# THE NORTHEASTERN FOREST-INVENTORY DATA-PROCESSING SYSTEM. IV. INFORMATION FOR PROGRAMMERS SUBSYSTEM EDIT.

by
Robert C. Peters and
Robert W. Wilson Jr.

U. S. FOREST SERVICE RESEARCH PAPER NE-72
1967

NORTHEASTERN FOREST EXPERIMENT STATION, UPPER DARBY, PA.
FOREST SERVICE, U.S. DEPARTMENT OF AGRICULTURE
RICHARD D. LANE, DIRECTOR

## About the Authors

ROBERT C. PETERS obtained his Bachelor's degree from the University of California in 1960 and his Master's at Yale University in 1961. He joined the Forest Service in 1961 as a research forester, and was assigned to the Station's biometrics unit from 1961 until 1965, when the unit was discontinued. Mr. Peters played a key role in the development of the data-processing system reported here.

ROBERT W. WILSON, JR. took his Bachelor's degree at The Pennsylvania State University in 1947 and his Master's and Ph.D. degrees at Yale University in 1948 and 1965, respectively. He joined the U. S. Forest Service in 1948 and has worked in various research capacities for the Northeastern Forest Experiment Station. From 1961 to 1965 he was in charge of the Station's biometrics unit at New Haven, Conn. He is assigned at present to the Forest Insect and Disease Laboratory at West Haven, Conn.

# THE NORTHEASTERN FOREST-INVENTORY DATA-PROCESSING SYSTEM. IV. INFORMATION FOR PROGRAMMERS SUBSYSTEM EDIT.

■

## Contents

## PREFACE

THIS paper is the fourth in a series of ten papers prepared to describe the forest-inventory data-processing system of the Northeastern Forest Experiment Station. This system was devised for using modern, large-scale, high-speed computers in processing forest-inventory data. The series will comprise the following papers:

I. Introduction.
II. Description of subsystem EDIT.
III. Operation of subsystem EDIT.
IV. Information for programmers — subsystem EDIT.
V. Description of subsystem TABLE.
VI. Operation of subsystem TABLE.
VII. Information for programmers—subsystem TABLE.
VIII. Description of subsystem OUTPUT.
IX. Operation of subsystem OUTPUT.
X. Information for programmers — subsystem OUTPUT.

# IV-A. INTRODUCTION

ONE of the major projects of the U.S. Forest Service is a nationwide forest survey, which is designed to obtain useful and timely information about the timber resources of the United States. In the course of the surveys, which are made mainly on a state-by-state basis, great masses of detailed data are collected about timber volumes, growth, timber cut, and other characteristics of the timber resource.

In recent years the volume of information obtained from forest-survey field plots has increased greatly. The task of compiling and analyzing this mass of data with mechanical computing machines was both cumbersome and time-consuming.

A solution to this problem was seen in the development of the high-speed electronic computers. The Northeastern Forest Experiment Station, which was responsible for conducting the forest survey of the heavily forested Northeastern States, investigated the possibilities and devised the Northeastern Forest-Inventory Data-Processing System.

This paper presents information for programmers on a part of the system, subsystem EDIT. A general description of program EDIT and detailed instructions for its use in solving data-processing problems are given in parts II, and III of this series. In the following chapters will be found selected programming information that will be useful if the programs must be modified for any reason. The program write-ups and information about the program source decks may be obtained from the Northeastern Forest Experiment Station, 6816 Market Street, Upper Darby, Pennsylvania 19082.

The program is written in the standard IBM FORTRAN IV language, and is operative at the Yale University Computer Center on an IBM 7094/7040 Direct Coupled System under the IBSYS DCS operating system with IBJOB processor.[1] It will operate with little or no modification on other comparable sytems. The main requirements for a machine on which to operate the standard

[1] Mention of a particular product should not be construed as an endorsement by the Forest Service or the U. S. Department of Agriculture.

version of the program are a 32K word core, a minimum of 36 bits per word, binary arithmetic capability, and 5 tape drives or equivalent input/output devices.

## IV-B. USE OF THE CALCULATE OPERATION

The standard version of program EDIT provides a dummy subroutine named CALCUL. This dummy subroutine may be replaced by another of the same name programmed in FORTRAN IV to generate values for new data fields in each record, or to change the values in existing (input) data fields. The subroutine has the following calling sequence and DIMENSION statement:

```
      SUBROUTINE CALCUL (NDATA, PDATA, KKVAR)
      DIMENSION NDATA(KKVAR), PDATA(KKVAR),
     1  FDATA(27), FREOUT(27), KTITLE(12),
      COMMON FDATA, FREOUT, KTITLE, ISTEP
```

The arrays NDATA and PDATA contain the data vector (the unit record, as stored in the computer). In the standard version of program EDIT, both arrays are 132 cells in length. To retrieve or to store information, the user must reference either PDATA or NDATA by appropriate data-field identification numbers. If the information is a fixed-point quantity, the name NDATA is used. If it is a floating-point quantity, the name PDATA is used.

For example, the following subroutine adds the values in data fields 1 and 2, and stores the resulting value in data field 27; and it divides the value in data field 101 by the value in data field 11, and stores the resulting value in data field 15:

```
      SUBROUTINE CALCUL (NDATA, PDATA, KKVAR)
      DIMENSION NDATA(KKVAR), PDATA(KKVAR),
     1  FDATA(27), FREOUT(27), KTITLE(12),
      COMMON FDATA, FREOUT, KTITLE, ISTEP
      NDATA(27) = NDATA(1) + NDATA(2)
      PDATA(15) = PDATA(101)/PDATA(11)
      RETURN
      END
```

The first (add) arithmetic statement illustrates an operation

done entirely in fixed-point arithmetic. The data fields in the right-hand side of the statement must have been read as fixed-point numbers or, if read as floating-point numbers, must have been converted to fixed point by the FIX operation. The second (divide) arithmetic statement illustrates an operation done entirely in floating-point arithmetic. The data fields in the right-hand side of the statement must have been read as floating-point numbers or, if read as fixed-point numbers, must have been converted to floating point by the FLOAT operation.

The variable ISTEP is the current program step. If more than one calculate operation is used, ISTEP can be used to branch to the appropriate part of the calculate routine.

## IV-C. MODIFICATION OF DIMENSIONED SPACE

The standard version of program EDIT carries restrictions on both the dimensions and the overall size of problem that can be handled in a single processing run. These restrictions are a result of the manner in which dimensioned space has been allocated (table 1) and the total space available in a given operating system. The program has been written so that all modifications of dimen-

**Table 1. — Summary of dimensioned-space restrictions and associated program variable and arrays**

| Item | Restriction | Variable | Arrays |
|---|---|---|---|
| Maximum number of cells in all input and output table entries | 8,000 | NDIMEN | ZIMP, IMP |
| Maximum number of fields/record | 132 | KKVAR | PDATA, NDATA |
| Maximum number of input and output tabes | 30 | NTBLE | NTAB, PNTAB |
| Maximum number of program operations | 100 | NOPER | NOVAR, NCONST, NAME, PNAME, CONST, IPROG |

sioned space can be made in the subprogram called MAINE. No other parts of the program need be touched for this purpose. The use of dimensioned space and the means of changing dimensions are discussed in detail below.

### Number of Data Fields per Input Record

In the standard version of program EDIT, up to 132 data fields can be used for input and output. To change this maximum the following steps must be taken:

1. In the subprogram MAINE, the variable named KKVAR must be set to the desired maximum value.
2. In the subprogram MAINE, the DIMENSION statement must be changed so that the dimension of arrays PDATA and NDATA equal the desired maximum value.

### Number of Program Operations

In the standard version of program EDIT, up to 100 operations can be used. To change this maximum, the following steps must be taken:

1. In the subprogram MAINE, the variable named NOPER must be set equal to the desired maximum value.
2. In the subprogram MAINE, the DIMENSION statement must be changed so that the first dimension of arrays NOVAR, NCONST, NAME, PNAME, and CONST, equals the desired maximum value. The dimension of array IPROG must also equal the desired maximum value.

### Number of Input and Output Tables

In the standard version of program EDIT, up to 30 input and output tables are allowed. To change this maximum the following steps must be taken:

1. In the subprogram MAINE, the variable named NTBLE must be set to the desired maximum value.
2. In the subprogram MAINE, the DIMENSION statement must be changed so that the first dimension of arrays NTAB and PNTAB equal the desired maximum value.

### Number of Cells in All Input and Output Tables

In the standard version of program EDIT, up to 8,000 locations are available for storing all input and output tables. To change this maximum the following steps must be taken:

1. In the subprogram MAINE, the variable named NDIMEN must be set to the desired maximum value.
2. In the subprogram MAINE, the DIMENSION statement must be changed so that the dimension of arrays, ZIMP and IMP equal the desired maximum value.

If the space required for the input and output table specified in a given control deck exceeds 8,000 locations, message 4 or message 23 (see part III-C) will be printed during the reading of the control deck and processing will halt. The space required can be computed as follows:

$$\sum_{i=1}^{M} (IT_i) + \sum_{j=1}^{N} (ADD_j)$$

Where

- $M$ = Total number of input tables.
- $IT_i$ = Total number of entries in the ith input table.
- $N$ = Total number of output tables (tables defined in an ADD operation).
- $ADD_j$ = Total number of cells in the jth output table (tables defined in an ADD operation).

## IV-D. PROGRAMMING FEATURES

The following information will be helpful to programmers who plan to modify the standard version of program EDIT for use on other computers or under other operating systems.

### Tape Assignments

In program EDIT the FORTRAN logical tape assignments are as follows:

| *Unit* | *Use* |
|---|---|
| 5 | Monitor input for program deck and job control deck. |
| 6 | Monitor print for error records, other messages, and tables |
| 7 | Monitor punch for update tables. |
| 15 | Input data in binary or BCD. |
| 19 | Output of correct records in binary or BCD |

These tape assignments can be changed to fit local conditions by loading appropriate file routines with the program. See your systems representative or the section entitled FORTRAN files in the IBM IBJOB processor manual, file number 7090-27.

## Use of Sense Switches and Sense Lights

No sense switches are used in program EDIT. All sense switches will be set at normal monitor setting.

Sense light 1 is turned on when an error is found in an input record. Sense light 2 is turned on when an error is found in scanning the job control deck. No other sense lights are used.

## Use of Program Halts

There are no halts in program EDIT.

## Use of the Overlay Feature

The standard version of the program is constructed so that the overlay feature can be used when sufficient storage is not available for the program and the data. The configuration is as follows:

| *Link* | *Contains Subprograms* |
|---|---|
| 0 | MAINE, PACK, UNPACK, RESTAR |
| 1 | MAGIC |
| 2 | CONTRL |
| 3 | EDIT, LEFTAB, ALLTAB, ERCARD |
| 4 | ENDMAG |

## Bit Manipulation

The standard version of program EDIT uses 36-bit fixed-point words. Some input tables (those used in the cross check, cross range check, generate, and add operations) are stored by packing several fields into a single word. Some operation controls (those for FIX, FLOAT, LOGIC, and ARITHE) are stored by packing

data-field identification numbers and operation options into the same word.

Words are packed and unpacked by the use of binary arithmetic. The program will not operate on a machine that uses either a larger or smaller number of bits in fixed-point operations unless the shift constants that are used to pack and unpack are changed to correspond with the number of bits per word in the particular machine.

## Subprogram Names and Functions

| | |
|---|---|
| MAINE. | The main calling sequence of program EDIT. It sets array dimensions; and calls MAGIC, CONTRL, EDIT, and ENDMAG — in that order. |
| MAGIC. | Reads all input table and operation cards. Checks for consistency and sets up necessary information for using each operation called. |
| CONTRL. | Reads update table (if present) and all control cards containing input and output record description cards. Sets up all necessary information for input/output. |
| EDIT. | Processes all input data according to operation described in control deck. |
| ENDMAG. | Reads all table output control cards and prints any table requested. |
| PACK. | Packs each field in a table entry with a 36-bit word. Called by EDITM and MAGIC. |
| ERCARD. | Writes out records which are in error. |
| LEFTAB. | Searches in IMP, ZIMP, array when leftmost table field is the argument. |
| ALLTAB. | Searches table in IMP, ZIMP array when entire entry is used as an argument. |
| UNPACK. | Unpacks a 36-bit word composed of a packed data field or a table entry. |
| RESTAR. | Punches out ADD tables (for restart or update) at end of processing, or if number of errors is greater than expected. |
| CALCUL. | A dummy subroutine which is called by the calculate operation. It may be programmed by the user to perform any record processing operations not provided for by the other program operations. |

## Important Arrays and Variables

The following are the principal arrays and variables used in program EDIT:

| *Array* | *Dimension* | *Description* |
|---|---|---|
| NW, W | 9 | Fixed and floating-point working storage |
| NPSTEP | 1 | Total number of program operations. |
| ISTEP | 1 | The index or identification number (sequential) of the program operation currently being executed. |
| IVAR | 1 | The number of input data fields. |
| NTAB, PNTAB | NTBLE x 15 | Indexing information for the named tables; where NTBLE is the number of input tables and the second dimension locations are used as follows:<br>1 = index of the first table value in IMP.<br>2 = index of the final table value in IMP.<br>3 = index of table mid-point.<br>4 = index of first quartile.<br>5 = index of third quartile.<br>6-14 = the number of positions, in octal, occupied by each field of a table entry.<br>15 = the number of fields in a table entry. |
| CVT | 8 | Field widths in a table entry, in bits. |
| FMT | 28 | Working storage for general format specifications. |
| NWORK | 9 | Working storage. |
| IPROG | NOPER | Identification number of the program operation used in each program step. |
| NOVAR | NOPER x 10 | Identification numbers of each data field used in each program step, in order listed on operation cards. |
| NCONST CONST | NOPER x 2 | Value of the constant used in each program step, if any. The value referred to as CO1 on the operation card is stored in the first location of the second dimension, and CO2 is stored in the second location. |
| NAME, PNAME | NOPER x 4 | Identification of input tables used in each program step, if any; where the locations of the second dimension are used as follows:<br>1 = the identification number of the first table used in the program step.<br>2 = the identification number of the second table used in the program step.<br>3 = the alphameric name of the first table.<br>4 = the alphameric name of the second table. |
| FDATA | 27 | Storage for the read-in format specification for input data and/or error output. |
| IMP, ZIMP | NDIMEN | Storage array for all input and output tables. |
| NDATA, PDATA | KKVAR | Storage array for all data fields (input and generated) of the unit record currently being processed. |
| IWIDTH, PWIDTH | 13 | Storage array for output-table scale factors. |
| FREOUT | 27 | Storage array for correct record output format specification. |

| | | |
|---|---|---|
| KTITLE | 12 | Storage array for the read-in title for all printed output. |
| NMODE | 1 | Number identifying the mode in which the input data is written: 1 equals binary; 2 equals BCD. |
| NTVAR | 1 | Total number of data fields output for a correct record. |
| NCHAR | 1 | Number identifying the output options:<br>1 = no correct record output.<br>2 = correct record output written in binary mode.<br>3 = correct record output written in BCD mode.<br>4 = correct record output punched. |

■