WHC-EP-0418

CDFTBL: A Statistical Program for Generating Cumulative Distribution Functions from Data

User's Manual, Version 1.0

Prepared for the U.S. Department of Energy
Office of Environmental Restoration and
Waste Management

Westinghouse
Hanford Company Richland, Washington

Hanford Operations and Engineering Contractor for the
U.S. Department of Energy under Contract DE-AC06-87RL10930

DISTRIBUTION [illegible] UNLIMITED

Approved for Public Release

LEGAL DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or any third party's use or the results of such use of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This report has been reproduced from the best available copy. Available in paper copy and microfiche.

Available to the U.S. Department of Energy
and its contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831
(615) 576-8401

Available to the public from the U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
(703) 487-4650

Printed in the United States of America

DISCLM-1.CHP (1-91)

INFORMATION RELEASE REQUEST

References: WHC-CM-3-4

Complete for all Types of Release

Purpose

- ☐ Speech or Presentation
  - ☐ Full Paper
  - ☐ Summary
  - ☐ Abstract
  - ☐ Visual Aid
  - (Check only one suffix)
- ☐ Speakers Bureau
- ☐ Poster Session
- ☐ Videotape
- ☐ Reference
- ☐ Technical Report
- ☐ Thesis or Dissertation
- ☐ Manual
- ☐ Brochure/Flier
- ☒ Software/Database
- ☐ Controlled Document
- ☐ Other _____

New ID Number: WHC-EP-0418

Existing ID Number (include revision, volume, etc.): PNL-7688

If previously cleared, list ID number.

Date Release Required: June 28, 1991

Title: CDFTBL: A Statistical Program for Generating Cumulative Distribution Functions from Data (Users Guide Version 1.0)

Unclassified Category: UC-

Impact Level: 4

Complete for Speech or Presentation

Title of Journal

Group or Society Sponsoring

Date(s) of Conference or Meeting

City/State

Will proceedings be published? ☐ Yes ☐ No

Will material be handed out? ☐ Yes ☐ No

Title of Conference or Meeting

CHECKLIST FOR SIGNATORIES

Review Required per WHC-CM-3-4	Yes	No	Reviewer Name (printed)	Signature	Date
Classification/Unclassified Controlled Nuclear Information	☐	☑			
Patent - General Counsel	☑	☐	Carol Disibio	Carol A. Disibio	6-14-91
Legal - General Counsel	☑	☐	Carol Disibio	Carol A. Disibio	6-14-91
Applied Technology/Export Controlled Information or International Program	☐	☑			
WHC Program	☐	☑			
Communications	☐	☑			
DOE-RL Program	☑	☐	M.J. Furman	M.J. Furman	6/21/91
Publications Services	☑	☐	Leslie A. Brown	Leslie A. Brown	6/27/91
Other Program	☐	☑	Reviewed as PNL-7688	contact Dr. M.J. Fayer	
References Available to Intended Audience	☑	☐	Leslie A. Brown	Leslie A. Brown	6/27/91
Transmit to DOE-HQ/Office of Scientific and Technical Information	☐	☑			

Information conforms to all applicable requirements. The above information is certified to be correct. * See comments on cover page grayed

Author/Requestor (Printed/Signature): J. C. Sonnichsen, Jr. — John C. Sonnichsen — Date: 6/11/91

Responsible Manager (Printed/Signature): J. C. Sonnichsen, Jr. For J. W. Cammann — John C. Sonnichsen — Date: 6/11/91

Intended Audience: ☐ Internal ☐ Sponsor ☒ External

INFORMATION RELEASE ADMINISTRATION APPROVAL STAMP

Stamp is required before release. Release is contingent upon resolution of mandatory comments.

APPROVED FOR RELEASE — ORIGINAL — 16 — DOCUMENT — 6/27/91

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Date Received: 6/12/91 NL6

BD-7600-062 (04/90) Part 1

WHC-EP--0418

DE92 002106

CDFTBL: A Statistical Program for Generating Cumulative Distribution Functions from Data

User's Manual, Version 1.0

Prepared by:

P. W. Eslinger
Pacific Northwest Laboratory

Date Published
June 1991

Prepared for the U.S. Department of Energy
Office of Environmental Restoration and
Waste Management

Westinghouse Hanford Company
P.O. Box 1970
Richland, Washington 99352

Hanford Operations and Engineering Contractor for the
U.S. Department of Energy under Contract DE-AC06-87RL10930

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Approved for Public Release

CDFTBL: A STATISTICAL PROGRAM FOR GENERATING CUMULATIVE DISTRIBUTION FUNCTIONS FROM DATA

USER'S MANUAL, VERSION 1.0

P. W. Eslinger
Pacific Northwest Laboratory

ABSTRACT

This document describes the theory underlying the CDFTBL code and gives details for using the code. The CDFTBL code provides an automated tool for generating a statistical cumulative distribution function that describes a set of field data. The cumulative distribution function is written in the form of a table of probabilities, which can be used in a Monte Carlo computer code. As a specific application, CDFTBL can be used to analyze field data collected for parameters required by the PORMC computer code.

Section 2.0 discusses the mathematical basis of the code. Section 3.0 discusses the code structure. Section 4.0 describes the free-format input command language, while Section 5.0 describes in detail the commands to run the program. Section 6.0 provides example program runs, and Section 7.0 provides references. The Appendix provides a program source listing.

CONTENTS

1.0 INTRODUCTION 1
1.1 MOTIVATION 1
1.2 PURPOSE 1

2.0 MATHEMATICAL BASIS 1
2.1 BACKGROUND 1
2.2 EMPIRICAL DISTRIBUTION FUNCTIONS 2
2.3 CUMULATIVE DISTRIBUTION FUNCTIONS BASED ON SAMPLE DENSITY ESTIMATES 3
2.4 RANDOM-SAMPLE ASSUMPTION 4
2.5 PRACTICAL USE CONSIDERATIONS 4

3.0 CODE STRUCTURE AND PROGRAM EXECUTION 5
3.1 INPUT AND OUTPUT FILES 5
3.1.1 The Input File 5
3.1.2 Description of the Output Files 6
3.2 RUNNING THE PROGRAM 6
3.3 SUBROUTINES AND THEIR FUNCTIONS 7
3.4 VARIABLE DEFINITIONS AND DIMENSION PARAMETERS 7
3.5 ERROR MESSAGES 7

4.0 DESCRIPTION OF FREE-FORMAT COMMAND LANGUAGE 11
4.1 RECORD TYPES 11
4.1.1 The Keyword Record 11
4.1.2 Continuation Records 12
4.1.3 Comment Records 12
4.2 ELEMENTS OF THE INPUT RECORD 13
4.2.1 The Keyword 13
4.2.2 The Modifier 13
4.2.3 The Numeric Field 14
4.2.4 The Separator Field 15
4.2.5 The Terminator Field 16
4.2.6 The Comment Field 16

5.0 DETAILED DESCRIPTION OF KEYWORD COMMANDS 17
5.1 DISTRIBU COMMAND 18
5.1.1 Purpose 18
5.1.2 Syntax 19
5.1.3 Examples 20
5.2 END COMMAND 20
5.2.1 Purpose 20
5.2.2 Syntax 20
5.2.3 Example 21
5.3 PLOT COMMAND 21
5.3.1 Purpose 21
5.3.2 Syntax 21
5.3.3 Examples 21

CONTENTS (continued)

5.4 PRINT COMMAND 22
5.4.1 Purpose 22
5.4.2 Syntax 22
5.4.3 Examples 22
5.5 PROBABIL COMMAND 23
5.5.1 Purpose 23
5.5.2 Syntax 23
5.5.3 Examples 24
5.6 SAMPLE COMMAND 24
5.6.1 Purpose 24
5.6.2 Syntax 24
5.6.3 Examples 24
5.7 TRANSFOR COMMAND 25
5.7.1 Purpose 25
5.7.2 Syntax 25
5.7.3 Examples 25
5.8 USER COMMAND 26
5.8.1 Purpose 26
5.8.2 Syntax 26
5.8.3 Example 26

6.0 EXAMPLE RUNS AND CODE VERIFICATION 27
6.1 EXAMPLE 1 27
6.2 EXAMPLE 2 27
6.3 EXAMPLE 3 34
6.4 EXAMPLE 4 34

7.0 REFERENCES 45

APPENDIX

A Source Code Listing A-1

LIST OF FIGURES

1 Interpolation Approach to Inverting the Cumulative Distribution Function 2

2 Subroutine Calling Hierarchy for the CDFTBL 10

LIST OF TABLES

1 Example Interactive Outputs by the CDFTBL 7

2 Modules and Their Function 8

3 Parameter Definitions in the CDFTBL Code 10

4 Keywords and Their Functions for the CDFTBL Code 18

5 **DISTRIBU** Keyword Numeric Value Description 19

6 **PLOT** Keyword Modifiers and Their Descriptions 21

7 **PRINT** Keyword Numeric Value Explanations 22

8 PROBABIL Keyword Numeric Value Explanations 23

9 Sample Keyword Numeric Value Explanations 25

10 TRANSFOR Keyword Numeric Value Explanations 26

11 Input Data File for Example 1: EXAMPLE1.DAT 28

12 Report File for Example 1: EXAMPLE1.RPT 29

13 Input Data File Example 2: EXAMPLE2.DAT 31

14 Report File for Example 2: EXAMPLE2.RPT 32

15 Input Data File Example 3: EXAMPLE3.DAT 35

16 Report File for Example 3: EXAMPLE3.RPT 36

17 Input Data File Example 4: EXAMPLE2.DAT 39

18 Report File for Example 4: EXAMPLE4.RPT 40

19 Plot File for Example 4: EXAMPLE4.PLT 42

This page intentionally left blank.

CDFTBL: A STATISTICAL PROGRAM FOR GENERATING CUMULATIVE DISTRIBUTION FUNCTIONS FROM DATA

USER'S MANUAL, VERSION 1.0

1.0 INTRODUCTION

1.1 MOTIVATION

When using a Monte Carlo computer code such as PORMC (Runchal and Sagar 1989) to analyze the effects of uncertainties in input data on a performance measure, statistical distributions must be chosen for selected input parameters. When data on the parameter of interest are available, the appropriate statistical distribution can be specified in one of two ways. First, a classical statistical distribution can be chosen that approximates the data, then the analytical form for the distribution can be incorporated into the Monte Carlo code. Second, the data can be used to generate a statistical distribution in the form of a table of values. This table can then be input into the Monte Carlo code without the intermediate step of choosing an appropriate analytical form for the distribution. An automated technique for building the table of values is, therefore, a useful analytical tool.

1.2 PURPOSE

This document describes the theory underlying the CDFTBL code and gives details for using the code. The CDFTBL code provides an automated tool for generating a statistical cumulative distribution function (CDF) from a set of input data.

2.0 MATHEMATICAL BASIS

2.1 BACKGROUND

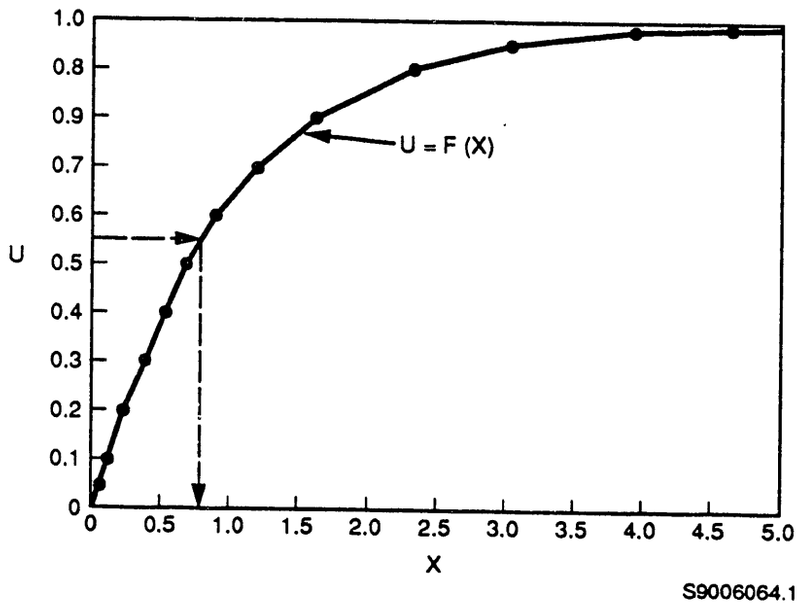
A Monte Carlo computer code must be able to generate sets of values that behave as if they were random samples from specified probability distributions. A variety of techniques can be used to generate the desired values, many of which depend on the particular distribution chosen. One method, the Probability Integral Transform (PIT) method (Mood et al. 1974, p. 202), can be applied to any continuous statistical distribution.

Let $F(x)$, $0 \leq F(x) \leq 1$, denote the CDF of a random variable X. To apply the PIT method for generating random variables, first generate a value, u, from the uniform (0,1) distribution. Often this can be done using the default random number generator on a computer system. As an alternative, the portable uniform generator written by Lewis et al. (1969) can be used. Next, given the

value u, solve the equation $F(x) = u$ for x [i.e., solve the equation $x = F^{-1}(u)$ for x]. The value x is from the distribution F(x).

The PIT method is sometimes difficult to implement because a convenient form for $F^{-1}(u)$ may not exist. However, if the distribution is defined in terms of a table of (x, F(x)) pairs, then linear interpolation can be used to choose the appropriate x value given a value u [=F(x)]. Figure 1 shows the interpolation approach. The interpolation can be performed at any desired accuracy by increasing the number of values entered in the table.

Figure 1. Interpolation Approach to Inverting the Cumulative Distribution Function.



The CDFTBL code generates a table of [x, F(x)] pairs given a set of field data. The table need not correspond to any classical analytical CDF. To generate the table, two options are (1) base the table on the empirical distribution function and (2) generate a CDF as the integral of a sample-based probability density function.

2.2 EMPIRICAL DISTRIBUTION FUNCTIONS

Consider the case in which N data points have been collected (a random sample). The data are assumed to come from some unknown distribution with CDF F(x). The empirical distribution function EF(x) (Mood et al. 1974, p. 264) estimates the CDF using the sample data. The empirical CDF is defined as follows:

$$EF(x) = n(x) / N$$

where n(x) is the number of data values that are less than or equal to x. If all of the data values are unique, EF(x) has jumps of height of 1/N at each of the data locations. The jumps in EF(x) have height k/N at location x if k of the data values have the same value x, i.e., if there are k ties in the data set at location x.

When a data set with k ties at a location x is encountered, the data are modified to eliminate the ties. The data are adjusted to have a spread of (1/k)% of the distance between the next largest and smallest data values. This approach ensures unique x values for separate probability levels in the output table. The data adjustments are made after data transformations are performed.

The analyst chooses a set of probabilities, $\{p_i\}$, to use in the output table of (x,p) pairs. Then, linear interpolation can be performed on the function EF(x) to find the x values corresponding to each of the p_i.

A special case arises when a chosen p_i is less than 1/N and the table of probabilities is being generated using the empirical CDF approach. In that case, the desired x value may be less than the smallest observed data value. The code assigns p_i to the smallest data value (x) for each p_i less than 1/N.

2.3 CUMULATIVE DISTRIBUTION FUNCTIONS BASED ON SAMPLE DENSITY ESTIMATES

Consider again the case in which N data points have been collected (a random sample). The data are assumed to come from some unknown distribution with CDF F(x). The CDF F(x) can be considered as the integral of the probability density function over the range $-\infty$ to x. Another approach to finding a sample-based CDF is to generate a density estimate based on the sample values and then integrate it to obtain the CDF.

A class of sample density estimators called kernel estimators was considered by Parzen (1962). The kernel density estimator has the form

$$g(y) = \frac{1}{nc_N s_N} \sum_{i=1}^{N} w[(y-x_i)/(c_N s_N)]$$

where

$\{c_N\}$ = a sequence of constants converging to zero at an appropriate rate

s_N = a scale (standard deviation) estimator

w = a smooth density on the real line

x_i = a sample data value.

For application purposes, w is defined as:

$$w(z) = \begin{cases} 0.75(1-z^2) & \text{if } |z|<1 \\ 0 & \text{otherwise.} \end{cases}$$

This kernel has been shown to have optimal mean-squared-error properties for density estimation (Epanechnikov 1969). The scale estimator used is the median absolute deviation divided by 0.6745, as suggested by Beran (1977). The sequence $\{c_N\}$ chosen is $c_N = 2.283*N^{-.287}$, which is based on work by Eslinger and Woodward (1990).

The CDFTBL code uses the sample data values to generate the sample density function, using g(y), at a large number of y values. This set of values is then integrated to find the sample-based CDF. Finally, a linear interpolation procedure is done on the sample-based CDF to find the x values that correspond to the desired set of probabilities, $\{p_i\}$, that are entered in the output probability table of (x,p) pairs.

2.4 RANDOM-SAMPLE ASSUMPTION

The most important assumption made about the data is that the collection of data values entered into the CDFTBL program form a (statistically) random sample. If this assumption is violated, the output CDF (in the form of a table) may not be representative of the statistical distribution actually defining the physical phenomena that were sampled. The computed CDF will always have the usual (continuous, nonnegative, and integrate to unity) properties even if the random-sample assumption is violated.

Most elementary statistics textbooks (e.g., Strait 1989, p. 330) discuss the concept of a random-sample. The essence of the random-sample concept is that all data values are collected under identical conditions and are associated with identical governing probability distributions.

2.5 PRACTICAL USE CONSIDERATIONS

The two estimation procedures have subtle differences. In general, the empirical CDF approach applies a probability mass of size 1/N at the discrete locations corresponding to the data locations, while the kernel-estimation approach spreads the same probability mass over a small region centered at the data location. No strong theoretical reason exists to substantiate the preference of one approach to the other. The author generally prefers the kernel-based approach to the empirical-based approach.

The empirical-based approach may be preferable when some data are obtained "close" to a region eliminated by physical model considerations (e.g., porosity is constrained between 0 and 1). The kernel-based approach spreads the probability mass associated with a data point over a small region; thus, it may assign a non-zero probability to an "impossible" region. The empirical-based approach will assign a zero probability to the regions below

the smallest data value and above the largest data value. When the user can define "impossible" regions from physical constraints, the ALIAS modifier (see Section 5.1) on the kernel-based approach can be used to force zero probabilities in the regions identified.

Both estimation procedures are "consistent," in that the sample-based CDF converges to the theoretical CDF when a large sample is drawn from some specified probability distribution (Serfling 1980, p. 57; Devroye 1987, p. 37). In addition, convergence rates based on asymptotic arguments are available from the same references. The code requires at least four data values to successfully generate the CDF table. More than four values are desirable because the procedure estimates the shape of an entire function. As a general rule, at least 20 values are required to obtain a representative sample-based CDF. It is preferable to have 100 or more values.

Often when data are collected, one or more values are "outliers" in that they do not lie within the main body of the data. The CDFTBL code does not implement any procedures that detect or treat outliers. The user must closely scrutinize the data used. As a general rule, only those data for which a collection or measurement error can be identified should be discarded; data should not be discarded simply because they are different.

3.0 CODE STRUCTURE AND PROGRAM EXECUTION

3.1 INPUT AND OUTPUT FILES

The CDFTBL program imposes a restrictive filenaming convention. The single input file must have the name xxxxxxxx.dat. The user is free to specify any character sequence for the string represented as xxxxxxxx. The program always writes a report file that will have the name xxxxxxxx.rpt. Depending on the choice of input commands, an optional plot file also will be written. The name of the plot file will be xxxxxxxx.plt. A more detailed description of the files is given below.

All inputs to the CDFTBL code are through the data file (except the name of the data file). All outputs (other than the prompt for the data file name) are written to files as well.

3.1.1 The Input File

The single input file for the CDFTBL program consists of two sections. The first section contains free-format keyword commands to select program options and define the data set. The second section contains the input data values from which the probability table is to be generated. The input file is a formatted file, containing only American Standard Code for Information Interchange (ASCII) characters that the user must generate using a file-editing program. Section 4.0 describes the free-format command language, and Section 5.0 describes in detail the keyword-controlled options. The data values do not require any special input format; however, each line of the input must have the same number of data values. The last line containing data

values may contain a smaller number of values if the number of data values is not a multiple of the number of values per line.

3.1.2 Description of the Output Files

Two separate output files are written by the CDFTBL code. Section 6.0 gives examples of these files.

The first file, called the report file, is written every time the code is executed. If errors are detected in the input keywords, the report file contains only error messages. If no input errors are detected, the report file contains a banner page with identifying information (i.e., code name and date, user name, and run identification number). In addition, it contains the output probability table.

The second file, called the plot file, contains information that can be sent to a general-purpose (x,y) plotting program. This file is written only if the PLOT keyword is used (see Section 5.3 for a description of the PLOT keyword). Three possible plot curves are in this file. First, a copy of the probability table may be written to the plot file. Second, the CDF from which the table values are extracted may be written to the plot file. Third, if the kernel-density estimation approach is used to generate the probability table, the probability density used in the kernel-density estimator may be written to the plot file.

3.2 RUNNING THE PROGRAM

The CDFTBL code runs in an interactive mode. Once the code has been compiled and loaded onto a specific computer, the user can initiate the program run by typing the command CDFTBL on an IBM[1] personal computer (PC). On a VAX[2] the user enters RUN CDFTBL. The code then displays a few lines of information and prompts the user for the data file name. Table 1 shows an example of the message output by the CDFTBL code.

Assume that the user has created the data file EXAMPLE1.DAT, which is listed in Section 6.0. The user would then enter the text string EXAMPLE1 and press the ENTER key. No further interactive responses are necessary.

The program currently uses an uppercase type style for the file extensions (i.e., .DAT, .RPT, and .PLT). This convention can be changed by modifying three statements (calls to subroutine CONCAT) in subroutine OPENER. No restrictions are placed on the case of the name the user enters.

---

[1]IBM is a registered trademark of International Business Machines Inc.

[2]VAX is a trademark of the Digital Equipment Corporation.

Table 1. Example Interactive Outputs by the CDFTBL Code.

```
                    Cumulative Distribution Function Estimation
CDFTBL   1.0            Run ID = 900117155358     User = XXXXXX X. XXXXXX

There is 1 input file and 2 output files
The input file name must have the form "FILE.DAT"
The output file names will have the form "FILE.XXX"
The following filenaming convention is used.
  FILE.DAT : Input - control data and sample values
  FILE.RPT : Report file
  FILE.PLT : (optional) plot file for the CDF

Enter the data file name (WITHOUT extension) >
```

3.3 SUBROUTINES AND THEIR FUNCTIONS

The CDFTBL code has 28 modules, in which a module is either a FORTRAN subroutine or function. To the extent possible, each module has been assigned a single, distinct mathematical function. Subroutine IDENC is the only subroutine that is machine or operating-system dependent. The code listing in the Appendix gives versions of the subroutine for the IBM PC using the Lahey F77L compiler (Lahey 1987) and the VAX VMS[1] operating system. Table 2 lists the modules alphabetically with a brief description of their major function.

Figure 2 shows the calling hierarchy for subroutines. Some utility subroutines (e.g., PAGER, SEPCHR, and XFKEY) are called from more than one routine.

3.4 VARIABLE DEFINITIONS AND DIMENSION PARAMETERS

The CDFTBL code was designed such that all global variables are contained in common blocks. Because the variables are defined in internal comments in the source code, a description of the variables is given here. The Appendix contains a complete listing of the source code.

The CDFTBL code uses PARAMETER statements to define the dimensions of various arrays. Depending on the problem that is run, the user may need to change the dimension of five arrays. Table 3 shows the parameters and their definitions.

3.5 ERROR MESSAGES

The CDFTBL code performs extensive checks on keyword inputs. If an error is detected, an error message is written to the report file and program execution is halted before the probability table is generated. Each error message identifies the subroutine that detected the error and gives an

[1]VMS is a trademark of the Digital Equipment Corporation.

Table 2. Modules and Their Function. (sheet 1 of 2)

Module	Description
BANNER	Writes a banner page to the report file.
CDFTBL	Main (top-level) module for the program. Schedules all functions for the code.
CEXIST	Part of the set of routines called by RDBLK to decode keyword commands. Determines if an eight-letter keyword is one of the modifiers for the current data record.
COMCHR	Part of the set of routines called by RDBLK to decode keyword commands. Determines if an input character is a comment identifier.
CONCAT	Used to append extensions to the end of a file name.
DCDSEP	Part of the set of routines called by RDBLK to decode keyword commands. Examines data in the input record and transfers character and numeric data into storage vectors.
EMPIR	Computes the output table of probabilities using the empirical distribution function method.
EPAN	Computes the output table of probabilities using the kernel-density integration method.
FINSEP	Part of the set of routines called by RDBLK to decode keyword commands. Locates all of the separators in the current input record.
FNAME	Part of the set of routines called by RDBLK to decode keyword commands. Extracts a file name (character string) from an input card image.
GALIAS	Performs density aliasing operations through calls to the function GKERN.
GKERN	Evaluates the kernel density estimate at a single location.
IDENC	Generates identification variables for each run of the code. Operating-system dependent.
INPUTS	Controls the reading of keyword commands and data values from the input data file.
OPENER	Opens the input and report files.
PAGER	Writes a one-line header to an output file.
PRNTER	Prints a table of the input data values to the report file.
RDBLK	Controlling module of the set of modules that decode keyword data.
SAVTTL	Part of the set of routines called by RDBLK to decode keyword commands. Moves the contents of the input record, following the first separator, into an output array.

Table 2. Modules and Their Function. (sheet 2 of 2)

Module	Description
SCALE	Computes the scaled median absolute deviation for the data set.
SEPCHR	Part of the set of routines called by RDBLK to decode keyword commands. Determines if a single input character is a separator character.
SORT	Performs an algebraic sort on a data vector.
TIES	Checks for any tied values in the input data set.
TIES2	Adjusts the data set to eliminate ties in the input data set.
TRANS	Performs transformations on the input data.
UPCASE	Part of the set of routines called by RDBLK to decode keyword commands. Converts alphabetic characters to uppercase.
WRITER	Writes the final results of the program to the report and plot files.
XFKEY	Part of the set of routines called by RDBLK to decode keyword commands. Transfers keywords from an input record to a storage location.

Figure 2. Subroutine Calling Hierarchy for the CDFTBL Code.

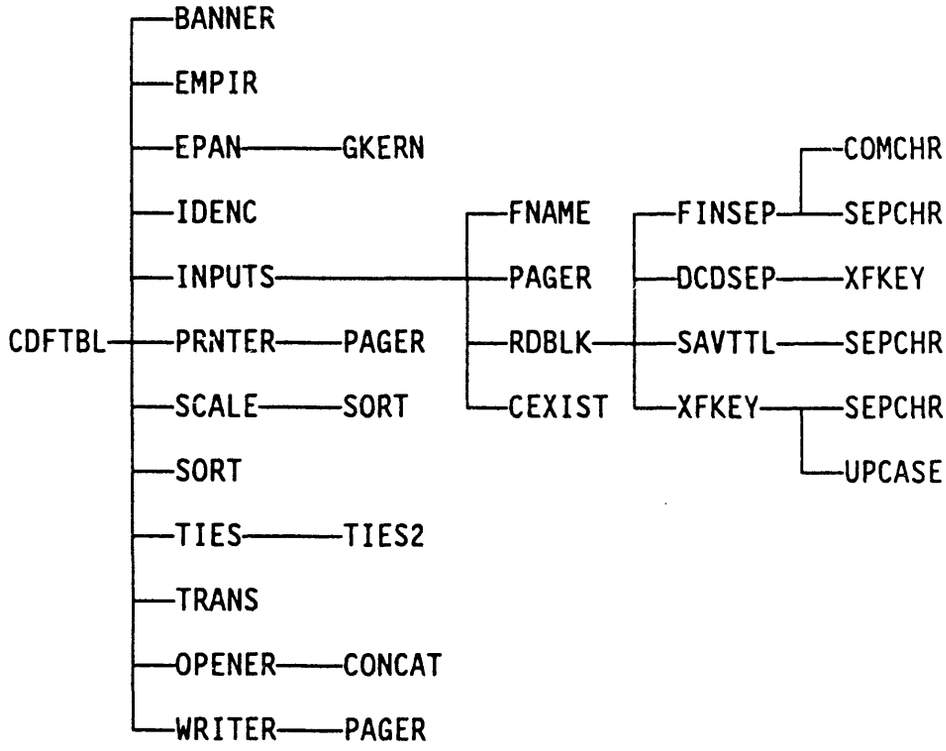


Table 3. Parameter Definitions in the CDFTBL Code.

Parameter	Value	Definition
MAXDAT	1000	Maximum number of input data values that can be used.
MAXINT	2000	Maximum number of integration steps that can be used with the kernel density method for generating the output probability table.
MXALP	500	Maximum number of modifiers that can be used in the output probability table.
NKEYS	50	Maximum number of modifiers that can be contained in a single keyword record, including any associated continuation records.
NVALS	501	Maximum number of numerical values that can be contained in a single keyword record, including any associated continuation records.

indication of the user action to correct the error. The error messages are not discussed here in additional detail because the run-time error messages give explicit instructions for correcting problems that are detected.

4.0 DESCRIPTION OF FREE-FORMAT COMMAND LANGUAGE

The user interface with CDFTBL is through a free-format command language. This command language reduces user input to a set of conversational, English-like commands. In most cases, the commands are largely free of any requirements of format and hierarchy.

The free-format data input routine (ADATA) used in PORFLO and PORMC was developed by A. K. Runchal (Runchal and Sagar 1989). A similar functionality in input structure was desired for CDFTBL because it is a preprocessor for PORMC. The RDBLK set of routines were developed separately by Westinghouse Hanford to provide this functionality. Although the routines are independent, the input conventions for CDFTBL are almost identical to those used in PORFLO. As such, the description that follows was taken from the PORFLO User's Guide (Runchal and Sagar 1989) and is included herein with only minor modification.

The free-form language reads the input data in distinct blocks. Each block begins with a keyword (Section 4.1.1) that identifies the nature of the data within the block. The character and numeric data in a block may include several lines of an input data file. The numeric data within a block apply to the keyword at the start of the block. Character data may be modifiers (i.e., user 'options'), comments, or even inconsequential text that enhance the user readability of the input. A block ends when the next keyword, or an end-of-file, is encountered.

Within the input file, three record types are recognized: keyword, comment, and continuation. These record types are described in Section 4.1. Within each record of the input file are six distinct data types, which are described in Section 4.2.

4.1 RECORD TYPES

Three types of records are described in detail in the following sections. Each 80-column card image (record) read from the data file is considered to correspond to one of these record types. Only the first 80 columns of a record are processed; any further characters are truncated.

4.1.1 The Keyword Record

Function: The keyword record indicates the purpose of the numeric and character data on the keyword record and any subsequent continuation records.

Structure: The keyword has the following attributes.

- A keyword record must begin with a keyword in column 1.
- Only one keyword per record is allowed.
- The keyword may be followed by modifiers and numerical fields.
- The keyword, the modifiers, and the numerical fields must be separated from one another by separator fields.
- Any character or numeric data in a keyword record after the first occurrence of a terminator is ignored.

4.1.2 Continuation Records

Function: A continuation record continues the input of numeric and character data started by the preceding keyword record.

Structure: The following rules are to be followed.

- The continuation record must begin with a separator as the first character of the record.
- A continuation record must always occur after a keyword record for that group.
- A continuation record must consist only of a combination of modifiers and numerical fields isolated from each other by separators.
- Any character or numerical data in a continuation record after the first occurrence of a terminator is ignored.
- Any number of continuation records may follow a keyword record.

4.1.3 Comment Records

Function: The comment record enhances the clarity and readability of the input.

Structure: The comment record has the following attributes.

- A comment record must begin with a slash (/), asterisk (*), dollar sign ($), or exclamation point (!) in the first column of the record. Any combination of characters can follow the first character.
- A comment record is not processed. No numerical or character data are extracted.

- A comment record cannot be extended by a continuation record.
- A comment record can be inserted anywhere in the input.

4.2 ELEMENTS OF THE INPUT RECORD

One or more of the following six basic elements form an input record: keyword, modifier, numeric field, separator field, terminator field, and comment field. These elements are described in the following subsections.

4.2.1 The Keyword

Function: The keyword identifies an input group.

Structure: The keyword has the following attributes.

- A keyword must begin in the first column of a record.
- The keyword must start with one of the characters 'a' through 'z' or 'A' through 'Z'.
- The keyword may consist of any character except separator or terminator characters. The concept of a keyword is similar to that of a variable name in FORTRAN.
- The keyword may be from one to eight characters in length.
- The keyword is terminated with the first occurrence of a valid separator or terminator character.
- The keyword may be entered in upper- or lowercase.

Examples:

ABCD, **A123**, and **B&C** are all valid examples of a keyword. The keyword specifications **ABCDEFGH**, **ABCDefgh**, and **AbCdEfGhXXX** (where X can be any character) are all equivalent because only the first eight characters are significant and the input is case-insensitive.

'AB, **$A**, **=junk**, and **.HELP**, are examples of invalid keyword specifications. Each starts with a separator or comment character.

Note that the specifications of **ABC'**, **ABC$**, and **ABC(** as keywords, although valid, are all equivalent to ABC because the last character of each example is either a separator or a terminator.

4.2.2 The Modifier

Any character information in an input record following a keyword, except that embedded in a numeric or comment field, is treated as a modifier.

Function: The modifier contains character data that help interpret the other data in the record. Many program options are implemented through the use of modifiers.

Structure: The modifier has the following attributes.

- A modifier in any input group, if present, must follow the keyword.
- The modifier is identical to the keyword in its structure, except it cannot start in column 1 of the record.
- The modifier must be separated from the keyword, other modifiers, or numeric data by a separator or terminator field.

Examples:

The structure of the modifier is identical to that of a keyword, except it cannot start in the first column of a record. Examples of keywords are given in Section 4.2.1.

4.2.3 The Numeric Field

Any numeric characters in a keyword or continuation record, except those embedded in a keyword, modifier, or comment field, are treated as numeric data.

Function: A numeric field contains numeric data for input variables.

Structure: The numeric field has the following attributes.

- A numeric field is a continuous string of characters that must begin with the numeric character set. In this context, the numeric character set consists of the numerals (0-9), the decimal point (.), and the plus (+) and minus (-) operators.
- The numeric field cannot contain embedded blanks or separators.
- A numeric field must consist only of the numeric character set defined above, the asterisk (*), and the exponent in lower- (e) or uppercase (E). It must not contain any other characters.
- The plus (+) or minus (-) sign, if present, must immediately precede the numerical value without any intervening blank or other characters; or, if used in an exponent, it must follow immediately after the exponent character e or E.
- The asterisk (*) or the exponent (E or e), if present, must be embedded; the numeric field must not begin or end with one of these characters.
- A numeric field must be separated from the keyword, modifiers, and other numeric fields by a valid separator or terminator field.

- A numeric field can be located anywhere on a keyword or continuation record, except in column 1.

- The numeric values can be specified in any of the following formats:

  - Integer (e.g., 999)
  - Real (e.g., 999.0 or -.99)
  - Exponent (e.g., 9.99E+02 or 9.99E-1).

- Successive, repetitive, or identical numeric values may be specified using the asterisk (*) option. Thus, (30., 30., 30.) may be represented as (3*30. or 3*3.E+1); embedded separators or nonnumeric characters must not appear in such specifications.

Examples:

The input character strings 1, 0.1234, .567, +123, -1.005, 1.2e+0, 1.35E0, and 3*1.2 are all valid examples of a numeric field. The input specifications of 123, 123., 1.23e02, +0.123E+3, 1*123, and 1*1.23E+02 are all equivalent.

The strings 1AB, 11X11, and 1+2 are examples of invalid numerical fields. The first two fields have embedded nonnumeric values, and the last field has a legal numeric character (+) embedded within the field.

A specification of 1.2)2 or 1.2=2, although valid, will be equivalent to specifying the two separate numeric fields of 1.2 and 2 because of the embedded separator character. A specification of 1.2$2 is equivalent to 1.2 because of the presence of the terminator character ($).

4.2.4 The Separator Field

Function: A separator field separates the keyword, the modifiers, and the numeric fields of an input record.

Structure: Any continuous string of characters in an input record that consists of only characters from the separator character set is treated as a separator field. The valid separator characters are the space ( ), the comma (,), the equal sign (=), the colon (:), the semicolon (;), the apostrophe ('), the left paren '(', and the right paren ')' characters.

Any character with a storage code (FORTRAN ICHAR function) less than 10 will also be treated as a separator character. This convention allows tab characters to be treated as separators.

Examples:

The sequence of characters, ';::)), ======, =', and (;) are all valid separator fields. However, the characters (a) and (1) are not valid separator fields. In the first case, the character 'a' will be processed as a modifier; in the second case, the character '1' will be processed as a numeric field.

4.2.5 The Terminator Field

Function: A terminator ends input from an individual keyword or continuation record; however, subsequent continuation cards are processed normally. It also provides a vehicle for the user to insert comments in those records.

Structure: The terminator field has the following attributes.

- The dollar sign ($) and exclamation point (!) are the only valid terminators.
- The terminator ends the input for the keyword or continuation record in which it occurs; input associated with that particular keyword may continue in a continuation record that follows.
- The terminator may appear anywhere in a record.
- Any characters following the terminator in that input record are not processed; rather, they are treated as comments.

Examples:

The character sequences XYZ !Comments now, $ any comments here, and 123.456$789 are all examples of sequences with embedded terminators. In the first sequence, XYZ will be treated as valid character data (either keyword or modifier, depending on its starting location in the input record), whereas the characters following the ! will be ignored. In the second example, the total sequence will be ignored. In the third example, 789 will be ignored, whereas 123.456 will be treated as numeric data.

4.2.6 The Comment Field

Function: A comment field provides the vehicle for the user to insert comments into the input stream to enhance the clarity and readability of the input. The comment field may be used in the middle of a keyword and continuation record sequence.

Structure: The comment field has the following attributes.

- A comment field may be in the form of a trailing comment or a comment record.
- A trailing comment field may occur in a keyword or continuation record. It must begin with a terminator character; any combination of characters can follow the terminator character. The comment field is terminated at the end of the 80th character of that record.
- A comment field in a comment record may consist of any combination of characters. The comment record must begin with a slash (/), asterisk (*), dollar sign ($), or exclamation point (!) in the first column of the record.

Examples:

In the input record ARRAY = 1., 2., 3., 4. $Example 1, the character string $Example 1 is an example of a trailing comment in a keyword record. Input processing stops with the $ character; all characters including and following the $ are ignored. The same result would be achieved if $ were replaced with !.

The following are examples of comment records:

```
/ARRAY = 1., 2., 3., 4. $Example 1
*ARRAY = 1., 2., 3., 4. !Example 1
!ARRAY = 1., 2., 3., 4. $Example 1
$ARRAY = 1., 2., 3., 4. !Example 1
```

All of these strings of characters will be treated as comment records.

5.0 DETAILED DESCRIPTION OF KEYWORD COMMANDS

The method of providing input data to CDFTBL is based on reading free-format keywords, details of which were described in Section 4.0. Each input command starts with a keyword that identifies the nature of the data to follow. The keyword is followed by alphanumeric data. The following is the notational convention for the description of input commands for CDFTBL.

BOLD The keywords for CDFTBL are shown in uppercase characters in boldface type. The string of keyword characters may be specified by the user in upper- or lowercase. Boldface is used here only for notational purposes; it <u>must not</u> be used as operator input.

CAPS Uppercase characters in standard typeface are modifiers of the CDFTBL keywords that are <u>significant</u> for machine interpretation of user input. The string of characters shown may be specified by the user in either upper- or lowercase.

char Lowercase characters denote information on keyword commands that are <u>not significant</u> for machine interpretation of user input, but improve the clarity or readability of the input. The string of characters shown may be specified by the user in either upper- or lowercase. In addition, they may be omitted or replaced by other character strings by the operator.

| The vertical bar indicates a choice; only one of the items separated by the bar (and enclosed in braces or square brackets) may be specified.

{ } Braces indicate that the enclosed item (or one of the enclosed items separated from the other enclosed items by vertical bars) is required and <u>must</u> be specified.

[ ] Square brackets indicate that the enclosed item is optional.

... Ellipses indicate that other, similar items may follow as shown.

Nn The n-th numeric value that is associated with an input command.

All input data for CDFTBL are associated with a keyword. A keyword is composed of from one to eight characters. The characters can be any from A to Z in upper- or lowercase. A keyword always begins in the first column of an 80-column record. Table 4 summarizes the keywords for CDFTBL and their functions. The remainder of Section 5.0 gives detailed descriptions of the keywords.

Table 4. Keywords and Their Functions for the CDFTBL Code.

Keyword	Input function
DISTRIBU	Chooses the technique for generating the table of probabilities.
END	Denotes the end of keyword inputs.
PLOT	Writes a data file that can be used for plotting purposes.
PRINT	Prints tables of data at specified locations in the processing.
PROBABIL	Defines the probabilities for the output table of values.
SAMPLE	Defines the size of the input data set.
TRANSFOR	Performs transformations on the input data values.
USER	Specifies a name for the user.

The RDBLK routines, which are used to decode the keyword file, convert all alphabetic characters in keywords and modifiers to uppercase. While the casual user need not know this, anyone who modifies the program should be aware of it. The module CEXIST, which checks modifiers against input options, does not make the conversion, but requires uppercase input.

The RDBLK routines also assume that the input characters '0' through '9', the characters 'a' through 'z', and the characters 'A' through 'Z' have contiguous storage codes. If the operating system/compiler uses a non-ASCII storage code, the numeric and uppercase conversion routines may not function properly.

5.1 DISTRIBU COMMAND

5.1.1 Purpose

The **DISTRIBU** keyword is used to choose the type of distribution estimator to use when computing the table of probabilities.

5.1.2 Syntax

DISTRIBU {EMPIRICA | KERNEL} [ALIAS N1 N2 N3] [BANDWIDT N4] [SCALE N5] [STEPS N6]

Only one of the two modifiers EMPIRICA and KERNEL can be used. If both are entered, the program will terminate because of the input error. In the following description, the number of sample values is denoted by NSIZE (numeric value N1 from the **SAMPLE** keyword).

Four additional modifiers (ALIAS, BANDWIDT, SCALE, and STEPS) may be used if the KERNEL option is selected. The four modifiers can be selected in any combination, but must be entered on the keyword record in alphabetical order. The numeric values N1 through N6 described in Table 5 denote the relative positions of the numeric entries. Selection of options may not require a total of six numeric values. The casual user is not likely to ever invoke any of the three modifiers BANDWIDT, SCALE, or STEPS.

Table 5. **DISTRIBU** Keyword Numeric Value Description. (sheet 1 of 2)

Modifier/ numeric field	Numeric value	Default value	Value remarks and explanations
EMPIRICA	--	--	Computes the probability table using the empirical distribution function. No additional modifiers or numeric values are processed.
KERNEL	--	--	Computes the probability table by using the kernel density estimator. Four additional modifiers (ALIAS, BANDWIDT, SCALE, and STEPS) may be used.
ALIAS	N1,N2, N3	--	This modifier applies only if KERNEL was specified and controls aliasing of the density value. The entry N1 can be 1, 2, or 3. If N1=1, the left tail of the density is truncated at the value N2 and is folded over (aliased) onto the remaining domain. If N1=2, the right tail of the density is truncated at the value N2 and is aliased onto the remaining domain. If N1=3, the density is truncated between N2 and N3 and is aliased onto the remaining domain.
BANDWIDT	N4	[a]	This modifier applies only if KERNEL was specified. If BANDWIDT is entered, it must be followed by a cn value for the kernel density estimator (see Section 2.3).

Table 5. **DISTRIBU** Keyword Numeric Value Description. (sheet 2 of 2)

Modifier/ numeric field	Numeric value	Default value	Value remarks and explanations
SCALE	N5	[b]	This modifier applies only if KERNEL was specified. If SCALE is entered, it must be followed by a standard deviation estimate for the data set.
STEPS	N6	1000	This modifier applies only if KERNEL was specified. If STEPS is entered, it must be followed by the number of integration steps to use in the trapezoidal integration of the kernel density.

[a]If BANDWIDT is not entered, c_0 is set to 2.283*NSIZE**(-.287).
[b]If SCALE is not entered, a value is generated internally using the median absolute deviation.

5.1.3 Examples

The first example below uses the empirical distribution function approach to generating the table of probability values. The second example uses the kernel-density approach to generating the probability values and uses 400 integration steps in the process. The third example uses the kernel-density approach to generating the table of probabilities and truncates the left tail of the density at -2. Using this aliasing option, the density value at x=-2+dx, for dx>0, will be computed as f(-2+dx) + f(-2-dx) where f(*) denotes the density estimate that would be obtained without aliasing. The density value will be set to 0 if x is less than -2.

DISTRIBU EMPIRICAl
DISTRIBU KERNEL using 400 integration STEPS
DISTRIBU KERNEL ALIAS 1 left tail truncated at -2.0

5.2 END COMMAND

5.2.1 Purpose

The **END** keyword informs the program that all keywords have been processed. Anything entered in the input file after this keyword will be treated as data values. The END keyword is required for proper program execution.

5.2.2 Syntax

END

There are no modifiers or numeric values for this keyword.

5.2.3 Example

The following example signals the program that the user has finished entering keywords and is ready to enter data values.

END of the keywords

5.3 PLOT COMMAND

5.3.1 Purpose

The **PLOT** keyword will cause an output data file to be written that can be used in a general-purpose line-plotting program. No plots will be generated if this keyword is omitted.

5.3.2 Syntax

PLOT [CUMULATIV] [DENSITY] [TABLE]

Three modifiers are available with the plot card. If none of the modifiers are present, no data will be written to the plot file. The three modifiers shown in Table 6 can be present in any combination and may be entered in any order.

Table 6. **PLOT** Keyword Modifiers and Their Descriptions.

Modifier	Remarks and explanations
CUMULATI	If the modifier CUMULATI is used, the CDF from which the table values are extracted is written to the plot file.
DENSITY	If the modifier DENSITY is used, the probability density used in the kernel density estimator is written to the plot file. This option is operational only if the keyword DISTRIBU KERNEL has been entered.
TABLE	If the modifier TABLE is used, a copy of the probability table is written to the plot file.

5.3.3 Examples

The first example writes a plot file containing the table of probability values and the entire CDF. The second example writes a plot file containing the kernel density used to generate the table of probability values.

PLOT TABLE CUMULATIve
PLOT KERNEL density function

5.4 PRINT COMMAND

5.4.1 Purpose

The **PRINT** keyword allows the user to optionally print a table of data values to the report file to examine the data before and after transformations have been performed. No data tables will be printed if this keyword is omitted. The print option applies to the data set after it has been sorted in increasing algebraic order.

5.4.2 Syntax

PRINT [BEFORE] [AFTER] [RANGE N1 N2]

Tables of the data can be written to the report file at two possible locations, before and after transformations are performed. A subset (range) of the data, rather than the entire data set, can be written. Enter the modifier BEFORE if data tables are desired before transformations are performed. Enter the modifier AFTER if data tables are desired after transformations are performed. In Table 7, the number of sample values is denoted by NSIZE (numeric value N1 from the **SAMPLE** keyword).

Table 7. **PRINT** Keyword Numeric Value Explanations.

Numeric field	Numeric value	Default	Remark and explanations
N1	1 to NSIZE	1	Lowest index (of the sorted) data set to write in the data table. This value is processed only if the modifier RANGE is present on the input card.
N2	N1 to NSIZE	NSIZE	Highest index (of the sorted) data set to write in the data table. This value is processed only if the modifier RANGE is present on the input card.

5.4.3 Examples

The first example prints the entire data set before and after data transformations are performed. The second example prints out the first 20 data values after transformations are performed.

PRINT BEFORE and AFTER
PRINT AFTER transformations for the index RANGE (1,20)

5.5 PROBABIL COMMAND

5.5.1 Purpose

The **PROBABIL** keyword can be used to define probability levels in the generated table to be written to the report file. A default table will be used if this table is omitted.

5.5.2 Syntax

PROBABIL { TABLE | INCREMEN } {N1} [N2 ..... Nn]

The probability levels to be entered in the output table of cumulative probabilities can be defined three ways. First, a default table can be used. Second, a table can be defined by explicitly listing the desired probability levels. Third, a table can be defined by specifying a single probability increment between table entries.

A default set of probabilities has been implemented that uses 35 probability levels. Omit this keyword to choose the default set of values. If this keyword is entered, it overwrites all default values. The default probabilities are 0.0, 0.01, 0.015, 0.02, 0.025, 0.03, 0.04, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.925, 0.95, 0.96, 0.97, 0.975, 0.98, 0.985, 0.99, and 1.0. Other options are described in Table 8.

Table 8. PROBABIL Keyword Numeric Value Explanations.

Numeric field	Numeric value	Remarks and explanations
N1	>0	If the modifier TABLE is used, N1 denotes the number of probability levels to include in the table of cumulative probability values. If the modifier INCREMEN is used, N1 denotes the common probability increment between entries in the table of cumulative probability values. A value of 0.01 is suggested.
N2, N3...	0<Ni<1	The next values are read only if the modifier TABLE is used, then they define the probabilities in the output table. The first probability must be 0.0, and the last probability must be 1.0. Each succeeding value must be greater than the previous value. Exactly N1 probability values must be entered.

Because the table of probabilities and associated values output by this program will be used in a linear interpolation routine in other programs, at least 20 probability levels should be specified. If the data come from a

distribution that has a wide (mathematical) support, the extreme values in the data tend to be widely separated from the body of the data. Thus, more probability levels near 0 and 1 are needed to maintain interpolation accuracy. The default set of probabilities is suggested for this situation.

5.5.3 Examples

The following two examples define a probability table that has 21 entries for probabilities with probability increments of 0.05 between 0 and 1 inclusive.

PROBABILity TABLE 21 individual values 0.0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5 0.55 0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95 1.0

PROBABILity INCREMENt 0.05

5.6 SAMPLE COMMAND

5.6.1 Purpose

The **SAMPLE** keyword defines the number of values in the sample data set and how many values are on each line of the data file. This keyword is required for program execution.

5.6.2 Syntax

SAMPLE {N1} [LINE N2]

The presence of the modifier LINE in the input will cause the numeric value N2 to be read and processed; otherwise, N2 will be ignored. Table 9 gives additional explanations for the numeric values.

5.6.3 Examples

The first two examples define a data set that has 40 values, with 1 value per line of the input data file. The third example defines a data set with 20 values, entered with 5 values per line.

SAMPLE 40
SAMPLE 40 LINE 1
SAMPLE size 20 with 5 values per LINE

Table 9. Sample Keyword Numeric Value Explanations.

Numeric field	Numeric value	Default value	Remarks and explanations
N1	Positive	None	The number of data values that will be used for generating the probability values. A lower limit of 4 is enforced; however, at least 20 are suggested.
N2	Positive	1	This entry identifies the number of data values per line in the input data file. Each line in the input file must contain N2 data values. The last line will contain fewer values if the N1 is not a multiple of N2.

5.7 TRANSFOR COMMAND

5.7.1 Purpose

The purpose of the **TRANSFOR** keyword is to allow the user to perform common transformations on the input data set before generation of the CDF. No transformations will be done if this keyword is omitted.

5.7.2 Syntax

TRANSFOR {N1} [N2, N3]

This keyword invokes a data transformation option. The first numeric entry is a transformation index.

5.7.3 Examples

The first example invokes a logarithm (base 10) transformation on the input data. The second example does a linear transformation that shifts the data by subtracting -2 (adding 2) and dividing the result by 3.

TRANSFORm 1
TRANSFORm type 3 shift -2 divide by 3

Table 10. TRANSFOR Keyword Numeric Value Explanations.

Numeric field	Numeric value	Default value	Remarks and explanations
N1	0	0	No data transformation will be performed.
	1		A log base 10 transformation will be performed. This requires that all data values be positive.
	3		A linear transformation will be performed. Also required are two additional numerical values, N2 and N3. The transformation is Xnew = (Xold-N2)/N3.
N2	Any	None	Shift value in the linear transformation.
N3	Nonzero	None	Divisor in the linear transformation.

5.8 USER COMMAND

5.8.1 Purpose

The **USER** keyword can be used to define a new user name. A default user name of "XXXXXX X. XXXXXX" is defined in subroutine IDENC and will be used unless the user enters this keyword. The user name is written to all output files.

5.8.2 Syntax

USER {"New Name"}

The user name can contain up to 16 characters. The name to be entered must be contained between double quotation marks. The default user name is retained if an illegal user name is encountered.

5.8.3 Example

In this example the default user name is changed to John Doe.

USER "John Doe"

6.0 EXAMPLE RUNS AND CODE VERIFICATION

This section contains four example runs of the code. The examples exercise different features of the code, illustrate the free-form command language, and serve as verification cases for the code.

Two methods were used to verify the code: hand calculations and comparison with analytical solutions. Hand calculations were performed to test the following: (1) data transformations, (2) kernel-density estimates, (3) integration of the kernel density to obtain a CDF, (4) interpolation of the kernel-based CDF to form the output-probability table, (5) empirical CDF estimates, and (6) interpolation of the empirical CDF to form the output-probability table. The data sorting subroutine was verified by inspection on several data sets.

Several runs of the code were made with large data sets sampled from known probability distributions. For each of these runs, the computed probability table was compared to the analytical CDF for the selected probability distribution. In each case the results were within the range expected from sampling variation.

The CDFTBL code has eight keywords, several of which have multiple options. The code was run enough times to test each keyword individually, and most of the options in their proper combinations. An attempt was made to code exhaustive checks on the keywords to eliminate illegal or unreasonable option combinations.

6.1 EXAMPLE 1

Example 1 illustrates the kernel-density estimation technique on a sample of 200 data, which were obtained as a random sample from a standard normal distribution. Table 11 shows the input file, and Table 12 shows the report file.

This example can be used to help verify the implementation of the kernel density estimator. The accuracy of the output CDF table can be checked by comparing it to the analytical standard normal CDF (Strait 1989). The results show reasonable agreement for a sample of 200 data. Another way to examine the accuracy of the estimation technique is to determine if the generated PDF can be considered as being "close" to the PDF of the standard normal distribution using a metric such as the Hellinger metric (Eslinger and Woodward 1990). The PDF generated by this example is acceptable as a standard-normal PDF using this metric.

6.2 EXAMPLE 2

Example 2 illustrates the empirical distribution function technique on a sample of 200 data, which were obtained as a random sample from a standard normal distribution. The data used in this example are the same as those used in example 1. Table 13 shows the input file; Table 14 shows the report file.

Table 11. Input Data File for Example 1: EXAMPLE1.DAT.

```
!
! Input file EXAMPLE1.DAT for the CDFTBL User's Manual
!
DISTRIBUTION KERNEL
PROBABILITY INCREMENT 0.025
SAMPLE size 200 with 5 values per LINE
USER "Paul W. Eslinger"
END
    0.64085    0.22846   -1.14595   -1.03808    0.25602
    0.12138   -0.38882    0.45407    0.26864   -0.45454
    0.24989    0.65894    0.77606    0.47764   -0.38137
   -1.79454   -1.00094    0.54271   -0.27218    0.89912
    0.17183   -2.53104    0.17009   -0.10363    1.27442
   -0.96684    0.31062   -0.91674   -0.80962    1.98233
   -0.48006   -0.94326    0.95084    0.41019    0.22383
    1.10860   -0.81396   -1.63515   -0.49617   -0.55313
   -0.06211   -0.55425   -1.75557   -0.70514    0.70862
   -0.24745   -1.12485   -0.28577   -1.01844   -0.78787
   -2.80329    0.02076    0.55397   -0.39109    0.20761
    0.15881    1.10716    1.96530   -1.71948    0.60075
    0.91340   -1.36126   -0.16939   -1.08821    0.21064
   -0.03287   -0.97539    1.01676   -1.49732    0.43162
   -1.11690    0.97984    0.44495   -0.41358    0.06351
   -1.40432   -0.13249    0.71608   -0.76421   -0.26686
   -0.78544    1.59368   -1.02657    1.36543    0.29006
   -0.26005   -0.57094    0.01283    0.07222   -0.54039
   -1.84836   -0.12579   -0.49458    0.54947   -0.14454
    0.56585   -0.21985   -0.92563   -0.76253    0.92402
    0.71942    0.08398    1.32623   -0.41513    1.75942
   -0.59664   -0.82225   -0.60857    0.03159   -0.67760
   -0.91995   -1.34402    0.61865   -0.08889   -0.64575
    0.60308    1.07314   -1.49750   -0.58747    0.86685
   -1.60187    0.06867    0.19505   -1.42170    0.00776
    0.07418   -0.21813   -0.10331   -1.63859   -0.61013
    0.99004    0.79026    0.69863   -1.18345    0.03783
   -1.28231   -0.12131   -1.41141    0.79185    0.03560
   -1.19759    1.08136    1.32379    0.09148   -1.95512
    1.66648   -0.06710    1.36977   -1.82089    0.46820
    0.81387   -0.03919    0.90694   -0.13993   -0.52143
   -0.12445   -0.74095    1.33433    0.69509   -0.03186
    1.23301    0.32693   -0.94073    0.51258    0.22641
    0.49101   -1.23658   -0.84951   -0.23464    0.21056
    2.01105    0.13155   -2.39130   -1.36184    1.21370
   -0.33275    1.36228   -0.60063    1.09603   -0.53799
    1.69000   -1.66245   -0.57094    0.12209   -1.34285
    1.44965   -0.31835   -0.15303   -0.22814    2.54004
    1.06771    0.06645    0.51676   -1.04566   -1.54310
   -2.66981    0.76398   -0.10199    1.09158    1.29710
```

Table 12. Report File for Example 1: EXAMPLE1.RPT. (sheet 1 of 2)

```
1

*****************************************************************************
*
```

```
 CCCCCC    DDDDDDD    FFFFFFFFF  TTTTTTTTT  BBBBBBBB   LL
CC    CC   DD    DD   FFFFFFFFF  TTTTTTTTT  BB     BB  LL
CC         DD     DD  FF            TT      BB    BB   LL
CC         DD     DD  FFFFFF        TT      BBBBBBBB   LL
CC         DD     DD  FF            TT      BB    BB   LL
CC    CC   DD    DD   FF            TT      BB     BB  LLLLLLLLL
 CCCCCC    DDDDDDD    FF            TT      BBBBBBBB   LLLLLLLLL
```

CDFTBL 1.000
Last Modified on 13 Jun 1990

Cumulative Distribution Function Estimation
from a User Supplied Data Set

Statistical Program Developed by Paul W. Eslinger
Keyword Input Routines Developed by Dave W. Langford

Current Run ID = 900620202841 User Name = Paul W. Eslinger

System Date = 06/20/90 System Time = 20:28:41

```
*****************************************************************************
*
1 Program: CDFTBL   Version:1.000    User: Paul W. Eslinger Run ID:
900620202841
```

Cumulative Probabilities Using Kernel Density Estimation

The density estimate was based on 200 values.
The scale estimate was 1.0123E+00
The kernel bandwidth value was 0.49901
The probability table has 41 pairs of values.

Table 12. Report File for Example 1: EXAMPLE1.RPT. (sheet 2 of 2)

X	Cumulative Probability
-3.3084E+00	0.0000
-2.0634E+00	0.0250
-1.7398E+00	0.0500
-1.5541E+00	0.0750
-1.4040E+00	0.1000
-1.2746E+00	0.1250
-1.1587E+00	0.1500
-1.0515E+00	0.1750
-9.5368E-01	0.2000
-8.6287E-01	0.2250
-7.7701E-01	0.2500
-6.9466E-01	0.2750
-6.1485E-01	0.3000
-5.3799E-01	0.3250
-4.6409E-01	0.3500
-3.9337E-01	0.3750
-3.2639E-01	0.4000
-2.6194E-01	0.4250
-2.0023E-01	0.4500
-1.4036E-01	0.4750
-8.0744E-02	0.5000
-2.0594E-02	0.5250
3.9702E-02	0.5500
1.0029E-01	0.5750
1.6167E-01	0.6000
2.2467E-01	0.6250
2.9015E-01	0.6500
3.5856E-01	0.6750
4.3140E-01	0.7000
5.0850E-01	0.7250
5.9105E-01	0.7500
6.7759E-01	0.7750
7.6808E-01	0.8000
8.6153E-01	0.8250
9.5813E-01	0.8500
1.0627E+00	0.8750
1.1804E+00	0.9000
1.3191E+00	0.9250
1.4973E+00	0.9500
1.7759E+00	0.9750
3.0452E+00	1.0000

Table 13. Input Data File Example 2: EXAMPLE2.DAT.

```
!
! Input file EXAMPLE2.DAT for the CDFTBL User's Manual
!
DISTRIBUTION empirical
PROBABILITY INCREMENT 0.025
SAMPLE size 200 with 5 values per LINE
USER " aul W. Eslinger"
END
    0.64085    0.22846   -1.14595   -1.03808    0.25602
    0.12138   -0.38882    0.45407    0.26864   -0.45454
    0.24989    0.65894    0.77606    0.47764   -0.38137
   -1.79454   -1.00094    0.54271   -0.27218    0.89912
    0.17183   -2.53104    0.17009   -0.10363    1.27442
   -0.96684    0.31062   -0.91674   -0.80962    1.98233
   -0.48006   -0.94326    0.95084    0.41019    0.22383
    1.10860   -0.81396   -1.63515   -0.49617   -0.55313
   -0.06211   -0.55425   -1.75557   -0.70514    0.70862
   -0.24745   -1.12485   -0.28577   -1.01844   -0.78787
   -2.80329    0.02076    0.55397   -0.39109    0.20761
    0.15881    1.10716    1.96530   -1.71948    0.60075
    0.91340   -1.36126   -0.16939   -1.08821    0.21064
   -0.03287   -0.97539    1.01676   -1.49732    0.43162
   -1.11690    0.97984    0.44495   -0.41358    0.06351
   -1.40432   -0.13249    0.71608   -0.76421   -0.26686
   -0.78544    1.59368   -1.02657    1.36543    0.29006
   -0.26005   -0.57094    0.01283    0.07222   -0.54039
   -1.84836   -0.12579   -0.49458    0.54947   -0.14454
    0.56585   -0.21985   -0.92563   -0.76253    0.92402
    0.71942    0.08398    1.32623   -0.41513    1.75942
   -0.59664   -0.82225   -0.60857    0.03159   -0.67760
   -0.91995   -1.34402    0.61865   -0.08889   -0.64575
    0.60308    1.07314   -1.49750   -0.58747    0.86685
   -1.60187    0.06867    0.19505   -1.42170    0.00776
    0.07418   -0.21813   -0.10331   -1.63859   -0.61013
    0.99004    0.79026    0.69863   -1.18345    0.03783
   -1.28231   -0.12131   -1.41141    0.79185    0.03560
   -1.19759    1.08136    1.32379    0.09148   -1.95512
    1.66648   -0.06710    1.36977   -1.82089    0.46820
    0.81387   -0.03919    0.90694   -0.13993   -0.52143
   -0.12445   -0.74095    1.33433    0.69509   -0.03186
    1.23301    0.32693   -0.94073    0.51258    0.22641
    0.49101   -1.23658   -0.84951   -0.23464    0.21056
    2.01105    0.13155   -2.39130   -1.36184    1.21370
   -0.33275    1.36228   -0.60063    1.09603   -0.53799
    1.69000   -1.66245   -0.57094    0.12209   -1.34285
    1.44965   -0.31835   -0.15303   -0.22814    2.54004
    1.06771    0.06645    0.51676   -1.04566   -1.54310
   -2.66981    0.76398   -0.10199    1.09158    1.29710
```

Table 14. Report File for Example 2: EXAMPLE2.RPT. (sheet 1 of 2)

```
1

*****************************************************************************
*

 CCCCCC    DDDDDDD    FFFFFFFFF  TTTTTTTTT  BBBBBBBB   LL
CC    CC   DD    DD   FFFFFFFFF  TTTTTTTTT  BB     BB  LL
CC         DD     DD  FF            TT      BB    BB   LL
CC         DD     DD  FFFFFF        TT      BBBBBBBB   LL
CC         DD     DD  FF            TT      BB    BB   LL
CC    CC   DD    DD   FF            TT      BB     BB  LLLLLLLLL
 CCCCCC    DDDDDDD    FF            TT      BBBBBBBB   LLLLLLLLL

                    CDFTBL   1.000
                Last Modified on 13 Jun 1990


            Cumulative Distribution Function Estimation
                  from a User Supplied Data Set

          Statistical Program Developed by Paul W. Eslinger
        Keyword Input Routines Developed by Dave W. Langford


      Current Run ID = 900620202852   User Name = Paul W. Eslinger

           System Date = 06/20/90   System Time = 20:28:52


*****************************************************************************
*
1 Program:  CDFTBL   Version:1.000    User:  Paul W. Eslinger Run ID:
900620202852


     Cumulative Probabilities Using The Empirical Distribution

      The density estimate was based on   200 values.
      The probability table has  41 pairs of values.
```

Table 14. Report File for Example 2: EXAMPLE2.RPT. (sheet 2 of 2)

Cumulative X	Probability
-2.8033E+00	0.0000
-1.9551E+00	0.0250
-1.7195E+00	0.0500
-1.5431E+00	0.0750
-1.4043E+00	0.1000
-1.2823E+00	0.1250
-1.1249E+00	0.1500
-1.0266E+00	0.1750
-9.4326E-01	0.2000
-8.4951E-01	0.2250
-7.8544E-01	0.2500
-6.7760E-01	0.2750
-5.9664E-01	0.3000
-5.5313E-01	0.3250
-4.9458E-01	0.3500
-3.9109E-01	0.3750
-2.8577E-01	0.4000
-2.3464E-01	0.4250
-1.5303E-01	0.4500
-1.2445E-01	0.4750
-8.8890E-02	0.5000
-3.1860E-02	0.5250
3.5600E-02	0.5500
7.2220E-02	0.5750
1.2209E-01	0.6000
1.9505E-01	0.6250
2.2641E-01	0.6500
2.9006E-01	0.6750
4.4495E-01	0.7000
5.1258E-01	0.7250
5.6585E-01	0.7500
6.5894E-01	0.7750
7.1942E-01	0.8000
8.1387E-01	0.8250
9.2402E-01	0.8500
1.0677E+00	0.8750
1.1072E+00	0.9000
1.2971E+00	0.9250
1.3654E+00	0.9500
1.6900E+00	0.9750
2.5400E+00	1.0000

This example can be used to help verify the implementation of the empirical distribution-function estimator. The accuracy of the output CDF table can be checked by comparing these results to the analytical standard-normal CDF (Strait 1989). The results show reasonable agreement for a sample of 200 data.

6.3 EXAMPLE 3

Example 3 illustrates the kernel-density estimation technique on a sample of 50 data, which were obtained as a random sample from a standard uniform distribution. A data transformation is performed to make simulated data from a uniform (-2,2) distribution. Table 15 shows the input file, and Table 16 shows the report file.

This example can be used to help verify the implementation of the data-transformation option and the kernel-density estimator. The data-transformation option can be verified by hand calculations because the data are written out after the transformation is performed. The analytical CDF for the uniform distribution is a straight line with a slope of 0.5 on the interval (-2,2). The results show reasonable agreement with the analytical CDF for a sample of 50 data.

6.4 EXAMPLE 4

Example 4 illustrates the kernel-density estimation technique on a sample of 20 data, which were obtained as a random sample from a standard uniform distribution. The ALIAS option is invoked because the density estimate should be zero outside the interval (0,1). A plot file is generated. The STEPS option is used simply to reduce the size of the plot file. The impact of the reduced number of integration steps is that the CDF integrates to 1.0028 instead of 1.0. Table 17 shows the input file, Table 18 shows the report file, and Table 19 shows the plot file.

This example can also be used to help verify the implementation of the kernel-density estimator. The ALIAS option correctly forces the CDF to have a value of zero for data below zero, and a value of one for data above one. If this example were run without the ALIAS option, the output CDF table would assign a nonzero probability outside the unit interval. The analytical CDF for the uniform distribution is a straight line with a slope of one on the unit interval. The results show reasonable agreement with the analytical CDF for a sample of 20 data, although 20 data are generally too few to obtain a density estimate that closely matches an analytical density.

Table 15. Input Data File Example 3: EXAMPLE3.DAT.

```
!
! Input file EXAMPLE3.DAT for the CDFTBL User's Manual
!
DISTRIBUTION KERNEL
PROBABILITY INCREMENT 0.05
TRANSFORM THE DATA TYPE 3 SHIFT BY 0.5 THEN DIVIDE BY 0.25
SAMPLE size 50 with 5 values per LINE
PRINT THE DATA BOTH BEFORE AND AFTER TRANSFORMATIONS
USER "Paul W. Eslinger"
END
    0.73919    0.59036    0.12591    0.14962    0.60103
    0.54830    0.34870    0.67511    0.60590    0.32472
    0.59866    0.74503    0.78114    0.68355    0.35146
    0.03636    0.15843    0.70634    0.39274    0.81571
    0.56822    0.00569    0.56753    0.45873    0.89874
    0.16681    0.62195    0.17964    0.20908    0.97628
    0.31559    0.17277    0.82916    0.65917    0.58856
    0.86620    0.20783    0.05101    0.30989    0.29009
    0.47524    0.28970    0.03958    0.24036    0.76072
    0.40228    0.13033    0.38753    0.15423    0.21539
```

Table 16. Report File for Example 3: EXAMPLE3.RPT. (sheet 1 of 3)

```
1

*******************************************************************************
*


 CCCCCC    DDDDDDD    FFFFFFFFF  TTTTTTTTT  BBBBBBBB   LL
CC    CC  DD    DD   FFFFFFFFF  TTTTTTTTT  BB     BB  LL
CC        DD     DD  FF            TT      BB    BB   LL
CC        DD     DD  FFFFFF        TT      BBBBBBBB   LL
CC        DD     DD  FF            TT      BB    BB   LL
CC    CC  DD    DD   FF            TT      BB     BB  LLLLLLLLL
 CCCCCC   DDDDDDD    FF            TT      BBBBBBBB   LLLLLLLLL


                CDFTBL   1.000
            Last Modified on 13 Jun 1990


        Cumulative Distribution Function Estimation
              from a User Supplied Data Set

       Statistical Program Developed by Paul W. Eslinger
     Keyword Input Routines Developed by Dave W. Langford


   Current Run ID = 900620202859   User Name = Paul W. Eslinger

        System Date = 06/20/90   System Time = 20:28:59
```

Table 16. Report File for Example 3: EXAMPLE3.RPT. (sheet 2 of 3)

```
*****************************************************************************
*
1 Program:  CDFTBL   Version:1.000    User:  Paul W. Eslinger Run ID:
900620202859

     Data Used in Generating the Probability Table
              Before Data Transformations

   5.69000E-03   3.63600E-02   3.95800E-02   5.10100E-02   1.25910E-01
   1.30330E-01   1.49620E-01   1.54230E-01   1.58430E-01   1.66810E-01
   1.72770E-01   1.79640E-01   2.07830E-01   2.09080E-01   2.15390E-01
   2.40360E-01   2.89700E-01   2.90090E-01   3.09890E-01   3.15590E-01
   3.24720E-01   3.48700E-01   3.51460E-01   3.87530E-01   3.92740E-01
   4.02280E-01   4.58730E-01   4.75240E-01   5.48300E-01   5.67530E-01
   5.68220E-01   5.88560E-01   5.90360E-01   5.98660E-01   6.01030E-01
   6.05900E-01   6.21950E-01   6.59170E-01   6.75110E-01   6.83550E-01
   7.06340E-01   7.39190E-01   7.45030E-01   7.60720E-01   7.81140E-01
   8.15710E-01   8.29160E-01   8.66200E-01   8.98740E-01   9.76280E-01
1 Program:  CDFTBL   Version:1.000    User:  Paul W. Eslinger Run ID:
900620202859

     Data Used in Generating the Probability Table
              After Data Transformations

  -1.97724E+00  -1.85456E+00  -1.84168E+00  -1.79596E+00  -1.49636E+00
  -1.47868E+00  -1.40152E+00  -1.38308E+00  -1.36628E+00  -1.33276E+00
  -1.30892E+00  -1.28144E+00  -1.16868E+00  -1.16368E+00  -1.13844E+00
  -1.03856E+00  -8.41200E-01  -8.39640E-01  -7.60440E-01  -7.37640E-01
  -7.01120E-01  -6.05200E-01  -5.94160E-01  -4.49880E-01  -4.29040E-01
  -3.90880E-01  -1.65080E-01  -9.90400E-02   1.93200E-01   2.70120E-01
   2.72880E-01   3.54240E-01   3.61440E-01   3.94640E-01   4.04120E-01
   4.23600E-01   4.87800E-01   6.36680E-01   7.00440E-01   7.34200E-01
   8.25360E-01   9.56760E-01   9.80120E-01   1.04288E+00   1.12456E+00
   1.26284E+00   1.31664E+00   1.46480E+00   1.59496E+00   1.90512E+00
1 Program:  CDFTBL   Version:1.000    User:  Paul W. Eslinger Run ID:
900620202859


     Cumulative Probabilities Using Kernel Density Estimation

      The density estimate was based on    50 values.
      The scale estimate was  1.3115E+00
      The kernel bandwidth value was 0.74285
      The probability table has  21 pairs of values.
```

Table 16. Report File for Example 3: EXAMPLE3.RPT. (sheet 3 of 3)

X	Cumulative Probability
-2.9515E+00	0.0000
-1.9806E+00	0.0500
-1.7084E+00	0.1000
-1.5002E+00	0.1500
-1.3219E+00	0.2000
-1.1584E+00	0.2500
-9.9828E-01	0.3000
-8.3467E-01	0.3500
-6.6190E-01	0.4000
-4.7667E-01	0.4500
-2.8259E-01	0.5000
-8.8112E-02	0.5500
1.0270E-01	0.6000
2.9297E-01	0.6500
4.8042E-01	0.7000
6.6569E-01	0.7500
8.5211E-01	0.8000
1.0516E+00	0.8500
1.2886E+00	0.9000
1.6242E+00	0.9500
2.8794E+00	1.0000

Table 17. Input Data File Example 4: EXAMPLE2.DAT.

```
!
! Input file EXAMPLE4.DAT for the CDFTBL User's Manual
!
DISTRIBUTION KERNEL ALIAS 3 (0.0,1.0) STEPS 100
$   The 100 steps are used simply to reduce the
$   size of the file in the User's Manual.
$   A value of 500 or larger is recommended.
$
PROBABILITY INCREMENT 0.025
SAMPLE size 20 with 2 values per LINE
USER "Paul W. Eslinger"
PLOT CUMULATIVE and TABLE
END
    0.73919    0.59036
    0.54830    0.34870
    0.59866    0.74503
    0.03636    0.15843
    0.56822    0.00569
    0.16681    0.62195
    0.31559    0.17277
    0.86620    0.20783
    0.86620    0.20783
    0.47524    0.28970
    0.40228    0.13033
```

Table 18. Report File for Example 4: EXAMPLE4.RPT. (sheet 1 of 2)

```
1

*****************************************************************************
*

         CCCCCC    DDDDDDD    FFFFFFFFF  TTTTTTTTT  BBBBBBBB   LL
        CC    CC  DD    DD   FFFFFFFFF  TTTTTTTTT  BB     BB  LL
        CC        DD     DD  FF            TT      BB    BB   LL
        CC        DD     DD  FFFFFF        TT      BBBBBBBB   LL
        CC        DD     DD  FF            TT      BB    BB   LL
        CC    CC  DD    DD   FF            TT      BB     BB  LLLLLLLLL
         CCCCCC   DDDDDDD    FF            TT      BBBBBBBB   LLLLLLLLL

                          CDFTBL   1.000
                     Last Modified on 13 Jun 1990


                 Cumulative Distribution Function Estimation
                        from a User Supplied Data Set

               Statistical Program Developed by Paul W. Eslinger
             Keyword Input Routines Developed by Dave W. Langford


           Current Run ID = 900620203830   User Name = Paul W. Eslinger

                System Date = 06/20/90   System Time = 20:38:30


*****************************************************************************
1 Program:  CDFTBL   Version:1.000    User:  Paul W. Eslinger Run ID:
900620203830


     Cumulative Probabilities Using Kernel Density Estimation

      The density estimate was based on    20 values.
      The scale estimate was  3.0698E-01
      The kernel bandwidth value was 0.96630
      The probability table has  41 pairs of values.
```

Table 18. Report File for Example 4: EXAMPLE4.RPT. (sheet 2 of 2)

X	Cumulative Probability
0.0000E+00	0.0000
1.4753E-02	0.0250
3.4027E-02	0.0500
5.3258E-02	0.0750
7.2563E-02	0.1000
9.2071E-02	0.1250
1.1181E-01	0.1500
1.3174E-01	0.1750
1.5179E-01	0.2000
1.7184E-01	0.2250
1.9199E-01	0.2500
2.1233E-01	0.2750
2.3309E-01	0.3000
2.5455E-01	0.3250
2.7685E-01	0.3500
2.9969E-01	0.3750
3.2258E-01	0.4000
3.4504E-01	0.4250
3.6704E-01	0.4500
3.8887E-01	0.4750
4.1091E-01	0.5000
4.3360E-01	0.5250
4.5721E-01	0.5500
4.8139E-01	0.5750
5.0568E-01	0.6000
5.2994E-01	0.6250
5.5421E-01	0.6500
5.7876E-01	0.6750
6.0336E-01	0.7000
6.2778E-01	0.7250
6.5213E-01	0.7500
6.7646E-01	0.7750
7.0105E-01	0.8000
7.2633E-01	0.8250
7.5288E-01	0.8500
7.8158E-01	0.8750
8.1344E-01	0.9000
8.5052E-01	0.9250
8.9345E-01	0.9500
9.4035E-01	0.9750
1.0000E+00	1.0000

Table 19. Plot File for Example 4: EXAMPLE4.PLT. (sheet 1 of 3)

1 Program: CDFTBL Version:1.000 User: Paul W. Eslinger Run ID: 900620203830

Cumulative Distribution Function

-2.90946E-01	0.00000
-2.76262E-01	0.00000
-2.61577E-01	0.00000
-2.46892E-01	0.00000
-2.32208E-01	0.00000
-2.17523E-01	0.00000
-2.02838E-01	0.00000
-1.88154E-01	0.00000
-1.73469E-01	0.00000
-1.58784E-01	0.00000
-1.44100E-01	0.00000
-1.29415E-01	0.00000
-1.14730E-01	0.00000
-1.00046E-01	0.00000
-8.53610E-02	0.00000
-7.06763E-02	0.00000
-5.59916E-02	0.00000
-4.13069E-02	0.00000
-2.66223E-02	0.00000
-1.19376E-02	0.00000
2.74708E-03	0.00948
1.74318E-02	0.02846
3.21164E-02	0.04752
4.68011E-02	0.06662
6.14858E-02	0.08568
7.61705E-02	0.10466
9.08551E-02	0.12346
1.05540E-01	0.14210
1.20224E-01	0.16061
1.34909E-01	0.17896
1.49594E-01	0.19726
1.64279E-01	0.21560
1.78963E-01	0.23387
1.93648E-01	0.25205
2.08333E-01	0.27013
2.23017E-01	0.28799
2.37702E-01	0.30550
2.52387E-01	0.32255
2.67071E-01	0.33915
2.81756E-01	0.35544
2.96441E-01	0.37147
3.11125E-01	0.38744
3.25810E-01	0.40354
3.40495E-01	0.41987
3.55179E-01	0.43646
3.69864E-01	0.45323
3.84549E-01	0.47006

Table 19. Plot File for Example 4: EXAMPLE4.PLT. (sheet 2 of 3)

3.99233E-01	0.48682
4.13918E-01	0.50339
4.28603E-01	0.51963
4.43287E-01	0.53542
4.57972E-01	0.55080
4.72657E-01	0.56599
4.87341E-01	0.58114
5.02026E-01	0.59625
5.16711E-01	0.61134
5.31395E-01	0.62650
5.46080E-01	0.64166
5.60765E-01	0.65672
5.75449E-01	0.67164
5.90134E-01	0.68654
6.04819E-01	0.70149
6.19503E-01	0.71651
6.34188E-01	0.73157
6.48873E-01	0.74665
6.63558E-01	0.76175
6.78242E-01	0.77683
6.92927E-01	0.79181
7.07612E-01	0.80660
7.22296E-01	0.82111
7.36981E-01	0.83525
7.51666E-01	0.84892
7.66350E-01	0.86203
7.81035E-01	0.87455
7.95720E-01	0.88650
8.10404E-01	0.89781
8.25089E-01	0.90841
8.39774E-01	0.91824
8.54458E-01	0.92748
8.69143E-01	0.93628
8.83828E-01	0.94469
8.98512E-01	0.95279
9.13197E-01	0.96069
9.27882E-01	0.96847
9.42566E-01	0.97616
9.57251E-01	0.98375
9.71936E-01	0.99130
9.86620E-01	0.99895
1.00131E+00	1.00280
1.01599E+00	1.00280
1.03067E+00	1.00280
1.04536E+00	1.00280
1.06004E+00	1.00280
1.07473E+00	1.00280
1.08941E+00	1.00280
1.10410E+00	1.00280
1.11878E+00	1.00280
1.13347E+00	1.00280
1.14815E+00	1.00280

Table 19. Plot File for Example 4: EXAMPLE4.PLT. (sheet 3 of 3)

```
1 Program:  CDFTBL   Version:1.000    User:  Paul W. Eslinger Run ID:
900620203830
 Cumulative Probability Table
   41
  1.16284E+00   1.00280
  0.00000E+00   0.00000
  1.47529E-02   0.02500
  3.40268E-02   0.05000
  5.32582E-02   0.07500
  7.25625E-02   0.10000
  9.20708E-02   0.12500
  1.11807E-01   0.15000
  1.31743E-01   0.17500
  1.51786E-01   0.20000
  1.71838E-01   0.22500
  1.91990E-01   0.25000
  2.12335E-01   0.27500
  2.33091E-01   0.30000
  2.54555E-01   0.32500
  2.76852E-01   0.35000
  2.99690E-01   0.37500
  3.22580E-01   0.40000
  3.45039E-01   0.42500
  3.67036E-01   0.45000
  3.88874E-01   0.47500
  4.10914E-01   0.50000
  4.33598E-01   0.52500
  4.57206E-01   0.55000
  4.81387E-01   0.57500
  5.05675E-01   0.60000
  5.29939E-01   0.62500
  5.54213E-01   0.65000
  5.78760E-01   0.67500
  6.03355E-01   0.70000
  6.27784E-01   0.72500
  6.52130E-01   0.75000
  6.76461E-01   0.77500
  7.01055E-01   0.80000
  7.26335E-01   0.82500
  7.52880E-01   0.85000
  7.81584E-01   0.87500
  8.13443E-01   0.90000
  8.50521E-01   0.92500
  8.93446E-01   0.95000
  9.40351E-01   0.97500
  1.00000E+00   1.00000
```

7.0 REFERENCES

Beran, R., 1977, "Minimum Hellinger Distance Estimates for Parametric Models," *The Annals of Statistics*, Vol. 5, No. 3, pp. 445-463.

Devroye, L., 1987, *A Course in Density Estimation*, Birkhauser, Boston, Massachusetts.

Eslinger, P. W., and W. A. Woodward, 1990, "Minimum Hellinger Distance Estimation for Normal Models," in *Journal of Statistical Computation and Simulation*.

Epanechnikov, V. A., 1969, "Non-parametric Estimation of a Multivariate Probability Density," *Theory of Probability and its Applications*, Vol. XIV, No. 1, pp. 153-158.

Lahey, 1987, *Reference Manual - Fortran 77 Language System for the Personal Computer*, Lahey Computer Systems, Inc., Incline Village, Nevada.

Lewis, P. A., A. S. Goodman, and J. M. Miller, 1969, "A Pseudo-random Number Generator for the System/360," *IBM Systems Journal*, Vol. 8, No. 2, pp. 136-146.

Mood, A. M., F. A. Graybill, and D. C. Boes, 1974, *Introduction to the Theory of Statistics*, Third Edition, McGraw-Hill Book Co., New York, New York.

Parzen, E., 1962, "On the Estimation of a Probability Density Function and the Mode," *Annals of Mathematical Statistics*, Vol. 33, pp. 1065-1076.

Runchal, A. K. and B. Sagar, 1989, *PORFLO-3: A Mathematical Model for Fluid Flow, Heat, and Mass Transport in Variably Saturated Geologic Media*, Users Manual, Version 1.0, WHC-EP-0041, Westinghouse Hanford Company, Richland, Washington.

Serfling, R. J., 1980, *Approximation Theorems of Mathematical Statistics*, John Wiley and Sons, New York, New York.

Strait, P. T., 1989, *A First Course in Probability and Statistics with Applications*, Second Edition, Harcourt Brace Jonanovich, San Diego, California.

This page intentionally left blank.

APPENDIX A

SOURCE CODE LISTING

This page intentionally left blank.

APPENDIX A

SOURCE CODE LISTING

This appendix contains a source code listing (Table A-1). The source code is listed as if it were ready for processing by the Cray UPDATE processor. The differences between this method and the usual FORTRAN compiler methods are (1) all of the source is contained in a single file, (2) items that would be 'INCLUDE' or 'INSERT' decks are identified by the '*COMDECK name', (3) items that would be separate FORTRAN subroutines or files are identified by '*DECK name', and (4) when '*CALL name' is encountered, it means to insert the COMDECK with that given name at the same location.

The listing given uses a version of subroutine IDENC, which is designed for the Lahey (Lahey 1987) FORTRAN 77 compiler. Table A-2 is a separate version of the subroutine (IDENC.VAX) that runs a VAX under the VMS operating system.

Table A-1. Source Code Listing. (sheet 1 of 62)

```
*COMDECK CDATA
C++++ Insert Deck CDATA ++++++++++++++++++++++++++++++++++++++++++++++++
C  History:  Paul W. Eslinger :  15 Jan 1990
C  Purpose:  Variables needed by the RDBLK routines
C  Variables:
C    CVALUE :  Vector of secondary keywords (modifiers) read from the
C              current keyword card
C    ILINE  :  Line pointer for the input keyword file
C    INFO   :  Character variable containing the input line to be
C              decoded by the RDBLK routines.
C    NAME   :  Name of the current keyword
C    NCVALU :  Number of secondary keywords (modifiers) contained in
C              the current keyword card
C    NKEYS  :  Maximum number of secondary keywords (modifiers) which
C              can be contained on a single keyword card
C    NVALS  :  Maximum number of numeric values which can be read from a
C              single keyword card
C    NVALUE :  Number of data values read from the current keyword card
C    VALUE  :  Numeric values read from a single keyword card
C
      PARAMETER (NKEYS =  50)
      PARAMETER (NVALS = 501)
C
      CHARACTER INFO*1, CVALUE*8, NAME*8
      COMMON /CDAT1/ NAME, INFO(80), CVALUE(NKEYS)
C
      COMMON /CDAT2/ ILINE, NVALUE, NCVALU, VALUE(NVALS)
C
C++++ Insert Deck CDATA ++++++++++++++++++++++++++++++++++++++++++++++++
*COMDECK FILES
C++++ START INSERT FILES ++++++++++++++++++++++++++++++++++++++++++++++
```

Table A-1. Source Code Listing. (sheet 2 of 62)

```
C  History:  Paul W. Eslinger :  15 Jan 1990
C  Purpose:  Variables needed for file handling
C  Variables:
C    IDAT   :  Unit number for FILE7
C    INFILE :  Input data file name (without file extension)
C    IRPT   :  Unit number for FILE8
C    IPLT   :  Unit number for FILE9
C    FILE7  :  Input data file name
C    FILE8  :  Report file name
C    FILE9  :  Plot file name
C
      CHARACTER*64 INFILE, FILE7, FILE8, FILE9
      COMMON /FILEC/ INFILE, FILE7, FILE8, FILE9
      COMMON /FILEN/ IDAT, IRPT, IPLT
C++++ END INSERT FILES ++++++++++++++++++++++++++++++++++++++++++++++++
*COMDECK IDENC
C++++ Start insert IDENC ++++++++++++++++++++++++++++++++++++++++++++++
C  History:  Paul W. Eslinger :  15 Jan 1990
C  Purpose:  Program identification
C  Variables:
C    CRUNID :  Run identification number
C    PRGDAT :  Program modification date
C    PRGNAM :  Program name
C    PRGVER :  Program version number
C    SYSDAT :  System date
C    SYSTIM :  System time
C    USRNAM :  User name
C
      CHARACTER*8  PRGNAM, PRGVER, SYSDAT, SYSTIM
      CHARACTER*12 PRGDAT, CRUNID
      CHARACTER*16 USRNAM
      COMMON /IDENC/ CRUNID, PRGNAM, PRGVER, PRGDAT, USRNAM,
     *               SYSDAT, SYSTIM
C
C++++ End insert IDENC ++++++++++++++++++++++++++++++++++++++++++++++++
*COMDECK SDATA
C++++ START INSERT SDATA ++++++++++++++++++++++++++++++++++++++++++++++
C  History:  Paul W. Eslinger :  15 Jan 1990
C  Purpose:  Data
C  Variables:
C    ALPHA  :  Probability levels for CDF Computation
C    BADKEY :  Logical flag which checks for bad keywords being
C              entered.
C               .TRUE. = at least one bad keyword was found,
C               .FALSE. = no bad keywords were found
C    CNE    :  Bandwidth parameter for kernel density estimator
C    DOPLOT :  Logical flag vector which controls generation of the
C              plot file. .TRUE. = write plot file, .FALSE. = no write
C    DOPRNT :  Logical flag vector for printing data values to the
C              output file.  .FALSE. = Do not print, .TRUE. = Print
C    FINT   :  Vector (temporary) containing the integral of the
```

Table A-1. Source Code Listing. (sheet 3 of 62)

```
C                 kernel density
C     GHAT   :    Vector containing the kernel density at the locations
C                 in the vector XINT
C     IALIAS :    Flag for aliasing on the kernel density
C     ALEFT  :    Left alias value for the kernel density
C     ARIGHT :    Right alias value for the kernel density
C     ICDF   :    Flag for type of table generation
C                  1 = Integrated kernel density
C                  2 = Empirical distribution function
C     IPRNT  :    Vector of length 2 containing bounding sample indices
C                 for the data write in subroutine PRNTER
C     ISCA   :    Initial guess for the standard deviation used by the
C                 kernel density estimation routine
C     ITRANS :    Flag for type of data transformations
C     MAXDAT :    Maximum number of data values
C     MAXINT :    Maximum number of integration steps
C     MXALP  :    Maximum number of alpha levels (number of probability
C                 levels at which to compute the cumulative distribution
C                 function and write it out in a table
C     NALPHA :    Number of alpha (probability) levels for CDF computation
C     NLINE  :    Number of data values per line in input file
C     NSIZE  :    Number of data values to use
C     NSTP   :    Number of integration steps to use in the kernel density
C                 integration
C     SN     :    Scale parameter used in the kernel density estimator
C     STEMP  :    Vector of the same size as SVAL used for work space
C     SVAL   :    Vector containing the data values
C     TVAL   :    Vector of length 2 containing parameters to be used
C                 in data transformations
C     XDEL   :    Delta spacing for integration between values in XINT
C     XINT   :    Vector of X locations for the trapezoidal integration
C                 of the kernel density
C     XMIN   :    Minimum integration limit for the kernel density
C     XMAX   :    Maximum integration limit for the kernel density
C     XPLOT  :    Vector containing the X values associated with the
C                 values in ALPHA
C
      PARAMETER ( MAXDAT = 1000 )
      PARAMETER ( MAXINT = 2000 )
      PARAMETER ( MXALP  =  500 )
C
      LOGICAL DOPLOT, BADKEY, DOPRNT
      COMMON /SCONT/ NSTP, NLINE, NSIZE, ITRANS, ICDF, BADKEY,
     *               DOPRNT(2), ALPHA(MXALP), DOPLOT(3), XPLOT(MXALP),
     *               NALPHA, TVAL(2), IPRNT(2), IALIAS, ALEFT, ARIGHT
C
      REAL ISCA
      COMMON /SDATA/ SVAL(MAXDAT), STEMP(MAXDAT), XINT(MAXINT),
     *               FINT(MAXINT), GHAT(MAXINT), CNE, SN, XDEL,
     *               ISCA, XMIN, XMAX
C++++ END INSERT SDATA ++++++++++++++++++++++++++++++++++++++++++++++++
```

Table A-1. Source Code Listing. (sheet 4 of 62)

```
*DECK CDFTBL
      PROGRAM CDFTBL
C*....PROGRAM CDFTBL
C***********************************************************************
C*  Purpose:
C*
C*    This program will compute the cumulative distribution function
C*    from a data set and output the results in the form of a table.
C*    Options include using the empirical distribution function or
C*    a kernel density estimator using the Epanechnikov kernel.
C*
C*    The sample values are input through a data file.  The data file
C*    also contains keywords which control the program execution.
C*    A list of keyword definitions can be found in subroutine INPUTS.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger :  01 Jun 1990 :  Version 1.0
C*
C*
C*  References:
C*
C*    Eslinger, P. W., 1990, "CDFTBL:  A Statistical Program for
C*    Generating Cumulative Distribution Functions from Field Data,"
C*    WHC-EP-0418, Westinghouse Hanford Company, Richland, Washington.
C*
C*
C*  Background References - Kernel Density Estimation:
C*
C*    Parzen, E., 1962, "On the Estimation of a Probability
C*    Density Function and the Mode," Annals of Mathematical
C*    Statistics," Vol. 33, pp. 1065-1076.
C*
C*    Epanechnikov, V. A., 1969, "Non-parametric Estimation of a
C*    Multivariate Probability Density," Theory of Probability and
C*    its Applications, Vol. XIV, No. 1, pp. 153-158.
C*
C*c
C*  Background References - Empirical Distribution Function Estimation:
C*
C*    Mood, A. M., F. A. Graybill, D. C. Boes, 1974, "Introduction
C*    to the Theory of Statistics", McGraw-Hill, Inc.,
C*    New York, New York, p. 264.
C*
C*
C*  Acknowledgment:
C*
C*    The RDBLK routines used to decode the keyword inputs in the
C*    data file were written by Dave Langford of Boeing Computer
C*    Services Richland (BCSR), Richland, Washington.
```

Table A-1. Source Code Listing. (sheet 5 of 62)

```
C*
C***********************************************************************
C
*CALL SDATA
*CALL FILES
C
C *** Get identification variables
C
      CALL IDENC
C
C *** Open the input and output files
C
      CALL OPENER
C
C *** Read the input control variables
C
      CALL INPUTS
C
C *** Print a banner page to the report file
C
      CALL BANNER
C
C *** Sort the data set in increasing algebraic order
C
      KFLAG = 1
      CALL SORT( SVAL, NSIZE, KFLAG )
C
C *** Echo the data to the report file if requested
C
      IF( DOPRNT(1) ) THEN
        IPNDX = 1
        CALL PRNTER( IPNDX )
      ENDIF
C
C *** Check for error abort with all identical data
C
      IF( SVAL(1) .EQ. SVAL(NSIZE) ) THEN
        WRITE(IRPT,1000)
 1000 FORMAT(/
     * ' *** Terminal error in program CDFTBL ***'/
     * '     The data set consisted of all identical values.')
        STOP
      ENDIF
C
C *** Adjust for ties if using the empirical distribution function
C
      IF( ICDF .EQ. 2 ) THEN
        CALL TIES
      ENDIF
C
```

Table A-1. Source Code Listing. (sheet 6 of 62)

```
C *** Check for, and compute, optional data transformations
C
      CALL TRANS
C
C *** Echo the data to the report file if requested
C
      IF( DOPRNT(2) ) THEN
        IPNDX = 2
        CALL PRNTER( IPNDX )
      ENDIF
C
C *** Compute the CDF estimates
C
      IF( ICDF .EQ. 1 ) THEN
        IF( ISCA .LE. 0.0 ) CALL SCALE
        CALL EPAN
      ENDIF
C
      IF( ICDF .EQ. 2 ) THEN
        CALL EMPIR
      ENDIF
C
C *** Write out the results
C
      CALL WRITER
C
      STOP
      END
*DECK BANNER
      SUBROUTINE BANNER
C*....SUBROUTINE BANNER
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine prints a banner page to the report file.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger :  13 Jun 1990 :  Original source
C*
C***********************************************************************
C
*CALL IDENC
*CALL FILES
C
      CHARACTER*64 HEAD(7)
C
C *** Define the header
C
```

Table A-1. Source Code Listing. (sheet 7 of 62)

```
      HEAD(1) =
     *' CCCCCC    DDDDDDD    FFFFFFFFF  TTTTTTTTT  BBBBBBBB   LL        '
      HEAD(2) =
     *'CC    CC   DD    DD   FFFFFFFFF  TTTTTTTTT  BB     BB  LL        '
      HEAD(3) =
     *'CC         DD     DD  FF            TT      BB    BB   LL        '
      HEAD(4) =
     *'CC         DD     DD  FFFFFF        TT      BBBBBBBB   LL        '
      HEAD(5) =
     *'CC         DD     DD  FF            TT      BB    BB   LL        '
      HEAD(6) =
     *'CC    CC   DD    DD   FF            TT      BB     BB  LLLLLLLLL'
      HEAD(7) =
     *' CCCCCC    DDDDDDD    FF            TT      BBBBBBBB   LLLLLLLLL'
C
      WRITE(IRPT,7000)
 7000 FORMAT(1H1)
C
      WRITE(IRPT,7010)
 7010 FORMAT(1X,79(1H*))
C
      WRITE(IRPT,7050)
C
      WRITE(IRPT,7015) HEAD(1)
      WRITE(IRPT,7015) HEAD(2)
      WRITE(IRPT,7015) HEAD(3)
      WRITE(IRPT,7015) HEAD(4)
      WRITE(IRPT,7015) HEAD(5)
      WRITE(IRPT,7015) HEAD(6)
      WRITE(IRPT,7015) HEAD(7)
 7015 FORMAT(9X,A64)
C
      WRITE(IRPT,7020) PRGNAM, PRGVER, PRGDAT
 7020 FORMAT(//32X,A8,1X,A8/26X,'Last Modified on ',A12)
C
      WRITE(IRPT,7025)
 7025 FORMAT(///
     * 14X,'     Cumulative Distribution Function Estimation    '/
     * 14X,'            from a User Supplied Data Set           '//
     * 14X,'  Statistical Program Developed by Paul W. Eslinger '/
     * 14X,'Keyword Input Routines Developed by Dave W. Langford')
C
      WRITE(IRPT,7030) CRUNID, USRNAM
 7030 FORMAT(///12X,'Current Run ID = ',A12,'   User Name = ',A16)
C
      WRITE(IRPT,7040) SYSDAT, SYSTIM
 7040 FORMAT(/17X,'System Date = ',A8,'   System Time = ',A8)
C
      WRITE(IRPT,7050)
 7050 FORMAT(///)
C
```

Table A-1. Source Code Listing. (sheet 8 of 62)

```
      WRITE(IRPT,7010)
C
      RETURN
      END
*DECK CEXIST
      LOGICAL FUNCTION CEXIST( KEYWRD )
C*....LOGICAL FUNCTION CEXIST( KEYWRD )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This is a logical function to determine if a eight letter keyword
C*    is one of the secondary keywords (modifiers) of the current data
C*    block.
C*
C*    This routine is a part of the set of routines called by RDBLK.
C*
C*
C*  History:
C*
C*    David W. Langford :  29 Dec 1989 :  Version 1.0
C*
C***********************************************************************
C
*CALL CDATA
C
      CHARACTER*8 KEYWRD
C
      CEXIST=.FALSE.
      DO 100 N = 1, NCVALU
         IF( CVALUE(N) .EQ. KEYWRD ) THEN
            CEXIST = .TRUE.
            RETURN
         ENDIF
  100 CONTINUE
C
      RETURN
      END
*DECK COMCHR
      LOGICAL FUNCTION COMCHR( IPLACE, CH )
C*....LOGICAL FUNCTION COMCHR( IPLACE, CH )
C***********************************************************************
C*
C*  Purpose:
C*
C*
C*    This subroutine determines if the character CH is a comment
C*    identifying character.  Comment lines are ignored, as are
C*    characters after a comment identifier on one line.
C*
C*    The call argument IPLACE indicates the position of character CH
```

Table A-1. Source Code Listing. (sheet 9 of 62)

```
C*    in the input line.  Note that the '/' and '*' characters indicate
C*    comments only if they are the first characters in a line.
C*
C*    This routine is a part of the set of routines called by RDBLK.
C*
C*
C*  History:
C*
C*    David W. Langford :  29 Dec 1989 :  Version 1.0
C*
C***********************************************************************
C
      CHARACTER*1 CH
C
      IF ((CH.EQ.'!').OR.(CH.EQ.'$')) THEN
         COMCHR=.TRUE.
      ELSE
         IF (((CH.EQ.'/').OR.(CH.EQ.'*')).AND.(IPLACE.EQ.1)) THEN
            COMCHR=.TRUE.
         ELSE
            COMCHR=.FALSE.
         ENDIF
      ENDIF
C
      RETURN
      END
*DECK CONCAT
      SUBROUTINE CONCAT( SOURCE, EXT, TARGET )
C*....SUBROUTINE CONCAT( SOURCE, EXT, TARGET )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine is used to concatenate an extension on the end
C*    of a file name in the variable SOURCE.  The new file name is
C*    output in the variable TARGET.  The extension is input in the
C*    variable EXT which is assumed to have length 3.  The extension
C*    is placed in TARGET starting at the first blank in SOURCE.  Both
C*    SOURCE and TARGET have length 64.
C*
C*    The assumptions of this routine are based on generating file
C*    names which can be used on an IBM personal computer.  I.e.,
C*    pathnames are allowed, but extensions have a maximum length of
C*    three and are placed after a period.  I.e., the output file
C*    name looks like "NAME.EXT".
C*
C*
C*  History:
C*
C*    Paul W. Eslinger :  15 Jan 1990 :  Original source
C*
```

Table A-1. Source Code Listing. (sheet 10 of 62)

```
C***********************************************************************
C
      CHARACTER BLANK*1, EXT*3, SOURCE*64, TARGET*64
C
      DATA BLANK / ' '/
C
C *** Find first blank character in the source name
C *** Only 60 characters are allowed before the period preceding
C     the extension
C
      DO 10 I = 1, 61
        IF( SOURCE(I:I) .EQ. BLANK ) THEN
          NUM = I
          GO TO 20
        ENDIF
   10 CONTINUE
C
C *** Output error message and terminate -- illegal file name
C
      WRITE(*,7000) SOURCE
 7000 FORMAT(///
     * ' *** Terminal error in subroutine CONCAT ***'/
     * '      Invalid source file name prefix.'/
     * '      SOURCE = ',A64)
      STOP
C
   20 CONTINUE
C
C *** Transfer the file name
C
      NEND = NUM - 1
      TARGET(1:NEND) = SOURCE(1:NEND)
C
C *** Now put the extension on
C
      TARGET(NUM:NUM) = '.'
C
      NUM = NUM + 1
      NEND = NUM + 2
      TARGET(NUM:NEND) = EXT(1:3)
C
C *** Fill the rest of the target name with blanks
C
      NUM = NUM + 3
      DO 30 I = NUM, 64
        TARGET(I:I) = BLANK
   30 CONTINUE
C
      RETURN
      END
*DECK DCDSEP
```

Table A-1. Source Code Listing. (sheet 11 of 62)

```
      SUBROUTINE DCDSEP( NUM, LOC )
C*....SUBROUTINE DCDSEP( NUM, LOC )
C***********************************************************************
C*
C*  Purpose:
C*
C*   This subroutine examines data in the input line (INFO) and
C*   transfers character data into the array CVALUE and numeric data
C*   into the array VALUE.  The vector LOC contains the field
C*   separators in the input line.  NUM indicates the number of
C*   separators present.
C*
C*    This routine is a part of the set of routines called by RDBLK.
C*
C*
C*  Notes:
C*
C*     1. Real numbers may be written in exponential format, but only
C*        an 'E' or an 'e' are recognized exponent indicators.  The
C*        double precision 'D' exponent indicator is not valid.
C*
C*     2. A '*' character interior to a line is taken to indicate a
C*        repetition of the following value or character field.  It
C*        must be preceded by an unsigned integer value.
C*
C*
C*  History:
C*
C*    David W. Langford :  29 Dec 1989 :  Version 1.0
C*
C***********************************************************************
C
*CALL CDATA
*CALL FILES
C
      CHARACTER*1 CH, DUMY(80)
      DIMENSION LOC(80)
      LOGICAL DECSET, EXPSET, MINUS
C
C             LOOP ON THE NUMBER OF SEPARATORS, MINUS ONE.
C             INITIALIZE CHARACTER COUNT BETWEEN ADJACENT SEPARATORS.
C
      DO 210 I=1,(NUM-1)
         J = I+1
         ISTART = LOC(I)+1
         NCHARS = LOC(J)-ISTART
         IF (NCHARS.LT.1) GO TO 210
C
C             INITIALIZATIONS FOR DECODING LOOP
C
         DECSET = .FALSE.
```

Table A-1. Source Code Listing. (sheet 12 of 62)

```
      EXPSET = .FALSE.
      IEND   = LOC(J)-1
      ISETNO = 0
      MULT   =-1
      NDIGIT = 0
      RDATA  = 0.0
C
      IS   = LOC(I)
      IEND = LOC(J)-1
C
C           CHECK FOR SIGN OF NUMERIC RESULT
C
  100 CONTINUE
      MINUS = .FALSE.
      IF (INFO(IS+1).EQ.'+') THEN
         IS = IS+1
         IF (IS.GE.IEND) GO TO 180
      ENDIF
      IF (INFO(IS+1).EQ.'-') THEN
         MINUS = .TRUE.
         IS    = IS+1
         IF (IS.GE.IEND) GO TO 180
      ENDIF
C
C           PRIMARY LOOP DECODES POSITIVE INTEGER VALUES
C
  110 CONTINUE
      ISETNO = ISETNO+1
      NUMB   = 0
      NDIGIT = 0
  120 CONTINUE
      IS = IS+1
      CH = INFO(IS)
      IF ((CH.GE.'0').AND.(CH.LE.'9')) THEN
         READ (CH,500) J
         NUMB   = NUMB*10+J
         NDIGIT = NDIGIT+1
         IF (IS.GE.IEND) THEN
            IF (MINUS) NUMB = -NUMB
            GO TO 140
         ENDIF
         GO TO 120
      ENDIF
      IF (MINUS) NUMB = -NUMB
C
C           CHECK FOR REPETITION FACTOR
C
      IF (CH.EQ.'*') THEN
         IF ((MULT.GT.0) .OR.(NDIGIT.LT.1)) GO TO 180
         IF ((IS.GE.IEND).OR.(ISETNO.GT.1)) GO TO 180
         IF (MINUS) GO TO 180
```

Table A-1. Source Code Listing. (sheet 13 of 62)

```
            MULT   = NUMB
            ISTART = IS+1
            NCHARS = LOC(I+1)-ISTART
            GO TO 100
         ENDIF
C
C              CHECK FOR DECIMAL POINT.  NOTE THAT 'MINUS' IS SAVED.
C
         IF (CH.EQ.'.') THEN
            IF (DECSET.OR.EXPSET) GO TO 180
            DECSET = .TRUE.
            IF (NDIGIT.GT.0) RDATA = FLOAT(NUMB)
            IF (IS.GE.IEND) GO TO 160
            GO TO 110
         ENDIF
C
C             CHECK FOR EXPONENT MARKER
C
         IF ((CH.EQ.'E').OR.(CH.EQ.'e')) THEN
            IF (EXPSET) GO TO 180
            IF ((ISETNO.LT.2).AND.(NDIGIT.LT.1)) GO TO 180
            EXPSET = .TRUE.
            IF (NDIGIT.GT.0) THEN
               D = FLOAT(NUMB)
               IF (DECSET) THEN
                  DO 130 J=1,NDIGIT
                     D = 0.1*D
  130             CONTINUE
               ENDIF
               RDATA = RDATA+D
            ENDIF
            IF (IS.GE.IEND) GO TO 160
            GO TO 100
         ENDIF
C
C             CANNOT DECIPHER STRING AS A NUMBER
C
         GO TO 180
C
C             FINISHED DECODING NUMERIC VALUE
C
  140    CONTINUE
         IF (EXPSET) THEN
            IF (NUMB.NE.0) RDATA = RDATA*(10.0**NUMB)
         ELSEIF (DECSET) THEN
            D = FLOAT(NUMB)
            DO 150 J=1,NDIGIT
               D = 0.1*D
  150       CONTINUE
            RDATA = RDATA+D
         ELSE
```

Table A-1. Source Code Listing. (sheet 14 of 62)

```
            RDATA = FLOAT(NUMB)
         ENDIF
C
C             FILL IN NUMERIC DATA.  ADD MULTIPLE VALUES, IF REQUESTED.
C
  160    CONTINUE
         NVALUE = NVALUE+1
         IF (NVALUE.GT.NVALS) GO TO 470
         VALUE(NVALUE) = RDATA
C
         IF (MULT.GT.1) THEN
            IF ((NVALUE+MULT-1).GT.NVALS) GO TO 470
            DO 170 J=2,MULT
               NVALUE = NVALUE+1
               VALUE(NVALUE) = RDATA
  170       CONTINUE
         ENDIF
         GO TO 210
C
C ---------------------------------------------------------------------
C
C             TRANSFER AS A CHARACTER VARIABLE.
C
  180    CONTINUE
         NCVALU = NCVALU+1
         IF (NCVALU.GT.NKEYS) GO TO 480
         K=0
         DO 190 J=ISTART,IEND
            K=K+1
            DUMY(K) = INFO(J)
  190    CONTINUE
         DUMY(K+1) = ' '
         CALL XFKEY (DUMY,CVALUE(NCVALU))
C
C             FILL IN MULTIPLE CHARACTER VALUES, IF REQUESTED.
C
         IF (MULT.GT.1) THEN
            IF ((NCVALU+MULT-1).GT.NKEYS) GO TO 480
            K = NCVALU
            DO 200 J=2,MULT
               NCVALU = NCVALU+1
               CVALUE(NCVALU) = CVALUE(K)
  200       CONTINUE
         ENDIF

  210 CONTINUE
C
      RETURN
C
C             ERRORS
C
```

Table A-1. Source Code Listing. (sheet 15 of 62)

```
  470 CONTINUE
      WRITE(IRPT,970) NVALS
      GO TO 490
  480 CONTINUE
      WRITE(IRPT,980) NKEYS
  490 CONTINUE
      WRITE(IRPT,990) ILINE
      STOP
C
  500 FORMAT(I1)
  510 FORMAT(80A1)
  970 FORMAT(/
     * ' *** Terminal error in subroutine DCDSEP ***'/
     * '     Error in the input data deck.  Too many numeric values'/
     * '     were found within a single data block.  Limit is ',I5,'.')
C
  980 FORMAT(/
     * ' *** Terminal error in subroutine DCDSEP ***'/
     * '     Error in the input data deck.  Too many character values'/
     * '     were found within a single data block.  Limit is ',I5,'.')
C
  990 FORMAT(/
     * ' *** Terminal error in subroutine DCDSEP ***'/
     * '     Error at line ',I5,' of the input data file.'/
     * '     Error stop.')
      END
*DECK EMPIR
      SUBROUTINE EMPIR
C*....SUBROUTINE EMPIR
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine computes the cumulative distribution function for
C*    the data in SVAL at the probability levels contained in ALPHA.
C*    The method is to interpolate between the points from the usual
C*    empirical distribution function.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger :  15 Jan 1990 :  Original source
C*
C***********************************************************************
C
*CALL SDATA
C
C *** Special actions for alpha levels of 0.0 and 1.0
C
      XPLOT(1) = SVAL(1)
      XPLOT(NALPHA) = SVAL(NSIZE)
```

Table A-1. Source Code Listing. (sheet 16 of 62)

```
C
C *** Loop over the ALPHA levels and compute the CDF by interpolation
C     between the sorted sample data values
C     All alpha levels smaller than 1/NSIZE are mapped to the first
C     data value.
C
      FN = NSIZE
      DO 30 I = 2, NALPHA-1
        IF( ALPHA(I) .LE. (1.0/FN) ) THEN
          XPLOT(I) = SVAL(1)
          GO TO 30
        ENDIF
        DO 20 J = 1, NSIZE-1
          FJ  =  J   /FN
          FJP = (J+1)/FN
          IF( ALPHA(I).GE.FJ .AND. ALPHA(I).LT.FJP ) THEN
            XPLOT(I) = FN*(ALPHA(I)-FJ) * (SVAL(J+1)-SVAL(J))
     *                 + SVAL(J)
            GO TO 30
          ENDIF
   20   CONTINUE
        XPLOT(I) = SVAL(NSIZE)
   30 CONTINUE
C
      RETURN
      END
*DECK EPAN
      SUBROUTINE EPAN
C*....SUBROUTINE EPAN
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine computes the cumulative distribution function
C*    for the kernel density estimate of the data in SVAL.  The table
C*    of outputs has probability values contained in the vector ALPHA
C*    and X locations in the vector XPLOT.
C*
C*    The method employed is to use trapezoidal integration for a
C*    large number of points to compute the cumulative distribution
C*    function (FINT).  Interpolation is then done on this set of
C*    probabilities to find the X values associated with the requested
C*    probabilities contained in the vector ALPHA.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger :  15 Jan 1990 :  Original source
C*
C***********************************************************************
C
```

Table A-1. Source Code Listing. (sheet 17 of 62)

```
      DOUBLE PRECISION SUM
C
*CALL FILES
*CALL SDATA
C
C---- Initialize -------------------------------------------------------
C
C *** Scale estimate required for the kernel density
C      => See subroutine INPUTS and/or subroutine SCALE
C
      SN = ISCA
      IF( SN .LE. 0.0 ) THEN
        WRITE(IRPT,1000) SN
 1000   FORMAT(/' *** Terminal Error in Subroutine EPAN ***'/
     *          '     Scale estimate for density estimation'/
     *          '     is not positive.  SN = ',1P,E12.4)
        STOP
      ENDIF
C
C *** Integration criteria - assume sorted values in SVAL
C ***   XMIN :  Minimum value required for integration
C ***   XMAX :  Maximum value required for integration
C
      XMIN = SVAL(1)     - CNE * SN
      XMAX = SVAL(NSIZE) + CNE * SN
C
C *** Set up integration step size
C
      XDEL = ( XMAX - XMIN ) / FLOAT( NSTP-1 )
C
C---- Compute the cumulative distribution function ---------------------
C
C *** GHAT :  The kernel density
C     Also, integrate the kernel density and place in FINT
C
      X = XMIN
      DO 10 I = 1, NSTP
C        GHAT(I) = GKERN(X)
        GHAT(I) = GALIAS( X )
        XINT(I) = X
        IF( I .EQ. 1 ) THEN
          SUM = 0.0
          FINT(I) = 0.0
        ELSE
          SUM = SUM + XDEL*(GHAT(I)+GHAT(I-1))/2.0
          FINT(I) = SUM
        ENDIF
        X = X + XDEL
   10 CONTINUE
C
C
```

Table A-1. Source Code Listing. (sheet 18 of 62)

```
C---- Interpolate to find the X values associated with ALPHA -----------
C
C *** Special actions for alpha levels of 0.0 and 1.0
C     The effective range of the data has been modified if density
C     aliasing has been done.
C
      XPLOT(1)      = XMIN
      XPLOT(NALPHA) = XMAX
C
      IF( IALIAS.EQ.1 .OR. IALIAS.EQ.3 ) THEN
        IF( ALEFT .GT. XMIN ) XPLOT(1) = ALEFT
      ENDIF
C
      IF( IALIAS.EQ.2 .OR. IALIAS.EQ.3 ) THEN
        IF( ARIGHT .LT. XMAX) XPLOT(NALPHA) = ARIGHT
      ENDIF
C
C *** Loop over the ALPHA levels and compute the CDF's
C
      DO 30 I = 2, NALPHA-1
        DO 20 J = 1, NSTP-1
          IF( FINT(J).LE.ALPHA(I) .AND. FINT(J+1).GT.ALPHA(I) ) THEN
            X = (XINT(J+1)-XINT(J)) *
     *          (ALPHA(I)-FINT(J))/(FINT(J+1)-FINT(J))
     *          + XINT(J)
            XPLOT(I) = X
          ENDIF
   20   CONTINUE
   30 CONTINUE
C
      RETURN
      END
*DECK FINSEP
      SUBROUTINE FINSEP( NUM, LOC )
C*....SUBROUTINE FINSEP( NUM, LOC )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine locates all the separators in the current input
C*    line, INFO.  The number of separators is placed in NUM, and the
C*    location of each is placed in array LOC.  A dummy separator is
C*    placed at position 81, if the last character in array INFO if a
C*    relevant character (not a comment or separator).
C*
C*    This routine is a part of the set of routines called by RDBLK.
C*
C*
C*  History:
C*
C*    David W. Langford :  29 Dec 1989 :  Version 1.0
```

Table A-1. Source Code Listing. (sheet 19 of 62)

```
C*
C***********************************************************************
C
*CALL CDATA
C
      DIMENSION LOC(80)
      EXTERNAL COMCHR, SEPCHR
      LOGICAL  COMCHR, SEPCHR
C
      NUM=0
      LENGTH=0
C
      DO 100 I=1,80
         IF (INFO(I).NE.' ') LENGTH=I
  100 CONTINUE
      IF (LENGTH.LT.80) LENGTH=LENGTH+1
      IF (LENGTH.EQ.0) RETURN
C
C             LOOP ON THE CHARACTERS OF THE INPUT LINE.  CHECK FIRST
C             FOR A COMMENT CHARACTER, THEN FOR A SEPARATOR CHARACTER.
C
      DO 110 I=1,LENGTH
         IF (COMCHR(I,INFO(I))) THEN
            IF ((I.GT.1).AND.(NUM.GT.0)) THEN
               IF (LOC(NUM).LT.(I-1)) THEN
                  NUM=NUM+1
                  LOC(NUM)=I
               ENDIF
            ENDIF
            RETURN
         ENDIF
         IF (SEPCHR(INFO(I))) THEN
            NUM=NUM+1
            LOC(NUM)=I
         ENDIF
  110 CONTINUE
C
      IF ((LENGTH.EQ.80).AND.(.NOT.SEPCHR(INFO(80)))) THEN
         NUM=NUM+1
         LOC(NUM)=81
      ENDIF
      RETURN
      END
*DECK FNAME
      SUBROUTINE FNAME( CARD, LENC, FILE, LENF, IERR )
C*....SUBROUTINE FNAME( CARD, LENC, FILE, LENF, IERR )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine extracts a file name from the information
```

Table A-1. Source Code Listing. (sheet 20 of 62)

```
C*    contained in the character vector CARD.  The file name must
C*    be enclosed in double apostrophes and cannot contain more than
C*    LENF characters.  This subroutine only reads CARD until two
C*    apostrophes are found.  Any characters after the second
C*    apostrophe are ignored.
C*
C*
C*  Call List Variables:
C*
C*    CARD :  Input (character) vector from which to extract the file
C*            name.  CARD is considered as a CHARACTER*1 vector of
C*            length LENC.
C*    LENC :  Input (integer) giving the maximum number of characters
C*            in CARD.
C*    FILE :  Output (character) array containing the file name which
C*            was extracted from CARD.  FILE is considered as a
C*            CHARACTER*1 vector of length LENF.  FILE is blank filled
C*            before any file name extraction is performed.
C*    LENF :  Input (integer) giving the maximum number of characters
C*            in the file name FILE.
C*    IERR :  Output (integer) error flag.
C*             0 = Normal termination
C*             1 = There were not two apostrophes in CARD.  A blank
C*                 file name will be returned.
C*             2 = Two adjacent apostrophes were located.  A blank
C*                 file name will be returned.
C*             3 = The file name would contain more than LENF
C*                 characters.  A blank file name is returned.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger :  29 Dec 1989 :  Version 1.0
C*
C***********************************************************************
C
      INTEGER LENC, LENF, IERR
      CHARACTER*1 CARD(LENC), FILE(LENF), APOS, BLANK
C
      DATA APOS /'"'/, BLANK /' '/
C
C *** Set the error flag and blank fill the file name
C
      IERR = 0
      DO 10 I = 1, LENF
        FILE(I) = BLANK
   10 CONTINUE
C
C *** Locate the apostrophes bracketing the file name
C
      I1 = 0
```

Table A-1. Source Code Listing. (sheet 21 of 62)

```
      I2 = 0
      DO 20 I = 1, LENC
        IF( CARD(I) .EQ. APOS ) THEN
          IF( I1 .EQ. 0 ) THEN
            I1 = I
          ELSE
            I2 = I
            GO TO 30
          ENDIF
        ENDIF
   20 CONTINUE
C
C *** Error checking
C
   30 CONTINUE
C
      IF( I1.EQ.0 .OR. I2.EQ.0 ) THEN
        IERR = 1
        RETURN
      ENDIF
C
      LENN = I2 - I1 - 1
      IF( LENN .LE. 0 ) THEN
        IERR = 2
        RETURN
      ENDIF
C
      IF( LENN .GT. LENF ) THEN
        IERR = 3
        RETURN
      ENDIF
C
C *** Transfer the file name
C
      DO 40 I = 1, LENN
        FILE(I) = CARD(I1+I)
   40 CONTINUE
C
      RETURN
      END
*DECK GALIAS
      REAL FUNCTION GALIAS( X )
C*....REAL FUNCTION GALIAS( X )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This function will schedule evaluation of the Epanechnikov kernel
C*    density at the location X using the data in SVAL.  A range
C*    aliasing option may be exercised.
C*
```

Table A-1. Source Code Listing. (sheet 22 of 62)

```
C*
C*  Inputs:
C*
C*    Call List...
C*    X      :  Value at which to evaluate the density
C*
C*    Common Blocks...
C*    IALIAS :  Flag controlling aliasing.
C*               0 = No aliasing
C*               1 = Left boundary aliasing only
C*               2 = Right boundary aliasing only
C*               3 = Both boundaries are aliased
C*    ALEFT  :  Location of left aliasing boundary
C*    ARIGHT :  Location of right aliasing boundary
C*    SVAL   :  Sorted (negative to positive) data set
C*    NSIZE  :  Number of points in the data set
C*    XMIN   :  Minimum location where the density is nonzero.
C*    XMAX   :  Maximum location where the density is nonzero.
C*
C*
C*  Output:
C*
C*    GALIAS :  Epanechnikov kernel (nonparametric) density estimate
C*              at the point X
C*
C*
C*  Note:
C*
C*    This implementation requires a sorted data set.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger :  12 Jun 1990 :  Original version
C*
C***********************************************************************
C
*CALL SDATA
C
C *** No aliasing desired
C
      IF( IALIAS .EQ. 0 ) THEN
        GALIAS = GKERN( X )
        RETURN
      ENDIF
C
      GALIAS = 0.0
C
C *** Left boundary (only) aliasing desired
C
      IF( IALIAS .EQ. 1 ) THEN
```

Table A-1. Source Code Listing. (sheet 23 of 62)

```
          IF( X .LE. ALEFT ) RETURN
          GALIAS = GKERN( X )
          DL     = X - ALEFT
          XALIAS = ALEFT - DL
          GALIAS = GALIAS + GKERN( XALIAS )
          RETURN
        ENDIF
C
C *** Right boundary (only) aliasing desired
C
        IF( IALIAS .EQ. 2 ) THEN
          IF( X .GE. ARIGHT ) RETURN
          GALIAS = GKERN( X )
          DR     = ARIGHT - X
          XALIAS = ARIGHT + DR
          GALIAS = GALIAS + GKERN( XALIAS )
          RETURN
        ENDIF
C
C *** Both boundary aliasing desired
C
        IF( IALIAS .EQ. 3 ) THEN
          IF( X.LE.ALEFT .OR. X.GE.ARIGHT ) RETURN
          GALIAS = GKERN( X )
          DL = X - ALEFT
          DR = ARIGHT - X
C
C ***     Left side aliasing
C
          IDX = 1
          XALIAS = ALEFT - DL
   10     CONTINUE
          IF( XALIAS .LE. XMIN ) GO TO 20
          GALIAS = GALIAS + GKERN( XALIAS )
          IDX = IDX + 1
          IF( MOD(IDX,2) .EQ. 0 ) THEN
            XALIAS = XALIAS - 2.0*DR
          ELSE
            XALIAS = XALIAS - 2.0*DL
          ENDIF
          GO TO 10
C
C ***     Right side aliasing
C
   20     CONTINUE
C
          IDX = 1
          XALIAS = ARIGHT + DR
   30     CONTINUE
          IF( XALIAS .GE. XMAX ) GO TO 40
          GALIAS = GALIAS + GKERN( XALIAS )
```

Table A-1. Source Code Listing. (sheet 24 of 62)

```
        IDX = IDX + 1
        IF( MOD(IDX,2) .EQ. 0 ) THEN
          XALIAS = XALIAS + 2.0*DL
        ELSE
          XALIAS = XALIAS + 2.0*DR
        ENDIF
        GO TO 30
C
   40   CONTINUE
        RETURN
      ENDIF
C
      STOP 'GALIAS-Error'
      END
*DECK GKERN
      REAL FUNCTION GKERN( X )
C*....REAL FUNCTION GKERN( X )
C*********************************************************************
C*
C*  Purpose:
C*
C*    This function will evaluate the Epanechnikov kernel density
C*    at the location X for the data in SVAL.
C*
C*
C*  Inputs:
C*
C*    SVAL  :  Sorted (negative to positive) data set
C*    X     :  Value at which to evaluate the density
C*    NSIZE :  Number of points in the data set
C*    CNE   :  Bandwidth parameter for Epanechnikov's kernel
C*    SN    :  Robust standard deviation estimate for this data set
C*
C*
C*  Output:
C*
C*    GKERN :  Epanechnikov kernel (nonparametric) density estimate
C*             at the point X
C*
C*
C*  Note:
C*
C*    This implementation requires a sorted data set.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 15 Jan 1990 : Original version
C*
C*********************************************************************
C
```

Table A-1. Source Code Listing. (sheet 25 of 62)

```
*CALL SDATA
C
      DOUBLE PRECISION G, W
C
      G  = 0.0
      HN = CNE * SN
C
C-----------------------------------------------------------------------
C  Decide which sample points need to be in the sum for the point X
C-----------------------------------------------------------------------
C
      ISTART = 1
      IEND   = NSIZE
C
C *** Lower and upper limits to the region of influence of X
C
      XLOW  = X - HN
      XHIGH = X + HN
C
C *** Check for X in the range of the data
C
      GKERN = 0.0
      IF( XHIGH .LE. SVAL(1) ) RETURN
      IF( XLOW .GE. SVAL(NSIZE) ) RETURN
C
C *** Find the first index where SVAL(INDEX) .GE. X
C
      I1 = 1
      I2 = NSIZE
      IF( XLOW .LE. SVAL(1) ) THEN
        ISTART = I1
        GO TO 20
      ENDIF
C
   10 CONTINUE
        K = ( I2 - I1 ) / 2
        IF( K .EQ. 0 ) THEN
          ISTART = I2
          GO TO 20
        ENDIF
        IF( SVAL(I1+K) .GT. XLOW ) THEN
          I2 = I2 - K
        ELSE
          I1 = I1 + K
        ENDIF
        GO TO 10
C
   20 CONTINUE
C
C *** Find the last index where SVAL(INDEX) .LE. X
C
```

Table A-1. Source Code Listing. (sheet 26 of 62)

```
      I2 = NSIZE
      IF( XHIGH .GE. SVAL(NSIZE) ) THEN
        IEND = I2
        GO TO 40
      ENDIF
C
   30 CONTINUE
        K = ( I2 - I1 ) / 2
        IF( K .EQ. 0 ) THEN
          IEND = I1
          GO TO 40
        ENDIF
        IF( SVAL(I1+K) .GT. XHIGH ) THEN
          I2 = I2 - K
        ELSE
          I1 = I1 + K
        ENDIF
        GO TO 30
C
   40 CONTINUE
C
C-----------------------------------------------------------------------
C           Do the sum for the kernel density estimate
C-----------------------------------------------------------------------
C
      DO 50 I = ISTART, IEND
        W = ( (X-SVAL(I))/HN ) ** 2
        G = G + 1.0 - W
   50 CONTINUE
C
C *** Normalize so the density integrates to unity
C
      GKERN = 0.75 * G / ( HN * FLOAT(NSIZE) )
C
      RETURN
      END
*DECK IDENC
      SUBROUTINE IDENC
C*....SUBROUTINE IDENC
C***********************************************************************
C*
C*  Purpose:
C*
C*    This routine generates identification variables for this run
C*    of the code.  Variables are:
C*
C*      1. A unique run identifier number, utilizing the date
C*         and time which this routine was called
C*
C*      2. Hard coded - Program name and version number
C*
```

Table A-1. Source Code Listing. (sheet 27 of 62)

```
C*      3. Hard coded - User name
C*
C*      4. Hard coded - Date that program was last modified
C*
C*      5. System date and time
C*
C*
C*  Operating system:
C*
C*    This subroutine is designed for the IBM PC using the Lahey
C*    FORTRAN 77 compiler.
C*
C*
C*  Variable Definitions:
C*
C*    CRUNID = CHARACTER*12 output run identifier, as follows, with
C*             digit 1 being the leftmost digit.
C*      Digits  1- 2 = Year
C*      Digits  3- 4 = Month
C*      Digits  5- 6 = Day (1 to 31)
C*      Digits  7- 8 = Hour (0 to 23)
C*      Digits  9-10 = Minute (0 to 59)
C*      Digits 11-12 = Second (0 to 59)
C*
C*    PRGNAM = CHARACTER*8 name of the program
C*    PRGVER = CHARACTER*8 version number of the program
C*    PRGDAT = CHARACTER*12 date stamp for program completion
C*    SYSDAT = CHARACTER*8 System date in the format mm-dd-yy
C*    SYSTIM = CHARACTER*8 System time in the format hh:mm:ss
C*    USRNAM = CHARACTER*16 name of the current user.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 18 Jan 1990 : Version 1.0
C*
C***********************************************************************
C
*CALL IDENC
C
C *** User name
C
      USRNAM = 'XXXXXX X. XXXXXX'
C
C *** Program name and version number
C
      PRGNAM = 'CDFTBL'
      PRGVER = '1.000'
C
C *** Program date
C
```

Table A-1. Source Code Listing. (sheet 28 of 62)

```
      PRGDAT = '13 Jun 1990'
C
C *** System time in the (CHARACTER) form hh:mm:ss
C
      CALL TIME( SYSTIM )
C
C *** System date in the (CHARACTER) form mm-dd-yy
C
      CALL DATE( SYSDAT )
C
C *** Run identification number
C
      CRUNID( 1: 2) = SYSDAT(7:8)
      CRUNID( 3: 4) = SYSDAT(1:2)
      CRUNID( 5: 6) = SYSDAT(4:5)
      CRUNID( 7: 8) = SYSTIM(1:2)
      CRUNID( 9:10) = SYSTIM(4:5)
      CRUNID(11:12) = SYSTIM(7:8)
C
      RETURN
      END
*DECK INPUTS
      SUBROUTINE INPUTS
C*....SUBROUTINE INPUTS
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine reads the control variables and data values from
C*    the input file.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 15 Jan 1990 : Original source
C*    Paul W. Eslinger : 15 Jun 1990 : Add ALIAS to the DISTRIBU card
C*
C*
C*  Variable definitions:
C*
C*    All the inputs come from the data file.  The first lines
C*    of the data file contain control information given by keywords,
C*    followed by the sample data.  A maximum of 8 letters are used to
C*    define individual keywords.  If the default value is desired,
C*    the keyword definition does not need to be placed in the input
C*    file.  Separators between numeric values, keywords and modifiers
C*    can be a space or the characters ",", "=", ":", ";", "'", "(",
C*    or ")".
C*
C*    Keyword   : Description
C*    ------------------------------------------------------------
```

Table A-1. Source Code Listing. (sheet 29 of 62)

```
C*
C*    $          : Comment designator.  Any text or numeric values
C*                 following the $ are ignored.
C*
C*    !          : Comment designator.  Any text or numeric values
C*                 following the ! are ignored.
C*
C*    /          : Comment designator if used in column 1 of a input
C*                 data line.  It is ignored if embedded within a line.
C*
C*    *          : Comment designator if used in column 1 of a input
C*                 data line.  It is treated as a repeat indicator if
C*                 it is embedded within a line.  I.e., 6*50. means
C*                 the same as entering the number 50.0 six times.
C*
C*    DISTRIBU   : Chose the type of distribution to use.  Two options
C*                 are available: The kernel density estimate and the
C*                 empirical distribution function estimate.  Modifiers
C*                 are EMPIRICA and KERNEL.  Only one of the two
C*                 modifiers "KERNEL" and "EMPIRICA" can be used.
C*                   EMPIRICA : Compute the CDF table by using the
C*                              empirical CDF.
C*                   KERNEL   : Compute the CDF table by using the
C*                              kernel density estimator.  Four
C*                              modifiers may be used.  If used,
C*                              the numerical entries are assumed
C*                              in the same order as the following
C*                              description.
C*                   ALIAS    : If ALIAS is entered, it is followed
C*                              by up to three numerical values.
C*                              The first is the alias flag.  0=no
C*                              aliasing, and is a single entry. 1=
C*                              alias the left boundary, and must be
C*                              followed by the boundary value. 2=
C*                              alias the right boundary, and must be
C*                              followed by the boundary value. 3=
C*                              alias both boundaries, and must be
C*                              followed by the two boundary values.
C*                   BANDWIDT : If BANDWIDT is entered it is followed
C*                              by a CN value for Epanechnikov's
C*                              kernel.  If BANDWIDT is not entered,
C*                              then CN is set to 2.283*NSIZE**(-.287)
C*                   SCALE    : If SCALE is entered it is followed by
C*                              a standard deviation estimate for the
C*                              data set.  If SCALE is not entered, a
C*                              value is generated internally using
C*                              a scaled median absolute deviation.
C*                   STEPS    : Number of integration steps to use in
C*                              the trapezoidal integration.  A value
C*                              of 100 is the default.
C*
```

Table A-1. Source Code Listing. (sheet 30 of 62)

```
C*     END        : Signals the end of the keyword definitions.  Data
C*                 entries start on the next line of the input file.
C*
C*     PLOT       : Write an optional plot file.  The file name is
C*                 defined in subroutine OPENER.  The following
C*                 modifiers can be entered in any combination:
C*                   CUMULATI : Write the cumulative distribution
C*                              function to the plot file.
C*                   DENSITY  : Write the kernel density to the plot
C*                              file if it was used.
C*                   TABLE    : Write the table of probabilities to
C*                              the plot file.
C*
C*     PRINT      : Echo the data to the report file.  Presence of the
C*                 keyword invokes the print.  This is especially
C*                 useful if a data transformation has been done.
C*                 Modifiers are:
C*                   BEFORE : Print the data before transformations
C*                   AFTER  : Print the data after transformations
C*                   RANGE  : N1 and N2 give the range of indices of
C*                            the (sorted) data to be printed.  The
C*                            default range is the entire data set.
C*
C*     PROBABIL   : Keyword to define the probability levels which will
C*                 written to the CDF table.  A default table uses
C*                 35 probability levels.  The numeric entries are the
C*                 number (NALPHA) of probability values followed by
C*                 NALPHA probabilities in strictly increasing order.
C*                 The first probability must be 0.0 and the last
C*                 probability must be 1.0.
C*
C*     SAMPLE     : Keyword defining the number of values in the sample
C*                 data set.  The first numerical entry is the number
C*                 of values in the data set.  There is no default
C*                 value for the sample size.  A single modifier can
C*                 defined.
C*                   LINE : Number of values on each line of the
C*                          input file.  The default value is 1.
C*
C*     TRANSFOR   : Variable transformation option.  The value of the
C*                 first numeric entry (N1) is a transformation index.
C*                   0 = No transformations
C*                   1 = Log base 10 transformation
C*                   2 = Log base E transformation
C*                   3 = Linear transformation.  Also required are two
C*                       additional values, N2 and N3.  The change is
C*                       XNEW = (XOLD-N2)/N3
C*                 The default value is no transformations.
C*
C*     USER       : Keyword to define a user name.  The user name can
C*                 contain up to 16 characters.  The correct format
```

Table A-1. Source Code Listing. (sheet 31 of 62)

```
C*                  is 'USER "user-name"', i.e., the name to be entered
C*                  is contained between double quotes.  The default name
C*                  is defined in subroutine IDENC.
C*     ------------------------------------------------------------------
C*
C*********************************************************************
C
*CALL SDATA
*CALL FILES
*CALL CDATA
*CALL IDENC
C
      LOGICAL CEXIST
      EXTERNAL CEXIST
C
      CHARACTER CARD*80, TMPNAM*16
C
C *** Patch in control file number to use RDBLK without modification
C
      IFIL = IDAT
C
C *** Set the line counter (ILINE) for subroutine RDBLK and the
C     maximum length (LENF) of a file=user name for subroutine FNAME,
C     and the maximum length (LENC) of a card image
C
      ILINE =  0
      LENC  = 80
      LENF  = 16
C
C ------------------------------------------------------------------
C                  Default values for the parameters
C ------------------------------------------------------------------
C
      CNE       =    0.0
      ICDF      =    0
      IPRNT(1)  =    0
      IPRNT(2)  =    0
      ISCA      =    0.0
      ITRANS    =    0
      NALPHA    =   35
      NLINE     =    1
      NSIZE     =    0
      NSTP      = 1000
      TVAL(1)   =    0.0
      TVAL(2)   =    0.0
C
      BADKEY    = .FALSE.
      DOPLOT(1) = .FALSE.
      DOPLOT(2) = .FALSE.
      DOPLOT(3) = .FALSE.
      DOPRNT(1) = .FALSE.
```

Table A-1. Source Code Listing. (sheet 32 of 62)

```
      DOPRNT(2) = .FALSE.
C
      IALIAS = 0
      ALEFT  = 0.0
      ARIGHT = 0.0
C
C *** ALPHA: Probability levels in the output table
C     Require : ALPHA(1) = 0.0 and ALPHA(NALPHA) = 1.0
C
      ALPHA( 1) = 0.0
      ALPHA( 2) =  .010
      ALPHA( 3) =  .015
      ALPHA( 4) =  .020
      ALPHA( 5) =  .025
      ALPHA( 6) =  .030
      ALPHA( 7) =  .040
      ALPHA( 8) =  .050
      ALPHA( 9) =  .075
      ALPHA(10) =  .100
      ALPHA(11) =  .150
      ALPHA(12) =  .200
      ALPHA(13) =  .250
      ALPHA(14) =  .300
      ALPHA(15) =  .350
      ALPHA(16) =  .400
      ALPHA(17) =  .450
      ALPHA(18) =  .500
      ALPHA(19) =  .550
      ALPHA(20) =  .600
      ALPHA(21) =  .650
      ALPHA(22) =  .700
      ALPHA(23) =  .750
      ALPHA(24) =  .800
      ALPHA(25) =  .850
      ALPHA(26) =  .900
      ALPHA(27) =  .925
      ALPHA(28) =  .950
      ALPHA(29) =  .960
      ALPHA(30) =  .970
      ALPHA(31) =  .975
      ALPHA(32) =  .980
      ALPHA(33) =  .985
      ALPHA(34) =  .990
      ALPHA(35) = 1.0
C
C ---------------------------------------------------------------------
C                Top of the loop on reading input cards
C ---------------------------------------------------------------------
C
   10 CONTINUE
C
```

Table A-1. Source Code Listing. (sheet 33 of 62)

```
      CALL RDBLK( IFIL, CARD )
C
C *** Check for the type of distribution to generate
C
      IF( NAME .EQ. 'DISTRIBU' ) THEN
C
        IF( CEXIST('EMPIRICA') .AND. CEXIST('KERNEL  ') ) THEN
          WRITE(IRPT,7035)
 7035     FORMAT(/
     *     ' *** Terminal error in subroutine INPUTS ***'/
     *     '     Both EMPIRICA and KERNEL were entered on the'
     *     '     DISTRIBU keyword card.')
          BADKEY = .TRUE.
          GO TO 10
        ENDIF
C
C ***   Check if the empirical CDF is to be used
C
        IF( CEXIST('EMPIRICA') ) THEN
          ICDF = 2
          GO TO 10
        ENDIF
C
C ***   Check if the kernel density estimator is to be used
C
        IF( CEXIST('KERNEL  ') ) THEN
          ICDF = 1
          IDX = 1
          IF( CEXIST('ALIAS   ') ) THEN
            IALIAS = VALUE(IDX)
            IDX = IDX + 1
            IF( IALIAS .EQ. 1 ) THEN
              ALEFT = VALUE(IDX)
              IDX = IDX + 1
            ENDIF
            IF( IALIAS .EQ. 2 ) THEN
              ARIGHT = VALUE(IDX)
              IDX = IDX + 1
            ENDIF
            IF( IALIAS .EQ. 3 ) THEN
              ALEFT = VALUE(IDX)
              IDX = IDX + 1
              ARIGHT = VALUE(IDX)
              IDX = IDX + 1
            ENDIF
          ENDIF
          IF( CEXIST('BANDWIDT') ) THEN
            CNE = VALUE(IDX)
            IDX = IDX + 1
          ENDIF
          IF( CEXIST('SCALE   ') ) THEN
```

```
      DATA APOS /''''/, BLANK /' '/
C
C *** Set the error flag and blank fill the file name
C
      IERR = 0
      DO 10 I = 1, LENF
        FILE(I) = BLANK
   10 CONTINUE
C
C *** Locate the apostrophes bracketing the file name
C
      I1 = 0
```


Table A-1. Source Code Listing. (sheet 34 of 62)

```
            ISCA = VALUE(IDX)
            IDX = IDX + 1
          ENDIF
          IF( CEXIST('STEPS   ') ) THEN
            NSTP = VALUE(IDX)
          ENDIF
          GO TO 10
        ENDIF
C
      ENDIF
C
C *** Check for end of the input deck
C
      IF( NAME .EQ. 'END     ' ) GO TO 20
C
C *** Check for end-of-file in the input deck
C
      IF( NAME .EQ. 'END-FILE' ) THEN
        CALL PAGER( IRPT, PRGNAM, PRGVER, CRUNID, USRNAM )
        WRITE(IRPT,7046)
 7046   FORMAT(/
     *     ' *** Terminal error in subroutine INPUTS ***'/
     *     '     Execution stop due to an unexpected'/
     *     '     end-of-file in the data file.')
        STOP
      ENDIF
C
C *** Optional plot of the CDF
C     The file name was defined in subroutine OPENER
C
      IF( NAME .EQ. 'PLOT    ' ) THEN
        IF( CEXIST('CUMULATI') ) DOPLOT(1) = .TRUE.
        IF( CEXIST('DENSITY ') ) DOPLOT(2) = .TRUE.
        IF( CEXIST('TABLE   ') ) DOPLOT(3) = .TRUE.
        IF( DOPLOT(1) .OR. DOPLOT(2) .OR. DOPLOT(3) )
     *    OPEN(IPLT,FILE=FILE9,STATUS='UNKNOWN')
        GO TO 10
      ENDIF
C
C *** Check if data is to be printed to the report file
C
      IF( NAME .EQ. 'PRINT' ) THEN
        IF( CEXIST('BEFORE  ') ) DOPRNT(1) = .TRUE.
        IF( CEXIST('AFTER   ') ) DOPRNT(2) = .TRUE.
        IF( CEXIST('RANGE   ') ) THEN
          IPRNT(1) = VALUE(1)
          IPRNT(2) = VALUE(2)
        ENDIF
        GO TO 10
      ENDIF
C
```

Table A-1. Source Code Listing. (sheet 35 of 62)

```
C *** Define new alpha levels
C
      IF( NAME .EQ. 'PROBABIL' ) THEN
C
        IF( CEXIST('TABLE   ') .AND. CEXIST('INCREMEN') ) THEN
          WRITE(IRPT,7036)
 7036     FORMAT(/
     *     ' *** Terminal error in subroutine INPUTS ***'/
     *     '     Both TABLE and INCREMEN were defined on the'
     *     '     PROBABIL keyword card.')
          BADKEY = .TRUE.
          GO TO 10
        ENDIF
C
        IF( CEXIST('TABLE   ') ) THEN
          NALPHA = VALUE(1)
          IF( NALPHA .GT. MXALP ) THEN
            WRITE(IRPT,7037) NALPHA, MXALP
 7037       FORMAT(/
     *     ' *** Terminal error in subroutine INPUTS ***'/
     *     '     Error on the keyword PROBABIL.  There were too'
     *     '     many entries in the probability table.'/
     *     '     Requested = ',I6,' Allowed (Parameter MXALP) = ',I6)
            BADKEY = .TRUE.
            GO TO 10
          ENDIF
          DO 17 I = 1, NALPHA
            ALPHA(I) = VALUE(I+1)
   17     CONTINUE
          GO TO 10
        ENDIF
C
        IF( CEXIST('INCREMEN') ) THEN
          PDEL = VALUE(1)
          IF( NALPHA .GT. MXALP ) THEN
            WRITE(IRPT,7037) NALPHA, MXALP
            BADKEY = .TRUE.
            GO TO 10
          ENDIF
          NALPHA = 1
   18     CONTINUE
          ALPHA(NALPHA) = (NALPHA-1)*PDEL
          IF( ALPHA(NALPHA) .GE. 1.0 ) THEN
            ALPHA(NALPHA) = 1.0
            GO TO 10
          ELSE
            NALPHA = NALPHA + 1
            IF( NALPHA .GT. MXALP ) THEN
              WRITE(IRPT,7037) NALPHA, MXALP
              BADKEY = .TRUE.
              GO TO 10
```

Table A-1. Source Code Listing. (sheet 36 of 62)

```
            ENDIF
            GO TO 18
          ENDIF
        ENDIF
C
      ENDIF
C
C *** Sample definition
C
      IF( NAME .EQ. 'SAMPLE  ' ) THEN
        NSIZE = VALUE(1)
        IF( CEXIST('LINE    ') ) NLINE = VALUE(2)
        GO TO 10
      ENDIF
C
C *** Get an new user name
C
      IF( NAME .EQ. 'USER    ' ) THEN
        CALL FNAME( CARD, LENC, TMPNAM, LENF, IERR )
        IF( IERR .EQ. 0 ) THEN
          USRNAM = TMPNAM
        ELSE
          WRITE(IRPT,7045) CARD, USRNAM
 7045     FORMAT(/
     *     ' *** Terminal error in subroutine INPUTS ***'/
     *     '     Bad user name from keyword USER'/
     *     '     Card = ',A80)
          BADKEY = .TRUE.
        ENDIF
        GO TO 10
      ENDIF
C
C *** Transformation option
C
      IF( NAME .EQ. 'TRANSFOR' ) THEN
        ITRANS = VALUE(1)
        TVAL(1) = VALUE(2)
        TVAL(2) = VALUE(3)
        GO TO 10
      ENDIF
C
C *** Error message for undefined keyword
C
      WRITE(IRPT,7200) NAME
 7200 FORMAT(/
     *     ' *** Terminal error in subroutine INPUTS ***'/
     *     '     The keyword (',A8,') is not defined.')
      BADKEY = .TRUE.
C
C *** Go try for another card
C
```

Table A-1. Source Code Listing. (sheet 37 of 62)

```
      GO TO 10
C
   20 CONTINUE
C
C ---------------------------------------------------------------------
C           Error checks and remaining default values
C ---------------------------------------------------------------------
C
      IF( NSTP .GT. MAXINT ) THEN
        WRITE(IRPT,7030) NSTP, MAXINT
 7030   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     The number of integration steps is greater than'/
     *   '     the maximum allowed.'/
     *   '     NSTP = ',I6,' Maximum dimension = ',I6/
     *   '     Change the parameter MAXINT and recompile.')
        BADKEY = .TRUE.
      ENDIF
C
      IF( NSTP .LT. 100 ) THEN
        WRITE(IRPT,7040) NSTP, MAXINT
 7040   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     The number of integration steps is too small'/
     *   '     NSTP = ',I6,' Maximum allowed = ',I6/
     *   '     Change the keyword KERNEL STEPS to at least 100.')
        BADKEY = .TRUE.
      ENDIF
C
C *** Error checks on NSIZE
C
      IF( NSIZE .GT. MAXDAT ) THEN
        WRITE(IRPT,7000) NSIZE, MAXDAT
 7000   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     The sample size is too large.'/
     *   '     Size = ',I6,' Maximum dimension = ',I6/
     *   '     Change the parameter MAXDAT and recompile.')
        BADKEY = .TRUE.
      ENDIF
C
      IF( NSIZE .LT. 4 ) THEN
        WRITE(IRPT,7010) NSIZE
 7010   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     The sample size is too small.'/
     *   '     Size = ',I6,' Minimum required = 4'/
     *   '     Obtain more data before running the program.')
        BADKEY = .TRUE.
      ENDIF
C
```

Table A-1. Source Code Listing. (sheet 38 of 62)

```
      IF( NLINE .LE. 0 ) THEN
        WRITE(IRPT,7012) NLINE
 7012   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     The number of data values on each line is negative.'/
     *   '     NLINE = ',I6,' Change the keyword "SAMPLE LINE".')
        BADKEY = .TRUE.
      ENDIF
C
C *** Stop for bad alpha levels
C
      IF( NALPHA.LT.2 .OR. NALPHA.GT.MXALP ) THEN
        WRITE(IRPT,7001) NALPHA
 7001   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     Bad value for the number of table entries'/
     *   '     NALPHA = ',I6/
     *   '     Change the keyword PROBABIL')
        BADKEY = .TRUE.
      ENDIF
C
      IF( ALPHA(1).NE.0.0 .OR. ALPHA(NALPHA).GT.1.0 ) THEN
        WRITE(IRPT,7002)
 7002   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     Bad entry for a probability level'/
     *   '     the table must start with 0.0 and end with 1.0'//
     *   '     The table is as follows.')
        WRITE(IRPT,7003) (ALPHA(I),I=1,NALPHA)
 7003   FORMAT(3X,5(2X,F6.4))
        BADKEY = .TRUE.
      ENDIF
C
C *** Default values on printing data values
C
      IF( DOPRNT(1) .OR. DOPRNT(2) ) THEN
        IF( IPRNT(1) .EQ. 0 ) IPRNT(1) = 1
        IF( IPRNT(2) .EQ. 0 ) IPRNT(2) = NSIZE
        IF( (IPRNT(1).GT.IPRNT(2)) .OR. (IPRNT(1).LT.1) .OR.
     *      (IPRNT(2).GT.NSIZE) ) THEN
          BADKEY = .TRUE.
          WRITE(IRPT,7021) IPRNT(1), IPRNT(2), NSIZE
 7021     FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     Illegal index range on keyword PRINT'/
     *   '     Range requested is ',I5,' to ',I5/
     *   '     There are ',I5,' data values.')
        ENDIF
      ENDIF
C
      IF( ICDF .EQ. 0 ) THEN
```

Table A-1. Source Code Listing. (sheet 39 of 62)

```
        WRITE(IRPT,7022)
 7022   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     No DISTRIBU card was entered or the card was invalid.')
        BADKEY = .TRUE.
      ENDIF
C
      IF( NALPHA .EQ. 0 ) THEN
        WRITE(IRPT,7122) NALPHA
 7122   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     No PROBABIL card was entered or the card was invalid.'/
     *   '     NALPHA = 0')
        BADKEY = .TRUE.
      ENDIF
C
      IF( IALIAS.EQ.3 .AND. (ALEFT.GE.ARIGHT) ) THEN
        WRITE(IRPT,7223) ALEFT, ARIGHT
 7223   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     The ALIAS option on the DISTRIBU card has the left'/
     *   '     boundary equal to or larger than the right boundary.'/
     *   '     ALEFT = ',1PE12.5,'  ARIGHT = ',E12.5)
        BADKEY = .TRUE.
      ENDIF
C
C *** Stop for bad keywords
C
      IF( BADKEY ) THEN
        CALL PAGER( IRPT, PRGNAM, PRGVER, CRUNID, USRNAM )
        WRITE(IRPT,7011)
 7011   FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     One or more bad keywords were encountered.')
        STOP 'BADKEY'
      ENDIF
C
C *** Default value for the bandwidth parameter
C
      IF( ICDF .EQ. 1 ) THEN
        IF( CNE .LE. 0.0 ) CNE = 2.283 * FLOAT(NSIZE)**(-0.287)
      ENDIF
C
C ------------------------------------------------------------------
C             Read in the sample data set
C ------------------------------------------------------------------
C
      NLEFT = MOD(NSIZE,NLINE)
      IF( NLEFT .EQ. 0 ) THEN
        DO 30 I = 1, NSIZE, NLINE
          READ(IDAT,*,END=40,ERR=40) (SVAL(I+J-1),J=1,NLINE)
```

Table A-1. Source Code Listing. (sheet 40 of 62)

```
   30   CONTINUE
      ELSE
        DO 35 I = 1, (NSIZE-NLEFT), NLINE
          READ(IDAT,*,END=40,ERR=40) (SVAL(I+J-1),J=1,NLINE)
   35   CONTINUE
        READ(IDAT,*,END=40,ERR=40) (SVAL(J),J=(NSIZE-NLEFT+1),NSIZE)
      ENDIF
C
C *** Normal termination for the input routine
C
      RETURN
C
C ---------------------------------------------------------------------
C        End-of-file or read error in the input data file
C ---------------------------------------------------------------------
C
   40 CONTINUE
C
      WRITE(IRPT,7080)
 7080 FORMAT(/
     *   ' *** Terminal error in subroutine INPUTS ***'/
     *   '     End-of-file or read error in the input file after'/
     *   '     reading the keywords but still reading sample data.')
      STOP
C
      END
*DECK OPENER
      SUBROUTINE OPENER
C*....SUBROUTINE OPENER
C*********************************************************************
C*
C*  Purpose
C*
C*    This subroutine opens input and output files.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 15 Jan 1990 : Original source
C*
C*
C*  File Definitions:
C*
C*      Unit  Definition
C*      ----  ------------------------------------
C*        7   File of control cards and sample data
C*        8   File containing table of the CDF
C*        9   Plot file of the CDF
C*
C*********************************************************************
C
```

Table A-1. Source Code Listing. (sheet 41 of 62)

```
*CALL FILES
*CALL IDENC
C
      LOGICAL THERE
C
C *** Assign FORTRAN unit numbers
C
      IDAT = 7
      IRPT = 8
      IPLT = 9
C
C *** Define the input and output files
C
      WRITE(*,1000) PRGNAM, PRGVER, CRUNID, USRNAM
 1000 FORMAT(
     * '            Cumulative Distribution Function Estimation'/
     * 1X,66('-')/
     * 1X,A8,1X,A8,'  Run ID = ',A12,'  User = ',A16/
     * 1X,66('-')/
     * ' There is 1 input file and 2 output files'/
     * ' The input file name must have the form "FILE.DAT"'/
     * ' The output file names will have the form "FILE.XXX"'/
     * ' The following file naming convention is used.'/
     * '   FILE.DAT : Input - control data and sample values'/
     * '   FILE.RPT : Report file'/
     * '   FILE.PLT : (optional) plot file for the CDF')
C
      WRITE(*,1010)
 1010 FORMAT(/' Enter the data file name (WITHOUT extension) > ')
      READ(*,1020) INFILE
 1020 FORMAT(A64)
C
C *** Form the concatenated file names
C
      CALL CONCAT( INFILE, 'DAT', FILE7 )
      CALL CONCAT( INFILE, 'RPT', FILE8 )
      CALL CONCAT( INFILE, 'PLT', FILE9 )
C
C *** Open the input file if it exists
C
      INQUIRE(FILE=FILE7,EXIST=THERE)
      IF( THERE ) THEN
        OPEN(IDAT,FILE=FILE7,STATUS='OLD')
      ELSE
        WRITE(*,1030) FILE7
 1030   FORMAT(/
     *   ' *** Terminal error in subroutine OPENER ***'/
     *   '     The requested input file does not exist.'/
     *   '     File = ',A64)
        STOP
      ENDIF
```

Table A-1. Source Code Listing. (sheet 42 of 62)

```
C
C *** Open the report file
C *** If requested, the plot file is opened in subroutine INPUTS
C
      OPEN(IRPT,FILE=FILE8,STATUS='UNKNOWN')
C
      RETURN
      END
*DECK PAGER
      SUBROUTINE PAGER( IUNIT, PRGNAM, PRGVER, CRUNID, USRNAM )
C*....SUBROUTINE PAGER( IUNIT, PRGNAM, PRGVER, CRUNID, USRNAM )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine writes a one line page header to an output file.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 15 Jan 1990 : Original source
C*
C*
C*  Variable definitions:
C*
C*    IUNIT  : Integer unit number for the output file
C*    PRGNAM : CHARACTER*8 program name
C*    PRGVER : CHARACTER*8 program version number
C*    USRNAM : CHARACTER*16 user name
C*    CRUNID : CHARACTER*12 run identification number
C*
C***********************************************************************
C
      CHARACTER*8  PRGNAM, PRGVER
      CHARACTER*12 CRUNID
      CHARACTER*16 USRNAM
C
      WRITE(IUNIT,7000) PRGNAM, PRGVER, USRNAM, CRUNID
 7000 FORMAT('1',
     * ' Program: ',A8,' Version:',A8,' User: ',A16,' Run ID: ',A12)
C
      RETURN
      END
*DECK PRNTER
      SUBROUTINE PRNTER( IPNDX )
C*....SUBROUTINE PRNTER( IPNDX )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine prints a table of the input data values to the
```

Table A-1. Source Code Listing. (sheet 43 of 62)

```
C*    report file.
C*
C*
C*  Call List Arguments:
C*
C*    IPNDX : Input Only
C*              IPNDX = 1 if PRNTER called before data transformations
C*              IPNDX = 2 if PRNTER called after data transformations
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 15 Jan 1990 : Original source
C*
C***********************************************************************
C
*CALL IDENC
*CALL FILES
*CALL SDATA
C
      CALL PAGER( IRPT, PRGNAM, PRGVER, CRUNID, USRNAM )
C
      WRITE(IRPT,1000)
 1000 FORMAT(/'     Data Used in Generating the Probability Table')
C
      IF( IPNDX .EQ. 1 ) THEN
        WRITE(IRPT,1010)
 1010   FORMAT('              Before Data Transformations'/)
      ENDIF
C
      IF( IPNDX .EQ. 2 ) THEN
        WRITE(IRPT,1020)
 1020   FORMAT('              After Data Transformations'/)
      ENDIF
C
      WRITE(IRPT,1030) (SVAL(I),I=IPRNT(1),IPRNT(2))
 1030 FORMAT(1P,5(1X,E13.5))
C
      RETURN
      END
*DECK RDBLK
      SUBROUTINE RDBLK( IFIL, CARD )
C*....SUBROUTINE RDBLK( IFIL, CARD )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This module reads a block of data from a keyword driven input
C*    data file (logical unit IFIL).  This will be a keyword card and
C*    and all subsequent continuation cards, to the next keyword card.
C*
```

Table A-1. Source Code Listing. (sheet 44 of 62)

```
C*
C*  Notes:
C*
C*    On input to this routine, it is assumed the next line of the
C*    input data file is already present in array INFO, except on the
C*    first call (ILINE=0).  ILINE must be initialized to zero by the
C*    main routine (or block data).  The array INFO must not be
C*    modified between calls to RDBLK.  It saves the previous keyword
C*    line from the input file.
C*
C*    Output from this set of subroutines includes the keyword from
C*    the keyword line (NAME, Char*8), the data values from that card
C*    and subsequent continuation cards (VALUE), and secondary keywords
C*    (modifiers) on the input card block (CVALUE).  All output
C*    keywords are stored in upper case characters.  All numeric values
C*    are stored as real numbers (not integer).
C*
C*    The routines RDBLK and UPCASE requires alphabetic characters to
C*    be 'represented' contiguously in memory (a through z, and A
C*    through Z).
C*
C*    In routine SEPCHR, a TAB character is a valid separator only if
C*    it converts to a character code of less than 10 decimal.
C*    ASCII code is generally expected, but not necessary if these
C*    conditions prevail.
C*
C*
C*  History:
C*
C*    Version 1.0, Module RDBLK
C*      Programmed by Dave Langford, BCSR, 28 March, 1989, FORTRAN F77
C*
C*    Version 2.0: 11 May 89,  CVALUE and NAME changed from CHAR*4
C*      variables to CHAR*8 variables.  Improves keyword flexibility.
C*      COMCHR added, to identify comment characters on input line.
C*
C*    Version 2.1: 14 Jun 1989,  Repetition factor added, allowing
C*      multiple entries in one statement (e.g., 15*1.0e+3).  Basic
C*      number decoder restructured.
C*
C*    Version 2.2: 15 Jan 1990,  Comments restructured and incorporated
C*      into the CDFTBL code.
C*
C*
C*  Variables:
C*
C*    CVALUE = Array of secondary keywords (modifiers) from the input
C*             data block.  Only the first eight characters are saved.
C*             (CHAR*8)
C*    IFIL   = Logical unit number for data read.
C*    ILINE  = Current line number of the input data file.
```

Table A-1. Source Code Listing. (sheet 45 of 62)

```
C*     INFO   = Saves the contents of the next line of the input file.
C*              (Dimension 80, Char*1)
C*     NAME   = Keyword from the current data block.  (CHAR*8)
C*     NCVALU = Number of secondary keywords (modifiers) from data block.
C*     NVALUE = Number of data values from the data block.
C*     CARD   = Returns the entire contents of the keyword line,
C*              following the first separator.
C*     VALUE  = Array of data values from the input data block.
C*              These are stored as real values (not integers).
C*
C*  Subroutines:
C*
C*     RDBLK   Controlling routine.
C*     CEXIST  Logical function to determine if an 8 character string
C*             is present among the secondary keywords (modifiers)
C*             contained in CVALUE.
C*     COMCHR  Identifies comment characters.
C*     DCDSEP  Parses the input line into numeric and character data.
C*     FINSEP  Locates separators in the input line.
C*     FNAME   Extracts a file name (character string) from a input
C*             keyword card.
C*     SAVTTL  Moves the entire contents of a keyword line, following
C*             the first set of separators, into an output character
C*             array.
C*     SEPCHR  Determines if a character is a legal separator.
C*     UPCASE  Converts single characters to upper case.  Depends on
C*             contiguous 'ordering' of alphabetic characters.
C*     XFKEY   Transfers a keyword from a character string to a char*8
C*             variable.  Blank fills for strings less than length 8.
C*
C***********************************************************************
C
*CALL CDATA
*CALL FILES
C
      CHARACTER*80 CARD
      DIMENSION LOC(80)
      EXTERNAL COMCHR, SEPCHR
      LOGICAL  COMCHR, SEPCHR
C
      IERR=0
C
C             READ FIRST LINE(S) OF DATA FILE, TO A KEYWORD CARD
C
      IF (ILINE.LE.0) THEN
  100    CONTINUE
         ILINE=ILINE+1
         READ (IFIL,500,END=470,ERR=480,IOSTAT=IERR) (INFO(I),I=1,80)
         IF (((INFO(1).LT.'A').OR.(INFO(1).GT.'Z')).AND.
     +       ((INFO(1).LT.'a').OR.(INFO(1).GT.'z'))) GO TO 100
      ENDIF
```

Table A-1. Source Code Listing. (sheet 46 of 62)

```
C
      NCVALU=0
      NVALUE=0
      DO 110 I=1,NKEYS
         CVALUE(I) = '        '
  110 CONTINUE
      DO 120 I=1,NVALS
         VALUE(I) = 0.0
  120 CONTINUE
C
C             MOVE KEYWORD INTO ARRAY NAME, WITH BLANK FILLING.
C
      CALL XFKEY (INFO(1),NAME)
C
C             MOVE LINE DATA INTO ARRAY CARD
C
      CALL SAVTTL (CARD)
C
C *** Return to the calling program if an END card is encountered
C     This is the graceful exit to signify the end of the input file
C
      IF( NAME .EQ. 'END' ) RETURN
C
C            LOCATE ALL SEPARATORS IN THIS LINE.
C            THIS IS TOP OF LOOP FOR READING CONTINUATION CARDS.
C
  130 CONTINUE
      CALL FINSEP (NUM,LOC)
C
C            MOVE CHARACTER DATA INTO ARRAY CVALUE,
C            AND NUMERIC DATA INTO ARRAY VALUE.
C
      IF (NUM.GT.1) CALL DCDSEP (NUM,LOC)
C
C             READ NEXT LINE AND REFORMAT
C
  140 CONTINUE
      ILINE=ILINE+1
      READ (IFIL,500,END=150,ERR=480,IOSTAT=IERR) (INFO(I),I=1,80)
C
C              RETURN IF NEXT KEYWORD CARD WAS FOUND
C
      IF (((INFO(1).GE.'A').AND.(INFO(1).LE.'Z')).OR.
     +    ((INFO(1).GE.'a').AND.(INFO(1).LE.'z'))) RETURN

C
C             IF COMMENT CARD, READ THE NEXT LINE.
C
      IF (COMCHR(1,INFO(1))) GO TO 140
C
C             CHECK FOR CONTINUATION CARD.  LOOP UP TO DECODE.
```

Table A-1. Source Code Listing. (sheet 47 of 62)

```
C
      IF (.NOT.SEPCHR(INFO(1))) THEN
         WRITE(IRPT,510)
         GO TO 490
      ENDIF
      GO TO 130
C
C             ARTIFICIAL END-FILE CARD GENERATED
C
  150 CONTINUE
      WRITE(IRPT,520)
      INFO(1)='E'
      INFO(2)='N'
      INFO(3)='D'
      INFO(4)='-'
      INFO(5)='F'
      INFO(6)='I'
      INFO(7)='L'
      INFO(8)='E'
      INFO(9)=' '
      RETURN
C
C             ERRORS
C
  470 CONTINUE
      WRITE(IRPT,970)
      GO TO 490
  480 CONTINUE
      WRITE(IRPT,980)
  490 CONTINUE
      WRITE(IRPT,990) ILINE,IERR
      STOP
C
  500 FORMAT(80A1)
  510 FORMAT(/
     *   ' *** Terminal error in subroutine RDBLK ***'/
     *   '     Error, A continuation card is expected, but the ',
     *   '     current line does not begin with a valid ',
     *   '     separator character.')
C
  520 FORMAT(/
     *   ' *** Warning error in subroutine RDBLK ***'/
     *   '     End-of-File encountered on the input file.'
     *   '     An artificial END-FILE card was generated.')
C
  970 FORMAT(/
     *   ' *** Terminal error in subroutine RDBLK ***'/
     *   '     End-of-File encountered in the input file'/
     *   '     prior to reading a keyword card.')
C
  980 FORMAT(/
```

Table A-1. Source Code Listing. (sheet 48 of 62)

```
     *   ' *** Terminal error in subroutine RDBLK ***'/
     *   '     Error reading data from input file.')
C
  990 FORMAT(/
     *   ' *** Terminal error in subroutine RDBLK ***'/
     *   '     Reading line number  ',I5/
     *   '     I/O Errors Status is ',I5/
     *   '     Error stop in routine RDBLK.')
      END
*DECK SAVTTL
      SUBROUTINE SAVTTL( CARD )
C*....SUBROUTINE SAVTTL( CARD )
C*********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine moves the contents of the input line, following
C*    the first separator, into the array CARD.
C*
C*    This routine is a part of the set of routines called by RDBLK.
C*
C*
C*  History:
C*
C*    David W. Langford : 29 Dec 1989 : Version 1.0
C*
C*********************************************************************
C
*CALL CDATA
C
      CHARACTER*80 CARD
      EXTERNAL SEPCHR
      LOGICAL  SEPCHR
C
      IST=0
      DO 100 I=1,80
         IF (IST.EQ.0) THEN
            IF (SEPCHR(INFO(I))) IST=I
         ELSE
            IF (.NOT.SEPCHR(INFO(I))) THEN
               IST=I
               GO TO 110
            ENDIF
         ENDIF
  100 CONTINUE
      WRITE(CARD,500) (INFO(I),I=1,80)
      RETURN
C
  110 CONTINUE
      NB=IST-1
      WRITE(CARD,500) (INFO(I),I=IST,80),(' ',I=1,NB)
```

Table A-1. Source Code Listing. (sheet 49 of 62)

```
      RETURN
  500 FORMAT(80A1)
      END
*DECK SCALE
      SUBROUTINE SCALE
C*....SUBROUTINE SCALE
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine will compute the scaled median absolute deviation
C*    for the data set.
C*
C*    This value is considered a statistically robust scale estimate.
C*
C*
C*  Inputs:
C*
C*    COMMON /SDATA/....
C*      SVAL  : Sample data vector - sorted in increasing size
C*      NSIZE : Number of values in the sample
C*
C*
C*  Outputs:
C*
C*    ISCA : Standard deviation estimate
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 15 Jan 1990 : Original source
C*
C***********************************************************************
C
*CALL SDATA
C
      LOGICAL EVEN
C
C *** Find if the sample size is odd or even
C
      IF( MOD(NSIZE,2) .EQ. 1 ) THEN
        EVEN = .FALSE.
      ELSE
        EVEN = .TRUE.
      ENDIF
C
C *** Location ( median )
C
      IF( EVEN ) THEN
        RMED = ( SVAL(NSIZE/2) + SVAL(NSIZE/2+1)) / 2.0
      ELSE
```

Table A-1. Source Code Listing. (sheet 50 of 62)

```
        RMED = SVAL( NSIZE/2 + 1 )
      ENDIF
C
C *** Scale = ( Median ABS(X-RMED) ) / 0.6745
C
      DO 10 I = 1, NSIZE
        STEMP(I) = ABS( SVAL(I) - RMED )
   10 CONTINUE
C
C *** Sort again
C
      KFLAG = 1
      CALL SORT( STEMP, NSIZE, KFLAG )
C
C *** Scale estimate
C
      IF( EVEN ) THEN
        ISCA = ( STEMP(NSIZE/2) + STEMP(NSIZE/2+1) ) / (2.0*0.6745)
      ELSE
        ISCA = STEMP(NSIZE/2+1) / 0.6745
      ENDIF
C
      RETURN
      END
*DECK SEPCHR
      LOGICAL FUNCTION SEPCHR( CH )
C*....LOGICAL FUNCTION SEPCHR( CH )
C*********************************************************************
C*
C*  Purpose:
C*
C*    This function determines if the character CH is a legal command
C*    line separator character.
C*
C*    This routine is a part of the set of routines called by RDBLK.
C*
C*
C*  History:
C*
C*    David W. Langford : 29 Dec 1989 : Version 1.0
C*
C*********************************************************************
C
      CHARACTER*1 CH
C
      IF( (ICHAR(CH).LT.10) .OR.
     +    (CH.EQ.' ') .OR. (CH.EQ.',')  .OR.
     +    (CH.EQ.'=') .OR. (CH.EQ.':')  .OR.
     +    (CH.EQ.';') .OR. (CH.EQ.'''') .OR.
     +    (CH.EQ.'(') .OR. (CH.EQ.')') ) THEN
         SEPCHR=.TRUE.
```

Table A-1. Source Code Listing. (sheet 51 of 62)

```
      ELSE
         SEPCHR=.FALSE.
      ENDIF
C
      RETURN
      END
*DECK SORT
      SUBROUTINE SORT( X, N, KFLAG )
C*....SUBROUTINE SORT( X, N, KFLAG )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine will sort the array X in increasing
C*    or decreasing order.
C*
C*
C*  Reference:
C*
C*    Singleton, R. C.
C*    Algorithm 347
C*    An Efficient Algorithm for Sorting with Minimal Storage
C*    C.A.C.M., 12(3), 1969, Pages 185-187.
C*
C*
C*  Variable Descriptions:
C*
C*    X     - Array of values to be sorted
C*    N     - Number of values in array X to be sorted
C*    KFLAG - Control parameter
C*              =  1 Means sort X in increasing order (ignoring Y)
C*              = -1 Means sort X in decreasing order (ignoring Y)
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 30 Mar 1988 : Extracted from subroutine SSORT
C*
C***********************************************************************
C
      DIMENSION X(N),IL(21),IU(21)
C
      NN = N
C
C *** Check for sort in increasing order
C
      IF (KFLAG .EQ. 1 ) GO TO 100
C
C *** Check for sort in decreasing order
C
      IF (KFLAG .NE. -1 ) STOP 'ERROR IN SORT - KFLAG'
```

Table A-1. Source Code Listing. (sheet 52 of 62)

```
C
C *** Alter array X to get decreasing order if needed
C
      DO 20 I = 1, NN
        X(I) = -X(I)
   20 CONTINUE
C
C *** Sort the vector
C
  100 CONTINUE
C
      M=1
      I=1
      J=NN
      R=.375
  110 IF (I .EQ. J) GO TO 155
  115 IF (R .GT. .5898437) GO TO 120
      R=R+3.90625E-2
      GO TO 125
  120 R=R-.21875
  125 K=I
C
C  *** Select a central element of the array and save it in location T
C
      IJ = I + IFIX (FLOAT (J-I) * R)
      T=X(IJ)
C
C *** If first element of array is greater than T, interchange with T
C
      IF (X(I) .LE. T) GO TO 130
      X(IJ)=X(I)
      X(I)=T
      T=X(IJ)
  130 L=J
C
C *** If last element of array is less than T, interchange with T
C
      IF (X(J) .GE. T) GO TO 140
      X(IJ)=X(J)
      X(J)=T
      T=X(IJ)
C
C *** If first element of array is greater than T, interchange with T
C
      IF (X(I) .LE. T) GO TO 140
      X(IJ)=X(I)
      X(I)=T
      T=X(IJ)
      GO TO 140
  135 TT=X(L)
      X(L)=X(K)
```

Table A-1. Source Code Listing. (sheet 53 of 62)

```
      X(K)=TT
C
C *** Find an element in the second half of the array which is
C *** smaller than T
C
  140 L=L-1
      IF (X(L) .GT. T) GO TO 140
C
C *** Find an element in the first half of the array which is
C *** greater than T
C
  145 K=K+1
      IF (X(K) .LT. T) GO TO 145
C
C *** Interchange these elements
C
      IF (K .LE. L) GO TO 135
C
C *** Save upper and lower subscripts of the array yet to be sorted
C
      IF (L-I .LE. J-K) GO TO 150
      IL(M)=I
      IU(M)=L
      I=K
      M=M+1
      GO TO 160
  150 IL(M)=K
      IU(M)=J
      J=L
      M=M+1
      GO TO 160
C
C *** Begin again on another portion of the unsorted array
C
  155 M=M-1
      IF (M .EQ. 0) GO TO 300
      I=IL(M)
      J=IU(M)
  160 IF (J-I .GE. 1) GO TO 125
      IF (I .EQ. 1) GO TO 110
      I=I-1
  165 I=I+1
      IF (I .EQ. J) GO TO 155
      T=X(I+1)
      IF (X(I) .LE. T) GO TO 165
      K=I
  170 X(K+1)=X(K)
      K=K-1
      IF (T .LT. X(K)) GO TO 170
      X(K+1)=T
      GO TO 165
```

Table A-1. Source Code Listing. (sheet 54 of 62)

```
C
C *** Clean up for sort in decreasing order
C
  300 IF( KFLAG .EQ. 1 ) RETURN
C
      DO 310 I = 1, NN
        X(I) = -X(I)
  310 CONTINUE
C
      RETURN
      END
*DECK TRANS
      SUBROUTINE TRANS
C*....SUBROUTINE TRANS
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine performs transformations on the input data
C*    set - if they are requested.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 15 Jan 1990 : Original source
C*
C*
C*  Variable definitions:
C*
C*    ITRANS : Variable transformation flag
C*             0 = No transformations
C*             1 = logarithm base 10
C*             2 = logarithm base e
C*             3 = linear transformation
C*             An error stop is generated for other values.
C*    NSIZE  : Number of sample values to transform
C*    SVAL   : Vector of length NSIZE containing the sample values
C*             which have been sorted in increasing algebraic order.
C*             The values are transformed and passed back in the
C*             same array locations.
C*
C***********************************************************************
C
*CALL SDATA
*CALL FILES
C
      IF( ITRANS .EQ. 0 ) RETURN
C
      IF( ITRANS.LT.1 .OR. ITRANS.GT.3 ) THEN
        WRITE(IRPT,7060) ITRANS
 7060   FORMAT(/
```

Table A-1. Source Code Listing. (sheet 55 of 62)

```
     *   ' *** Terminal error in subroutine TRANS ***'/
     *   '     A transformation was requested when ITRANS was not'/
     *   '     in the range 1 to 3.  Input value = ',I6)
        STOP
      ENDIF
C
C
C *** Common logarithm transformation
C
      IF( ITRANS .EQ. 1 ) THEN
C
C ***   Check for illegal data for the logarithm transformation
C
        IF( SVAL(1) .LE. 0.0 ) THEN
          WRITE(IRPT,6030) SVAL(1)
 6030     FORMAT(/
     *   ' *** Terminal error in subroutine TRANS ***'/
     *   '     Minimum data point = ',1PE13.6/
     *   '     Cannot use a logarithm transformation')
          STOP
        ENDIF
C
        DO 10 I = 1, NSIZE
          SVAL(I) = ALOG10( SVAL(I) )
   10   CONTINUE
      ENDIF
C
C *** Natural logarithm transformation
C
      IF( ITRANS .EQ. 2 ) THEN
C
C ***   Check for illegal data for the logarithm transformation
C
        IF( SVAL(1) .LE. 0.0 ) THEN
          WRITE(IRPT,6030) SVAL(1)
          STOP
        ENDIF
C
        DO 20 I = 1, NSIZE
          SVAL(I) = ALOG( SVAL(I) )
   20   CONTINUE
C
      ENDIF
C
C *** Linear transformation
C
      IF( ITRANS .EQ. 3 ) THEN
        DO 30 I = 1, NSIZE
          SVAL(I) = (SVAL(I)-TVAL(1))/TVAL(2)
   30   CONTINUE
      ENDIF
```

Table A-1. Source Code Listing. (sheet 56 of 62)

```
C
      RETURN
      END
*DECK TIES
      SUBROUTINE TIES
C*....SUBROUTINE TIES
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine checks for ties among input data values and
C*    performs an arbitrary spreading to eliminate the ties.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 1 Jun 1990 : Original source
C*
C*
C*  Variable definitions:
C*
C*    NSIZE  : Number of sample values
C*    SVAL   : Vector of length NSIZE containing the sample values
C*             which have been sorted in increasing algebraic order.
C*
C***********************************************************************
C
*CALL SDATA
*CALL FILES
C
      LOGICAL TIE
C
C *** Check for the existence of ties in the data set
C
      TIE = .FALSE.
      DO 10 I = 2, NSIZE
        IF( SVAL(I) .EQ. SVAL(I-1) ) THEN
          IF( .NOT. TIE ) THEN
            TIE = .TRUE.
            J = I-1
            K = I
            IF( I .EQ. NSIZE ) CALL TIE2( J, K )
          ELSE
            K = I
            IF( I .EQ. NSIZE ) CALL TIE2( J, K )
          ENDIF
        ELSE
          IF( TIE ) THEN
            CALL TIE2( J, K )
            TIE = .FALSE.
          ENDIF
```

Table A-1. Source Code Listing. (sheet 57 of 62)

```
        ENDIF
   10 CONTINUE
C
      RETURN
      END
*DECK TIE2
      SUBROUTINE TIE2( J, K )
C*....SUBROUTINE TIE2( J, K )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine is called from subroutine TIE to eliminate ties
C*    among data values in the input set.  The approach is to shift the
C*    input data values in locations J through K by a slight amount so
C*    that they no longer have identical values.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 1 Jun 1990 : Original source
C*
C*
C*  Variable definitions:
C*
C*    NSIZE : Number of sample values
C*    SVAL  : Vector of length NSIZE containing the sample values
C*            which have been sorted in increasing algebraic order.
C*    J     : Index of the first data in the set of tied values.
C*    K     : Index of the last data in the set of tied values.
C*
C***********************************************************************
C
*CALL SDATA
C
      M  = K - J + 1
C
      XINC = 0.01
      IF( J .EQ. 1 ) THEN
        DELX = XINC * (SVAL(K+1)-SVAL(J)) / M
      ELSE IF( K .EQ. NSIZE ) THEN
        DELX = XINC * (SVAL(K)-SVAL(J-1)) / M
      ELSE
        DELX = XINC * (SVAL(K+1)-SVAL(J-1)) / M
      ENDIF
C
      DO 10 I = J, K
        IF( J .EQ. 1 ) THEN
          DINC = I - J
        ELSEIF( K .EQ. NSIZE ) THEN
          DINC = I - K
```

Table A-1. Source Code Listing. (sheet 58 of 62)

```
        ELSE
          DINC = I - J - FLOAT(M)/2 + 0.5
        ENDIF
        SVAL(I) = SVAL(I) + DINC*DELX
   10 CONTINUE
C
      RETURN
      END
*DECK UPCASE
      SUBROUTINE UPCASE( CH )
C*....SUBROUTINE UPCASE( CH )
C*********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine takes the character CH and ensures it is upper
C*    case.  It also assumes that character storage codes are
C*    alphabetically contiguous.
C*
C*
C*  History:
C*
C*    Dave Langford :  1 Jun 1989 : Original Source
C*    Paul Eslinger : 15 Jan 1990 : Add explicit character numbers
C*
C*********************************************************************
C
      CHARACTER*1 CH
C
C *** Determine the offset between 'A' and 'a' to use in the expression
C         CH = CHAR( ICHAR(CH) + ICHAR('A') - ICHAR('a') )
C *** The ASCII Character set uses ICHAR('A')=65 and ICHAR('a')=97
C
      IF( (CH.GE.'a') .AND. (CH.LE.'z') ) CH = CHAR(ICHAR(CH)-32)
C
      RETURN
      END
*DECK WRITER
      SUBROUTINE WRITER
C*....SUBROUTINE WRITER
C*********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine writes the final results of the program to the
C*    report file.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 15 Jan 1990 : Original source
```

Table A-1. Source Code Listing. (sheet 59 of 62)

```
C*
C***********************************************************************
C
*CALL SDATA
*CALL FILES
*CALL IDENC
C
C-----------------------------------------------------------------------
C             Write results to the report file
C-----------------------------------------------------------------------
C
C *** Page header
C
      CALL PAGER( IRPT, PRGNAM, PRGVER, CRUNID, USRNAM )
C
C *** Write the table of probability levels for different X
C
      IF( ICDF .EQ. 1 ) THEN
        WRITE(IRPT,1000) NSIZE, SN, CNE
 1000   FORMAT(//
     *  5X,'Cumulative Probabilities Using Kernel Density Estimation'//
     *  5X,' The density estimate was based on ',I5,' values.'/
     *  5X,' The scale estimate was ',1PE11.4,0P/
     *  5X,' The kernel bandwidth value was ',F7.5)
      ENDIF
C
      IF( ICDF .EQ. 2 ) THEN
        WRITE(IRPT,1005) NSIZE
 1005   FORMAT(//
     *  5X,'Cumulative Probabilities Using The Empirical Distribution'//
     *  5X,' The density estimate was based on ',I5,' values.')
      ENDIF
C
      WRITE(IRPT,1007) NALPHA
 1007 FORMAT(5X,' The probability table has ',I3,' pairs of values.')
C
      WRITE(IRPT,1010)
 1010 FORMAT(///10X,'              Cumulative'/
     *          10X,'     X        Probability'/
     *          10X,'-----------   -----------')
C
      DO 10 I = 1, NALPHA
        WRITE(IRPT,1020) XPLOT(I), ALPHA(I)
 1020   FORMAT(10X,1P,E11.4,3X,2X,0P,F6.4)
   10 CONTINUE
C
C-----------------------------------------------------------------------
C                Write results to the plot file
C             Each plot is preceded by a page header
C-----------------------------------------------------------------------
C
```

Table A-1. Source Code Listing. (sheet 60 of 62)

```
      IF( DOPLOT(1) ) THEN
        CALL PAGER( IPLT, PRGNAM, PRGVER, CRUNID, USRNAM )
        WRITE(IPLT,1040)
 1040   FORMAT(' Cumulative Distribution Function')
C
        IF( ICDF .EQ. 1 ) THEN
          DO 20 I = 1, NSTP
            WRITE(IPLT,1030) XINT(I), FINT(I)
 1030       FORMAT(1X,1PE12.5,2X,0PF8.5)
   20     CONTINUE
        ENDIF
C
        IF( ICDF .EQ. 2 ) THEN
          FN = NSIZE
          XD = (SVAL(NSIZE-1)-SVAL(2)) / 1000.0
          X = SVAL(1) - XD
          P = 0.0
          WRITE(IPLT,1030) X, P
          DO 30 I = 1, NSIZE
            X = SVAL(I)
            P = (I-1)/FN
            WRITE(IPLT,1030) X, P
            P = I/FN
            WRITE(IPLT,1030) X, P
   30     CONTINUE
          X = SVAL(NSIZE)+XD
          P = 1.0
          WRITE(IPLT,1030) X, P
        ENDIF
C
      ENDIF
C
C-----------------------------------------------------------------------
C
      IF( DOPLOT(2) ) THEN
        IF( ICDF .EQ. 1 ) THEN
          CALL PAGER( IPLT, PRGNAM, PRGVER, CRUNID, USRNAM )
          WRITE(IPLT,1050)
 1050     FORMAT(' Kernel Density Function')
          DO 40 I = 1, NSTP
            WRITE(IPLT,1070) XINT(I), GHAT(I)
 1070       FORMAT(1X,1P,E12.4,1X,E12.4)
   40     CONTINUE
        ENDIF
      ENDIF
C
C-----------------------------------------------------------------------
C
      IF( DOPLOT(3) ) THEN
        CALL PAGER( IPLT, PRGNAM, PRGVER, CRUNID, USRNAM )
        WRITE(IPLT,1060)
```

Table A-1. Source Code Listing. (sheet 61 of 62)

```
 1060   FORMAT(' Cumulative Probability Table')
        WRITE(IPLT,1065) NALPHA
 1065   FORMAT(1X,I4)
        DO 50 I = 1, NALPHA
          WRITE(IPLT,1030) XPLOT(I), ALPHA(I)
   50   CONTINUE
      ENDIF
C
      RETURN
      END
*DECK XFKEY
      SUBROUTINE XFKEY( INFO, KEYWRD )
C*....SUBROUTINE XFKEY( INFO, KEYWRD )
C***********************************************************************
C*
C*  Purpose:
C*
C*    This subroutine examines the first eight characters in array INFO
C*    in order to move that data into variable KEYWRD.  If a separator
C*    is present, the relevant characters are transferred to variable
C*    KEYWRD, and the remainder of KEYWRD is blank filled.
C*
C*    This routine is a part of the set of routines called by RDBLK.
C*
C*
C*  History:
C*
C*    David W. Langford : 29 Dec 1989 : Version 1.0
C*
C***********************************************************************
C
      CHARACTER*1 INFO(8), JNFO(8)
      CHARACTER*8 KEYWRD
      EXTERNAL SEPCHR
      LOGICAL  SEPCHR
C
      DO 110 I=1,8
         JNFO(I)=INFO(I)
         IF (SEPCHR(INFO(I))) THEN
            DO 100 J=I,8
               JNFO(J)=' '
  100       CONTINUE
            GO TO 120
         ENDIF
  110 CONTINUE
  120 CONTINUE
C
      DO 130 I=1,8
         CALL UPCASE (JNFO(I))
  130 CONTINUE
      WRITE(KEYWRD,500) (JNFO(I),I=1,8)
```

Table A-1. Source Code Listing. (sheet 62 of 62)

```
  500 FORMAT(8A1)
C
      RETURN
      END
```

Table A-2. Listing of the VAX Version of Subroutine IDENC (sheet 1 of 3)

```
*DECK IDENC
      SUBROUTINE IDENC
C*....SUBROUTINE IDENC
C*********************************************************************
C*
C*  Purpose:
C*
C*    This routine generates identification variables for this run
C*    of the code.  Variables are:
C*
C*      1. A unique run identifier number, utilizing the date
C*         and time which this routine was called
C*
C*      2. Hard coded - Program name and version number
C*
C*      3. Hard coded - User name
C*
C*      4. Hard coded - Date that program was last modified
C*
C*      5. System date and time
C*
C*
C*  Operating system:
C*
C*    This subroutine is designed for the FORTRAN 77 compiler using
C*    a VMS operating system.
C*
C*
C*  Variable Definitions:
C*
C*    CRUNID = CHARACTER*12 output run identifier, as follows, with
C*             digit 1 being the leftmost digit.
C*      Digits  1- 2 = Year
C*      Digits  3- 4 = Month
C*      Digits  5- 6 = Day (1 to 31)
C*      Digits  7- 8 = Hour (0 to 23)
C*      Digits  9-10 = Minute (0 to 59)
```

Table A-1. Source Code Listing. (sheet 2 of 3)

```
C*      Digits 11-12 = Second (0 to 59)
C*
C*    PRGNAM = CHARACTER*8 name of the program
C*    PRGVER = CHARACTER*8 version number of the program
C*    PRGDAT = CHARACTER*12 date stamp for program completion
C*    SYSDAT = CHARACTER*8 System date in the format mm-dd-yy
C*    SYSTIM = CHARACTER*8 System time in the format hh:mm:ss
C*    USRNAM = CHARACTER*16 name of the current user.
C*
C*
C*  History:
C*
C*    Paul W. Eslinger : 18 Jan 1990 : Version 1.0
C*
C*
C***********************************************************************
C
*CALL IDENC
C
C *** User name
C
      USRNAM = 'XXXXXX X. XXXXXX'
C
C *** Program name and version number
C
      PRGNAM = 'CDFTBL'
      PRGVER = '1.0'
C
C *** Program date
C
      PRGDAT = '13 Jun 1990'
C
C *** System time in the (CHARACTER) form hh:mm:ss
C
      CALL TIME( SYSTIM )
C
C *** System date in the (CHARACTER) form mm:dd:yy
C
      CALL IDATE(IMO,IDA,IYR)
      WRITE(SYSDAT,1000) IYR,IMO,IDA
 1000 FORMAT(I2,':',I2,':',I2)
C
C *** Create run identification number
C
      CRUNID( 1: 2) = SYSDAT(1:2)
      CRUNID( 3: 4) = SYSDAT(4:5)
      CRUNID( 5: 6) = SYSDAT(7:8)
      CRUNID( 7: 8) = SYSTIM(1:2)
      CRUNID( 9:10) = SYSTIM(4:5)
      CRUNID(11:12) = SYSTIM(7:8)
C
```

Table A-1. Source Code Listing. (sheet 3 of 3)

```
C *** Substitute a zero for any blank
C
      DO 10 I=1,12
        IF( CRUNID(I:I) .EQ. ' ' ) CRUNID(I:I) = '0'
   10 CONTINUE
C
      RETURN
      END
```

DISTRIBUTION

Number of copies		
OFFSITE		
5	U.S. Department of Energy 1000 Independence Avenue, S.W. Forrestal Building Washington, D.C. 20585	
	L. P. Duffy	EM-1
	C. Frank	EM-50
	S. Lien	EM-54
	J. Lytle	EM-30
	R. P. Whitfield	EM-40
1	U.S. Department of Energy 12800 Middlebrook Road Germantown, Maryland 20874	
	K. Hain	EM-54
12	U.S. Department of Energy Office of Scientific and Technical Information P.O. Box 62 Oak Ridge, Tennessee 37831	
1	Bureau of Economic Geology University of Texas at Austin University Station Box X Austin, Texas 78713	
	B. Scanlon	
1	Colorado School of Mines Department of Chemistry and Geochemistry Golden, Colorado	
	D. Langmuir	
3	EG&G Idaho, Inc. P.O. Box 1625 MS 2107 Idaho Falls, Idaho 83415	
	R. G. Baca	
	R. R. Seitz	
	J. C. Walton	

DISTRIBUTION (cont)

Number of copies		
OFFSITE		
2	Lawrence Livermore National Laboratory University of California P.O. Box 808 Livermore, California 94550	
	K. Pruess L. D. Ramspott	
1	Los Alamos National Laboratory P.O. Box 1663 Los Alamos, New Mexico 87545	
	B. J. Travis	
1	New Mexico State University Department of Agronomy P.O. Box 30003 Las Cruces, New Mexico 88003	
	R. Hills	
1	Oak Ridge National Laboratory P.O. Box Y Oak Ridge, Tennessee 37830	
	J. O. Blomeke	
1	Office of Civilian Radioactive Waste Management Forrestal Building Washington, D.C. 20585	
	K. H. Kale	RW-20
3	Office of Remedial Action and Waste Technology 19901 Germantown Road Germantown, Maryland 20585	
	J. A. Coleman	EM-35
	T. W. McIntosh	EM-343
	H. F. Walter	EM-343

DISTRIBUTION (cont)

Number of copies	
OFFSITE	
11	Southwest Research Institute Division 20 6220 Calebra Road P.O. Drawer 0510 San Antonio, Texas 28510-78228 W. Patrick B. Sagar (10)
2	University of Arizona Department of Water Resources Tucson, Arizona N. J. Aimo S. P. Neuman
2	University of California at Riverside Department of Soils Riverside, California 92502 W. A. Jury M. Th. van Genuchten
1	University of Massachusetts Department of Plant and Soil Science 12A Stockbridge Hall Amherst, Massachusetts 01003 D. Hillel
1	U.S. Environmental Protection Agency Washington, D.C. D. Sherwood B5-01
1	U.S. Environmental Protection Agency Office of Radiation Programs (ANR-458) 401 M Street, S.W. Washington, D.C. 20460 S. Myers
1	U.S. Geological Survey Federal Center, MS-413 Denver, Colorado 80225 E. P. Weeks

DISTRIBUTION (cont)

Number of copies

OFFSITE

1	U.S. Geological Survey Water Resources Division Low-Level Radioactive Waste Program 12201 Sunrise Valley Drive Reston, Virginia 22092	
	J. Fischer	
2	U.S. Nuclear Regulatory Commission Division of Engineering Safety Washington, D.C. 20555	
	S. Coplan	NLS-260
	T. J. Nicholson	NL-005
2	Westinghouse Idaho Nuclear Company, Inc. P.O. Box 4000 Idaho Falls, Idaho 83401	
	J. A. Berreth	
	D. A. Knecht	

ONSITE

7	U.S. Department of Energy-Richland Operations Office	
	J. C. Bartlett	A1-10
	R. D. Freeberg	A5-19
	M. J. Furman	A5-21
	J. M. Hennig	A5-21
	S. K. Moy	A5-21
	G. W. Rosenwald	A4-02
	Public Reading Room	A1-65
25	Pacific Northwest Laboratory	
	S. Q. Bennett	K6-35
	M. P. Bergeron	K6-77
	P. W. Eslinger	K6-96
	J. W. Falco	K6-78
	M. J. Fayer (2)	K6-77
	M. D. Freshley	K6-77
	P. C. Hays	K6-86

DISTRIBUTION (cont)

Number of copies

ONSITE

Pacific Northwest Laboratory (cont)

	D. J. Holford	K6-77
	J. F. Keller	K1-51
	C. T. Kincaid	K6-77
	W. E. Nichols	K6-77
	K. R. Roberson	K6-77
	M. L. Rockhold	K6-77
	C. S. Simmons	K6-77
	R. L. Skaggs	K6-77
	J. L. Smoot	K6-77
	J. A. Stottlemyre	K6-75
	S. K. Wurstner	K6-77
	Technical Files (5)	K1-11
	Clearance Office	K1-02

Westinghouse Hanford Company

24	L. C. Brown	H4-51
	J. W. Cammann	H4-14
	L. B. Collard	H4-14
	M. P. Connelly	H4-14
	J. D. Davis	H4-55
	K. R. Fecht	H4-56
	R. L. Jackson	H4-56
	R. Khaleel	H4-14
	N. W. Kline	H0-31
	D. Langford	B2-25
	A. G. Law	H4-56
	T. LeGore	B4-63
	R. E. Lerch	B2-35
	J. C. Sonnichsen (5)	H4-14
	D. D. Wodrich	B3-72
	R. D. Wojtasek	L4-92
	Central Files	L8-04
	Document Clearance Administration	H4-17
	Document Processing and Distribution (2)	L8-15

END

DATE FILMED

11/26/91