ = **K** put in column **C**

LAMBDA RANDOM numbers starting with number **n** with θ = **K** put in column **C**

LOGISTIC RANDOM numbers put in column **C**

LOGISTIC RANDOM numbers starting with number **n** put in column **C**

LOGNORMAL RANDOM numbers put in column **C**

LOGNORMAL RANDOM numbers starting with number **n** put in column **C**

NEGBINOMIAL RANDOM numbers with p = **K** and n = **K** put in column **C**

NEGBINOMIAL RANDOM numbers, start number **n**, p = **K**, n = **K** put in column **C**

NORMAL RANDOM numbers put in column **C**

NORMAL RANDOM numbers starting with number **n** put in column **C**

PARETO RANDOM numbers with θ = **K** put in column **C**

PARETO RANDOM numbers starting with number **n** with θ = **K** put in column **C**

POISSON RANDOM numbers with mean λ = **K** put in column **C**

POISSON RANDOM numbers starting with number **n**, mean λ = **K** put in column **C**

T RANDOM numbers with degrees of freedom k = **K** put in column **C**

T RANDOM numbers starting with number **n** with d.f. k = **K** put in column **C**

UNIFORM RANDOM numbers put in column **C**

The instruction produces numbers pseudo randomly distributed between 0 and 1 in every row down to NRMAX.

UNIFORM RANDOM numbers starting with number **n** put in column **C**

UNIFORM RANDOM numbers starting with seed = **K** put in column **C**

The seed **K** is used as the starting random number. The value of **K** must be written with a decimal point. If the number **K** is equal to or greater than 1.0, 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** may be 1.0 or any value between 1.0 and 8192.0 can be chosen. All numbers greater than 8192.0 are reduced module 8192.

WEIBULL RANDOM numbers with θ = **K** put in column **C**

WEIBULL RANDOM numbers starting with number **n**, θ = **K** put in column **C**

7.5 Probability Plotting

CAUCHY PLOT, DEXPONENTIAL PLOT, EXPONENTIAL PLOT,
EXTREME PLOT, GAMMA PLOT, HALFNORMAL PLOT,
LAMBDA PLOT, LOGISTIC PLOT, LOGNORMAL PLOT,
NORMAL PLOT, PARETO PLOT, POISSON PLOT,
UNIFORM PLOT, WEIBULL PLOT

The probability plot instructions are fully described in section C3.9. They are listed here only for completeness.

CAUCHY PLOT of column **C**

DEXPONENTIAL PLOT of column **C**

EXPONENTIAL PLOT of column **C**

EXTREME PLOT of column **C**

EXTREME PLOT with parameter **K** of column **C**

GAMMA PLOT with parameter **K** of column **C**

HALFNORMAL PLOT of column **C**

LAMBDA PLOT with parameter **K** of column **C**

LOGISTIC PLOT of column **C**

LOGNORMAL PLOT of column **C**

NORMAL PLOT of column **C**

PARETO PLOT with parameter **K** of column **C**

POISSON PLOT with parameter **K** of column **C**

UNIFORM PLOT of column **C**

WEIBULL PLOT with parameter **K** of column **C**

7.6 Random Samples of Digits

SAMPLE WITHR, SAMPLE WITHOUTR

Two instructions are available for obtaining random samples of size **n** with replacement or without replacement from the finite population 1, 2, ..., **N**. In both cases NRMAX is always reset to the sample size **n**.

SAMPLE WITHR of size **n** from population of size **N** put in column **C**

Sampling is random with replacement. The sample size **n** may be larger than the population size **N**. The instructions

SAMPLE WITHR of size **10** from a population of size **2** put in column **7**
SUBTRACT 1.0 from column **7** and put in column **7**

would produce a Bernoulli sequence of zeros and ones of length 10 in column 7.

SAMPLE WITHR start with integer **n** size **n** of population **N** put in column **C**

This instruction is similar to the above instruction except a specified starting integer value is given.

SAMPLE WITHOUTR of size **n** from population of size **N** put in column **C**

Sampling is random without replacement. Hence, the sample size **n** must always be less than or equal to the population size **N**. If the sample size **n** is equal to the population size **N**, a random permutation of the numbers 1, 2, ..., **N** is put in the designated column.

The instruction is very useful whenever randomization is needed. In a designed 3x4 crossed classification experiment, treatments can be randomly assigned to the experimental units by numbering the treatments and experimental units from 1 to 12 and using an instruction such as

SAMPLE WITHOUTR of size **12** from population of size **12** put in column **23**

to obtain a random sample of size 12 to randomly assign the treatments to the experimental units.

SAMPLE WITHOUTR start with **n** size **n** of population **N** put in column **C**

This instruction is same as above except a starting value is specified.

8. NUMERICAL ANALYSIS

The instructions in this section describe mathematical functions frequently encountered in physical and engineering problems.

8.1 Special Integrals

CERF, COSINTEGRAL, EEXPINTEGRAL, EINTEGRAL, ELLIPTICAL FIRST, ELLIPTICAL SECOND, ERROR, EXPINTEGRAL, GAMMA, HCOSINTEGRAL, HSININTEGRAL, NEGEINTEGRAL, SININTEGRAL, STRUVE ONE, STRUVE ZERO

A full description of the complete elliptical integrals and Struve functions may be found in Abramowitz, M.; Stegun, I. A., Handbook of Mathematical Functions, Natl. Bur. Stand. (U.S.) Applied Mathematics Series 55; 1964 June. A description of the computational methods used to calculate the error, complementary error and special integrals function may be found in Stegun, I. A.; Zucker, R., Automatic computing methods for special functions, J. Res. Natl. Bur. Stand. (U.S.). 74B: 211-224; 1970, Stegun, Irene A.; Zucker, Ruth. Automatic computing methods for special functions, Part II, The exponential integral E_n (x). J. Res., Natl. Bur. Stand. (U.S.). 78B; 199-216; 1974 and Stegun, Irene A.; Zucker, Ruth, Automatic computing methods for special functions, Part III, The sine, cosine, exponential integrals and related functions, J. Res. Natl. Bur. Stand. (U.S.). 80B, 291-311; 1976.

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. If $|x|$ is such that erfc(x) is outside of the range, erfc(x) is set equal to zero. 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 2.

COSINTEGRAL of **E** put in column **C**

Computes values of the cosine integral

$$\text{Ci}(x)=\gamma+\ln(x)+ \int_0^x [(\cos(t)-1)/t]dt,\ x>0,$$

where γ=Euler's constant= 0.57721566...

For the numbers 1.0, 2.0, 3.0, and 20.0 in column 1, the instruction

COSINTEGRAL of column **1** put in column **2**

would put the numbers .33740392, .42298083, .11962979, and .044419821 into column 2.

If the value(s) specified by **E** is (are) negative or zero, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR NEGATIVE OR ZERO VALUES.

However, negative values, which produce complex answers, are allowed in the optional form with three arguments described below.

COSINTEGRAL of **E** put real values in column **C** and imaginary values in column **C**

Computes values of the cosine integral when x is negative. The result is complex with real and imaginary parts given by

$$Ci(-x)=Ci(x)-\pi i$$

For the numbers -1.0, -2.0, -5.0 and -20.0 in column 3, the instruction

COSINTEGRAL of col **3** put real part in col **7** and imaginary part in col **8**

would put the numbers .33740392, .42298083, -.19002975, and .044419821 into column 7 and the numbers -3.1415927, -3.1415927, -3.1415927 and -3.1415927 into column 8.

If any value specified by **E** is zero, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR ZERO VALUES.

EEXPINTEGRAL for n of **E** put in column **C**

Computes values of

$$e^{x}E_{n}(x),\ x\geqslant 0,\ n=0,1,\ 2,...,$$

where $E_n(x)$ is described under EXPINTEGRAL; except n must be greater than 1 when x equals zero. For the numbers 1.0, 2.0, 3.0, and 20.0 in column 1, the instruction

EEXPINTEGRAL for n = **2** of column **1** put in column **2**

would put the numbers .40365264, .27734277, .21374878, and .045629090 into column 2.

If the value(s) specified by **E** is (are) negative, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR NEGATIVE VALUES.

If x = 0 and n = 0 or 1, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR ZERO VALUES.

If n is a negative integer, the following fatal error message is printed:

INCORRECT ARGUMENT IN INSTRUCTION.

If n is not an integer, the following fatal error message is given:

IMPROPER TYPE OF ARGUMENT.

EINTEGRAL of **E** put in column **C**

Computes values of the integral

$$Ei(x) = -\int_{-x}^{\infty} (e^{-t}/t)dt,\ x>0.$$

For the numbers 1.0, 2.0, 3.0, and 20.0 in column 1, the instruction

EINTEGRAL of column **1** put in column **2**

would put the numbers 1.8951178, 4.9542344, 9.9338325, 25,615,653. into column 2.

If the value(s) specified by **E** is (are) negative or zero, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR NEGATIVE OR ZERO VALUES.

ELLIPTICAL FIRST integral of x=**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 fault message is given:

X FOR ELLIPTICAL INTEGRALS IS GREATER THAN OR EQUAL TO ONE.

Let column 23 contain the following values .1, .3 and .5, then the instruction

ELLIPTICAL FIRST integral of x in 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 x=**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 fault message is given:

X FOR ELLIPTICAL INTEGRALS IS GREATER THAN OR EQUAL TO ONE.

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

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. If $|x|$ is such that erf(x) is outside of the range, erf(x) is set equal to one. 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.

Values of the normal probability integral

$$\text{Gau}(x)=(1/\sqrt{2\pi})\int_{-\infty}^x e^{-t^2/2}\,dt$$

can be obtained from values of the error function using the relation

$$\text{Gau}(x)=\tfrac{1}{2}[1+\text{erf}(x/\sqrt{2})].$$

EXPINTEGRAL for **n** of **E** put in column **C**

Computes values of the exponential integral

$$E_n(x)=\int_1^\infty t^{-n}e^{-xt}dt;\ n=0, 1, 2, \ldots, x\geqslant 0;$$

except **n** must be greater than 1 when x equals zero.

For the numbers 1.0, 2.0, 3.0 and 20.0 in column 1, the instruction

EXPINTEGRAL for n = **2** of column **1** put in column **2**

would put the numbers .14849551, .037534262, .010641925, and 9.4048564-11 into column 2.

If the value(s) specified by **E** is (are) negative, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR NEGATIVE VALUES.

If x = 0 and **n** = 0 or 1, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR ZERO VALUES.

If **n** is a negative integer, the following fatal error message is printed:

INCORRECT ARGUMENT IN INSTRUCTION.

If **n** is not an integer, the following fatal error message is printed:

IMPROPER TYPE OF ARGUMENT.

GAMMA function of **E** put in column **C**

Puts the incomplete gamma function of the specified value(s) in the designated column. For any positive integer n, the incomplete gamma function of n equals (n−1) factorial. Gamma of zero is equal to one. The

incomplete gamma function is not defined for negative integers. For the numbers 4.0, 1.0, 1.3, 1.7, 0.7, -0.7, and -1.3 in column 14, the instruction

GAMMA of column **14** put in column **26**

would put the numbers 6.0000000, 1.0000000, 0.89747070, 0.90863873, 1.2980554, -4.2736699, and 3.3283470 in column 26. Note, $\Gamma(1.3) = (0.3)(-0.7)\Gamma(-0.7)$ and $\Gamma(1.7) = 0.7(-0.3)(-1.3)\Gamma(-1.3)$.

HCOSINTEGRAL of **E** put in column **C**

Computes values of the hyperbolic cosine integral

$$\text{Chi}(x)=\gamma+\ln(x)+\int_0^x[(\cosh(t)-1)/t]dt,\ x>0,$$

where γ = Euler's constant = 0.57721566...

For the numbers 1.0, 2.0, 3.0, and 20.0 in column 1, the instruction

HCOSINTEGRAL of column **1** put in column **2**

would put the numbers .83786694, 2.4526669, 4.9603921, and 12,807,826. into column 2.

If the value(s) specified by **E** is (are) negative or zero, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR NEGATIVE OR ZERO VALUES.

However, negative values, which produce complex answers, are allowed in the optional form with three arguments described below.

HCOSINTEGRAL of **E** put real part in column **C** and imaginary part in column **C**

Computes values of the hyperbolic cosine integral when x is negative. The result is complex with real and imaginary parts given by

$$\text{Chi}(-x)=\text{Chi}(x)-\pi i.$$

For the numbers -1.0, -2.0, -5.0 and -20.0 in column 3, the instruction

HCOSINTEGRAL of col **3** put real part in col **7** and imaginary part in col **8**

would put the numbers .83786694, 2.4526669, 20.092063, and 12,807,826. into column 7 and the numbers -3.1415927, -3.1415927, -3.1415927 and -3.1415927 into column 8.

If any x-value is zero, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR ZERO VALUES.

HSININTEGRAL of **E** put in column **C**

Computes values of the hyperbolic cosine integral

$$\text{Shi}(x)=\int_0^x[\sinh(t)/t]dt.$$

When x is negative, the integral is evaluated as Shi(−x) = −Shi(x). For the numbers 1.0, -1.0, 2.0, 3.0, and 20.0 in column 1, the instruction

HSININTEGRAL of column **1** put in column **2**

would put the numbers 1.0572509, −1.0572509, 2.5015674, 4.9734405, and 12,807,826. into column 2.

NEGEINTEGRAL of **E** put in column **C**

Computes values of the integral

$$e^{-x}Ei(x),\ x>0,$$

where Ei(x) is described above under EINTEGRAL. For the numbers 1.0, 2.0, 3.0, and 20.0 in column 1, the instruction

NEGEINTEGRAL of column **1** put in column **2**

would put the numbers .69717488, .67048271, .49457640, and .052797795 into column 2.

If the value(s) specified by **E** is (are) not positive, the following arithmetic fault message is printed:

FUNCTION NOT DEFINED FOR NEGATIVE OR ZERO VALUES.

SININTEGRAL of **E** put in column **C**

Computes values of the sine integral

$$Si(x)=\int_0^x [\sin(t)/t]dt.$$

When x is negative, the integral is evaluated as Si(−x) = −Si(x).
For the numbers 1.0, −1.0, 2.0, 3.0, and 20.0 in column 1, the instruction

SININTEGRAL of column **1** put in column **2**

would put the numbers .94608307, −.94608307, 1.6054130, 1.8486525, and 1.5482417 into column 2.

STRUVE ONE integral of **E** put in column **C**

STRUVE ONE evaluates for real, positive values of x, the Struve function

$$H_1(x)=(2/\pi)[x^2/(1^2.3)-x^4/(1^2.3^2.5)+x^6/(1^2.3^2.5^2.7)-\dots].$$

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, .86315421 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

$$H_0(x)=(2/\pi)[x-x^3/(1^2.3^2)+x^5/(1^2.3^2.5^2)-\ldots\ ].$$

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.

8.2 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 second and third 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(S) OUTSIDE (n)COLUMN WORKSHEET.

All the polynomials are computed using the appropriate recursion formulas. The equations for these recursion formulas may be found in Abramowitz, M.; Stegun, I. A., Handbook of Mathematical Functions, Natl. Bur. Stand. (U.S.) Appl. Math. Ser. 55; 1964 June.

The examples below assume that column 11 contains .5, 1.0, 1.5 and 2.0.

HERMITE polynomial of order **n** of column **C** put in column **C** and successive columns

HERMITE evaluates the polynomial

$$H_n(x)=n!\sum_{m=0}^{N}(-1)^m(2x)^{n-2m}/(m!(n-2m)!),$$

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.

Row/Column	$H_1(x)$ *2*	$H_2(x)$ *3*	$H_3(x)$ *4*	$H_4(x)$ *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 column **C** put in column **C** and successive columns

LAGUERRE evaluates the polynomial

$$L_n(x) = \sum_{m=0}^{n} (-1)^m K x^m / m!,$$

where $K = \binom{n}{m}$. The instruction

LAGUERRE polynomial of order **3** of col **11** put in col **7** and successive columns

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 column **C** put in column **C** and successive columns

This instruction computes the polynomial

$$P_n(x) = (1/2^n) \sum_{m=0}^{N} (-1)^m K x^{n-2m} T,$$

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 successive 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.

	$NL_1(x)$	$NL_2(x)$	$NL_3(x)$
Row/Column	*2*	*3*	*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 column **C** put in column **C** and successive cols

TCHEBYSHEV computes the Chebyshev polynomial of the first kind

$$T_n(x)=(n/2)\sum_{m=0}^{N}(-1)^m(2x)^{n-2m}(n-m-1)!/(m!(n-2m)!),$$

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.

	T (x)	$T_2(x)$
Row/Column	*50*	*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 successive cols

UCHEBYSHEV evaluates the Chebyshev polynomial of the second kind

$$U_n(x)=\sum_{m=0}^{N}(-1)^m(2x)^{n-2m}(n-m)!/(m!(n-2m)!),$$

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:

	$U_1(x)$	$U_2(x)$	$U_3(x)$	$U_4(x)$
Row/column	*23*	*24*	*25*	*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

8.3 Differences

DIFFERENCES, DIVDIFFERENCES, SDIFFERENCES, SDIVDIFFERENCES

Differences and, in particular, divided differences are used extensively in numerical analysis for studying the functional form of data, interpolation, inverse interpolation, numerical integration and numerical differentiation. A description of differences may be found in Hamming, R. W., Numerical Methods For Scientists and Engineers. New York: MacGraw-Hill Book Company; 1962 in particular 3-13 and 98-101 and Hildebrand, F. B., Introduction to Numerical Analysis, New York: McGraw-Hill Book Company; 1956, Chapter 2 in particular.

DIFFERENCES of y in column **C**

The DIFFERENCES instruction automatically prints a table of the 1st, 2nd, ..., 7th differences; providing NRMAX is at least eight. If NRMAX is less than eight, the instruction prints the maximum number of differences; namely (NRMAX-1). The numbers in the first column are printed on every other line to produce a more readable table.

The instructions

```
SET Y IN COLUMN 2
0,     .20134,        .30452,      .52110
DIFFERENCES  OF COLUMN 2
```

would produce the following printing:

```
COLUMN      2
        Y            1ST              2ND              3RD
   0.
                  .20134000
    .20134000                  -.098159997
                  .10318000                        .21156000
    .30452000                   .11340000
                  .21658000
    .52110000
```

The numbers in the designated column appear under the title Y. The appropriate column heading appears above Y. Any entry in the table on the right of the first column is formed by subtracting the number just above it on the left from the number just below it on the left. For example, .21685000−.10318000 = .11340000.

DIFFERENCES of y in column **C** put in columns **C, C ,..., C**

Similar to the above form, except differences are stored in the worksheet in consecutive rows starting with row 1. The number of arguments determines the number of differences which are stored. The first differences

are stored in the column designated by the 2nd argument, the second differences are stored in the column designated by the 3rd argument, etc. and the kth differences are stored in the column designated by the last argument. The value of k must be less than NRMAX. The appropriate column headings are printed above the titles 1ST, 2ND, etc.

DIVDIFFERENCES for x in column **C** and y in column **C**

The DIVDIFFERENCES instruction automatically prints a table of the 1st, 2nd, ..., 6th divided differences; providing NRMAX is at least seven. If NRMAX is less than seven, the instruction prints the maximum number of divided differences, namely (NRMAX-1). The numbers in the first two columns for x and y are printed on every other line to produce a more readable table.

The instructions

```
SET X IN COLUMN 1
0, .2, .3, .5
SET Y IN COLUMN 2
0 .20134, .30452, .52110
DIVDIFFERENCES FOR X IN COLUMN 1 AND Y IN COLUMN 2
```

would produce the following printing:

COLUMN 1	COLUMN 2			
X	Y	1ST	2ND	3RD
0.	0.			
		1.0067000		
.20000000	.20134000		.083666841	
		1.0318000		.17333259
.30000000	.30452000		.17033313	
		1.0829000		
.50000000	.52110000			

The first two columns of the table give the values of x and y in the columns designated by the first two arguments of the instruction. The jth entry in the column for the ith order divided differences is computed as:

$$f_{i,j}=(f_{i-1,j+1}-f_{i-1,j})/(x_{i+j}-x_j), \text{ with } f_{0,j}=y_j .$$

For example, with $i=3$ and $j=1$,

$$\begin{aligned} f_{3,1} &= (f_{2,2}-f_{2,1})/(x_4-x_1), \\ &= (.17033313-.083666841)/(0.5-0.0), \\ &= .17333259. \end{aligned}$$

DIVDIFFERENCES for x in column **C** and y in column **C** put in columns **C**, **C** ,..., **C**

Similar to the above form, except divided differences are stored in the worksheet. The number of arguments determines the number of divided differences which are stored. The first divided differences are stored in the column designated by the 3rd argument, the second divided differences are stored in the column designated by the 4th argument, etc. and the kth divided differences are stored in the column designated by the last argument. The value of k must be less than (NRMAX−1). The appropriate column headings are printed above the titles 1ST, 2ND, etc.

SDIFFERENCES of y in column **C** put in columns **C**, **C** ,..., **C**

Same as the optional form of DIFFERENCE instruction, except the automatic printing is suppressed. If the number of arguments is one, the following informative diagnostic is printed:

THE INSTRUCTION WAS IGNORED BECAUSE...
COMMAND BEGINS WITH S AND STORAGE MUST BE REQUESTED.

SDIVDIFFERENCES for x in **C** and y in **C** put in columns **C**, **C** ,..., **C**

Same as the optional form of DIVDIFFERENCE instruction, except the automatic printing is suppressed. If the number of arguments is two, the following informative diagnostic is printed:

THE INSTRUCTION WAS IGNORED BECAUSE...
COMMAND BEGINS WITH S AND STORAGE MUST BE REQUESTED.

8.4 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.

If no values are located for any of the three instructions, the value of NRMAX is not changed and the following informative diagnostic message is printed:

ITERATION DID NOT FIND ANY ROOTS.

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):

(n) ROW WORKSHEET IS TOO SHORT TO ACCOMMODATE
ALL THE VALUES GENERATED BY THIS INSTRUCTION.

NRMAX may be either increased or decreased and informative diagnostic is printed.

ISETUP x in **C**, y in **C**, desired y in **C**, put in column **C** and next three columns

This instruction is used initially to set up values needed by the instruction ITERATE and 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). The instruction looks for two adjacent values of y, y_j and y_{j+1}, in column C2 which bracket the desired value, Y, in column C3, i.e., $y<Y_i<y_j$, 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, $\bar{Y}_i$, 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 $\Delta=(x_{j+1}-x_j)/4$.

NRMAX is set equal to 5 times the number of values located. 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 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 CONSTANT.

The following set of instructions uses ISETUP to find the values of x, approximating X, for which Y = cos(X) = .6, .7 and .8.

```
GENERATE X EQUAL TO 0 (.1) 1.0 PUT IN COL 1
GENERATE Y EQUAL TO .6 (.1) .8 IN COL 14
RESET NRMAX TO 11
COS OF X IN COL 1 PUT RESULT IN COL 12
PRINT COLS 1 12 14 WITH 6.0 SIGNIFICANT DIGITS
ISETUP X IN COL 1, Y IN COL 12 DESIRED Y IN COL 14 START STORING IN COL 1
TITLE1 RESULTS OF INSTRUCTION ISETUP
PRINT COLS 1***4 WITH 6.0 SIGNIFICANT DIGITS
```

The results of using this set of instructions are:

```
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
```

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 12 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**

This instruction is similar to the ISETUP instruction except it locates all sets of values for the desired value y, the third argument of the instruction, which must be a constant with a decimal point. 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. NRMAX is set equal to 5 times the number of sets located.

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.

```
NOTE SOLVE SIN(X)=0.0 FOR X BETWEEN 1 AND 5
GENERATE TRIAL VALUES OF X=1. (.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 AND 31
2./ISOLATE X IN COL 1 FOR Y IN COL 2 VALUE=0.0 PUT IN COLS 1 AND 31
PERFORM INSTRUCTIONS 1 THRU 2 5 TIMES
SPACE
PRINT NOTE
NPRINT COLS 1, 2 AND 31
```

The results of using the above set of instructions are:

```
SOLVE SIN (X) = 0.0 FOR X BETWEEN 1 AND 5

     COLUMN 1        COLUMN 2        COLUMN 31
      1.0000000       .84147099
      1.5000000       .99749499
      2.0000000       .90929743
      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      -.014656822
      3.1875000      -.045891223
      3.2187500      -.077080813
      3.2500000      -.10819513

     COLUMN 1        COLUMN 2        COLUMN 31
      3.1250000     .016591892        3.1406250
      3.1328125     .0087800408
      3.1406250     .00096765344
      3.1484375    -.0068447929
      3.1562500    -.014656822

     COLUMN 1        COLUMN 2        COLUMN 31
      3.1406250     .00096765344      3.1445313
      3.1425781    -.00098547126
      3.1445313    -.0029385922
      3.1464844    -.0048917019
      3.1484375    -.0068447929

     COLUMN 1        COLUMN 2        COLUMN 31
      3.1406250     .00096765344      3.1416016
      3.1411133    -.00098547126
      3.1416016    -.0029385922
      3.1420898    -.0048917019
      3.1425781    -.0068447929
```

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, = 3.1415927, each time ISOLATE is used. Column 1 contains the 5 nearest values of x for which sin(x) = 0.0 ($1 \leqslant x < 5$) 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 columns **C** and **C**

This instruction is the same as the one above, except the (**p**+2) nearest values of x are located rather than five, and NRMAX is set equal to (**p**+2) times the number of sets located.

ITERATE x in **C** y in **C** desired y in column **C** put in **C** and next three columns

The ITERATE instruction is used in the repeat mode and performs exactly like ISETUP. ITERATE is used to refine the values of x until the specified absolute tolerance is met. When the following instructions are added to the instructions in the example for ISETUP,

1/COS OF X IN COL 1 PUT Y IN COL 12
2/ITERATE X IN COL 1, Y IN COL 12, DESIRED Y IN COL 14 START STORING IN COL 1
3/IFEQ IF COL 3 IS EQUAL TO COL 4 WITH TOLERANCE OF .0001 TERMINATE PERFORM
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

the results after the tolerance has been satisfied are:

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

The values in column 3 must be equal to the values in column 4 within the tolerance of .0001.

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.

8.5 Analysis

HARMONIC, INTERPOLATE, MAXMIN, SOLVE

HARMONIC analysis of y in column **C** for **n** ordinates, put coefficients in column **C**

HARMONIC evaluates the coefficients A and B for the periodic function

$$y_i = A_0 + \sum_{i=1}^{r} A_k \cos(k\theta_i) + \sum_{i=1}^{s} B_k \sin(k\theta_i),$$

where i=1,2,...,n and 2 n 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

$$A_0, A_1, \ldots, A_{n/2}, B_1, B_2, \ldots, B_{(n-2)/2}.$$

If n is odd, r=s and the coefficients are stored in the sequence

$$A_0, A_1, \ldots, A_{(n-1)/2}, B_1, B_2, \ldots, B_{(n-1)/2}.$$

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:

Row/Column	y *1*	A and B coefficients *2*
1	9.3	9.2166667
2	15.0	−6.9027765
3	17.4	3.4499999
4	23.0	3.3000000
5	37.0	−.61666668
6	31.0	.60277666
7	15.3	.24999998
8	4.0	21.089590
9	−8.0	−2.8001487
10	−13.2	1.9666667
11	−14.2	1.0680981
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**, points **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**+3**p**+**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 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 .24740158 and .34289404 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, $y = ax+bx+c$, 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:

NO EXTREMA WERE FOUND.

If an apparent extremum is found where the x values are identical, the extremum is ignored and the following informative diagnostic is given:

A TRIAD OF X'S WITH AT LEAST TWO IDENTICAL VALUES WAS FOUND AND IGNORED.

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	3.6000000+02	1.0000002		

The command EXTREMA is a synonym for MAXMIN.

SOLVE linear equations with coefficients in **R,C** size **rxc** constants in **C**, put solution in **C**

SOLVE provides the solution of a set of simultaneous linear equations, Ax = y, where A is a square matrix of coefficients, y is a vector of constants and x is the solution vector. The first four arguments specify the

beginning location of the matrix A (a row and column number), and the size of the matrix (number of rows and columns in the matrix). The beginning location of the matrix is the location of the number in the upper left-hand corner of the matrix. The matrix must be in consecutive rows and columns. The 3rd and 4th arguments specifying the size must be equal. The fifth argument, a column number, contains a vector of constants, y. The results, x, are stored in the column of the last argument.

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 (value).

For a complete description of the meaning of the error bound, see the instruction MINVERT in section C11.5.

If **r**≠**c**, the following informative diagnostic will be given:

NO. OF ROWS NOT = TO NO. COLS. LARGEST SQUARE MATRIX WAS USED.

Furthermore, if **r** and/or **c** are inadvertently too large, the following fatal error message is given:

ARRAY OR MATRIX OUTSIDE (n) ROW x (n) COLUMN WORKSHEET.

In order to solve a set of equations a large amount of scratch area is needed. If 2(**r**+4) 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:

$$A=\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \quad y=\begin{bmatrix} 0.5 \\ 1.0 \\ 1.5 \\ 2.0 \\ 2.5 \\ 3.0 \\ 3.5 \\ 4.0 \\ 4.5 \\ 5.0 \end{bmatrix}$$

where the value of a (the upper left-hand corner) is in row 1 of column 5 and y is in column 1, 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
9.9999999
19.500000
28.000000
35.000000
40.000000
42.500000
42.000000
38.000000
30.000000
17.500000

8.6 Integration

GAUSS QUADRATURE

GAUSS QUADRATURE n=**K** points, from a=**K** to b=**K**, put x in **C**, weights in **C**

GAUSS QUADRATURE generates the n abscissas, x_j, and the corresponding weight coefficients, w_i, for the Gaussian quadrature formula

$$\int_a^b f(x)dx = \sum_{i=1}^{n} w_i f(x_i),$$

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

9. REPEATED USE OF INSTRUCTIONS

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 C9.1 and C9.2 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.

9.1 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.

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 can not be stored, but a PERFORM instruction may 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. If the instruction number is written with a decimal point, it will not only create a fatal error here, but also 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 stored 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 C9.3 and C9.4. The 1st argument must be less than or equal to the 2nd argument. The maximum number of PERFORM instructions that can be nested is eight. 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 the following fatal error will occur:

INSTRUCTION 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

NUMBERED PERFORM INSTRUCTION EXECUTES ITSELF.

For further details see section B2.2 on the use of PERFORM. The commands EXECUTE and REPEAT are commonly used synonyms for PERFORM.

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.

9.2 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 and the original arguments of the modified instruction are not retained. The instruction being incremented must be a stored (numbered) instruction. 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.

If asterisks are used to define an argument (see sec. B1.7) 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**

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.

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.

An INCREMENT instruction can increment another INCREMENT instruction but can not 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.

RESTORE instruction **N** to , **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.

9.3 Branching, Three Arguments

COMPARE, IFEQ, IFNE

The instructions described in sections C9.3 and C9.4 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 C9.4 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

RESET NRMAX = **1**
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 an .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

$$|\text{ARG1}-\text{ARG2}|<|\text{ARG3}|.$$

Also, if ARG1=0 or ARG2=0, the following arithmetic fault message is given:

ONE OF THE VALUES COMPARED IS ZERO. ABSOLUTE TOLERANCE USED.

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

3/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

5/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.

9.4 Branching, Two Arguments

IFEQ, IFGE, IFGT, IFLE, IFLT, IFNE

These six instructions each have exactly two arguments and may only be used as stored instructions in the repeat mode. 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 C9.3. 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.

10. 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 C10.1 and C10.2 include examples 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:

ARRAY OR MATRIX OUTSIDE (n) ROW (n) COLUMN WORKSHEET.

The instruction ALABEL may be used to label arrays. For a complete description of the ALABEL instruction see section C1.3.

10.1 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 **rxc**, 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 **rxc**, 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 **rxc** of the second array is omitted from the instruction. The third form (7 arguments) is used when the second array is a constant or column.

The ADIVIDE, AMULTIPLY and ARAISE instructions in this section can produce the same type of arithmetic faults as the analagous instructions of section C4.1. See section C4.1 and section B3.3 for further details.

AADD the array **R,C** size **rxc** to array **R,C** size **rxc** 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 **rxc** to array **R,C** and put in **R,C**

Same as above, except the row and column size of the second array are omitted.

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 **rxc** by array **R,C** size **rxc** 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 **rxc** 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 **rxc** 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 **rxc** by **R,C** size **rxc** 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 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	4.0	0.0	16.0
5	18.0	−8.0	−8.0

AMULTIPLY array in **R,C** size **rxc** 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 **rxc** 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 **rxc** to array **R,C** size **rxc** 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 **rxc** 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 **rxc** 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 **rxc** minus **R,C** size **rxc** 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 sec. C4.1). 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 **rxc** 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 **rxc** 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

10.2 Data Manipulation

ADEFINE, AERASE, AMOVE, ATRANSPOSE

The instructions in this section provide means of manipulating arrays for data analysis.

ADEFINE the array in **R,C** of size **rxc** 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 **rxc**

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.

AMOVE the array in **R,C** of size **rxc** 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 C5.2, 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 **rxc** 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 fifth 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

10.3 Summarization

AAVERAGE, ACOALESCE

These two instructions are similar. The instruction ACOALESCE computes certain sums as described, whereas the instruction AAVERAGE computes the corresponding averages.

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 WAS NOT FOUND.

The fourth argument **c**, (fifth argument in optional form), must be greater than one; otherwise the instruction is ignored and an informative diagnostic is printed to that effect.

Examples in this section are based on the following array A 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 A in **R,C** size **rxc** put array B in **R,C**

The m unique values of the first column of the first array A are put into the first column of the second array B, where m < r (third argument). The values put in the 2nd column through the cth column of the second array B are as follows

$$b_{ij} = \sum_{n=1}^{r} p_n a_{nj} / \sum_{n=1}^{r} p_n, \quad i=1,...,m; \quad j=2,...,c$$

where

$$p_n = 1, \text{ if } b_{i1}=a_{k1} \text{ and}$$
$$p_n = 0, \text{ if } b_{i1}\neq a_{k1} \qquad i=1,...,n; \quad k=1,...,r.$$

The instruction

AAVERAGE on first col of array A in **21,11** size **5x4** and put **B** in **41,31**

would yield array B

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 contains 3 unique values of column 11 of array A. The value in row 41 of column 33 is the $b_{3,1}$th element and is evaluated as follows

$$b_{3,1}=[1.0(1.0)+0.0(1.0)+1.0(0.0)+0.0(2.0)+0.0(3.0)]/[1.0+0.0+1.0+0.0+0.0]=1.0/2.0=.5$$

The other values are obtained in a similar manner.

AAVERAGE on **K** in first column of array A in **R,C** size **rxc** put B in **R,C**

Similar to the above instruction, except the second array B has only (1) row and **c** columns. The first column contains the constant **K**.

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 column of array A in **R,C** size **rxc** put array B in **R,C**

The instruction is similar to the AAVERAGE instruction. The m unique values of the first column of the first array A are put in the first column of the second array B, where $0<m<r$ (third argument). The values put in the 2nd column through the cth column of the second array B are as follows

$$b_{ij} = \sum_{n=1}^{r} p_n a_{nj} \qquad i=1,...,m; \quad j=2,...,c$$

where

$$p_n = 1, \text{ if } b_{i,1}=a_{k,1} \text{ and}$$
$$p_n = 0, \text{ if } b_{i,1}\neq a_{k,1}, \qquad \text{for} \quad k=1,...,r.$$

a_{ik} are elements of array A (the first array) and b_{ij} are elements of array B. The remaining rows are constructed in a similar way. The instruction

ACOALESCE on first col of array A in **21,11** size **5x4**, put array B 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 value in row 41 of column 32 is the $b_{2,1}$th element and is compiled as

$$b_{2,1} = 1.0(0.0)+0.0(2.0)+1.0(2.0)+0.0(1.0)+0.0(1.0) = 0.0+0.0+2.0+0.0+0.0 = 2.0$$

ACOALESCE on **K** in first column of array A in **R,C** size **rxc** put B in **R,C**

Similar to above instruction, except the size of the second array B is 1 row and **c** columns. The first column of array B has the value of **K**. The instruction

ACOALESCE on **1.0** in first col of array A in **21,11** of size **5x4** put array B 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.

10.4 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 fifth form, is given on page 290. 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 C6.6. Items 15 and 16 are the *within* sums of squares and should not be confused with the *between* sums of squares printed by TWOWAY. The WIDTH and/or BRIEF instructions have no effect on the automatic printout. The maximum number of characters per line is 72.

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:

```
THE INSTRUCTION WAS IGNORED BECAUSE...
   COMMAND BEGINS WITH S AND STORAGE MUST BE REQUESTED.
```

APROPERTIES of the array in **R,C** of size **rxc**

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 **rxc** 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 **rxc**, put column averages in **R,C**

The **c** column averages are put in the *row* designated by the last pair of arguments.

APROPERTIES of **R,C** size **rxc**, put properties in **C**, column averages 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 **rxc**, put column averages in **R,C**, row averages **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 **rxc**, in **C**, averages 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. For the following set of instructions,

```
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 ARRAY IN ROW 1 COLUMN 1 SIZE 4X5 PUT ARRAY IN ROW 3 COLUMN 7
APROPERTIES OF ARRAY IN 3,7 SIZE 4X5 PUT COL AVE's IN 7,7 ROW AVE'S In 3,12
SPACE 2
APRINT ARRAY IN ROW 3 OF COLUMN 7 OF SIZE 5X6
```

the output is:

```
PROPERTIES OF THE 4X5 ARRAY STARTING AT (3,7)
   GENERAL
       1   TRACE (  4 VALUES USED)                        0.
       2   TRACE NUMBER TWO                            -100.00000

       3   MAXIMUM NUMBER                               10.000000
       4   MINUMUM NUMBER                               -9.0000000
       5   MAXIMUM NUMBER IN ABSOLUTE VALUE             10.000000
       6   MINUMUM NUMBER IN ABSOLUTE VALUE              0.
       7   MINIMUM NON-ZERO ABSOLUTE VALUE               1.0000000

       8   NUMBER OF POSITIVE NUMBERS                   10
       9   NUMBER OF ZERO NUMBERS                        1
      10   NUMBER OF NEGATIVE NUMBERS                    9

      11   SUM OF TERMS                                 10.000000
      12   AVERAGE                                        .50000000
      13   SUM OF SQUARES                              670.00000
      14   SUM OF SQUARES ABOUT MEAN                   665.00000
      15   WITHIN ROWS SUM OF SQUARES                   40.000000
      16   WITHIN COLS SUM OF SQUARES                  625.00000
      17   SUM OF ABSOLUTE VALUES                      100.00000
      18   AVERAGE OF ABSOLUTE VALUES                    5.0000000

-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.

                                    *****
```

SAPROPERTIES of the array in **R,C** of size **rxc** put in column **C**

SAPROPERTIES of array in **R,C** size **rxc**, put column averages in **R,C**

SAPROPERTIES of **R,C** size **rxc**, put properties in **C**, column averages in **R,C**

SAPROPERTIES of **R,C** size **rxc**, put column averages in **R,C**, row averages **R,C**

SAPROPERTIES array **R,C** size **rxc**, in **C**, averages in **R,C** and **R,C**

10.5 Printing

APRINT, APRINT "L"

These two instructions are listed here for completeness, but they are described in section C2.6.

APRINT the array in **R,C** of size **rxc**

APRINT "L" format, the array in **R,C** of size **rxc**

10.6 Matrix Synonyms

No knowledge of matrix algebra is required to use the instructions described in this section. 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 C11. Each matrix equivalent of an array instruction is listed below. (See section B1.12 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**

11. 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. Each command (except SMPROPERTIES) begins with the letter M to denote a matrix operation.

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), except in MMATVEC. The beginning location of a matrix is the location of the number in the upper left-hand corner of the matrix.

The four arguments designating a matrix in the worksheet are:

R = the row number of the value in the upper left hand corner of the matrix
C = the column number of the value in the upper left hand corner of the matrix
r = the number of rows in the matrix
c = the number of columns in the matrix

Some of the instructions have additional arguments which may designate a second or third matrix.

Care should be used with instructions for matrix operations to avoid designating a matrix which is partially outside the worksheet. In most instructions, if the arguments specify a matrix partially outside the worksheet, the following fatal error message appears:

ARRAY OR MARIX OUTSIDE (n) ROW x (n) COLUMN WORKSHEET.

11.1 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 **rxc** to have all elements equal to **K**

All the **rxc** 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), 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 **rxc** equal to **E** on the diagonal

---

The main diagonal elements of the **rxc** 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 **rxc**

Every number in the **rxc** 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 **rxc**

All the non-diagonal elements of the specified matrix in row **R** of column **C** of size **rxc** are set equal to zero. The main diagonal elements are set equal to one. If the size **rxc** 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

11.2 Moving Operations

MMATVEC, MMOVE, MTRANSPOSE, MVECDIAGONAL, MVECMAT

These commands are very useful for data manipulation.

MMATVEC make column **C** into the matrix in **R,C** of size **rxc**, row by row

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 **rxc** 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 col **21** into a matrix in location **6,3** size **2x3**, rowwise

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 column vector in **R,C** into matrix in **R,C** size **rxc**

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 **rxc** 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 C5.2, 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 **rxc** into matrix in **R,C**

The transpose of the matrix beginning in row **R** of column **C** of size **rxc** 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 **cxr**. 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
8				3.0	10.0

MVECDIAGONAL the diagonal of matrix in **R,C** size **rxc** is put in column **C**

The elements on the main diagonal of the matrix beginning in row **R** of column **C** of size **rxc** 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 diagonal of matrix in **R,C** size **rxc**, put in column vector **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.

MVECMAT vectorize row by row matrix in **R,C** size **rxc** into column **C**

The elements in the matrix beginning in row **R** of column **C** of size **rxc** 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 **rxc**, 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 **rxc**, 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.

11.3 Matrix Arithmetic

MADD, MKRONECKER, MMULTIPLY, MRAISE, MSCALAR, MSUBTRACT

For the instructions in this section, define the matrix $A = (a_{i,j})$, where a is the element in the ith row and jth column. Other matrices, such as B and C, are defined in a similar manner.

MADD A in **R,C** size **rxc** to B in **R,C** size **rxc** put C 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

$$c_{ij}=a_{ij}+b_{ij}, \text{ where } i=1,2,...,r \text{ and } j=1,2,...,c.$$

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 A in **R,C** of size **rxc** to B in **R,C** put **C** 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 of A in **R,C** size **rxc**, by B in **R,C** size **rxc**, put C in **R,C**

The Kronecker product $A \circledcirc B = C$ 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 mxr rows by nxs columns. The matrix C may be partioned as follows:

$$C = A \circledcirc B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}$$

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 A in **R,C** size **rxc** by B in **R,C** size **rxc** 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:

$$c_{ij} = \sum_{u=1}^{n} a_{iu} b_{uj}, \text{ where } i = 1,2,\dots,r \text{ (3rd argument) and } j = 1,2,\dots,c \text{ (8th argument).}$$

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 computed as a double precision number and the summation is also performed in double precision arithmetic. This method of computation is used to provide greater accuracy. For large matrices it will increase the computing time appreciably.

MRAISE the matrix A in **R,C** of size **rxc** 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 A in **R,C** of size **rxc** by s = **K**, put matrix Y in **R,C**

The scalar product Y =sA is computed, where A is the first matrix and s is the scalar **K**. The matrices Y and A have the same size, **rxc**. The elements of Y are

$$y_{ij} = sa_{ij}, \text{ where } i = 1,2,...,r \text{ and } j = 1,2,...,c.$$

MSCALAR is actually a special case of AMULTIPLY (sec C10.1). 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 A in **R,C** size **rxc**, minus B in **R,C** size **rxc**, put C in **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, **rxc**. The elements of C are

$$c_{ij}=a_{ij}-b_{ij}\text{ , where } i=1,2,...,r \text{ and } j=1,2,...,c.$$

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 C4.1. 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 **rxc** minus matrix **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 assumed to be the same as the size of the first matrix.

11.4 Special Matrix Multiplication

M(AD), M(AV), M(DA), M(V'A), M(X'X), M(XX'), M(X'AX), M(XAX')

All 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 80 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 C11.3.

M(AD) A in **R,C** size **rxc** times diagonal of matrix D in **C**, put Y in **R,C**

The first matrix A is *post*multiplied 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

$$y_{ij}=a_{ij}d_i, \text{ where } i=1,2,...,r \text{ and } j=1,2,...,c.$$

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) A in **R,C** size **rxc** times vector V in column **C** put vector Y in column **C**

The first matrix, A, is *post*multiplied by the vector V in column **C** to form the column vector Y = AV, with elements

$$y_i=\sum_{u=1}^{c} v_u a_{iu}, \text{ where } i=1,2,...,r.$$

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) A in **R,C** size **rxc** multiply by V in **C** put vector Y in **R,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) A in **R,C** size **rxc** premultiply by diagonal of D in **C**, put Y in **R,C**

The first matrix, A, is *pre*multiplied 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

$$y_{ij}=d_i a_{ij}\text{, where } i=1,2,...,r \text{ and } j=1,2,...,c.$$

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) A in **R,C** size **rxc** premultiply by vector V in column **C**, put row vector Y in **R**

The matrix A is *pre*multiplied by the transpose of a column vector, V, to form the row vector Y = V′A, with elements

$$y_j=\sum_{u=1}^{r} v_u a_{uj}\text{, where } j = 1,2,...,c.$$

The result is put in row **R**, 6th argument, of the first **c** columns of the worksheet. 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 = V̀A*	
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) A in **R, C** size **rxc**, premultiply by vector V in column **C**, put Y in **R,C**

Same as above instruction; except storage of the row vector begins in the designated column, rather than in column 1.

M(X′X) X matrix in R,C of size rxc put matrix Y in R,C

The matrix X is *pre* multiplied 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 **cxc**. The instruction

M(X′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 rxc put matrix Y in R,C

Similar to the above instruction; except the matrix X is *post* multiplied 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 **rxr**. The instruction

M(XX′) X matrix in 4,2 size 3x2 put matrix 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 in R,C size rxc, X in R,C size rxc, put Y in 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,

$$y_{ij} = \sum_{u=1}^{n} \sum_{v=1}^{n} a_{uv}x_{ui}x_{vj}, \text{ where } i=1,2,...,m \text{ and } j=1,2,...,m \text{ (8th argument).}$$

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 = X̀AX*			
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 in **R,C** size **rxc**, X in **R,C** size **rxc**, put Y in **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,

$$y_{ij} = \sum_{u=1}^{n} \sum_{v=1}^{n} a_{uv} x_{iu} x_{jv}, \text{ where } i=1,2,...,m \text{ and } j=1,2,...,m \text{ (7th argument).}$$

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 = XAX̀*		
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

11.5 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. The computations performed by MORTHO and MTRIANGULARIZE are related to the computations performed by the FIT instruction described in section C6.4.

MEIGEN of matrix A in **R,C** of size **rxc**, put eigenvalues λ 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 ascending 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.

The MEIGEN instruction uses procedures given in, Smith, B. T.; Boyle, J. M.; Garbon, B. S.; Ikebe, Y.; Klema, V. C. Moler, C. B. Matrix Eigensystem Routines - EISPACK Guide, Lecture Notes in Computer Science, 6, Springer-Veriag; 1974.

If the matrix A is not symmetric, the following fatal error mesage is printed:

MATRIX IS NOT SYMMETRIC.

If A is not square, 3rd and 4th arguments unequal, the following informative diagnostic is printed:

NO. ROWS NOT = TO NO. COLS. LARGEST SQUARE MATRIX WAS USED.

If all eigenvalues are not found, the eigenvalues stored in the fifth argument are not ordered and the following informative diagnostic is printed:

ITERATION FAILED TO FIND AN EIGENVALUE (OR EIGENVECTOR),
(n) UNORDERED VALUES FOUND.

MEIGEN of matrix A in **R,C** size **rxc**, put eigenvectors x in **R,C**

This form of MEIGEN computes the eigenvectors, but not the eigenvalues. For the matrix A and eigenvalues, λ_1 , the eigenvectors, x, satisfy the relation $Ax_i=\lambda_i x_i$, where i = 1,2,...,**r=c**. The eigenvectors are stored as a matrix, determined by the 5th and 6th (last two) arguments. The eigenvectors are stored as column vectors according to the ascending 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 A in **R,C** size **rxc**, put eigenvalues in **C** eigenvectors 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	72.0	56.0	−64.0	−64.0
3	56.0	72.0	−64.0	−64.0
4	−64.0	−64.0	72.0	56.0
5	−64.0	−64.0	56.0	72.0

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	−1.1802143-07	*2*	.50000000	−2.3403118-08	.70710678	.50000000
2	15.999998	*3*	.49999999	−1.5923433-08	−.70710678	.50000000
3	16.000000	*4*	.49999995	−.70710680	4.4801595-09	−.49999998
4	2.5599998+02	*5*	.50000003	.70710674	0.	−.50000001

MINVERT the matrix A in **R,C** of size **rxc**, put A 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^{-1}, A . The commands INVERT and MINVERT are synonymous. The MINVERT instruction uses routines given in, Forsythe, G. E.; Moler, C. B. Computer Solution of Linear Algebra Systems. Prentice-Hall; 1967.

Since matrix A must be square, the 3rd and 4th arguments, **r** and **c**, must be equal. If **r**≠**c**, the following informative diagnostic is printed:

NO. ROWS NOT = TO NO. COLS. LARGEST SQUARE MATRIX WAS USED.

If **r** and **c** are inadvertently too large, the following fatal error occurs:

ARRAY OR MATRIX OUTSIDE (n) ROW X (n) COLUMN WORKSHEET.

In order to invert a matrix, a large amount of space (scratch area) is required in the computer. If 2**r**(**r**+4) is greater than the scratch area (12,500 standard size), the following fatal error message is printed:

INSUFFICIENT SCRATCH AREA.

(To obtain square matrices larger than 62x62 but less than 82x82, the worksheet would have to be re-dimensioned.)

Finally, if the matrix is singular, the following fatal error message is printed:

MATRIX IS (NEARLY) SINGULAR.

In the LIST OF DATA, INSTRUCTIONS 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 in example below, is

++++ **** SMALLEST ERROR BOUND ON INVERTED MATRIX IS .0

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:

$$(1)\ N(Z) = \left[\sum_{i,j}^{n} |z_{ij}|^2\right]^{\frac{1}{2}}$$

$$(2)\ N(Z) = n \max_{i,j} |z_{ij}|$$

$$(3)\ N(Z) = \max_{i} \sum_{j=1}^{n} |z_{ij}| .$$

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 .

The error bound is a very conservative estimate of the error in computing the inverse.

An example of the use of MINVERT to invert a (scaled) Hilbert matrix is shown below. (The scaling factor 60.0 was used to make each element an integer.)

```
OMNITAB 80 MINVERT EXAMPLE
3X3 HILBERT MATRIX (MATRIX A)
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 (A)
ROW/COL          4              5              6
    1        .15000000     -.60000000      .50000000
    2       -.60000000     3.2000000     -3.0000000
    3        .50000000    -3.0000000      3.0000000

 EXACT INVERSE OF HILBERT MATRIX
ROW/COL          7              8              9
    1        .15000000     -.59999999      .50000000
    2       -.59999999     3.2000000     -3.0000000
    3        .50000000    -3.0000000      3.0000000



OMNITAB 80 MINVERT EXAMPLE

                  LIST OF DATA, INSTRUCTIONS AND DIAGNOSTICS


READ MATRIX A (HILBERT MATRIX) AND EXACT INVERSE INTO COLS 1***6
 60   30   20    9  -36   30
 30   20   15  -36  192 -180
 20   15   12   30 -180  180

     * INFORMATIVE DIAGNOSTIC FOR THE ABOVE INSTRUCTION -
        NRMAX HAS BEEN RESET FROM    0 TO    3.

WIDTH 80
ADIVIDE MATRIX B IN ROW 1 COL 4 SIZE 3X3 BY SCALER 60. PUT IN ROW 1 COL 7
MINVERT MATRIX A IN ROW 1 COL 1 SIZE 3X3 PUT INVERSE IN ROW 1 COL 4
 ++++ SMALLEST ERROR BOUND ON INVERTED MATRIX IS  .0
NOTE 3X3 HILBERT MATRIX (MATRIX A)
MPRINT ROW 1 COL 1 SIZE 3X3
SPACE
NOTE  INVERSE OF HILBERT MATRIX (A)
MPRINT MATRICES IN ROW 1 COL 4 SIZE 3X3
SPACE
NOTE  EXACT INVERSE OF HILBERT MATRIX
MPRINT MATRICES IN ROW 1 COL 7 SIZE 3X3
STOP

          NATIONAL BUREAU OF STANDARDS. GAITHERSBURG, MD. 208994
                 OMNITAB 80 VERSION 6.03   DECEMBER 30, 1982
```

MORTHO X **R,C** size **rxc** 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 **rxc** matrix as indicated by the 6th and 7th (last two) arguments. See Davis, P. J. Orthonormalizing codes in numerical analysis and 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:

MATRIX EXTENDS BEYOND (n) ROW BY (n) COLUMN WORKSHEET.
ONLY PART OF THE MATRIX IS STORED IN THE WORKSHEET.

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 $\Phi = XA$, where A is an upper triangular matrix. Each element of the matrix shown below is multiplied by 16.0 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 (sec. C6.4), $\Phi = XA$ is computed from the set of vectors X. The matrices A and X are related by $X'WX = (A)'(A)$, where W is a diagonal matrix with the column of weights on the diagonal.

MORTHO X **R,C** size **rxc** weights **E** put orthonormal vectors in **R,C** and A in **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 orthonormal 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 1,1 size **7x4** wts in col **10**, put orthonormal in **11,41** and A in **21,41**

would put the transformation matrix, A, in the worksheet as shown below. For ease in reading, each element in matrix A is multiplied by 16.

Row/Column	*41*	*42*	*43*	*44*
21	8.0	−6.0	−5.0	−17.0
22	0.0	4.0	−2.0	−10.0
23	0.0	0.0	4.0	−4.0
24	0.0	0.0	0.0	16.0

See the description of the second form of MTRIANGULARIZE for further remarks.

MTRIANGULARIZE the matrix A in **R,C** size **rxc** put matrix T 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. Matrix 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 **rxr** and all the elements above the principal diagonal are equal to zero. If any of the following three conditions exist:

(1) **r≠c**,

(2) any leading submatrix is singular, or

(3) A is not symmetric, $||a_{ij}/a_{ji}|-1|>10^{-6}$,

the following fatal error is printed

INCORRECT ARGUMENT IN INSTRUCTION.

Formulas for performing the triangularization are described, starting on page 6-38, in Natrella, M. G. Experimental Statistics, Natl. Bur. of Stand. (U.S.) Handb. 91; 1963 August.

MTRIANGULARIZE matrix A in **R,C** size **rxc** put T 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 2r(r+4) is greater than the size of the worksheet, the following fatal error occurs:

INSUFFICIENT SCRATCH AREA.

Using the following matrix, A:

Row/Column	*2*	*3*	*4*	*5*
3	4.0	6.0	8.0	10.0
4	6.0	25.0	20.0	27.0
5	8.0	20.0	36.0	30.0
6	10.0	27.0	30.0	36.0

MTRIANGULARIZE the matrix A in **3,2** size **4x4** put T in **3,7** inverse **3,11.**

would compute the triangular matrix T and the inverse matrix of T and put the matrices in the worksheet as follows:

MATRIX T

Row/Column	*7*	*8*	*9*	*10*
3	2.0	0.0	0.0	0.0
4	3.0	4.0	0.0	0.0
5	4.0	2.0	4.0	0.0
6	5.0	3.0	1.0	1.0

Inverse Matrix of T

Row/Column	*11*	*12*	*13*	14
3	0.5000	.0000	.0000	.0000
4	−.3750	.2500	.0000	.0000
5	.3125	−.1250	.2500	.0000
6	−1.0625	.6250	−.2500	1.0000

11.6 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 C10.4.

MPROPERTIES of the matrix in **R,C** of size **rxc**

The instruction prints 31 properties for a square matrix, or 20 properties for a non-square matrix, but provides no storage of results. The printing of the properties shows the row in which the property is stored, if the other options of MPROPERTIES are used, the description of the property and the value of the property. Items 1 through 18, 28 and 29 are always printed, for both square and non-square matrices.

The automatic printing of MPROPERTIES for both a square matrix and a non-square matrix using the following set of instructions,

```
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
```

is as follows:

```
PROPERTIES OF THE 4X4 MATRIX STARTING AT (1,1)
     ROW - FOR PROPERTIES STORED IN COLUMN   10
   GENERAL
       1   TRACE (  4 VALUES USED)                          101.00000
       2   TRACE NUMBER TWO                                1255.0000

       3   MAXIMUM NUMBER                                   36.000000
       4   MINUMUM NUMBER                                    4.0000000
       5   MAXIMUM NUMBER IN ABSOLUTE VALUE                 36.000000
       6   MINUMUM NUMBER IN ABSOLUTE VALUE                  4.0000000
       7   MINIMUM NON-ZERO ABSOLUTE VALUE                   4.0000000

       8   NUMBER OF POSITIVE NUMBERS                       16
       9   NUMBER OF ZERO NUMBERS                            0
      10   NUMBER OF NEGATIVE NUMBERS                        0

      11   SUM OF TERMS                                     303.00000
      12   AVERAGE                                           18.937500
      13   SUM OF SQUARES                                  7691.0000
      14   SUM OF SQUARES ABOUT MEAN                       1952.9375
      15   WITHIN ROWS SUM OF SQUARES                      1112.7500
      16   WITHIN COLS SUM OF SQUARES                      1112.7500
      17   SUM OF ABSOLUTE VALUES                           303.00000
      18   AVERAGE OF ABSOLUTE VALUES                        18.937500
   SPECIFIC
      19   DETERMINANT                                     1023.9998
      20   RANK                                               4
                 NORMS
      21   SQUARE ROOT OF SUM OF B(I,J)**2                    0.
      22   N*MAXIMUM(B(I,J))                                  0.
      23   MAXIMUM VALUE OF ROW SUM                           0.
      24   NORMALITY                                            YES*(1)
      25   SYMMETRY                                             YES*(1)
      26   SKEW SYMMETRY                                         NO*(0)
      27   DIAGONALITY                                           NO*(0)
      28   ORTHOGONALITY:  A'A = I                               NO*(0)
      29                   A'A = DIAGONAL MATRIX                 NO*(0)
      30   TRIANGULAR                                           NO**(0)
      31   STOCHASTIC (ROW AND/OR COL SUMS=1)                  NO***(0)


*    IF ANSWER IS NO: (R,C)= 0. IF ANSWER IS YES: (R,C)= 1, IF EXACT OR
     (R,C)= 2, IF TOLERANCE IS MET.

TRIANGULAR
**   (R,C)=0, IF ANSWER IS NO. (R,C)=1, IF UPPER PART OF MATRIX=0.
     (R,C)=2, IF LOWER PART=0. (R,C)=3, IF ALL OFF DIAGONAL TERMS=0.

STOCHASTIC
***  (R,C)=0, IF MATRIX IS NOT STOCHASTIC. (R,C)=1, IF EACH ROW SUM=1.
     (R,C)=2, IF EACH COL SUM=1. (R,C)=3, IF EACH ROW AND COL SUM=1.
```

```
PROPERTIES OF THE 3X4 MATRIX STARTING AT (1,1)
   GENERAL
       1   TRACE (  3 VALUES USED)                          65.000000
       2   TRACE NUMBER TWO                                644.00000

       3   MAXIMUM NUMBER                                   36.000000
       4   MINUMUM NUMBER                                    4.0000000
       5   MAXIMUM NUMBER IN ABSOLUTE VALUE                 36.000000
       6   MINUMUM NUMBER IN ABSOLUTE VALUE                  4.0000000
       7   MINIMUM NON-ZERO ABSOLUTE VALUE                   4.0000000

       8   NUMBER OF POSITIVE NUMBERS                       12
       9   NUMBER OF ZERO NUMBERS                            0
      10   NUMBER OF NEGATIVE NUMBERS                        0

      11   SUM OF TERMS                                    200.00000
      12   AVERAGE                                          16.666667
      13   SUM OF SQUARES                                 4666.0000
      14   SUM OF SQUARES ABOUT MEAN                      1332.6667
      15   WITHIN ROWS SUM OF SQUARES                      740.00000
      16   WITHIN COLS SUM OF SQUARES                      829.33334
      17   SUM OF ABSOLUTE VALUES                          200.00000
      18   AVERAGE OF ABSOLUTE VALUES                       16.666667
   SPECIFIC
      28   ORTHOGONALITY:  A'A = I                              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 COLUMN WISE.
    ( (R,C)=I: IF I=1 OR 3, ORTHOGONALITY IS EXACT;
               IF I=2 OR 4, RELATIVE WITHIN ERROR BOUND OF .1E-6.)
```

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 number two, is evaluated using the formula

$$\sum_{K=2}^{n} \sum_{i=1}^{k-1} (a_{ii}a_{kk} - a_{ki}a_{ik}),$$ where n is the smaller of **r** and **c**.

Trace number two 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 C6.6. 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 Perlis, S. Theory of Matrices. Reading, Mass: 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, 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-wise 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 C11.5 for further details.

Items 24 through 27 indicate whether a square matrix is a normal symmetric, skew symmetric or diagonal matrix, respectively. 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 $1x10^{-7}$.

A matrix is triangular, item 30, if all the elements below (or above) the principal are 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 Cox, D. R.; Miller, H. D. The theory of stochastic processes. New York, NY: John Wiley and Sons.

The commands WIDTH and/or BRIEF have no effect on the automatic printout.

MPROPERTIES of matrix **R,C** size **rxc** put properties in column **C**

Same as the first form of MPROPERTIES, except the properties of the matrix are printed and also put in the column designated by the 5th (last) argument.

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 **R,C** size **rxc** put row of column averages in **R,C**

Same as the first form of MPROPERTIES, except 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 309, 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 **rxc** put properties in **C**, column averages 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 **rxc** put column averages in **R,C** and row averages in **R,C**

This form of the instruction prints the properties of a matrix and 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 309, 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 **rxc** put in **C**, averages 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.

SMPROPERTIES of matrix **R,C** size **rxc** put properties in column **C**

All forms of the MPROPERTIES 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. This instruction and the remaining instructions listed below, are not described since they are similar to the last five MPROPERTIES instructions described previously. 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:

THIS INSTRUCTION WAS IGNORED BECAUSE...
COMMAND BEGINS WITH S AND STORAGE MUST BE REQUESTED.

SMPROPERTIES of **R,C** size **rxc** put row of column averages in **R,C**

SMPROPERTIES of **R,C** size **rxc** put properties in **C**, column averages in **R,C**

SMPROPERTIES of **R,C** size **rxc** put column averages in **R,C**, row averages in **R,C**

SMPROPERTIES of **R,C** size **rxc** put in **C**, averages in **R,C** and **R,C**

11.7 Printing

MPRINT, MPRINT "L"

These two instructions are listed here for completeness, but they are described in section C2.6.

MPRINT the matrix in **R,C** of size **rxc**

MPRINT "L" format, the matrix in **R,C** of size **rxc**

12. 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 arguments. Also, included are instructions for computing zeros of Bessel functions and a definite integral.

12.1 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. 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.; Stegun, I. A. Chapter 9 in Handbook of Mathematical Functions, Natl. Bur. Stand. (U.S.) Handb. AMS 55; 1964 June.

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 fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

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 fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

12.2 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 the largest value of exp(K) that can be evaluated in a single precision floating point arithmetic without overflow. Whenever scaling does occur, the following arithmetic fault will be given:

VALUE SCALED TO AVOID OVERFLOW OR UNDERFLOW.

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 (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 column **38** and put results in column **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 $|x| \geqslant K$, $I_0(x)$ 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 $|x| \geqslant K$. If $x=0.0$, the result is set equal to zero and the following arithmetic fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

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 $|x| \geqslant K$. If $x=0.0$, the result is set equal to zero and the following arithmetic fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

12.3 Modified Functions With Extreme Valued Argument

EXIONE, EXIZERO, EXKONE, EXKZERO

The formulas for evaluating these functions are the same as those in section C12.2, 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

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 fault is given;

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

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 fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

12.4 Complex Functions; Angle $=\pi/4$ (Kelvin Functions)

KBIONE, KBIZERO, KBKONE, KBKZERO

The Bessel functions of complex argument, $R\exp(i\pi/4)$, 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 the largest value that can be evaluated in a single precision floating point arithmetic without overflow and the following arithmetic fault will be given.

VALUE SCALED TO AVOID OVERFLOW OR UNDERFLOW.

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 $I_1(R\exp(i\pi/4))$. 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 in rows one, two and three the number -.48745418 in 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, $I_0(R\exp(i\pi/4))$. 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 $K_0(R\exp(i\pi/4))$. 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(iπ/4)). Whenever necessary, the scale factor used is exp(R/$\sqrt{2}$).

12.5 Complex Functions With Extreme Valued Real Argument (Kelvin Functions)

KEXIONE, KEXIZERO, KEXKONE, KEXKZERO

These instructions are similar to the instructions in section C12.4. 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 arguments of order one with a scale factor, exp(-R/$\sqrt{2}$) I_1 (Rexp(iπ/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(iπ/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(iπ/4)). If R is less than or equal to zero, the following arithmetic fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

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(iπ/4)). If R is less than or equal to zero, the following arithmetic fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

12.6 Complex Functions With Arbitrary Angle, 0<A>π/2

CIONE, CIZERO, CKONE, CKZERO

These instructions are similar to the instructions in section C12.4. Whereas in section C12.4 the angle is assumed to be π/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 π/2.

The results will be scaled, if the absolute value of Rcos(A) is greater than or equal to the largest value that can be evaluated in a single precision floating point arithmetic without overflow, and the following arithmetic fault will be given:

VALUE SCALED TO AVOID OVERFLOW OR UNDERFLOW.

CIONE of R=**E**, A=**E** put real part in column **C** and imaginary part in column **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 column **C** and imaginary part in column **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 column **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 column **C** and imaginary part in column **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.

12.7 Complex Functions With Extreme Real Argument

CEIONE, CEIZERO, CEKONE, CEKZERO

These commands are the same as those in C12.6, except each of the functions is always multiplied by a scale factor.

CEIONE of R=**E**, A=**E** put real part in column **C** and imaginary part in column **C**

CEIONE evaluates the Bessel function of complex arguments of order one with a scale factor, $\exp(-R\cos(A))I_1(R\exp(iA))$.

CEIZERO of R=**E**, A=**E** put real part in column **C** and imaginary part in column **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 column **C** and imaginary part in column **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 fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

CEKZERO of R=**E**, A=**E** put real part in column **C** and imaginary part in column **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 fault is given:

SQRT, LOG, OR RAISE OF NEGATIVE NUMBER.

12.8 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 column **C** and $J_0(x_S)$ in column **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;

		x	$J_0(x_s)$
s	*Row/column*	*17*	*24*
1	*1*	3.8317060	−.40275940
2	*2*	7.0155866	.30011575

ZEROS BJZERO put x in column **C** and $J_1(x_s)$ in column **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	J (x)
s	*Row/Column*	*1*	*2*
1	*1*	2.4048256	.51914749
2	*2*	5.5200781	−.34026480

12.9 Bessel Functions 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 greater than 100 for BESIN and BESJN or greater than 88 for BESKN, the following informative diagnostic is given:

THE INSTRUCTION WAS IGNORED BECAUSE...
VALUE OF FUNCTION IS TOO LARGE OR TOO SMALL.

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*	27.239872
I_1 (5)	*2*	24.335642
I_2 (5)	*3*	17.505615

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.

12.10 Integral

INTJO

INTJO of x=**E** put in column **C**

INTJO evaluates the definite integral of the Bessel function of order zero:

$$f(x) = \int_0^x J_0(t)dt.$$

13. THERMODYNAMICS

This section contains ll instructions which are useful for thermodynamic calculations. The instructions in section C13.1 have general use.

13.1 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:

POSITIVE, INSTEAD OF NEGATIVE, TEMPERATURES USED.

CTOF for Celsius **E** put Fahrenheit equivalent in column **C**

The instruction converts degrees Celsius to degrees Fahrenheit using the relation

$$°F = 32.0 + 1.8\ °C.$$

FTOC Fahrenheit is **E** and put Celsius equivalent in column **C**

The instruction converts degrees Fahrenheit to degrees Celsius using the relation

$$°C = (°F - 32.0)/1.8.$$

13.2 Systems of Units

CGS, SI

Eighteen physical constants may be referenced in either centimeter-gram-second (CGS) units or in the Systéme International (SI) units. If the system of units is not specified, then SI units will be used.

The use of fundamental physical constants is given and discussed in section B1.13. The fundamental physical constants listed in the table are in the OMNITAB 80 system to the full significance given in Abramowitz, M.; Stegun, I. A. Handbook of Mathematical Functions, Natl. Bur. Stand., (U.S.) Appl. Math. Ser. 55, 1964, page 7.

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

		Value	
OMNITAB	*Physical Constant*	*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, Natl. Bur. Stand. (U.S.) Handb. 102; 1966. Values of the constants are given on pages 42 and 43.

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, where NRMAX=4,

CGS system
DEFINE *CTWO* into column **5**

the value entered in the first 4 rows of column 5 is 1.43879.

SI system of fundamental physical constants, Systéme International

This instruction resets the units of the fundamental physical constants in the SI system, or Systéme International d'Unites.

13.3 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 NUMBERS.
FIRST (n) NUMBERS WERE STORED.

If NRMAX is less than 103, 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 compounds 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 atomic no.=**1**, **2** parts hydrogen, atomic no.=**8**, 1 part oxygen put in col **31**

If NRMAX=4, then column 31 contains 18.0152, 18.0152, 18.0152 and 18.0152.

13.4 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(S) OUTSIDE (n) COLUMN WORKSHEET.

Negative temperatures and wave numbers are not permitted and cause the following fatal error:

INCORRECT ARGUMENT IN INSTRUCTION.

BOLDISTRIBUTION for temp **E**, wave nos **C**, degeneracies **C** put table in **C** and beyond

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 non-zero degeneracy to obtain n pairs, which may be above or below NRMAX. The table generated has NRMAX rows and n columns.

If only one non-zero 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 a fatal error. If all zero degeneracies are found in the specified column, an informative diagnostic is given and the instruction is ignored.

EINSTEIN of temperatures **E**, wave numbers **E**, put table in **C** and successive columns

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 2 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	.33757956	.50193889	.83951845	.87936628	.72218466
2	1.50	1.43879	.25248246	.43082537	.68330783	.83184856	.61986723

EINSTEIN temperatures **E**, wave numbers **E**, **R=K**, put in **C** and successive columns

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 **E**, wave numbers in **C**, degeneracies in **C**, put in **C** and beyond

Evaluates the following three equations and stores them as a table:

$$Q^{\circ} = \sum_{i=1}^{n} g_i \exp(-hcE_i/kT)$$

$$Q^{1} = \sum_{i=1}^{n} g_i (hcE_i/kT) \exp(-hcE_i/kT)$$

$$Q^{2} = \sum_{i=1}^{n} g_i (hcE_i/kT)^2 \exp(-hcE_i/kT)$$

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 8.6720195, .34316519 and .15559990 into columns 11, 12 and 13 of rows 1, 2 and 3.

PFATOMIC temp in **E** mol wt **E** wave numbers in **C** degeneracies **C**, put in **C** and beyond

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	$-(F^{\circ}E_o^{\circ})/RT = 2.5 x \ln(T) + 1.5 x \ln(M) - 3.66495 + \ln(Q^0)$
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 and7500.0 into columns 41 to 46 of rows 1, 2 and 3.

PFTRANSLATIONAL temp is **E** mol wt **E** put in **C** and successive columns

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	$-(F^{\circ} - E_o^{\circ})/RT = 2.5 x \ln(T) + 1.5 x \ln(M) - 3.66495$
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	$(H^{\circ} - E_o^{\circ})/R$

PART D: LIST OF INSTRUCTIONS DESCRIBED IN PART C

Instructions are listed alphabetically. All array instructions begin with the letter A and all matrix instructions begin with the letter M, except for SAPROPERTIES and SMPROPERTIES which suppress automatic printing. When an instruction has optional forms, the alternative forms are given just below and are indented. Abbreviations and synonyms are not listed separately, but are enclosed in parentheses and listed under the related instruction. OMNITAB 80 instructions consist of a command name, arguments and descriptive text.

The command name is given in bold face capital letters. At least one blank character must follow the command name. Some command names have one or two qualifiers denoted by "**L**", where "**L**" indicates either the letter A, B, C, D, E or F. The qualifiers (without quotation marks) are part of the command name and at least one blank character must precede and follow each qualifier. The RESET "**V**" instruction has the qualifier "**V**", where "**V**" denotes the letter V, W, X, Y or Z. 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 bold face letter indicates the type of argument (number) allowed. Lower case letters always represent integers without a decimal point. Examples are **r** = the number of rows and **c** = the number of columns. Capital bold face letters are used as follows:

C = a COLUMN number without a decimal point,
E = EITHER a column number or a constant,
K = a CONSTANT with a decimal point,
N = an instruction NUMBER with or without a decimal point and
R = a ROW number without a decimal point.

Descriptive text which clarifies the meaning of the instruction, command name and type of argument is printed in lower case letters. Descriptive text is non-essential and it is used only to clarify the meaning of the instruction. The dollar sign, $, precedes additional comments and information.

To the right of the instruction, auxiliary information in code is given. The code consists of single characters separated by commas, except the first character may be a number indicating the number of arguments in the instruction. Use of this information can prevent errors.

Codes are defined as follows.

B = this instruction may store values *b*elow NRMAX.
C = this instruction *c*annot be stored for repeated execution.
D = number of arguments in the instruction must be o*dd*, but cannot exceed 100.
E = number of arguments in the instruction must be *e*ven, but cannot exceed 100.
M = this instruction *m*ust be stored for repeated execution.
N = execution of this instruction may or will affect the value of *N*RMAX.
P = this instruction produces *p*rinting.
V = the number of arguments is *v*ariable, but cannot exceed 100.
W = this instruction will *w*ork 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.

On the extreme right, the page number of the instruction described in Part C is given.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args *o*dd; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage; A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store; (R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage; A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store; (R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; B=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

U.S. GOVERNMENT PRINTING OFFICE:1986-181-076/40198

N=affect *N*RMAX; P=*p*rint; S=*s*ynonym; V=args *v*ariable; W=*w*ork below NRMAX; X=no storage;
A=*a*bbreviation; **B**=*b*elow NRMAX; C=*c*annot store; D=args o*dd*; E=args *e*ven; M=*m*ust store;
(R)=ROW number; (small letter)=always integer; qualifier "L"=LETTER A, B, C, D, E or F.

FORM NBS-114A (REV. 11-84)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS/SP-701	2. Performing Organ. Report No.	3. Publication Date November 1986

4. TITLE AND SUBTITLE

OMNITAB 80: An Interpretive System for Statistical and Numerical Data Analysis

5. AUTHOR(S)

Sally T. Peavy, Shirley G. Bremer, Ruth N. Varner, and David Hogben

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

same as item 6

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 86-600589
(Supersedes NBS Technical Note 552 and NBSIR 77-1276)

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. 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 80 is a highly integrated general purpose programming language and statistical software computing system. The system enables the user to use a digital computer to perform statistical and numerical data analysis without having any prior knowledge of computers or programming languages. The system responds to simple instructions to obtain accurate results since reliable, varied and sophisticated algorithms for data analysis and manipulation are referenced. It may be used either interactively or in batch mode. OMNITAB 80 has been installed nationally and internationally.

OMNITAB has been completely written to make it as machine independent as possible. This document describes Version 6.0. 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. 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.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

array and matrix operations; Bessel functions; computer language; data analysis; data manipulation; linear least squares fit; mathematical and statistical software system; numerical analysis; plotting; portable software system; probability functions; statistical analysis; tabulation; thermodynamic properties

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA 22161

14. NO. OF PRINTED PAGES

353

15. Price

USCOMM-DC 85-6006

NBS Technical Publications

Periodical

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

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.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Bureau of Standards
Gaithersburg, MD 20899