DOE/PE/79009--T20

DOE/PE/79009--T20

DE89 011437

WRATES

Programmer's Manual

~~Draft II~~

August 1988

Prepared by
N. Roukos
R. Ziemer

Meta Systems Inc.
58 Charles Street
Cambridge, MA 02141
617-621-0580

Final
Per Anna Barba
per Jeffrey Shaw, PE-30
FC01-86 PE 79009
M

NOTICE

This report was prepared by Meta Systems Inc in the course of performing work contracted for and sponsored by the New York State Energy Research and Development Authority (hereafter the "Authority"). The opinions expressed in this report do not necessarily reflect those of the Authority or the State of New York and reference to any specific product, service, process or method does not necessarily constitute an implied or expressed recommendation or endorsement of same. Further, the Authority and the State of New York make no warranties or representations, expressed or implied, as to the fitness for particular purpose, merchantability of any product, apparatus or service or the usefulness, completeness or accuracy of any processes, methods or other information described, disclosed or referred to in this report. The Authority and the state of New York make no representation that the use of any product, apparatus, process, method or other information will not infringe privately owned rights and will assume no liability for damages resulting upon any information contained in this report.

CONTENTS

ILLUSTRATIONS

TABLES

Section 1

EXECUTIVE SUMMARY

Wheeling is the transmission of electrical energy from a seller to a buyer through transmission lines owned by the wheeling utility. The Wheeling Rate Evaluation Simulator (WRATES) is a computer program developed by Meta Systems to determine the price of wheeling services based on the true cost of wheeling. WRATES can evaluate four types of wheeling:

- Utility to Utility, through the transmission network of one or more interconnected wheeling utilities or pools.
- Utility to Private User, where the private user may or may not be located within the utility
- Private Generator to Utility, where the private generator may or may not be located inside the wheeling utility
- Private Generator to Private User, where both, one or none of the wheeling parties are within the wheeling utility

In light of recent inquiries by the Federal Energy Regulatory Commission (FERC) regarding transmission access, phases like "economic wheeling" and "common carrier" have become subjects of utmost interest. WRATES can provide valuable information about such vital issues to policy makers, regulators, industrials and utilities.

One key issue underlying the wheeling debate is the determination of the rates a wheeling utility should charge. WRATES is a tool for evaluating wheeling rates that are based on:

- Marginal Operating Costs with
- Revenue Reconciliation for Capital Recovery.

The marginal operating costs are determined by fuel cost, transmission losses and operational costs of dealing with generation and line

capacity limits. Line capacity limits affect the costs of system redispatch to prevent line overloads and/or the capital costs of building new lines. Generation capacity limits, when present, result in emergency purchases, activation of interruptible contracts or other similar actions. If marginal operating costs as described above are charged, they may over-recover or under-recover revenue requirements. Therefore, they generally have to be modified to achieve revenue reconciliation; i.e. adjusted so that the utility or pool recovers the "allowed" operating and capital costs. Revenue reconciliation for transmission embedded capital can be done for the transmission system as a whole or on a line by line basis which is especially useful for new transmission lines. Revenue reconciliation for embedded generation costs can be incorporated when the wheeling utility has an "obligation to serve" one or both parties involved in the wheeling.

The theory underlying WRATES is directly applicable to the complex networks of generating plants, transmission lines and distribution areas. However, WRATES is programmed for the evaluation of simplified networks which can provide a useful and realistic approximation of these very complex, real-world networks and is intended for policy studies which explore different approaches and evaluate the general nature of the wheeling rates.

WRATES handle network flows and losses using either a DC load flow approximation for 25 buses and 200 lines or the results of an AC load flow for a much larger network. When working with its internal DC load flow, WRATES can model up to 5 independently dispatched entities (either utilities or pools), each with its own Automatic Generation Control.

The basic theory underlying WRATES evolved from the theory of a spot price based energy marketplace where private users and private generators within a given utility pay or are paid at rates based on marginal operating costs with revenue reconciliation. However, WRATES does not assume the existence of a spot price based energy marketplace. The results of WRATES can be used to derive time-of-use (TOU) rates, constant rates over the period of the wheeling contract, or any other structured rates that vary with certain system characteristics without being completely dynamic. The derived rates can be applied to any market irrespective of the existing rate structure.

Section 2

INTRODUCTION

This document is the programmer's manual for version 01 of the Wheeling Rate Evaluation Simulator "WRATES".

COMPUTER MODEL

Because of its size, WRATES is implemented as a set of four (4) modules: SCENGEN, ELDM, COMPUTM, and IWHEEL (Figure 2-1). Sequential and direct access files provide the medium for transmitting information between the modules (see Section 7).

The user prepares the input data for WRATES in a LOTUS™ 123 environment. The LOTUS™ 123 macro \W creates a sequential DOS file, SCENGEN.DAT, which will hold the input data to SCENGEN.

SCENGEN has three (3) main functions: checking the input data, creating the scenario data, and writing the output to both the scenario file, FSCEN.FIL, and the spot price file, FSPOT.FIL. Both the scenario and the spot price files are direct access files. If SCENGEN detects an error, it continues the data checking and prints the error messages in the sequential file ERROR.FIL but halts the execution of the other modules of WRATES. Otherwise, SCENGEN produces a complete set of ELDM input data for each scenario and stores it in the scenario file, FSCEN.FIL.

ELDM is the next module in line. It is called and executed as many times as there are scenarios. It reads the input data from the FSCEN.FIL and develops an economic generation dispatch for the conditions before and/or after wheeling. The resulting spot prices are written to FSPOT.FIL. Warning messages are stored in WARNING.FIL; bus generations, line flows, line losses, and fuel costs are written to the DOS file ELDM.OUT. If there are pools and the wheeling rate is to apportioned among members of the pool, a direct access file, FSPLIT.FIL, is created to hold the results of an economic dispatch for a wheeling quantity equal to the specified quantity + 10 MW.

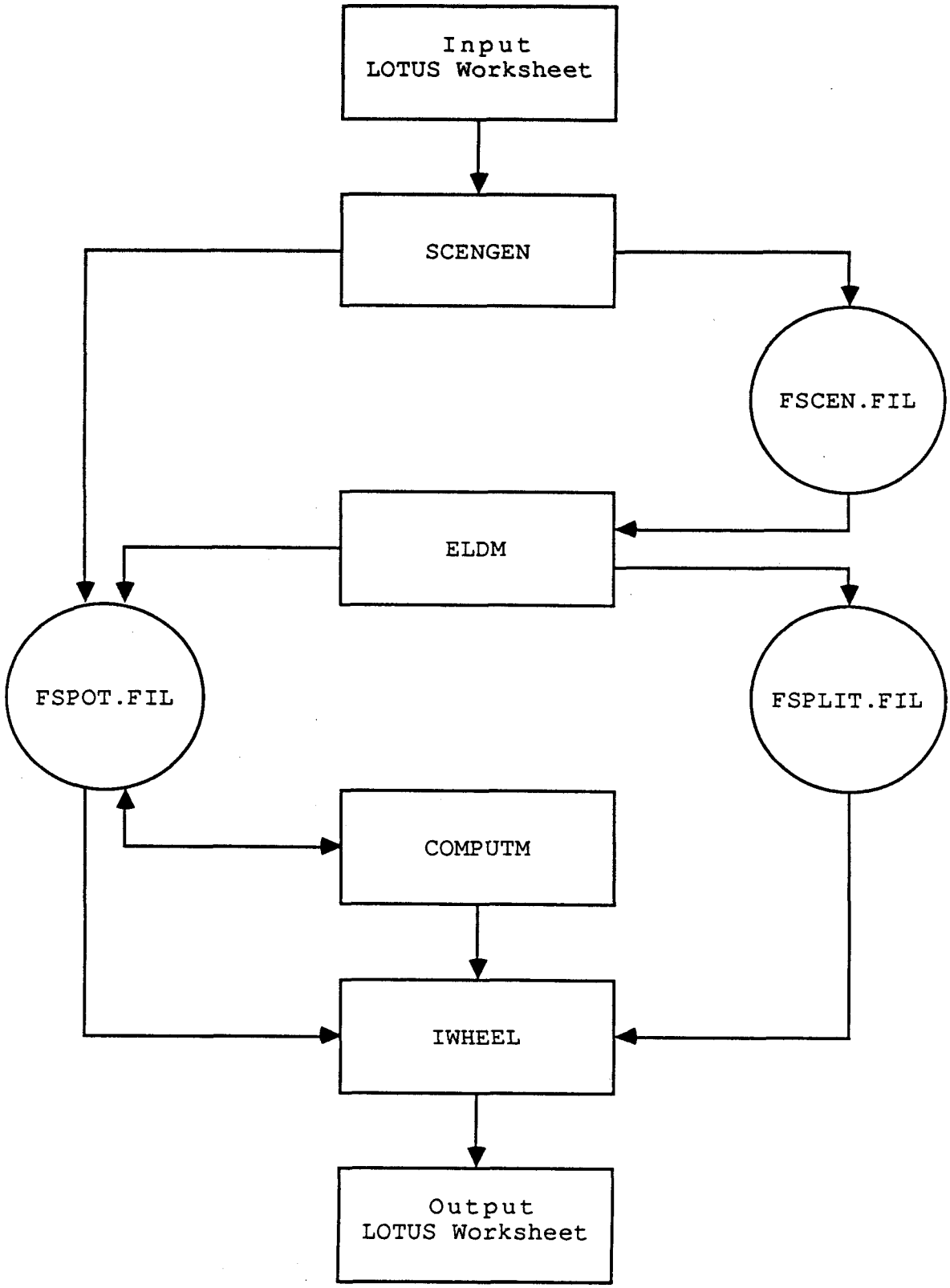

Figure 2-1 WRATES Flow Diagram

COMPUTM is called next. If all of the revenue reconciliation multipliers are provided, or if the user did not request revenue reconciled rates, COMPUTM is exited. Otherwise, COMPUTM reads from FSPOT.FIL the pre-wheeling spot prices for each scenario and accumulates the yearly gross revenues as well as the yearly fuel costs. It then computes the requested revenue reconciliation multipliers and writes them in FSPOT.FIL.

IWHEEL is the last module of WRATES. For each scenario it reads the spot prices for the economic dispatch that includes the wheeling transaction. It computes the ideal wheeling rate for each utility and pool as well as the reconciled rate when required, and writes them to IWHEEL.OUT. After all the scenarios are processed, IWHEEL computes the coordinates of points describing the ideal and the reconciled wheeling rate duration curves for each of the wheeling utilities, and writes them to FDURC.FIL.

Then LOTUS^{TM} 123 is invoked and the macro \0 is automatically called to sort the wheeling rates by utility and display them in a worksheet format. Macro \G can also be invoked to graph the wheeling rate duration curves.

COMPUTER ENVIRONMENT AND REQUIREMENTS

WRATES was developed on an IBM/AT operating under DOS. It is coded in FORTRAN 77. Its input is prepared in LOTUS^{TM} 123 and its output is also displayed on a LOTUS^{TM} 123 spreadsheet. WRATES was compiled and debugged using the Lahey F77$_L$ compiler. The executable modules were created using the PC-DOS "link" command of DOS 3.2.

Hardware Requirements

Running WRATES requires an IBM AT or a compatible computer with:

- 640 Kbytes of memory;
- A math coprocessor;
- A hard disk;
- A 1.2 Megabyte (high density) floppy disk drive (because WRATES is provided on "high density" diskettes);
- A monitor (monochrome or color);

- A printer that can print 132 characters per line either with a wide carriage or via the compressed mode.

Software Requirements

The software requirements of WRATES are:

- DOS operating System version 3.0 or higher;
- LOTUS™ 123 version 2.0 or higher (WRATES does not run from Symphony);
- Lahey F77$_L$ compiler version 2.2 or higher (this is needed if WRATES is recompiled on the user's computer).

MODULE DESIGN

The following guidelines were used for the design of all four modules of the program:

- The source code for each module is stored in a separate directory that has the same name as the module. For example, the source code for the SCENGEN module is in directory \SCENGEN.
- All the common variables of a module are defined in the file COMMON.FOR. This file is included at the beginning of every subroutine of that module.
- All the common variables of a module are initialized in the block data subprogram stored in the BLKDATA.FOR file of the module directory.
- All the array dimensions are specified in a "parameter" statement located at the beginning of the COMMON.FOR file. Therefore, changing those dimensions requires only modifying the parameter statement in each module (a total of 4) instead of changing the dimension of each affected variable in both COMMON.FOR and BLKDATA.FOR of each module.
- The flow of a module is controlled by the main routine stored in the file called "Module name.FOR." For example, the main program of the COMPUTM module is stored in COMPUTM.FOR. All the module's subroutines are called by the main routine which is the only routine allowed to call any subroutine.
- Each subroutine is stored in a separate DOS file, the name of which is "subroutine.FOR". For example, subroutine SODS is stored in the DOS file SODS.FOR.
- All the input/output files used in a module are opened at the beginning of the main routine of that module.

Section 3

SCENARIO GENERATION MODULE "SCENGEN"

The Scenario generation module, SCENGEN, is the first module of WRATES. It is called by the batch file WRATES.BAT after the input data file SCENGEN.DAT is created from the LOTUS™ 123 input spreadsheet. Its function is to create a complete set of input data to ELDM for each scenario, and to write this data to the direct access scenario file FSCEN.FIL. The flow chart of SCENGEN is shown in Figure 3-1. A key to the symbols used in the flow charts is contained in the list of symbols preceding Section 1.

PROGRAM STRUCTURE

The flow of the SCENGEN module is controlled by the main program stored in SCENGEN.FOR which calls a total of ten (10) subroutines. The input data to this module is stored in SCENGEN.DAT and it is organized as follows. First, there is a section describing the base case data; it has the title block, the pool affiliation data, the wheeling transaction data, the net energy data, the bus data and finally the line data. This data does not represent any particular state of the electrical system, it is rather a block of data that needs to be modified for each scenario. Second, there is a data section for each scenario; it has the demand at each bus and the data that differ from the base case. SCENGEN first reads then checks and stores the base case data. The module then reads the input data for each scenario and combines it with the base case data to create a complete set of input data to ELDM for each scenario. The scenario data is written to the scenario file FSCEN.FIL. During its execution, SCENGEN checks the input data. If an error occurs, no further writing to the scenario file is allowed, but the reading and checking of the input data continues until either the number of errors exceeds the maximum allowed, or the end of the data is reached. The following is a description of the function of each subroutine in SCENGEN in the same order as they are called by the main routine in SCENGEN.FOR.

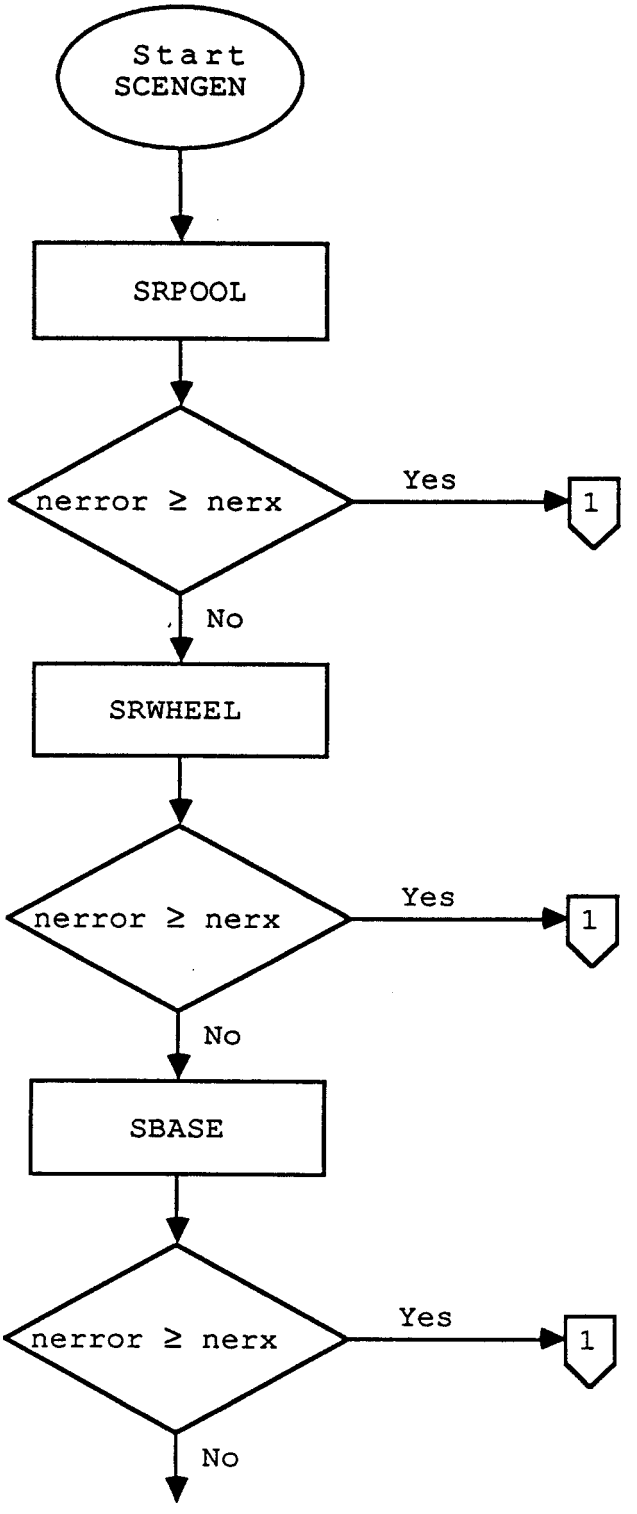

Figure 3-1. Flow Chart for the SCENGEN Module

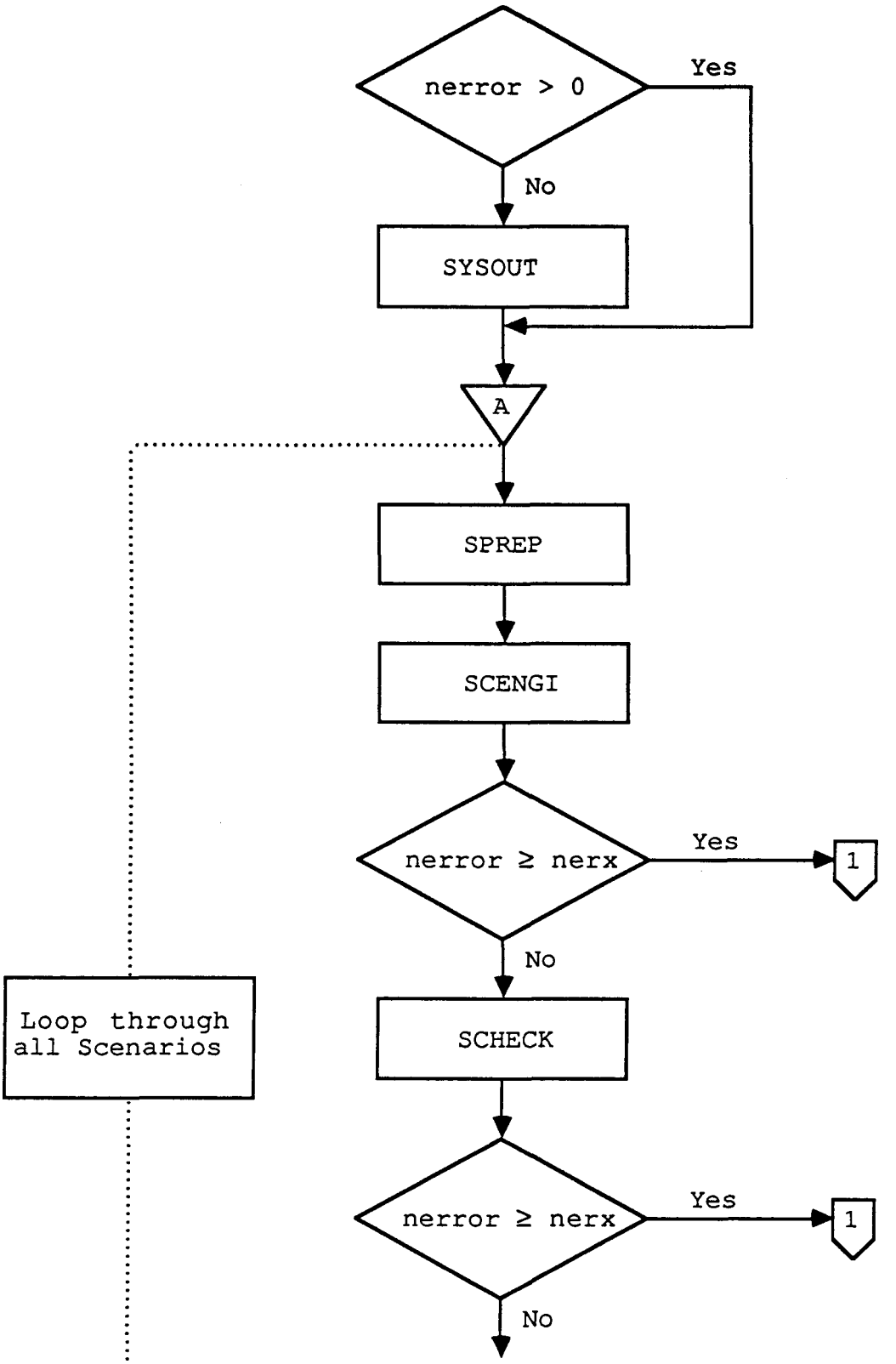

Figure 3-1 (continued). Flow Chart for the SCENGEN Module

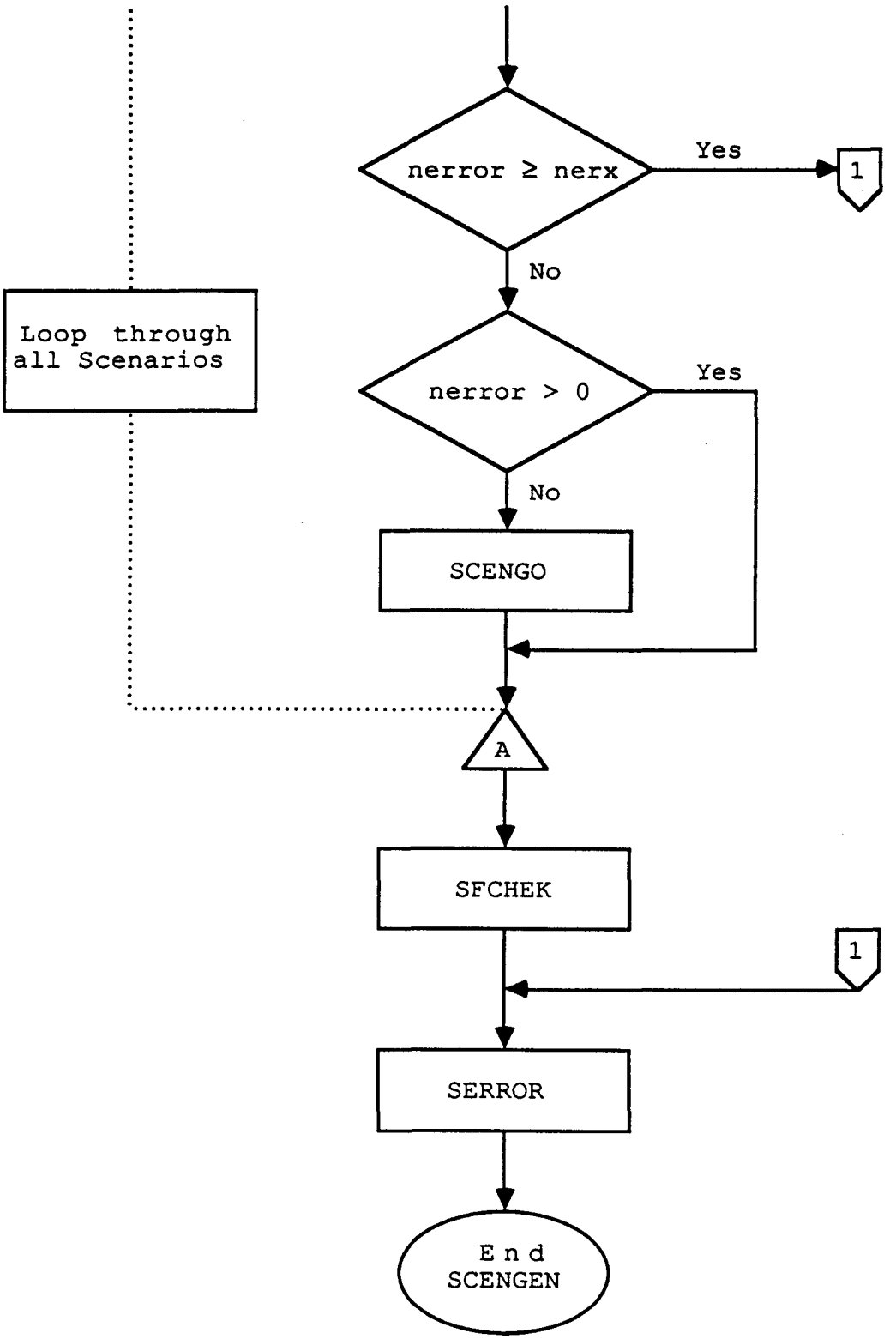

Figure 3-1 (Continued). Flow Chart for the SCENGEN Module

Subroutine SRPOOL

This subroutine reads the title of the study, the number of buses, lines and pool in the system as well as the pool affiliation of each utility if any. A utility could be either independently dispatched, or could belong to a pool in which case it is centrally dispatched with all the other members of the pool. SRPOOL also checks the pool affiliation data for consistency and completeness.

Subroutine SRWHEEL

This subroutine is called after SRPOOL if the number of errors detected so far does not exceed the allowed member of errors. SRWHEEL reads the quantity to be wheeled as well as the location of the buyer and seller. It also reads the wheeling reconciliation parameters of the wheeling utilities. Note that the wheeling reconciliation is not allowed for pools or their member utilities.

Subroutine SBASE

This subroutine is called after SRWHEEL if the number of errors detected so far does not exceed the allowed number of errors. It reads the base case data section and checks it for consistency and completeness.

Subroutine SYSOUT

This subroutine is called after SBASE, and only if no errors are detected so far. SYSOUT writes the system record (record 1) and the pool record (record 2) to the scenario file, FSCEN.FIL. It also writes the system record (record 1), the pool record (record 2) and the revenue reconciliation record (record 3) to the spot price file, FSPOT.FIL.

Subroutine SPREP

This subroutine is called once for each scenario right before the scenario data is read. It sets the demand at each bus to zero and copies the net energy data, the bus data, and the line data from the arrays holding the base case data to the arrays that will hold the

scenario data.

Subroutine SCENGI

This subroutine reads the scenario data. It reads the demand at each bus, the modifications to the net energy interchange data, the marginal cost curve data, the cost of unserved energy data and the line data. It then stores the data in the arrays prepared by subroutine SPREP.

Subroutine SCHECK

This subroutine checks the consistency and validity of the scenario data. For a list of the errors that are detected, see Section 9.

Subroutine SCENGO

This subroutine is called only if the input data is free of errors so far. The function of this subroutine is to write the scenario data in the direct access scenario file FSCEN.FIL. All the data related to one scenario are stored in one record of FSCEN.FIL. This record has a number equal to the scenario number plus two (2); e.g., the data related to scenario 5 will be stored in record 7 of FSCEN.FIL.

Subroutine SFCHEK

This subroutine is called once after all the input data is read. It performs further checking of the input data. For a list of the errors SFCHEK detects, see Section 9.

Subroutine SERROR

This subroutine is called either when the number of errors detected is equal to the maximum number (NERX) set by the user in BLKDATA.FOR, or just before exiting the program. SERROR writes an error number and a short message in SERROR.FIL for each error detected by SCENGEN. For a list of the errors detected by SCENGEN, see Section 9.

LOTUS^{TM} 123 INTERFACE

The input data for the SCENGEN module of the WRATES program is entered into a LOTUS^{TM} 123 spreadsheet. A LOTUS^{TM} 123 macro, \W, is provided to transfer the input data to a DOS sequential ASCII file called SCENGEN.DAT. This file constitutes the input to the SCENGEN module

which is coded in FORTRAN.

The macro \W must be located in all of the input spreadsheets. In the blank input spreadsheet provided with WRATES, WRATES.WK1, the macro \W is located beginning in cell C1000. This location will probably be different in other input spreadsheets, depending on the number of rows inserted by the user.

During its execution, the macro performs seven steps.

- Step 1: The user is prompted as to whether or not he/she wishes to save the spreadsheet. It is saved if so desired by the user.
- Step 2: Blank cells are replaced with zeroes in the wheeling data portion of the spreadsheet, where some input data were optional.
- Step 3: The data is printed to the file SCENGEN.DAT. If the file already exists, it is overwritten; if the file does not exist, it is created. At this point SCENGEN.DAT contains data, headers and other labelling text.
- Step 4: Except for the title of the case study, the data portion of the spreadsheet is erased, and the numbers from the SCENGEN.DAT file are re-imported into the spreadsheet.
- Step 5: Blank rows appearing between the data are removed from the spreadsheet.
- Step 6: The data portion of the spreadsheet is again printed to the SCENGEN.DAT file.
- Step 7: The macro exits LOTUS^{TM} 123.

After exiting LOTUS^{TM} 123, control returns to the WRATES batch file. The resulting file, SCENGEN.DAT, is an ASCII text file in the format required for the input to the SCENGEN module.

COMMON BLOCKS AND VARIABLE DICTIONARY

This section describes the common variables used in the SCENGEN module. Two files define and initialize the variables in this module. The first, COMMON.FOR, defines the common blocks. The code in this file is included in all the subroutines of the SCENGEN module. The second, BLKDATA.FOR, is a subprogram that initializes the variables defined in the common blocks. Table 3-1 is a listing of the common

variables of SCENGEN. The array variables are dimensioned by parameters that are set in a parameter statement in COMMON.FOR. These parameters are:

- o ns maximum number of scenarios
- o nlx maximum number of lines
- o nbx maximum number of buses
- o nux maximum number of utilities
- o npx maximum number of pools
- o nptx maximum number of points on the supply curves
- o niux maximum number of independently dispatched entitites
- o nerx maximum number of errors
- o nupx maximum number of independently dispatched entitites

Table 3-1

COMMON VARIABLES IN THE SCENGEN MODULE

Variable	Dimension	Type	Definition
Common Block /CSYS/			
NBUS		I	number of buses
NLINE		I	number of lines
NUTIL		I	total number of utilities
NPOOL		I	number of pools
NIUTIL		I	number of independently dispatched utilities
NUP		I	number of economically dispatched entities
NWUTIL		I	total number of wheeling utilities including pool affiliated ones
NWLINE		I	total number of lines for which wheeling revenues or revenue reconciliation multipliers are to be computed
Common Block /CLINE/			
XRES(i)	i=1,nlx	R	resistance of line i (Ohms/V^2)
XINDUC(i)	i=1,nlx	R	inductance of line i (Ohms/V^2)
IBEGB(i)	i=1,nlx	I	number of the beginning bus of line i
IENDB(i)	i=1,nlx	I	number of the end bus of line i
ICONST(i)	i=1,nlx	I	type of constraint for line i 0 - no constraint 1 - soft constraint 2 - hard constraint
XFLPOS(i)	i=1,nlx	R	positive flow limit for line i (MW)
XPEPOS(i)	i=1,nlx	R	penalty parameter if flow in line i exceeds limit in positive direction

Table 3-1 (continued)

Variable	Dimension	Type	Definition
XFLNEG(i)	i=1,nlx	R	negative flow limit for line i (MW)
XPENEG(i)	i=1,nlx	R	penalty parameter if flow in line i exceeds limit in negative direction
ILUTIL(i)	i=1,nlx	I	utility to which line i belongs
Common Block /CLINY/			
YRES(i)	i=1,nlx	R	base case value for the resistance of line i ($Ohms/V^2$)
YINDUC(i)	i=1,nlx	R	base case value for the inductance of line i ($Ohms/V^2$)
JCONST(i)	i=1,nlx	I	base case value for the constraint on line i: 0 - no constraint 1 - soft constraint 2 - hard constraint
YFLPOS(i)	i=1,nlx	R	base case value for the positive flow limit for line i (MW)
YPEPOS(i)	i=1,nlx	R	base case value for the penalty parameter if flow in line i exceeds limit in the positive direction
YFLNEG(i)	i=1,nlx	R	base case value for the negative flow limit for line i (MW)
YPENEG(i)	i=1,nlx	R	base case value for the penalty parameter if flow in line i exceeds limit in negative direction
Common Block /CBUS/			
XDEMD(i)	i=1,nbx	R	demand at bus i (MW)
IBUTIL(i)	i=1,nbx	I	utility to which bus i belongs

Table 3-1 (continued)

Variable	Dimension	Type	Definition
Common Block /CSUPLY/			
XGENCP(i,j)	i=1,nbx j=1,nptx	R	generation level at point j on the supply curve for bus i (MW)
XGENCO(i,j)	i=1,nbx j=1,nptx	R	marginal cost at point j on the supply curve for bus i ($/MWh)
IGENPT(i)	i=1,nbx	I	number of points on the supply curve for bus i
YGENCP(i,j)	i=1,nbx j=1,nptx	R	base case value for the generation level at point j on the supply curve for bus i (MW)
YGENCO(i,j)	i=1,nbx j=1,nptx	R	base case value for the marginal cost at point j on the supply curve for bus i ($/MWh)
XUNEGM(i)	i=1,nbx	R	cost of unserved energy for bus i ($/MWh)
XUNESL(i)	i=1,nbx	R	slope of unserved energy cost for bus i ($/MWh/MWh)
XEXDGM(i)	i=1,nbx	R	cost of excess demand for bus i ($/MWh)
XEXDSL(i)	i=1,nbx	R	slope of the cost of excess demand for bus i ($/MWh/MWh)
Common Block /CPOOL/			
IUPOOL(i)	i=1,nux	I	pool to which utility i belongs
IUP(i)	i=1,nux	I	user assigned number of the utility that is independently dispatched
IDISNU(i)	i=1,nux	I	program assigned number to the independently dispatch utility i IUP(IDISNU(i)) = i
NPRANK(i)	i=1,npx	I	number of utilities in pool i
Common Block /CNETEI/			
XNETEI(i)	i=1,nupx	R	net energy interchange for independently dispatched entity i (MW)

Table 3-1 (continued)

Variable	Dimension	Type	Definition
YNETEI(i)	i=1,nupx	R	base case value for the net energy interchange for independently dispatched entity i (MW)
Common Block /SCENAR/			
KSCEN		I	current scenario number
NSCEN		I	number of scenarios
PSCEN(i)	i=1,ns	R	probability of scenario i
Common Block /CSWGB/			
ISWGBN		I	system swing bus, usually same as buying bus
ISWGBS		I	swing bus of the selling party
Common Block /CWHEEL/			
XWHEEL		R	quantity of power wheeled (MW)
JTSELL		I	type of selling party: 1 - bus 2 - utility 3 - pool
NSELL		I	number of the seller
JTBUY		I	type of buying party: 1 - bus 2 - utility 3 - pool
NBUYER		I	number of the buyer
Common Block /CREVU/			
NUTW(i)	i=1,nux	I	utility number of wheeling utility i
ICLASS(i)	i=1,nux	I	obligation class for wheeling utility i: 1 - obligation to serve 2 - no obligation to serve 3 - obligation/no obligation to serve

Table 3-1 (continued)

Variable	Dimension	Type	Definition
IOPTON(i)	i=1,nux	I	rate option for wheeling utility i: 1 - aggregate 2 - disaggregate 3 - decomposed
IREVM(i)	i=1,nux	I	revenue multiplier status: 1 - provided 2 - to be computed
XTREVM(i)	i=1,nux	R	total revenue reconciliation multiplier for utility i
XGREVM(i)	i=1,nux	R	generation revenue reconciliation multiplier for utility i
XNREVM(i)	i=1,nux	R	network revenue reconciliation multiplier for utility i
XTREV(i)	i=1,nux	R	total capital revenues for independently dispatched utility i (K$)
XGREV(i)	i=1,nux	R	generation capital revenues for independently dispatched utility i (K$)
XNREV(i)	i=1,nux	R	network capital revenues for independently dispatched utility i (K$)
NLTW(i)	i=1,nlx	I	line number of each line with the decomposed option
XLREVM(i)	i=1,nlx	R	revenue reconciliation multiplier for line i
XLREV(i)	i=1,nlx	R	line i capital revenues (K$)
Common Block /CFILES/			
IN		I	unit number of input data file
IOUT		I	unit number of error file
IFSCEN		I	unit number of scenario file
IFSPOT		I	unit number of spot price file
ILOOP		I	unit number of counter file

Table 3-1 (continued)

Variable	Dimension	Type	Definition
Common Block /CNAME/			
NAME		A80	title of study
Common Block /COPT/			
JELDM		I	ELDM option: 1 - run ELDM with and without wheeling 2 - do not run ELDM 3 - run ELDM with wheeling only 4 - run ELDM without wheeling only
JRATE		I	rate option: 1 - ideal rates only 2 - ideal and reconciled rates
JSTAT		I	statistics option: 1 - do not run yearly statistics 2 - run yearly statistics
JPOOL		I	pool analysis option: 1 - do not run pool analysis 2 - run pool analysis i.e. split pool wheeling costs among its member utilities
Common Block /CERROR/			
NERROR		I	number of errors
IERN(i)	i=1,nerx	I	type of error i
IFVAR(i)	i=1,nerx	I	first variable in error message of error i
ISVAR(i)	i=1,nerx	I	second variable in error message of error i

Section 4

ECONOMIC LOAD DISPATCH MODULE "ELDM"

The Economic Load Dispatch Module, ELDM, is the largest module of WRATES. It is invoked by WRATES.BAT once for each scenario. It first reads the system topology and the scenario data from the scenario file FSCEN.FIL. Next, for that scenario, it optimally dispatches generation to satisfy the demand at each bus and the net energy interchange of each utility, given the marginal cost curve at each generating bus and the line flow constraints. The generation dispatch is performed with wheeling and/or without wheeling, depending on the value of the input parameter JELDM. Finally, the resulting generation levels, spot prices at each bus, and line flows are written into the spot price file FSPOT.FIL, and in the output file ELDM.OUT. If the system has pools and the user has requested apportioning wheeling rates to pool numbers, one more economic generation dispatch is performed. This dispatch is for a wheeling quantity equal to the specified wheeling quantity + 10 MW. The results of this dispatch are written into the split file FSPLIT file and in the output file ELDM.OUT. The flow chart of ELDM is shown in figure 4-1.

PROGRAM STRUCTURE

The flow of the ELDM module is controlled by the main program stored in ELDM.FOR, which calls a total of 27 subroutines. Four nested loops form the optimization algorithm. The three inner loops are run on a utility by utility basis. The innermost one is called the "spot price and generation loop" and it computes the generation levels that correspond to the spot prices at each bus of that utility. The second, called the "gamma loop," varies the value of gamma until the energy balance is met for that utility. The third one is called the "mu loop" and ensures that the flow on the line with a hard constraint, if any, does not exceed the flow limit set in the input data. Finally, the outermost loop, or "system loop" is performed as many times as necessary until the generation levels and line flows within each utility are consistent with the rest of the system.

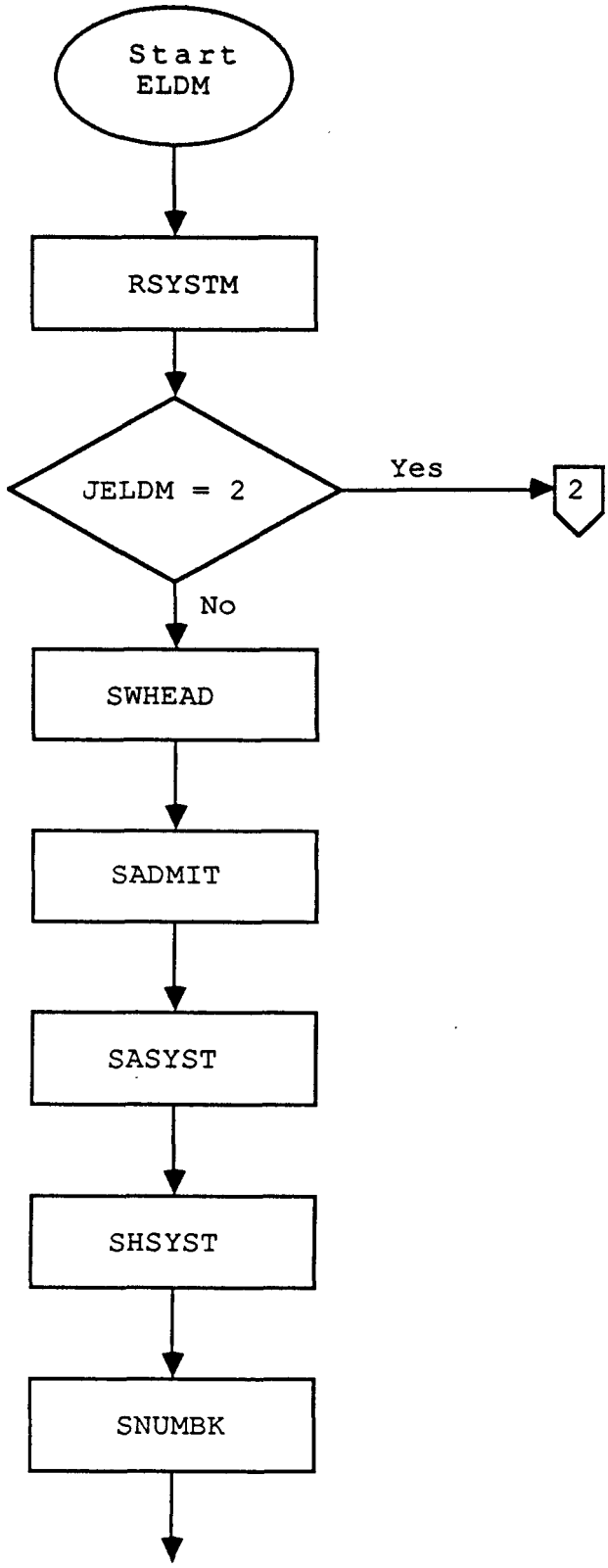

Figure 4-1. Flow Chart for the ELDM Module

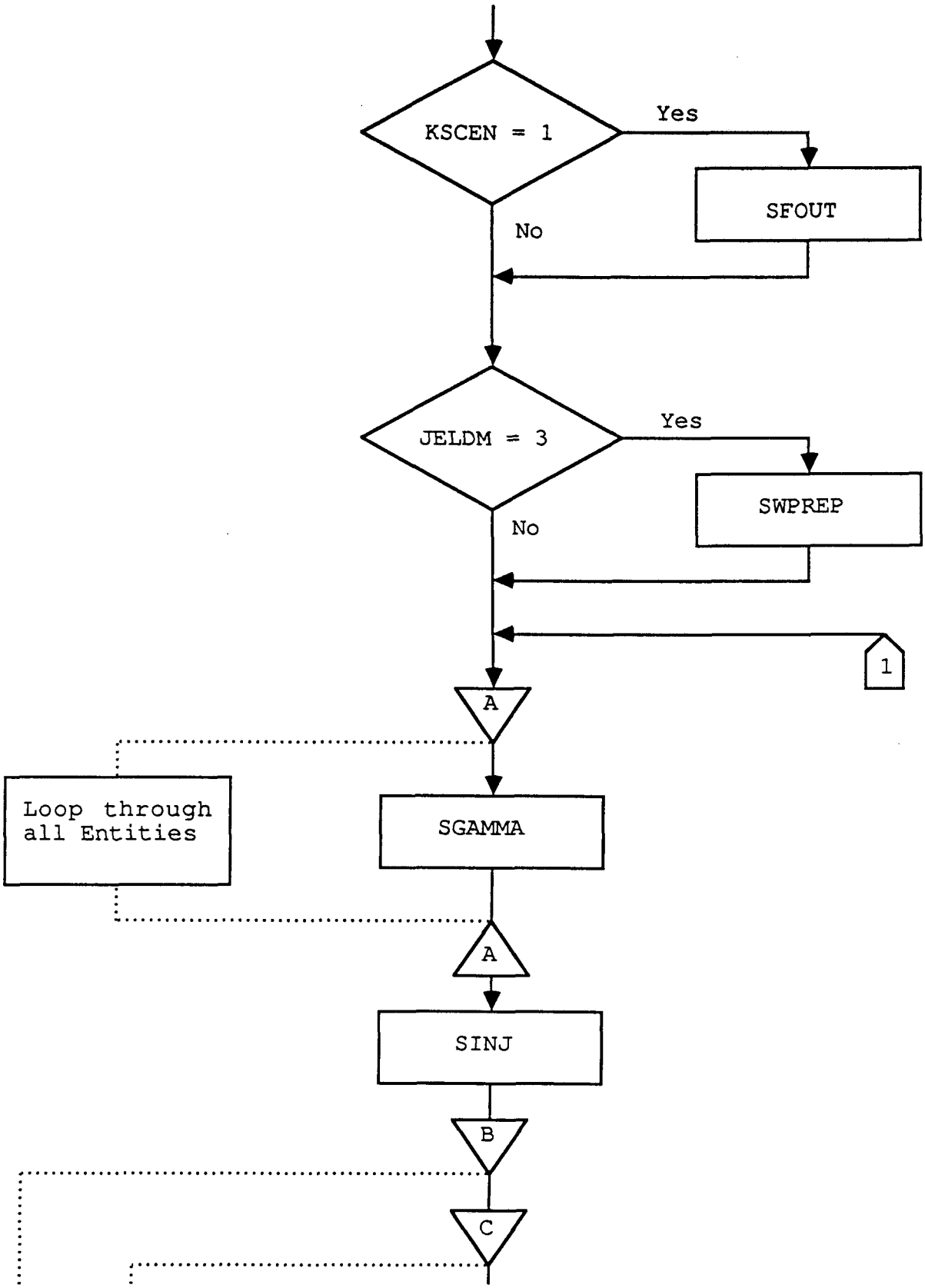

Figure 4-1 (Continued). Flow Chart for the ELDM Module

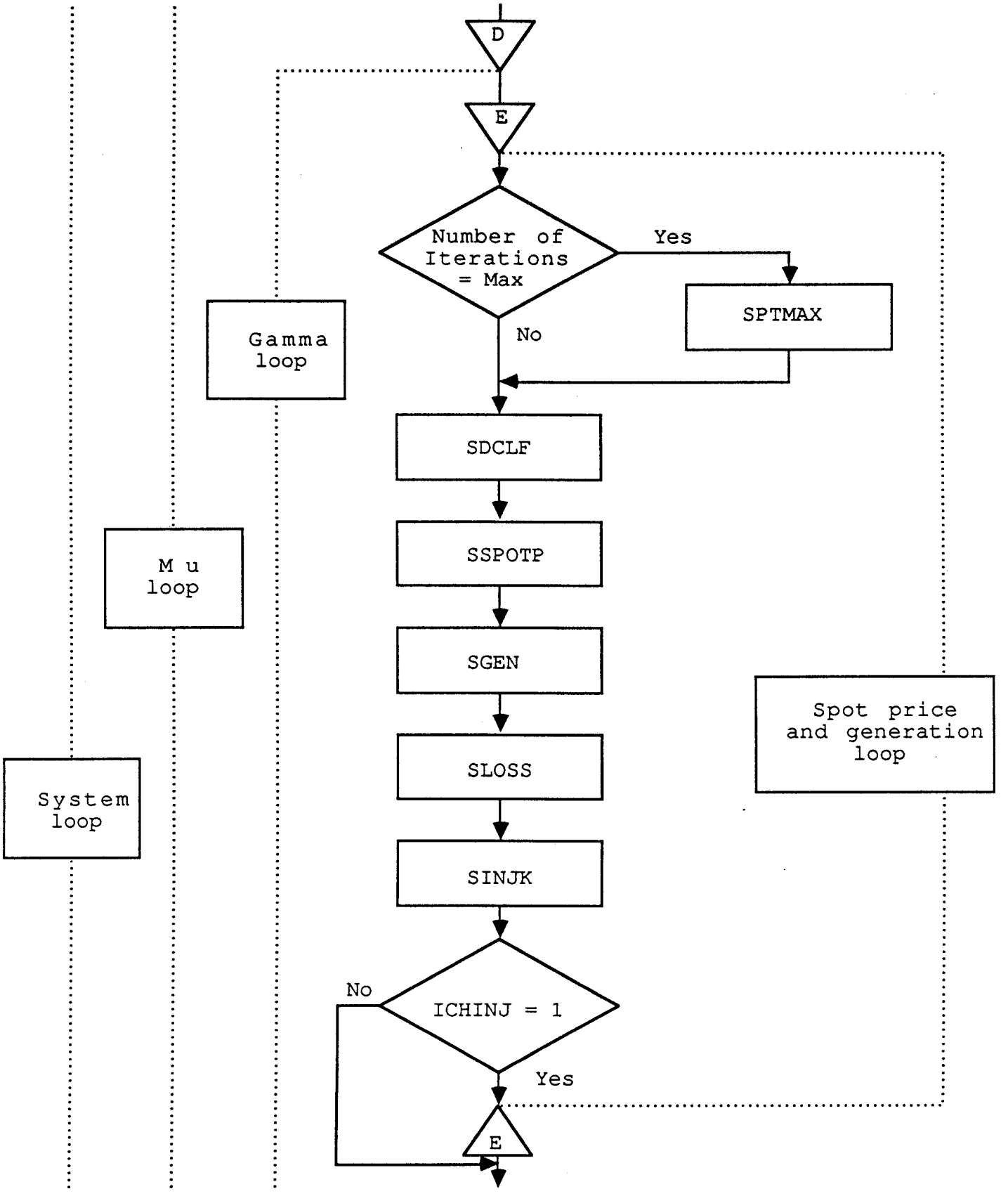

Figure 4-1 (Continued). Flow Chart for the ELDM Module

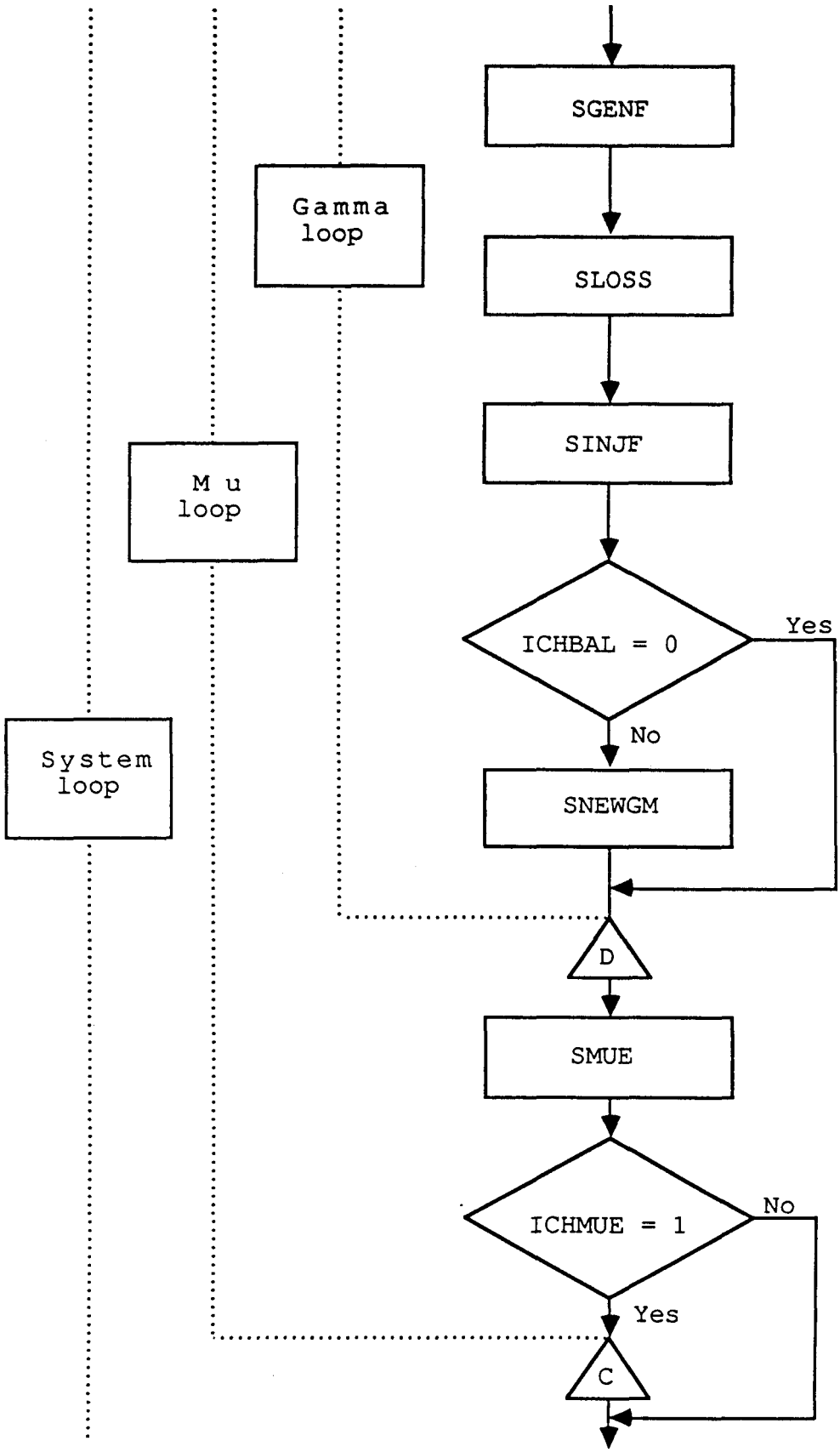

Figure 4-1 (Continued). Flow Chart for the ELDM Module

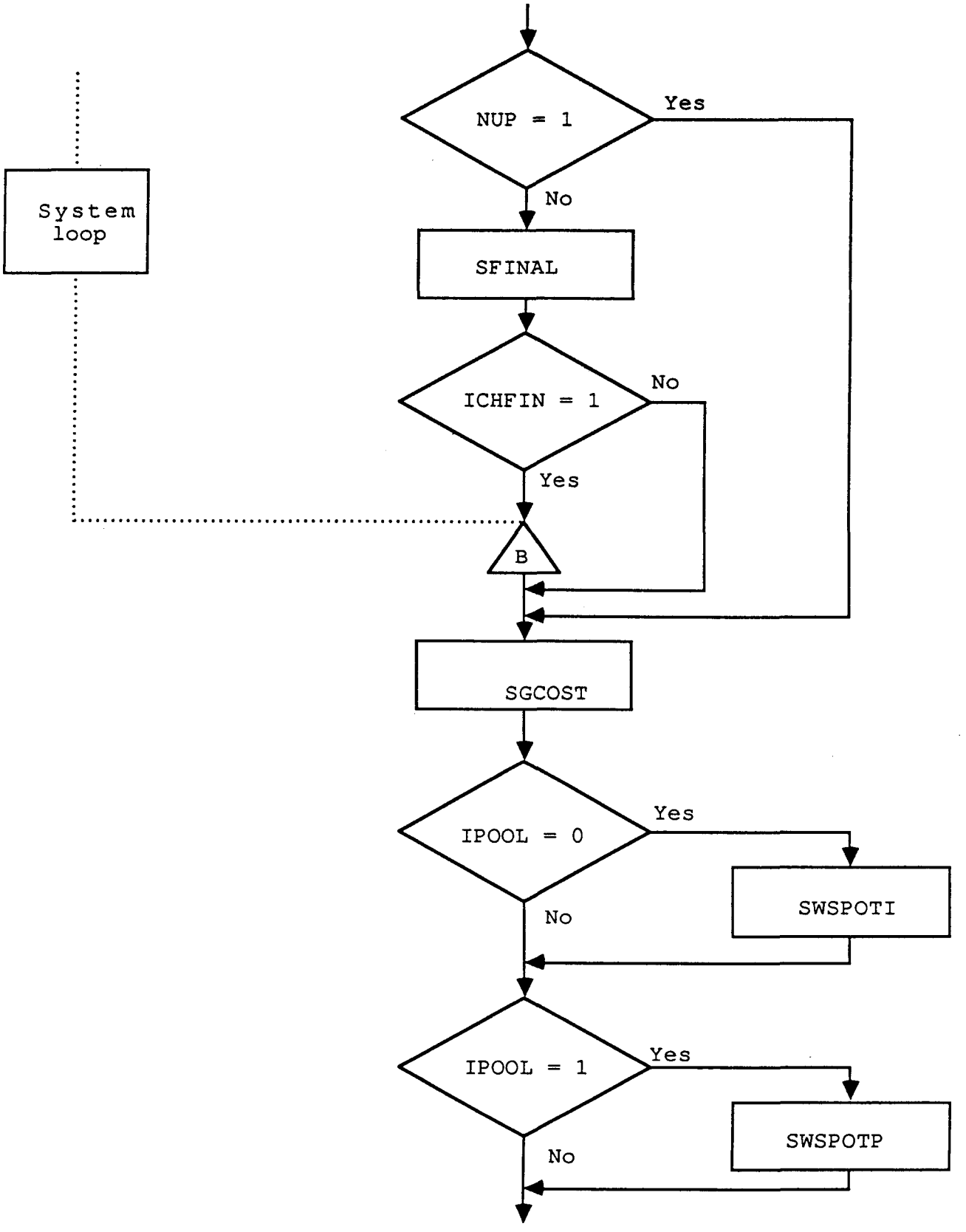

Figure 4-1 (Continued). Flow Chart for the ELDM Module

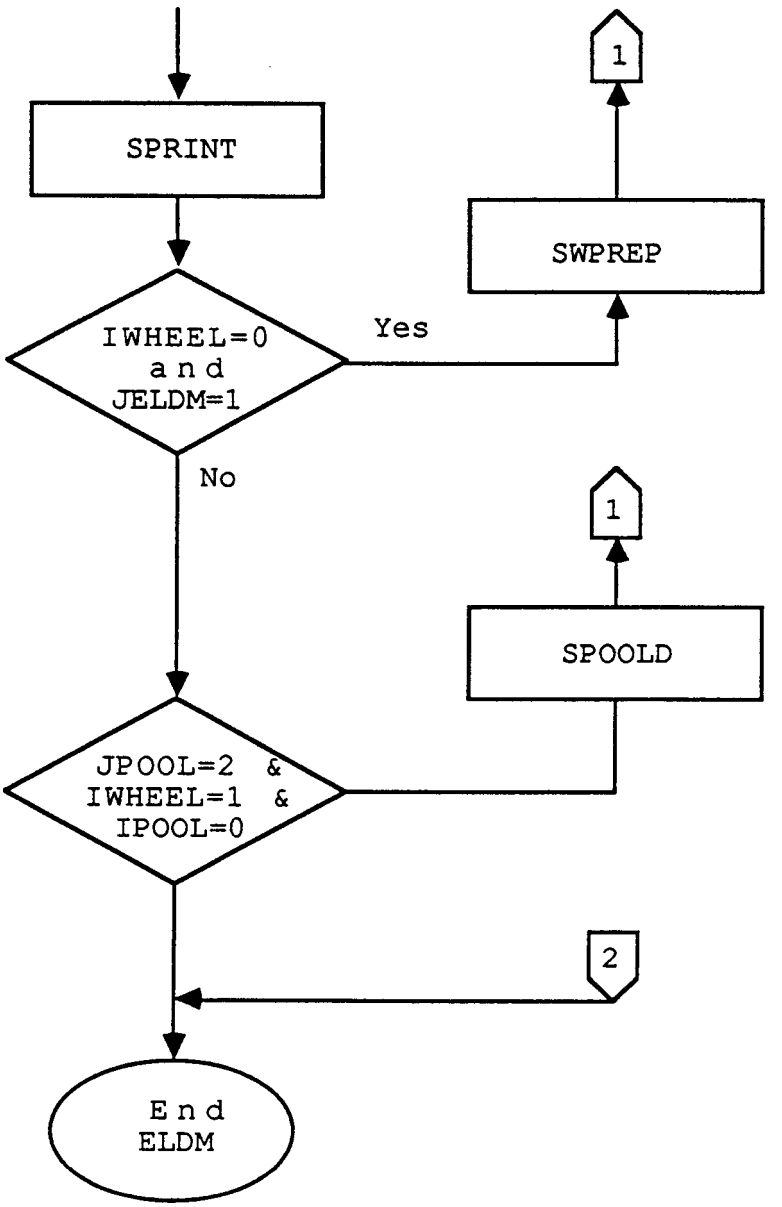

Figure 4-1 (Continued). Flow Chart for the ELDM Module

The following is a description of the function of the ELDM subroutines in the same order as they are called by the main routine.

Subroutine RSYSTM

This subroutine reads the system record (record 1), the pool record (record 2) and the record corresponding to the scenario that ELDM is simulating from the scenario file FSCEN.FIL.

Subroutine SWHEAD

This subroutine writes a heading in both the warning file "WARNING.FIL" and the output file "ELDM.OUT" of the economic load dispatch module.

Subroutine SASYST

This subroutine creates the network incidence matrix A. It also rearranges the buses such that the swing bus is the last one, and all the buses following the swing bus have their rank reduced by one. The array IPROGB holds the new bus numbers while the array IUSERB holds the user assigned bus numbers; and

```
IUSERB(IPROGB(IBUS))  =   IBUS
```

Subroutine SHSYST

This subroutine computes the transfer admittance matrix H which has one row for each line and one column for each bus excluding the system swing bus. In this routine, there is a matrix inversion algorithm, and it is based on the Gauss elimination method.

Subroutine SNUMBK

This subroutine assigns to each line and to each bus a number reflecting their order in their respective independently dispatched entities. The arrays ISYSTL and ISYSTB hold the network numbers for lines and buses respectively. For example, if lines 7, 18 and 21 are the only lines belonging to pool 2; then

```
ISYSTL(1,2)    =  7
ISYSTL (2,2)   = 18
ISYSTL (3,2)   = 21
```

Similarly for buses, if POOL 1 has only two buses: 3 and 9; then

```
ISYSTB (1,1)   =  3
ISYSTB (2,1)   =  9
```

Subroutine SFOUT

This subroutine is called only when ELDM is simulating the 1st scenario. It writes the topology record (record 4) into the direct access spot price file "FSPOT.FIL".

Subroutine SWPREP

This subroutine is called only when ELDM is dispatching the system with the wheeling transaction. It basically simulates the effect of that transaction. If the seller or buyer is a bus, it modifies the demand at that bus as well as the net energy interchange of the independently dispatched entity (utility or pool) to which that bus belongs. If the seller or buyer is a utility, it modifies the net energy interchange of that utility if it is independently dispatched, or it modifies the net energy interchange of the pool to which that utility belongs. Finally if the seller or buyer is a pool, it modifies the net energy interchange of that pool.

Subroutine SADMIT

This subroutine computes the admittances of all the lines of the network. It also computes for each line the variable XLR, which is the parameter such that the product of XLR and the square of the line flow is equal to the losses on the line.

Subroutine SGAMMA

This subroutine is called once for each independently dispatched entity. It computes an initial guess for gamma. This initial guess is selected such that with this value of gamma the total generation within the utility (or pool) will equal the total demand of that entity plus its net energy interchange.

Subroutine SINJ

This subroutine computes the injection at each bus of the system. Injection at a bus is defined as the value of generation at that bus minus the value of demand at that bus.

Subroutine SPTMAX

This subroutine is called only if, after the maximum allowed number of iterations (NITSMX), the innermost loop of ELDM did not produce a set of consistent spot prices and generation levels at each bus for a specific value of gamma. SPTMAX then computes a new value of gamma depending on whether previous values of gamma have resulted in converging solutions or not.

Subroutine SDCLF

This subroutine computes the flows through all the lines of the network by multiplying the transfer admittance matrix (XHSYST) by the injection vector (XINJ). It also calculates the derivative of the losses represented at each bus of the system with respect to a change in injection at each bus of the utility that is being dispatched.

Subroutine SSPOTP

This subroutine computes the spot price at each bus of the independently dispatched entity that is being dispatched. The network quality of supply terms are computed within this subroutine.

Subroutine SGEN

This subroutine computes an estimate of the generation levels at each generating bus of the utility being dispatched given the spot price at each bus. This estimate is not the exact value corresponding to the spot price on the marginal cost curve; rather, it is a weighted average of the previous value and the value corresponding to the latest spot price.

Subroutine SLOSS

This subroutine computes the losses of each line of the entity being dispatched. It also associates half of the losses of each line with the bus at the beginning of that line and the other half with the bus at the end of that line. This association is the direct result of the expansion of the DC load flow equations, and is necessary to obtain accurate values for the spot prices.

Subroutine SINJK

Subroutine SINJK computes the injection at each bus of the utility being considered. Here, the injection at a bus is defined as the generation at that bus minus the demand at that bus, minus the losses associated with that bus. The losses are equal to half of the losses for all the lines connected to that bus. The newly calculated values of the injections are then compared at each bus to the previous values. If the injections at any bus are different by an amount larger than the tolerance allowed, the flag ICHINJ is set to 1, to indicate that at least one more iteration of the "spot price and generation loop" is needed.

Subroutine SGENF

This subroutine is called right after the "spot price and generation loop" and yields a consistent set of generations and spot prices at each bus of the independently dispatch entity. It computes the final values of generation based on the marginal cost curve at each bus; those values correspond to the last calculated spot price.

Subroutine SINJF

This subroutine is called after SGENF and SLOSS yield the final values of the generation and losses, respectively. It calculates the injection at each bus of the utility being considered, in the same way as SINJK.

Subroutine SENGBL

This subroutine checks the energy balance for the independently dispatched entity utility being considered. The energy balance equation is:

Generation - Demand - Losses - Net Energy Interchange = delta

If delta is not within the tolerance (TOLENG) specified by the user in BLKDATA.FOR, the flag ICHBAL is set to 1, indicating that the gamma loop needs at least one more iteration; otherwise the gamma loop is exited.

Subroutine SNEWGM

This subroutine is only called when the energy balance is not achieved. Its function is to estimate a new value of gamma, the marginal cost of generation for the independently dispatched entity being considered. This new value is selected between a lower bound (FGAM) and an upper bound (SGAM) derived from previous iterations.

Subroutine SMUE

Each independently dispatched entity can have up to one line with a hard constraint on its flow. This subroutine checks the flow on that line. If it exceeds the flow limit, SMUE estimates a value for the Lagrange multiplier "mu" that will reduce the flow on the line to its limit. In the first iteration, "mu" is set equal to the value of gamma, the marginal generating cost for the utility to which the line belongs. In subsequent iterations, a lower bound and an upper bound on the value of "mu" are derived and updated with each iteration until the final value of mu is computed. This value is such that the line flow on the line with a hard constraint is equal to its limit within a tolerance TOLINJ set by the user in BLKDATA.FOR.

Subroutine SFINAL

SFINAL is part of the outermost or system loop. Its function is to ensure that all the bus generations derived for each independently dispatched entity form a feasible and consistent solution for the whole system. Subroutine SFINAL checks if the new values of the injections at each bus of the system are equal to the values derived in the previous iteration within the tolerance (TOLINJ) set in BLKDATA.FOR. If the injection at any one bus is different than its previous value by an amount larger than the tolerance, the flag ICHFIN is set to 1 to indicate that at least one more iteration in the system loop is required. Otherwise the system loop is exited.

Subroutine SGCOST

This subroutine computes the cost of generation at each generating bus of the system by calculating the areas under their respective marginal cost curves. It also computes the total generation for each utility by summing the costs at each bus of the utility and for each pool by summing the costs at each bus of the pool.

Subroutine SWSPOTT

This subroutine writes the spot price record for the scenario being simulated in the spot price file FSPOT.FIL (see Section 7).

Subroutine SWSPOTP

This subroutine writes the generations costs for all pools and all utilities in the split file "FSPLIT.FIL," if the wheeling rate for pools is to be apportioned among member utilities.

Subroutine SPRINT

This subroutine writes the results of the Economic Load Dispatch Module in the ASCII file ELDM.OUT (see Section 7).

Subroutine SPOOLD

This subroutine is called when ELDM is dispatching the system for a wheeling quantity equal to the specified quantity + 10 MW. It simulates the effect off that transaction in the same fashion as SWPREP.

COMMON BLOCKS AND VARIABLE DICTIONARY

This section describes the common variables used in the ELDM module. Two files define and initialize the variables in this module. The first, COMMON.FOR, defines the common blocks. The code in this file is included in all subroutines of the ELDM module. The second, BLKDATA.FOR, is a subprogram that initializes all the variables defined in the common blocks. Table 4-1 is a listing of the common variables of ELDM. The array variables are dimensioned by parameters that are set in a parameter statement in COMMON.FOR. These parameters are:

- o nlx maximum number of lines
- o nbx maximum number of buses
- o nux manximum number of utilities
- o npx maximum number of pools
- o nptx maximum number of points on supply curve
- o nupx maximum number of independently dispatched entities

Table 4-1

COMMON VARIABLES IN THE ELDM MODULE

Variable	Dimension	Type	Definition
Common Block /CSYS/			
KUP		I	current independently dispatched entity
NBUS		I	total number of buses
NLINE		I	total number of lines
NUTIL		I	total number of utilities
NPOOL		I	total number of pools
NIUTIL		I	total number of independently dispatched utilities
NUP		I	total number of independently dispatched entities = NPOOL + NIUTIL
NBUSUP(i)	i=1,nupx	I	number of buses in each independently dispatched entity
NLINEUP(i)	i=1,nupx	I	number of lines in each independently dispatched entity
Common Block /CPOOL/			
IUPOOL(i)	i=1,nux	I	pool to which utility i belongs
IUP(i)	i=1,nux	I	user assigned number of the independently dispatched utility i
IDISNU(i)	i=1,nux	I	program assigned number to the independently dispatched utility i; IUP(IDISNU(i)) = i
ILPOOL(i)	i=1,nlx	I	pool to which line i belongs
IBPOOL(i)	i=1,nbx	I	pool to which bus i belongs.
NPRANK(i)	i=1,npx	I	number of utilities in pool i
IPUTIL(i,j)	i=1,nux j=1,npx	I	number of the i^{th} utility in the j^{th} pool

Table 4-1 (continued)

Variable	Dimension	Type	Definition
Common Block /CLINE/			
XRES(i)	i=1,nlx	R	resistance of line i (Ohms/V^2)
XINDUC(i)	i=1,nlx	R	inductance of line i (Ohms/V^2)
XADMIT(i)	i=1,nlx	R	admittance of line i
XLR(i)	i=1,nlx	R	parameter such that line losses = XLR x (line flow squared)
IBEGB(i)	i=1,nlx	I	number of the beginning bus of line i
IENDB(i)	i=1,nlx	I	number of the end bus of line i
XFLPOS(i)	i=1,nlx	R	positive flow limit for line i (MW)
XPEPOS(i)	i=1,nlx	R	penalty parameter if flow in line i exceeds limit in positive direction
XFLNEG(i)	i=1,nlx	R	negative flow limit for line i (MW)
XPENEG(i)	i=1,nlx	R	penalty parameter if flow in line i exceeds limit in negative direction
ILUTIL(i)	i=1,nlx	I	utility to which line i belongs
ICONST(i)	i=1,nlx	I	type of constraint for line i: 0 - no constraint 1 - soft constraint 2 - hard constraint
Common Block /CBUS/			
XDEMD(i)	i=1,nbx	R	demand at bus i (MW)
XGENB(i)	i=1,nbx	R	generation at bus i (MW)
IBUTIL(i)	i=1,nbx	I	utility to which bus i belongs
XSPOTP(i)	i=1,nbx	R	spot price at bus i ($/MWh)
XLSPOT(i,j)	i=1,nlx j=1,nbx	R	component of the spot price at bus j, associated with line i

Table 4-1 (continued)

Variable	Dimension	Type	Definition
Common Block /CMSYST/			
IASYST(i,j)	i=1,nlx j=1,nbx	I	incidence matrix
XHSYST(i,j)	i=1,nlx j=1,nbx	R	transfer admittance matrix
XDDEV(i,j)	i=1,nbx j=1,nbx	R	derivative of losses associated with bus i due to changes in injection at bus j
Common Block /CCOST/			
XBCOST(i)	i=1,nbx	R	production cost at bus i ($)
XUGCB(i)	i=1,nux	R	production cost for utility i, before wheeling ($)
XUGCA(i)	i=1,nux	R	production cost for utility i, including the effect of wheeling ($)
XPGCB(i)	i=1,npx	R	production cost for pool i, before wheeling ($)
XPGCA(i)	i=1,npx	R	production cost for pool i including the effect of wheeling ($)
Common Block /CSUPLY/			
XGENCP(i,j)	i=1,nbx j=1,nptx	R	generation level at point j of the supply curve for bus i (MW)
XGENCO(i,j)	i=1,nbx j=1,nptx	R	marginal cost at point j on supply curve for bus i ($/MWh)
IGENPT(i)	i=1,nbx	I	number of points on the supply curve for bus i
XUNEGM(i)	i=1,nbx	R	cost of unserved energy for bus i ($/MWh)
XUNESL(i)	i=1,nbx	R	slope of unserved energy cost for bus i ($/MWh/MWh)

Table 4-1 (continued)

Variable	Dimension	Type	Definition
XEXDGM(i)	i=1,nbx	R	cost of excess demand for bus i ($/MWh)
XEXDSL(i)	i=1,nbx	R	slope of the cost of excess demand for bus i ($/MWh/MWh)
DUNE		R	amount of energy used for sloping the inserted energy section of the marginal cost curves
Common Block /CNETEI/			
XNETEI(i)	i=1,nupx	R	net energy interchange for the independently dispatched entity i (MW)
Common Block /CSWGB/			
ISWGBN		I	system swing bus
ISWGBS		I	swing bus of the selling utility or seller bus
Common Block /CLINEF/			
XLINF(i)	i=1,nlx	R	line flow (MW)
XPENF(i)	i=1,nlx	R	derivative of the penalty function due to a flow on line i exceeding its flow limit ($/MW)
XPENFC(i)	i=1,nlx	R	penalty function due to a flow on line i exceeding its flow limit ($)
Common Block /CWHEEL/			
IWHEEL		I	wheeling flag: 0 - dispatch without wheeling 1- dispatch with wheeling
XWHEEL		R	power wheeled (MW)
JTSELL		I	type of selling party: 1 - bus 2 - utility 3 - pool
NSELL		I	number of the selling party

Table 4-1 (continued)

Variable	Dimension	Type	Definition
JTBUY		I	type of buying party: 1 - bus 2 - utility 3 - pool
NBUYER		I	number of the buying party
Common Block /COMPUT/			
IUSERB(i)	i=1,nbx	I	user assigned number of the bus with an internally assigned number equal to i
IPROGB(i)	i=1,nbx	I	internally assigned number for bus i
ISYSTL(i,j)	i=1,nlx j=1,nupx	I	line number of the i^{th} line in the j^{th} independently dispatched entity
ISYSTB(i,j)	i=1,nbx j=1,nupx	I	bus number of the i^{th} bus in the the j^{th} independently dispatched entity
Common Block /CGAMMA/			
XGAMMA(i)	i=1,nupx	R	marginal cost of generation for independently dispatched entity i ($/MWh)
NITGAM		I	iteration number in the gamma loop
NITGMX		I	maximum number of iterations in the gamma loop
TOLENG		R	tolerance for the energy balance (MW)
FGAM(i)	i=1,nupx	R	lower bound of the interval containing the final value of gamma for entity i
SGAM(i)	i=1,nupx	R	upper bound of the interval containing the final value of gamma for entity i

Table 4-1 (continued)

Variable	Dimension	Type	Definition
FTGEN(i)	i=1,nupx	R	total generation in entity i corresponding to gamma = FGAM(i)
STGEN(i)	i=1,nupx	R	total generation in entity i corresponding to gamma = SGAM(i)
Common Block /CINJ/			
XINJ(i)	i=1,nbx	R	injection at bus i (MW)
XINJP(i)	i=1,nbx	R	previous value of injection at bus i used in subroutine SINJK (MW)
XINJPP(i)	i=1,nbx	R	previous value of injection at bus i used in subroutine SFINAL (MW)
NITSPT		I	iteration number in the innermost loop of ELDM (spot price and generation loop)
NITSMX		I	maximum number of iterations allowed
KITSPT		I	number of times subroutine SPTMAX is called within the current gamma loop
NITFIN		I	iteration number in the system loop
NITFMX		I	maximum number of iterations allowed for the system loop
TOLINJ		R	tolerance for the injections (MW)
Common Block /CMUE/			
XMUE(i)	i=1,nupx	R	Lagrange multiplier for the hard line constraint in independently dispatched entity i ($/MWh)
NITMUE		I	iteration number in the mu loop
NITMMX		I	maximum number of iterations allowed for the mu loop
TOLLF		R	tolerance for the line flow on the lines with hard constraints (MW)

Table 4-1 (continued)

Variable	Dimension	Type	Definition
ILMUE(i)	i=1,nupx	I	number of the line with hard constraint in independently dispatched entity i
FVMUE(i)	i=1,nupx	R	lower bound of the mu interval that contains the final value of mu for entity i
SVMUE(i)	i=1,nupx	R	upper bound of the mu interval that contains the final value of mu for entity i
FLINF(i)	i=1,nupx	R	flow on the line with a hard constraint in entity corresponding to FVMUE (i)
SLINF(i)	i=1,nupx	R	flow on the line with a hard constraint in entity corresponding to SVMUE (i)
Common Block /CNAME/			
NAME		A80	title of study
Common Block /CPREV/			
ENGDLT		R	difference between generation and (demand + losses + net energy interchange)
ALPHAG(i)	i=1,nbx	R	acceleration factor for the iterations within SGEN subroutine
JSIGN(i)	i=1,nbx	I	sign of the difference between generation values at bus i given by 2 subsequent iterations in subroutines SGEN
Common Block /CLOSS/			
XLOSS(i)	i=1,nupx	R	losses in independently dispatched entity i (MW)
DLOSS(i)	i=1,nbx	R	losses associated with bus i (MW)
XLOSSL(i)	i=1,nlx	R	losses of line i (MW)

Table 4-1 (continued)

Variable	Dimension	Type	Definition
Common Block /CFILES/			
IFSCEN		I	unit number of scenario file FSCEN.FIL
IFLOOP		I	unit number of counter file FLOOPONE.OUT
IOUT		I	unit number of printout file ELDM.OUT
IFSPOT		I	unit number of spot price file FSPOT.FIL
IFSPLIT		I	unit number of the split file FSPLIT.FIL
IWARN		I	unit number of warning file WARNING.FIL
Common Block /COPTON/			
JELDM		I	ELDM option: 1 - run ELDM with and without wheeling 2 - do not run ELDM 3 - run ELDM with wheeling only 4 - run ELDM without wheeling only
JPOOL		I	pool option: 1 - do not split the wheeling costs among utilities affiliated with pools 2 - split wheeling costs among utilities affiliated with pools
IPOOL		I	pool flag: 0 - system is dispatched with the wheeling quantity as specified 1 - system is dispatched with the wheeling quantity equal to XWHEEL + 10

Table 4-1 (continued)

Variable	Dimension	Type	Definition
KSCEN		I	current scenario number
PSCEN		R	probability of scenario (%)
NSCEN		I	number of scenarios

Section 5

REVENUE RECONCILIATION MULTIPLIER MODULE "COMPUTM"

Revenue reconciliation multipliers are needed to compute reconciled wheeling rates. In WRATES, the user has the option to specify those multipliers or to input capital revenue requirements and have WRATES compute the multipliers. The user can use either option for each wheeling utility not belonging to a pool. COMPUTM is the module that computes the revenue reconciliation multipliers. It is called by WRATES.BAT after the Economic Load Dispatch module is run for all the scenarios. In COMPUTM, the computation of the revenue reconciliation multipliers is based on yearly pre-wheeling revenues and capital requirements. Three types of revenue reconciliation options are allowed: aggregate, disaggregate and decomposed. In the aggregate case, only one multiplier is computed. In the disaggregate case, two multipliers are computed: one for the generation and one for the network. In the decomposed case one multiplier is computed for the generation and one multiplier is computed for each line of the utility. The flow chart of COMPUTM is shown in Figure 5-1.

PROGRAM STRUCTURE

The flow of the COMPUTM module is controlled by the main program stored in COMPUTM.FOR which calls a total of thirteen (13) subroutines. COMPUTM has two consecutive scenario loops. In each of them, it loops through the scenarios and reads the pre-wheeling spot price and fuel data from the spot price file FSPOT.FIL. The first loop is always executed. It computes the aggregate revenue reconciliation multiplier in the aggregate case, or the generation revenue reconciliation multiplier in the disaggregate and decomposed cases. The second loop is skipped for the aggregate case; it is executed to compute the network revenue reconciliation multiplier in the disaggregate case and the line by line multipliers for the decomposed case. The reason for having two loops is that the generation reconciliation multiplier has to be known before either the network or the line multipliers can be computed. The following is a

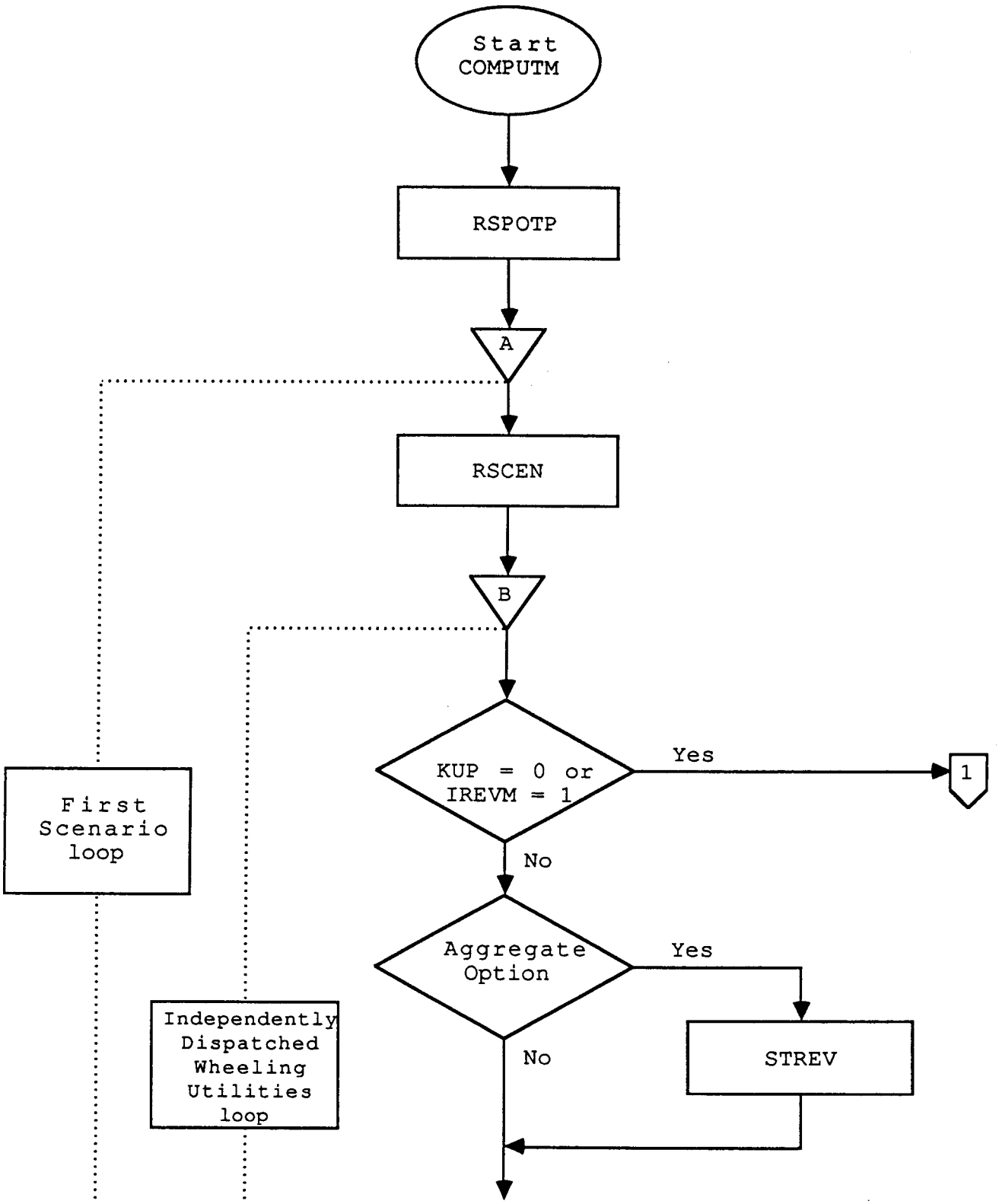

Figure 5-1. Flow Chart for the COMPUTM Module

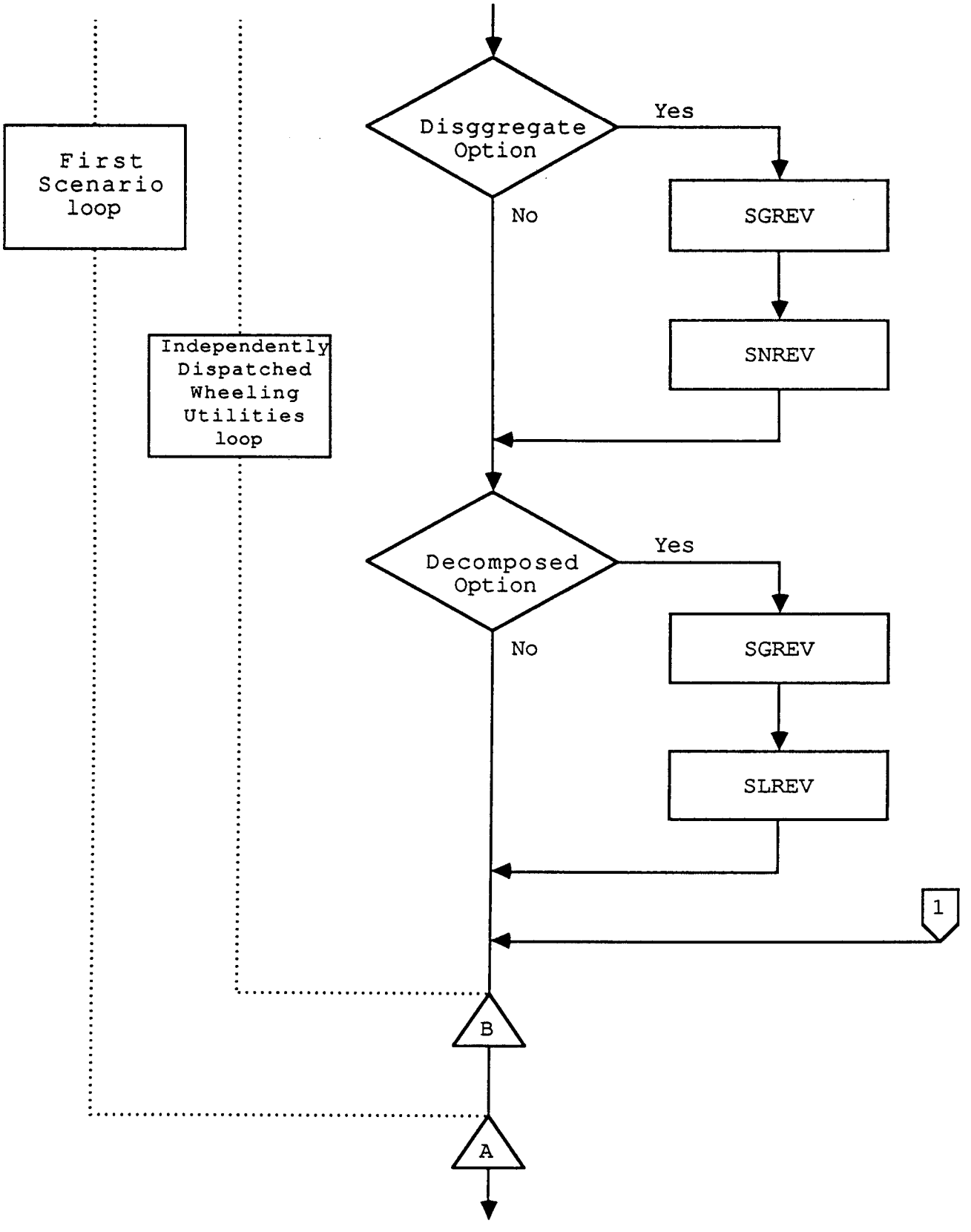

Figure 5-1 (Continued). Flow Chart for the COMPUTM Module

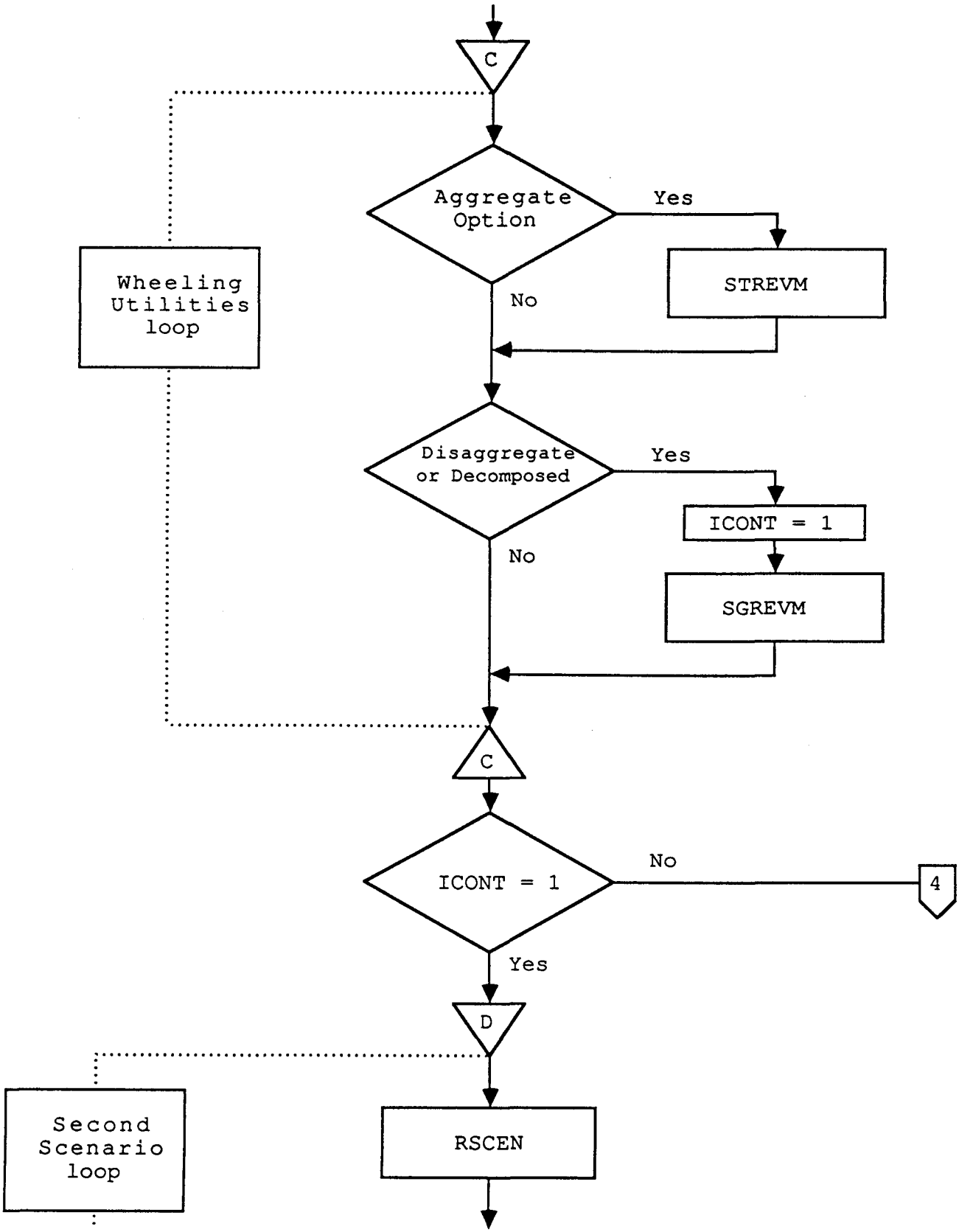

Figure 5-1 (Continued). Flow Chart for the COMPUTM Module

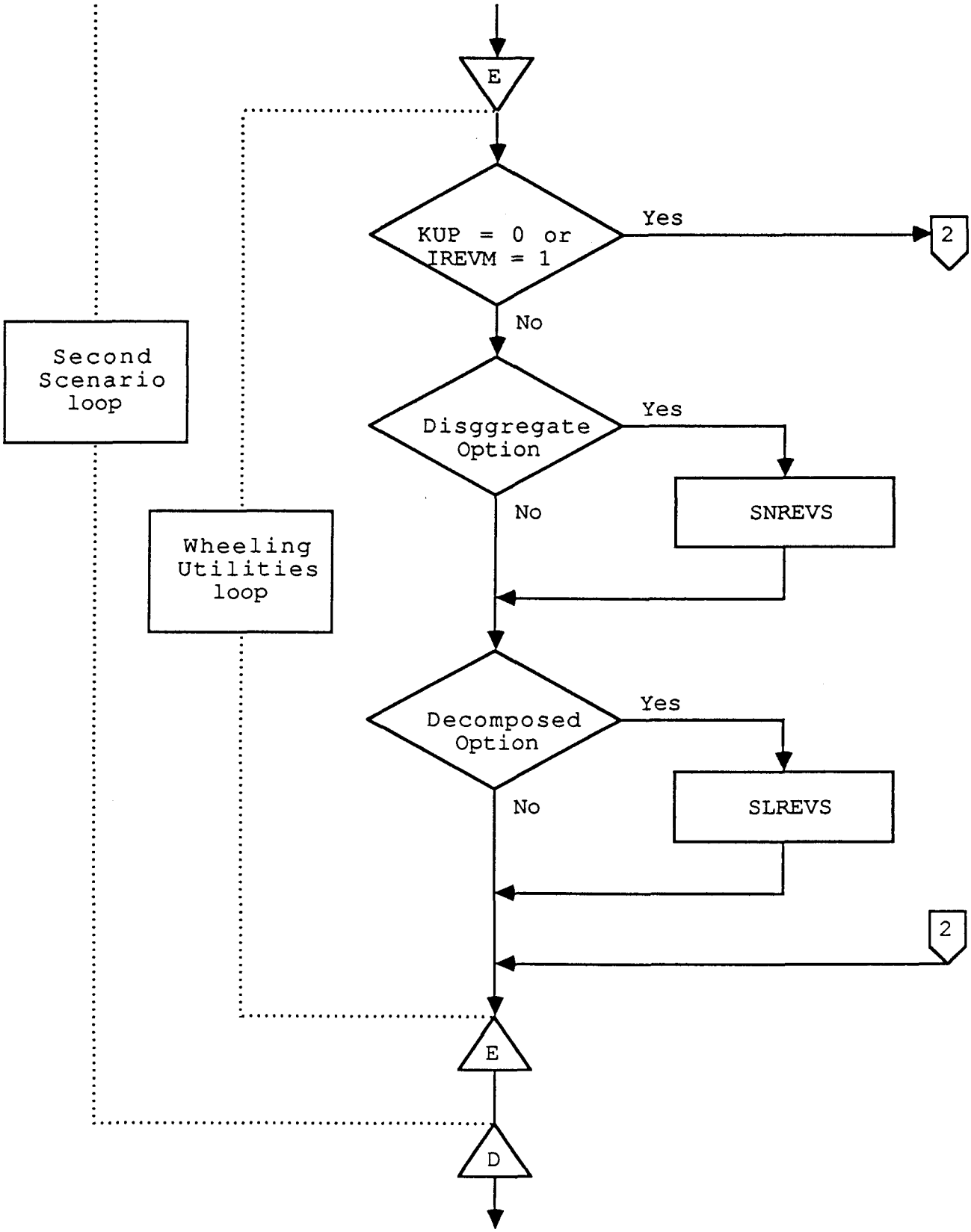

Figure 5-1 (Continued). Flow Chart for the COMPUTM Module

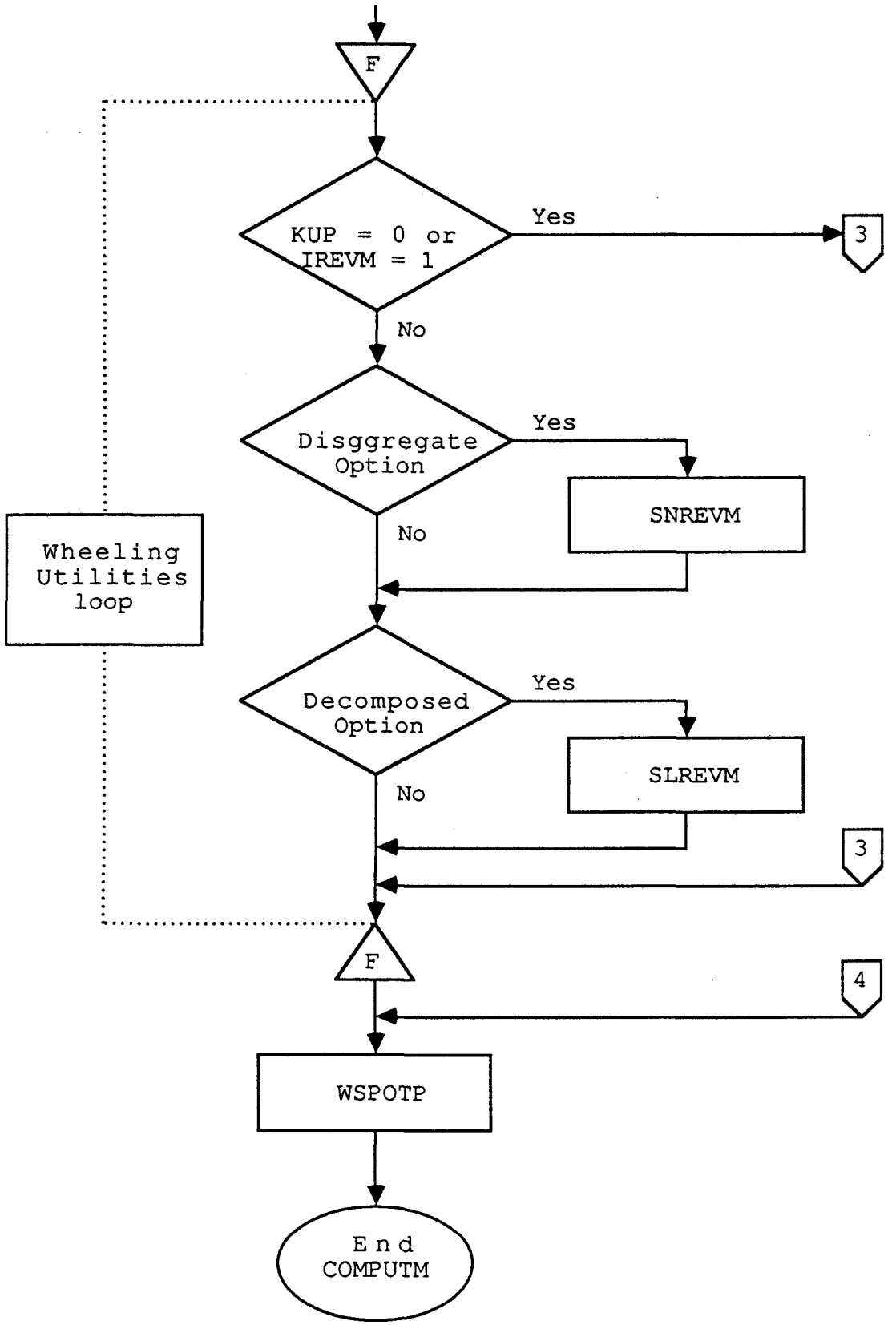

Figure 5-1 (Continued). Flow Chart for the COMPUTM Module

description of the function of all the subroutines of COMPUTM, in the same order as they are called by the main routine in COMPUTM.FOR

Subroutine RSPOTP

RSPOTP is the first subroutine called by COMPUTM. It reads the first four (4) records of the spot price file FSPOT.FIL.

Subroutine RSCEN

This subroutine is called in each scenario loop as many times as there are scenarios. RSCEN reads the record of FSPOT.FIL that holds the pre-wheeling spot price and dispatch information for the scenario under consideration. The number of this record is equal to twice the scenario number plus 3; e.g. if the scenario number is 7, the number of that record is equal to (2 x 7 + 3) = 17.

Subroutine STREV

This subroutine is part of the first "scenario loop". It is called once for each wheeling utility for which the aggregate option is specified. It accumulates the yearly ideal revenues of the utility as well as the yearly fuel cost. It also computes the terms required for the calculation of the aggregate multiplier.

Subroutine SGREV

This subroutine is part of the first "scenario loop." It is called once for each wheeling utility for which either the disaggregate or the decomposed option is specified. It accumulates the yearly ideal generation revenues of the utility as well as the yearly fuel costs.

Subroutine SNREV

This subroutine is also part of the first "scenario loop." It is called once for each wheeling utility for which the disaggregate option is specified. It accumulates the yearly ideal network revenues and one of the terms required for the calculation of the revenue reconciliation network multiplier.

Subroutine SLREV

This subroutine is also part of the first scenario loop. It is called once for each wheeling utility for which the decomposed option is

specified. It accumulates the ideal yearly revenues for each line of the utility; and it also computes some of the required terms for the calculation of the line revenue reconciliation multipliers.

Subroutine STREVM

This subroutine is called after the first "scenario loop" is completed. It is skipped if none of the wheeling utilities has the aggregate option specified. Its function is to compute the aggregate revenue reconciliation multipliers using the yearly terms that were accumulated in subroutine STREV of the first "scenario loop."

Subroutine SGREVM

This subroutine is also called after the first "scenario loop" is completed. It is skipped for the wheeling utilities for which the aggregate option is specified. Its function is to compute the generation revenue reconciliation multiplier using the yearly terms that were accumulated in subroutine SGREV of the first "scenario loop."

Subroutine SNREVS

This subroutine is part of the second "scenario loop." It is called once for each wheeling utility for which the disaggregate option is specified. Its function is to accumulate the yearly cost of losses for the wheeling utility knowing the generation revenue reconciliation multiplier of that utility.

Subroutine SLREVS

This subroutine is also part of the second "scenario loop." It is called once for each wheeling utility for which the decomposed option is specified. Its function is to accumulate the yearly cost of losses for each line of the utility knowing the generation revenue reconciliation multiplier of that utility.

Subroutine SNREVM

This subroutine is called once for each wheeling utility for which the disaggregate option is specified, right after the "second scenario loop" is completed. It computes the network revenue reconciliation multipliers using the yearly terms computed in subroutines SNREV and SNREVS.

Subroutine SLREVM

This subroutine is called once for each wheeling utility for which the decomposed option is specified, right after the second "scenario loop" is completed. It computes the line revenue reconciliation multipliers using the yearly terms computed in subroutines SLREV and SLREVS.

Subroutine WSPOTP

This subroutine is the last routine called by COMPUTM. Its function is to write the calculated revenue reconciliation multipliers in the wheeling record (record 3) of the spot price file.

COMMON BLOCKS AND VARIABLE DICTIONARY

This section describes the common variables used in the COMPUTM module. Two files define and initialize the variables in this module. The first, COMMON.FOR, defines the common blocks. The code in this file is included in all the subroutines of COMPUTM. The second, BLKDATA.FOR, is a subprogram that initializes all the variables defined in the common blocks. Table 5-1 is a listing of the common variables of COMPUTM. The array variables are dimensioned by parameters that are set in a parameter statement in COMMON.FOR. The parameters are:

- o nlx maximum number of lines
- o nbx maximum number of buses
- o nux maximum number of utilities

Table 5-1

COMMON VARIABLES IN THE COMPUTM MODULE

Variable	Dimension	Type	Definition
Common Block /CSYS/			
KUP		I	current independently dispatched utility
KUTIL		I	current utility
NBUS		I	total number of buses
NLINE		I	total number of lines
NUTIL		I	total number of utilities
NPOOL		I	total number of pools
NIUTIL		I	number of independently dispatched utilities
NUP		I	number of independently dispatched entities
NBUSUP(i)	i=1,nupx	I	number of buses in the independently dispatched entity i
NLINEUP(i)	i=1,nupx	I	number of lines in the independently dispatched entity i
Common Block /CPOOL/			
IUPOOL	i=1,nux	I	pool to which utility i belongs
NPRANK(i)	i=1,npx	I	number of utilities affiliated with pool i
IUP(i)	i=1,nux	I	user assigned number of independently dispatched utility i
IDISNU(i)	i=1,nux	I	program assigned number for the independently dispatched utility i IUP(IDISNU(i)) = i
ILPOOL(i)	i=1,nlx	I	number of the independently dispatched entity to which line i belongs

Table 5-1 (continued)

Variable	Dimension	Type	Definition
IBPOOL(i)	i=1,nbx	I	number of the independently dispatched entity to which bus i belongs
Common Block /COPTON/			
ICLASS(i)	i=1,nux	I	obligation class for wheeling utility i: 1 - obligation to serve 2 - no obligation to serve 3 - obligation/no obligation to serve
IOPTON(i)	i=1,nux	I	rate option for utility i: 1 - aggregate 2 - disaggregate 3 - decomposed
IREVM(i)	i=1,nux	I	revenue multiplier status: 1 - provided 2 - to be computed
JRATE		I	rate option: 1 -ideal rates only 2 -ideal and reconciled rates
JREVM		I	0 - all IREVM = 1 1 - at least one IREVM = 2
Common Block /CPRICE/			
XNETEI(i)	i=1,nupx	R	net energy interchange for independently dispatched entity i (MW)
XGAMMA(i)	i=1,nupx	R	marginal generation cost in independently dispatched entity i ($/MWh)
XSPOTP(i)	i=1,nbx	R	spot price at bus i ($/MWh)
XNSPOT(i)	i=1,nbx	R	network component of spot price at bus i ($/MWh)
XLSPOT(i,j)	i=1,nlx j=1,nbx	R	line component of spot price at bus j for line i ($/MWh)
XTFUEL(i)	i=1,nux	R	hourly fuel cost for independently dispatched entity i ($)

Table 5-1 (continued)

Variable	Dimension	Type	Definition
XCFUEL(i)	i=1,nux	R	yearly fuel cost for independently dispatched entity i ($)
XLOSS(i)	i=1,nupx	R	losses within independently dispatched entity i (MW)
XLOSSL(i)	i=1,nlx	R	losses of line i (MW)
XPENFC(i)	i=1,nlx	R	penalty function due to a flow on line i exceeding its flow limit ($)
Common Block/CRECON/			
XTREVM(i)	i=1,nux	R	total revenue reconciliation multiplier for utility i
XGREVM(i)	i=1,nux	R	generation revenue reconciliation multiplier for utility i
XNREVM(i)	i=1,nux	R	network revenue reconciliation multiplier for utility i
XLREVM(i)	i=1,nlx	R	line revenue reconciliation multiplier for line i
XTREV(i)	i=1,nux	R	total capital revenues for utility i (K$)
XGREV(i)	i=1,nux	R	generation capital revenues for utility i (K$)
XNREV(i)	i=1,nux	R	network capital revenues for utility i (K$)
XLREV(i)	i=1,nlx	R	line capital revenues for line i (K$)
XIR(i)	i=1,nux	R	ideal revenues for utility i
XIRA(i)	i=1,nux	R	array that holds intermediate results for the calculation of the aggregate revenue reconciliation multiplier for utility i
XGIR(i)	i=1,nux	R	ideal generation revenues for utility i
XNIR(i)	i=1,nux	R	ideal network revenues for utility i

Table 5-1 (continued)

Variable	Dimension	Type	Definition
XNIRA(i)	i=1,nux	R	array that holds intermediate results for the calculation of the network revenue reconciliation multiplier
XLPR(i)	i=1,nux	R	a second array that holds intermediate results for the calculation of the network revenue reconciliation multiplier
XLIR(i)	i=1,nlx	R	ideal line revenues
XLIRA(i)	i=1,nlx	R	array that holds intermediate results for the calculation of the line revenue reconciliation multiplier
XLOR(i)	i=1,nlx	R	a second array that holds intermediate results for the calculation of the line revenue reconciliation multiplier
Common Block /CBUS/			
XDEMD(i)	i=1,nbx	R	demand at bus i (MW)
XGENB(i)	i=1,nbx	R	generation at bus i (MW)
Common Block /CWHEEL/			
XWHEEL		R	power wheeled (MW)
JTSELL		I	type of selling party: 1 - bus 2 - utility 3 - pool
NSELL		I	number of the selling party
JTBUY		I	type of buying party: 1 - bus 2 - utility 3 - pool
NBUYER		I	number of the buying party
NWUTIL		I	number of wheeling utilities
NUTW(i)	i=1,nux	I	utility number of the wheeling utilities

Table 5-1 (continued)

Variable	Dimension	Type	Definition
NWLINE		I	number of lines for which the decomposed option is specified
NLTW(i)	i=1,nlx	I	line number for each line for which the decomposed option is specified
NLHOLD(i)	i=1,nlx	I	order of the lines in the NLTW array i.e. if NLTW(2) = 7 then NLHOLD(7) = 2
Common Block /COMPUT/			
IUSERB(i)	i=1,nbx	I	array holding the user assigned number of each bus, i being the internally assigned number of that bus
ISYSTB(i,j)	i=1,nbx j=1,nupx	I	bus number of the i^{th} bus in the independently dispatched entity j
ISYSTL(i,j)	i=1,nlx j=1,nupx	I	line number of the i^{th} line in the independently dispatched entity j
Common Block /CNAME/			
NAME		A80	title of study
Common Block /CSCEN/			
ISCEN		I	current scenario
PSCEN		R	probability of scenario (%)
HSCEN		I	hours in each scenario
NSCEN		I	number of scenarios
Common Block /CFILES/			
IFSPOT		I	unit number of spot price file

Section 6

WHEELING RATE MODULE "IWHEEL"

WRATES can investigate four (4) types of wheeling: utility to utility, bus to bus, utility to bus and bus to utility. WRATES can also model three (3) wheeling obligation classes; class O when the wheeler is obligated to serve both wheeling parties; class N where the wheeler is not obligated to serve any of the wheeling parties; and class ON where the wheeler is obligated to serve one wheeling party but not the other. The Wheeling Rate Module "IWHEEL" computes ideal wheeling rates based on the spot prices computed in ELDM for the generation dispatch including the wheeling transaction. It also computes revenue reconciled wheeling rates based on the revenue reconciliation parameters that are either user specified or computed in the COMPUTM module. The flow chart of IWHEEL is shown in Figure 6-1.

PROGRAM STRUCTURE

The flow of the IWHEEL module is controlled by the main program stored in IWHEEL.FOR which calls a total of 21 subroutines. For each scenario, IWHEEL reads from FSPOT.FIL the spot prices for the economic dispatch that include the wheeling transaction. IWHEEL also reads the generation costs from the split file 'FSPLIT.FIL" if there are pools and the user has requested that the wheeling rate be apportioned among pool numbers. IWHEEL computes the ideal wheeling rate for each utility, as well as the reconciled rate when required (i.e. JRATE = 2), and writes them to IWHEEL.OUT. For the equations describing the wheeling rates, refer to Appendix A2 of the user's manual. After all the scenarios are processed, IWHEEL computes the yearly statistics if required (i.e. JSTAT = 2). Finally, it computes the coordinates of points describing the ideal and the reconciled wheeling rates duration curves for each of the wheeling utilities and writes them to the file FDURC.FIL.

The following is a description of all the subroutines in IWHEEL, in the same order as they are called by the main routine that is stored in IWHEEL.FOR.

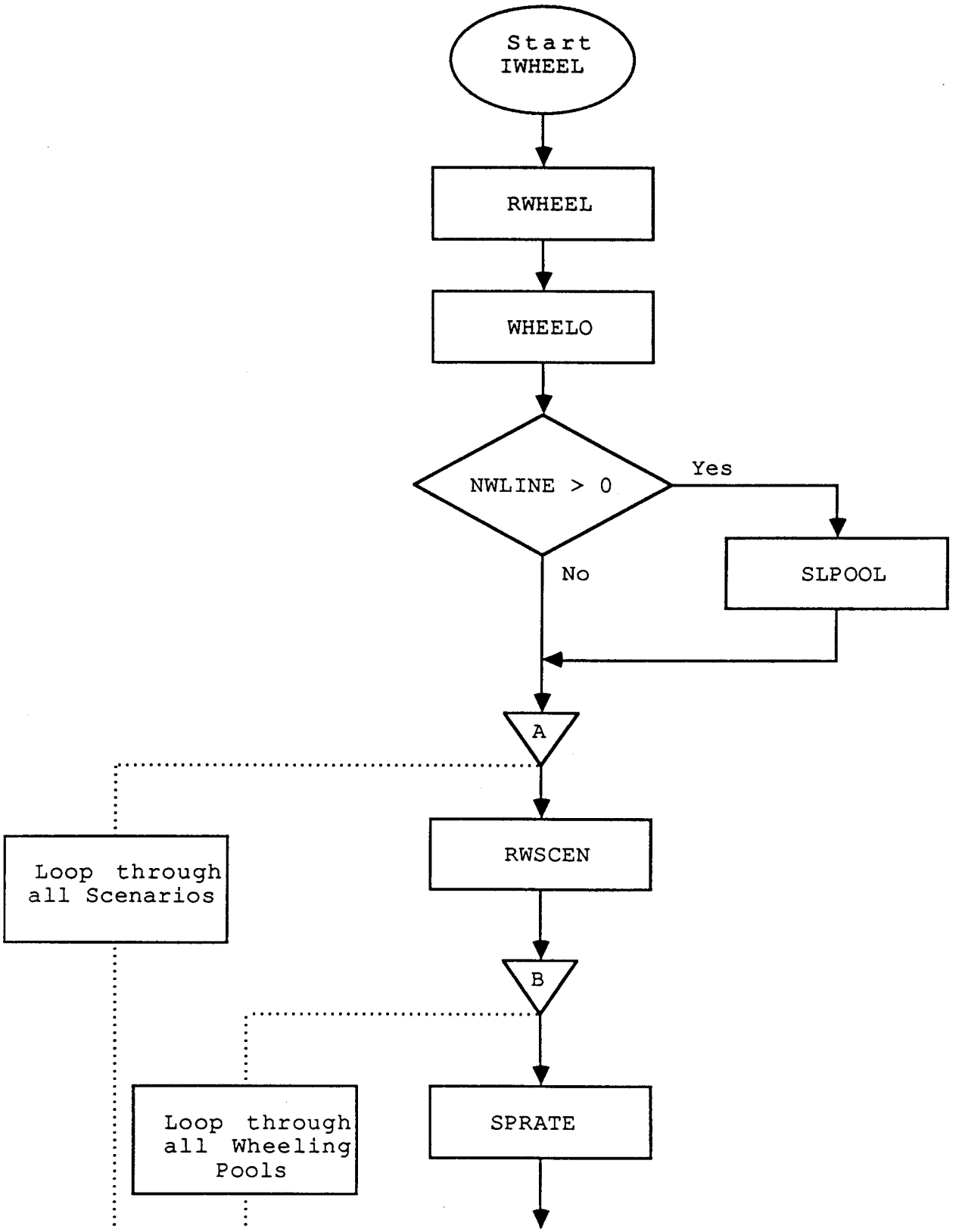

Figure 6-1. Flow Chart for the IWHEEL Module

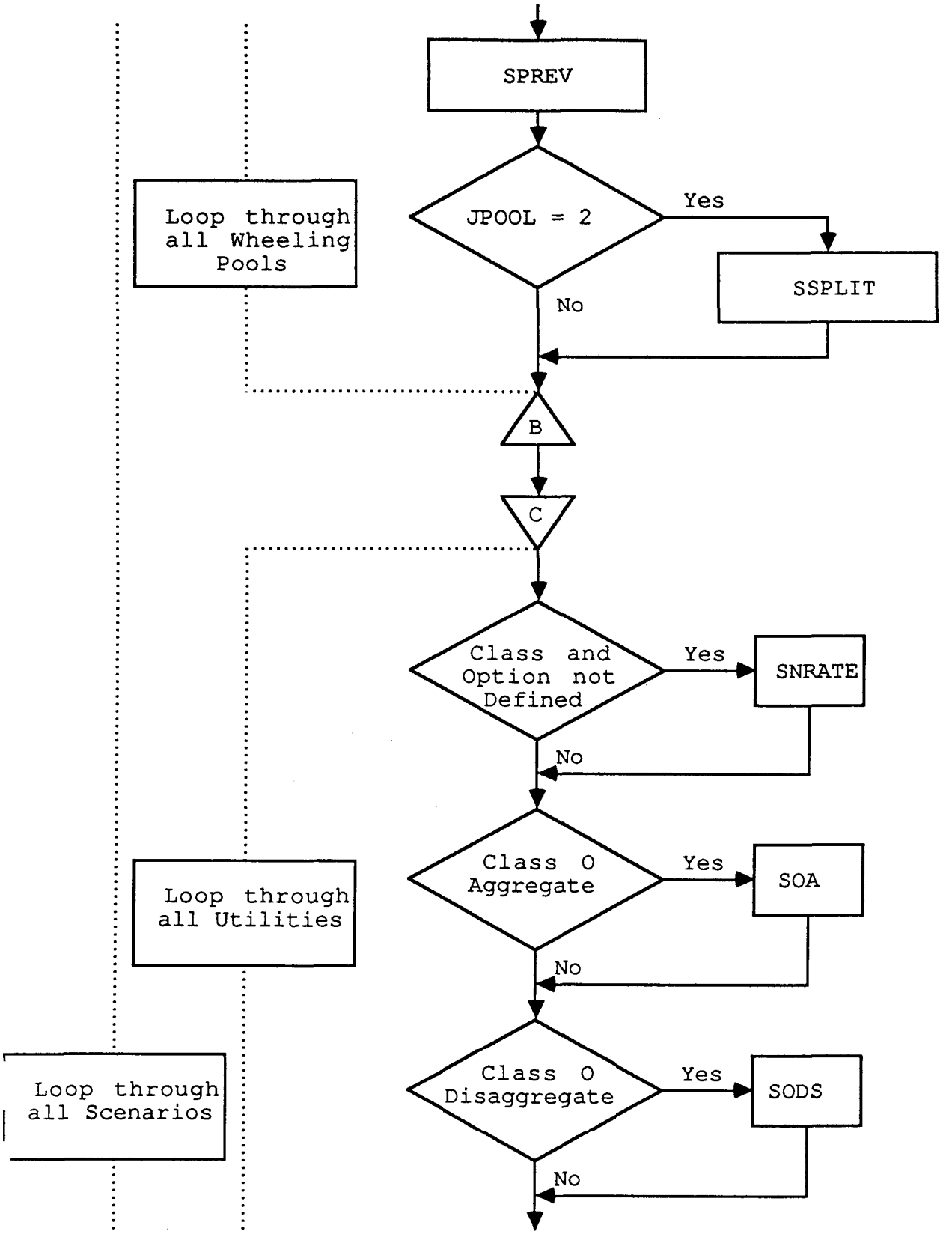

Figure 6-1 (Continued). Flow Chart for the IWHEEL Module

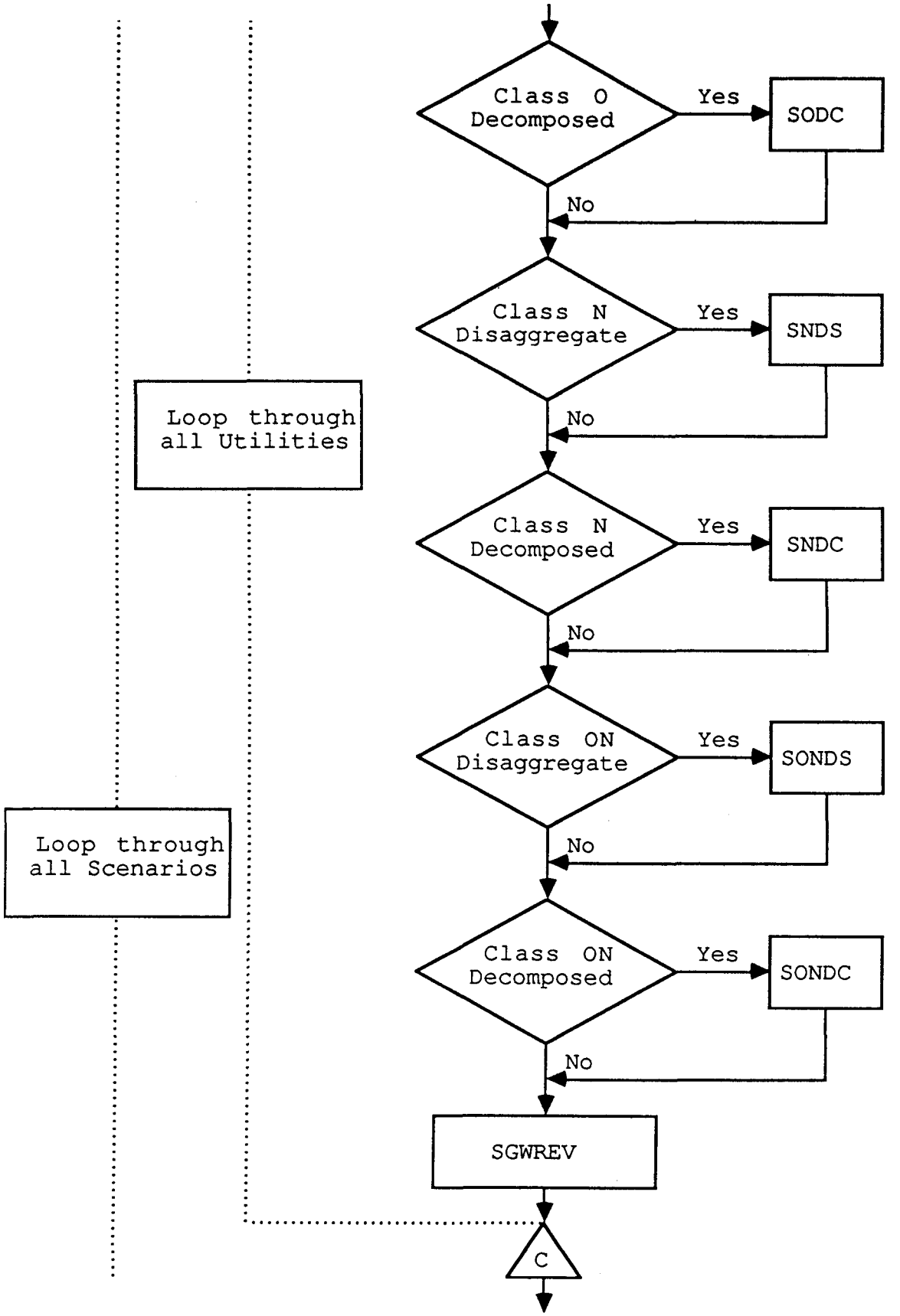

Figure 6-1 (Continued). Flow Chart for the IWHEEL Module

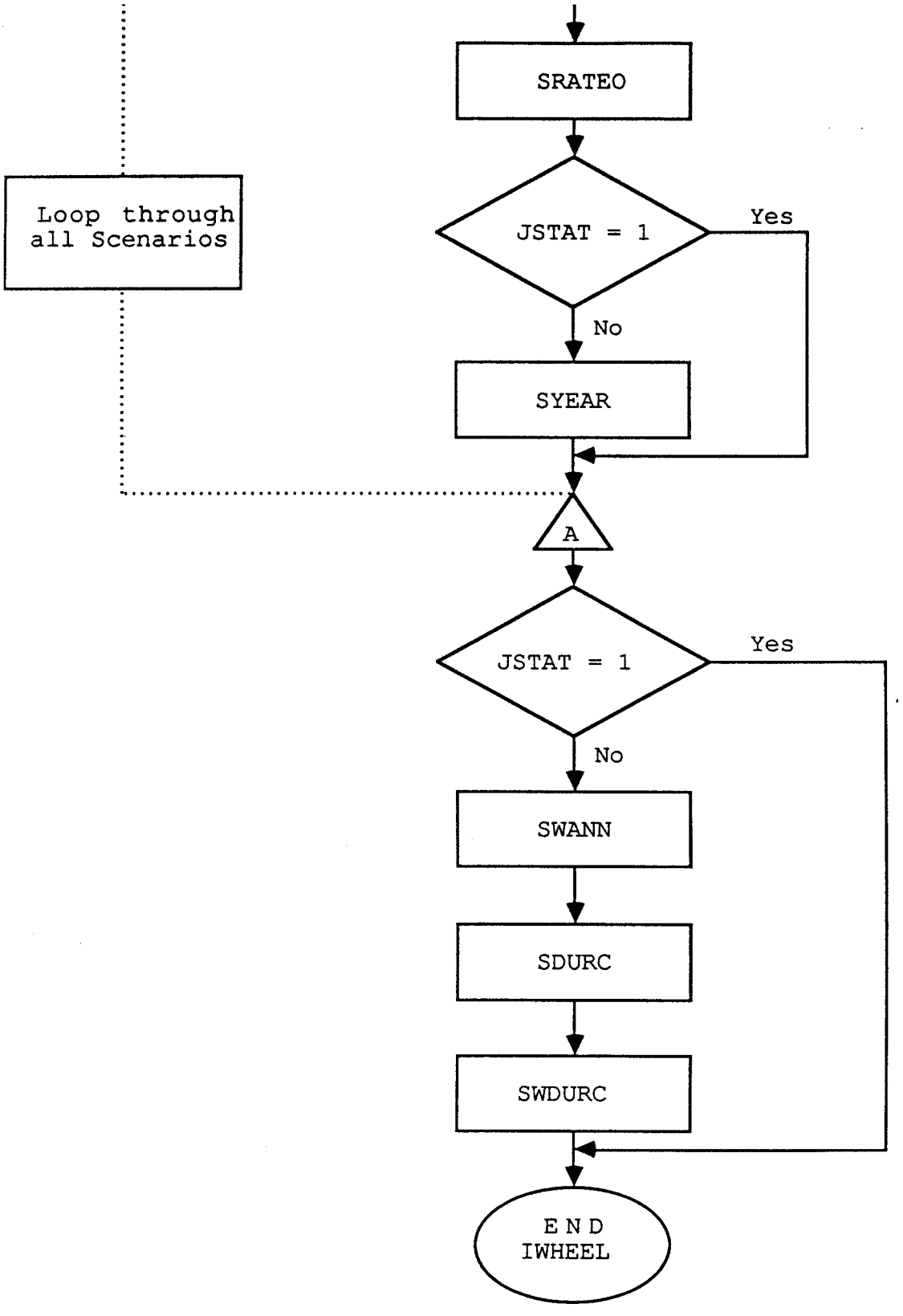

Figure 6-1 (Continued). Flow Chart for the IWHEEL Module

Subroutine RWHEEL

This subroutine is the first routine of the module IWHEEL. It reads the first four (4) records of the spot price file FSPOT.FIL, and processes the input data to fill variable arrays needed for the computation of wheeling rates.

Subroutine WHEELO

This subroutine writes the title block, the wheeling transaction data and the revenue reconciliation data to the file IWHEEL.OUT. It also computes the number of wheeling pools and the number of independently dispatched wheeling utilities. This file is later accessed by the LOTUS^{TM} 123 output macro that transfers the data to LOTUS^{TM} 123 worksheet.

Subroutine SLPOOL

This subroutine determines if there are lines with the decomposed option that belong to pools.

Subroutine RWSCEN

This subroutine is called once for each scenario; it reads the record in FSPOT.FIL holding the spot price data for the economic dispatch that includes the wheeling transaction. The number of this record is equal to twice the scenario number plus four (4), e.g., if the scenario number is 8, this record number is (2 x 8 + 4 =) 20. RWSCEN also reads the cost of generation before wheeling for the same scenario. With this data, it computes the cost of wheeling to each of the wheeling utilities by subtracting the generating cost without wheeling from the generating cost including the effect of the wheeling transaction.

Subroutine SPRATE

This subroutine computes the ideal wheeling rate for all the wheeling pools of the system. Note that WRATES does not compute reconciled rates for pools.

Subroutine SPREV

This subroutine computes the gross and net wheeling revenues for all the wheeling pools of the system.

Subroutine SOA

Within each scenario, subroutine SOA is called once for every wheeling utility that is obligated to serve both the buyer and the seller and for which the aggregate option is specified. It basically computes both the ideal and the reconciled wheeling rates for those utilities.

Subroutine SODS

Within each scenario, subroutine SODS is called once for every wheeling utility that is obligated to serve both the buyer and the seller and for which the disaggregate option is specified. It computes the ideal wheeling rate and the generation and network components of the reconciled wheeling rate for those utilities. The total reconciled wheeling rate is then calculated as the sum of the generation and network components.

Subroutine SODC

Within each scenario, subroutine SODC is called once for every wheeling utility that is obligated to serve both the buyer and the seller and for which the decomposed option is specified. It computes the ideal wheeling rate and the generation and line by line components of the reconciled wheeling rate for those utilities. The total reconciled rate is then calculated as the sum of the generation and all of the line components for the utility.

Subroutine SNDS

SNDS is similar to SODS except that the wheeling rates computed in SNDS are for the wheeling utilities that have no obligation to serve neither the seller nor the buyer.

Subroutine SNDC

SNDC is similar to SODC except that the wheeling rates computed in SNDC are for the wheeling utilities that have no obligation to serve neither the seller nor the buyer.

Subroutine SONDS

SONDS is similar to SODS, except that the wheeling rates computed in SONDS are for the wheeling utilities that have an obligation to serve either the seller or the buyer but not both.

Subroutine SONDC

Subroutine SONDC is similar to SODC, except that the wheeling rates computed in SONDC are for the wheeling utilities that have an obligation to serve either the seller or the buyer but not both.

Subroutine SGWREV

This subroutine is called for each wheeling utility after its wheeling rate is computed. SGWREV computes both the gross and net wheeling revenues of that utility.

Subroutine SRATEO

For each scenario, this subroutine writes the wheeling rates of all the wheeling utilities and pools in the IWHEEL.OUT file. This file is later accessed by the LOTUS™ 123 output worksheet.

Subroutine SYEAR

This subroutine is called for each scenario after the wheeling rates of all the wheeling utilities and pools have been computed. SYEAR is skipped if the user did not request yearly statistics (JSTAT = 1). It basically computes the weighted average wheeling rates and accumulates the yearly wheeling revenues for the wheeling utilities.

Subroutine SWANN

This subroutine is called after all the scenarios have been processed. Like SYEAR, it is skipped if the user did not request yearly statistics (JSTAT = 1). SWANN writes the yearly statistics to the output file IWHEEL.OUT.

Subroutine SDURC

This subroutine is called to build the wheeling rate duration curves of both the ideal and the reconciled rates for each wheeling utility. First it sorts both the ideal and reconciled wheeling rates in

descending order using the insertion method. Then it calculates the cumulative probabilities of those rates. The cumulative probability of a random variable at the value x is defined as the probability of this variable being larger or equal to x. The calculated cumulative probabilities will be the ordinates of the points on the wheeling rate duration curves.

Subroutine SWDURC

This subroutine is the last routine called by IWHEEL. Its function is to write the coordinates of the points on the wheeling rate duration curves to the output file FDURC.FIL. This file is later accessed by the LOTUS™ 123 output macro and the duration curves plotted in LOTUS™ 123 at the user's request.

LOTUS™ 123 INTERFACE

The IWHEEL module writes the wheeling results into the file IWHEEL.OUT, and the duration curve data into the file FDURC.FIL. Both files are DOS ASCII files. The last step of WRATES is to place this output data in a LOTUS™ 123 spreadsheet, complete with headers and labels. WRATES performs this task with two macros: an automatically executed one \0, and a user invoked one \G. Both macros are stored in the LOTUS™ 123 worksheet file TEMPLATE.OUT. Upon completion of the IWHEEL module, WRATES.BAT copies TEMPLATE.OUT to another worksheet file named AUTO123.WK1. Next, WRATES enters LOTUS™ 123. The AUTO123.WK1 file is automatically loaded into the spreadsheet, and the macro \0 is executed as soon as the spreadsheet is loaded.

Layout of the Spreadsheet

The TEMPLATE.OUT spreadsheet has five major sections: the output table, the work area, the graph table, the macro area, and the header area. The output table appears in columns A through L. The length of the output table depends on the number of utilities in the system, the number of pools in the system, whether the decomposed option was selected and whether yearly statistics were requested. The work area is in columns M through Y. When IWHEEL.OUT and FDURC.FIL are imported to the spreadsheet, they are manipulated by the macros in this area. The graph table is located in columns Z through AX and rows 1 through 111. The macro area is in columns BC to BM. The graph macro \G begins in cell BE2 and the output macro \0 begins in cell BK2. The

header area is in columns BV to CG and rows 1 through 64; this is where the headers for the output table are stored.

Macro \0

This macro is executed automatically when LOTUS™ 123 is invoked. Its function is to sort and display the wheeling rates computed by IWHEEL that are stored in IWHEEL.OUT. This macro has one main routine and 5 subroutines.

Main Routine. The main routine begins in cell BK2 of the spreadsheet file named TEMPLATE.OUT. It first imports the file IWHEEL.OUT into the work area of the spreadsheet. Next, it moves the title of the study, the system information and the utility revenue reconciliation data into the output table. It checks the data to determine if the decomposed option had been selected, in which case line revenue reconciliation information is present in the output data. If the line data is present, the routine copies the line header, located in the range named HEADRL, into the output table and the line data is moved under this header. The rest of the output data contains the wheeling rates. Depending on the system configuration and the output requested of the program, data may exist for up to five separate levels of wheeling rates. The macro sorts the data so that it is no longer grouped by scenario but by these five levels. After the data is sorted in the proper order, the macro determines which data is present and calls the appropriate subroutines to place these data in the output table. Five subroutines are used to place the five types of wheeling data into the appropriate tables. In addition to these five subroutines, there are five subroutines that are called if annual data is present. One of these subroutines is called by each of the five subroutines for the different wheeling rates. The subroutines are:

POOL, ANNP	Wheeling rates for pools;
PUTILITY, ANNUP	Utility components of pool wheeling rates;
PLINE, ANNLP	Line components of pool wheeling rates;
IUTILITY, ANNUI	Wheeling rates for independently dispatched utilities;
ILINE, ANNLI	Wheeling rates for lines within independently dispatched utilities.

Graph Macro \G

The graph macro \G begins in cell BE2 of the output spreadsheet TEMPLATE.OUT. The purpose of this macro is to graph the wheeling rate duration curves printed in FDURC.FIL. The graph table is located in columns Z through AX. Named graphs have been created with the ranges in this table corresponding to the data ranges in the named graphs. For example, named graph 'IDEAL UTILITY 1' has its X range defined as Z12 ... Z111 and its first data range defined as AA12 ... AA111. The macro \G imports the file FDURC.FIL into the work area and copies the data for each graph into the graph table. Then it enters the graph menu and prompts the user to choose the graph to make current.

COMMON BLOCKS AND VARIABLE DICTIONARY

This section describes the common variables used in the IWHEEL module. Two files define and initialize the variables in the module. The first, COMMON.FOR, defines the common blocks. The code in this file is included in all the subroutines of IWHEEL. The second, BLKDATA.FOR, is a subprogram that initializes the variables defined in the common blocks. Table 6-1 is a listing of the common variables. The array variables are dimensioned by parameters that are set in a parameter statement in COMMON.FOR. The parameters are:

- o nlx maximum number of lines
- o nbx maximum number of buses
- o nux maximum number of utilities
- o nsx maximum number of scenarios

Table 6-1

COMMON VARIABLES IN THE IWHEEL MODULE

Variable	Dimension	Type	Definition
Common Block /CSYS/			
NBUS		I	number of buses
NLINE		I	number of lines
NUTIL		I	number of utilities
NPOOL		I	number of pools
NIUTIL		I	number of independently dispatched utilities
NUP		I	number of independently dispatched entities
NWP		I	number of wheeling pools
NWIU		I	number of independently dispatched wheeling utilities
NWUTIL		I	number of wheeling utilities
NWLINE		I	number of lines for which the decomposed option is requested
JRATE		I	wheeling rate option: 1 - ideal rates only 2 - ideal and reconciled rates
JSTAT		I	pool analysis 1 - do not run yearly statistics 2 - run yearly statistics
JPOOL		I	pool analysis 1 - do not run pool analysis 2 - run pool analysis
Common Block /CNOW/			
KUTIL		I	current utility
KP		I	current pool
KUP		I	current independently dispatched entitity

Table 6-1 (continued)

Variable	Dimension	Type	Definition
Common Block /CPOOL/			
IUPOOL(i)	i=1,nux	I	pool to which utility i belongs
NPRANK(i)	i=1,npx	I	number of utilites affiliated with pool i
IUP(i)	i=1,nux	I	user assigned number of the utility that has the dispatch number i
IDISNU(i)	i=1,nux	I	program assigned number for the independently dispatch utility i
NBUSUP(i)	i=1,nupx	I	number of buses belonging to the independently dispatched entity i
NLINEUP(i)	i=1,nupx	I	number of lines belonging to the independenly dispatch entity
ISYSTB(i,j)	i=1,nbx j=1,nupx	I I	bus number of the i^{th} bus in the independently dispatch entity j
ISYSTL(i,j)	i=1,nlx, j=1,nupx	I	line number of the i^{th} line in the independently dispatched entity j
NORDER (i)	i=1,nlx	I	order of the i_{t}h line within the pool to which it belongs
Common Block /CLOAD/			
XGENB(i)	i=1,nbx	R	generation at bus i (MW)
XDEMD(i)	i=1,nbx	R	demand at bus i (MW)
XLOSS(i)	i=1,nupx	R	losses within independently dispatched entity i (MW)
XNETEI(i)	i=1,nupx	R	net energy interchange within independently dispatched entity i (MW)
Common Block /CPRICE/			
XSPOTP(i)	i=1,nbx	R	spot price at bus i
XGAMMA(i)	i=1,nupx	R	marginal generation cost for independently dispatched entity i

Table 6-1 (continued)

Variable	Dimension	Type	Definition
XLR(i)	i=1,nlx	R	line parameter such that line losses equal XLR times the square of the line flow
XLSPOT(i,j)	i=1,nlx j=1,nbx	R	line i component of the spot price at bus j
XPENF(i)	i=1,nlx	R	derivative of the penalty function due to a flow on line i exceeding its flow limit
XPENFC(i)	i=1,nlx	R	penalty function due to a flow on line i exceeding its flow limit ($)
XP(i)	i=1,nlx	R	derivative of the penalty function due to a flow on line i exceeding its flow limit (prewheeling)
XPS(i)	i=1,nlx	R	derivative of the penalty function due to a flow on line i exceeding its flow limit (incremental case)
Common Block /CREVP/			
NUMBUP	i=1,nupx	I	number of the independently dispatched wheeling entity:
ITYPUP(i)	i=1,nupx	I	type of independently dispatched wheeling entity: 1 - utility 2 - pool
IOPT(i)	i=1,nupx	I	rate option of the wheeling pools 1 - aggregate 2 - disaggregate 3 - decomposed
Common Block /CREVU/			
NUTW(i)	i=1,nux	I	number of each wheeling utility
ICLASS(i)	i=1,nux	I	obligation class for wheeling utility i: 1 - obligation to serve 2 - no obligation to serve 3 - obligation/no obligation to serve

Table 6-1 (continued)

Variable	Dimension	Type	Definition
IOPTON(i)	i=1,nux	I	rate option for utility i: 1 - aggregate 2 - disaggregate 3 - decomposed
XTREVM(i)	i=1,nux	R	total revenue reconciliation multiplier
XGREVM(i)	i=1,nux	R	generation revenue reconciliation multiplier
XNREVM(i)	i=1,nux	R	network revenue reconciliation multiplier
IREVM(i)	i=1,nux	I	revenue multiplier status 1 - provided 2 - to be computed
XTREV(i)	i=1,nux	R	total capital revenues (K$)
XGREV(i)	i=1,nux	R	generation capital revenues (K$)
XNREV(i)	i=1,nux	R	network capital revenues (K$)
NLTW(i)	i=1,nlx	I	number of each line with the decomposed option
XLREVM(i)	i=1,nlx	R	line revenue reconciliation multiplier
XLREV(i)	i=1,nlx	R	line capital revenues (K$)
Common Block /CYEAR/			
YRATEI(i)	i=1,nux	R	yearly ideal wheeling rate ($/MWh)
YRATER(i)	i=1,nux	R	yearly reconciled wheeling rate ($/MWh)
YRATRG(i)	i=1,nux	R	generation component of the yearly reconciled wheeling rate ($/MWh)
YRATRN(i)	i=1,nux	R	network component of the yearly reconciled wheeling rate ($/MWh)

Table 6-1 (continued)

Variable	Dimension	Type	Definition
YRATRL(i)	i=1,nlx	R	line component of the yearly reconciled wheeling rate ($/MWh)
YWCST(i)	i=1,nux	R	yearly cost of wheeling (K$)
YWREV(i)	i=1,nux	R	yearly wheeling revenues (K$)
YWREVG(i)	i=1,nux	R	generation component of the yearly wheeling revenues (K$)
YWREVN(i)	i=1,nux	R	network component of the yearly wheeling revenues (K$)
YNTREV(i)	i=1,nux	R	yearly net wheeling revenues (K$) YNETREV = YWREV-YWCST
YWREVL(i)	i=1,nlx	R	line components of the yearly wheeling revenues (K$)
Common Block /CWHEEL/			
XWHEEL		R	power wheeled (MW)
JTSELL		I	type of selling party: 1 - bus 2 - utility 3 - pool
NSELL		I	number of the selling party
JTBUY		I	type of buying party: 1 - bus 2 - utility 3 - pool
NBUYER		I	number of the buying party
Common Block /COMPUT/			
IUSERB(i)	i=1,nbx	I	user assigned number for bus i
IPROGB(i)	i=1,nbx	I	program assigned number for the bus that has a user assiged number equal to i

Table 6-1 (continued)

Variable	Dimension	Type	Definition
Common Block /CIDENT/			
IBUTIL(i)	i=1,nbx	I	utility number of bus i
ILUTIL(i)	i=1,nlx	I	utility number of line i
IBPOOL(i)	i=1,nbx	I	utility number of line i
ILPOOL(i)	i=1,nlx	I	pool to which line i belongs
Common Block /CLINE/			
XLINF(i)	i=1,nlx	R	flow on line i (MW)
XLOSSL(i)	i=1,nlx	R	losses in line I (MW)
XH(i)	i=1,nlx	R	changes of flow through line i due to a change in injection at the seller bus or at the swing bus of the selling utility
Common Block /CGWREV/			
XWFUEL(i)	i=1,nux	R	generation cost of utiity i including the effect of the wheeling transaction ($)
XFUEL(i)	i=1,nux	R	generation costs of utility i excluding the effect of the wheeling transaction
XPGCA(i)	i=1,npx	R	genertion costs of pool i including the effect of the wheeling transaction ($)
XPGCB(i)	i=1,npx	R	generation costs of pool i excluding the effect of the wheeling transaction ($)
XWCST(i)	i=1,nux	R	scenario hourly cost of wheeling for utility i (K$)
XWREV(i)	i=1,nux	R	scenario hourly wheeling revenues for utility i (K$)
XWREVG(i)	i=1,nux	R	scenario hourly generation component of the wheeling revenues in utility i (K$)

Table 6-1 (continued)

Variable	Dimension	Type	Definition
XWREVN(i)	i=1,nux	R	scenario hourly network component of the wheeling revenues in utility i (K$)
XNTREV(i)	i=1,nux	R	scenario hourly net wheeling revenues in utility i (K$)
XNTPER(i)	i=1,nux	R	scenario hourly net revenues per MWh wheeled for utility i ($)
XWREVL(i)	i=1,nlx	R	scenario hourly line component of the wheeling revenues (K$)
Common Block /CRATES/			
XRATEI(i)	i=1,nux	R	scenario hourly ideal wheeling rate ($/MWh)
XRATER(i)	i=1,nux	R	scenario hourly reconciled wheeling rate ($/MWh)
XRATIG(i)	i=1,nux	R	scenario hourly generation component of the ideal wheeling rate ($/MWh)
XRATIN(i)	i=1,nux	R	scenario hourly network component of the ideal wheeling rate ($/MWh)
XRATIL(i)	i=1,nlx	R	scenario hourly line component of the ideal wheeling rate ($/MWh)
XRATRG(i)	i=1,nux	R	scenario hourly generation component of the reconciled wheeling rate ($/MWh)
XRATRN(i)	i=1,nux	R	scenario hourly network component of the reconciled wheeling rate ($/MWh)
XRATRL(i)	i=1,nlx	R	scenario hourly line component of the reconciled wheeling rate ($/MWh)
Common Block /CPRATE/			
XRATEP(i)	i=1,npx	R	scenario (hourly) ideal wheeling rate of pool i ($MWh)

Table 6-1 (continued)

Variable	Dimension	Type	Definition
XWPREV(i)	i=1,npx	R	scenario (hourly) wheeling revenues of pool i (K$)
XNPREV(i)	i=1,npx	R	scenario (hourly) net wheeling revenues of pool i (K$)
XNPPER(i)	i=1,npx	R	scenario (hourly) wheeling revenues of pool i per unit of energy wheeled ($MWh)
XWPCST(i)	i=1,npx	R	scenario (hourly) wheeling cost for pool i (K$)
YRATEP(i)	i=1,npx	R	yearly wheeling rate for pool i ($/MWh)
YWPREV(i)	i=1,npx	R	yearly gross wheeling revenues for pool i (K$)
YNPREV(i)	i=1,npx	R	yearly net wheeling revenues for pool i (K$)
YWPCST(i)	i=1,npx	R	yearly wheeling costs for pool i (K$)
Common Block /CSPLIT/			
IWFLAG(npx)	i=1,npx	I	flag indication whether a wheeling rate has been requested for pool i: 0 - no rate has been requested for pool i 1 - a wheeling rate has been requested for pool i
XPRATES(i)	i=1,npx	R	scenario (hourly) wheeling rate for pool i when the wheeled quantity is equal to the specified amount + 10
XPGCI(i)	i=1,npx	R	generation costs of pool i when the wheeled quantity is equal to the specified amount + 10
XIFUEL(i)	i=1,nux	R	generation costs of utility i when the wheeled quantity is equal to the specified amount + 10
Common Block /CDURC/			
WHEELT(i)	i=1,nsx	R	probability of scenario i (%)

Table 6-1 (continued)

Variable	Dimension	Type	Definition
WHEELI(i,j)	i=1,nsx j=1,nux	R	value of the ideal wheeling rate for utility j at the i^{th} point of the duration curve rate
WHEELR(i,j)	i=1,nsx j=1,nux	R	value of the reconciled wheeling rate for utility j at the i^{th} point of the rate duration curve
XPROBI(i,j)	i=1,nsx j=1,nux	R	cumulative probability of WHEELI(i,j)
XPROBR(i,j)	i=1,nsx j=1,nux	R	cumulative probability of WHEELR(i,j)
Common Block /CNAME/			
NAME		A80	title of study
Common Block /CINNER/			
NSCEN		I	number of scenarios
KSCEN		I	current scenario number
PSCEN		R	probability of scenario (%)
Common Block /CFILES/			
IFSPOT		I	unit number of spot price file
IFSPLIT		I	unit number of the "split file"
IOUT		I	unit number of the out put file IWHEELOUT
IFDURC		I	unit number of the duration curve file

Section 7

FILE DESCRIPTION

This section presents detailed descriptions of the files read and written by WRATES. The input and output file are described as well as the intermediate and message files.

SCENARIO FILE

The scenario file, FSCEN.FIL, is a direct access unformatted file written by the SCENGEN module and read by the ELDM module. It has 3 record types and a maximum of 52 records. The first two records are wrwritten by subroutine SYSOUT of the SCENGEN module and read by RSYSTM of the ELDM module. All of the following records (one for each scenario) are written by SCENGO.FOR of the SCENGEN module and read also by RSYSTM.FOR of the ELDM module.

The scenario file has the following characteristics:

- FORTRAN unit number: 10
- Maximum Record Length: 10,528 bytes
- Maximum Number of Records: 52
- Maximum Size of File: 547,456 bytes

Table 7-1 lists and describes the FSCEN.FIL variables. In this table KSCEN refers to the scenario number.

Table 7-1

VARIABLES IN SCENARIO FILE "FSCEN.FIL"

Variable	Dimension	Type	Definition
Record 1 (type 1) System Record			
NAME		A80	title of study case
NPOOL		I	number of pools
NIUTIL		I	number of independently dispatched utilities
NUTIL		I	total number of utilities
NUP			number of economically dispatched entities
NBUS		I	number of buses
NLINE		I	number of lines
ISWGBN		I	system swing bus number
XWHEEL		R	wheeling quantity (MW)
JTSELL		I	type of selling party: 1 - bus 2 - utility 3 - pool
NSELL		I	number of the selling party
JTBUY		I	type of the buying party: 1 - bus 2 - utility 3 - pool
NBUYER		I	number of the buying party
ISWBGS		I	swing bus of the selling utility or number of the selling bus
NSCEN		I	number of scenarios
JELDM		I	ELDM option: 1 - run ELDM with and without wheeling 2 - do not run ELDM 3 - run ELDM with wheeling only 4 - run ELDM without wheeling only

Table 7-1 (continued)

Variable	Dimension	Type	Definition
JRATE		I	wheeling rate option: 1 - ideal rates only 2 - ideal and reconciled rates
JSTAT		I	statistics option: 1 - do not run yearly statistics 2 - run yearly statistics
JPOOL		I	pool analysis 1 - do not run pool analysis 2 - run pool analysis, i.e. apportion the pool wheeling rate among the member utilities
Record 2 (type 2) Pool Record			
IUPOOL(i)	i=1,NUTIL	I	pool to which the utility belongs
IUP(i)	i=1, NUPX	I	user assigned number of the utility that is independently dispatched
IDISNU(i)	i=1,NUTIL		program assigned number to the independently dispatched utility i IUP(IDISNU(i)) = 1
Record KSCEN + 2 (type 3) Scenario Data			
KSCEN		I	scenario number
PSCEN		R	probability of scenario (%)
XNETEI(i)	i=1,NUP	R	utility (or pool) net energy interchange (MW)
IBUTIL(i)	i=1,NBUS	I	utility to which the bus belongs

Table 7-1 (continued)

Variable	Dimension	Type	Definition
XDEMD(i)	i=1,NBUS	R	demand at each bus (MW)
XGENCP(i,j)	i=1,NBUS j=1,NPTX	R	generation levels at points on the supply curve (MW)
XGENCO(i,j)	i=1,NBUS j=1,NPTX	R	marginal costs at points on the supply curve ($/MWh)
XUNEGM(i)	i=1,NBUS	R	cost of unserved energy at onset ($/MW)
XUNESL(i)	i=1,NBUS	R	slope of unserved energy cost at each bus ($/MW/MW)
IGENPT(i)	i=1,NBUS	R	number of points on the marginal cost curve for each bus
ILUTIL(i)	i=1,NLINE	I	utility to which the line belongs
IBEGB(i)	i=1,NLINE	I	number of the bus at the beginning of each line
IENDB(i)	i=1,NLINE	I	number of the bus at the end of each line
XRES(i)	i=1,NLINE	R	resistance of each line (Ohms/V^2)
XINDUC(i)	i=1,NLINE	R	inductance of each line (Ohms/V^2)
ICONST(i)	i=1,NLINE	I	type of constraint for each line: 0 - no constraint 1 - soft constraint 2 - hard constraint
XFLPOS(i)	i=1,NLINE	R	positive flow limit (MW)
XFLNEG(i)	i=1,NLINE	R	negative flow limit (MW)
XPEPOS(i)	i=1,NLINE	R	penalty parameter if flow exceeds limit in the positive direction
XPENEG(i)	i=1,NLINE	R	penalty parameter if flow exceeds limit in the negative direction

SPOT PRICE FILE

The spot price file, FSPOT.FIL, is a direct access unformatted file which contains computed spot price information. The file is written by the SCENGEN module and the ELDM module. It is read by the COMPUTM and the IWHEEL modules. The file contains five record types. The first four record types occur only once and are the respective first four records of the file. The remaining records (2 for each scenario) are of the fifth type.

The first two records, the system record and the pool record, are written by both subroutine RSPOTP of the COMPUTM module and subroutine RWHEEL of the IWHEEL module. The third record is the wheeling record. It contains wheeling information and revenue reconciliation data. It is written by subroutine SYSOUT of the SCENGEN module and read by subroutine RSPOTP of the COMPUTM module. If revenue reconciliation multipliers are computed in COMPUTM, subroutine WSPOTP of that module rewrites this record. Lastly, the record is read by subroutine RWHEEL of the IWHEEL module. The fourth record is the topology record. It is written by subroutine SFOUT of the ELDM module and read by subroutine RSPOTP of the COMPUTM module and subroutine RWHEEL of the IWHEEL module. The fifth record type is the spot price record. There are two spot price records for each scenario: the odd numbered records contain spot price data without wheeling, and the even numbered records contain spot price data with wheeling. All of those records are written by subroutine SWSPOTI of the ELDM module. The odd numbered records are read by subroutine RSCEN of the COMPUTM module and by subroutine RWSCEN of the IWHEEL module. The even numbered records are only read by subroutine RWSCEN of the IWHEEL module.

The spot price file has the following characteristics:

- FORTRAN unit number: 15
- Maximum record length: 25,948 bytes
- Maximum number of records: 104
- Maximum size of file: 2,698,592 bytes

Table 7.2 lists and describes the FSPOT.FIL variables. In this table KSCEN refers to the scenario number.

Table 7-2

VARIABLES IN THE SPOT PRICE FILE "FSPOT.FIL"

Variable	Dimension	Type	Definition
Record 1 (type 1) System Record			
NAME		A80	title of case study
NPOOL		I	number of pools
NIUTIL		I	number of independently dispatched utilities
NUTIL		I	number of utilities
NUP		I	number of economically dispatched entities
NBUS		I	number of buses
NLINE		I	number of lines
ISWGBN		I	system swing bus
XWHEEL		R	wheeling quantity (MW)
JTSELL		I	type of selling party: 1 - bus 2 - utility 3 - pool
NSELL		I	number of the selling party
JTBUY		I	type of the buying party: 1 - bus 2 - utility 3 - pool
NBUYER		I	number of the buying party
ISWGBS		I	swing bus of the selling party
NSCEN		I	number of scenarios
NWLINE		I	number of lines for which wheeling revenues or revenue reconciliation multipliers are to be computed
NWUTIL		I	number of wheeling utilities for which revenue reconcilation parameters are specified

Table 7-2 (continued)

Variable	Dimension	Type	Definition
JELDM		I	ELDM option: 1 - run ELDM with and without wheeling 2 - do not run ELDM 3 - run ELDM with wheeling only 4 - run ELDM without wheeling only
JRATE		I	wheeling rate option: 1 - ideal rates only 2 - ideal and reconciled rates
JSTAT		I	statistics option: 1 - do not run yearly statistics 2 - run yearly statistics
JPOOL		I	pool analysis option 1 - do not run pool analysis 2 - run pool analysis i.e. apportion the pool wheeling rate among the member utilities
Record 2 (type 2): Pool Record			
IUPOOL(i)	i=1,NUTIL	I	pool to which the utility belongs
NPRANK(i)	i=1,NPOOL	I	number of utilities in pool i
IUP(i)	i=1,NUPX,	I	user assigned number of the independently dispatched utility i
IDISNU(i)	i=1,NUTIL	I	program assigned number to the independently dispatched utility i IUP(IDISNU(i)) = i
Record 3 (type 3): Wheeling Record			
NUTW(i)	i=1,NWUTIL	I	user assigned number of each wheeling utility for which revenue reconciliation parameters are specified
IOPTON(i)	i=1,NUTW(j) j=1, NWUTIL	I	rate option for each wheeling utility for which revenue reconciliation parameters are specified 1 - aggregate 2 - disaggregate 3 - decomposed

Table 7-2 (continued)

Variable	Dimension	Type	Definition
ICLASS(i)	i=1,NUTW(j) j=1,NWUTIL	I	obligation class for each wheeling utility for which revenue reconciliation parameters are specified 1 - obligation to serve 2 - no obligation to serve 3 - obligation/no obligation to serve
IREVM(i)	i=1,NUTW(j) j=1,NWUTIL	I	multiplier status for each wheeling utility for which revenue reconciliation parameters are specified 1 - revenue multiplier provided 2 - revenue multiplier to be computed
XTREVM(i)	i=1,NUTW(j) j=1,NWUTIL	R	total revenue reconciliation for each wheeling utility for which revenue reconciliation parameters are specified
XGREVM(i)	i=1,NUTW(j) j=1,NWUTIL	R	generation revenue reconciliation multiplier for each wheeling utility for which revenue reconciliation parameters are specified
XNREVM(i)	i=1,NUTW(j) j=1, NWUTIL	R	network revenue reconciliation multiplier for each wheeling utility for which revenue reconciliation parameters are specified
XTREV(i)	i=1,NUTW(j) j=1,NWUTIL	R	total revenue requirement (K$) for each wheeling utility for which revenue reconciliation parameters are specified
XGREV(i)	i=1,NUTW(j) j=1,NWUTIL	R	generation revenue requirement (K$) for each wheeling utility for which revenue reconciliation parameters are specified
XNREV(i)	i=1,NUTW(j) j=1,NWUTIL	R	network revenue requirement (K$) for each wheeling utility for which revenue reconciliation parameters are specified

Table 7-2 (continued)

Variable	Dimension	Type	Definition
NLTW(i)	i=1,NWLINE	I	line number of each line for which reconciliation multipliers or revenue requirements are to be computed
XLREVM(i)	i=1,NLTW(j) j=1,NWLINE	R	line by line revenue reconciliation multiplier
XLREV(i)	i=1,NLTW(j) j=1,NWLINE	R	line by line revenue requirements
Record (type 4): Topology Record			
NBUSUP(i)	i=1,NUP	I	number of buses in each independently dispatched entity
NLINEUP(i)	i=1,NUP	I	number of lines in each independently dispatched entity
IUSERB(i)	i=1,NBUS	I	user assigned bus number
IBUTIL(i)	i=1,NBUS	I	utility to which bus i belongs
ILUTIL(i)	i=1,NLINE	I	utility to which line i belongs
IBPOOL(i)	i=1,NBUS	I	pool to which bus i belongs
ILPOOL(i)	i=1,NLINE	I	pool to which line i belongs
ISYSTB(i,j)	i=1,NBUSUP(j) j=1,NUP	I	system bus number of the i^{th} bus in the j^{th} independently dispatched entity
ISYSTL(i,j)	i=1,NLINEUP(j) j=1,NUP	I	system line number of the i^{th} line in the j^{th} independently dispatched entity
Record 2*KSCEN + 3 (type 5) Spot Price Without Wheeling			
KSCEN		I	scenario number
PSCEN		R	scenario probability (%)
XUGCB(i)	i=1,NUTIL	R	fuel cost for each utility before wheeling ($)
XPGCB(i)	i=1,NPOOL	R	fuel cost for each pool before wheeling ($)

Table 7-2 (continued)

Variable	Dimension	Type	Definition
XPENFC(i)	i=1,NLINE	R	penalty function due to flow constraint ($)
XGENB(i)	i=1,NBUS	R	generation at each bus (MW)
XDEMD(i)	i=1,NBUS	R	demand at each bus (MW)
XLINF(i)	i=1,NLINE	R	line flow (MW)
XLOSSL(i)	i=1,NLINE	R	losses on each line (MW)
XNETEI(i)	i=1,NUP	R	net energy interchange for each independently dispatched entity (MW)
XLOSS(i)	i=1,NUP	R	losses within each independently dispatched entity (MW)
XGAMMA(i)	i=1,NUP	R	gamma for each independently dispatched entity ($/MWh)
XSPOTP(i)	i=1,NBUS	R	Spot price at each bus ($/MWh)
XLSPOT(i,j)	i=1,NLINEUP (IBPOOL(j)) j=1,NBUS	R	line components of the spot price at each bus ($/MWh)
Record 2*KSCEN + 4 (type 5) Spot Price with Wheeling			
KSCEN		I	scenario number
PSCEN		R	scenario probability (%)
XUGCA(i)	i=1,NUTIL	R	fuel cost for each utility including the effect of wheeling ($)
XPGCA(i)	i=1,NPOOL	R	fuel cost for each pool including the effect of wheeling ($)
XPENFC(i)	i=1,NLINE	R	penalty function due to flow constraints ($)
XGENB(i)	i=1,NBUS	R	generation at each bus (MW)
XDEMD(i)	i=1,NBUS	R	demand at each bus (MW)
XLINF(i)	i=1,NLINE	R	line flow (MW)
XLOSSL(i)	i=1,NLINE	R	losses on each line (MW)

Table 7-2 (continued)

Variable	Dimension	Type	Definition
XNETEI(i)	i=1,NUP	R	net energy interchange for each independently dispatched entity (MW)
XLOSS(i)	i=1,NUP	R	losses in each utility (or in each pool when JPOOL = 2) (MW)
XGAMMA(i)	i=1,NUP	R	gamma of each independently dispatched entity ($/MWh)
XSPOTP(i)	i=1,NBUS	R	spot price at each bus ($/MWh)
XLSPOT(i,j)	i=1,NLINEUP (IBPOOL(j)) j=1,NBUS	R	line component of the spot price at each bus ($/MWh)
XH(i)	i=1,NLINE	R	change in line flow due to an injection change at the swing bus
XPENF(i)	i=1,NLINE	R	derivative of the penalty function due to overflow on each line
XLR(i)	i=1,NLINE	R	parameter such that line losses = XLR x line flow squared

SPLIT FILE FSPLIT.FIL

The split file, FSPLIT.FIL, is a direct access unformatted file written by subroutine SWSPOTP of the ELDM module when the user specifies that the pool wheeling rates have to be apportioned among pool members. It has one record type and a maximum of 50 records (one per scenario). This file is read by subroutine SSPLIT of the IWHEEL module. It holds the result of the economic dispatch for the case of wheeling a quantity equal to the specified quantity and 10 MW.

The split file has the following charactertiscs:

- o FORTRAN unit number: 16
- o Maximum record length: 1776 bytes
- o Maximum number of Records: 50
- o Maximum size of file: 88,800 bytes

Table 7-3 lists and describes the FSPLIT.FIL variables.

Table 7-3

VARIABLES IN SPLIT FILE "FSPLIT.FIL"

Variable	Dimension	Type	Definition
Record KSCEN (type 1)			
KSCEN		I	scenario number
PSCEN		R	scenario probability %
XUGCA(i)	i=1,NUTIL	R	fuel costs for each utility including the effect of wheeling (XWHEEL=10)
XPGCA(i)	i=1,NPOOL	R	fuel costs for each pool including the effect of wheeling (XWHEEL+10)
XPENFC(i)	i=1,NLINE	R	penalty function due to flow constraints ($)

ERROR FILE ERROR.FIL

The error file ERROR.FIL, contains all of the errors detected in the input data by the SCENGEN module. Subroutine SERROR in the SCENGEN module opens the error file, writes the header information and all the error messages. A sample error file is shown in Figure 7-1. The actual errors are detected in subroutines SPPOOL, SRWHEEL, SBASE, SCENGI, SCHECK and SFCHECK of the SCENGEN module (See Section 9).

ELDM OUTPUT FILE ELDM.OUT

The results of the economic load dispatch module are written to the DOS ASCII file ELDM.OUT. An example of the ELDM.OUT file is shown in Figure 7-2. The results are printed by scenario; and for each scenario there could be up to three sets of results: one for the prewheeling conditions, one for the conditions including the wheeling transaction and one for the "incremental wheeling transaction". At the beginning of each set is a message written by the ELDM main routine indicating whether or not ELDM was able to find a coherent set of bus generations that satisfied the energy balance for all utilities. A set of results consists of the title of the study, system information, utility data, bus data and line data. As each scenario is processed, ELDM appends its results to the end of the file. Each time WRATES is invoked, the data stored in ELDM.OUT from previous cases are deleted and replaced by the new data; so it is important to copy the contents of ELDM.OUT to another file or to rename ELDM.OUT in case the previous data is to be saved.

LOOP FILE FLOOPONE.OUT

The file FLOOPONE.OUT is a counter file which is primarily used by WRATES to keep track of the last scenario ELDM has processed. The file is created by the SCENGEN module at the beginning of the main program. If a problem with the input data is detected during the execution of the SCENGEN module, FLOOPONE.OUT is deleted. The non-existence of the file causes the batch file, WRATES.BAT, to halt execution of the program. If the input data is error free, SCENGEN writes a "0" in FLOOPONE.OUT. From there on, FLOOPONE.OUT is updated by ELDM and it contains a single line with one number indicating the last scenario processed by ELDM. ELDM is called and the value of the number in FLOOPONE.OUT is incremented by one. When the last scenario is processed, ELDM deletes FLOOPONE.OUT. This signals to the WRATES

Wheeling Rate Evaluation Simulator
WRATES
Module: SCENGEN
Error File

Title of problem appears here.

Errors
------

err 190--- Line 3 belongs to pool 1; no capital
revenue is allowed for this line.

err 350--- In scenario 2
there is more than one hard constraint in pool 1.

err 420---
Wheeling utility 1 is the same as the buying utility.

Figure 7-1. Sample Error File ERROR.FIL

```
##################################################
  WHEELING  RATE  EVALUATION  SIMULATOR --- WRATES

        developed  by  META SYSTEMS INC

                Version  I
              * * * * * * * *

        Economic  Load  Dispatch  Output
##################################################


Title of Study
--------------
  Title of problem appears here.


**********
congratulations, wrates found a set of bus generations
that satisfy the energy balance in all the utilities
after  5 system iterations
**********


Scenario number  1

        number of pools = 1

        number of utilities = 6

        number of buses =  11

        number of lines =  14

          200.00 MW  wheeled from utility 1 to utility 6

------------------------------------------------------------
```

Figure 7-2. Sample ELDM Output File ELDM.OUT

utility data (without wheeling)
scenario 1

pool	number of affiliated utilities	number of buses	number of lines	net energy interchange (exports minus imports) (MW)
1	3	5	8	-400.00

independently dispatched utility	number of buses	number of lines	net energy interchange (exports minus imports) (MW)
1	1	1	1000.00
2	4	5	-100.00
6	1	0	-500.00

bus data (without wheeling)
scenario 1

bus number	pool number	utility number	demand (MW)	generation (MWh)	spot price ($/MWh)	generation costs ($)
6	1	3	0.00	4500.10	77.28	127570.40
8	1	3	4000.00	0.00	85.75	0.00
		3				127570.40
7	1	4	2500.00	3017.23	78.02	220904.50
9	1	4	3000.00	1990.14	84.95	152412.00
		4				373316.40
10	1	5	2200.00	2099.62	86.00	161767.40
		5				161767.40
	1					662654.30
1		1	3000.00	4000.10	35.00	101853.50
		1				101853.50

Figure 7-2 (Continued). Sample ELDM Output File ELDM.OUT

2		2	300.00	1923.14	79.23	124880.60
3		2	1500.00	0.00	82.01	0.00
4		2	1000.00	0.00	87.76	0.00
5		2	700.00	1557.82	85.58	98181.03
						223061.60
		2				
11		6	2000.00	1500.00	123.33	174783.40
						174783.40
		6				

line data (without wheeling)
scenario 1

line number	pool number	utility number	head bus	tail bus	line flow head (MW)	line flow tail (MW)	constraint 1 - soft 2 - hard	flow limit (MW)	losses (MWh)
10	1	3	9	8	490.56	-488.17			2.3
11	1	3	6	8	3791.09	-3599.90			191.1
12	1	3	8	10	88.07	-87.95			0.1
									193.7
		3							
6	1	4	5	7	1259.75	-1244.08			15.6
8	1	4	7	6	-348.05	349.27			1.2
9	1	4	7	9	2109.36	-2019.85			89.5
13	1	4	9	10	519.42	-516.21			3.2
									109.6
		4							
14	1	5	10	11	503.78	-500.00			3.7
									3.7
		5							
									307.1
	1								
1		1	1	2	87.26	-87.17			0.1
									0.1
		1							
2		2	1	3	912.84	-896.47			16.3
3		2	3	2	-1681.54	1710.30			28.7
4		2	3	5	1078.01	-1055.26			22.7
5		2	5	4	653.32	-642.82			10.5
7		2	4	6	-357.18	359.75			2.5
									80.9
		2							

Figure 7-2 (Continued). Sample ELDM Output File ELDM.OUT

batch file that the ELDM portion of the program is complete and the program can call the COMPUTM module.

SCENARIO GENERATION FILE, SCENGEN.DATA

The scenario generation file, SCENGEN.DAT, is a DOS ASCII file which is the input data file to the SCENGEN module. There are two major sections in the file. The first section contains the base case data, which includes the title of the study, the wheeling transaction data, the net energy interchange data, the bus data, and the line data. The second section of the file contains data for each scenario. The scenario data includes the demand at each bus and any other data that differs from the base case.

The file is read by subroutines SRPOOL, SRWHEEL, SBASE and SCENGI of the SCENGEN module. The first three subroutines read the base case section of the file, and subroutine SCENGI reads the scenario data section.

The file SCENGEN.DAT is created by invoking the macro \W, in the LOTUS^{TM} 123 input spreadsheet. The spreadsheet contains tables that allow the user to input data. If SCENGEN.DAT already exists, it is overwritten; if it does not already exist, it is created by the macro.

IWHEEL OUTPUT FILE IWHEEL.OUT

The output of the IWHEEL module is written to the DOS ASCII file IWHEEL.OUT. It contains the final wheeling rate results of WRATES. Three subroutines in the IWHEEL module write to the file: WHEELO, SRATEO and SWANN. Subroutine WHEELO writes the name of the study, the wheeling contract information, the revenue reconciliation data for the utilities, and, if the decomposed option has been selected, the revenue reconciliation data for the lines. Subroutine SRATEO writes the wheeling rates for each scenario. Subroutine SWANN writes the annual rates. IWHEEL.OUT is read by the output macro \0, located in the LOTUS^{TM} 123 spreadsheet TEMPLATE.OUT. The macro \0 transfers the wheeling rate data to LOTUS, sorts them and places them in a table with the appropriate headings.

DURATION CURVE FILE FDURC.FIL

The wheeling rate duration curve file, FDURC.FIL, is a DOS ASCII file written by subroutine SWDURC of the IWHEEL module. It contains the cumulative probabilities of the calculated ideal and reconciled wheeling rates for each utility. This file is read by the LOTUS™ 123 macro \G, contained in the output spreadsheet, TEMPLATE.OUT. The macro reads the data and places it in a table ready for graphing.

Section 8

RUNNING WRATES - BATCH PROCESSING

WRATES.BAT is a batch file that controls the flow of the WRATES program. It is a series of DOS commands created under IBM PC DOS version 3.2. The following requirements must be met for WRATES.BAT to operate properly:

- The four executable modules, SCENGEN.EXE, ELDM.EXE, COMPUTM.EXE and IWHEEL.EXE should be in the same (current) directory as WRATES.BAT.
- LOTUS™ 123 version 2.0 and higher should be accessible from the current directory. This is best accomplished using the DOS path command in the AUTOEXEC.BAT file.
- The LOTUS™ macro files TEMPLATE.IN and TEMPLATE.OUT should also be stored in the same directory as WRATES.BAT.

The batch file WRATES.BAT executes the four WRATES modules as follows:

- The fileS SCENGEN.DAT, ERROR.FIL, FLOOPONE.OUT, ELDM.OUT IWHEEL.OUT, WARNING.FIL are deleted. During its execution WRATES recreates those files and stores in them the data related to the current problem.
- TEMPLATE.IN is copied to the file AUTO123.WK1
- LOTUS™ 123 is invoked. The file AUTO123.WK1 is immediately loaded. It contains an autoexec macro \0, that prompts the user for an input file to edit. After the selected file is loaded into the spreadsheet, control of the program is in the hands of the user. The user can now edit the spreadsheet and restart the program by invoking the macro \W. Control return to the batch program when LOTUS™ 123 is exited. However, the user can choose to end the batch file in one of two ways: either by quitting from LOTUS™ 123 without invoking the macro \W, or, if the macro \W is in progress, the user can press a control break to halt execution of the macro and then quit from LOTUS™ 123.
- The existence of SCENGEN.DAT is checked. If the input

spreadsheet is exited without invoking the macro \W, then SCENGEN.DAT has not been created and WRATES.BAT ends.

- If SCENGEN.DAT exists the SCENGEN module is called. If errors in the input data are detected, then the file FLOOPONE.OUT is not created; otherwise FLOOPONE.OUT is created and a zero is written into it.

- The existence of the file FLOOPONE.OUT is checked. If FLOOPONE.OUT does not exist, the execution of WRATES.BAT is terminated.

- ELDM module is called once for each scenario. As each scenario is processed, the number in FLOOPONE.OUT is incremented by one until all scenarios are run. If an error occurs in the ELDM module for a scenario, it does not affect the results of the previous or the following scenarios, it only affects yearly statistics. After the last scenario is processed, FLOOPONE.OUT is deleted; this signals the batch file to exit the ELDM loop.

- The COMPUTM module is called and executed once.

- The IWHEEL module is called once. The output of the IWHEEL module is placed into the file IWHEEL.OUT.

- TEMPLATE.OUT is copied to the file AUTO123.WK1.

- LOTUS^{TM} 123 is invoked. AUTO123.WK1 is immediately loaded into the spreadsheet. This file contains an autoexec macro \0 that loads the IWHEEL.OUT file into the spreadsheet. The user can inspect the results, print them save them or just exist LOTUS^{TM} 123.

- End of batch file.

Section 9

ERROR AND WARNING MESSAGES

The following errors are detected by WRATES. They are grouped by module. For each error, both the message and the subroutine in which the error is detected are listed; an appropriate response to correct the error is also suggested..

SCENGEN Module Error Messages

These errors are detected in the SCENGEN module and printed in the error file, ERROR.FIL.

error 110: Pool affiliation is specified for utility - and the total number of utilities is only --.
Subroutine: SRPOOL
Response: Check the utility numbers in the pool affiliation table; none should exceed the total number of utilities specified in the title section.

error 115: The pool affiliation for utility -- is specified twice.
Subroutine: SRPOOL
Response: Check the utility numbers in the pool affiliation table; none should be specified twice.

error 120: In the pool affiliation table, the pool number -- appears but the total number of pools is -- .
Subroutine: SRPOOL
Response: Check the pool numbers in the pool affiliation table; none should exceed the number of pools specified in the title section.

error 125 The number of independently dispatched entities-- exceeds the maximum allowed (--).
Subroutine: SRPOOL
Response: Check the pool affiliation table; most probably a pool affiliated utility has not been specified in that table.

error 130: The specified number of pools (--) is larger than the number of pools in the pool affiliation table (--).
Subroutine: SRPOOL
Response: In the pool affiliation table, utilities were assigned to fewer pools than was specified in the title section.

error 135: The number of independently dispatched utilities (--) is different from the number specified in the input data (-).
Subroutine: SRPOOL

Response: Check the pool affiliation table; most probably a pool affiliated utility has not been specified in that table.

error 140: Utility -- belong to pool --; therefore no revenue reconciliation data can be specified for it.
Subroutine: SRWHEEL
Response: Remove that utility from the "Revenue Reconciliation parameters" table.

error 150: In the net energy interchange table, the pool number-- exceeds the total number of pools (--).
Subroutine: SBASE
Response: Check the pool numbers in the net energy interchange table for pools; none should exceed the number of pools specified in the title section.

error 155: In the net energy interchange table, the number of specified utilities -- exceeds the number of independent utilities --.
Subroutine: SBASE
Response: Check the list of utilities in the net energy interchange table for independently dispatched utilities. The list should not be longer than the number of independently dispatched utilities.

error 160: Utility affiliation was only specified for -- of-- buses.
Subroutine: SBASE
Response: In the network and supply curve data section of the input spreadsheet, the utility affiliation should be specified for all of the buses in the first table of the spreadsheet. Check that each of the buses specified in the system is represented in the utility affiliation table.

error 165: Line data was only specified for -- of -- lines.
Subroutine: SBASE
Response: In the network and supply curve data section of the input spreadsheet, line data must be specified for all lines in the system. Check that each of the lines specified in the system is represented in the line data table.

error 170: Utility to Utility wheeling is not allowed within a pool.
Subroutine: SBASE
Response: Check the wheeling data, more specifically the seller type and the buyer type; if they are right, check the number of the buyer and the number of the seller

error 175: Bus to utility wheeling is not allowed within a pool
Subroutine: SBASE
Response: same as error 170

error 180: Utility to bus wheeling is not allowed within a pool
Subroutine: SBASE
Response: same as error 170

error 185: Line -- belong to pool --; no revenue reconciliation is allowed for this line
Subroutine: SBASE

Response: Check the line numbers in the table listing the lines with the decomposed option. Delete the line multiplier specified for lines belonging to the pools.

error 190 Line -- belongs to pool --; no capital revenue is allowed for this line
Subroutine: SBASE
Response: Check the line numbers in the table listing the lines with the decomposed option.

error 220: In scenario __ net energy interchange data is given for utility - which is a pool affiliated utility.
Subroutine: SCENGI
Response: Check the net energy interchange table for independently dispatched utilities in the specified scenario. Also check that there is a row with -1 in its column A at the end of each table for the specified scenario.

error 310: The probability of scenario __ exceeds 100%
Subroutine: SCHECK
Response: In the scenario data section of the input spreadsheet, the probability of each scenario must be between 0 and 100 percent. For the given scenario, check that the second column of the first table is between 0 and 100 percent.

error 320: The probability of scenario __ is less than or equal to zero.
Subroutine: SCHECK
Response: In the scenario data section of the input spreadsheet, the probability of each scenario must not be less than zero. For the given scenario, check that the second column of the first table is between 0 and 100 percent.

error 330: In scenario __ the sum of the net energy interchanges is not equal to zero.
Subroutine: SCHECK
Response: In each scenario, the net energy interchanges must sum to zero to maintain the energy balance. For the given scenario, check that the net energy interchange specified in the base case data and in the scenario data sum to zero.

error 350: In scenario __ there is more than one hard constraint in pool __.
Subroutine: SCHECK
Response: In each scenario, there can only be one hard constraint for each pool. For the given scenario, check that the given pool has at most one hard constraint in both the line data within the scenario and also the line data listed under the network and supply curve data.

error 355: In scenario -- there is more than one hard constraint in utility --
Subroutine: SCHECK
Response: In each scenario, there can only be one hard constraint for each independenly utility. For the given scenario, check that the given utility has at most one hard constraint in both the line data within the scenario and

also the line data listed under the network and supply curve data.

error 360: In scenario __ line __ does not belong to the correct utility.
Subroutine: SCHECK
Response: The utility specified for each line in the line data table must match either the line's beginning bus or the line's end bus. Check the line data for the given line in both the scenario data and the base case data.

error 370: In scenario __ the generation levels on the supply curve of bus __ are not in the right order.
Subroutine: SCHECK
Response: The generation levels in the marginal cost curves must be in increasing order. Check both the scenario data and the base case data to make sure that the marginal cost curve for the given bus exists and that the generation levels are in increasing order.

error 380: In scenario __ the marginal costs on the supply curve of bus __ are not in the right order.
Subroutine: SCHECK
Response: The marginal costs in the marginal cost curves must be in increasing order. Check both the scenario data and the base case data to make sure that the marginal cost curve for the given bus exists and that the marginal costs are in increasing order.

error 385 The slope of the unserved energy for bus -- is less than zero
Subroutine: SCHECK
Response: Check the unserved energy table. No numberin column C can be less than zero

Error 390: The unserved energy cost for bus -- is less than its production costs.
Subroutine: SCHECK
Response: Compare the marginal cost curve specified at that bus to the cost of unserved energy. The cost of unserved energy should be larger than the production cost specified for the last point of the supply curve.

error 410: The sum of the probabilities of all scenarios is different from 100%.
Subroutine: SFCHEK
Response: In the scenario data section of the input spreadsheet, the probability of a scenario occurring appears in the first table of each scenario. If the yearly statistics are required, the sum of the probabilities of all scenarios must equal 100 percent. If the statistics option is equal to 2 in the option table, make sure that the sum of the probabilities of all the scenarios is 100 percent.

error 420: Wheeling utility __ is the same as the buying utility.
Subroutine: SFCHEK
Response: Within the wheeling data, if the buying party is a utility, this utility cannot also be a wheeling utility. Check the wheeling data. If the buyer type in column E

is 2 (i.e. utility), the utility number of the buyer cannot be listed in the second table (wheeling utility data).

error 430: Wheeling utility __ is the same as the selling utility.
Subroutine: SFCHEK
Response: Within the wheeling data, if the selling party is a utility, this utility cannot also be a wheeling utility. Check the first table of the wheeling data. If the seller type in column C is 2 (i.e. utility), the utility number of the seller cannot be listed in the second table (wheeling utility data).

error 440: Wheeling utility __ is not represented in the system.
Subroutine: SFCHEK
Response: In the wheeling data section of the input spreadsheet, the number of each wheeling utility must not be greater than the number of utilities in the system.

error 450: The aggregate option is not allowed with no obligation to serve (class n). Check utility __
Subroutine: SFCHEK
Response: The aggregate option (option 1) for a wheeling utility is not allowed with no obligation to serve (class 2). For the given utility, check columns B and C of the second table in the wheeling data section of the input spreadsheet to see that this condition is met.

error 460: The aggregate option is not allowed with obligation/no obligation to serve (class on). Check utility __
Subroutine: SFCHEK
Response: The aggregate option (option 1) for a wheeling utility is not allowed with obligation/no obligation to serve (class 3). For the given utility, check columns B and C of the second table in the wheeling data section of the input spreadsheet to see that this condition is met.

error 470: The obligation to serve class is allowed only if either the seller or the buyer belongs to the wheeling utility. Check utility --.
Subroutine: SFCHECK

error 475: The number of buses in utility __ is equal to zero.
Subroutine: SFCHEK
Response: In the network and supply curve data section of the input spreadsheet, the second table contains the utility designation for each bus. Check that at least one bus belongs to the given utility.

error 480: The number of utilities in pool -- is equal to zero.
Subroutine: SFCHEK
Response: Check the pool affiliation table

ELDM Error Messages

The following error messages are detected in the ELDM module and are printed in the warning file, WARNING.FIL.

warning/
error 610: Within iteration __ wrates could not find an acceptable value for mu in __ iterations in utility __.

Subroutine: ELDM

Response: This error indicates that ELDM could not find a value for the Lagrange multiplier "mu" such that the flow on the line with a hard constraint in the given utility is equal to the flow limit specified for that line. There are two possible solutions to this problem:

- Increase the maximum number of iterations allowed for the mu loop. This number is stored in the variable NITMMX of the common /CMUE/, and it is initialized in file BLKDATA.FOR of the ELDM module.

- Increase the tolerance on the line flow of the line with a hard constraint. This number is stored in variable TOLLF of the common /CMUE/ and is also initialized in the file BLKDATA.FOR of the ELDM module.

If this error occurs in the last system loop, the line flow on the line with a hard constraint in the specified utility will not be equal to the flow limit on that line, and therefore the dispatch generated does not satisfy all the specified constraints. In that case a "sorry message" is printed in ELDM. However, if this error occurs only within one or more intermediate system loops, the error message is only printed in the warning file.

warning/
error 620: In system loop __ and mu loop __ wrates could not find an acceptable value for gamma in __ iterations for utility __.

Subroutine: ELDM

Response: This error indicates that ELDM could not compute a value of gamma for which the energy balance is met. There are two possible solutions to this problem:

- Increase the maximum number of iterations allowed for the gamma loop. This number is stored in NITGMX of common /CGAMMA/, and it is initialized in file BLKDATA.FOR of the ELDM module. The drawback to this solution is that it increases the execution time of the ELDM module significantly.

- Increase the tolerance on the energy balance. This number is stored in variable TOLENG of common /CGAMMA/ and it is initialized in file BLKDATA.FOR of the ELDM module.

If this error occurs in the last "mu loop" of the last "system loop", then the dispatch generated by ELDM will not satisfy the energy balance within the specified utility. However, if this error occurs within an intermediate loop, it can be disregarded.

warning 630: An initial guess of gamma for utility __ could not be made in __ iterations.

Subroutine: SGAMMA

Response: This warning is generated if ELDM cannot calculate an initial guess for gamma for which the generation is equal the demand plus the net energy interchange of the given utility, with the losses set to zero. The program uses the value of gamma calculated during the last iteration.

error 640: In system loop __ and mu loop __ and gamma loop __ wrates could not find a consistent set of spot prices for utility __.

Subroutine: SPTMAX

Response: This error is fatal, and when it occur ELDM is terminated. It indicates that ELDM could not calculate a value of gamma such that the spot price at all buses of the specified utility converge to a final value. There are two possible solutions to this problem:

- o Increase the maximum number of iterations allowed for the "spot price and generation loop." This number is stored in the variable NITSMX of the common /CINJ/, and is initialized in file BLKDATA.FOR of the ELDM module. This adjustment to resolve the problem is likely to increase the execution time of ELDM substantially.

- o Increase the tolerance on the injection at each bus. This number is stored in variable TOLINJ of the common /CINJ/ and is also initialized in the file BLKDATA.FOR of the ELDM module.

Other Messages

The following three messages are written by the ELDM module and appear in the ELDM.OUT file. One of these messages appears at the beginning of each set of load dispatch results.

Message: ? ? ? ? ? ? ? ? ? ?
SORRY wrates could not find a set of bus generations that satisfy the energy balance in all the utilities after __ iterations.
The following results do not constitute an acceptable solution
? ? ? ? ? ? ? ? ? ?

Subroutine: ELDM

Response: This message is printed when ELDM cannot find a coherent set of generations at all buses of the system. When this occurs, ELDM prints the results of the last system iteration. The user might find a clue to the problem by examining the printed output in ELDM.OUT.

Message: ? ? ? ? ? ? ? ? ? ?
SORRY wrates could not find a set of bus generations that satisfy the line constraints in all the utilities.
? ? ? ? ? ? ? ? ? ?

Subroutine: ELDM

Response: May be there is no way to bypass the line constraints.

Message: **********
congratulations, wrates found a set of bus generations that satisfy the energy balance in all the utilities after __ system iterations
****
Subroutine: ELDM
Response: Most probably, there are no errors.

Appendix A

DESCRIPTION OF DISKETTES

This package contains two "high density" 5 1/4 inch diskettes labelled "WRATES - Ver 1 Source Code" and "WRATES - Ver 1 Executable Modules"

WRATES, SOURCE CODE

This diskette has four (4) directories, one for each module: \SCENGEN, \ELDM, \COMPUTM, and \IWHEEL. Each directory contains:

- Common blocks (COMMON.FOR);
- Sub-programs (XXXX.FOR);
- A command file to compile all the sub-programs of the module (COMPIL.BAT);
- A response file to create the executable module (RESPONSE.FIL).

WRATES, EXECUTABLE MODULES

WRATES II has one (1) directory. It contains:

- Four executable modules:
  SCENGEN.EXE
  ELDM.EXE
  COMPUTM.EXE
  IWHEEL.EXE
- A command file WRATES.BAT that directs the sequential execution of the separate modules that compose WRATES;
- A LOTUS™ 123 autoexec file TEMPLATE.IN that directs the automatic execution of certain macros when the input worksheet is to be prepared;
- A LOTUS™ 123 autoexec file TEMPLATE.OUT that directs the automatic execution of certain macros when the output is to be sorted and displayed in a LOTUS™ 123 environment;
- A sample input worksheet - SCENARIO.WK1;

- A sample output worksheet - RATES.WK1;
- A sample ELDM output file - ELDM.OUT.

Appendix B

INSTALLATION PROCEDURE

The WRATES program is supplied on two high density diskettes as described in Appendix A. The first diskette, has the source code and need not be installed to run WRATES. The second diskette contains all the executable modules and files necessary to run WRATES and must be installed properly for the program to run.

INSTALLING THE SOURCE CODE

The Source Code diskette has four directories, corresponding to the four modules: SCENGEN, ELDM, COMPUTM and IWHEEL. The best way to install the source code is to store each module in its own directory as it is provided on the diskette. Note that there are files in different directories with the same name, and therefore it is unwise to combine the directories. The steps outlined below describe the procedure to install SCENGEN; however this applies to all four modules of WRATES.

- Get into the root directory on the hard disk (often it is the C disk) by typing "CD\ (return)."
- Make a new SCENGEN directory by typing "MD SCENGEN (return)".
- Change to the SCENGEN directory on the hard drive by typing "CD\ SCENGEN (return)".
- Insert the SOURCE CODE diskette into the high density floppy drive.
- Log onto the floppy drive by typing "A: (return)".
- Change to the SCENGEN directory on the floppy diskette by typing CD\SCENGEN (return).
- Copy all of the files in the SCENGEN directory on the floppy drive to the SCENGEN directory on the hard drive by typing "COPY *.* C: (return)".

The installation of the other modules is the same. To facilitate editing and re-compiling of the program, your program editor, the DOS LINK command, and the FORTRAN compiler should be accessible from the

newly created directories on the hard disk. The COMPIL.BAT file in each of the directories is set up to compile all of the modules using the Lahey 77L compiler; if you use another compiler, COMPIL.BAT has to be modified. The RESPONSE.FIL is used with the DOS LINK command to create the executable modules from the object files and may also have to be modified if another FORTRAN compiler is used.

INSTALLING THE EXECUTABLE MODULE

The entire contents of this diskette must be transferred to the hard disk for WRATES to run correctly. The procedure for installing the files is described below:

- Get into the root directory on the hard disk by typing "CD\ (return)".
- Make a new WRATES directory by typing "MD WRATES (return)".
- Change to the WRATES directory on the hard drive by typing "CD WRATES (return)".
- Insert the Executable Modules disk into the high density floppy drive.
- Log onto the floppy drive by typing "A: (return)".
- Copy all of the files on the floppy drive to the WRATES directory on the hard drive by typing *'COPY *.* C: (return)".
- Modify the path command in the AUTOEXEC.BAT file of the root directory of the hard disk so that LOTUS™ 123 can be accessed from the WRATES directory. (For more information on the PATH command and the AUTOEXEC.BAT file, please refer to your DOS manual.)
- Re-boot the machine before you run WRATES for the first time.

This completes the installation of the WRATES program. To begin your first WRATES session, make sure you are in the WRATES directory and simply type WRATES. And remember, no program works from the first time.