AGENCY FOR INTERNATIONAL DEVELOPMENT
WASHINGTON, D. C. 20523

BIBLIOGRAPHIC INPUT SHEET

FOR AID USE ONLY
Batch 63

1. SUBJECT CLASSIFICATION

A. PRIMARY

TEMPORARY

B. SECONDARY

2. TITLE AND SUBTITLE
User's guide to SYSOPT: an interactive system optimization computer program

3. AUTHOR(S)
Buchner,Marcus

4. DOCUMENT DATE	5. NUMBER OF PAGES	6. ARC NUMBER
1976	254p.	ARC

7. REFERENCE ORGANIZATION NAME AND ADDRESS
Mich.State

8. SUPPLEMENTARY NOTES *(Sponsoring Organization, Publishers, Availability)*
(In Computer Library for Agr.Systems Simulation. CLASS doc.no.12)

9. ABSTRACT
(LIBRARY & INFO.SCIENCE R&D)

10. CONTROL NUMBER
PN-AAE-015

11. PRICE OF DOCUMENT

12. DESCRIPTORS
Data processing
Decision theory
Manuals
Models
Simulation
SYSOPT?

13. PROJECT NUMBER

14. CONTRACT NUMBER
CSD-2975 Res.

15. TYPE OF DOCUMENT

AID 590-1 (4-74)

USER'S GUIDE TO SYSOPT:
AN INTERACTIVE SYSTEM OPTIMIZATION COMPUTER PROGRAM

Marcus Buchner

COMPUTER LIBRARY FOR AGRICULTURAL SYSTEM SIMULATION

CLASS Document No. 12

Agricultural Sector Analysis and Simulation Projects
Supported under Contract No. AID/csd-2975
U. S. Agency for International Development

Department of Agricultural Economics
Michigan State University
East Lansing, Michigan

October 1976

TABLE OF CONTENTS

	Page
LIST OF TABLES	v
LIST OF FIGURES	v
PREFACE	ix
I. INTRODUCTION	1
II. SYSOPT CAPABILITIES AND OPERATION	5
General Description	6
SYSOPT Interaction and Output	12
Pattern Recognition Sub-Component:	14
Interaction/Output Flow Chart	14
Interaction/Output Explanations	21
Optimization Sub-Component:	27
Interaction/Output Flow Chart	27
Interaction/Output Explanations	44
Examples of Interactive Use of SYSOPT	60
Pattern Recognition Sub-Components	62
Optimization Sub-Component	68
III. PROGRAM DOCUMENTATION	81
SYSOPT Simulation Programming	82
User Supplied Simulation Subroutines SIM, MMM and NNN	93
Variable Dimensioning	98
Main Program and Subroutine Descriptions	100
Pattern Recognition Sub-Component	101
Program MAINPR	101
Subroutine SUBPR	104
Subroutine SMOOTH	114
Subroutine PLOT	120
Subroutine PLY	125
Subroutine PLEDGE	127

Page

Optimization Sub-Component 129

Program MAINOPT 129
Subroutine SUBOPT 132
Subroutines DATA1, DATA2, DATA3, DATA4 ... 143
Subroutine VAR 158
Subroutine FUNC 160
Subroutine GLOBE 167
Subroutine CONV 169
Subroutine TRANSFE 171
Subroutine SENS 173
Subroutines BOX, CONSX, CENTR, CHECK, and CONST 178
Subroutine COMP 192
Subroutine BOTM 194
Function NEXT 201
Subroutine TRANS 203
Subroutine SPEED 206
Subroutines SEQREG, SEQSW, SWITCH 213

Subroutine Control Flow Charts 221

SYSOPT Use of Input-Output Tapes 225

Alphabetical Glossary of Component Variables 232

REFERENCES 245

LIST OF TABLE

Page

III-1. Time Series Values for Example Problem 226

LIST OF FIGURES

II-1. Structure Connections to Component 8

II-2. Key to Flow Chart Symbols 13

II-3. Pattern Recognition Sub-Component: Interaction/ Output Flow Chart 14

II-4. Pattern Recognition Sub-Component: Interaction/ Output Explanations 21

II-5. Optimization Sub-Component: Interaction/Output Flow Chart 27

II-6. Optimization Sub-Component: Interaction/Output Explanations 44

II-7. Pattern Recognition Sub-Component 62

II-8. Optimization Recognition Sub-Component 68

III-1. Adaption of SIMEX1 Simulation 83

III-2. An Example of a SIMEX1 Simulation Constructed to Interact with SYSOPT 88

III-3. Subprogram Control Flow Chart: Pattern Recognition Sub-Component 222

III-4. Subprogram Control Flow Chart: Optimization Sub-Component 223

COMPUTER LIBRARY FOR AGRICULTURAL SYSTEMS SIMULATION

The Computer Library for Agricultural Systems Simulation (CLASS) is one of the four major activities of the Agricultural Sector Analysis and Simulation Projects at Michigan State Unviersity under U. S. Agency for International Development Contract AID/csd-2975. The other three major interrelated project activities include theoretical and methodological research, the Development Analysis Study Program, and field activities, primarily in the Republic of Korea.

The project objective is to develop an approach to institutionalizing an analytical capacity for planning, policy formulation, program development, and project implementation for agricultural sector development within the public decision making structure of developing countries. A major component of the analytical capacity is a series of system simulation models tailored to the needs of the individual country. Much of the experience gained from the field activity and the knowledge gained from the theoretical and methodological research added to the present stock of knowledge about building and maintaining analytical capacities for agricultural sector development can be preserved and extended in the training provided through the Development Analysis Study Program and in the stock of model, component, and utility routine computer software documented in the Computer Library for Agricultural Systems Simulation.

In full operation, the Computer Library for Agricultural Systems Simulation (CLASS) acquires, catalogs, maintains and distributes computer programs and associated documentation. These computer programs are of generalized simulation models, components, and routines designed specifically for the analysis of agricultural development problems and processes. In particular, the library sets standards of admissibility for programs and documentation; catalogs and indexes programs and documentation so as to facilitate their retrieval by users seeking a set of programs to be used in a specific problem analysis; and distributes programs and documentation to users.

To enhance the effectiveness of the library, its functions also include identifying and soliciting needed models; actively bringing programs and documentation up to the library's standards; and providing limited consultation in identifying and implementing appropriate library programs for a particular application. A subsidiary function of the library in conjunction with the identification and solicitation of models is to survey and catalog ongoing research in agricultural systems modeling and simulation.

The CLASS document publication series is the main vehicle for informing potential users of the substance of CLASS holdings and activities.

July, 1976

George E. Rossmiller
Director
Agricultural Sector Analysis and
Simulation Projects

PREFACE

This guide documents the FORTRAN program SYSOPT, an interactive system optimization component, which can be used to assist in solving parameter estimation or policy decision problems in large complex simulation models. The program was developed as a result of research involving the design of an efficient component for use in the analysis of socio-economic models. The original research, presented in my Ph.D. dissertation [2] concentrated on a literature review of the field of optimization techniques and components, a detailed statement of the problems faced in parameter estimation or policy decision making with Complex Models, a theoretical description of the program SYSOPT, and a discussion of test results. This guide provides an introductory description of SYSOPT in Part I, a comprehensive discussion of the capabilities and operation of SYSOPT in Part II along with illustrative examples, and a detailed subroutine by subroutine listing and description in Part III.

As with the use of any complex program, a certain amount of background knowledge is necessary in order to understand the operation and capabilities of SYSOPT. In this regard, SYSOPT users can be divided into three categories: decision-makers, programmers, and system scientists. As a minimal background, decision-makers should become familiar with the material presented in Parts I and II of this guide. In addition, a programmer, who is familiar with SYSOPT execution on the particular computer installation the decision-maker is using, should be available

for consultation. In the case of a programmer using SYSOPT, a careful reading of this entire guide, paying particular emphasis to Part III, and my dissertation should provide a sufficient background to adapt the program for particular computers and to make minor programming changes. Finally, a systems scientist, in order to make major alterations in SYSOPT to adapt additional pattern recognition or optimization techniques for inclusion in the program should be thoroughly familiar with both this guide, and my dissertation as well as optimization literature in terms of algorithm and component development.

Preparing a document of this size and complexity requires the assistance of many people. I would particularly like to thank Dr. Tom Manetsch for his direction and guidance and Dr. Tom Carroll for his suggestions and careful editing. The typing of the manuscript was accomplished through the efforts of Ms. Judy Pardee, Ms. Edith Nosow, and Ms. Kyong Soo Kim. Their efforts are greatly appreciated. Funding for the development and documentation of SYSOPT was obtained from the Agricultural Sector Analysis and Simulation Projects which is supported under Contract AID/csd-2975 of the U. S. Agency for International Development.

I. INTRODUCTION

SYSOPT is a computer program that can be used to assist in the solution of parameter estimation or policy decision problems resulting from the use of complex simulation models. The use of large, nonlinear, and dynamic models to represent real world phenomena has been largely due to the digital computer's capability to yield numerical solutions for complex differential and difference equations. Because analytical solutions of parameter estimation or policy decision problems with these models are often extremely difficult, if not impossible, to obtain, the use of numerical optimization techniques have been employed to produce approximate solutions.

SYSOPT is an interactive computer program, or component, that contains a "pattern recognition" section and an "optimization" section to assist in the efficient examination of either parameter or policy variable spaces. The component incorporates human interaction with graphical behavioral study and a four level numerical optimization algorithm to locate optimal parameter estimates and/or policy decisions. SYSOPT was developed by reference to an efficiency measure depending on four factors (1) component costs resulting from the use of the computer, (2) the ease with which interaction with the component can be achieved, (3) the difference between component solutions and true problem solutions, and (4) compatibility of the component with different computers. Particular emphasis in the development was placed upon the problem solution accuracy, the number of simulation runs necessary to achieve convergence to the solutions, the interactive capability to change search structure and parameters during the component execution, and computer memory and central processor use.

In the component, the pattern recognition section has the primary capability of examining user specified model behavior, while the optimization section has the capability of performing efficient numerical optimizations. Both sections are programmed to be FORTRAN compatible. Thus while each subcomponent is built to allow interaction with a general form of simulation model, the FORTRAN compability allows execution on a variety of computers.

In particular, the pattern recognition subcomponent allows an examination of model behavior in either a printed or graphical form. In addition, comparisons between model outputs, real world data, and smoothed real world data can be made with the component. Therefore this section of SYSOPT will assist in the solution of parameter estimation and policy decision problems in three principle ways: (1) through rapid isolation of regions in the input variable space representing invalid or insensitive behavior, (2) through rapid isolation of regions in the variable space needing detailed investigation, and (3) through assistance in the determination of the validity of the model.

On the other hand, the optimization subcomponent contains an efficient four-level interactive optimization algorithm. In the algorithm, the Complex method and Powell's method, both direct search numerical optimization methods, are integrated into a search technique that is effective both far from and in the vicinity of a local optimum. In addition, methods have been incorporated into the algorithm to perform global optimum searches and to insure compatibility between the Complex method and Powell's method. Thus information gained from using one level of the algorithm is used to initialize another level, and the convergence rate of the entire component is increased.

In summary, SYSOPT can assist in the solution of problems often encountered in a systems approach such as model validation, model tuning, and model control. Although the straightforward application of the automatic optimization search techniques in the component can lead to incorrect or illogical solutions to parameter estimation or policy decision problems, the use of the component together with proper human interactions can greatly assist in obtaining useful models and useful results from models.

Part II of this report contains a general non-technical description of the capabilities of SYSOPT and its operation and capabilities. This part provides the background on SYSOPT needed by the analyst who either is responsible for tuning the model for further use or who is responsible for using the model to make policy decisions. It addresses problems that need to be solved in order to properly execute the program on a computer, and thus, includes discussions of basic simulation adaptation, the meaning of output/interrogatives in the program, examples of the use of the program, and computer input tape usage.

Part III contains the technical program documentation needed by the person who either wishes to have a detailed understanding of SYSOPT execution or who desires to adapt the program to fit particular needs. For the programmer, detailed descriptions of all the subroutines functions and variables in the program, a subroutine control flow chart, a discussion of how to adapt a general simulation model to correctly interact with SYSOPT, and an explanation of the variable dimensioning capability of the program, are all included in the third section of this guide. It is hoped that the descriptions, explanations, and discussions of SYSOPT use and programming will allow the efficient solution of parameter estimation and policy decision problems of interest to the reader.

II. SYSOPT CAPABILITIES AND OPERATION

This section presents a general, non-technical, description of the SYSOPT optimization component. This description consists of a non-mathematical discussion of the overall structure of the component and its relationship to the other elements of a problem investigation such as the simulation, the real world, and the user. In addition the internal structure and organization of the component is briefly examined.

General Description

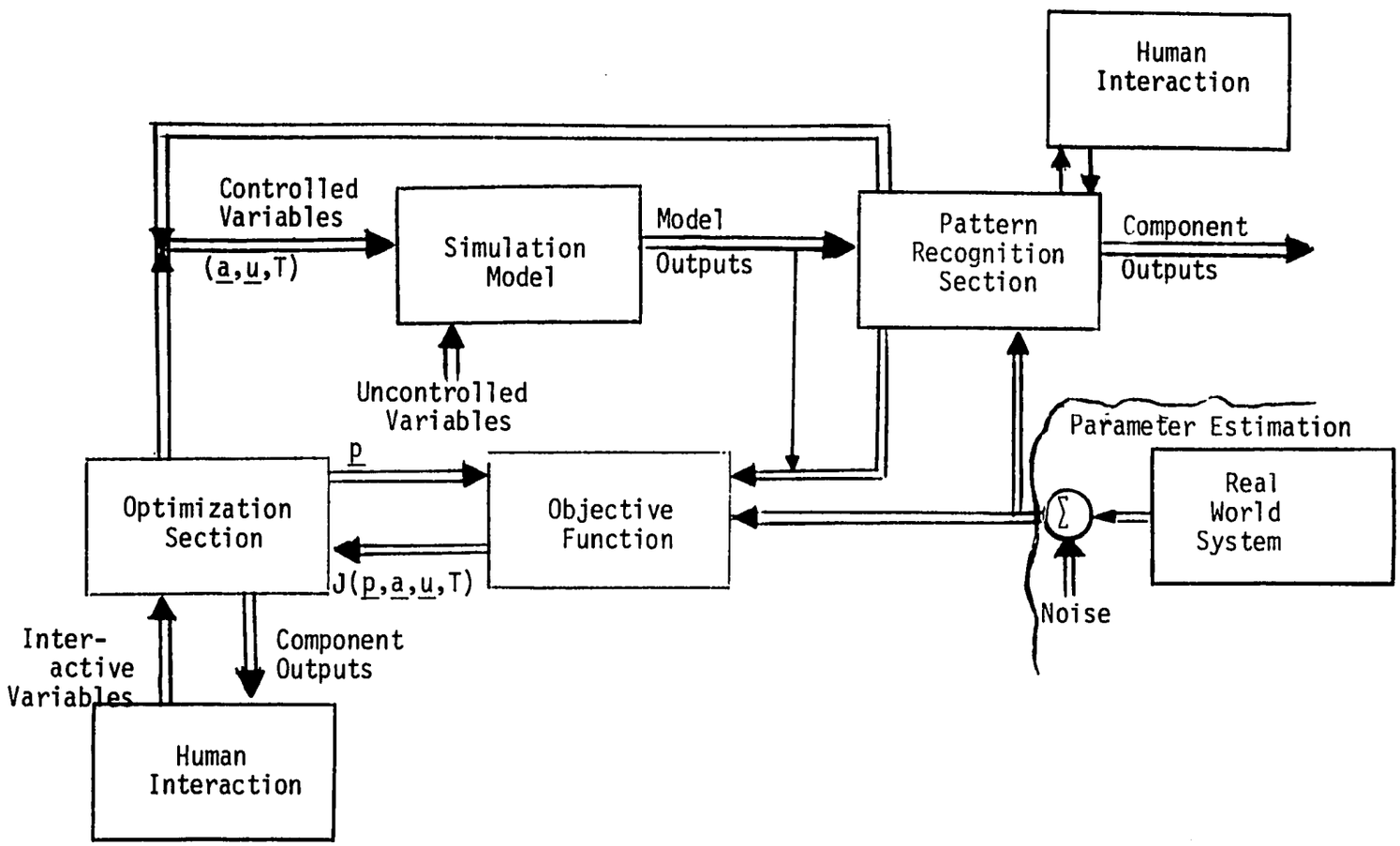
The behavior of a model can often be studied or improved by printing or plotting real world or model time series and by solving a sequence of user specified numerical optimization problems. With the aid of simulation, the optimization component described in this guide can be used to examine model behavior or solve complex model generated problems. The user of the component, either the model builder or policy analyst, can thus be assisted to efficiently solve parameter estimation or policy decision problems. To accomplish this the component is constructed in two sections: (1) a pattern recognition subcomponent and (2) an optimization subcomponent.

The purpose of the pattern recognition subcomponent, as its name implies, is to determine if the model exhibits reasonable behavioral characteristics and also to select regions of the input variable space for future investigation in the optimization section of the component. Thus, patterns of behavior of the model are used to aid in solving the parameter estimation and policy decision problems. This subcomponent is designed to allow the examination of the behavior of a simulation in a batch or an interactive mode of execution on a computer. Because accuracy of the problem solution can sometimes be improved in a

parameter estimation problem if filtering techniques are employed, this subcomponent incorporates data-smoothing algorithms. In addition, it is important to be able to study the model outputs for different sets of values of the input variables. Therefore, data smoothing of real world time series and printing and/or plotting of model time series are two of the major capabilities incorporated in this section of the component.

The optimization subcomponent is designed to be executed only in an interactive mode with the user and with several possible levels of interaction. The different levels of interaction are included to allow the user to select an appropriate format for the study of the model. This section of the component also contains features useful in solving many of the problems associated with nonlinear static optimizations of complex functions. The objective function, search variable sets, optimization method parameters, and constraints on the variables can all be changed during the execution of the component. In addition to being able to direct various optimizations to be performed, the user can also study the sensitivity of the objective function to changes in the input variable values. The purpose of this subcomponent is to solve numerical optimization problems that will lead to good parameter estimates or policy decisions.

The flow chart in Figure II-1 illustrates the data flows between the simulation model, the objective function, the real world, the user, and the optimization component. It is important to realize that only one subcomponent is in execution at a given time. In general, one would first investigate model behavior characteristics with the pattern recognition subcomponent and then use the optimization subcomponent to obtain a problem solution. However, in many situations an iterative process is necessary



p is the parameter set that specifies the objective function J

a is the set of model parameters being studied

u is the model control variables being studied

T is the length of the time horizon of the model [0,T]

FIGURE II-1. STRUCTURAL CONNECTIONS TO COMPONENT

to obtain a good problem solution where the pattern recognition and optimization sections of the component are each called into execution a number of times.

As a result of the availability of the CDC 6500 computer for the development of SYSOPT, certain features of the use and operation of the component are particular to this computer. However with relatively small programming changes, (see programmer's section) the component should be executable on a wide variety of computers. The component can be stored in the computer on a magnetic tape or disk. It is only necessary to have the simulation to calculate the model outputs, and two user supplied subroutines to specify the form of the objective function stored in the computer in order for the component to execute properly. Once this is done the execution of the entire component can be co-ordinated from a teletype terminal with any large plots or printout from the pattern recognition section being disposed to a batch printer. The design of the component is general enough so that no programming changes need to be made to change the investigation from one simulation to another. An example of the operation of the component from a teletype and a guide to its use is provided in the last section of Part II.

At this point a description of the component's overall organization and the various subroutines included in it will enable a user to have an understanding of the program execution. As previously discussed, the component is programmed in two parts (1) the pattern recognition subcomponent and (2) the optimization subcomponent. Both subcomponents represent complete computer programs capable of independent execution. The pattern recognition section consists of a main program with three subroutines while the optimization section contains a main program and fifteen subroutines. Together they occupy 12,000 decimal

words of core storage in the computer during execution.

The pattern recognition subcomponent is constructed to give maximum flexibility to the user of the component in examining model behavior. This part of the component contains a main program, MAINPR, and three subroutines, SUBPR, SMOOTH, and PLOT. The main program inputs key variables that affect the size of arrays needed in the remainder of the program. In addition, a method to control variable dimensioning of arrays is included in MAINPR so that core storage is minimized during the execution of the program.

SUBPR, the heart of the subcomponent, contains the majority of input-output functions, data transfers, interactive decisions, and logic execution in the program. SMOOTH is a subroutine designed to remove random fluctuations from the real world time series with a first- or second-order fixed-memory polynomial filter. Finally, the subroutine PLOT performs the function of plotting, with automatic scaling, or printing either model outputs or real world time series as functions of time.

The capabilities available to the user which can be obtained by fixing the values of certain variables in the program include:

(1) Smoothing of real world time series with a first- or second-order polynomial filter.

(2) Printing or plotting all or a subset of the model outputs for any specified set of values of the controlled variables. This can be done for different time intervals, as specified by the user.

(3) Execution of the simulation in a normal mode (see section on user-supplied subroutines).

The optimization subcomponent, containing the algorithms for solving optimization problems in an interactive mode, is the foundation of the entire component. While the pattern recognition subcomponent is a necessary part of the overall process of efficient parameter estimation and

policy decision making, it is only a convenient means of examining model behavior and selecting areas of the variable space for further investigation. In contrast to this, the optimization subcomponent can perform efficient numerical optimizations critical to solving parameter estimation and policy decision problems of complex models.

This section of the component requires approximately two-thirds of the core storage space or 7,000 decimal words, necessary for execution of the complete component. In addition, although restricted operation of the component is possible in a batch mode with a well defined problem, execution of this subcomponent is intended to be directed from a teletype terminal with a high priority of entrance into the computer.

Specific capabilities of the optimization subcomponent which the user may choose include:

(1) Sensitivity testing on the objective function with respect to all, or a user specified subset, of the input variables.

(2) A four-level numerical optimization algorithm consisting of a global search to examine the input variable space for multiple local optima; a coarse search, the Complex method, for examining relatively broad areas of the input variable space; a fine search, Powell's method, to examine regions of the input variable space in the vicinity of a local optimum; and a method to locate a global optimum along a very sharp ridge a valley in the input variable space.

(3) A compatibility algorithm to insure that the information gained about the input variable objective function space from using the Complex method will be used to initialize Powell's method and thus increase the speed of convergence of the component.

(4) The ability to easily, through interaction on a teletype, change the objective function, search variable sets, optimization algorithm parameters, etc, and

(5) The capability to compute local or global polynomial approximations to the objective function over regions of the input variable space.

With a familiarity of the operation and use of SYSOPT as provided in the remainder of this guide, the component can be used to provide numerical

approximations to the solution of parameter estimation and policy decision problems in complex simulation models.

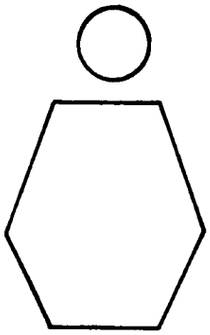
SYSOPT Interaction and Output

In order to the use the SYSOPT program to investigate system behavior or system optimization, a knowledge of the expected interactions and outputs of the problem is necessary. With the detailed explanations and flow charts of SYSOPT behavior presented in Figures II-3, II-4, II-5, and II-6, a user should sufficiently understand the meaning of all possible internal decisions, interactive inputs, and data outputs for both the pattern recognition and optimization subcomponents for proper execution of the program.

Figure II-2 presents a key to the symbols used in the interaction/output flow charts. Included in this section is a complete explanation of each interrogative and output statement occurring during program execution along with annotated examples, Figures II-7 and II-8. These explanations should serve to guide a user through program execution for the first several attempts at using SYSOPT for solving parameter estimation or policy decision problems.

FIGURE II-2

KEY TO FLOW CHART SYMBOLS



connector

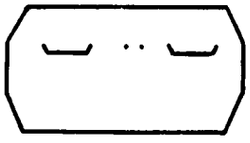
internal decision to the program

INTERACTIVE INTERROGATIVE
(REPLY FROM USER NECESSARY)

xxx is the SYSOPT subroutine from which the interogative is printed

in INTERACTIVE INTERROGATIVE BOX denotes that the INTERROGATIVE MAY BE REPEATED

connecting two or more flow chart boxes denotes a sequence of boxes that may be repeated



OUTPUT from SYSOPT
(NO REPLY FROM USER NECESSARY)

denotes specific alphanumeric output data

FIGURE II-3

PATTERN RECOGNITION SUB-COMPONENT:
INTERACTION/OUTPUT FLOW CHART

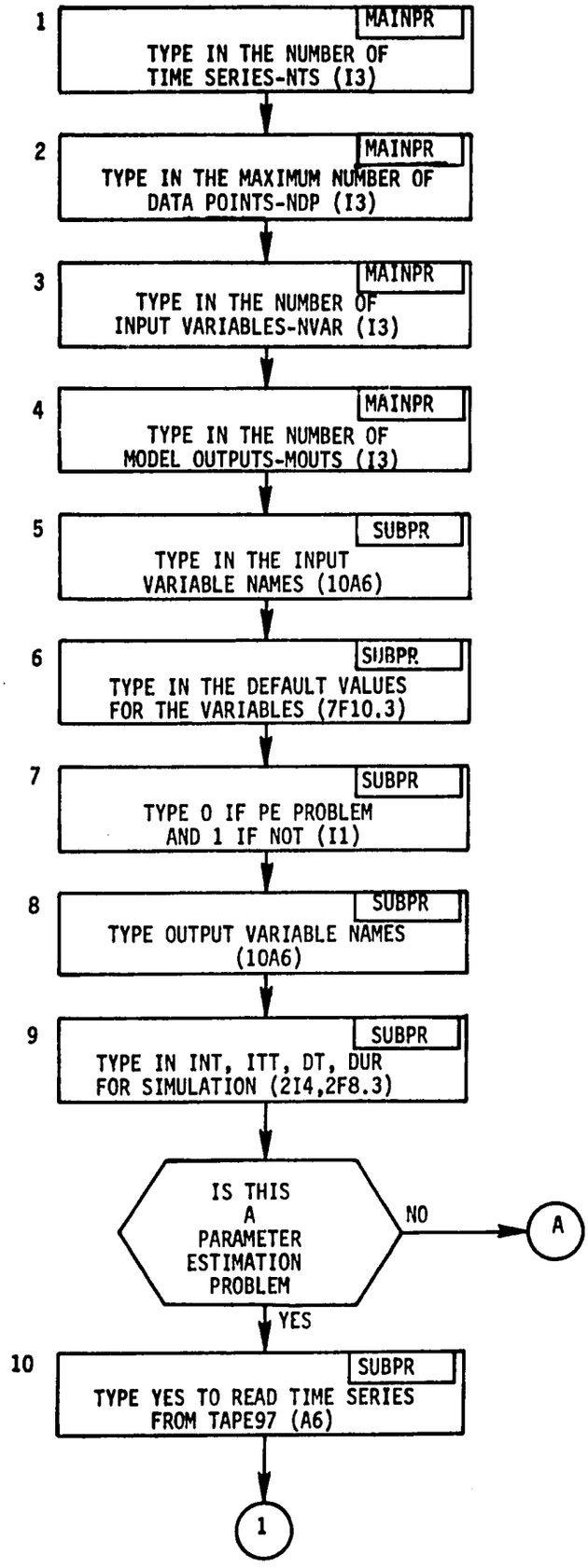


Figure II-3 (cont'd)

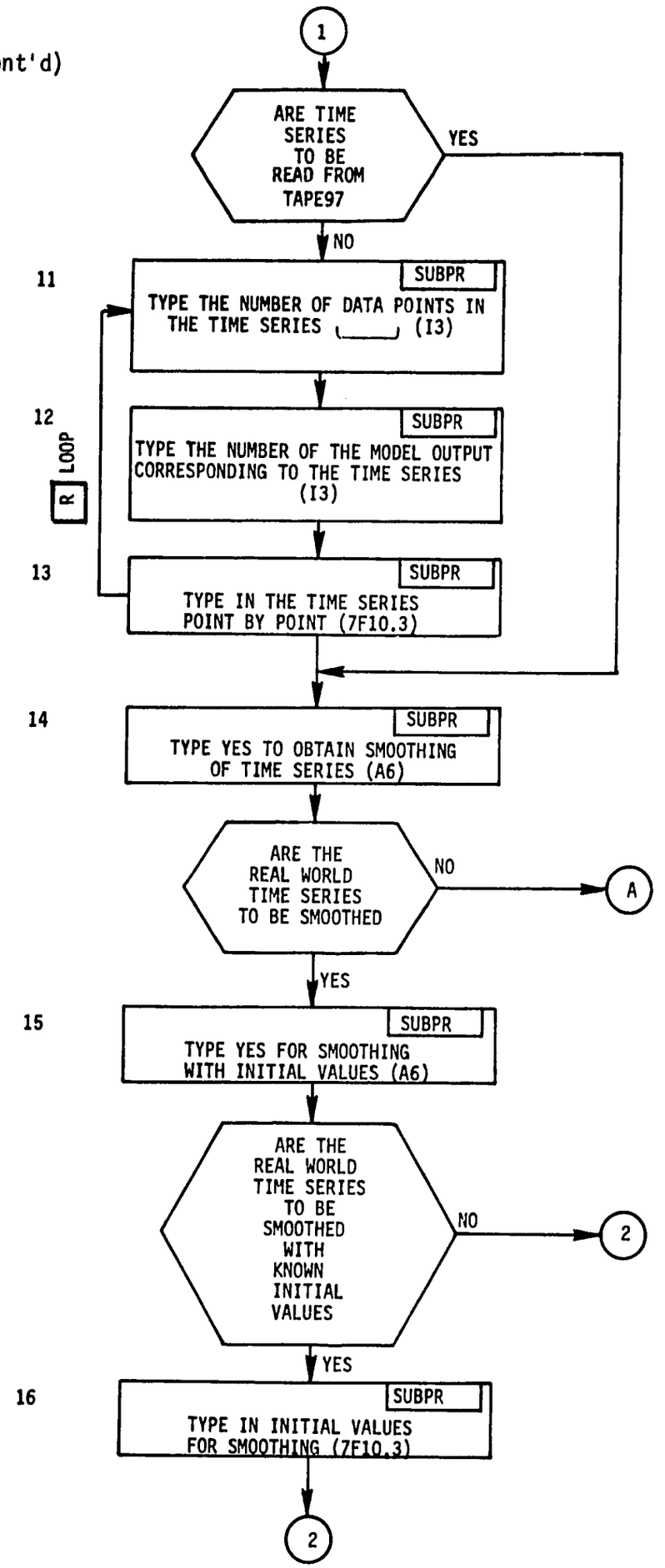


Figure II-3 (cont'd)

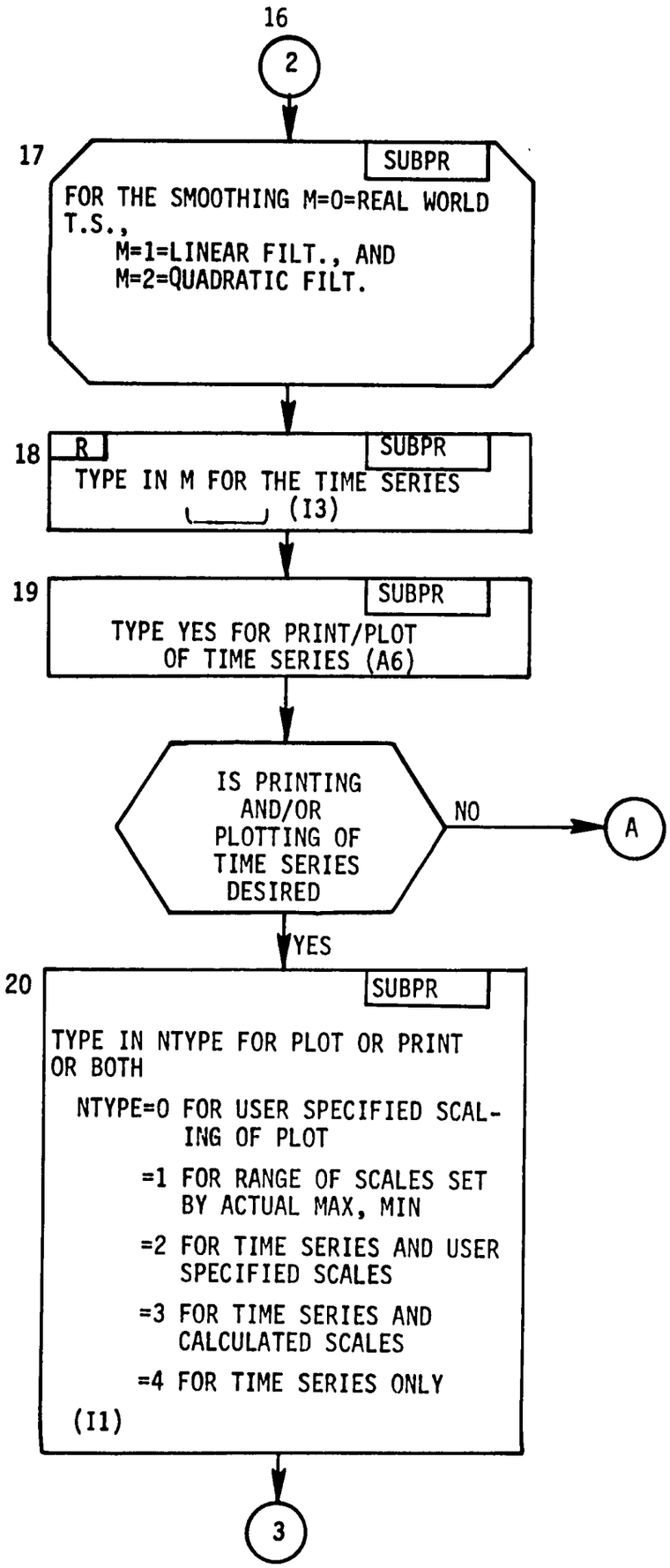


Figure II-3 (cont'd)

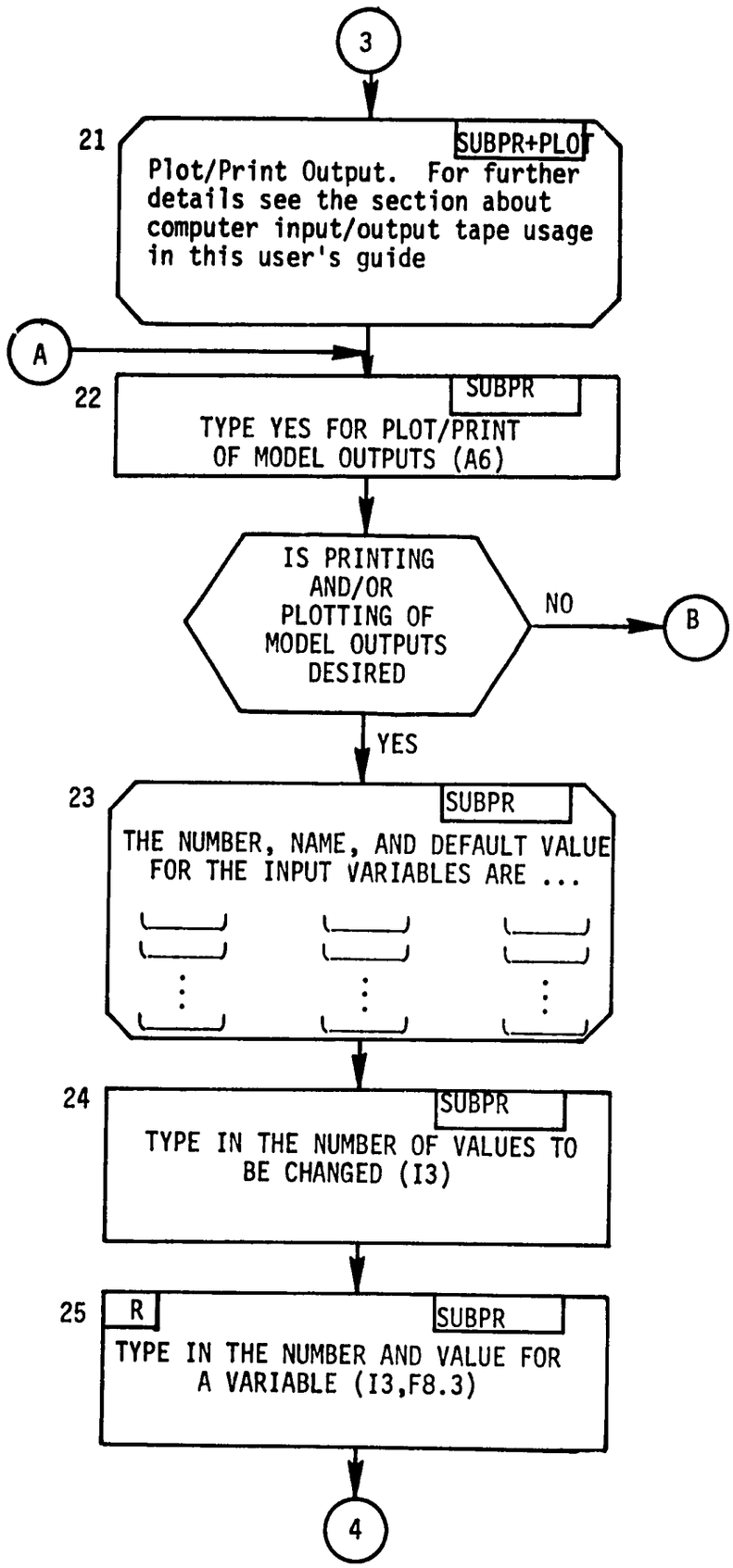


Figure II-3 (cont'd)

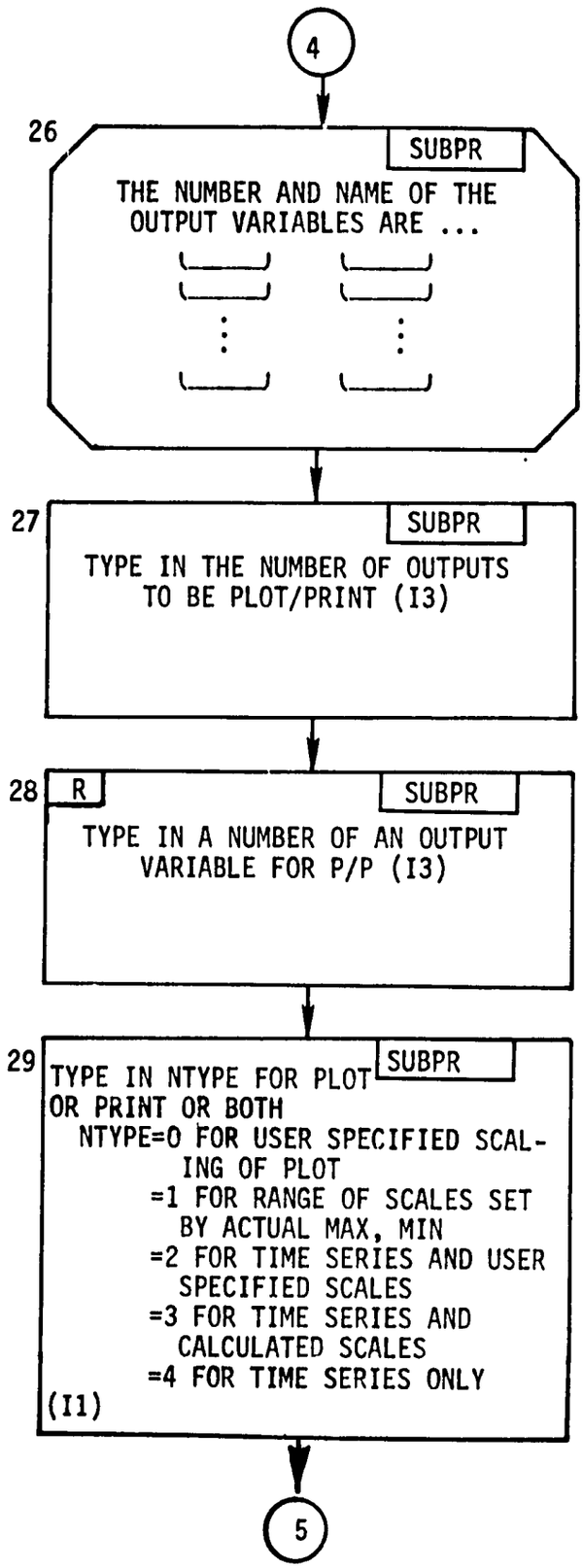


Figure II-3 (cont'd)

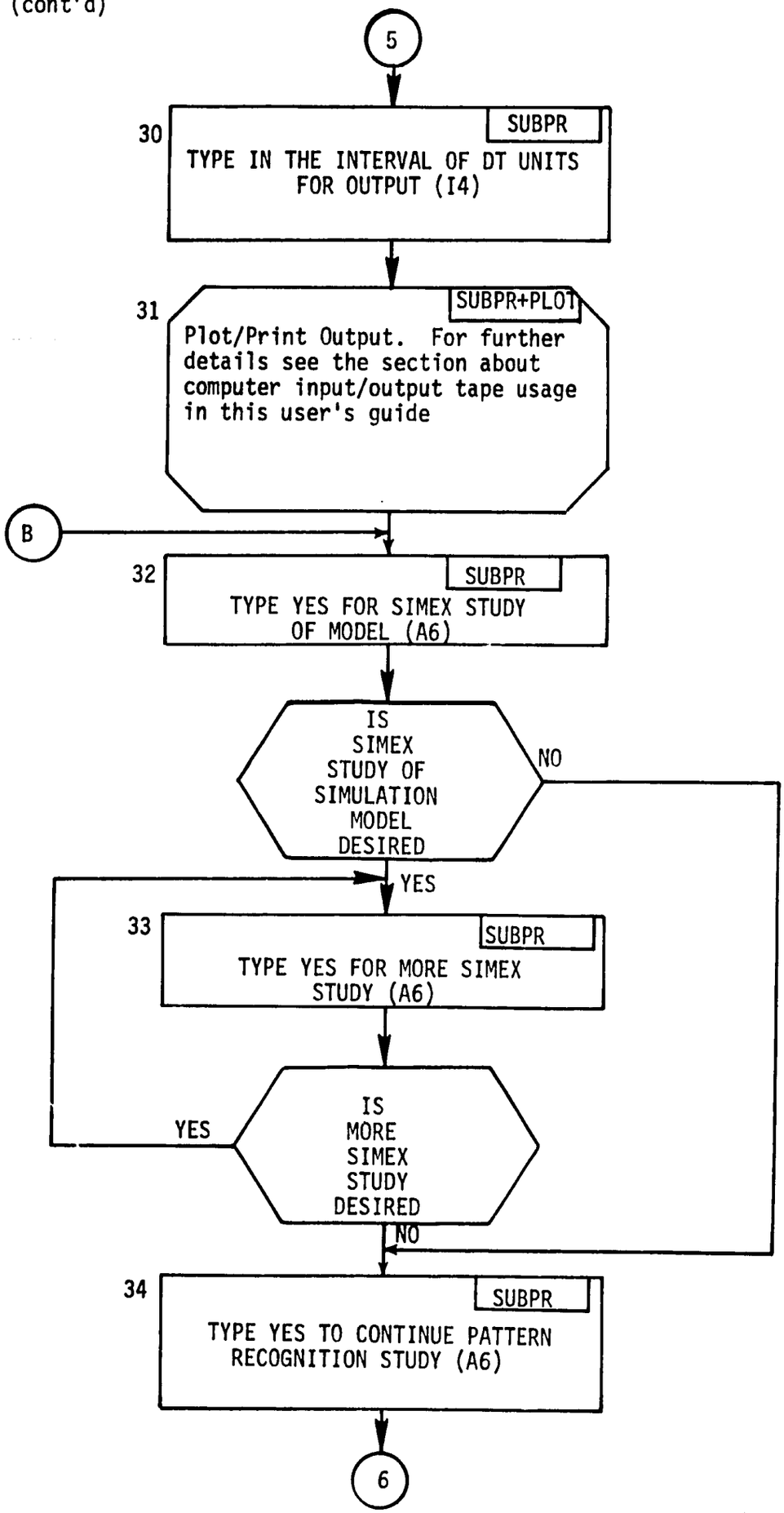


Figure II-3 (cont'd)

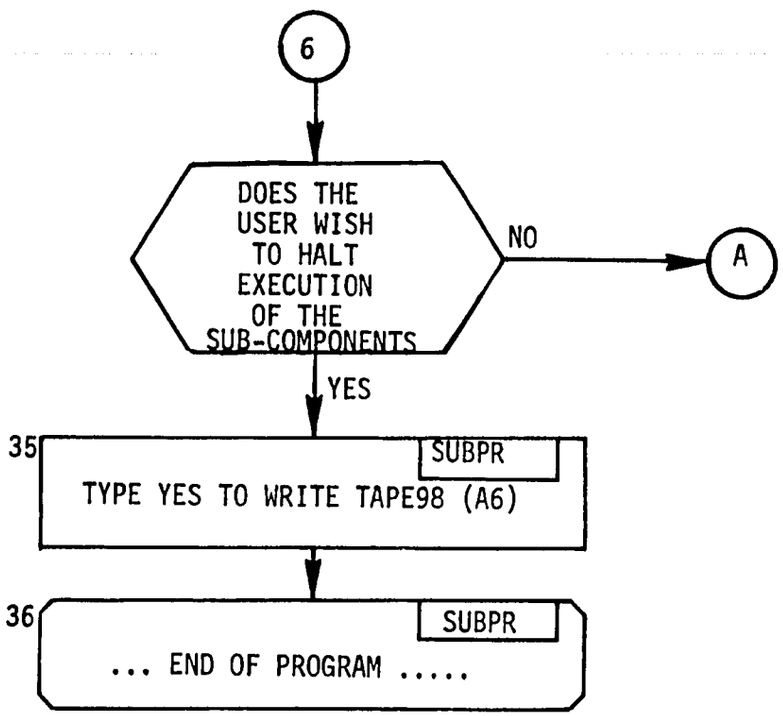


FIGURE II-4

Pattern Recognition Sub-Component: Interaction/Output Explanations

Box #

1 "TYPE IN THE NUMBER OF TIME SERIES-NTS (I3)"

NTS is the number of time series (or the number of variables represented in the real world data set) used in the calculation of the least squares error criterion between the real world data and the model outputs. In an optimal control problem, set NTS=MOUTS.

2 "TYPE IN THE MAXIMUM NUMBER OF DATA POINTS-NDP (I3)"

NDP is the maximum number of data points in the time series. For an optimal control problem, set NDP=1.

3 "TYPE IN THE NUMBER OF INPUT VARIABLES-NVAR (I3)"

NVAR is the number of input variables in the simulation model:
XIN(1),XIN(2), ..., XIN(NVAR)

4 "TYPE IN THE NUMBER OF MODEL OUTPUTS-MOUTS (I3)"

MOUTS is the number of model outputs in the simulation model:
XOUT(1),XOUT(2), ..., XOUT(MOUTS)

5 "TYPE IN THE INPUT VARIABLE NAMES (10A6)"

All input variables in the simulation model are referred to by their names in the pattern recognition sub-component. These names must correspond to the order that the input variables are defined in the model. Note also that the names must be input ten to a live on the teletype.

6 "TYPE IN THE DEFAULT VALUES FOR THE VARIABLES (7F10.3)"

This calls for the default or base values for the input variables. Again the input must correspond to the order that the variables were defined in the simulation model. As desired, the user can change these values, during component execution.

7 "TYPE Ø IF PE PROBLEM AND 1 IF NOT (I1)"

Type 0 if a parameter estimation (PE) problem is being investigated with the component and 1 if an optimal control (OC) problem, or policy decision problem, is being investigated with the component.

Box #

8 "TYPE OUTPUT VARIABLE NAMES (10A6)"

All output variables in the simulation model are referred to by their names in the sub-component. These names must correspond to the order that the output variables are defined in the model. Note that the names must be input ten to a live on the teletype.

9 "TYPE IN INT, ITT, DT, DUR FOR SIMULATION (2I4, 2F8.3)"

Type, in I4 format, INT, the number of printer spaces between points on the plots of the time series. This variables spacing capability on plots was incorporated in SYSOPT to allow simplified comparison of time series plots and model output plots which may have different sampling intervals; in I4 format, ITT, the number of DT time units desired between model outputs calculated in the simulation; in F8.3 format, DT, the basic time step for the simulation model; and in F8.3 format, DUR, the length of the time horizon of the simulation.

10 "TYPE YES TO READ TIME SERIES FROM TAPE97 (A6)"

If the local file TAPE97 was constructed according to the instructions outlined in the section on SYSOPT use of computer tapes, then typing YES in response to this question causes input to the component of the values of the time series.

11 "TYPE THE NUMBER OF DATA POINTS IN THE TIME SERIES _____ (I3)"

(PE problem only). Input the number of data points in the indicated time series. The Jth time series need not correspond to the Jth output variable of the simulation.

12 "TYPE THE NUMBER OF THE MODEL OUTPUT CORRESPONDING TO THE TIME SERIES (I3)"

Type in the number of the model output, as defined by the order specified in response to question 8, to which question 11 refers.

13 "TYPE IN THE TIME SERIES POINT BY POINT (7F10.3)"

For the time series corresponding to the previous enquiries in questions 11 and 12, input the data points, seven to a line, in the indicated format.

Figure II-4 (cont'd)

Box #

14 "TYPE YES TO OBTAIN SMOOTHING OF TIME SERIES (A6)"

If the user desires to smooth any of the time series within either a linear or quadratic fixed memory polynomial fitter, type YES. If there is a significant zero-mean uncorrelated noise component in the time series, and if the sampling interval is small relative to the expected changes in the time series data points, then smoothing of the time series probably yields improved parameter estimates.

15 "TYPE YES FOR SMOOTHING WITH INITIAL VALUES (A6)"

If the initial values of the time series are known and can be relied upon as being accurate, then type YES, as the smoothing algorithm places particular weight upon known initial values.

16 "TYPE IN INITIAL VALUES FOR SMOOTHING (7F10.3)"

Type, in the specified format, the initial values of the time series. The order of input must correspond to the order that the time series were input into the component and associated with the model outputs. Note that the initial values should be input seven to a line of teletype.

17 "FOR THE SMOOTHING M=Ø=REAL WORLD T.S.,
M=1=LINEAR FILT., AND
M=2=QUADRATIC FILT."

This is computer output providing the user with a list of options available for the following interrogative: M=1 causes the smoothing of the specified time series with a linear fitter, M=2 causes the smoothing of the specified time series with a quadratic fitter.

18 "TYPE IN M FOR THE TIME SERIES ____ (I3)"

Type in the desired value of M as determined from the available values defined by the output in box 17.

19 "TYPE YES FOR PRINT/PLOT OF TIME SERIES (A6)"

If the user desires to have a printout a plot of the actual real world and, if calculated with the component, the smoothed real world time series, then type YES in response to this question. If only model outputs are to be investigated, then type NO.

Figure II-4 (cont'd)

Box #

20 "TYPE IN NTYPE FOR PLOT OR PRINT OR BOTH
NTYPE = 0 FOR USER SPECIFIED SCALING OF PLOT
= 1 FOR RANGE OF SCALES SET BY ACTUAL MAX, MIN
= 2 FOR TIME SERIES AND USER SPECIFIED SCALES
= 3 FOR TIME SERIES AND CALCULATED SCALES
= 4 FOR TIME SERIES ONLY
(I1)"

Type 1, 3, or 4: 1 for plot only of time series with scales of plot set by minimum and maximum values of the data, 3 for plot and printout of time series with scales of plot set by minimum and maximum values of the data, and 4 for printout of time series values only. This is for the actual real world time series and, if calculated, with the component, the smoothed real world time series.

21 - Plot/Print Output - for further information see the PLOT subroutine description in the programmer's section of this guide and the computer input/output section in the user's section of this guide.

22 "TYPE YES FOR PLOT/PRINT OF MODEL OUTPUTS (A6)"

If the user desires to have a printout or plot of the model outputs, or subsets of the model outputs, then type YES in response to this question.

23 "THE NUMBER, NAME, AND DEFAULT VALUE FOR THE INPUT VARIABLES ARE ...

_____	_____	_____
⋮	⋮	⋮
_____	_____	_____

This output indicates the current values of the input variables.

24 "TYPE IN THE NUMBER OF VALUES TO BE CHANGED (I3)"

If the user wishes to change any of the input variable default values, then type in the number of values to be changed. If no changes need be changed, then type "ØØØ."

25 "TYPE IN THE NUMBER AND VALUE FOR A VARIABLE (I3, F8.3)"

This interrogative must be answered for each variable whose default value is to be changed. The number of the input variable must correspond to the list output in box 23.

Figure II-4 (cont'd)

Box #

26 "THE NUMBER AND NAME OF THE OUTPUT VARIABLES ARE ..."

This is a list of the output variable names and their corresponding numbers for answering the following two interrogatives.

27 "TYPE IN THE NUMBER OF OUTPUTS TO BE PLOT/PRINT (I3)"

The response to this question specifies the n-mber of model outputs, contained in the list defined by box 26, for which printout and/or plots are desired.

28 "TYPE IN A NUMBER OF AN OUTPUT VARIABLE FOR P/P (I3)"

This interrogative must be answered for each model output for which a print/plot (P/P) is desired. The number of the output variable must correspond to the proper output variable name as indicated in the output box 26.

29 "TYPE IN TYPE FOR PLOT OR PRINT OR BOTH
NTYPE = 0 FOR USER SPECIFIED SCALING OF PLOT
= 1 FOR RANGE OF SCALES SET BY ACTUAL MAX, MIN
= 2 FOR TIME SERIES AND USER SPECIFIED SCALES
= 3 FOR TIME SERIES AND CALCULATED SCALES
= 4 FOR TIME SERIES ONLY
(I1)"

Type 1, 3, or 4: 1 for plot only of model outputs with scales of plot set by minimum and maximum values of the output data, 3 for plot and printout of model outputs with scales of plot set by minimum and maximum values of the output data, and 4 for printout of model output values only.

30 "TYPE IN THE INTERVAL OF DT UNITS FOR OUTPUT (I4)"

This is the variable ITT defined in response to question 9. Note that this allows this variable to be changed during component execution.

31 - Plot/Print Output - for further information see the PLOT subroutine description in the programmer's section of this guide and the computer input/output section in the user's section of this guide.

Figure II-4 (cont'd)

Box #

32 "TYPE YES FOR SIMEX STUDY OF MODEL (A6)"

Type YES for regular study and execution of the simulation model.

33 "TYPE YES FOR MORE SIMEX STUDY (A6)"

If the user wishes to execute the simulation a number of times, this allows new data sets to be input to the model.

34 "TYPE YES TO CONTINUE PATTERN RECOGNITION STUDY (A6)"

If the user wishes to halt execution of the pattern recognition sub-component, then NO should be input in response to this question. If YES is the response then execution of the program resumes with interrogative 22.

35 "TYPE YES TO WRITE TAPE98 (A6)"

TAPE98 is a computer local file that will contain, if a YES is input in response to this question, data to be transferred from the pattern recognition sub-component to the optimization sub-component. The data consists of:

1. whether the problem being investigated is a parameter estimation or optimal control problem,
2. the number of input variables,
3. the number of input variables,
4. the number of output variables,
5. the names of the output variables, and
6. the time series values if a parameter estimation problem is being studies.

36 "... END OF PROGRAM ..."

This indicates that the program has halted execution.

FIGURE II-5

OPTIMIZATION SUB-COMPONENT:
INTERACTION/OUTPUT FLOW CHART

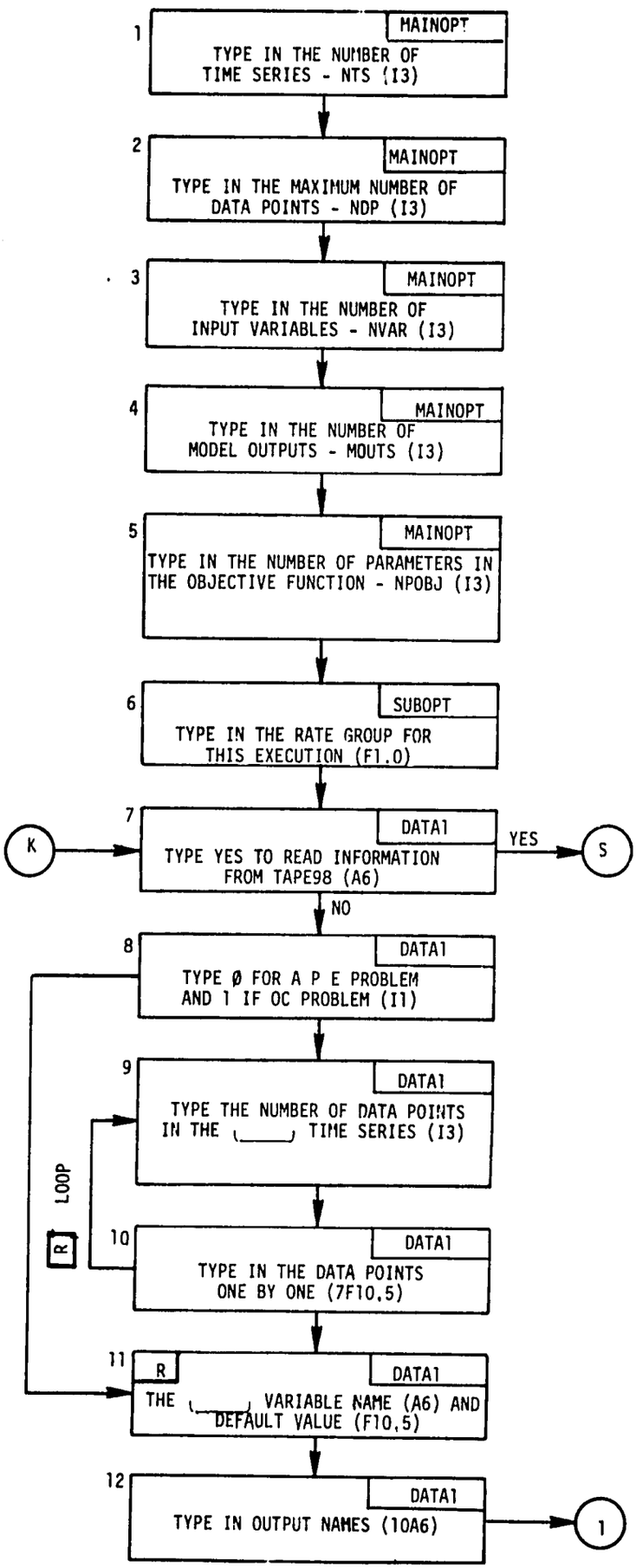


Figure II-5 (cont'd)

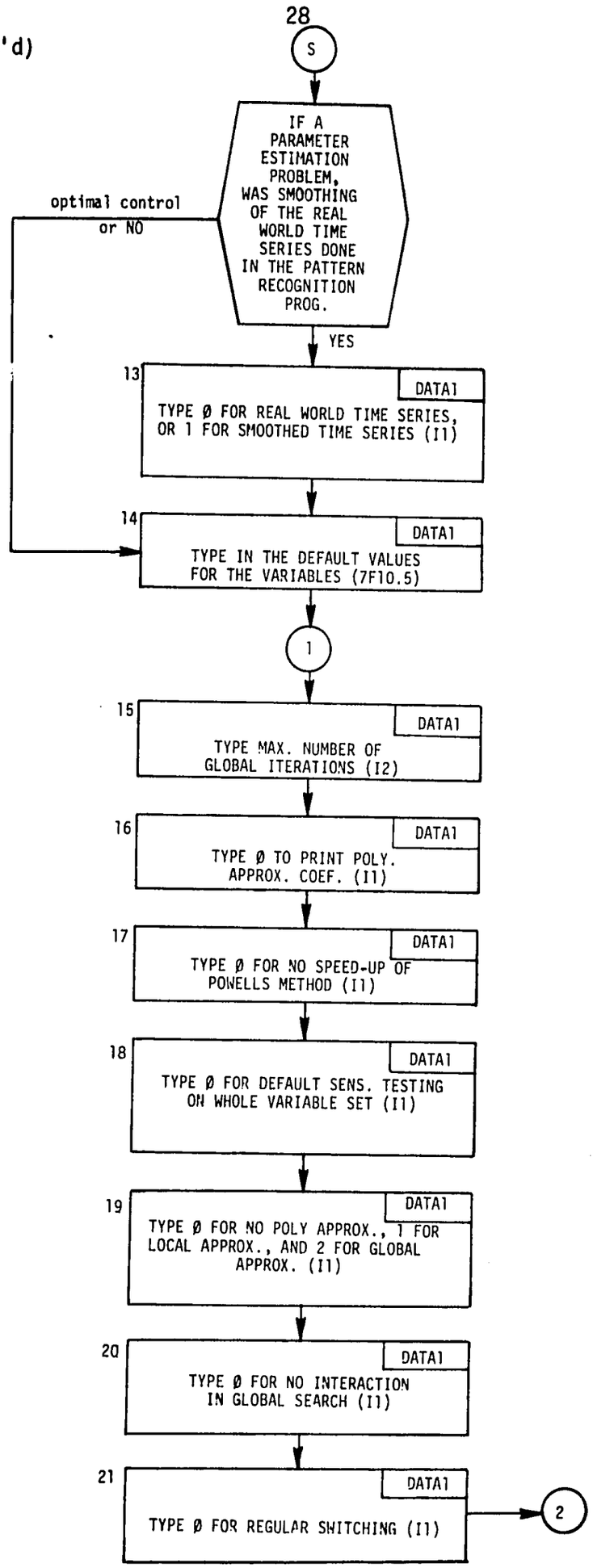


Figure II-5 (cont'd)

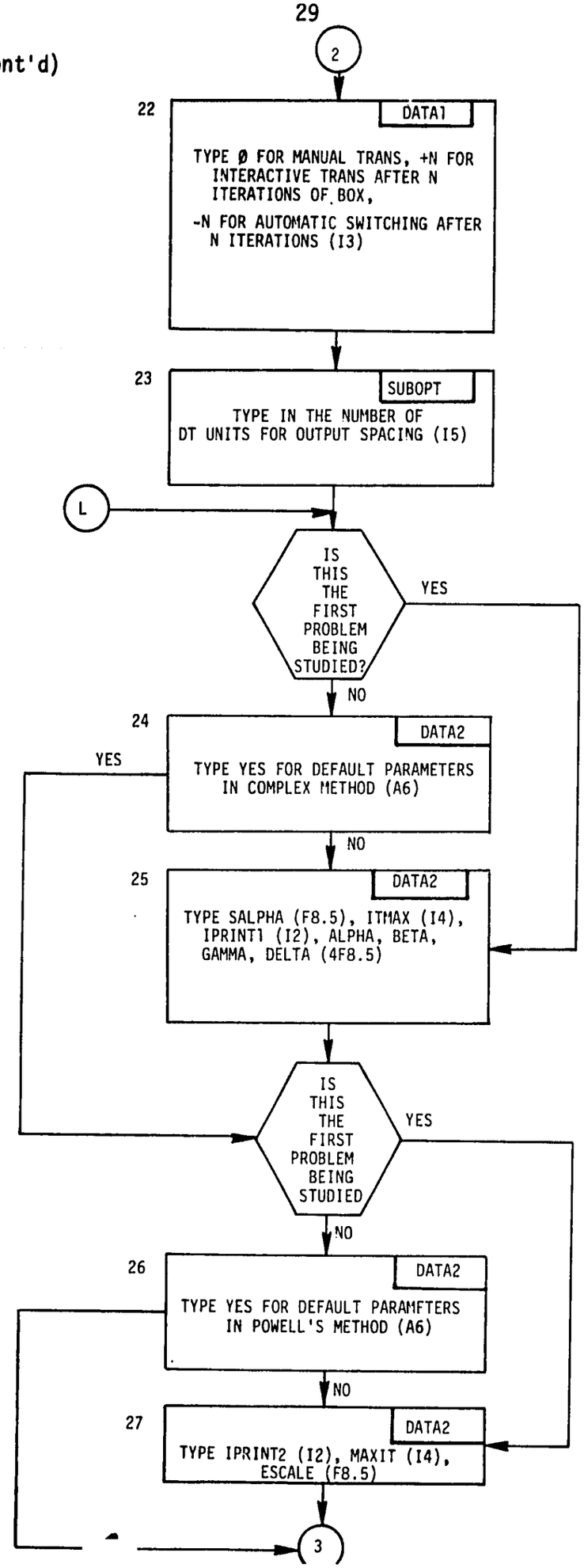


Figure II-5 (cont'd)

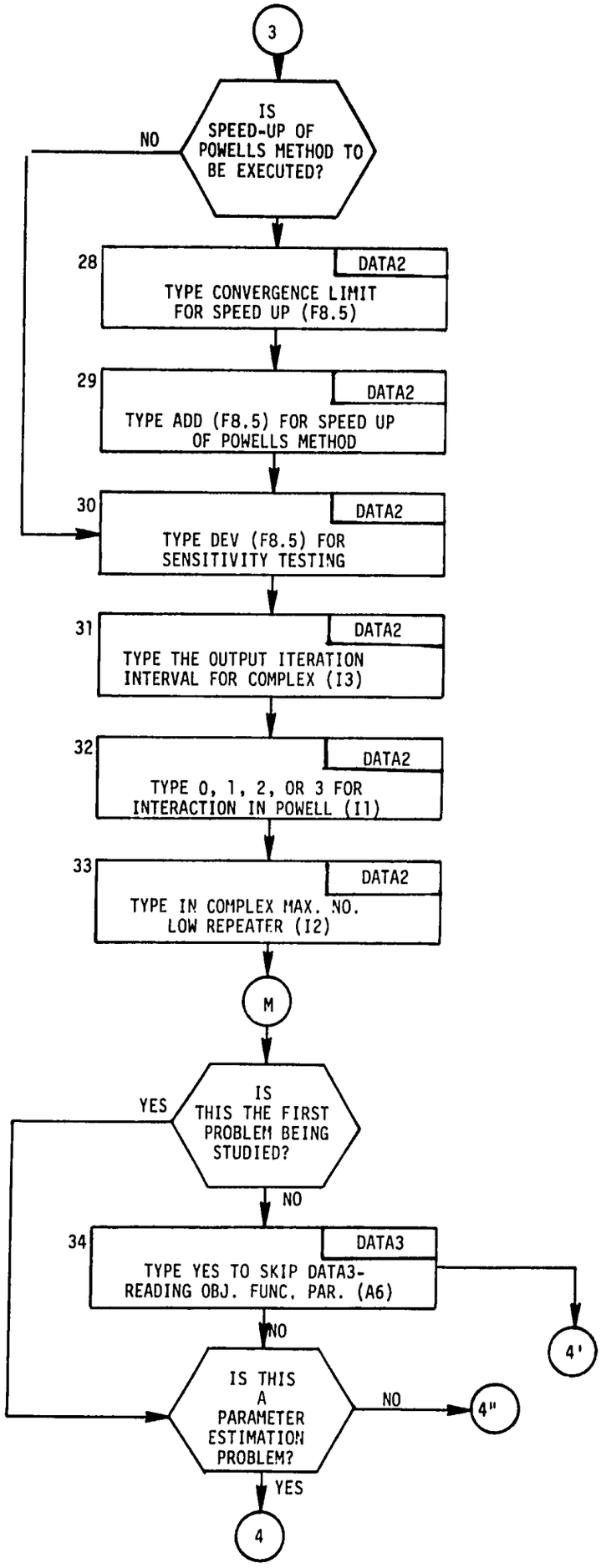


Figure II-5 (cont'd)

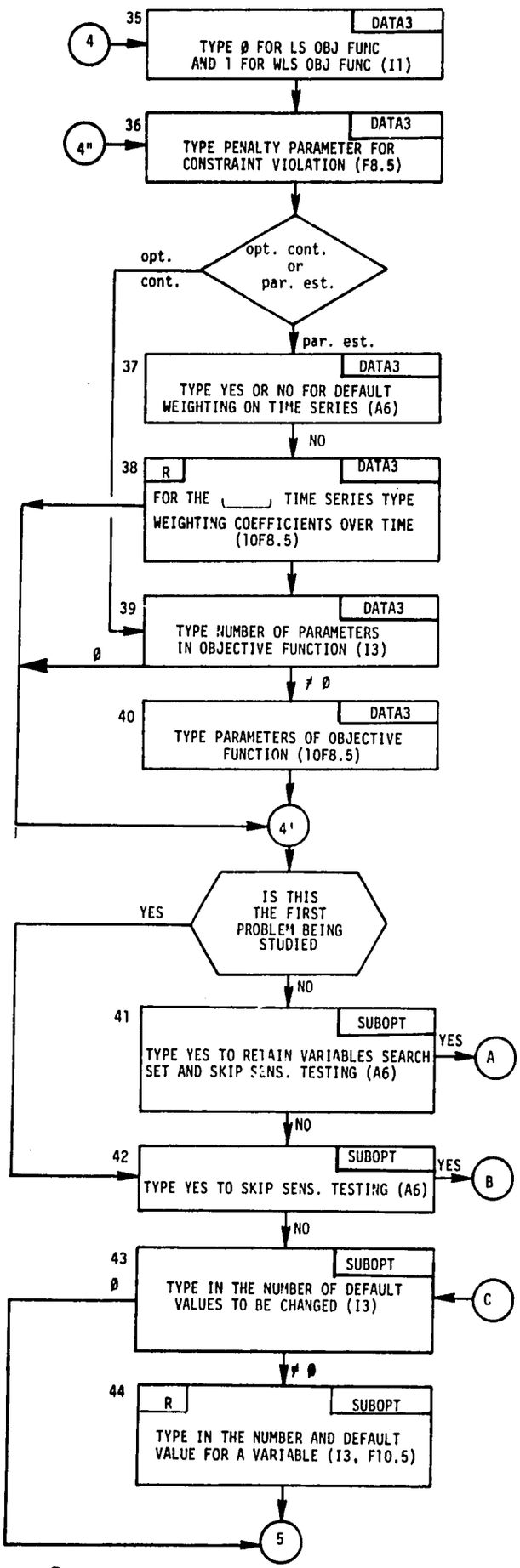


Figure II-5 (cont'd)

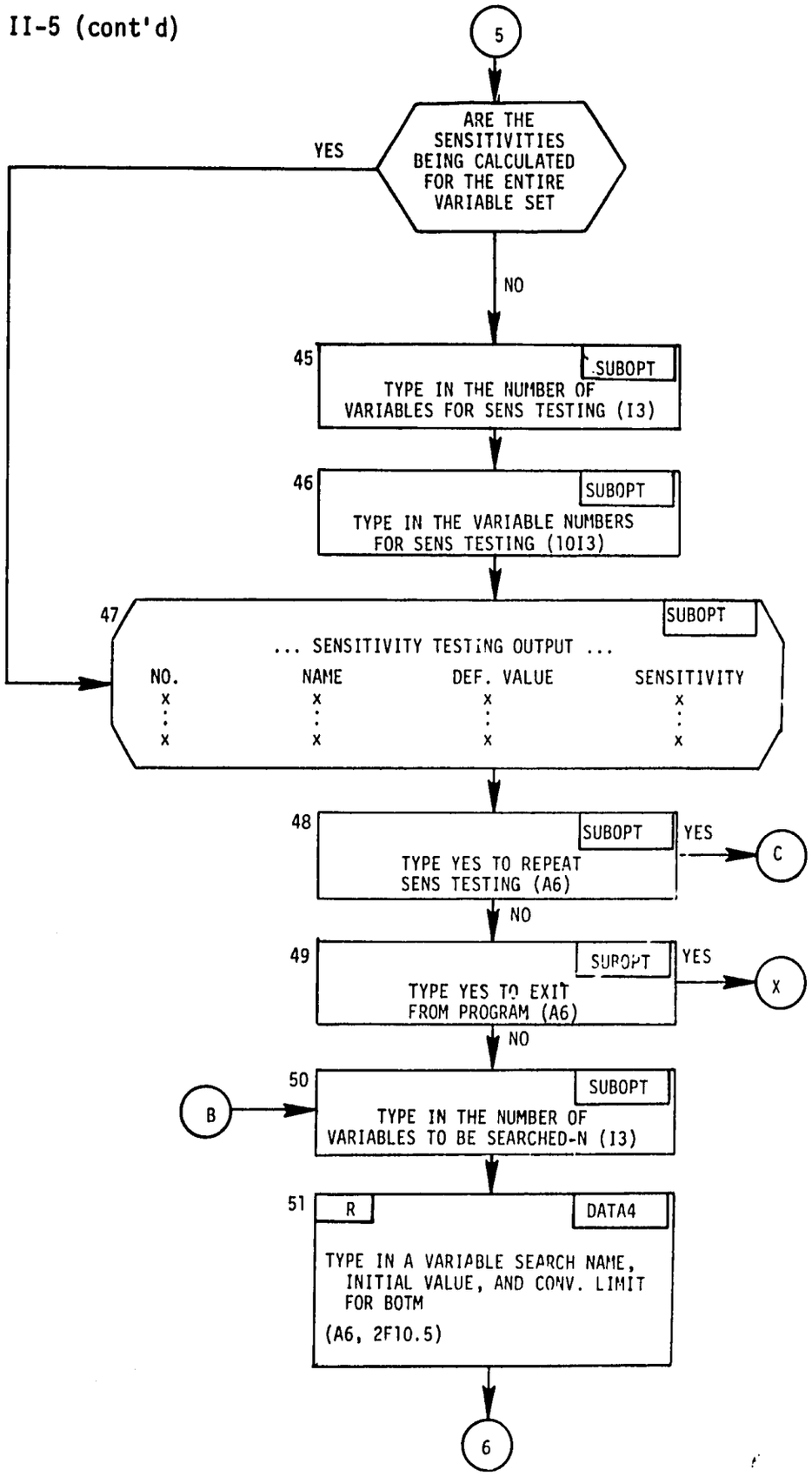


Figure II-5 (cont'd)

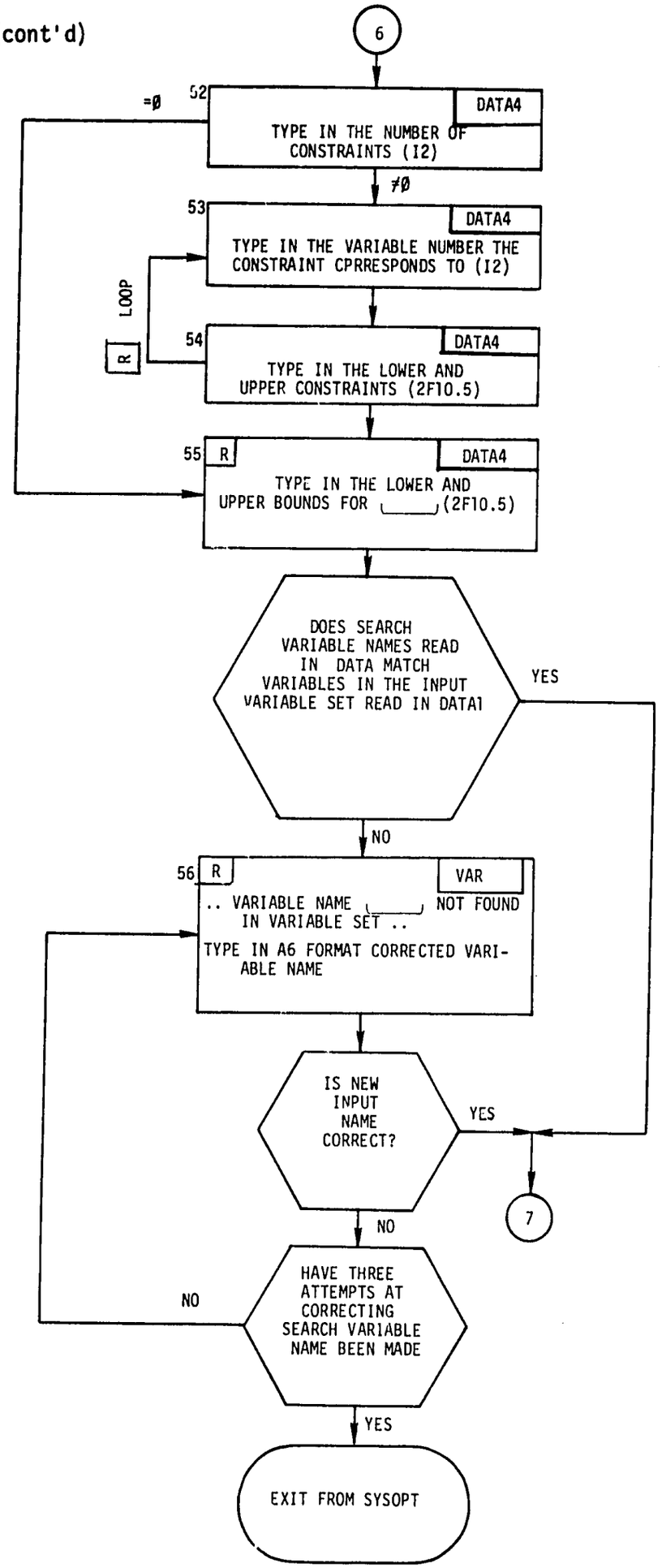


Figure II-5 (cont'd)

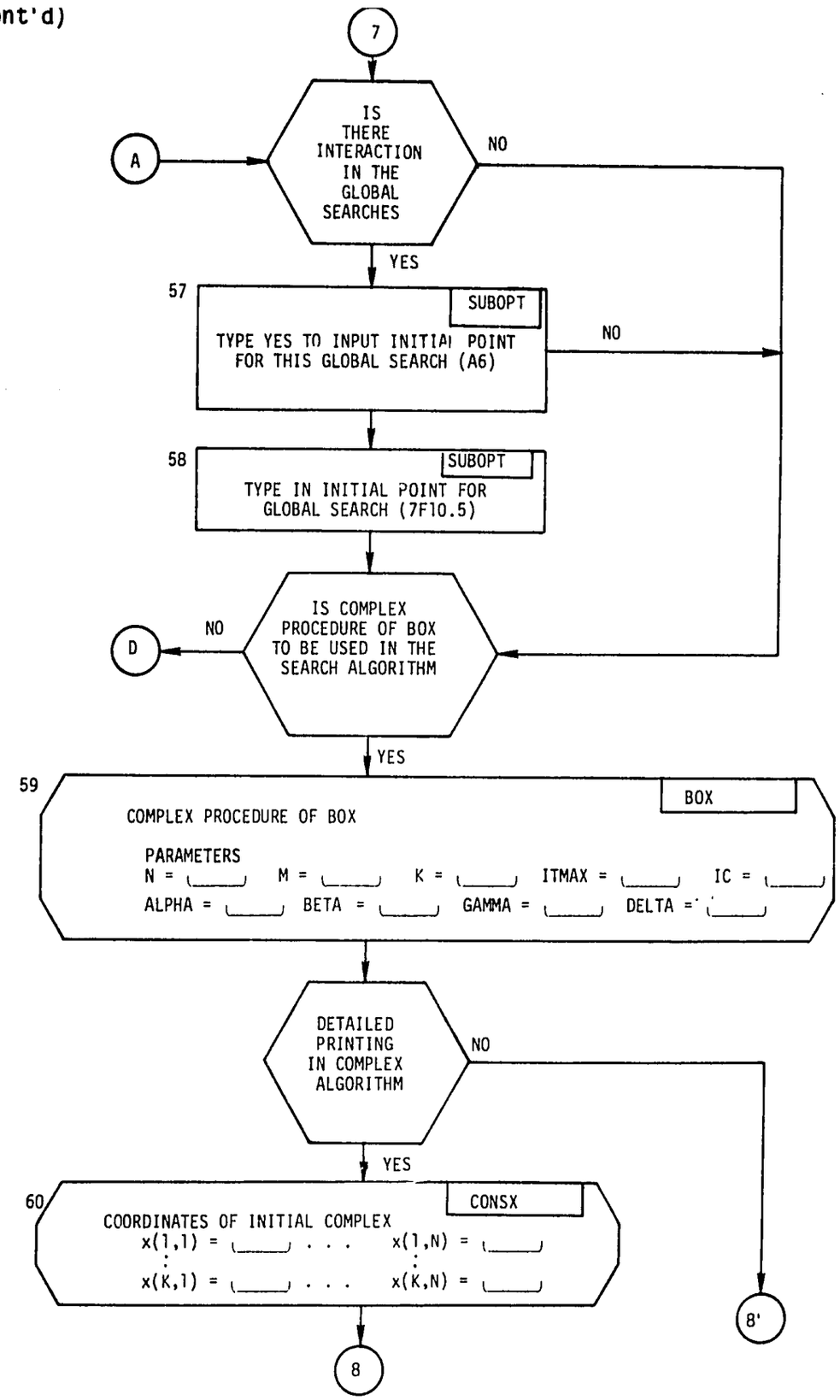


Figure II-5 (cont'd)

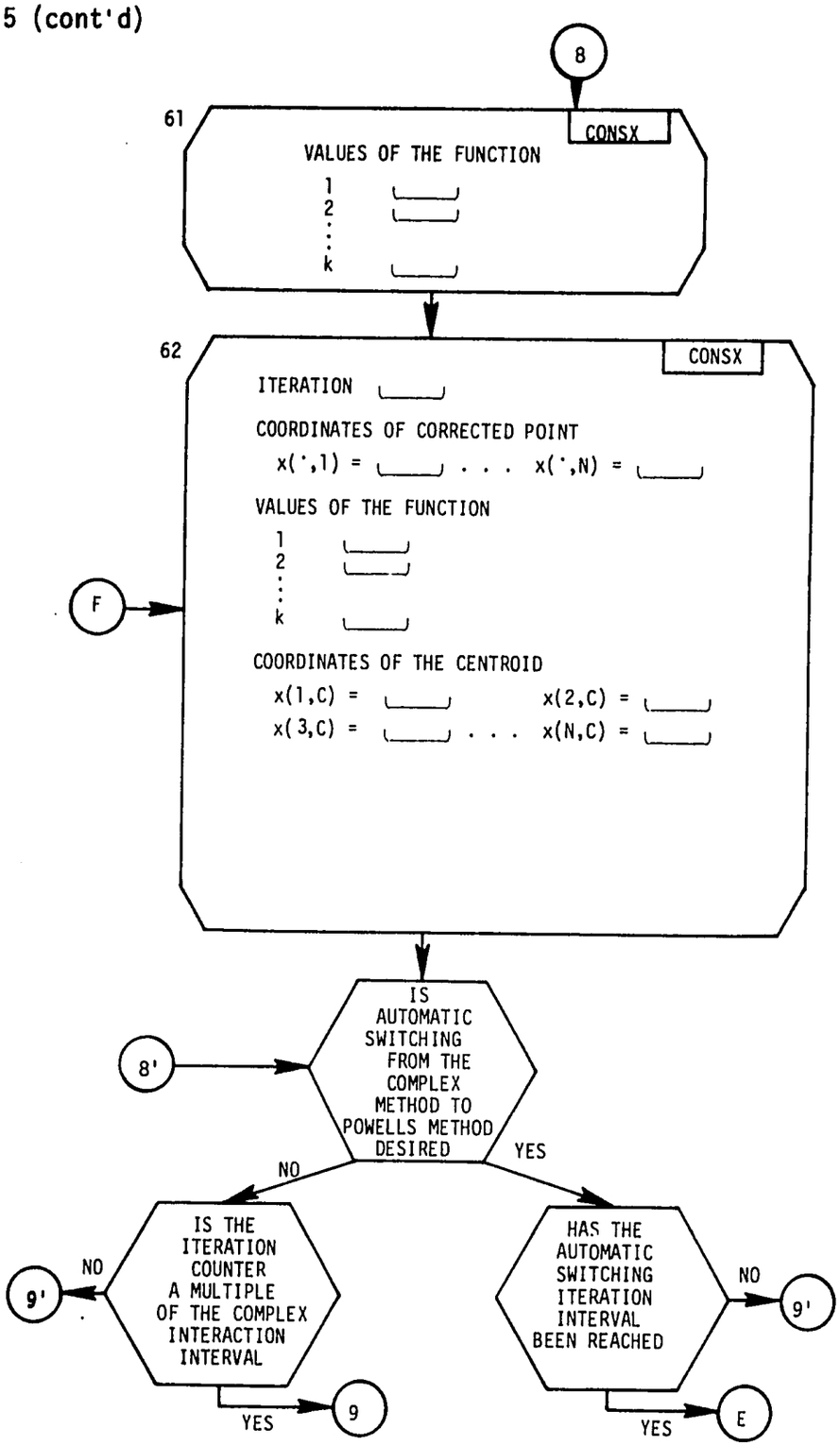


Figure II-5 (cont'd)

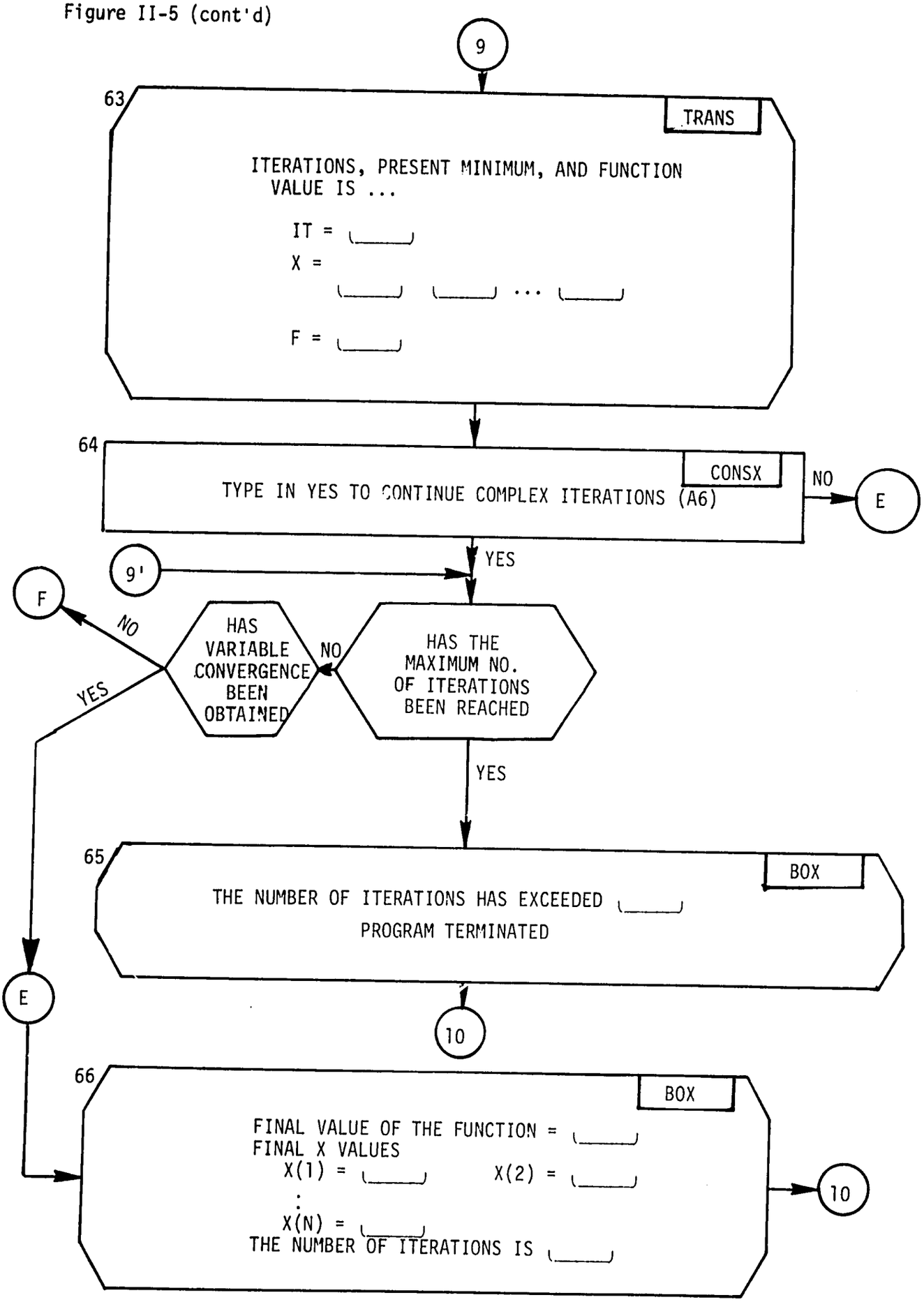


Figure II-5 (cont'd)

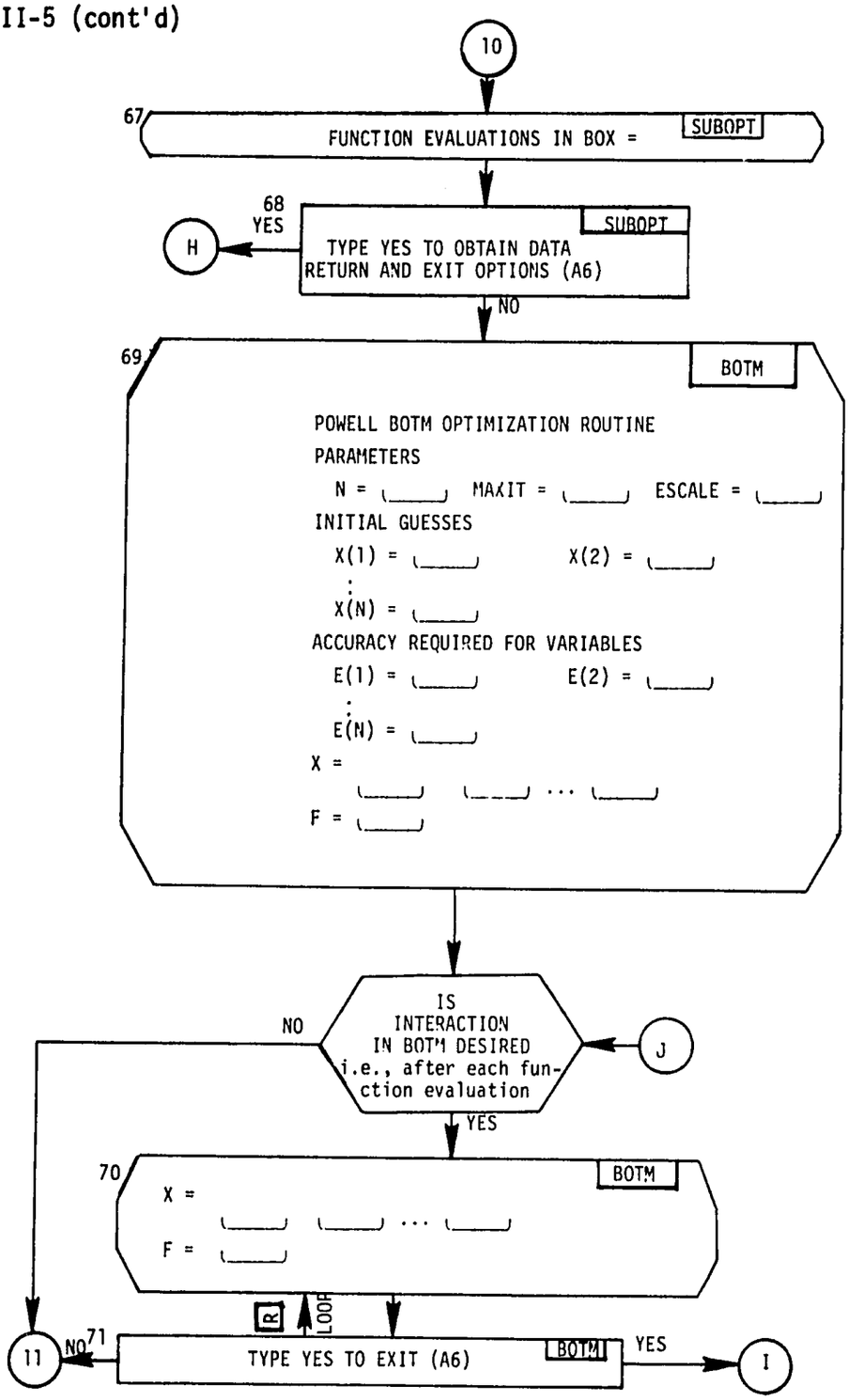


Figure II-5 (cont'd)

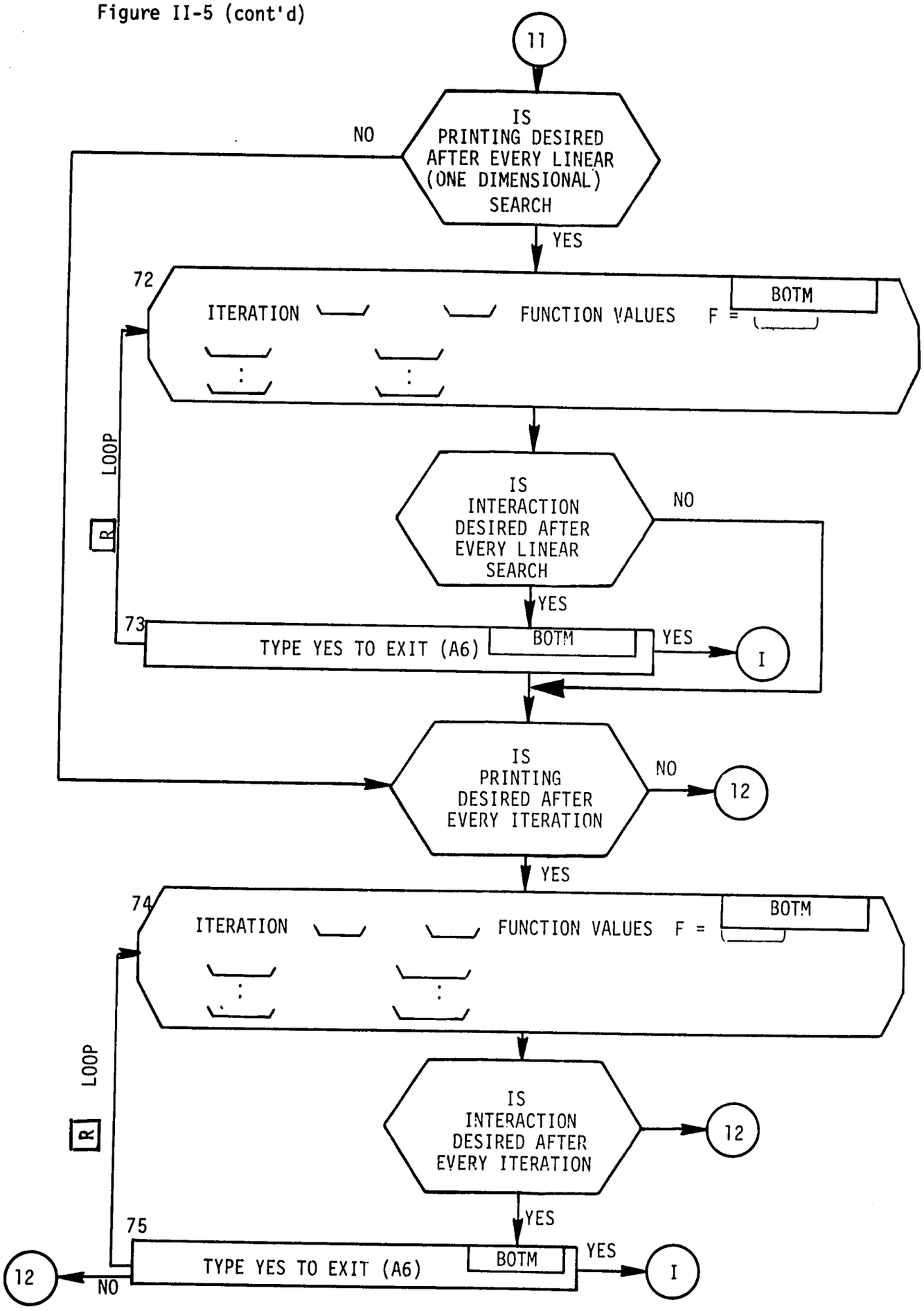


Figure II-5 (cont'd)

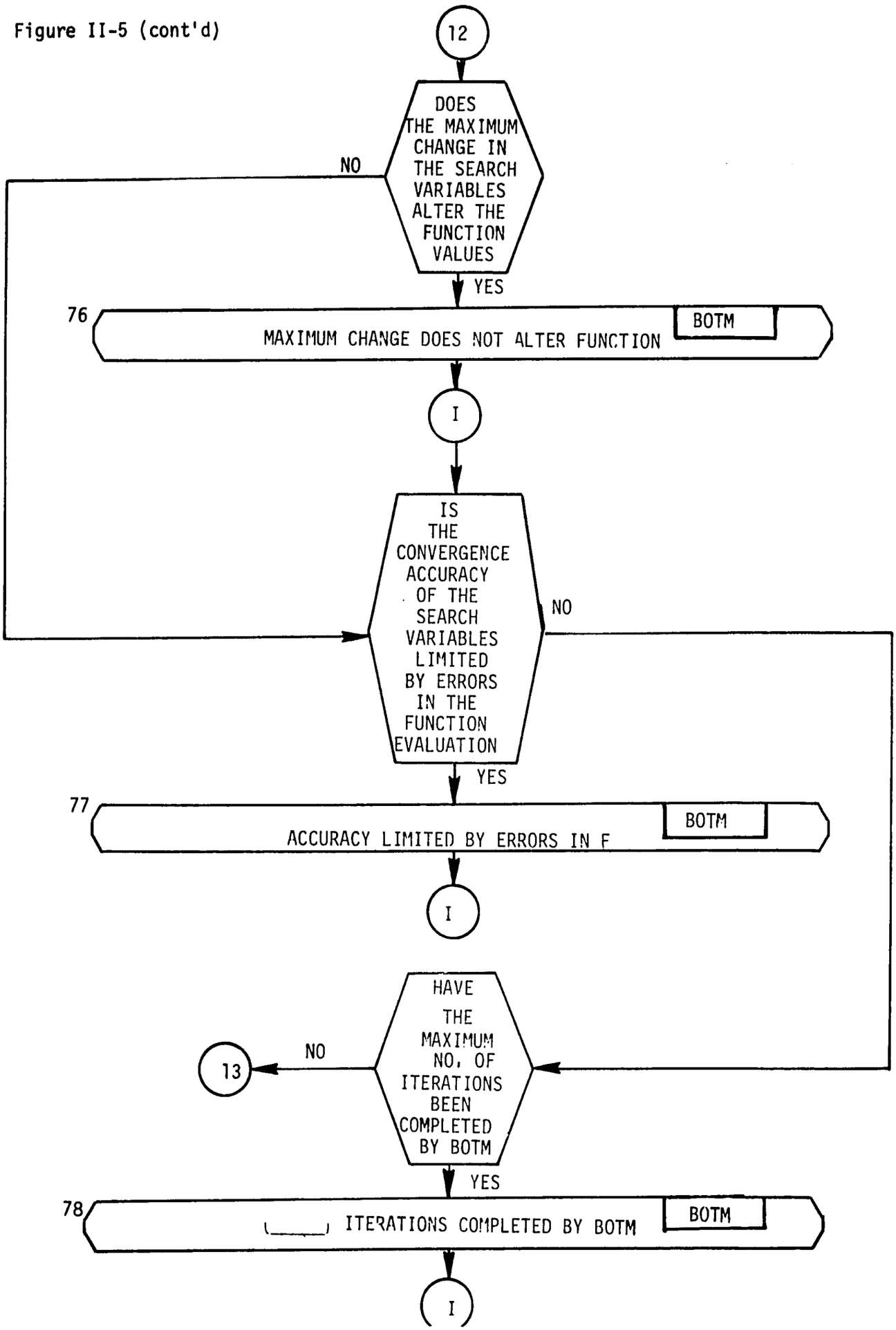


Figure II-5 (cont'd)

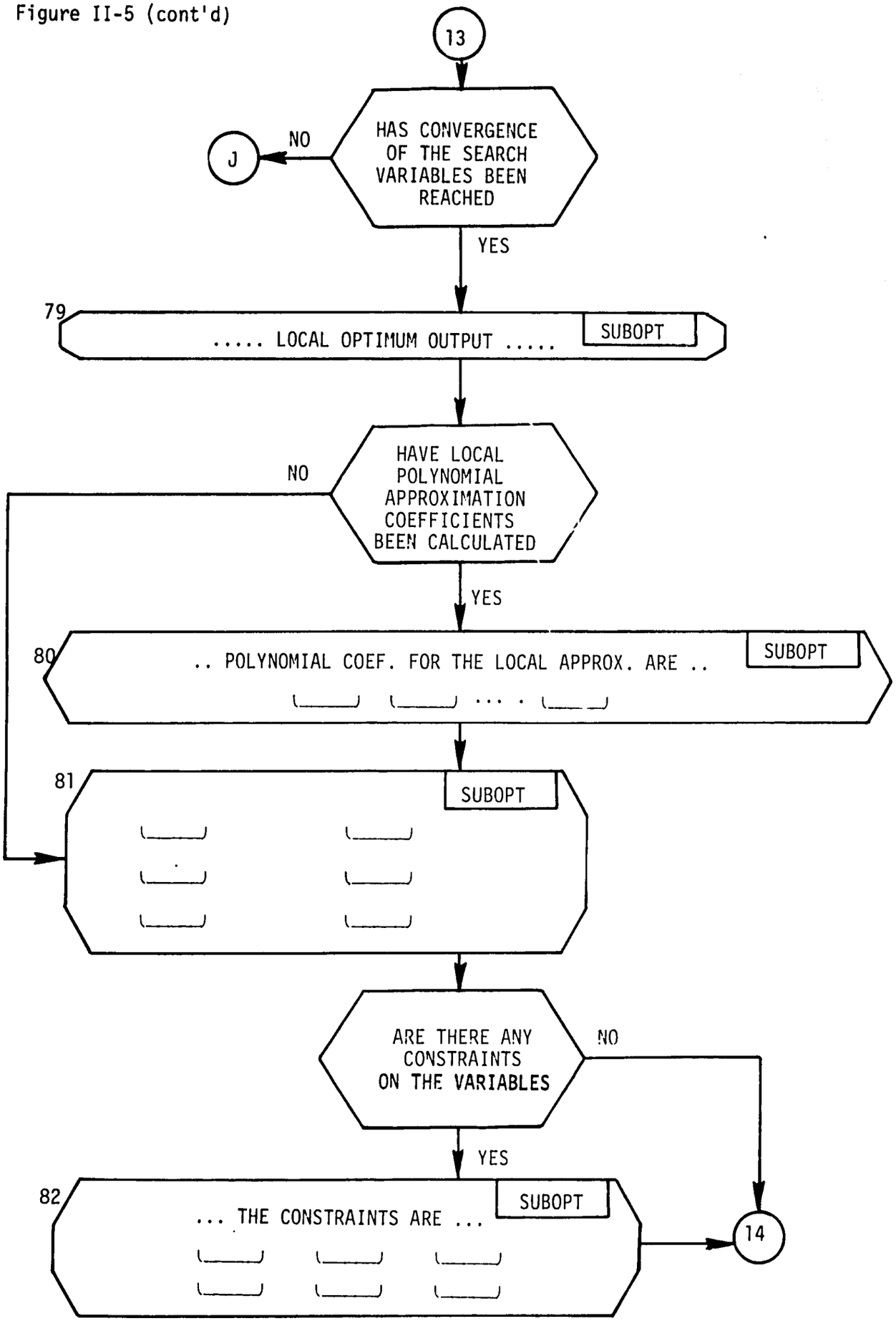


Figure II-5 (cont'd)

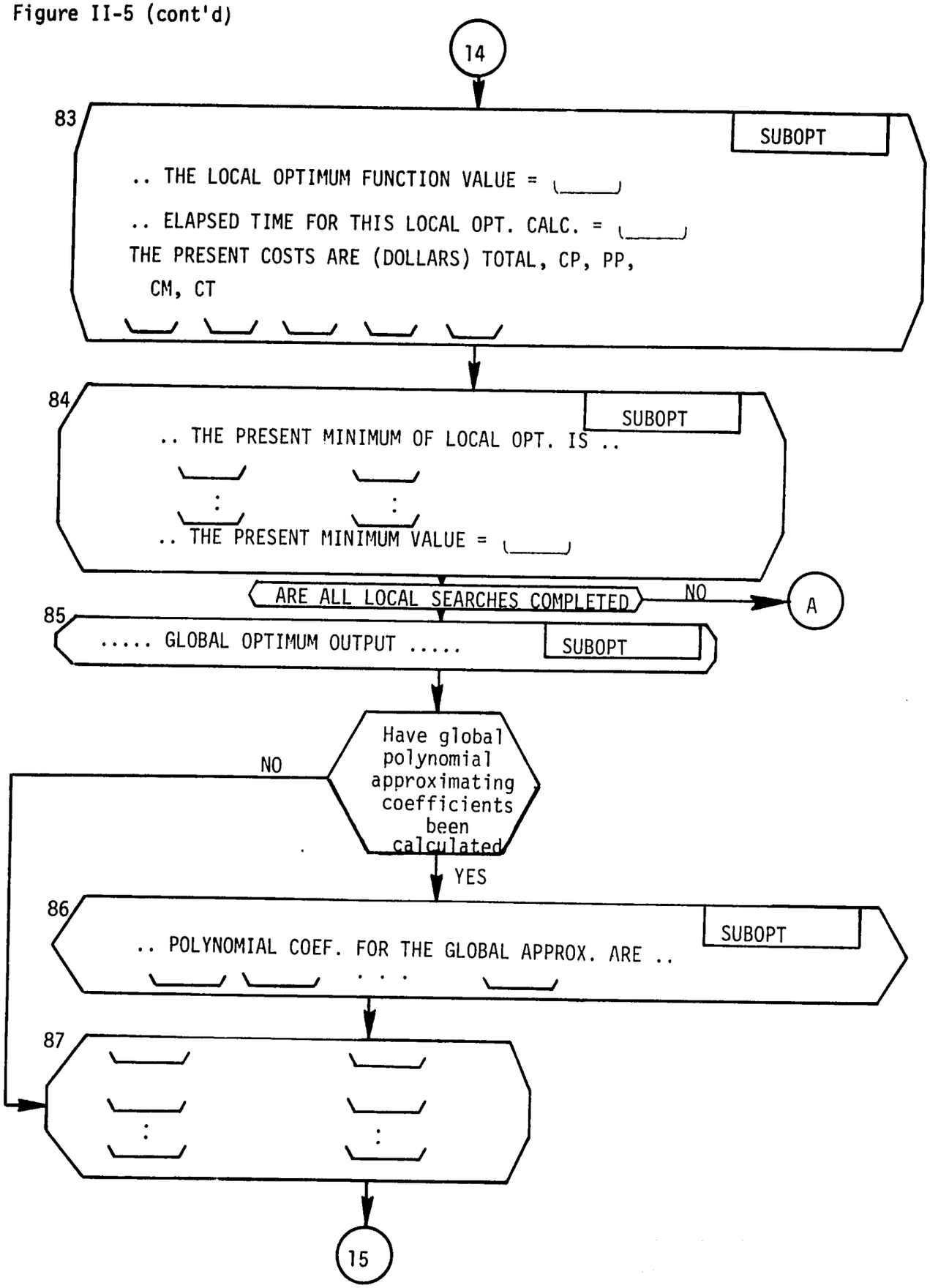


Figure II-5 (cont'd)

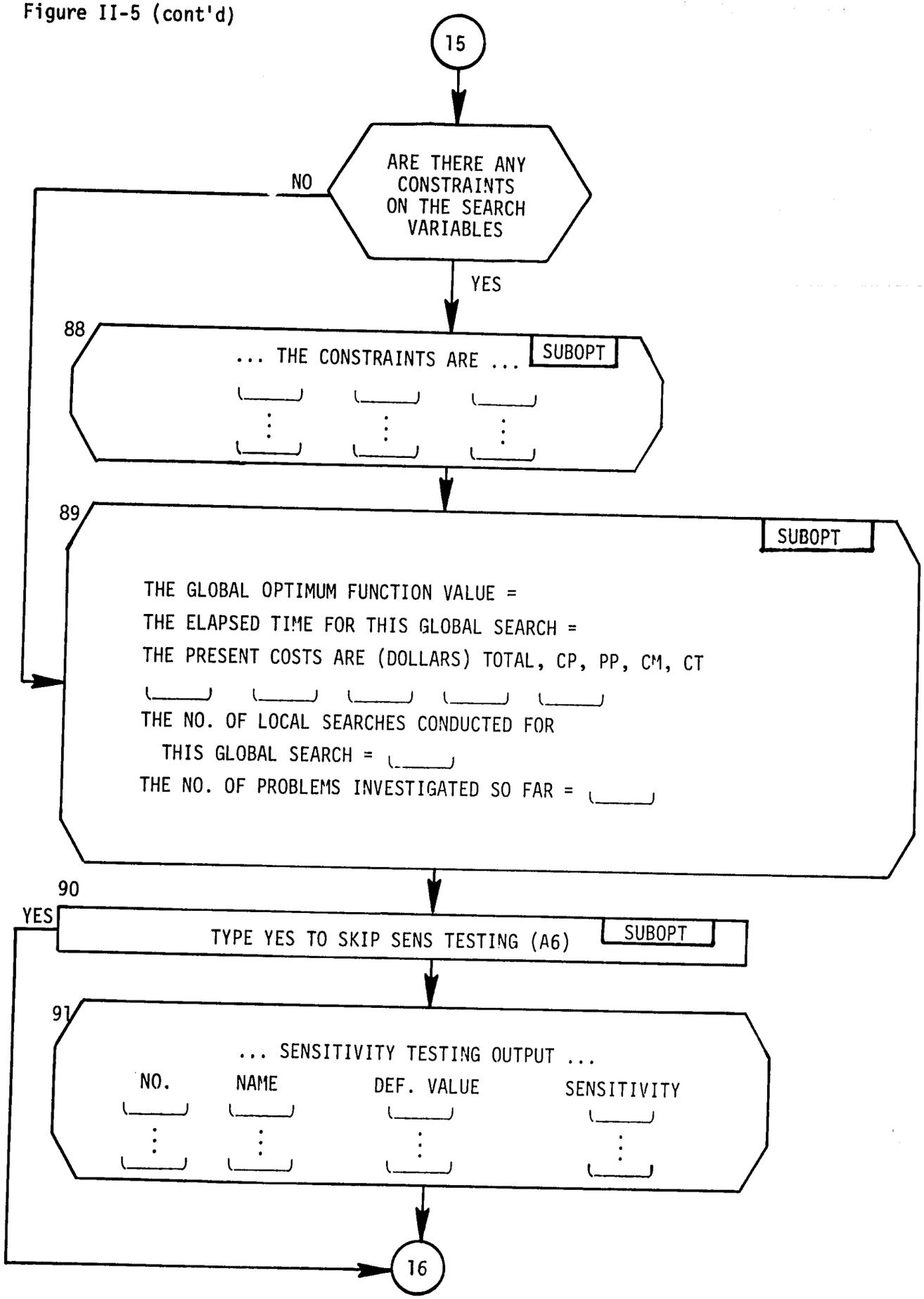


Figure II-5 (cont'd)

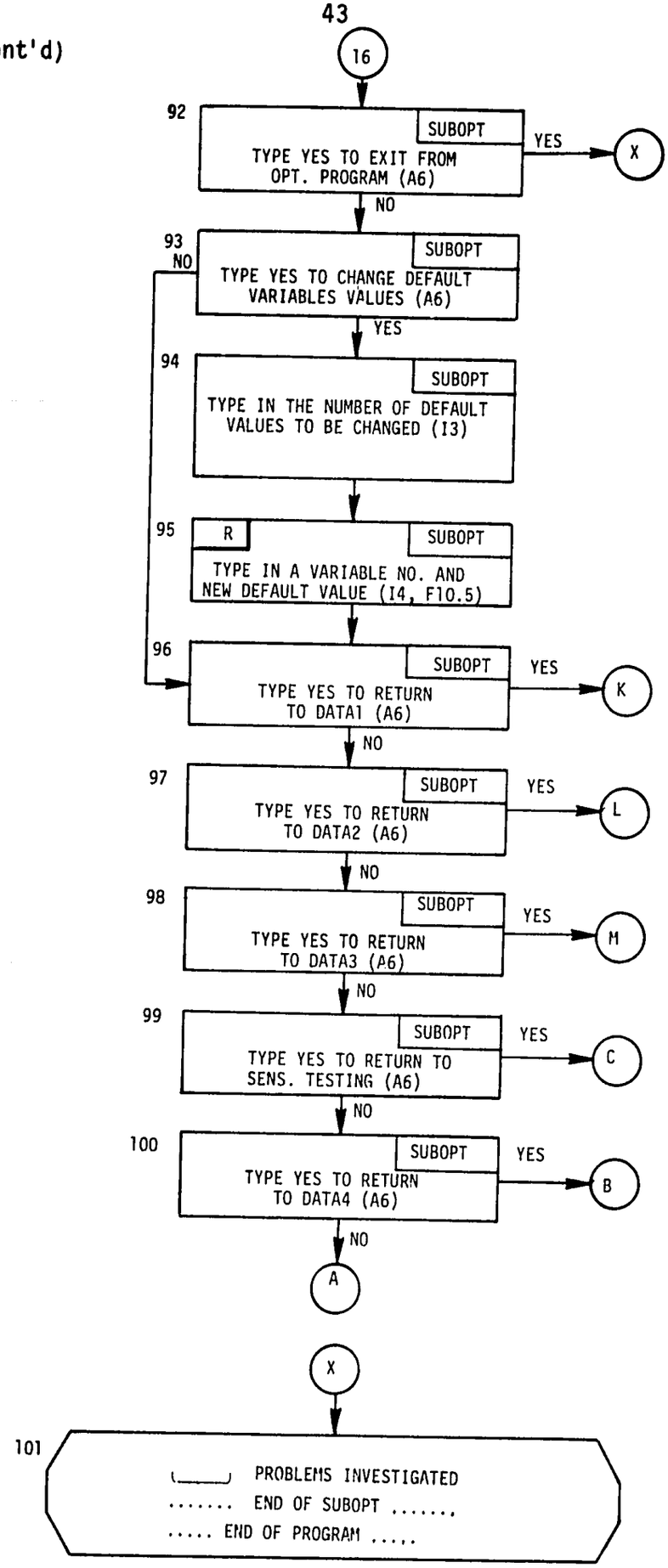


FIGURE II-6

Optimization Subcomponent: Interaction/Output Explanations

Box #

1 "TYPE IN THE NUMBER OF TIME SERIES-NTS (I3)"

NTS is the number of time series (or number of variables represented in the real world data set) used in the calculation of the least squares error criterion between the real world data and the model outputs. In an optimal control problem NTS=MOUTS.

2 "TYPE IN THE MAXIMUM NUMBER OF DATA POINTS-NDP (I3)"

NDP is the maximum number of data points in the time series. For an optimal control problem NDP=1.

3 "TYPE IN THE NUMBER OF INPUT VARIABLES-NVAR (I3)"

NVAR is the number of input variables in the simulation model:
XOUT(1),XOUT(2),...,XOUT(MOUTS).

4 "TYPE IN THE NUMBER OF MODEL OUTPUTS-MOUTS (I3)"

MOUTS is the number of model outputs in the simulation model:
XOUT(1),XOUT(2),...,XOUT(MOUTS).

5 "TYPE IN THE NUMBER OF PARAMETERS IN THE OBJECTIVE FUNCTION-NPOBJ (I3)"

NPOBJ is the number of parameters specifying the form of the objective function. These parameters can be changed during the investigation of a problem.

6 "TYPE IN THE RATE GROUP FOR THIS EXECUTION (F1.0)"

The RATE GROUP is a CDC method of pricing computer usage. During the day and for high priority of entrance into the computer the rate group = 3; the lowest rate group = 1 for either extremely low priority of entrance into the computer during the day or execution late at night and during the very early morning.

7 "TYPE YES TO READ INFORMATION FROM TAPE98 (A6)"

TAPE98 contains data transferred from the pattern recognition subcomponent to the optimization subcomponent-see section on computer input/output tape usage.

8 "TYPE Ø FOR A PE PROBLEM AND 1 IF OC PROBLEM (I1)"

Type Ø if a parameter estimation (PE) problem is being investigated and 1 if an optimal control (OC) problem is being investigated.

Figure II-6 (cont'd)

Box #

9 "TYPE THE NUMBER OF DATA POINTS IN THE ⌊_____⌋ TIME SERIES (I3)"

(PE problem only) Input the number of data points in the indicated time series. The Jth time series must correspond to the Jth output variable of the simulation.

10 "TYPE IN THE DATA POINTS ONE BY ONE (7F10.5)"

(PE problem only) For the time series corresponding to the previous inquiring in BOX 9, input the data points, seven to a line, in the indicated format.

11 "THE ⌊_____⌋VARIABLE NAME (A6) AND DEFAULT VALUE (F10.5)"

This calls for the input of the input variable names and their corresponding default (base) values. The Jth variable must correspond to the Jth input variable of the simulation model.

12 "TYPE IN OUTPUT NAMES (10A6)"

This calls for the input of the output variable names corresponding to the output variables in the simulation. The names are input 10 to a line in the indicated format.

13 "TYPE Ø FOR REAL WORLD TIME SERIES, OR 1 FOR SMOOTHED TIME SERIES (I1)"

To calculate the least squares error criterion for a parameter estimation problem, either the real world time series or the smoothed real world time series (from the pattern recognition subcomponent) can be used for comparison against the simulations model outputs.

14 "TYPE IN THE DEFAULT VALUES FOR THE VARIABLES (7F10.5)"

These are the input variable values not in the search set that are used for calculating the model outputs.

15 "TYPE MAX. NUMBER OF GLOBAL ITERATIONS (I2)"

This is the maximum number of local optimizations that will be conducted during any one global search.

16 "TYPE Ø TO PRINT POLY. APPROX. COEF (I1)"

If a polynomial approximation is computed over the search variable set space, then typing a Ø prints all the polynomial coefficients.

Figure II-6 (cont'd)

Box #

17 "TYPE Ø FOR NO SPEED-UP OF POWELL's METHOD (I1)"

Powell's speed-up method is used to increase the rate of convergence of the component along extremely sharp ridges or valleys in the objective function.

18 "TYPE Ø FOR DEFAULT SENS. TESTING ON WHOLE VARIABLE SET (I1)"

Typing a Ø will cause the subcomponent to automatically perform sensitivity tests or all the inputvariables.

19 "TYPE Ø FOR NO POLY APPROX., 1 FOR LOCAL APPROX., AND 2 FOR GLOBAL APPROX. (I1)"

Type a Ø for no polynomial (quadratic) approximation to the objective function to be calculated, type 1 for a polynomial approximation in the vicinity of each local optima to be calculated, and type 2 for a global polynomial approximation over the entire search variable space to be calculated. Note that ifthe switching capability between the complex method and Powell's method is to be used, then a local polynomial approximation must be calculated.

20 "TYPE Ø FOR NO INTERACTION IN GLOBAL SEARCH (I1)"

If automatic generation of initial points in the global search is desired, type Ø. Otherwise, the user will be queried each type an initial point for a local search is needed by the subcomponent.

21 "TYPE Ø FOR REGULAR SWITCHING (I1)"

Type an integer ≠Ø for compatibility switching (Q-conjugate) between the complex method and Powell's method. Compatibility switching means that the Hessian of the approximating polynomial, calculated during the complex method, will be used to construct an initial direction set to begin Powell's method. If a Ø is input here, they only the approximation to the local optimum found in the complex method will be used to begin Powell's method.

22 "TYPE Ø FOR MANUAL TRANS., +N INTERACTIVE TRANS AFTER N ITERATIONS OF BOX,
-N FOR AUTOMATIC SWITCHING AFTER N ITERATIONS (I3)"

The input specifies the type of switching between the complex method and Powell's method. A Ø specifies an interaction point after every iteration of BOX, a +N specifies an interaction point after every N

Figure II-6 (cont'd)

Box #

iterations, and a -N specifies automatic switching, regardless of how the search has proceeded, after N iterations of BOX.

23 "TYPE IN THE NUMBER OF DT UNITS FOR OUTPUT SPACING (I5)"

This is the desired number of DT time steps between model outputs calculated in the simulation.

24 "TYPE YES FOR DEFAULT PARAMETERS IN COMPLEX METHOD (A6)"

Typing YES retains the complex method parameter values most recently input to the subcomponent.

25 "TYPE SALPHA (F8.5), ITMAX (I4), IPRINT1 (I2), ALPHA, BETA, GAMMA, DELTA (4F8.5)"

SALPHA (≥ 1) is a scaling factor that controls the size of the initial complex,

ITMAX is the maximum number of iterations allowed in the complex method,

IPRINT1 is a parameter controlling printing of intermediate iterations-IPRINT1=1 causes intermediate values to print on each iteration, IPRINT1=0 supresses printing until the final solution is obtained-independently of the output iteration interval defined later,

ALPHA is a reflection factor of the complex (usually chosen to be 1.3),

BETA is a convergence parameter between successive iteration variable values,

GAMMA is a convergence parameter for variable values along constraints, and

DELTA is the explicit constraint violation correction.

26 "TYPE YES FOR DEFAULT PARAMETERS IN POWELLS METHOD (A6)"

Typing YES retains the Powell parameter values most recently input to the subcomponent.

27 "TYPE IPRINT2 (I2), MAXIT (I4), ESCALE (F8.5)"

For Powell's method, IPRINT2 is a parameter controlling printing of intermediate results-IPRINT2=2 causes intermediate values to be printed after every linear search, IPRINT2=1 causes intermediate values to be printed after every iteration, and IPRINT2=Ø supresses printing until the final solution is obtained, MAXIT is

Figure II-6 (cont'd)

Box #

the maximum number of iterations, and ESCALE is a search scaling parameter controlling the search step size.

28 "TYPE CONVERGENCE LIMIT FOR SPEED UP (F8.5)"

This parameter determines whether the speed up procedure of Powell's method will be executed in order to obtain an improvement in the global optimum approximation. If, in computing the optimal value of the most sensitive search variable with the speed up algorithm, the value obtained differs from the corresponding presumed global optimum value by less than this convergence limit, then the presumed global optimum will be taken to be the true global optimum. In other words if it is found that the presumed global optimum, from Powell's method, is the true global optimum then there were no sharp ridges a valleys in the objective function to prematurely halt the search.

29 "TYPE ADD (F8.5) FOR SPEED UP OF POWELLS METHOD"

ADD is the initial basic interval for stepping the most sensitive variable values along a sharp ridge a valley in the objective function input variable space.

30 "TYPE DEV (F8.5) FOR SENSITIVITY TESTING"

DEV is the parameter controlling the % change in the independent variables in the sensitivity testing.

31 "TYPE THE OUTPUT ITERATION INTERVAL FOR COMPLEX (I3)"

This is the iteration interval for printing the present minimum function value and corresponding search variable values in the complex method.

32 "TYPE Ø, 1, 2, OR 3 FOR INTERATION IN POWELL (I1)"

Type Ø for no interaction in Powell's method,
1 for interaction after every iteration,
2 for interaction after every linear search, and
3 for interaction after every function evaluation.

33 "TYPE IN COMPLEX MAX. NO. LOW REPEATER (I2)"

This variable limits the number of times one point in the search variable space can repeat having the lowest function value. In the complex method the inclusion of this variable was found to be necessary to prevent certain infinite loops from being executed.

Figure II-6 (cont'd)

Box #

34 "TYPE YES TO SKIP DATA3-READING OBJ. FUNC. PAR. (A6)"

DATA3 inputs the parameters of the objective function.

35 "TYPE Ø FOR LS OBJ FUNC AND 1 FOR WLS OBJ FUNC (I1)"

The least squares objective function has the form:

$$\sum P_i \ (Y_i - \overline{Y}_i)^2$$

whereas the weighted least squares objective function has the form

$$\sum P_i \left(\frac{Y_i - \overline{Y}_i}{Y_i} \right)^2$$

with P_i the weights on the residuals,
Y_i the real world time series, and
$\overline{Y}_i$ the model outputs.
(PE problems only)

36 "TYPE PENALTY PARAMETER FOR CONSTRAINT VIOLATION (F8.5)"

The value of this parameter is added to the objective function if a constraint is violated.

37 "TYPE YES OR NO FOR DEFAULT WEIGHTING ON TIME SERIES (A6)"

Default weighting implies that P_i=1.0 for all i in the objective function either for least squares or for weighted least squares error criteria
(PE problems only)

38 "FOR THE _____ TIME SERIES TYPE WEIGHTING COEFFICIENTS OVER TIME (10F8.5)"

If default weighting of time series was not specified, then this calls for the input of P_i for the specified time series.

39 "TYPE NUMBER OF PARAMETERS IN OBJECTIVE FUNCTION (I3)"

This is the number of parameters specifying the form of the objective function.
(OC problem only)

Figure II-6 (cont'd)

Box #

40 "TYPE PARAMETERS OF OBJECTIVE FUNCTION (10F8.5)"

The values that are input must correspond to the parameters in the objective function as coded in subroutines MMM and NNN.
(OC problem only)

41 "TYPE YES TO RETAIN VARIABLE SEARCH SET AND SKIP SENS TESTING (A6)"

Type YES to retain the search variable set that was used in the previous optimization problem and skip all sensitivity testing.
(SENS denotes sensitivity)

42 "TYPE YES TO SKIP SENS. TESTING (A6)"

Typing YES will cause the program execution to skip sensitivity testing and proceed immediately with the input of the desired search variable set.

43 "TYPE IN THE NUMBER OF DEFAULT VALUES TO BE CHANGED (I3)"

Type in the number of input variables for which the base values are desired to be changed.

44 "TYPE IN THE NUMBER AND DEFAULT VALUE FOR A VARIABLE (I3, F10.5)"

Type in the number of an input variable corresponding to its definition in the simulation model and the input variables new base, or default value.

45 "TYPE IN THE NUMBER OF VARIABLES FOR SENS. TESTING (I3)"

Sensitivity tests can be performed on subsets of the input variable set. The input to this question determines the size of the variable set for which sensitivity testing is desired and, consequently, the number of simulation runs necessary to perform the sensitivity calculations.

46 "TYPE IN THE VARIABLE NUMBERS FOR SENS TESTING (10I3)"

These numbers should correspond to the input variable numbers in the simulation model.

47 "... SENSITIVITY TESTING OUTPUT ...
NO. NAME DEF. VALUE SENSITIVITY"

This is the output from the sensitivity calculations performed in response to questions 45 and 46 and

Figure II-6 (cont'd)

Box #

consists of each input variable number, input variable name, input variable default value, and objective function sensitivity for the specified variables.

48 "TYPE YES TO REPEAT SENSITIVITY TESTING (A6)"

If additional sensitivity calculations are desired either for new input variables or new default values, type YES in response to this question.

49 "TYPE YES TO EXIT FROM PROGRAM (A6)"

If immediate exit from SYSOPT is desired as a result of the sensitivity calculations, type YES in response to this question.

50 "TYPE IN THE NUMBER OF VARIABLES TO BE SEARCHED - N (I3)"

N is the size of the search variable set, the set over which the objective function will be minimized, and a subset of the input variable set ($N \leq NVAR$).

51 "TYPE IN A VARIABLE SEARCH NAME, INITIAL VALUE, AND CONV. LIMIT FOR BOTM. (A6, 2F10.5)"

This specifies the search variable set elements and their corresponding successive iteration convergence limits. The initial value of the variable input here is superceded by initial values input later in the program.

52 "TYPE IN THE NUMBER OF CONSTRAINTS (I2)"

This is the number of hard constraints for the search variables ($\leq N$).

53 "TYPE IN THE VARIABLE NUMBER THE CONSTRAINT CORRESPONDS TO (I2)"

This is a search variable number corresponding to the order in which the search variables were specified in response to question 51.

54 "TYPE IN THE LOWER AND UPPER CONSTRAINTS (2F10.5)"

These constraint values correspond to the search constraint variable defined by the preceding question 53.

Figure II-6 (cont'd)

Box #

55 "TYPE IN THE LOWER AND UPPER BOUNDS FOR _____ (2F10.5)"

All variables in the search set must have lower and upper bounds specified for the generation of initial points in the global search and for defining general variable boundaries for the complex method. Type the lower and upper boundary values for the specified search variable.

56 "... VARIABLE NAME _____ NOT FOUND IN VARIABLE SET ... TYPE IN A6 FORMAT CORRECTED VARIABLE NAME"

This indicates that the specified search variable name did not match with any of the input variable names. Three attempts at correcting the improperly spelled input variable names are allowed. After three failures the program execution ends.

57 "TYPE YES TO INPUT INITIAL POINT FOR THIS GLOBAL SEARCH (A6)"

This question is executed if indicated by the answer to question 20. If the user wants to input the initial point for the search variables in the local search, then type YES. If automatic generation randomly within the boundary values, of the initial point is desired, then type NO in response to the question.

58 "TYPE IN INITIAL POINT FOR GLOBAL SEARCH (7F10.5)"

If the answer to question 57 is YES, then this question calls for input of the actual initial point of the local search for a particular global search.

59 "COMPLEX PROCEDURE OF BOX PARAMETERS
N= M= K= ITMAX= IC=
ALPHA= BETA= GAMMA= DELTA="

This output reminds the user of the present values of the parameters of the complex method. These parameters were input in response to question 25.

60 "COORDINATES OF INITIAL COMPLEX
X(1,1)=...X(1,N)
:
X(K,1)=...X(K,N)="

This output is the N coordinates of the K=N+1 vertices of the initial complex that were generated randomly (according to the scaling factor SALPHA) in the search variable space bounded by BOUNSL, BOUNDSV.

Box #

61 "VALUES OF THE FUNCTION
1
2
.
.
.
K"

This output consists of the objective function values corresponding to the vertices of the initial complex output in box 60. (detailed printing option only)

62 "ITERATION COORDINATES OF CORRECTED POINT
X(.,1)=...X(1,N)=
VALUES OF THE FUNCTION
1
2
.
.
.
K

COORDINATES OF THE CENTROID
X(1,C)= X(2,C)=

X(B,C)= . . . X(N,C)="

This output of the complex method corresponds to the vertices of the complex and the coordinates of the centroid of the complex at the specified iteration. The values of the objective function correspond to the vertices and the corrected point is the new vertex of the complex at that iteration. (detailed printing option only)

63 "ITERATIONS, PRESENT MINIMUM, AND FUNCTION VALUE IS ...
IT=
X=
F="

At the specified iteration, corresponding to the output iteration interval input in response to question 31, the present iteration number, the coordinates of the vertex having the minimum objective function value, and the actual objective function value is output in this format.

64 "TYPE IN YES TO CONTINUE COMPLEX ITERATIONS (A6)"

In response to the output data already given to the user, a decision must be made at this point whether to continue execution of the complex method, switch to Powell's Method, or exercise data and exit options. Typing YES causes continued execution of the complex method.

Box #

65 "THE NUMBER OF ITERATIONS HAS EXCEEDED (_____)
PROGRAM TERMINATED"

The number of complex iterations has exceeded the specified value defined as the maximum number of permitted iterations in response to question 25.

66 "FINAL VALUE OF THE FUNCTION =
FINAL X VALUES
X(1)= X(2)=
:
X(N)=
THE NUMBER OF ITERATIONS IS "

This output consists of the best approximation to the local optimum as determined by the complex method. Included in the output are the coordinates of the approximate optimum, the corresponding objective function value, and the number of iterations required to execute the complex method.

67 "FUNCTION EVALUATIONS IN BOX ="

This output specifies the number of objective function evaluations and the number of simulation runs, necessary to complete the number of complex iterations specified in output 66.

68 "TYPE YES TO OBTAIN DATA RETURN AND EXIT OPTIONS (A6)"

Type YES to discontinue the present local search and either exit from the program or change the search variable set, input variable default values, optimization method parameters, or in general any data input in response to questions 7-55.

69 "POWELL BOTM OPTIMIZATION ROUTINE PARAMETERS
N= MAXIT= ESCALE=
INITIAL GUESSES
X(1)= X(2)=
:
X(N)=
ACCURACY REQUIRED FOR VARIABLES
E(1)= E(2)=
:
E(N)=
X=
F=

Box #

This output specifies the most recent values for the parameters in Powell's method, as input in response to questions 27, 50, and 51; the initial point in the search, as determined from the results of the complex method; and the objective function value corresponding to the initial point. (X = initial point)

70 "X=
F= "

After each function evaluation, if this print option is chosen, the values of the search variables and the corresponding objective function value is output to the user.

71 "TYPE YES TO EXIT (A6)"

Typing YES in response to this question causes an immediate halt to the execution of Powell's method and transfer of control to the SUBOPT subroutine. After typing YES, the best approximation to the local optimum, as calculated up to this point in the program is used as the local optimum for the remainder of program execution.

72 "ITERATION
FUNCTION VALUES
F = _____ _____ ... _____"

For each Powell, this output specifies the iteration number, the number of function evaluations made up to and including this current linear search, the current optimal objective function value, and the corresponding current optimal search variable values.

73 "TYPE YES TO EXIT (A6)"

See explanation to question 71.

74 "ITERATION
FUNCTION VALUES
F = _____ ... _____"

See explanation to question 72, except that this output occurs only after each Powell iteration instead of linear search.

75 "TYPE YES TO EXIT (A6)"

See explanation to question 71.

Box #

76 "MAXIMUM CHANGE DOES NOT ALTER FUNCTION"

If the maximum change in the search variables does not alter the objective function values, then this message is printed on the output file and execution of Powell's method is halted.

77 "ACCURACY LIMITED BY ERRORS IN F"

If the convergence accuracy of the search variables is limited by errors in the objective function evaluation, then this message is printed on the output file and execution of Powell's method is halted.

78 "(_____) ITERATIONS COMPLETED BY BOTM"

If the maximum number of iterations, as specified by the response to question 27, have been reached in Powell's method, then this message is printed on the output file and execution of the method is halted.

79 "..... LOCAL OPTIMUM OUTPUT ....."

This output indicates that the following output will pertain to the results of the current local search.

80 ".. POLYNOMIAL COEF. FOR THE LOCAL APPROX. ARE .."

If in response to question 16, the user types Ø, then the polynomial coefficients of the approximating quadratic to the objective function near the current local optimum will be output here. The output of the coefficients proceeds with the squared terms, the cross product terms, the linear terms, and the constant term. (For specific definitions see subroutine descriptions in the programmer's section of this guide.)

81 "(_____) (_____)
(_____) (_____)
⋮ ⋮
(_____) (_____)

This output specifies the local optimum approximation as calculated by the four-level search algorithm. The order of output corresponds to the definition of the search variable set in response to question 51.

82 "... THE CONSTRAINTS ARE ..."

This output corresponds to the response to questions 53 and 54. The data are output to remind the user of the current active constraint set for the search variables.

Figure II-6 (cont'd)

Box #

83 ".. THE LOCAL OPTIMUM FUNCTION VALUE =
.. ELAPSED TIME FOR THIS LOCAL OPT. CALC =
THE PRESENT COSTS ARE (DOLLARS) TOTAL, CP, PP, CM, CT"

This output informs the user of the objective function value corresponding to the local optimum search variablue values in output box 81, and the amount of computer resources used in the SYSOPT program execution consisting of CP (central processor), PP (peripheral processor), CM (central memory), and CT (teletype) costs.

84 ".. THE PRESENT MINIMUM OF LOCAL OPT. IS ..
⌊_____⌋ ⌊_____⌋
⋮ ⋮
⌊_____⌋ ⌊_____⌋
.. THE PRESENT MINIMUM VALUE = "

This output consists of the local optimum having the lease objective function value among all local optima computed for the current global search. Both the search variable values and the corresponding objective function values are printed.

85 "..... GLOBAL OPTIMUM OUTPUT ....."

This output indicates that the following output will pertain to the results of the current global search.

86 ".. POLYNOMIAL COEF. FOR THE GLOBAL APPROX. ARE .."

If in response to question 16, the user typed Ø, then the polynomial coefficients of the approximating quadratic to the objective function over the entire search variable space will be output here. The output of the coefficients proceeds with the squared terms, the cross product terms, the linear terms, and the constant term. (For specific definitions see subroutine descriptions in the programmer's section of this guide.)

87 "⌊_____⌋ ⌊_____⌋
⌊_____⌋ ⌊_____⌋
⋮ ⋮
⌊_____⌋ ⌊_____⌋"

This output specifies the global optimum approximation as calculated by the program. The order of the output corresponds to the definition of the search variable set in response to question 51.

Figure II-6 (cont'd)

Box #

88 "... THE CONSTRAINTS ARE ..."

This output corresponds to the response to questions 53 and 54. The data are output to remind the user of the current active constraint set for the search variables.

89 "THE GLOBAL OPTIMUM FUNCTION VALUE =
THE ELAPSED TIME FOR THIS GLOBAL SEARCH =
THE PRESENT COSTS ARE (DOLLARS) TOTAL, CP, PP, CM, CT
_____ ... _____
THE NO. OF LOCAL SEARCHES CONDUCTED FOR THIS GLOBAL SEARCH =
THE NO. OF PROBLEMS INVESTIGATED SO FAR = "

This output informs the user of the objective function value corresponding to the global optimum search variable values in output 87, the amount of computer resources used in the SYSOPT program execution consisting of CP (central processor), PP (peripheral processor), CM (central memory), and CT (teletype) costs, the number of local searches performed for the current global search, and the number of global searches (problems) investigating with SYSOPT.

90 "TYPE YES TO SKIP SENS TESTING (A6)"

If desired, sensitivity testing of the whole search variable set at the global optimum can be done by typing NO in response to this question. Typing YES skips the execution of these calculations.

91 "... SENSITIVITY TESTING OUTPUT ...

NO.	NAME	DEF. VALUE	SENSITIVITY
_____	_____	_____	_____
⋮	⋮	⋮	⋮
_____	_____	_____	_____ "

This output corresponds to the calculations performed in response to question 90 and consists of the number, name, value, and sensitivity of each input variable at the global optimum.

92 "TYPE YES TO EXIT FROM OPT. PROGRAM (A6)"

Typing YES causes an immediate exit from SYSOPT.

93 "TYPE YES TO CHANGE DEFAULT VARIABLE VALUES (A6)"

If, as a result of the results of the previous problem solutions, the user wishes to change any input variable default value, YES should be input in response to this question.

Figure II-6 (cont'd)

Box #

94 "TYPE IN THE NUMBER OF DEFAULT VALUES TO BE CHANGED (I3)"

This is the number of input variables for which new default values are desired by the user.

95 "TYPE IN A VARIABLE NO. AND NEW DEFAULT VALUE (I4, F10.5)"

For each variable in the input variable set for which a new default value is desired, the variable number (as defined in response to question 11) and new default value must be input.

96 "TYPE YES TO RETURN TO DATA1 (A6)"

Typing YES causes a return in the execution of the program to question 7.

97 "TYPE YES TO RETURN TO DATA2 (A6)"

Typing YES causes a return in the execution of the program to question 24.

98 "TYPE YES TO RETURN TO DATA3 (A6)"

Typing YES causes a return in the execution of the program to question 34.

99 "TYPE YES TO RETURN TO SENS. TESTING (A6)"

Typing YES causes a return in the execution of the program to question 43.

100 "TYPE YES TO RETURN TO DATA4 (A6)"

Typing YES causes a return in the execution of the program to question 50. TYPING NO causes a default return in the execution of the program to question 57.

101 "(_____) PROBLEMS INVESTIGATED
..... END OF SUBOPT .....
..... END OF PROGRAM ....."

This output consists of the number of problems (global searches) investigated with the SYSOPT program and a message indicating the end of program execution.

Example of Interactive Use of SYSOPT Components

The following example of SYSOPT use, complete with a detailed line by line explanation of user responses will serve to illustrate the execution and operation of both the pattern recognition subcomponent and the optimization subcomponent. The problem being investigated with the component in the example is parameter estimation of seven variables when the values of six output variables are known for a fited period of time. While the pattern recognition subcomponent was used to smooth the time series and to generate the input for the optimization subcomponent, the optimization subcomponent was used to perform sensitivity calculations and numerical optimization searches.

The simulation model that generated the parameter estimation problem that is being investigated in the example is described in the "SYSOPT Simulation Programming" section of this guide and involves the modeling of a single commodity market with price control exercised by the government through purchases and sales in the open market. There are eight input variables (parameters) in the model: C1, a parameter relating excess of demand over supply, or supply over demand, and the rate of change of the price level DELP, the mean production delay time, DEMC, SUPC, which relate a general form of demand and supply vs. price curves to the particular system being studied; and KP, KR, KPI, KRI, which are feedback gains in the government regulation control loops. On the other hand, there are six output variables in the model: DEM, the demand for the commodity; P, the price of the commodity; GINV, the size of the government inventory; GIMP, the rate at which the imports are received; PLGIMP, the amount of outstanding orders for the commodity that have not been filled on the open market; and

CP, the rate at which the government purchases or sells the commodity in the market. These six variables represent measurable quantities in a real economic system. In the example, and in the testing of the component, errors in measurement of these output variables, have been modeled with zero mean random variables.

The following Figures II-7 and II-8, of the application of SYSOPT to solve a parameter estimation problem involving the single commodity model, will illustrate the use of the program to solve other parameter estimation or policy decision problems.

FIGURE II-7

Pattern Recognition Sub-Component

```
01/08/75 +MSU HUSTLER 2 L239 L SD 26.27  01/03/75

 TYPE PASSWORD, PN, AND USER ID.
 xxxxxx,090549,BUCHNER

 SS57458, LINE  42
 LAST ACCESS:  S 01/07/75 23:04
 RUNS:  20    BALANCE: $   69.43
 SYSTEMS WILL GO DOWN 345AM 6 JAN FOR SYSTEM DEV -UP AT 0900.

 READY 01.03.21

 OK.PROMPT.ATTACH,A,SIM98AA.PURGE,A.
  ATTACH,A,SIM98AA.
  PURGE.A.
 OK-ATTACH,LGO,PRMAIN.ATTACH,B,PRONE.ATTACH,C,PRTWO.ATTACH,TAPE97,SIM3.
  ATTACH,LGO,PRMAIN.
  ATTACH,B,PRONE.
  ATTACH,C,PRTWO.
  ATTACH,TAPE97,SIM3.
 OK-LOAD,A.
 LOAD,B,LOAD,C.LGO.
  EXEC BEGUN.01.04.49.
 TYPE IN THE NUMBER OF TIME SERIES-NTS (I3)
 *006
 TYPE IN THE MAXIMUM NUMBER OF DATA POINTS-NDP (I3)
 *020
 TYPE IN THE NUMBER OF INPUT VARIABLES-NVAR (I3)
 *003
 TYPE IN THE NUMBER OF MODEL OUTPUTS-MOUTS (I3)
 *006
 TYPE IN THE INPUT VARIABLE NAMES (10A6)
 *C1    DEMC  SUPD  DELP  KRI   KPI   KP    KR
 TYPE IN THE DEFAULT VALUES FOR THE VARIABLES (7F10.3)
 *2.        1.        1.        1.        1.        1.      .5
 *0.
 TYPE 0 IF P E PROBLEM AND 1 IF NOT (I1)
 *0
 TYPE OUTPUT VARIABLE NAMES (10A6)
 *P     GP    GIMP  DEM   GINV   PLGIMP
 TYPE IN IMT,ITT,DT,DUP FOR SIMULATION(2I4,2F8.3)
 *00010001.05      1.
 TYPE YES TO READ TIME SERIES FROM TAPE97 (A6)
 *YES
 TYPE YES TO OBTAIN SMOOTHING OF TIME SERIES (A6)
 *YES
 TYPE YES FOR SMOOTHING WITH INITIAL VALUES (A6)
 *NO
```

Figure II-7 (cont'd)

```
45        FOR THE SMOOTHING M=Ø=REAL WORLD T. S., M=1=LINEAR FILT.,
              AND M=2=QUADRATIC FIL
          .
          TYPE IN M FOR THE TIME SERIES P        (I3)
          *002
          TYPE IN M FOR THE TIME SERIES GP       (I3)
50.       *002
          TYPE IN M FOR THE TIME SERIES GIMP     (I3)
          *001
          TYPE IN M FOR THE TIME SERIES DEM      (I3)
          *001
55.       TYPE IN M FOR THE TIME SERIES GINV     (I3)
          *001
          TYPE IN M FOR THE TIME SERIES PLGIMP   (I3)
          *001
          TYPE YES FOR PRINT/PLOT OF TIME SERIES (A6)
60.       *NO
          TYPE YES FOR PLOT/PRINT OF MODEL OUTPUTS  (A6)
          *NO
          TYPE YES FOR SIMEX STUDY OF MODEL  (A6)
          *NO
65.       TYPE YES TO CONTINUE PATTERN RECOGNITION STUDY  (A6)
          *NO
          TYPE YES TO WRITE TAPE98 (A6)
          *YES
          .....END OF PROGRAM.....
70.        END MAINPR
          OK-CATALOG,TAPE98,SIM98AA,ID=BUCHE-NER,PP=999.
           CATALOG,TAPE98.SIM98AA,ID=BUCHNER,PP=999.
```

Figure II-7 (cont'd)

Line Numbers	Explanation
1-8	Computer output and user input to obtain access to the CDC 6500 computer at Michigan State University.
9	User input to initiate execution of computer data input programs (OK. and PROMPT.), attach the permanent file SIM98AA, and purge the permanent file SIM98AA. The program OK notifies the user of the teletype that the 6500 is waiting for a operating system command by printing OK. On the other hand PROMPT notifies the user of an executing program that the program is waiting for input of data by printing.*
12	User input to attach the permanent files necessary for pattern recognition sub-component execution. PRMAIN, PRONE, PRTWO contain the library code for SYSOPT SIM3 contains the real world time series that will be used in formqlating the parameter estimation problem in the optimization sub-component. Note that the simulation model was not attached to a local file as it will not be needed in the program execution. However, if plot/prints of model output, or normal simulation execution is desired the model must be attached and loaded into the computer core with the other subroutines.
13-16	Computer output indicating that the proper permanent files have been attached to the users present local file.
18	User input to load the SYSOPT program (pattern recognition sub-component) into central memory and begin execution.
19	Computer output indicating program execution has begun.
21	User input specifying that six different time series will be used to formulate the parameter estimation problem to be solved with the component.
23	User input specifying that there are 20 data points in each of the six time series.
25	User input specifying that there are 8 input variables (parameters) in the simulation model.
27	User input specifying that there are six simulation model output variables corresponding to the six time series.

Line Numbers	Explanation
29	User input specifying the names of the eight input variables C1, DEMC, SUPC, DELP, KRI, KPI, KP, and KR.
31,32	User input specifying that the default, or base, values of the eight input variables are C1=2.0, DEMC=1.0, SUPC=1.0, DELP=1.0, KRI=1.0, KPI=1.0, and KR=0.2.
34	User input specifying that the problem being investigated a parameter estimation problem in contract to an optimal control problem. If the user specifies a parameter estimation problem is being studied, then a least squares error criterion is automatically calculated for evaluating the objective function in the optimization sub-component and time series printing/ plotting and smoothing is permitted in thepattern recognition sub-component.
36	User input specifying the names of the output variables in the simulation model.
38	User input specifying the re is to be INT = 1 printer line space between points on the time series plots (if desired) ITT = 1 DT time unit between model outputs that are calculated, and passed to the SYSOPT program, with the simulation model, DT = .05 yr basic time unit step for the simulation model, and DUR = 1.0 yr is the length of the simulation model.
40	User input specifying that the program should read the time series for the parameter estimation problem from TAPE97.
42	User input specifying that smoothing of the real world time series is desired for the parameter estimation problem formulation.
44	User input specifying that the timeseries smoothing is to be calculated without known initial values.
48	User input specifying a quadratic polynomial filter is to be used for smoothing the time series corresponding to the output variable P.

Figure II-7 (cont'd)

Line Numbers	Explanation
50	User input specifying a quadratic polynomial filter is to be used for smoothing the time series corresponding to the output variable GP.
52	User input specifying a linear polynomial filter is to be used for smoothing the time series corresponding to the output variable GIMP.
54	User input specifying a linear polynomial filter is to be used for smoothing the time series corresponding to the output variable DEM.
56	User input specifying a linear polynomial filter is to be used for smoothing the time series corresponding to the output variable GINV.
58	User input specifying a linear polynomial filter is to be used for smoothing the time series corresponding to the output variable PLGIMP.
60	User input specifying that no plots and/or printout of the real world and/or smoothed time series is desired.
62	User input specifying that no plots and/or printout of the simulation model outputs is desired.
64	User input specifying that no SIMEX (or normal) execution of the model is desired.
66	User input specifying that execution of the pattern recognition sub-component should be halted.
68	User input specifying that TAPE98 should be constructed to simplify the input of data to the optimization sub-component. The information contained on the local file indicates: 1. whether the problem being investigated is a parameter estimation or optimal control (parameter estimation in the example) 2. the number of input variables (8) 3. the names of the input variables (C1, DEMC, SUPC, DELP, KRI, KPI, KP, and KR) 4. the number of output variables 5. the names of the output variables P, GP, GIMP, DEM, GINV, PLGIMP

Figure II-7 (cont'd)

Line Numbers	Explanation
	6. the time series values if a parameter estimation problem is being studied.
69,70	Computer output indicating that TAPE98 has been constructed and that the program has halted execution.
71	User input specifying that the local file, TAPE98, containing the input to the optimization sub-component should be permanently stored in the computer under the file name TAPE98AA.
72	Computer output indicating the successful completion of storing TAPE98 as TAPE98AA permanently on the disc space in the computer.

FIGURE II-8

Optimization Sub-Component

```
01/07/75 MSU HUSTLER 2 L339 LSD 36.27  01/03/75

TYPE PASSWORD, PN, AND USER ID.
xxxxxxxxxx, 090549,BUCHNER.
INVALID PROBLEM NO.

TYPE PASSWORD, PN, AND USER ID.
xxxxxxxxxx,090540,BUCHNER.

SS56787, LINE  46
LAST ACCESS: V 01/07/75 00:09
RUNS:  15   BALANCE: $   95.61

READY 00.22.17

OK.PROMPT.RTL,220.RFL,45000.
OK-ATTACH,TAPE98,SIM98AAA,ATTACH,LGO,OPTMAIN.ATTACH,A,OPTSIM,
 ATTACJ,TAPE98,SIM98AAA.
 ATTACH,LGO,OPTMAIN.
 ATTACH,A,OPTSIM.
OK-ATTACH,B,OPTBOK.ATTACH,C,OPTBOTM.ATTACH,D,OPTSUB.ATTACH,E,OPTDATA.
 ATTACH,B,OPTBOK.
 ATTACH,C,OPTBOTM.
 ATTACH,D,OPTSUB.
 ATTACH,E,OPTDATA.
OK-LOAD,A,LOAD,B.LOAD,C.LOAD,D.LOAD,E.LGO.
 EXEC BEGUN.00.24.45.
TYPE IN THE NUMBER OF TIME SERIES-NTS (I3)
*006
TYPE IN THE MAXIMUM NUMBER OF DATA POINTS-NDP (I3)
*020
TYPE IN THE NUMBER OF INPUT VARIABLES-NVAR (I3)
*008
TYPE IN THE NUMBER OF MODEL OUTPUTS-MOUTS (I3)
*006
TYPE IN THE NUMBER OF PARAMETERS IN THE OBJECTIVE FUNCTION-NPOBJ (I3)
*000
TYPE IN THE RATE GROUP FOR THIS EXECUTION (F1.0)
*1
TYPE YES TO READ INFORMATION FROM TAPE98 (A6)
*YES
TYPE IN THE DEFAULT VALUES FOR THE VARIABLES (7F10.5)
*2.        1.        1.        1.        1.        8.
*.2
TYPE MAX. NUMBER OF GLOBAL ITERATIONS (I2)
*00
TYPE 0 TO PRINT POLY. APPROX. COEF (I1)
*1
TYPE 0 FOR NO SPEED-UP OF PPWELLS METHOD (I1)
*0
TYPE 0 FOR DEFAULT SENS. TESTING ON WHOLE VARIABLE SET (I1)
```

Figure II-8 (cont'd)

```
*0
TYPE 0 FOR NO POLY APPROX., 1 FOR LOCAL APPROX., AND 2 FOR GLOBAL
     APPROX. (I1)
*0
TYPE 0 FOR NO INTERACTION IN GLOBAL SEARCH (I1)
*1
TYPE 0 FOR REGULAR SWITCHING (I1)
*0
TYPE 0 FOR MANUAL TRANS, +N FOR INTERACTIVE TRANS AFTER N ITERATIONS
     OF BOX,
-N FOR AUTOMATIC SWITCHING AFTER N ITERATIONS (I3)
*030
TYPE IN THE NUMBER OF DT UNITS FOR OUTPUT SPACING (I5)
*00001
TYPE SALPHA(F8.5), ITMAX(I4), IPRINT1(I2), ALPHA, BETA, GAMMA,
     DELTA(4F8.5)
*4/
5.      0200001.3    .0001    18.     .0001
TYPE IPRINT2(I2), MAXIT(I4), ESCALE(F8.5)
*020050 50.
TYPE DEV (F8.5) FOR SENSITIVITY TESTING
*.0001
TYPE THE OUTPUT ITERATION INTERVAL FOR COMPLEX (I3)
*030
TYPE 0, 1, 2, OR 3 FOR INTERACTION IN POWELL (I1)
*1
TYPE IN COMPLEX MAX. NO. LOW REPEATER (I2)
*15
TYPE PENALTY PARAMETER FOR CONSTRAINT VIALATION (F8.5)
*999999.
TYPE YES OR NO FOR DEFAULT WEIGHTING ON TIME SERIES (A6)
*YES
TYPE YES TO SKIP SENS TESTING (A6)
*NO
TYPE IN THE NUMBER OF DEFAULT VALUES TO BE CHANGED (I3)
*000
...SENSITIVITY TESTING OUTPUT....
NO.   NAME   DEF. VALUE    SENSITIVITY
  1   C1          2.00000   -45.70315
  2   DEMC        1.00000   149.64612
  3   SUPC        1.00000   -20.29275
  4   DELP        1.00000     -.95169
  5   KRI         1.00000      .05771
  6   KPI         1.00000      .08238
  7   KP          8.00000   -26.70358
  8   KR           .20000   -26.15767
TYPE YES TO REPEAT SENS TESTING (A6)
*NO
TYPE YES TO EXIT FROM PROGRAM (A6)
TYPE IN THE NUMBER OF VARIABLES TO BE SEARCHED-N (I3)
*007
TYPE IN A VARIABLE SEARCH NAME,INITIAL VALUE, AND CONV. LIMIT FOR BOTM
(A6,2F10.5)
*C1    2.         .01
TYPE IN A VARIABLE SEARCH NAME,INITIAL VALUE, AND CONV. LIMIT FOR BOTM
```

Figure II-8 (cont'd)

```
100.    (A6,2F10.5)
        *DEMC  1.          .01
        TYPE IN A VARIABLE SEARCH NAME,INITIAL VALUE, AND CONV. LIMIT FOR BOTM
        (A6,2F10.5)
        *SUPC  1.          .0
105.    TYPE IN A VARIABLE SE/  .I NAME,INITIAL VALUE, AND CONV. LIMIT FOR BOTM
        (A6,2F10.5)
        *KRI   1.          .01
        TYPE IN A VARIABLE SEARCH NAME,INITIAL VALUE, AND CONV. LIMIT FOR BOTM
        (A6,2F10.5)
110.    *KPI   1.          .01
        TYPE IN A VARIABLE SEARCH NAME,INITIAL VALUE, AND CONV. LIMIT FOR BOTM
        (A6,2F10.5)
        *KP    8.          .01
        TYPE IN A VARIABLE SEARCH NAME,INITIAL VALUE, AND CONV. LIMIT FOR BOTM
115.    (A6,2F10.5)
        *KR    .2          .002
        TYPE IN THE NUMBER OF CONSTRAINTS (I2)
        *00
        TYPE IN THE LOWER AND UPPER BOUNDS FOR C1       (2F10.5)
120.    *1.5        2.5
        TYPE IN THE LOWER AND UPPER BOUNDS FOR DEMC     (2F10.5)
        *.9         1.1
        TYPE IN THE LOWER AND UPPER BOUNDS FOR SUPC     (2F10.5)
        *.7         1.3
125.    TYPE IN THE LOWER AND UPPER BOUNDS FOR KRI      (2F10.5)
        *.5         1.5
        TYPE IN THE LOWER AND UPPER BOUNDS FOR KPI      (2F10.5)
        *.5         1.5
        TYPE IN THE LOWER AND UPPER BOUNDS FOR KP       (2F10.5)
130.    *.4          10.
        TYPE IN THE LOWER AND UPPER BOUNDS FOR KR       (2F10.5)
        *.1          .3
        TYPE YES TO INPUT INITIAL POINT FOR THIS GLOBAL SEARCH (A6)
        *YES
135.    TYPE IN INITIAL POINT FOR GLOBAL SEARCH (7F10.5)
        *1.8       .95       .8        1.2       1.2       6.       .16

                        COMPLEX PROCEDURE OF BOX
         PARAMETERS
         N =  7   M =  7   K =  8  ITMAX =  200   IC =  0
140.     ALPHA =  1.30     BETA =      .00010   GAMMA = 18  DELTA = .00010
        ITERATIONS, PRESENT MINIMUM, AND FUNCTION VALUE IS...
        IT = 30
        X=
           1.7902     .9502     .3538    1.0997    1.1871    6.9258    .1749
145.    F=      -13.84932
        TYPE IN YES TO CONTINUE COMPLEX ITERATIONS (A6)
        *YES
        ITERATIONS, PRESENT MINIMUM, AND FUNCTION VALUE IS...
        IT=  60
```

```
X=
   1.7995      .9774     .9577     1.0393     1.1091     8.6210     .2118
F=       -1.57210
TYPE IN YES TO CONTINUE COMPLEX ITERATIONS (A6)
*YES
ITERATIONS, PRESENT MINIMUM, AND FUNCTION VALUE IS...
IT=  90
X=
   1.7960      .9837     .9433     1.1032     1.0892     3.6554     .2205
F=       -1.09518
TYPE IN YES TO CONTINUE COMPLEX ITERATIONS (A6)
*YES
ITERATIONS, PRESENT MINIMUM, AND FUNCTION VALUE IS...
IT= 120
X=
   1.8188      .9849     .9529     1.0976     1.0894     8.7313     .2129
F=       -1.02948
TYPE IN YES TO CONTINUE COMPLEX ITERATIONS (A6)
*NO

 FINAL VALUE OF THE FUNCTION =       -.10294803E+01
 FINAL X VALUES
 X( 1) =        .18188167E+01
 X( 2) =        .98488746E+00
 X( 3) =        .95289452E+00
 X( 4) =        .10976128E+01
 X( 5) =        .10894196E+01
 X( 6) =        .87313027E+01
 X( 7) =        .21289932E+00
THE NUMBEROF ITERATIONS IS      120
FUNCTION EVALUATIONS IN BOX=       209
TYPE YES TO OBTAIN DATA RETURN AND EXIT OPTIONS (A6)
*NO

            POWELL BOTM OPTIMIZATION ROUTINE
 PARAMETERS
 N =  7    MAXIT =    50    ESCALE =   50.00
 INITIAL GUESSES
 X( 1) =   1.81881670E+00   X( 2) =  9.34887464E-01
 X( 3) =   9.52894524E-01   X( 4) =  1.09761276E+00
 X( 5) =   1.08941959E+00   X( 6) =  8.73130268E+00
 X( 7) =   2.12899321E-01   X(
 ACCURACY REQUIRED FOR VARIABLES
 E( 1) =    .10000000E-01   E( 2) =   .10000000E-01
 E( 3) =    .10000000E-01   E( 4) =   .10000000E-01
 E( 5) =    .10000000E-01   E( 6) =   .10000000E-01
 E( 7) =    .20000000E-02   E(
X=
    .18188E+01     .98489E+00     .95289E+00     .10976E+01
    .10894E+01     .87313E+01     .21290E+00
F=     .10295E+01
ITERATION    1        28 FUNCTION VALUES        F =  .49779320E+00
```

Figure II-8 (cont'd)

```
          .18178576E+01      .98488746E+00
          .95011791E+00      .95644769E+00
          .10693660E+01      .87543522E+01
          .21239932E+00
     TYPE YES TO EXIT (A6)
     *NO
    ITERATIONS     2       47 FUNCTION VALUES       F =  .27507219E+00
          .18176235E+01     .99415469E+00
          .99334455E+00     .92199087E+00
          .32864750E+00     .88783891E+01
          .21289932E+00
    TYPE YES TO EXIT (A6)
    *NO
    ITERATION      3       66 FUNCTION VALUES       F =  .23109390E+00
          .18159481E+01     .99539470E+00
          .99540893E+00     .90597668E+00
          .97911440E+00     .89089037E+01
          .21289932E+00
    TYPE YES TO EXIT (A6)
    *NO
    ITERATION      4       84 FUNCTION VALUES       F =  .20706302E+00
          .18158266E+01     .99634478E+00
          .99488929E+00     .88809328E+00
          .11432549E+01     .89395448E+01
          .21289932E+00
    TYPE YES TO EXIT (A6)
    *NO
      :                      :                              :

    ITERATION     28      536 FUNCTION VALUES       F =  .13227100E+00
          .20277988E+01     .10003768E+01
          .10031842E+01     .10136299E+01
          .91205574E+00     .79332991E+01
          .19649694E+00
    TYPE YES TO EXIT (A6)
    *NO
    ITERATION     29      551 FUNCTION VALUES       F =  .13227049E+00
          .20276804E+01     .10003693E+01
          .10031589E+01     .10136180E+01
          .91078688E+00     .79840242E+01
          .19649621E+00
    TYPE YES TO EXIT (A6)
    *NO
    ITERATION     30      567 FUNCTION VALUES       F =  .13226969E+00
          .20278165E+01     .10003570E+01
          .10031268E+01     .10136078E+01
          .91122368E+00     .79333648E+01
          .19648340E+00
    .....LOCAL OPTIMUM OUTPUT.....
```

Figure II-8 (cont'd)

```
          1   C1          2.02782
          2   DEMC        1.00036
          3   SUPC        1.00313
          4   KRI         1.01361
          5   KPI          .91122
          6   KP          7.93336
          7   KR           .19648
     . THE LOCAL OPTIMUM FUNCTION VALUE=      .13227E+00
     ..ELAPSED TIME FOR THIS LOCAL OPT. CALC.=  33.87900
     THE PRESENT COSTS ARE (DOLLARS) TOTAL,CP,PP,CM,CT
        1.58437    .33347    .02988    .66423    .00000
     ..THE PRESENT MINIMUM OF LOCAL OPT. IS..
          1   C1          2/02782
          2   DEMC        1.00036
          3   SUPC        1.00313
          4   KRI         1.01361
          5   KPI          .91122
          6   KP          7.93336
          7   KR           .19648
     THE PRESENT MINIMUM VALUE =          .13227E+00
     .....GLOBAL OPTIMUM OUTPUT.....
          1  C1          2.02782
          2  DEMC        1.00036
          3  SUPC        1.00313
          4  KRI         1.01361
          5  KPI          .91122
          6  KP          7.93336
          7  KR           .19648
     THE GLOBAL OPTIMUM FUNCTION VALUE=     .13227E+00
     THE ELAPSED TIME FOR THIS GLOBAL SEARCH=  33.91900
     THE PRESENT COSTS ARE (DOLLARS) TOTAL,CP,PP,CM,CT
        1.54107    .83487    .02730    .67890     .00000
     THE NO. OF LOCAL SEARCHES CONDUCTED FOR THIS GLOBAL SEARCH=   1
     THE NO. OF PROBLEMS INVESTIGATED SO FAR=   1
     ...SENSITIVITY TESTING OUTPUT...
     NO.   NAME   DEF. VALUE    SENSITIVITY
        1  C1         2.02782    10.81179
        2  DEMC       1.00036    33.06841
        3  SUPC       1.00313      .80919
        4  DELP       1.00000      .01562
        5  KRI        1.01361      .05837
        6  KPI         .91122      .00144
        7  KP         7.93336     3.16483
        8  KR          .19648     2.90210
     TYPE YES TO EXIT FROM OPT PROGRAM (A6)
     *YES
        1  PROBLEMS INVESTIGATED
     .......END OF PROGRAM.......
     .....END OF PROGRAM.....
      END MAINOPT
     OK-
```

Figure II-8 (cont'd)

Line Numbers	Explanation
1-10	Computer output and user input to obtain access to the CDC 6500 computer at Michigan State University.
11	User input to initiate execution of data input programs (OK. and PROMPT.), raise the time limit of program execution to 220 seconds (RTL, 220.), and raise the central memory allocation to 45000 octal words (RFL, 45000.).
12,16	User input to attach the permanent computer files necessary for optimization subcomponent execution. SIM98AAA contains the input from the pattern recognition subcomponent OPTMAIN, OPTBOX, OPTBOTM, OPTSUB, OPTDATA contain the binary code for SYSOPT OPTSIM contains the user-supplied simulation model
13-15, 17-20	Computer output indicating the proper permanent files have been attached to the user's present local file.
21	User input to load the SYSOPT program (optimization subcomponent) into central memory and begin execution.
22	Computer output indicating program execution has begun.
24	User input to specify six different time series will be used for parameter estimation.
26	User input to specify that there are 2Ø data points in each of the six time series.
28	User input to specify that there are 8 input variables (parameters) in the simulation model.
30	User input to specify that there are six simulation model outputs corresponding to the six time series.
32	User input to specify that there are no parameters to be varied in the objective function. For a parameter estimation problem, NPOBJ should always be set equal to Ø.
34	User input to specify rate group 1 of program execution. Note that line 1 and line 10 indicate that the program was executed Jan. 7, 1975, at 22 minutes after midnight which resulted in the low-rate group of execution.

Figure II-8

Line Numbers	Explanation
36	User input to specify that there is input from TAPE98, which was generated in the pattern recognition sub-component. This input serves to define: 1. Whether the problem being investigated is a parameter estimation or optimal control problem (parameter estimation) 2. The number of input variables (8) 3. The names of the input variables (C1, DEMC, SUPC, DELP, KRI, KPI, KP, KR) 4. The number of output variables (6) 5. The names of the output variables P, GP, GIMP, DEM, GINV, PLGIMP 6. The time series values.
38,39	User input to specify the default values of the input variables (C1=2, DEMC=1., SUPC=1., KRI=1., KPI=1., KP=8., KR=0.2)
41	User input to specify the maximum number of local searches to be conducted for the global search. Typing Ø causes 1 local search to be made.
43	User input specifying that polynomial approximating coefficients will not be printed
45	User input to specify that Powell's speed-up algorithm should not be executed
47	User input to specify sensitivity testing to be performed on the entire input variable set.
49	User input to specify that no polynomial approximation to the objective function should be made
51	User input specifying that interaction input--input of initial points of local searches--in the global search is desired.
53	User input specifying that regular switching between the complex method and Powell's method is desired.

Figure II-8 (cont'd)

Line Numbers	Explanation
56	User input specifying that interaction with the complex method is desired after every 30 iterations of the algorithm.
58	User input to specify that the simulation model output at time j*1*DT corresponds to the time series point j.
61	User input specifying the parameters of the complex method. SALPHA = complex scaling factor = 5.0 ITMAX = maximum number of iterations = 200 IPRINT1 = print option = 1 ALPHA = complex reflection factor = 1.3 BETA = search variable convergence limit = .0001 GAMMA = constraint convergence factor = 18. DELTA = search variable minimum variation = .0001
63	User input specifying parameters of Powell's method IPRINT2 = print option = 2 MAXIT = maximum number of iterations = 50 ESCALE = variable step scaling factor = 50.
65	User input specifying a .01% variation in input variable values for the sensitivity calculations.
67	User input specifying a 30-iteration interval for output from the complex method. Note that this corresponds to the complex interaction interval deferred in line 56.
69	User input specifying interaction in Powell's method after every iteration.
71	User input specifying a 15-iteration maximum of a vertex in the complex method having the minimum function value. (This value should be chosen to be twice the number of search variables.)
73	User input specifying that 99,999. should be added to the objective function each time a variable constraint is violated.
75	User input specifying that all residuals in the least squares objective function (always used in a problem specified as parameter estimation) will be equally weighted.
77	User input specifying that the sensitivity testing is not to be skipped.

Figure II-8 (cont'd)

Line Numbers	Explanation
79	User input specifying that there are no input variable default values to be changed before the sensitivity calculations.
80-89	Computer output giving the results of the sensitivity calculations performed on the entire input variable set
91	User input specifying that the sensitivity calculations should not be repeated (even with changed default values of the input variables).
93	User input specifying that no exit from the program should be made.
95	User input specifying that there will be seven (7) search variables in the local search.
98, 101, 104, 107, 110, 113, and 116	User input specifying the name and iteration convergence limit for Powell's method for each variable in the search variable set; C1, DEMC, SUPC, KRI, KPI, KP, and KR. Note that the input variable DELP was omitted from the search variable set. Also the initial values input at the point in the program will be superceded by the initial values input later.
118	User input specifying that there are no constraints on the search variables.
120, 122, 124, 126, 128, 130, and 132	User input specifying the lower and upper bounds for the variables in the search variable set. These are used for defining the general region of the search variable space that is to be investigated by the program.
134	User input specifying that the input of the initial point of the local search for the global search is desired.
136	User input specifying the initial point of the search to be C1 = 1.8, DEMC = .95, SUPC = 0.8, KRI = 1.2, point KRI = 1.2, KPI = 1.2, KP = 6., and KR = .16.
137-140	Computer output indicating the beginning of complex execution and the values of complex parameters.

Figure II-8 (cont'd)

Line Numbers	Explanation
141-168	Computer output of the complex method every 30 iterations, as specified by the response on line 67. The output consists of the iteration number, the search variable values having the current minimum function value, and the current minimum objective function value. For example, at iteration 90, the present minimum of search variables is C1 = 1.7960, DEMC = .9837, SUPC = .9433, KRI = 1.1082, KPI = 1.0892, KP = 8.6554, and KR = .2205 Corresponding to these values of the search variables is the objective function value, -1.09518. (Note that while least squares error criteria are positive, because the complex method is programmed to perform maximization, the component performs minimizations by maximizing the negative of the objective function. Therefore, the least squares error is 1.09518.)
147, 154, 161, and 168	User input specifying whether or not to continue complex execution. Note that the interaction interval is also 30 iterations as specified by the response on line 56. Lines 147, 154, and 161 indicate that the complex execution should continue, while line 168 indicates that the algorithm should halt.
169-179	Computer output indicating that the final results of the complex method took 209 function evaluations (simulation runs) in 120 iterations. The final least squares error objective function value is -(-1.02948) = 1.02948 occurring when the search variables have the values C1 = 1.8188, DEMC = .98489, SUPC = .95289, KRI = 1.0976, KPI = 1.0894, KP = 8.73130, and KR = .212899.
181	User input specifying that an exit from the program or a return to the data input hierarchy is not desired.
182-194	Computer output indicating the beginning of Powell's method the values of parameters, the initial values of the search variables (the final values from the complex method), and the variable iteration convergence values.
195-198	Computer output indicating the objective function evaluation in Powell's method corresponding to the initial values of the search variables.

Figure II-8 (cont'd)

Line Numbers	Explanation
199-248	Computer output of Powell's method for each iteration as specified in line 69. the output consists of the iteration number, the number of function values needed by the search up to and including the current iteration, the current optimal objective function value, and the corresponding optimal search variable values. For example, at iteration 28 the approximation to the local optimum was C1 = 2.027799, DEMC = 1.00038, $SUPC_+$= 1.0032, KRI = 1.01363, KPI = .91206, KP = 7.9333, and KR = .196497 with a corresponding objective function value of .132271, which was obtained in 536 objective function evaluations.
205, 212, 219, 226, 236, and 243	User input specifying whether or not to continue Powell's the execution of method. All the responses shown indicate that the execution should not be externally halted before the variable convergence is obtained.
249	Computer output indicating that local optimum information will follow.
250-256	Computer output indicating the local optimum search variable values.
257	Computer output indicating the local optimum function value, .13227, corresponds to the local optimum search variable values printed on lines 250-256.
258	Computer coutput indicating that 33.879 seconds of central processor computer time was needed to perform the most recent local optimum calculation.
259-260	Computer output indicating that the computer resources used in program cost approximately $1.52 and consist of $.83 for central processor use, $.03 for peripheral processor use, $.66 for central memory use, and $0.00 for teletype use (not accounted for at the time of this program execution).
261-269	Computer output indicating that the current search variable values having the lowest objective function value of all the local optima calculating during the global search up to the point corresponds to the values shown. As there was only one local optimum calculated for the global search, the present minimum of the local optima corresponds to the only local optimum.

Figure II-8 (cont'd)

Line Numbers	Explanation
270	Computer output indicating that global optimum information will follow.
271-277	Computer output indicating the global optimum search variable values. Again, as only one local optimum was calculated, the global optimum equals the local optimum.
278	Computer output indicating that .13227 is the global optimum's function value.
279	Computer output indicating that 33.919 seconds of central processor computer time was needed to perform the global search.
280, 281	Computer output indicating that the computer resources used in the program cost approximately $1.54 and consists of $.83 for central processor use, $.03 for peripheral processor use, and $.68 for central memory use.
282	Computer output indicating that one local search was conducted for the global search.
283	Computer output indicating that one global search was investigated with the program.
284-293	Computer output indicating the results of sensitivity calculations on the entire input variable set at the global optimum variable values.
295	User input specifying an exit from the program is desired.
296	Computer output summarising that one problem was investigated with the subcomponent.
297-299	Computer output indicating the end of the optimization subcomponent's execution.
300	Computer output indicating that the CDC 6500 is ready for additional user commands.

III. PROGRAM DOCUMENTATION

SYSOPT Simulation Programming

In order to correctly interact with and guarantee proper data flows within the SYSOPT program, the simulation model subroutine, SIM, must be in a specific form. It is assumed that the simulation whose behavior is to be studied was constructed using the SIMEX1 FORTRAN executive program described in Reference 9. The SIMEX1 executive routine provides both an organized format for constructing the simulation and convenient output for studying model behavior in addition to permitting multiple runs of the simulation with different values assigned to selected model variables on each run. If the simulation does not incorporate the SIMEX1 format, then the programming changes outlined in the next section should be made to adapt a model for SYSOPT execution.

Given a simulation model programmed with SIMEX1, it is a relatively simple task to adapt the model for use in the component. Only a minimal acquaintance with FORTRAN is necessary to insure that the adapted simulation will properly interact with SYSOPT. In addition, the programming modifications described in this section do not change the input/output capabilities of the original simulation as outlined in Reference 9. Thus, without further programming changes, it is possible to either execute multiple runs of the simulation, normal SIMEX1 study; or interact with the component, SYSOPT model optimization.

Figure III-1 contains an annotated computer program having the proper form for the SIM subroutine. Here, starred (*) cards represent additions to the SIMEX1 executive subroutine. As can be seen from the program, no reprogramming of the original simulation is necessary beyond the insertion

Figure III-1. Adaption of SIMEX Simulation

```
CARD
NUMBER                                                          *ADDITION
                                FORTRAN STATEMENT                TO SIMEX

  1          SUBROUTINE SIM(XIN,COUT,Z,TD,P,INT,IPLOT,TSAVE,     *these cards
  2               NTS,NSET,ISWITCH,NVAR,MOUTS,NPOBJ,P2,NTIMES,IS) *convert the
  3          DIMENSION XIN(1),XOUT(1),P(1),TSAVE(NTS,1)          *simulation
  4          REAL ....                                            program into
  5          COMMON/BLOCK/...                                     a subroutine
  6          COMMON/BLOCK1/...
  7          ....
  8          COMMON/BLOCKq/...
  9          DATA NAMVAR/.../
 10          DATA NVAR/.../
 11          IF(ISWITCH.EQ.1) GO TO 10001                        *
 12          READ 900, NRUN
 13   900    FORMAT(I1)
 14   10001  IF(ISWITCH.NE.1) GO TO 10002                        *these cards
 15          NRUN=1                                              *initialize
 16          IINT=1                                              *SIMEX run
 17          JTS=1                                               *parameters
 18   10002  CONTINUE                                            *
 19          DO 500 IRUN=1,NRUN
 20          IC1=
 21          IC2=
 22          ...
 23          VMP1=
 24          VMP2=
 25          ...
 26          DUR=
 27          DT=
 28          DETPRT=
 29          SELPRT=
 30          BEGPRT=
 31          PRTCHG=
 32          PRTVL1=
 33          PRTVL2=
 34          IF(ISWITCH.NE.1) GO TO 10003                        *these cards
 35          VAR1=XIN(1)                                         *assign the
 36          VAR2=XIN(2)                                         *proper values
 37          ...                                                 *to the input
 38          VARNVAR=XIN(NVAR)                                   *variables of
 39          GO TO 10004                                         *the
 40   10003  CONTINUE                                            *simulation
 41          PRINT 901,IRUN
 42          CALL NAMLST (NAMVAR, NVAR, ERROR)
 43          IF(ERROR.NE.0)STOP
 44   10004  CONTINUE
 45          NITER=DUR/DT+.0001
 46          IF(DETPRT.EQ.0)SELPRT=1
```

Figure III-1 (cont'd)

```
CARD
NUMBER                    FORTRAN STATEMENT                                  *ADDITION
                                                                              TO SIMEX
 47          T=0.
 48          PRTIME=BEGPRT
 49          PRTVL=PRTVL1
 50          TD=DT
 51          DO 400 ITER=1,NITER
 52          T=T+DT
 53          Z=T                                                            *
 54          ...call routines to compute one cycle of the simulation model
 55          IF(ISWITCH.NE.1)GO TO 10007                                    *these cards
 56          XOUT(1)=MOUT1                                                  *(1)assign
 57          XOUT(2)=MOUT2                                                  *values to
 58          ...                                                            *the output
 59          XOUT(MOUTS)=MOUTMOUTS                                          *variables,
 60          IF(IS(1).NE.0.AND.IPLOT.EQ.0)CALL NNN(T,DT,P2,XIN,XOUT,        *(2) update
 61               P,NVAR,MOUTS,NPOBJ,NSET)                                  *the function
 62          IF(IS(1).NE.0.AND.IPLOT.EQ.0)GO TO 10006                       *N in the cost
 63          IINT=IINT+1                                                    *functional,
 64          IF(IINT.NE.INT)GO TO 10006                                     *(3)write the
 65          IINT=0                                                         *output var-
 66          IF(IPLOT.EQ.1)WRITE(99)T,(XOUT(J),J=1,MOUTS)                   *iables on a
 67          IF(IPLOT.EQ.2)WRITE(99)T,(XOUT(J),J=1,MOUTS),                  *file for
 68               (TSAVE(J,JTS),J=1,NTS)                                    *plotting in
 69          IF(IPLOT.EQ.2)JTS=JTS+1                                        *the pattern
 70          IF(IPLOT.NE.0)NTIMES=NTIMES+1                                  *recognition
 71          IF(IPLOT.NE.0)GO TO 10006                                      *sub-component
 72          DO 10005 JJ=1,NTS                                              *and (4) up-
 73    10005 TSAVE(JJ,JTS)=XOUT(JJ)                                         *date counter
 74          JTS=JTS+1                                                      *variables.
 75    10006 CONTINUE                                                       *
 76          GO TO 400                                                      *
 77    10007 CONTINUE                                                       *
 78          IF(T+1E-08.LT.PRTIME)GO TO 200
 79          IF(T.EQ.PRTCHG)PRTVL=PRTVL2
 80          PRTIME=T+PRTVL
 81          IF(SELPRT.EQ.0)GO TO 100
 82          PRINT XXX,V1,V2...
 83    100   IF(DETPRT.EQ.0)GO TO 200
 84          PRINT XXX,VA,VB...
 85    200   CONTINUE
 86    400   CONTINUE
 87    500   CONTINUE
 88    XXX   FORMAT(...)
 89          ...
 90          RETURN
 91          END
```

of 32 + MOUTS + NVAR cards, with MOUTS being the number of model outputs and NVAR being the number of input variables in the simulation. These additional FORTRAN cards perform three principal functions: converting the simulation program into a subroutine with the proper argument list, assigning SYSOPT variable values to the model inputs, and bypassing PRINT/READ simulation statements. If a simulation model is constructed using the modified SIMEX1 executive routine outlined in Figure III-1 and if the modifications are made so that all variable names and statement numbers are unique, then SIM will properly interact with both the pattern recognition and optimization subcomponents.

The modifications that must be made to a simulation program include seven FORTRAN statement groups. Using the card numbers shown in Figure III-1 for reference, these groups can be divided as follows: (1) 1,2,3; (2) 11,14; (3) 14,15,16,17,18; (4) 34-40,44; (5) 50,53; (6) 55-77; and (7) 90.

1. The first group of statements converts the original simulation program into a subroutine, SIM, with the proper argument list to interact with SYSOPT:

```
      SUBROUTINE SIM(XIN,XOUT,Z,TD,D,INT,IPLOT,TSAVE,
            NTS,NSET,ISWITCH,NVAR,MOUTS,NPOBJ,P2,NTIMES,IS)
      DIMENSION XIN(1),XOUT(1),P(1),TSAVE(NTS,1)
```

2. The second group of statements bypasses the READ statement for NRUN, the number of multiple simulation runs, when executing SYSOPT in either the pattern recognition or optimization mode:

```
      IF(ISWITCH.EQ.1) GO TO 10001
10001 IF(ISWITCH.NE.1) GO TO 10002
```

3. The third modification group consisting of five statements initializing NRUN, the number of multiple simulation runs; IINT, a counter for determining the simulation output interval; and JTS, another counter for an output array of the model:

```
10001 IF(ISWITCH.NE.1) GO TO 10002
      NRUN = 1
      IINT = 1
      JTS = 1
10002 CONTINUE
```

4. Card statements 34-40 and 44 assign the input variable values from SYSOPT to the corresponding variables in the simulation:

```
      IF(ISWITCH.NE.1) GO TO 10003
      VAR1 = XIN(1)
      VAR2 = XIN(2)
      ...
      VARNVAR = XIN(NVAR)
      GO TO 10004
10003 CONTINUE
10004 CONTINUE
```

Where NVAR is the number of input variables in the simulation and VAR_1, VAR_2, ..., VAR_{NVAR} are the NVAR input variable names.

5. Statements 50 and 53 provide transfer of the DT step size and the final time value from the model to the SYSOPT program:

```
      TD = DT
      Z = T
```

6. The sixth modification group, consisting of statements 55-70, performs several functions: (1) assigning model output values to SYSOPT variables, (2) calling subroutine NNN to update an integral objective function, (3) selecting the time step for which output of print and/or plots of time series in the pattern recognition sub-component is desired, (4) writing on a local file, TAPE99, the model output values for appropriate simulation times, and (5) storing model output values for use in calculating a least squares error criterion for a parameter estimation problem:

```
      IF(ISWITCH.NE.1) GO TO 10007
      XOUT(1) = MOUT1
      XOUT(2) = MOUT2
      ...
      XOUT(MOUTS) = MOUTMOUTS
      IF(IS(1).NE.0.AND.IPLOT.EQ.0) CALL NNN(T,DT,P2,XIN,
            XOUT,P,NVAR,MOUTS,NPOBJ,NSET)
      IF(IS(1).NE.0.AND.IPLOT.EQ.0) GO TO 10006
      IINT = IINT+1
      IF(IINT.NE.INT) GO TO 10006
      IINT = 0
```

```
      IF(IPLOT.EQ.1)WRITE(99)T,(XOUT(J),J = 1,MOUTS)
      IF(IPLOT.EQ.2)WRITE(99)T,(XOUT(J),J = 1,MOUTS),
     1      (TSAVE(J,JTS),J = 1,NTS)
      IF(IPLOT.EQ.2)JTS = JTS+1
      IF(IPLOT.NE.0)NTIMES = NTIMES+1
      IF(IPLOT.NE.0)GO TO 10006
      DO 10005 J = 1,NTS
10005 TSAVE(JJ,JTS) = XOUT(JJ)
      JTS = JTS+1
10006 CONTINUE
      GO TO 400
10007 CONTINUE
```

7. Finally, statement 90 transfers control of the execution from the simulation SIM to the SYSOPT program:

```
RETURN
```

An example of a SIMEX1 simulation constructed to interact with SYSOPT is given in Figure III-2. This simulation models the behavior of a single commodity economic market in which government can affect the price level by buying and selling the commodity. In the simulation there are NVAR = 8 model inputs representing economic parameters: C1, DEMC, SUPC, DELP, KRI, KPI, KP and KR. There are also MOUTS = 6 output variables representing significant economic measurable quantities: P, GP, GIMP, DEM, GINV, and PLGIMP. As can easily be verified using Figure 3, this simulation is in the proper form for interaction with SYSOPT. The only changes that have been made to the original program are those that are outlined here. In fact, SYSOPT has been used to solve parameter estimation problems with this particular simulation.

The next section of the guide describes how arbitrary objective functions can be input to SYSOPT.

Figure III-2. An Example of a SIMEX1 Simulation Constructed to Interact with SYSOPT

```
SUBROUTINE  SIM                                          CDC 6500 FTN V3.0-P380 OPT=1   01/09/75
      SUBROUTINE SIM(XIN,XOUT,Z,TD,PP,INT,IPLOT,TSAVE,NTS,NSET,ISWITCH,
     1  NVAR,MOUTS,NPOBJ,P2,NTIMES,IS)
      DIMENSION XIN(1),XOUT(1),PP(1),TSAVE(NTS,1),IS(1)
      REAL KR,KP,KRI,KPI
      COMMON /BLOCK/  DUR,DT,DETPRT,SELPRT,BEGPRT,PRTCHG,PRTVL1,PRTVL2,
     1 C1,KR,KP,S1,KRI,KPI,D1,PD,GINVD,S0,D0,C2,PZERO
      COMMON/PRIVATE/DELP,CROUTPR,SUPPR,KD,DEM,IDTPR,DEMC,SUPC
      COMMON/GOV/PWLD,DELI,KIMP,CROUTPB,GINV,GIMP,GCOST,TGCOST,IDTPB
     1  ,PLGIMP
      COMMON/SYSVAR/T,GP,P,PPR
      DIMENSION CROUTPB(6),CROUTPR(3)
      DIMENSION NAMVAR(21)
      DATA NAMVAR / 3HDUR,     2HDT,      6HDETPRT, 6HSELPRT, 6HBEGPRT,
     1              6HPRTCHG, 6HPRTVL1, 6HPRTVL2,
     2 2HC1,2HKR,2HKP,2HS1,3HKRI,3HKPI,2HD1,2HPD,5HGINVD,2HS0,2HD0,2HC2,
     2 5HPZERO/
      DATA NVR/21/
      NTIMES=0
      IF(ISWITCH.EQ.1) GO TO 10001
      READ900,NRUN
  900 FORMAT(I2)
10001 IF(ISWITCH.NE.1) GO TO 10002
      NRUN=1

      IINT=0
      JTS=1
10002 CONTINUE
C
C          BEGIN RUN LOOP
C
      DO 500 IRUN=1,NRUN
C
C          INITIAL CONDITIONS
      PLGIMP=0.
      PZERO=100.
      SUPPR=100.
      TGCOST=0.
      GINV=50.
      GIMP=0.
      GP=0.
C
C
C          PARAMETERS
      SUPC=1.0
      DEMC=1.0
      KD=3
      KD=1
      DELP=1.

      C1=2.
      KR=.2

      S1=1.
      KRI=1.
      KPI=1.
      D1=-1.

      PD=90.
      S0=10.
      GINVD=50.
      D0=200.
      C2=10.

      DELI=0.4
      KIMP=6
      KIMP=3
      PWLD=105.
C
C          RUN PARAMETERS
C
      DUR=1.0

      DT=.05
      DETPRT = 1.
      SELPRT = 0.
      BEGPRT=.05
      PRTCHG = 100.
      PRTVL1=.05
      PRTVL2 = 5.
   30 CONTINUE
C
```

Figure III-2 (cont'd)

```
SUBROUTINE  SIM                                          CDC 6500 FTN V3.0-P380 OPT=1   01/09/75
                IF(ISWITCH.NE.1) GO TO 10003
                C1=XIN(1)
                DEMC=XIN(2)
                SUPC=XIN(3)
                DELP=XIN(4)
                KRI=XIN(5)
                KPI=XIN(6)

                KP=XIN(7)
                KR=XIN(8)
                GO TO 10004
          10003 CONTINUE
          C           READ RUN PARAMETERS
          C
                PRINT 901,IRUN
            901 FORMAT(37H1NON-DEFAULT PARAMETER VALUES FOR RUN,I2)
                CALL NAMLST(NAMVAR,NVR,ERROR)
                IF(ERROR.NE.0.) STOP
          10004 CONTINUE
                NITER=DUR/DT+.0001
                IF (DETPRT .EQ. 0.) SELPRT=1.
                T = 0.
                IDTPR=1.0+2.0*FLOAT(KD)*DT/DELP
                IDTPB=1.0+2.0*FLOAT(KIMP)*DT/DELI
                P=PZERO
                DO 1 I=1,KIMP
          1     CROUTPB(I)=0.
                DO 2 I=1,KD
          2     CROUTPR(I)=SUPPR/FLOAT(IDTPR)
                PRTIME = BEGPRT
                PRTVL = PRTVL1
          C
          C           BEGIN TIME LOOP
                TD=DT
          C
                DO 400 ITER=1,NITER
                T = T+DT
                Z=T
          C
                CALL PVSEC
                CALL GOVSEC
                IF(ISWITCH.NE.1) GO TO 10007
                XOUT(1)=P
                XOUT(2)=GP
                XOUT(3)=GIMP
                XOUT(4)=DEM
                XOUT(5)=GINV
                XOUT(6)=PLGIMP
                IF(IS(1).NE.0.AND .IPLOT.EQ.0) CALL NNN(T,DT,P2,XIN,XOUT,PP,NVAR,
               1  MOUTS,NPOBJ,NSET)
                IF(IS(1).NE.0.AND.IPLOT.EQ.0) GO TO 10006
                IINT=IINT+1
                IF(IINT.NE.INT) GO TO 10006
                IINT=0

                IF(IPLOT.EQ.1) WRITE(99)T,(XOUT(J),J=1,MOUTS)
                IF(IPLOT.EQ.2) WRITE(99)T,(XOUT(J),J=1,MOUTS),(TSAVE(J,JTS),J=1,
               1  NTS)
                IF(IPLOT.EQ.2) JTS=JTS+1
                IF(IPLOT.NE.0) NTIMES=NTIMES+1
                IF(IPLOT.NE.0) GO TO 10006
                DO 10005 JJ=1,NTS
          10005 TSAVE(JJ,JTS)=XOUT(JJ)
                JTS=JTS+1
          10006 CONTINUE
                GO TO 400
          10007 CONTINUE
          C
                IF(T+1E-08.LT.PRTIME) GO TO 200
          C
          C           PRINT RESULTS
          C
                IF(T.EQ.PRTCHG) PRTVL = PRTVL2
                PRTIME = T + PRTVL
                IF(SELPRT.EQ.0.)GO TO 100
          C
          C           SELECTED OUTPUT
          C

                PRINT903,T
                PRINT 905,P,PD,DEM,GIMP,GINV,GINVD,GCOST,TGCOST
```

Figure III-2 (cont'd)


```
SUBROUTINE  SIM                                          CDC 6500 FTN V3.0-P380 OPT=1   01/09/75
      PRINT 906,GP,PPR
C
  100 IF(DETPRT.EQ.0.) GO TO 200
C
C         DETAILED OUTPUT
C
      PRINT920,T
      PRINT 905,P,PD,DEM,GIMP,GINV,GINVD,GCOST,TGCOST
      PRINT 906,GP,PPR
C
C
  200 CONTINUE
  400 CONTINUE
  500 CONTINUE
      IF(ISWITCH.NE.1) PRINT 999
C
C         FORMAT STATEMENTS
C
  903 FORMAT(34H3SELECTED OUTPUT VARIABLES AT TIME,F6.2)
  920 FORMAT(34H3DETAILED OUTPUT VARIABLES AT TIME,F6.2)
  905 FORMAT(1H0,9X,1HP,11X,2HPD,10X,3HDEM,9X,4HGIMP,8X,4HGINV,8X,
     1   5HGINVD,7X,5HGCOST,7X,6HTGCOST/1H0,8X,8E12.4)
  906 FORMAT(1H0,8X,2HGP,10X,3HPPR/1H0,8X,2E12.4//)
  999 FORMAT(1H1)
      RETURN
      END
```

```
SUBROUTINE  GOVSEC                                       CDC 6500 FTN V3.0-P380 OPT=1   01/09/75
      SUBROUTINE GOVSEC
      REAL KR,KP,KRI,KPI,INT
      COMMON /BLOCK/  DUR,DT,DETPRT,SELPRT,BEGPRT,PRTCHG,PRTVL1,PRTVL2,
     1 C1,KR,KP,S1,KRI,KPI,D1,PD,GINVD,S0,D0,C2,PZERO
      COMMON/GOV/PWLD,DELI,KIMP,CROUTPB,GINV,GIMP,GCOST,TGCOST,IDTPB
     1   ,PLGIMP
      COMMON/SYSVAR/T,GP,P,PPR
      DIMENSION CROUTPB(6)
      GPPR=KP*(PD-P)-KR*PPR
      GP=GPPR
      IF(GPPR.LT.0..AND.GINV.LE.0.) GP=0.
      E=(GINVD-GINV)*KPI
      INT=E-KRI*(GIMP+GP)
      GIMPO=INT
      IF(INT.LT.0.) GIMPO=0.
      GCOST=C2*GINV+PWLD*GIMP+GP*P
      GINV=GINV+DT*(GIMP+GP)
      TGCOST=TGCOST+DT*GCOST
      CALL DELDT(GIMPO,GIMP,CROUTPB,DELI,IDTPB,DT,KIMP)
      PLGIMP=PLGIMP+DT*(GIMPO-GIMP)
      RETURN
      END
```

Figure III-2 (cont'd)

SUBROUTINE PVSEC CDC 6500 FTN V3.0-P380 OPT=1 01/09/75

```
      SUBROUTINE PVSEC
      REAL KR,KP,KRI,KPI
      COMMON /BLOCK/  DUR,DT,DETPRT,SELPRT,BEGPRT,PRTCHG,PRTVL1,PRTVL2,
     1 C1,KR,KP,S1,KRI,KPI,D1,PD,GINVD,S0,D0,C2,PZERO
      COMMON/PRIVATE/DELP,CROUTPR,SUPPR,KD,DEM,IDTPR,DEMC,SUPC
      COMMON/SYSVAR/T,GP,P,PPR
      DIMENSION CROUTPR(3)
      DIMENSION ARG(6)
      DIMENSION DVAL(6)
      DIMENSION SVAL(6)
      DATA KPTS/6/
      DATA (ARG(J),J=1,6)/10.,20.,60.,100.,150.,200./
      DATA (DVAL(J),J=1,6)/300.,170.,120.,100.,85.,75./
      DATA (SVAL(J),J=1,6)/0.,50.,85.,100.,110.,115./
      DEM=DEMC*TABLI(DVAL,ARG,P,KPTS)
      SUP=SUPPR+S0*SIN(6.283*T)
      SUPD=SUPC*TABLI(SVAL,ARG,P,KPTS)
      PPR=C1*(DEM+GP-SUP)
      P=P+DT*PPR
      CALL DELDT(SUPD,SUPPR,CROUTPR,DELP,IDTPR,DT,KD)
      RETURN
      END
```

SUBROUTINE DELDT CDC 6500 FTN V3.0-P380 OPT=1 01/09/75

```
      SUBROUTINE DELDT(RINR,ROUTR,CROUTR,DEL,IDT,DT,K)
      DIMENSION CROUTR(1)
      DEL1=(DEL*FLOAT(IDT))/(FLOAT(K)*DT)
      ROUTR=0.
      DO 2 J=1,IDT
      RIN=RINR/FLOAT(IDT)
      DO 1 I=1,K
      ABC=CROUTR(I)
      CROUTR(I)=ABC+(RIN-ABC)/DEL1
      RIN=ABC
      ROUTR=ROUTR+CROUTR(K)
      RETURN
      END
```

```
SUBROUTINE  NAMLST                                    CDC 6500 FTN V3.0-P380 OPT=1  01/09/75
                    SUBROUTINE NAMLST(NAMVAR, NVAR, ERROR)                                  240
                    INTEGER BLANK,DOLLAR                                                    250
                    DIMENSION DUMMY(1),IDUMMY(1),NAMVAR(1)                                  260
                    DIMENSION INT(4),LAST(4),NAME(4),VALUE(4)                               270
                    COMMON /BLOCK/ DUMMY                                                    275
                    EQUIVALENCE (DUMMY(1),IDUMMY(1))                                        280
                    DATA BLANK,DOLLAR / 1H , 1H$ /                                          290
                    ERROR=0.                                                                300
              10    READ 900,(NAME(I),INT(I),VALUE(I),LAST(I),I=1,4)
                    IF(EOF(1).NE.0.)RETURN
                    DO 50 I=1,4                                                             320
                    IF(NAME(I).EQ.BLANK)GO TO 45
                    DO 30 J=1,NVAR                                                          340
                    IF(NAME(I).EQ.NAMVAR(J))GO TO 35                                        350
              30    CONTINUE                                                                360
                    PRINT 901,NAME(I)
                    ERROR = 1.                                                              380
                    RETURN                                                                  390
              35    PRINT 902,NAME(I),VALUE(I)
                    IF(INT(I).NE.BLANK)GO TO 40
                    DUMMY(J) = VALUE(I)                                                     410
                    GO TO 45                                                                420
              40    IDUMMY(J) = VALUE(I)                                                    430
              45    IF(LAST(I).EQ.DOLLAR)RETURN                                             440
              50    CONTINUE                                                                450
                    GO TO 10                                                                460
             900    FORMAT(4(A6,1X,A1,E9.4,A1))                                             470
             901    FORMAT(24H0**ERROR** VARIABLE NAME,A7,10H NOT FOUND)                    480
             902    FORMAT(1H0,A6,3H = ,E12.4)
                    END                                                                     490
ARD NO.  SEVERITY          DIAGNOSTIC
  27        1      28 CD 27 EW.D OR DW.D DESCRIPTOR BAD FOR OUTPUT, W SHOULD SATISFY W-7 .GE. D
```

```
FUNCTION    TABLI                                     CDC 6500 FTN V3.0-P380 OPT=1  01/09/75
                    FUNCTION TABLI(VAL,ARG,DUMMY,K)
                    DIMENSION VAL(1),ARG(1)
                    DUM=AMAX1(AMIN1(DUMMY,ARG(K)),ARG(1))
                    DO 1 I=2,K
                    IF(DUM.GT.ARG(I)) GO TO 1
                    TABLI=(DUM-ARG(I-1))*(VAL(I)-VAL(I-1))/(ARG(I)-ARG(I-1))+VAL(I-1)
                    RETURN
              1     CONTINUE
                    RETURN
                    END
```

User Supplied Simulation Subroutines SIM, MMM, and NNN

As described in the previous section, a simulation built with the SIMEX1 executive routine can easily be modified to properly interact with the sub-components. However, with relatively little programming effort, virtually any existing or contemplated FORTRAN simulation can be adapted to interact with SYSOPT. There are six steps necessary to complete this process.

First, the simulation's main program must be converted into a subroutine compatible with the component. Essentially, only three cards are necessary to accomplish this conversion in FORTRAN.

```
(1)  SUBROUTINE SIM(XIN,XOUT,Z,TD,P,INT,IPLOT,TSAVE,NTS,NSET,ISWITCH,
          NVAR,MOUTS,NPOBJ,P2,NTIMES,IS)
(2)  DIMENSION XIN(1),XOUT(1),P(1),TSAVE(NTS,1)
(3)  RETURN
```

The program name card in the simulation must be replaced by the first card SUBROUTINE SIM(...). Next, the DIMENSION statement must be inserted between the SUBROUTINE SIM(...) card and the first executable statement. Finally the RETURN statement must be inserted immediately before the END statement in the original main program.

The second change that must be implemented is the suppression of all reading and printing of data from the simulation. This should be accomplished with conditional GO TO statements depending on the variable ISWITCH which takes on the value 1 only if SIM is being called from the component (except when normal simulation execution is specified from the pattern recognition sub-component). An example of this conditional GO TO is:

```
    IF(ISWITCH.EQ.1) GO TO 10
    READ 99, VAR1, VAR2, ..., VARN, read simulation input parameters
    PRINT 89, VAR2, VAR4          , print simulation input parameters
 10 CONTINUE
```

Note that in addition to skipping the input and output functions of the simulation, functions necessary for the adaptation of the program to interact with SYSOPT must likewise be bypassed in a normal simulation executive mode.

In order to allow SYSOPT to calculate values of the objective function, the input variable values must be communicated to the simulation model. Thus, the variable array XIN, in the SIM argument list, must assign values to the simulation input variables. The programming steps to accomplish this are inserted as close to the major time increment loop as possible so as to eliminate re-assignment of the input variable values by the simulation. If the simulation input variables are VAR_1, VAR_2, ..., VAR_{NVAR}, then the statements to perform these assignments are:

```
      IF(ISWITCH.NE.1) GO TO 11
      VAR1 = XIN(1)
      VAR2 = XIN(2)
      .
      .
      .
      VARNVAR = XIN (NVAR)
   11 CONTINUE
```

In addition to providing the external assignment of simulation input variables, the communication of output variables of the simulation and updating of any integral objective function components (subroutine NNN) must also be provided. This is accomplished by the following statements which are inserted in the simulation as near after the statements that compute one cycle (DT step) of the model:

```
      IF(ISWITCH.NE.1) GO TO 12
      XOUT(1) = MOUT1
      XOUT(2) = MOUT2
      .
      .
      .
      XOUT(MOUTS) = MOUTMOUTS
```

```
      IF(IS(1).NE.0.AND.IPLOT.EQ.0)
          CALL NNN(T,DT,P2,XIN,XOUT,P,NVAR,MOUTS,NPOBJ,NSET)
   12 CONTINUE
```

where $MOUT_1$, $MOUT_2$, ..., $MOUT_{MOUTS}$ are the simulation output variables,

IS(1) = 0 for a parameter estimation problem

= 1 for an optimal control problem

IPLOT ≠ 0 for plotting model outputs (through the pattern recognition sub-component)

= 0 otherwise

The fifth change needed to make a simulation compatible with SYSOPT is the addition of FORTRAN statements to write tapes (disc files) for use as plot input in the pattern recognition sub-component. This interaction between the pattern recognition sub-component and subroutine SIM, through TAPE99, is accomplished with the following statements placed after the output variable assignment statements:

```
      IF(IS(1).NE.0.AND.IPLOT.EQ.0) GO TO 13
      IINT = IINT+1
      IF(IINT.NE.INT) GO TO 13
      IINT = 0
      IF(IPLOT.EQ.1)WRITE(99)T,(XOUT(J),J=1,MOUTS)
      IF(IPLOT.EQ.2)WRITE(99)T,(XOUT(J)J=1,MOUTS),(TSAVE(J,JTS),J=1,NTS)
      IF(IPLOT.EQ.2)JTS = JTS+1
      IF(IPLOT.NE.0)NTINES = NTIMES+1
      IF(IPLOT.NE.0) GO TO 13
      DO 14 JJ = 1,NTS
   14 TSAVE(JJ,JTS) = XOUT(JJ)
      JTS = JTS+1
   13 CONTINUE
```

Also set JTS = 1 and NTIMES = 0 initially.

Included in this last set of FORTRAN changes to the simulation subroutine is a DO loop which assigns XOUT(JJ) to TSAVE(JJ,JTS). For parameter estimation and plotting purposes this matches the real world time series with the proper time sequenced model outputs.

JTS is a counter for an index of the TSAVE array

IPLOT = 0 for no plotting of model ouptuts

= 1 for plotting of only model outputs

= 2 for plotting of model outputs and real world (or smoothed) time series

INT is an output index such that every DT*INT time units the value of TSAVE(JJ,JTS) is assigned the value of the model outputs are stored for future plotting or comparison with real world time series.

If these relatively simple programming changes are made to a simulation, being careful not to duplicate symbolic names in the simulation subroutine, then the resulting continuous simulation model will properly interact with SYSOPT.

Finally, the user supplied subroutines NNN and MMM specify the form for the objective function in a policy decision problem where

$$J[\underline{a}, \underline{u}, T] = P1(\underline{y}(T), T, \underline{p}) + \int_{s=0}^{T} P2(\underline{y}(s), u(s), s, \underline{p})ds$$

The programmed form for these subroutines is

```
      SUBROUTINE MMM(T, DT, P1, XIN, XOUT, P, NVAR, MOUTS, NPOBJ)
      DIMENSION XIN(NVAR), XOUT(MOUTS), P(NPOBJ)
      P1 =  user supplied function of XIN, XOUT, T, and P to penalize
            or reward final time behavior
      RETURN
      END
```

and

```
      SUBROUTINE NNN(T, DT, P2, XIN, XOUT, P, NVAR, MOUTS, NPOBJ, NSET)
      DIMENSION XIN(NVAR), XOUT(MOUTS), P(NPOBJ)
      IF(NSET.EQ.0) GO TO 1
      P2 + P2 + DT* (user supplied function  of XIN, XOUT, P, and T to
                     penalize or reward behavior over the interval
                    [0,T])
      RETURN
1     P2=0.
      RETURN
      END
```

The variable array P in both the subroutines MMM and NNN is the set of parameters that is specified in the component (DATA subroutines) to allow changes in the objective function to be made during execution.

It is important for proper operation of the component that the user supplied subroutines be in the exact form specified here. The programming necessary for the subroutines is certainly simple and can be accomplished by anyone who is at all familiar with FORTRAN.

Variable Dimensioning

Variable dimensioning, or dynamic memory, is an important capability in programs using large amounts of computer core storage. This is particularly true when a program contains arrays whose sizes depend upon input variable values. Both to eliminate the need for reprogramming and to minimize the use of core storage, a variable dimensioning capability is programmed into the pattern recognition and optimization sub-components of SYSOPT. This section describes the implementation of this useful capability.

There are several requirements of the variable dimensioning capability that result from the functions it must perform in the sub-components. These include variable dimensioning for both real and integer arrays, reducing and enlarging the size of arrays during the program execution, and allowing input of array sizes in an interactive program execution mode.

The implementation of this capability involves six programming phases:

(1) Dimension BLANK COMMON to one storage location in the main program:

```
COMMON COM(1)
```

(2) Compute the size of fixed storage needed for the program execution using subroutine NEXT to simplify the calculations: For K arrays having sizes N1, N2, ..., NK

```
LAST = 1
L(1) = NEXT(LAST,N1)
L(2) = NEXT(LAST,N2)
.        .     .
.        .     .
.        .     .
L(K) = NEXT(LAST,NK)
```

where L(I)+J-1 is the location in the COM array of the Jth element of the Ith array and LAST is the size of core storage needed for the K arrays.

(3) Extend the size of BLANK COMMON in the main program

```
CALL SETFL(COM(LAST))
```

where SETFL is a CDC 6500 library function that extends (or reduces) the length of BLANK COMMON to the size LAST.

(4) Call the secondary subroutines in which the program dimensioned arrays are to be used passing both arrays and array sizes as arguments:

```
Call:  CALL SUB(COM(L(1)), COM(L(2)), ..., COM(L(K)),
            COM(LAST), N1, N2, ..., NK)

Subroutine:  SUBROUTINE SUB(A1, A2, ..., AK, COM(1), N1, N2, ..., NK)
             DIMENSION A1(N1), A2(N2), ..., AK(NK), COM(1)
             REAL ... list of real arrays
             INTEGER ... list of integer arrays
```

(5) In SUB, extend or reduce BLANK COMMON as necessary for arrays whose sizes are dependent upon variables input in SUB. These arrays are either for use internal to SUB (see (6)) or in other subroutines called from SUB. To accomplish this, NEXT and SETFL are used as needed.

(6) Because COM is a real array internal to SUB, it is necessary to use the subroutine CONV to convert the integers stored as reals in COM back into integers:

For the Jth element in the real array COM corresponding to location L(I).

```
CALL CONV(COM(L(I)+J-1),II)
```

assigns to the integer II the integer value stored as a real in COM.

Although this variable dimensioning capability is programmed for the CDC 6500 computer, programming this capability into IBM or other computers should not be difficult. Most available computers have some type of dynamic storage allocation features that can be used to implement a variable dimensioning scheme.

Main Program and Subroutine Descriptions

The subroutines in the pattern recognition sub-component and optimization sub-component of SYSOPT are described in this section. These descriptions consist of variable definition lists, program listings and discussions of subroutine application, use, and special features. The intent of these descriptions is to give a FORTRAN programmer sufficient information to understand the execution of the individual subroutines. With the Subroutine Control Flow Charts and these descriptions, a programmer can make minor changes in the SYSOPT program in order to suit individual needs, e.g., different computer installations.

Although the general discussions of subroutine execution are self-explanatory, the format used in the variable definition lists needs explanation. As there are many variables in the component that are common to different subroutines, variables with multiple references are generally defined in at most two subroutine descriptions: (1) in MAINOPT(MAINPR), SUBOPT(SUBPR), or the DATA subroutines, and (2) in subroutines in which the variable is most frequently used. Thus, there are numerous cross-references between the variable lists. In addition, array definitions are often made by reference to individual elements.

While there is no substitute for actual programming and use of a complex program, these subroutine descriptions can be a valuable aid for a programmer in understanding the operation, use, and execution of SYSOPT.

Pattern Recognition Sub-Component

Program MAINPR

Program MAINPR, the main program in the pattern recognition sub-component, inputs, in either a batch or an interactive mode, the variable values that fix array sizes which are necessary for proper component execution. As arrays in the program are formed as parts of one large array, COM, stored in BLANK COMMON, space is allocated in MAINPR as required for a particular problem or simulation being studied. Fixing the exact amount of core storage needed for a specific problem results in both time and cost savings to the user as recompilation and reprogramming of SYSOPT is not necessary to properly interact with different simulations.

During the execution of MAINPR, variables can be input with the SYSOPT interactive capability (see the "SYSOPT Interaction and Output" section of this guide). On the other hand, with the use of the Interaction/Output Flow Chart, a detailed data deck can be structured that will eliminate the need for interaction, if so desired. Therefore, to match the user's needs, the pattern recognition sub-component allows either direct interaction with the executing program or independent execution for a well-defined problem.

MAINPR also transfers execution of SYSOPT to the subroutine SUBPR. In accomplishing this, specific elements of the COM array are associated with specific variable arrays for the remainder of the execution of the program. On return from SUBPR a message

. . . . . END OF PROGRAM . . . . .

is printed on the output file to indicate the end of program execution. At that time, should the user wish to continue execution of SYSOPT with the optimization sub-component, it is necessary to load and execute program MAINOPT.

Significant Component Variables

NTS	contains the number of variables represented by the time series TTS in a parameter estimation problem. In an optimal control problem, set NTS = MOUTS.
NDP	contains the maximum number of data points in the NTS time series. In an optimal control problem set NDP = 1.
NVAR	contains the number of input parameters or control variables, XIN(1), XIN(2), ..., XIN(NVAR) in the simulation SIM.
MOUTS	contains the number of output variables, XOUT(1), ..., XOUT(MOUTS), in the simulation SIM.
COM	is an array in blank common containing all the program's arrays whose sizes depend on the values of the input variables.
LA	contains a pointer (reference location in the COM array) to the NAME array.
LB	contains a pointer to the XIN array.
LC	contains a pointer to the XOUT array.
LD	contains a pointer to the ONAMES array.
LE	contains a pointer to the PNAMES array.
LF	contains a pointer to the NTSMO array.
LG	contains a pointer to the TTS array.
LH	contains a pointer to the TSAVE array.
LI	contains a pointer to the STS array.
LJ	contains a pointer to the ETS array.
LK	contains a pointer to the INDVAR array.
LL	contains a pointer to the TS array.
LM	contains a pointer to the IS array.
LN	contains a pointer to the TIP array.
LO	contains a pointer to the NDPTS array.
LAST	contains the current length of the COM array.

```
      PROGRAM MAINPR(TAPE98=65,TAPE99=65,TAPE6=65,INPUT=65,OUTPUT=65,
     1   TAPE97=65,TAPE1=INPUT)

            TITLE-MAIN PROGRAM FOR THE PATTERN RECOGNITION SUB-COMPONENT
                OF SYSOPT

            PURPOSE-INPUT OF VARIABLES FIXING SIZE OF SUB-COMPONENT
                ARRAYS, EXTENDING BLANK COMMON TO MAKE ROOM FOR THESE
                ARRAYS AND TRANSFER OF PROGRAM EXECUTION TO THE SUBPR
                SUBROUTINE

            FUNCTIONS OR
            SUBROUTINES CALLED-NEXT, SETFL, SUBPR

      COMMON COM(1)
            INPUT AND COMPUTE SIZES OF PROBLEM DEPENDENT ARRAYS
      PRINT 999
      READ 899,NTS
      PRINT 998
      READ 899,NDP
      PRINT 997
      READ 899,NVAR
      PRINT 996
      READ 899,MOUTS
      NIND=4*NVAR+1
      NTD=NTS*NDP
      NTD2=2*NTD
      NTW=NTS+MOUTS
            COMPUTE SIZE OF EXTENDED COMMON BY UPDATING LAST WITH CALLS
            TO NEXT
      LAST=1
      LA=NEXT(LAST,NVAR)
      LB=NEXT(LAST,NVAR)
      LC=NEXT(LAST,NTW)
      LD=NEXT(LAST,MOUTS)
      LE=NEXT(LAST,NTW)
      LF=NEXT(LAST,MOUTS)
      LG=NEXT(LAST,NTD)
      LH=NEXT(LAST,NTD)
      LI=NEXT(LAST,NDP)
      LJ=NEXT(LAST,NDP)
      LK=NEXT(LAST,NIND)
      LL=NEXT(LAST,NTD2)
      LM=NEXT(LAST,20)
      LN=NEXT(LAST,NTS)
      LO=NEXT(LAST,NTS)
            SETFL (A CDC ROUTINE) EXTENDS BLANK COMMON TO #LAST( WORDS.
                IN EFFECT CALL SETFL (COM(LAST)) HAS THE SAME RESULT
                AS REPLACING
                COMMON COM(1)
                      BY
                COMMON COM(LAST)
      CALL SETFL(COM(LAST))
            TRANSFER CONTROL OF PROGRAM EXECUTION TO SUBROUTINE SUBPR,
                PASSING COM ELEMENTS TO ASSOCIATE WITH FIRST ELEMENTS
                OF SUB-COMPONENT ARRAYS IN SUBPR
      CALL SUBPR(COM(LA),COM(LB),COM(LC),COM(LD),COM(LE),COM(LF),COM(LG)
     1,COM(LH),COM(LI),COM(LJ),COM(LK),COM(LL),COM(LM),COM(LN),COM(LO),
     2NTS,NTW,NDP,NVAR,MOUTS,NIND)
      PRINT 995
899   FORMAT(I3)
999   FORMAT(* TYPE IN THE NUMBER OF TIME SERIES-NTS (I3)*)
998   FORMAT(* TYPE IN THE MAXIMUM NUMBER OF DATA POINTS-NDP (I3)*)
997   FORMAT(* TYPE IN THE NUMBER OF INPUT VARIABLES-NVAR (I3)*)
996   FORMAT(* TYPE IN THE NUMBER OF MODEL OUTPUTS-MOUTS (I3)*)
995   FORMAT(* .....END OF PROGRAM.....*)
      END
```

Subroutine SUBPR

This subroutine handles all the necessary input of simulation model parameters, decisions based on interactive variable values, and logic execution of the sub-component. Calls to subroutines that smooth the real world time series, print and/or plot time series, and calculate model outputs are all made from SUBPR.

The flow chart of the pattern recognition sub-componen in the "Subroutine Control Flow Chart" is essentially of the structure and logic of this subroutine.

Through input of variables to SUBPR, there are many data manipulation and plot/print options available to the user. For example, in the sub-component it is possible to compare in either a graphical or tabular manner real world data, model outputs, and smoothed time series. Thus, decisions can be made as to whether or not to use the actual or smoothed time series for further investigation with the optimization sub-component. In either a parameter estimation or policy decision problem the graphical and tabular output can be used to examine the behavior of the model to determine whether the model exhibits reasonable behavior and also to isolate feasible regions of input variable values for further study.

Included in SUBPR is the option of writing on a permanent file in the computer, system variables such as input and output variable names, and real world and smoothed time series. This file can then be stored for later use with the optimization sub-component, thus minimizing the input of duplicate information to the different sections. For further information on this capability see the section "SYSOPT Use of Input-Output Tapes" in a later section of this guide.

There are six major sub-component functions that are either executed or directed from SUBPR. These are: (1) Input of variable and problem description parameters, (2) Smoothing of real world time series (if desired) by an appropriate linear or quadratic polynomial filter, (3) Generation of model outputs for either comparison with the smoothed real world time series or for model behavioral studies, (4) Execution of the simulation SIM in a normal mode, (5) Generation of the local computer file for information transfer between the pattern recognition and optimization sub-components, and (6) Plotting and/or a printing of real world time series, smoothed real world time series, and model outputs. These functions are performed in the order implied by the "System Interaction/Output Flow Chart" found in the User's section of this guide. Upon completion of the execution of SUBPR, control is returned to MAINPR.

Subroutine Argument List

NAME(J)	contains the name of the Jth input parameter or control variable (maximum of six characters). J = 1,NVAR.
XIN(J)	contains the current value of the Jth input variable. J = 1,NVAR.
XOUT(J)	contains the current value of the Jth output variable. XOUT is updated every DT time step in the simulation SIM. J = 1,MOUTS. In addition XOUT is used for temporary storage in the subroutine PLOT. J = 1,MOUTS+NTS.
ONAMES(J)	contains the name of the Jth output variable. J = 1,MOUTS.
PNAMES(J)	contains the name of the Jth variable that is to be printed/plotted with the PLOT subroutine. J = 1,NTS+MOUTS. Variable names A, B, C, D, ... correspond to the smoothed time series corresponding to XIN(1), XIN(2), XIN(3), XIN(4), ...
NTSMO(J)	contains the number of the model output corresponding to the Jth time series. This array is used for selecting subsets of model outputs and time series for printing/plotting. J = 1,MOUTS.
TTS(J,JJ)	contains the JJ^{th} element of the Jth time series for a parameter estimation problem. J = 1,NTS. JJ = 1,NDPTS(J).
TSAVE(J,JJ)	contains the model (SIM) output corresponding to the JJ^{th} point of the Jth time series. J = 1,NTS. JJ = 1,NDPTS(J).
STS(J)	contains the Jth point of a real world time series that is to be smoothed by the polynomial filter subroutine SMOOTH (temporary storage).
ETS(J)	contains the Jth point of a smoothed real world time series obtained by smoothing STS in SMOOTH (temporary storage).

INDVAR — contains the subscripts of output variables that are to be printed/plotted with subroutine PLOT. In particular

INDVAR(1) = m_1
INDVAR(2) = $-m_1$
INDVAR(3) = m_2
INDVAR(4) = $-m_2$
. .
. .
. .
INDVAR(2J-1) = m_J
INDVAR(2J) = $-m_J$
INDVAR(2J+1) = 0

causes the variables corresponding to the $m_1, m_2, \ldots, m_J$ subscripts to be printed and/or plotted.
Dimensioned to 4*NVAR+1.

TS(1,J,JJ) — contains the JJ^{th} element of the Jth time series (smoothed real world or actual real world) that is to be input into the optimization sub-component for solving a parameter estimation problem.
J = 1,NTS.
JJ = 1,NDPTS(J).

IS — is an integer array containing component control variables affecting execution paths of the pattern recognition sub-component.

TIP(J) — contains the initial point (if known) of the Jth time series in a parameter estimation problem.
J = 1,NTS.

NDPTS(J) — contains the number of data points in the Jth time series for a parameter estimation problem.
J = 1,NTS.

NTS — defined in MAINPR variable lists

NTW — contains MOUTS+NTS

NDP — defined in MAINPR variable lists

NVAR — defined in MAINPR variable lists

MOUTS — defined in MAINPR variable lists

NIND — contains 4*NVAR+1

Additional Significant Component Variables

KEY — = Ø if there was no smoothing of the real world time series

≠ Ø if there was any smoothing, linear or quadratic

IPLOT	= Ø for normal objective function evaluation (see optimization sub-component subroutine FUNC)
	≠ Ø for writing model outputs on a computer tape for plotting
ISWITCH	= 1 if the simulation model is to be used for objective function evaluation or plotting
	≠ 1 for normal execution of the simulation model (print, interaction, etc.)
INT	contains the number of DT time steps corresponding to the real world data sampling interval in a parameter estimation problem or the desired sampling interval in a behavioral study of the model
ITT	contains a line printer parameter that causes blank lines to be printed so as to match time scales between a real world data study and a behavioral study of the model SIM. ITT = n causes the printer to skip n lines between plot points when plotting the real world or smoothed real world time series
DT	contains the basic time step size of the simulation model SIM
DUR	contains the length of the time horizon of the model SIM
IS(1)	= Ø for a parameter estimation problem and
	= 1 for an optimal control problem

```
      SUBROUTINE SUBPR(NAME,XIN,XOUT,ONAMES,PNAMES,NTSMO,TTS,TSAVE,STS,
     1ETS,INDVAR,TS,IS,TIP,NDPTS,NTS,NTW,NDP,NVAR,MOUTS,NIND)
C
C           TITLE-SUBPR IS THE MAIN SUBROUTINE IN THE PATTERN RECOGNITION
C               SUB-COMPONENT OF SYSOPT
C
C           PURPOSE-INPUT OF VARIABLE AND PROBLEM DESCRIPTION PARAMETERS,
C               CONTROL SMOOTHING OF TIME SERIES, CONTROL SIMULATION
C               EXECUTION, DIRECT PLOT/PRINT OUTPUT OPTIONS, AND
C               GENERATE COMPUTER FILE FOR INFORMATION TRANSFER FROM THE
C               PATTERN RECOGNITION SUB-COMPONENT TO THE OPTIMIZATION
C               SUB-COMPONENT
C
C           FUNCTIONS OR SUBROUTINES CALLED-SMOOTH, PLEDGE, SIM, PLOT
C
      DIMENSION NAME(NVAR),XIN(NVAR),XOUT(NTW  ),ONAMES(MOUTS),
     1PNAMES(NTW),NTSMO(MOUTS),TS(1,NTS,NDP),STS(NDP),ETS(NDP),
     2INDVAR(NIND),IS(20),TIP(NTS),NDPTS(NTS),TNAMES(20)
      DIMENSION TTS(NTS,NDP),TSAVE(NTS,NDP)
      DATA TNAMES/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
     1 1HN,1HO,1HP,1HQ,1HR,1HS,1HT/
C
C           INITIALIZE KEY, IPLOT, AND ISWITCH TO INDICATE NO SMOOTHING
C               OF REAL WORLD TIME SERIES, WRITING MODEL (SIM) OUTPUT
C                   ON A LOCAL COMPUTER FILE FOR PLOTTING, AND
C               SIMULATION MODEL EXECUTION IS TO BE USED FOR EITHER
C                   OBJECTIVE FUNCTION EVALUATION OR PLOTTING
C
      KEY=0
      IPLOT=1
      ISWITCH=1
C
C           INPUT VARIABLE AND PROBLEM DESCRIPTION PARAMETERS IN AN
C               INTERACTIVE
C           MODE
C           -INPUT VARIABLE NAMES-
      PRINT 998
      READ 898,(NAME(J),J=1,NVAR)
C           -INPUT VARIABLE DEFAULT VALUES-
      PRINT 997
      READ 897,(XIN(J),J=1,NVAR)
C           -INSTRUCTION ELEMENT IS(1) = 0 FOR A PARAMETER ESTIMATION
C               PROBLEM AND IS(1) ≠ 0 FOR AN OPTIMAL CONTROL PROBLEM-
      PRINT 996
      READ 896,IS(1)
C           -OUTPUT VARIABLE NAMES-
      PRINT 994
      READ 894,(ONAMES(J),J=1,MOUTS)
C           -SIMULATION AND PLOTTING PARAMETERS
      PRINT 991
      READ 891,INT,ITT,DT,DUR
C
C           TEST FOR PARAMETER ESTIMATION OR OPTIMAL CONTROL PROBLEM
      IF(IS(1).NE.0) GO TO 16
C
C           INPUT REAL WORLD TIME SERIES FROM TAPE97
      PRINT 1007
      READ 2007,ANS
      IF(ANS.EQ.2HNO) GO TO 100
C           READ TIME SERIES FROM TAPE97
      DO 101 J=1,NTS
      NDPTS(J)=NDP
101   NTSMO(J)=J
      DO 102 JJ=1,NDP
102   READ(97) (TTS(J,JJ),J=1,NTS)
      GO TO 103
100   CONTINUE
C           INPUT REAL WORLD TIME SERIES IN AN INTERACTIVE MODE
      DO 1 J=1,NTS
      PRINT 993,J
      READ 893,NDPTS(J)
      PRINT 9931
      READ 8931,NTSMO(J)
      JJ=NDPTS(J)
      PRINT 992
1     READ 892,(TTS(J,JJJ),JJJ=1,JJ)
103   CONTINUE
```

```
C
C           SMOOTH REAL WORLD TIME SERIES
      PRINT 990
      READ 890,ANS
      IF(ANS.EQ.2HNO) GO TO 2
      KEY=1
C           SMOOTH WITH INITIAL VALUES OF TIME SERIES
      PRINT 989
      READ 889,ANS
      IF(ANS.EQ.2HNO) GO TO 3
      ITIP=1
C           INPUT INITIAL VALUES OF TIME SERIES
      PRINT 988
      READ 888,(TIP(J),J=1,NTS)
      GO TO 4
3     CONTINUE
      ITIP=0
4     CONTINUE
C           SELECT NO SMOOTHING, LINEAR SMOOTHING, OR QUADRATIC SMOOTHING
C                FOR EACH INDIVIDUAL TIME SERIES
      PRINT 1010
      DO 8 I=1,NTS
      IDIM=NDPTS(I)
      PRINT 1008,ONAMES(I)
      READ 1009,M
      DO 9 J=1,IDIM
      IF(M.EQ.0) ETS(J)=TS(I,J)
9     STS(J)=TTS(I,J)
      IF(ITIP.EQ.1) TI=TIP(I)
      IF(M.EQ.0) GO TO 60
C           CALL SUBROUTINE SMOOTH TO PERFORM THE SMOOTHING OF THE TIME
C                SERIES
C                      STS IS THE TIME SERIES TO BE SMOOTHED
C                      ETS IS THE SMOOTHED TIME SERIES ON RETURN
C                      IDIM IS THE LENGTH OF STS (OR ETS)
C                      ITIP = 0 IF NO INITIAL VALUE IS USED FOR SMOOTHING
C                           ≠ 0 OTHERWISE
C                      TI IS THE INITIAL VALUE OF STS (IF KNOWN)
C                      M IS THE DEGREE OF THE POLYNOMIAL FILTER M=1 OR 2
      CALL SMOOTH(STS,ETS,IDIM,ITIP,TI ,M)
60    DO 10 J=1,IDIM
10    TS(1,I,J)=ETS(J)
8     CONTINUE
11    CONTINUE
C
C           PRINT/PLOT OF TIME SERIES
      PRINT 986
      READ 886,ANS
      IF(ANS.EQ.2HNO) GO TO 12
C           CONSTRUCT ARRAYS FOR PLOTTING
      DO 15 J=1,MOUTS
15    PNAMES(J)=ONAMES(J)
      DO 25 J=1,NTS
      NTSJ=MOUTS+J
25    PNAMES(NTSJ)=TNAMES(J)
      NTW=NTS+MOUTS
C           CALL SUBROUTINE PLEDGE TO PRINT HEADER INFORMATION ON TAPE99
C                FOR PLOT/PRINT.
C                      PNAMES CONTAINS THE NAMES OF ALL POSSIBLE PLOT
C                      VARIABLES
C                      XOUT IS A DUMMY ARGUMENT
      CALL PLEDGE(99,NTW,PNAMES,XOUT)
C
C           CALL SIM TO GENERATE MODEL OUTPUTS CORRESPONDING TO XIN
      I=INT*ITT
      IPLOT=2
      CALL SIM(XIN,XOUT,T,DT,PP,I,IPLOT,TTS,NTS,NSET,ISWITCH,NVAR,MOUTS,
     1  1,P2,NTIMES,IS)
C
C           PRINT HEADER INFORMATION ON PLOT OUTPUT FILE-TAPE6
      WRITE(6,1000) (J,NAME(J),XIN(J),J=1,NVAR)
      WRITE(6,1004)
      WRITE(6,1006) (J,TNAMES(J),ONAMES(NTSMO(J)),J=1,NTS)
C
C           CONSTRUCT INDVAR ARRAY TO SELECT OUTPUT VARIABLES TO BE
C                PRINT/PLOT
      II=1
```

```
      DO 35 J=1,NTS
      INDVAR(II)=NTSMO(J)
      II=II+1
      INDVAR(II)=-NTSMO(J)
      II=II+1
35    CONTINUE
      DO 45 J=1,NTS
      INDVAR(II)=MOUTS+J
      II=II+1
      INDVAR(II)=-MOUTS-J
      II=II+1
45    CONTINUE
      INDVAR(II)=0
      PRINT 984
C           SELECT PRINT AND/OR PLOT OPTIONS
      READ 884,NTYPE
C           CALL SUBROUTINE PLOT TO PLOT AND/OR
C           PRINT ACTUAL REAL WORLD TIME SERIES
      CALL PLOT(99,XOUT,INDVAR,NTIMES,NTYPE,INT)
      REWIND 99
      IF(KEY.EQ.0) GO TO 16
C           IF SMOOTHING OF TIME SERIES WAS DONE, PRINT/PLOT SMOOTHED
C                 TIME SERIES
      DO 56 M=1,1
      DO 55 J=1,NTS
      JJ=NDPTS(J)
      DO 55 JJJ=1,JJ
55    TSAVE(J,JJJ)=TS(M,J,JJJ)
      CALL PLEDGE(99,NTW,PNAMES,XOUT)
      WRITE(6,100)M
C
C           CALL SIM TO GENERATE MODEL OUTPUTS FOR COMPARISON WITH
C                 SMOOTHED TIME SERIES
      CALL SIM(XIN,XOUT,T,DT,PP,I,IPLOT,TSAVE,NTS,NSET,ISWITCH,NVAR,
     1  MOUTS,1,P2,NTIMES,IS)
      CALL PLOT(99,XOUT,INDVAR,NTIMES,NTYPE,INT)
      REWIND 99
56    CONTINUE
2     CONTINUE
16    CONTINUE
12    CONTINUE
C           PLOT/PRINT OF MODEL OUTPUTS FOR VALUES OF XIN AS INPUT BY
C                 USER
      INT=1
      PRINT 800
      READ 700,ANS
      IF(ANS.EQ.2HNO) GO TO 90
C           PRINT INPUT VARIABLE NAMES AND PRESENT DEFAULT VALUES-
C                 ALLOWING USER TO MAKE CHANGES IF DESIRED
      PRINT 799,(J,NAME(J),XIN(J),J=1,NVAR)
      PRINT 798
      READ 698,NCHG
      IF(NCHG.EQ.0) GO TO 23
      DO 21 J=1,NCHG
      PRINT 797
      READ 697,JJ,XIN(JJ)
21    CONTINUE
23    CONTINUE
C           SELECT OUTPUT VARIABLES DESIRED FOR PRINT/PLOT AND CONSTRUCT
C                 APPROPRIATE INDVAR ARRAY
      PRINT 796,(J,ONAMES(J),J=1,MOUTS)
      PRINT 795
      READ 695,NOS
      JJJ=1
      DO 22 J=1,NOS
      PRINT 794
      READ 694,JJ
      INDVAR(JJJ)=JJ
      JJJ=JJJ+1
      INDVAR(JJJ)=-JJ
      JJJ=JJJ+1
22    CONTINUE
      INDVAR(JJJ)=0
      PRINT 984
      READ 693,NTYPE
      PRINT 792
      READ 692,ITT
```

```
      NTIMES=DUR/(DT*ITT)+.0001
C
C           PRINT HEADER INFORMATION FOR INPUT TO PLOT SUBROUTINE
      CALL PLEDGE(99,MOUTS,ONAMES,XOUT)
C
C           CALL SIM TO GENERATE MODEL OUTPUTS CORRESPONDING TO XIN
      IPLOT=1
      ISWITCH=1
      CALL SIM(XIN,XOUT,T,DT,PP,ITT,IPLOT,TSAVE,NTS,NSET,ISWITCH,NVAR,
     1  MOUTS,1,P2,NTIMES,IS)
C           PRINT OUTPUT INFORMATION ON TAPE6-PLOT OUTPUT FILE
      WRITE( 6,1000) (J,NAME(J),XIN(J),J=1,NVAR)
C
C           CALL PLOT SUBROUTINE TO OUTPUT PLOT/PRINT ON TAPE6
      CALL PLOT(99,XOUT,INOVAR,NTIMES,NTYPE,INT)
      REWIND 99
90    CONTINUE
C
C           NORMAL EXECUTION OF SIM SUBROUTINE (SIMEX STUDY)
      PRINT 791
      READ 691,ANS
      IF(ANS.EQ.2HNO) GO TO 98
C           NORMAL SIM EXECUTION
91    ISWITCH=0
      CALL SIM(XIN,XOUT,T,DT,PP,ITT,IPLOT,TSAVE,NTS,NSET,ISWITCH,NVAR,
     1  MOUTS,1,P2,NTIMES,IS)
      ISWITCH=1
C
C           CONTINUE NORMAL SIM EXECUTION STUDY
      PRINT 790
      READ 690,ANS
      IF(ANS.EQ.3HYES) GO TO 91
98    CONTINUE
C
C           CONTINUE PATTERN RECOGNITION STUDY
      PRINT 789
      READ 689,ANS
      IF(ANS.EQ.3HYES) GO TO 12
C
C           WRITE TAPE98 FOR INFORMATION TRANSFER TO THE OPTIMIZATION
C                SUB-COMPONENT
      PRINT 1002
      READ 1003,ANS
      IF(ANS.EQ.2HNO) GO TO 99
19    CONTINUE
      WRITE(98)IS(1)
      WRITE(98)NVAR
      WRITE(98)(NAME(J),J=1,NVAR)
      WRITE(98)MOUTS
      WRITE(98)(ONAMES(J),J=1,MOUTS)
      IF(IS(1).NE.0) GO TO 18
      WRITE(98)(NDPTS(J),J=1,NTS)
      WRITE(98)KEY
      DO 17 J=1,NTS
      JJ=NDPTS(J)
      WRITE(98)(TTS(J,JJJ),JJJ=1,JJ)
17    CONTINUE
      IF(KEY.EQ.0) GO TO 18
      DO 171 M=1,1
      DO 172 J=1,NTS
      JJ=NDPTS(J)
      WRITE(98)(TS(M,J,JJJ),JJJ=1,JJ)
172   CONTINUE
171   CONTINUE
18    CONTINUE
99    CONTINUE
      RETURN
1000  FORMAT(*1THE NUMBER, NAME, AND DEFAULT VALUES OF THE INPUT VARIABL
     1ES FOR THIS PRINT/PLOT ARE...*/(* *,I3,3X,A6,3X,F8.3))
1001  FORMAT(*1THIS PLOT/PRINT CONTAINS THE REAL WORLD TIME SERIES (X),
     1THE SMOOTHED TIME SERIES M=1 (XX), AND M=2 (XXX)*)
1002  FORMAT(* TYPE YES TO WRITE TAPE98 (A6)*)
1003  FORMAT(A6)
1004  FORMAT(1H0, * THIS PLOT/PRINT CONTAINS THE REAL WORLD DATA AND THE
     1 CORRESPONDING MODEL OUTPUTS*/* THE VARIABLES CORRESPOND AS FOLLOW
     2S...*/)
1005  FORMAT(1H1,* THIS PLOT CONTAINS THE MODEL OUTPUTS AND THE SMOOTHED
```

```
     1 (M=*,I2,*)TIME SERIES WITH VARIABLES CORRESPONDING AS IN THE LAST
     1 PRINT/PLOT*/)
1006  FORMAT(1H ,I3,3X,A6,3X,A6)
1007  FORMAT(* TYPE YES TO READ TIME SERIES FROM TAPE97 (A6)*)
2307  FORMAT(A6)
1008  FORMAT(* TYPE IN M FOR THE TIME SERIES *, A6,*  (I3)*)
1009  FORMAT(I3)
1010  FORMAT(* FOR THE SMOOTHING M=0=REAL WORLD T. S., M=1=LINEAR FILT.,
     1 AND M=2=QUADRATIC FILT. *)
9931  FORMAT(* TYPE THE NUMBER OF THE MODEL OUTPUT CORRESPONDING TO THE
     1TIME SERIES (I3)*)
8931  FORMAT(I3)
999   FORMAT(* TYPE IN NUMBER OF INPUT VARIABLES (I3)*)
899   FORMAT(I3)
998   FORMAT(* TYPE IN THE INPUT VARIABLE NAMES (10A6)*)
898   FORMAT(10A6)
997   FORMAT(* TYPE IN THE DEFAULT VALUES FOR THE VARIABLES (7F10.3)*)
897   FORMAT(7F10.3)
996   FORMAT(* TYPE 0 IF P E PROBLEM AND 1 IF NOT (I1)*)
896   FORMAT(I1)
995   FORMAT(* TYPE NUMBER OF MODEL OUTPUTS (I3)*)
895   FORMAT(I3)
994   FORMAT(* TYPE OUTPUT VARIABLE NAMES (10A6)*)
894   FORMAT(10A6)
993   FORMAT(* TYPE THE NUMBER OF DATA POINTS IN THE TIME SERIES*,I3,* (
     1I3)*)
893   FORMAT(I3)
992   FORMAT(* TYPE IN THE TIME SERIES POINT BY POINT (7F10.3)*)
892   FORMAT(7F10.3)
991   FORMAT(* TYPE IN INT,ITT,DT,DUR FOR SIMULATION(2I4,2F8.3)*)
891   FORMAT(2I4,2F8.3)
990   FORMAT(* TYPE YES TO OBTAIN SMOOTHING OF TIME SERIES (A6)*)
890   FORMAT(A6)
989   FORMAT(* TYPE YES FOR SMOOTHING WITH INITIAL VALUES (A6)*)
889   FORMAT(A6)
988   FORMAT(* TYPE IN INITIAL VALUES FOR SMOOTHING (7F10.3)*)
888   FORMAT(7F10.3)
986   FORMAT(* TYPE YES FOR PRINT/PLOT OF TIME SERIES (A6)*)
886   FORMAT(A6)
985   FORMAT(* TYPE NO FOR SEPARATE OUTPUT OF TIME SERIES (A6)*)
885   FORMAT(A6)
984   FORMAT(* TYPE IN NTYPE FOR PLOT OR PRINT OR BOTH*/*    NTYPE=0 FOR
     1USER SPECIFIED SCALING OF PLOT*/*          =1  FOR RANGE OF SCALES S
     2ET BY ACTUAL MAX, MIN*/*          =2  FOR TIME SERIES AND USER SPECI
     3FIED SCALES*/*          =3  FOR TIME SERIES AND CALCULATED SCALES*/*
     4          =4  FOR TIME SERIES ONLY*/ * (I1)*)
884   FORMAT(I1)
800   FORMAT(* TYPE YES FOR PLOT/PRINT OF MODEL OUTPUTS (A6)*)
700   FORMAT(A6)
799   FORMAT(* THE NUMBER, NAME, AND DEFAULT VALUE FOR THE INPUT VARIABL
     1ES ARE...*/(* *,I3,3X,A6,3X,F8.3))
798   FORMAT(* TYPE IN THE NUMBER OF VALUES TO BE CHANGED (I3)*)
698   FORMAT(I3)
797   FORMAT(* TYPE IN THE NUMBER AND VALUE FOR A VARIABLE (I3,F8.3)*)
697   FORMAT(I3,F8.3)
796   FORMAT(* THE NUMBER AND NAME OF THE OUTPUT VARIABLES ARE...*/(* *,
     1I3,3X,A6))
795   FORMAT(* TYPE IN THE NUMBER OF OUTPUTS TO BE PLOT/PRINT (I3)*)
695   FORMAT(I3)
794   FORMAT(* TYPE IN A NUMBER OF AN OUTPUT VARIABLE FOR P/P (I3)*)
694   FORMAT(I3)
693   FORMAT(I1)
792   FORMAT(* TYPE IN THE INTERVAL OF DT UNITS FOR OUTPUT (I4)*)
692   FORMAT(I4)
791   FORMAT(* TYPE YES FOR SIMEX STUDY OF MODEL (A6)*)
691   FORMAT(A6)
790   FORMAT(* TYPE YES FOR MORE SIMEX STUDY (A6)*)
690   FORMAT(A6)
789   FORMAT(* TYPE YES TO CONTINUE PATTERN RECOGNITION STUDY (A6)*)
689   FORMAT(A6)
      END
```

Subroutine SMOOTH

Smoothing of the real world time series is accomplished in the pattern recognition sub-component within the subroutine SMOOTH. By interaction with the SUBPR subroutine, a user may select either a linear or quadratic moving polynomial filter. In order to reduce the amount of computation necessary to perform the smoothing the subroutine uses Legendre orthogonal polynomials. The smoothing capability of SYSOPT allows the user to filter an extraneous zero-mean noise component out of the real world time series. Thus, (1) more accurate comparisons between real world data and model outputs can be made, and (2) the smoothed time series can provide a better basis for solving parameter estimation problems with the optimization sub-component.

To explain how the subroutine performs smoothing of time series, it is necessary to develop the mathematical techniques used in moving polynomial filters [5]. First, defining

$$C_j = \frac{(L+j+1)^{(j+1)}}{(2j+1)L^{(j)}}$$

with L+1 equal to the number of data points used for smoothing,

$$I^{(j)} = I(I-1)(I-2)\ldots(I-j+1)$$

and

$$W_j(x) = \frac{1}{c_j} \sum_{k=0}^{j} (-1)^k \binom{j}{k} \binom{j+k}{k} \frac{x^{(k)}}{L^{(k)}}$$

then it can be shown that

$$\sum_{i=0}^{n} \sum_{j=0}^{n} W_j(x)W_i(x) = \begin{cases} 1 & i=j \\ 0 & i \neq j \end{cases}$$

If the L+1 data points to be smoothed by the filter are:

$Y_{n-L}, Y_{n-L+1}, \ldots, Y_n$

and m is the order of the filter, then the smoothing polynomial is defined to be $(p^*(x))_n$-the polynomial of degree m that minimizes a sum of squared errors criterion ℓ_n, where

$$\ell_n = \sum_{x=0}^{L} (Y_{n-L+x} - (p(x))_n)^2$$

and $(p(x))_n = \sum_{j=0}^{m} (b_j)_n W_j(x)$

In order to find the $(p^*(x))_n$ that minimizes ℓ_n, one sets

$$\frac{d\ell_n}{d(b_j)_n} = 0 \text{ for } j = 0,1,\ldots,m$$

obtaining

$$(p^*(x))_n = \sum_{j=0}^{m} (b_j^*)_n W_j(x)$$

where $(b_j^*)_n = \sum_{x=0}^{L} Y_{n-L+x} W_j(x)$ for $j = 0,1,\ldots,m$

The subroutine SMOOTH has the capability of smoothing with m equal to one or two depending on whether the user wishes to have a linear or quadratic filter. The value of m that should be used in a given application

depends on the form of the real world time series, the amount of information it contains, and the type of noise corrupting it.

A value of L+1=4 or L=3 was chosen for the smoothing subroutine so that the least number of data points consistent with smoothing would be used. Thus, the smoothing process, performed with a minimum loss of information, does not degenerate into curve fitting through data points.

The subroutine has the additional capability of smoothing with or without initial values of time series. In other words if a time series is known to have started at a particular value, then this information is weighted heavily in the construction of the initial portion of the smoothed time series. The value of the integer switch ITIP determines if this capability is incorporated into the smoothing.

In the subroutine PONE(x) and PTWO(x) are the first and second degree Legendre orthogonal polynomials that are used in the smoothing process. Similarly C(I) corresponds to C_j defined earlier. There is additional programming logic in the subroutine to handle the case in which IDIM is odd and the case in which IDIM is even, where IDIM is the number of points in the time series. Finally the smoothed time series is begun and ended with points of the original time series as there are not sufficient data for smoothing.

Subroutine Argument List

TS(J)	contains the Jth element of the time series to be smoothed by a linear or quadratic polynomial filter. J = 1,IDIM.
ETS(J)	on return from SMOOTH, contains the Jth element of the smoothed time series. J = 1,IDIM.
IDIM	contains the length of the real world time series TS
ITIP	= 1 if an initial value of the time series TS is known with confidence
	≠ 1 if no initial value of the time series TS can be relied upon as accurate
TIP	if ITIP = 1, then TIP contains the initial value of the time series TS
	if ITIP ≠ 1, then TIP is not used
M	contains the desired degree of the polynomial filter, M = 1 or 2.

```
      SUBROUTINE SMOOTH(TS,ETS,IDIM,ITIP,TIP,M)
C
C           TITLE-SMOOTH IS A SUBROUTINE FOR SMOOTHING REAL WORLD TIME
C               SERIES
C
C           PURPOSE-FILTER ZERO-MEAN CORRUPTING NOISE FROM TIME SERIES
C               TO:  (1) ALLOW MORE ACCURATE COMPARISONS BETWEEN TIME
C               SERIES AND MODEL OUTPUTS, AND (2) PROVIDE A BETTER
C               BASIS FOR SOLVING PARAMETER ESTIMATION PROBLEMS WITH THE
C               OPTIMIZATION SUB-COMPONENT
C
C           FUNCTIONS OR SUBROUTINES CALLED-PO(DEFINED INTERNAL TO
C               SUBROUTINE), PONE(DEFINED INTERNAL TO SUBROUTINE),
C               AND PO(DEFINED INTERNAL TO SUBROUTINE)
C
      DIMENSION TS(1),ETS(1),Y(4),C(3)
      REAL L
C           CONSTRUCT ZERO, FIRST, AND SECOND DEGREE LEGENDRE ORTHOGONAL
C               POLYNOMIALS
      PO(X)=1.0
      PONE(X)=1.0-2.*X/3.
      PTWO(X)=1.-2.*X+X*(X-1)
      L=3.
C           CALCULATE C(J) - LEGENDRE POLYNOMIAL WEIGHTING COEFFICIENTS
      C(1)=L+1
      C(2)=(L+2)*(L+1)/(3.*L)
      C(3)=(L+3)*(L+2)*(L+1)/(5.*L*(L-1))
      IDIV=(IDIM-1)/2
C           CALCULATE ITEST-SWITCH THAT DETERMINES WHETHER IDIM IS ODD OR
C               EVEN
C               ITEST = 0 IF IDIM IS EVEN
C               ITEST ≠ 0 IF IDIM IS ODD
      ITEST=IDIM-2*(IDIM/2)
C
C           GO THROUGH ACTUAL SMOOTHING LOOP IDIV (GREATEST INTEGER ≤
C               IDIM/2) TIMES EACH PASS THROUGH LOOP, COMPUTE TWO
C               SMOOTHED ELEMENTS OF ETS FROM FOUR DATA POINTS OF TS.
C               ETS(J) AND ETS(J+1) ARE COMPUTED FROM TS(J-1),TS(J),
C               TS(J+1),TS(J+2), EXCEPT FOR THE END POINTS ETS(1),
C               ETS(2),ETS(IDIM-1), AND ETS(IDIM) WHICH MAY BE SET
C               EQUAL TO THE END POINTS OF TS.  (DIFFERENT PROCEDURES
C               FOR IDIM ODD AND EVEN.)  NOTE ALSO THAT IF AN INITIAL
C               VALUE IS KNOWN (ITIP = 1), THEN TIP IS USED IN THE
C               SMOOTHING PROCESS.
      DO 1 I=1,IDIV
      J=(I-1)*2
      IF(ITIP.EQ.1) Y(1)=TIP
      IF(ITIP.EQ.1) GO TO 3
      J=(I-1)*2+1
3     CONTINUE
      JJ=J+1
      ETS(JJ)=0.0
      JJJ=JJ+1
      ETS(JJJ)=0.0
      JONE=1
      JTWO=2
      IF(I.NE.1.OR.ITIP.EQ.0)  Y(1)=TS(J)
      Y(2)=TS(J+1)
      Y(3)=TS(J+2)
      IF(I.EQ.IDIV.AND.ITIP.EQ.0.AND.ITEST.NE.0) Y(4)=Y(3)
      IF(I.EQ.IDIV.AND.ITIP.EQ.0.AND.ITEST.NE.0)GO TO 4
      Y(4)=TS(J+3)
4     CONTINUE
      DO 2 K=1,4
      A=K
C           COMPUTE OPTIMAL WEIGHTING COEFFICIENTS FOR DATA POINTS
C               Y(1), Y(2), Y(3), Y(4)
      SUMO=PO(A-1)*PO(JONE)/C(1)+PONE(A-1)*PONE(JONE)/C(2)
      IF(M.EQ.2)SUMO=SUMO+PTWO(A-1)*PTWO(JONE)/C(3)
      SUMT=PO(A-1)*PO(JTWO)/C(1)+PONE(A-1)*PONE(JTWO)/C(2)
      IF(M.EQ.2)  SUMT=SUMT+PTWO(A-1)*PTWO(JTWO)/C(3)
      ETS(JJ)=ETS(JJ)+SUMO*Y(K)
      ETS(JJJ)=ETS(JJJ)+SUMT*Y(K)
2     CONTINUE
1     CONTINUE
C           ASSIGN END POINTS OF SMOOTHED TIME SERIES
      IMONE=IDIM-1
```

```
IF(ITIP.EQ.J) ETS(1)=TS(1)
IF(ITEST.EQ.J) ETS(IMONE)=TS(IMONE)
ETS(IDIM)=TS(IDIM)
RETURN
END
```

Subroutine PLOT

Subroutine PLOT, a modified version of the plotting subroutine in FORDYN [4], has the capability of printing in tabular form or plotting in graphical form up to twenty variables as functions of time. As the PLOT subroutine writes all of its output on a local file, TAPE6, in the computer, any output can be disposed to a convenient printer on request from the user. This subroutine has been incorporated into the pattern recognition sub-component to plot and/or print model, real world, or smoothed time series.

At present, in SYSOPT the user may choose one of three output options through selection of the value of the variable NTYPE. If NTYPE is set equal to 1, a plot, with scaling automatically set by reference to the maximum and minimum values of the data, of the desired output variables will be made on TAPE6. On the other hand if the user selects NTYPE = 3, then both a plot with automatic scaling and a tabular print of the output will be made. Finally, if the user wishes only to have the tabular output data, then NTYPE = 4 should be selected.

In the subroutine, LUNIT represents the local file containing the maximum number of data values, a list of the output variable names, and the output data records.

Subroutine Argument List*

LUNIT	contains the unit input/output number on which the output names, and output data values were printed. (In SYSOPT LUNIT = 99).
VECTOR	contains temporary storage for values of the output variables
INDVAR	defined in SUBPR variable lists
NTIME	contains the number of data points for printing/ plotting from PLOT
NTYPE	= 1 for range of scales of plots set by actual MAXIMUM and MINIMUM values of the output variables
	= 3 for printing of time series and for calculated scales of the plots from maximum and minimum values of the output variables
	= 4 for printing only of the output variables
INT	defined as ITT in SUBPR variable lists

*See Reference [4] for further explanations and definitions of variables in PLOT.

```
      SUBROUTINE PLOT(LUNIT,VECTOR,INDVAR,NTIME,NTYPE,INT)
C
C           TITLE-PLOT IS A SUBROUTINE THAT PRINTS AND/OR PLOTS
C               VARIABLES IN SYSOPT
C
C           PURPOSE-PRINTING AND/OR PLOTTING OF ACTUAL REAL WORLD TIME
C               SERIES, SMOOTHED REAL WORLD TIME SERIES, AND MODEL
C               OUTPUTS AS FUNCTIONS OF TIME.  UP TO TWENTY VARIABLE
C               PRINT/PLOTS CAN BE OUTPUT FROM ONE CALL TO THE SUBROUTINE
C
C           FUNCTIONS OR SUBROUTINES CALLED-NONE
C
      DIMENSION  ALPHA(26)
      INTEGER OUT,PLUS,BLANK,DUP,ALPHA
      DIMENSION  OUT(101),DUP(20)
      DIMENSION VECTOR(1)
      DIMENSION  INDVAR(1)
      DIMENSION VALUES(26)
      DIMENSION  GREAT(26)
      DIMENSION  SMALL(26)
      DIMENSION  NAMES(26)
C
      DATA  ALPHA /1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
     +          1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
      DATA  PLUS /1H+/
      DATA  BLANK/1H /
      DATA  LUOUT /6/
      REWIND LUNIT
C         RECORD NUMBER 1     M --  MAXIMUM NUMBER OF DATA VALUES - 1 REC
C         RECORD NUMBER 2     NAMES -- VARIABLE NAMES
C         RECORD NUMBER 3     GREAT -- VECTOR OF MAXIMUM VALUES
C         RECORD NUMBER 4     SMALL -- VECTOR OF MINIMUM VALUES
C         RECORD NUMBER 5+NTIME     TIME,VECTOR -- PLOT DATA
C                   INPUT THE INITIAL RECORD OF CONTROL INFORMATION
      READ(LUNIT) M
C                   INPUT THE VECTOR OF VARIABLE NAMES
      READ (LUNIT) (VECTOR(K),K=1,M)
      CALL  PLY (VECTOR,INDVAR,NAMES,M,N)
C
      IF (NTYPE) 2000,3000,2000
C                 A PLOT WITH THE SCALES SET BY THE RANGE
C                 OF NUMBERS IS DESIRED.
C                 THIS SHOULD NOT BE DONE OFTEN, AS IT
C                 INVOLVES A DOUBLE SCAN OF THE FILE.
 2000 CONTINUE
C                       THE  FOLLOWING BEGINS A INVOLVED SET
C                       OF IF CHECKING TO GIVE THE VERSATILITY
C                       OF SEVERAL INPUT TYPES.
C
C                       0     -     USER SPECIFIED SCALING OF PLOT
C
C                       1     -     RANGE OF SCALES SET BY ACTUAL MAX,MIN
C
C                       2     -     TIME SERIES AND USER SPECIFIED SCALES
C
C                       3     -     TIME SERIES AND CALCULATED SCALES
C
C                       4     -     TIME SERIES, BUT NO PLOT
C
      READ (LUNIT)
      READ (LUNIT)
      IF (NTYPE-1) 2010,2030,2010
 2010 IF (NTYPE-2) 2050,2020,2020
 2020 WRITE (LUOUT,1005)
      WRITE(LUOUT,1006)(NAMES(I),I=1,N)
      IF(NTYPE-3)2050,2030,2050
 2030 DO 2040 I=1,N
      GREAT(I) = -9.E20
      SMALL(I) =  9.E20
 2040 CONTINUE
 2050 DO 2200 I=1,NTIME
      READ (LUNIT) TIME,(VECTOR(K),K=1,M)
      CALL PLY (VECTOR,INDVAR,VALUES,M,N)
      IF (NTYPE-2)2070,2060,2060
 2060 WRITE (LUOUT,1007) TIME,(VALUES(K),K=1,N)
 2070 IF(NTYPE-3) 2080,2090,2080
 2080 IF(NTYPE-1) 2200,2090,2200
```

```
 2090 DO 2100 J=1,N
      GREAT(J)= AMAX1(GREAT(J),VALUES(J))
      SMALL(J)= AMIN1(SMALL(J),VALUES(J))
 2100 CONTINUE
 2200 CONTINUE
      IF(NTYPE-4)2210,9999,2210
 2210 CONTINUE
      REWIND LUNIT
 2220 CONTINUE
      IF (NTYPE-1) 2225,2240,2225
 2225 CONTINUE
      IF(NTYPE-3)2230,2240,2230
 2230 CONTINUE
      DO 2235 J=1,3
      READ(LUNIT)
 2235 CONTINUE
      GO TO 3000
 2240 DO 2250 J=1,4
 2250 READ (LUNIT)
      GO TO 4000
 3000 CONTINUE
C                          INPUT THE VECTOR OF MAXIMUM SCALINGS
      READ (LUNIT) (VECTOR(K),K=1,M)
      CALL  PLY (VECTOR,INDVAR,GREAT,M,N)
C                          INPUT THE VECTOR OF MINIMUM SCALINGS
      READ (LUNIT) (VECTOR(K),K=1,M)
      CALL PLY (VECTOR,INDVAR,SMALL,M,N)
 4000 CONTINUE
C                          NOW BEGIN BY PRINTING THE SCALE VALUES OF
C                          THE PLOT FOR EACH DATA GROUP.
      IF(NTYPE.GE.2) WRITE(LUOUT,1001)
      IF(NTYPE.LT.2) WRITE(LUOUT,1009)
      DO  10 I = 1,N
      A=(GREAT(I)-SMALL(I))*.25+SMALL(I)
      B=(GREAT(I)-SMALL(I))*.5+SMALL(I)
      C=(GREAT(I)-SMALL(I))*.75+SMALL(I)
      WRITE (LUOUT,1002) SMALL(I),A,B,C,GREAT(I),NAMES(I),ALPHA(I)
 10   CONTINUE
C                          PRINT THE  PERCENT POINTS ON THE BASELINE
C                          BEFORE BEGINNING THE PLOT
      WRITE(LUOUT,1003) (I,I=1,10)
C                          BEGIN MAJOR PLOT LOOP TO PUT POINTS ONTO THE PRINTER
      DO 1000  ITIME = 1,NTIME
      READ (LUNIT) TIME,(VECTOR(K),K=1,M)
      CALL PLY (VECTOR,INDVAR,VALUES,M,N)
      DO  20   I=1,101
 20   OUT(I) = 0
      DO  30   I=1,20
 30   DUP(I) = 0
      N = MIN0 (N,26)
      NDUP = 0
      DO 200   I = 1,N
C                          CALCULATE PERCENT (ROUNDED UP).
      IF((GREAT(I)-SMALL(I)) .LT. .001) GO TO 200
      IPT=(VALUES(I)-SMALL(I))/(GREAT(I)-SMALL(I))*100.+.5
      IF (IPT)   40,70,50
 40   IPT = 0
      GO TO 70
 50   IF (IPT-100) 70,70,60
 60   IPT = 100
 70   IPT = IPT +1
C                          CHECK TO SEE IF DUPLICATION OCCURS
      IF (OUT(IPT)) 80,150,80
C                          BEGIN A FLAGGING PROCESS FOR DUPLICATE
 80   NDUP = NDUP + 1
      DUP(NDUP) = BLANK
      DO 90   J= 1,NDUP
      IF(OUT(IPT)-DUP(J)) 90,100,90
 90   CONTINUE
      GO TO 140
C                          DUPLICATE WAS FOUND IN STRING
C                          FIND THE END OF THAT GROUP AND
C                          ADD THE NEW VALUE
 100  DO 110 K = J,NDUP
      IF (DUP(K)-BLANK) 110,120,110
 110  CONTINUE
      GO TO 140
```

```
 120   IDUP=NDUP-K+1
       DO 130 L=1,IDUP
       JDUP = NDUP - L + 1
       DUP(JDUP+1) = DUP(JDUP)
 130   CONTINUE
C                          VALUE ADDED TO DUPLICATE STRING
       DUP(K) = ALPHA(I)
       GO TO 160
C                          NEW DUPLICATE, ADD TO END OF ARRAY
 140   NDUP = NDUP + 1
       DUP(NDUP) = OUT(IPT)
       NDUP = NDUP + 1
       DUP(NDUP) = ALPHA(I)
       GO TO 160
C                          NO DUPLICATE, SET OUTPUT ARRAY POSITION
 150   OUT(IPT) = ALPHA(I)
 160   CONTINUE
 200   CONTINUE
C
C                          ALL  THE NON-BLANK PTS ARE SET
C                          SO   PROCEED TO FILL IN SPECIAL POINTS
       DO  220 I = 1,101,10
       IF (OUT(I)) 220,210,220
 210   OUT(I) = PLUS
 220   CONTINUE
       DO 240 I = 1,101
       IF (OUT(I)) 240,230,240
 230   OUT(I) = BLANK
 240   CONTINUE
       DO 260 I = 1,20
       IF(DUP(I))260,250,260
 250   DUP(I)=BLANK
 260   CONTINUE
C                          THE OUTPUT VECTOR IS NOW FINISHED
C                          AND READY TO BE PRINTED
       NINT=INT-1
       IF(NINT.EQ.0) GO TO 262
       DO 261  I=1,NINT
261    WRITE(LUOUT,1008)
262    CONTINUE
       WRITE (LUOUT,1004) TIME,OUT,DUP
 1000  CONTINUE
       WRITE (LUOUT,1003)(I,I=1,10)
 9999  RETURN
 1001  FORMAT (1H1,50X,14HSCALES OF PLOT)
 1002  FORMAT (5X ,4(G10.4,15X),G10.4, 2X,A10,2X,A1)
 1003  FORMAT (1H ,10X,10(1H0,7X,I2),1H0,5X,10HDUPLICATES)
 1004  FORMAT (1H ,F9.3,1X,101A1,20A1)
1005   FORMAT(1H0,50X,18HTIME SERIES OUTPUT)
1006   FORMAT(1H0,10X,10(A10,2X)/(11X,10(A10,2X)))
1007   FORMAT(1H ,F9.3,(T10,10G12.4))
1008   FORMAT(1H )
1009   FORMAT(1H0,50X,14HSCALES OF PLOT)
       END
```

Subroutine PLY

Subroutine PLY performs two functions necessary for the proper execution of PLOT. These are: (1) transferring data from the VECTOR array, which contains output variable information, to the OUT array, which is used for constructing the plots, and (2) verifying that the indices specified in the INDVAR array are within proper bounds. In short, PLY acts as an interface between simulation outputs, sub-component inputs, and the PLOT subroutine.

Subroutine Argument List

VECTOR	defined in PLOT variable lists
INDEX	defined as INDVAR in SUBPR variable lists
OUT(J)	contains the value of the m_Jth output variable = VECTOR(m_J) on return from PLY
M	contains the number of output variables
N	contains the number of output variables to be print/plot in PLOT

```
      SUBROUTINE PLY  (VECTOR,INDEX,OUT,M,N)
C
C           TITLE-PLY IS A SUBROUTINE THAT FORMS OUTPUT VECTORS FOR
C               USE IN PLOT
C
C           PURPOSE-CHECK THAT INDICES OF OUTPUT VARIABLES FOR PRINTING/
C               PLOTTING ARE WITHIN PROPER BOUNDS AND CONSTRUCT AN ARRAY,
C               OUT, THAT CONTAINS THE SPECIFIC OUTPUT VARIABLE VALUES
C               THAT ARE TO BE PLOTTED/PRINTED.
C
C           FUNCTIONS OR SUBROUTINES CALLED-NONE
C
      DIMENSION  VECTOR(1),INDEX(1),OUT(1)
      DATA  LUOUT /6/
      IP = 0
      N  = 0
      IF(INDEX(1)) 5,5,10
 5    ILO = 1
      IHI=M
      GO TO 40
 10   IP = IP+1
      ILO= INDEX(IP)
      IF (ILO) 999,60,20
 20   IP = IP+1
      IHI= -INDEX(IP)
      IF (IHI) 30,30,40
 30   IP = IP-1
      IHI= ILO
 40   DO 50 J = ILO,IHI
      N  = N+1
      OUT(N) = VECTOR(J)
 50   CONTINUE
      GO TO 10
 60   RETURN
 999  WRITE (LUOUT,1001) IP ,INDEX(IP)
      GO TO 10
 1001 FORMAT  (6H0*****,10X,15HWARNING: INDEX(,I2,23H) IS IN ERROR. VALU
     +E = ,I5,10X, 6H(PLY) )
      END
```

PLEDGE

Another simple subroutine that is needed to insure the proper execution of PLOT is PLEDGE. PLEDGE writes, on a local input file for PLOT, variable information consisting of the maximum number of data values, and a list of the output variable names.

Subroutine Argument List

LU	contains the input/output unit number on which the following information is to be written 1. Size of variable set to be plotted 2. Output variable names to be plotted
N	contains the size of the variable set to be plotted
NAMES(J)	contains the Jth name of the output variables J = 1,N
XOUT	defined in SUBPR variable lists (in PLEDGE, XOUT is used as a dummy argument

```
      SUBROUTINE PLEDGE(LU,N,NAMES,XOUT)
C
C           TITLE-PLEDGE IS A SUBROUTINE THAT PRINTS OUTPUT VARIABLE
C               INFORMATION ON THE BEGINNING OF A LOCAL FILE
C
C           PURPOSE-WRITE THE SIZE OF THE OUTPUT VARIABLE SET AND THE
C               CORRESPONDING OUTPUT VARIABLE NAMES ON A LOCAL FILE
C               DEFINED BY UNIT NUMBER LU
C
C           FUNCTIONS OR SUBROUTINES CALLED-NONE
C
      DIMENSION XOUT(1),NAMES(1)
      WRITE(LU)N
      WRITE(LU)(NAMES(J),J=1,N)
      WRITE(LU)(XOUT(J),J=1,N)
      WRITE(LU)(XOUT(J),J=1,N)
      RETURN
      END
```

Optimization Sub-Component

Program MAINOPT

The main program in the optimization sub-component of SYSOPT has four functions. The first is the input of component variables that remain constant throughout the problem investigation. This includes the size of the input and output variable sets, the number of parameters in the objective function, and the size and number of the real world time series if a parameter estimation problem is being studied. Calculation of the amount of core necessary to store component arrays, whose sizes depend upon these constants, is another function performed in this main program. Third, the length of BLANK COMMON is extended (see section on Variable Dimensioning) to accommodate these size invariant arrays. Finally, to complete execution of the program, control is transferred to the subroutine SUBOPT.

MAINOPT executes in a manner similar to MAINPR of the pattern recognition sub-component. In both routines the main objective is to properly assign elements of the COM array with elements of particular arrays in the remainder of the program. In addition, as in MAINPR, a message:

. . . . END OF PROGRAM . . . . .

is printed at completion of execution of the optimization sub-component.

Significant Component Variables

NTS	contains the number of variables represented by the time series TTS (see DATA1 subroutine) in a parameter estimation problem. In an optimal control problem. NTS = 1.
NDP	contains the maximum number of data points in the NTS time series.
NVAR	contains the number of input parameters or control variables XIN(1), XIN(2), ..., XIN(NVAR) in the simulation.

NPOBJ — contains the number of parameters specifying the form of the objective function in an optimal control problem.

NTD — contains the maximum number of data points in all the NTS time series (primarily used for the variable dimensioning capability).

L(J) — contains the reference location in the COM array (see the section on the Variable Dimensioning Capability) for storing the elements of the Jth component array in the following list:

J	Component Array Name
1	NAME
2	XVAR
3	X
4	S
5	Y
6	NSUB
7	TTS
8	TSAVE
9	NDPTS
10	IS
11	ONAMES
12	XOUT
13	P
14	W
15	XIN

LAST — contains the current size of the COM array

COM — is an array in blank common containing all the program's arrays whose sizes depend on the values of the input variables.

```
      PROGRAM MAINOPT(TAPE98=65,OUTPUT=65,INPUT=65)
C
C           TITLE - MAIN PROGRAM FOR THE OPTIMIZATION SUB-COMPONENT OF
C               SYSOPT
C
C           PURPOSE - INPUT OF VARIABLES FIXING SIZE OF SUB-COMPONENT
C               ARRAYS, EXTENDING BLANK COMMON TO MAKE ROOM FOR THESE
C               ARRAYS, AND TRANSFER OF PROGRAM EXECUTION TO THE SUBOPT
C               SUBROUTINE
C
C           FUNCTIONS OR SUBROUTINES CALLED -NEXT,SETFL,SUBOPT
C
      DIMENSION L(15)
      COMMON COM(1)
C           INPUT AND COMPUTE SIZES OF PROBLEM DEPENDENT ARRAYS - DEFAULT
C               VALUES = 1
      PRINT 999
      READ 899,NTS
      IF(NTS.EQ.0) NTS=1
      PRINT 998
      READ 899,NDP
      IF(NDP.EQ.0)NDP=1
      PRINT 997
      READ 899,NVAR
      IF(NVAR.EQ.0) NVAR=1
      PRINT 996
      READ 899,MOUTS
      IF(MOUTS.EQ.0)MOUTS=1
      PRINT 995
      READ 899,NPOBJ
      IF(NPOBJ.EQ.0)NPOBJ=1
      NTD=NTS*NDP
C
C           COMPUTE SIZE OF EXTENDED COMMON BY UPDATING LAST WITH CALLS
C               TO NEXT
      LAST=1
      DO 1 J=1,6
1     L(J)=NEXT(LAST,NVAR)
      L(7)=NEXT(LAST,NTD)
      L(8)=NEXT(LAST,NTD)
      L(9)=NEXT(LAST,NTS)
      L(10)=NEXT(LAST,20)
      L(11)=NEXT(LAST,MOUTS)
      L(12)=NEXT(LAST,MOUTS)
      L(13)=NEXT(LAST,NPOBJ)
      L(14)=NEXT(LAST,NTD)
      L(15)=NEXT(LAST,NVAR)
C
C           SETFL(A CDC ROUTINE) EXTENDS BLANK COMMON TO *LAST* WORDS.
C               IN EFFECT CALL SETFL(COM(LAST)) HAS THE SAME RESULT AS
C               REPLACING
C                           COMMON COM(1)
C                                BY
C                           COMMON COM(LAST)
      CALL SETFL(COM(LAST))
C
C           TRANSFER CONTROL OF PROGRAM EXECUTION TO SUBROUTINE SUBOPT,
C               PASSING COM ELEMENTS TO ASSOCIATE WITH FIRST ELEMENTS
C               OF SUB-COMPONENT ARRAYS IN SUBOPT
      CALL SUBOPT(COM(L(1)),COM(L(2)),COM(L(3)),COM(L(4)),COM(L(5)),
     1 COM(L(6)),COM(L(7)),COM(L(8)),COM(L(9)),COM(L(10)),COM(L(11)),
     2 COM(L(12)),COM(L(13)),COM(L(14)),COM(L(15)),COM(LAST),NTS,NDP,
     3 NVAR,MOUTS,NTD,NPOBJ)
      PRINT 994
899   FORMAT(I3)
999   FORMAT(* TYPE IN THE NUMBER OF TIME SERIES-NTS (I3)*)
998   FORMAT(* TYPE IN THE MAXIMUM NUMBER OF DATA POINTS-NDP (I3)*)
997   FORMAT(* TYPE IN THE NUMBER OF INPUT VARIABLES-NVAR (I3)*)
996   FORMAT(* TYPE IN THE NUMBER OF MODEL OUTPUTS-MOUTS (I3)*)
995   FORMAT(* TYPE IN THE NUMBER OF PARAMETERS IN THE OBJECTIVE FUNCTIO
     1N-NPOBJ (I3)*)
994   FORMAT(* .....END OF PROGRAM.....*)
      END
```

Subroutine SUBOPT

Subroutine SUBOPT functions as an executive routine in the optimization sub-component. Virtually all the basic logic and interactive capability of the program occurs in this subroutine. In addition, arrays whose sizes depend upon the size of the search variable set, those that are problem dependent, are properly dimensioned by core storage allocations performed in SUBOPT.

The execution paths of this subroutine are principally determined from the instruction array IS which is input in the DATA subroutine hierarchy. This integer array contains switches that influence which algorithms or subroutines are called from SUBOPT, the extent of printing in the program, and also the extent of interaction with the sub-component. Output from this subroutine consists of local and global optimum information, sensitivity analysis results, interactive questions, and running totals of program execution costs. Much of the output can be used to determine how to most efficiently utilize the capabilities of the sub-component in solving parameter estimation and optimal control problems.

As can be seen from either the subroutine listing or optimization sub-component flow charts, SUBOPT has three main functions which are executed through calls to various subroutines in the component. The first of these, input of data, is controlled through calls to the DATA subroutine hierarchy: DATA1, DATA2, DATA3, DATA4, and VAR. On the first execution of SUBOPT all these subroutines are executed in order to properly load data into the program. However, on subsequent executions of SUBOPT and iterative loops within SUBOPT, these data input subroutines can be bypassed upon interactive command from the

user. Thus input of duplicate information to the component is minimized. Sensitivity study is the second primary function performed in SUBOPT. Through interaction, the user can choose which input variable sensitivities of the objective function should be calculated by the component. This sensitivity calculation may be performed at each global optimum and at an arbitrary number of points in the input variable space. This capability aids the user in selecting proper search variable sets, validating the simulation model, and, in parameter estimation problems, determining what data are critical. Finally, the last primary function of SUBOPT is the numerical optimization of arbitrary objective functions. The optimizations may be executed in either a global or local manner with local searches conducted using a two-level method or algorithm. While initial points of local searches are input by the user or calculated from a uniform probability distribution over the input variable space in order to form a global search algorithm, local searches are formed of a coarse search algorithm, examining broad areas of the input variable space, the Complex Method, and a fine, search algorithm, examining local areas of the output variable space, Powell's Method.

Upon completion of execution of SUBOPT the message:

. . . . . END OF SUBOPT . . . . .

is printed and control of execution is returned to MAINOPT.

SUBOPT

<u>Subroutine Argument List</u>

NAME(J)	contains the name of the Jth input parameter or control variable (maximum of six characters). J = 1,NVAR.
XVAR(J)	contains the default value of the Jth input variable. J = 1,NVAR.

X(J)	contains the value of the Jth search variable for function evaluation. J = 1,N.
S(J)	contains the sensitivity of the objective function with respect to the NSUB(J)th input variable. J = 1,NSENS.
Y(J)	contains temporary results of calculations.
NSUB(J)	contains the number of the input variable for which sensitivity testing is desired. J = 1,NSENS. If NSUB(1) = 0, then the sensitivities of all the input variables will be calculated.
TTS(J,JJ)	contains the JJ^{th} element of the Jth time series for a parameter estimation problem. J = 1,NTS and JJ = 1,NDPTS(JJ).
TSAVE(J,JJ)	contains the model output corresponding to the JJ^{th} data point of the Jth time series, J = 1,NTS and JJ = 1,NDPTS(J).
NDPTS(J)	contains the number of data points in the Jth time series for a parameter estimation problem. J = 1,NTS.
IS	is an integer array containing component control variables affecting the paths of the optimization algorithm execution. Individual elements are defined in appropriate subroutine variable lists.
ONAMES(J)	contains the name of the Jth output variable. J = 1,MOUTS.
XOUT(J)	contains the value for the Jth output variable of the simulation. XOUT is updated every DT time step of the simulation. J = 1,MOUTS.
P(J)	contains the parameter value for the Jth parameter of the objective function in an optimal control problem. J = 1,NPOBJ.
W(J,JJ)	contains the weighting factor for the JJ^{th} element of the Jth time series for the least squares objective function in a parameter estimation problem. By default W(J,JJ) = 1.0. J = 1,NTS and JJ = 1,NDPTS(J).
XIN(J)	contains the value for the Jth input variable of the simulation. J = 1,NVAR.
COM	see variable dimensioning section.
NTS	defined in MAINOPT variable lists.
NDP	defined in MAINOPT variable lists.

NVAR defined in MAINOPT variable lists.

MOUTS defined in MAINOPT variable list.

NTD defined in MAINOPT variable list.

NPOBJ defined in MAINOPT variable list.

Additional Significant Component Variable

BEGTIME contains the beginning CPU execution time of the SYSOPT program.

IS(13) = 0 for solving the first problem with the component

≠ 0 otherwise.

L(J) contains the reference location in the COM array (see the section on the variable dimensioning capability) for storing the elements of the Jth component array in the following list:

J	Component Array
1	NAMV
2	XZ
3	CONS
4	CONSTL
5	CONSTU
6	BOUNDSL (G)
7	BOUNDSU (H)
8	IVAR
9	E
10	not used
11	not used
12	XC
13	W
14	U
15	not used
16	AOLD
17	ANEW
18	UU
19	POLD
20	PNEW
21	X
22	XXX
23	F
24	global optimum variable values (no separate array name)

NGLOBE	contains the number of global searches that have been completed by the component
BEGGOTM	contains the CPU time when the current global search was initiated
COM(L(J)+I-1)	contains the Ith element in the Jth array (see table under definition of L(J))
ENDLOTM	contains the CPU time of the end of the current local search
BEGLOTM	contains the CPU time when the current local search was initiated
FMIN	contains the current global minimum of the problem under investigation
EF	contains the current local minimum of the problem under investigation
ENDGOTM	contains the CPU time of the end of the current global search
FXG	contains the function value of the global optimum of the problem under investigation (at the end of a global search FXG = FMIN)

```
      SUBROUTINE SUBOPT(NAME,XVAR,X,S,Y,NSUB,TTS,TSAVE,NDPTS,IS,ONAMES
     1,XOUT,P,W,XIN,COM,NTS,NDP,NVAR,MOUTS,NTD,NPOBJ)
      DIMENSION NAME(NVAR),XVAR(NVAR),X(NVAR),S(NVAR),Y(NVAR),NSUB(NVAR)
     1 ,TTS(NTS,NDP),TSAVE(NTS,NDP),NDPTS(NTS),IS(20),ONAMES(MOUTS),
     1 XOUT(MOUTS),P(NPOBJ),W(NTS,NDP),XIN(NVAR),COM(1)
C
C           TITLE - MAIN SUBROUTINE IN THE OPTIMIZATION SUB-COMPONENT
C
C           PURPOSE - SUBOPT CONTROLS THE BASIC LOGIC EXECUTION FLOW IN
C                THE SUB-COMPONENT IN ADDITION TO INTERACTION AND OUTPUT
C
C           FUNCTIONS OR SUBROUTINES CALLED - SECOND, DATA1, DATA2,
C                DATA3, SENS, SETFL, NEXT, DATA4, VAR, SEQSW, GLOBE,
C                RANSET, BOX, SWITCH, BOTM, CONV, COMP, SPEED
C
      DIMENSION L(30)
C
C           FOR FIRST ITERATION THROUGH SUBOPT SET IS(13) = 0
      IS(13)=0
C           SET BEGTIME TO CPU TIME
      BEGTIME=SECOND(A)
C           INITIALIZE COST PARAMETERS
      COSTMP=1.0
      PRINT 1040
      READ 2040,RG
1     CONTINUE
C           CALL DATA1 IN THE DATA INPUT SUBROUTINE HIERARCHY - INPUT OF
C                LEAST FREQUENTLY CHANGED VARIABLE VALUES
      CALL DATA1(NDPTS,TTS,NAME,XVAR,ONAMES,IS,NTS,NVAR,MOUTS,NDP)
      PRINT 999
      READ 1999,INT
2     CONTINUE
C           CALL DATA2 TO INPUT OPTIMIZATION ALGORITHM PARAMETERS
C                (COMPLEX METHOD + POWELLS METHOD)
      CALL DATA2(SALPHA,ITMAX,IPRINT1,ALPHA,BETA,GAMMA,IPRINT2,MAXIT,
     1 ESCALE,ERST,IS,ADD,DEV,DELTA)
3     CONTINUE
C           CALL DATA3 TO INPUT OBJECTIVE FUNCTION PARAMETERS (PARAMETER
C                ESTIMATION OR OPTIMAL CONTROL PROBLEM)
      CALL DATA3(IS,NDPTS,W,P,PENALTY,NTS,NDP,NPOBJ)
      IF(IS(13).EQ.0) GO TO  31
C           INPUT YES TO RETAIN VARIABLE SEARCH SET AND SKIP SENS.
C                TESTING
      PRINT 1000
      READ 2000,ANS
      IF(ANS.EQ.3HYES) GO TO 9
31    CONTINUE
C           INPUT YES TO SKIP SENSITIVITY TESTING
      PRINT 1045
      READ  2041,ANS
      IF(ANS.EQ.3HYES) GO TO 810
C           INPUT NUMBER OF VARIABLES AND VARIABLE VALUES TO BE CHANGED
C                (FROM DEFAULT VALUES) FOR SENSITIVITY TESTING
4     CONTINUE
      PRINT 1002
      READ 2002,NCHG
      IF(NCHG.EQ.0) GO TO 5
      DO 6 J=1,NCHG
      PRINT 1003
6     READ 2003,JJ,XVAR(JJ)
5     CONTINUE
C           IF IS(7) = 0 CALCULATE OBJECTIVE FUNCTION SENSITIVITIES FOR
C                ALL INPUT VARIABLES
      IF(IS(7).EQ.0) GO TO 7
C           INPUT DESIRED INPUT VARIABLE SUBSET FOR SENSITIVITY TESTING
      PRINT 1004
      READ 2004,NSENS
      PRINT 1005
      READ 2005,(NSUB(J),J=1,NSENS)
      GO TO 8
7     CONTINUE
      NSENS=NVAR
      NSUB(1)=0
8     CONTINUE
C           CALL SENS TO COMPUTE SENSITIVITIES
      CALL SENS(XVAR,NSENS,S,Y,DEV,NSUB,NVAR,IVAR,XIN,XOUT,IS,NTS,
     1   NDPTS,NDP,1,W,INT,0,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,
```

```
      2   TSAVE,AOLD,ANEW,UU,POLD,PNEW,MOUTS,1)
       PRINT 1006
       DO 81 J=1,NSENS
       NJ=NSUB(J)
 81    PRINT 10061,NJ,NAME(NJ),XVAR(NJ),S(J)
 C          INPUT YES TO REPEAT SENSITIVITY TESTING
       PRINT 1007
       READ 2007,ANS
       IF(ANS.EQ.3HYES) GO TO 4
 C          INPUT YES TO EXIT FROM PROGRAM
       PRINT 1008
       READ 2008,ANS
       IF(ANS.EQ.3HYES) GO TO 99
 810   CONTINUE
 C          INPUT SIZE OF SEARCH VARIABLE SET
       PRINT 1009
       READ 2009,N
 C          REDUCE EXTENSION OF BLANK COMMON TO SIZE NEEDED FOR ARRAYS
 C              DIMENSIONED IN MAINOPT
       LAST=1
       CALL SETFL(COM(LAST))
 C          COMPUTE SIZE OF EXTENDED BLANK
 C          COMMON FOR ARRAYS WHOSE SIZES DEPEND ON N
       DO 91 J=1,12
 91    L(J)=NEXT(LAST,N)
       NW=N*(N+3)
       L(13)=NEXT(LAST,NW)
       NSQ=N*N
       L(14)=NEXT(LAST,NSQ)
       L(15)=NEXT(LAST,NSQ)
       NREG=2*N+1+N*(N-1)/2
       NREGSQ=NREG*NREG
       IF(IS(9).EQ.0) NREG=NREGSQ=1
       L(16)=NEXT(LAST,NREG)
       L(17)=NEXT(LAST,NREG)
       L(18)=NEXT(LAST,NREG)
       L(19)=NEXT(LAST,NREGSQ)
       L(20)=NEXT(LAST,NREGSQ)
       NONE=N+1
       NXX=NONE*N
       L(21)=NEXT(LAST,NXX)
       L(22)=NEXT(LAST,NVAR)
       L(23)=NEXT(LAST,NONE)
       L(24)=NEXT(LAST,N)
 C          EXTEND BLANK COMMON (ABOVE THE SIZE REQUIRED BY THE EXTENSION
 C              IN MAINOPT) TO THE SIZE NECESSARY TO STORE THOSE ARRAYS
 C              WHOSE SIZES DEPEND ON N
       CALL SETFL(COM(LAST))
 C          CALL DATA4 TO INPUT SEARCH VARIABLE INFORMATION - NOTE
 C              ELEMENTS OF COM ARRAY BEING ASSOCIATED WITH SUB-COMPONENT
 C              ARRAYS
       CALL DATA4(IS,COM(L(1)),COM(L(2)),COM(L(3)),COM(L(4)),COM(L(5)),
      1 COM(L(6)),COM(L(7)),N,NCONST,COM(L( 9)))
 C          CALL VAR TO VERIFY THAT THE SEARCH VARIABLE NAMES CORRESPOND
 C              TO ACTUAL INPUT VARIABLE NAMES
       CALL VAR(NAME,COM(L(1)),COM(L(8)),NVAR,N)
 9     CONTINUE
 C          SET IS(13) = 1 FOR SUBSEQUENT ITERATIONS THROUGH SUPOPT
       IS(13)=1
       NGLOBE=0
 C          IF A GLOBAL POLYNOMIAL APPROXIMATION OF THE OBJECTIVE
 C              FUNCTION IS TO BE MADE, INITIALIZE SEQSW VARIABLES
       IF(IS(9).NE.2) GO TO 97
       ISET=1
       CALL SEQSW(X,N,F,ISET,NREG,COM(L(16)),COM(L(17)),COM(L(19)),
      1  COM(L(20)),COM(L(18)))
 97    CONTINUE
 C          SET JGL TO DESIRED NUMBER OF LOCAL SEARCHES TO BE MADE FOR
 C              THE GLOBAL SEARCH
       JGL=IS(3)
       IF(IS(3).EQ.0) JGL=1
 C          BEGGOTM IS THE CPU TIME AT THE BEGINNING OF THE GLOBAL SEARCH
       BEGGOTM=SECOND(A)
       DO 11 IGL=1,JGL
 C          BEGLOTM IS THE CPU TIME AT THE BEGINNING OF THE LOCAL
 C              SEARCHES
       BEGLOTM=SECOND(A)
```

```
C           IS THE INITIAL POINT FOR THE LOCAL SEARCH TO BE RANDOMLY
C                GENERATED WITHIN THE INPUT VARIABLE SPACE
      IF(IS(8).EQ.0) GO TO 121
      PRINT 1010
      READ 2010,ANS
      IF(ANS.EQ.2HNO) GO TO 121
C           INPUT INITIAL POINT FOR THE LOCAL SEARCH
      PRINT 10101
      READ 20101,(COM(L(2)+J-1),J=1,N)
      GO TO 12
C           RANDOMLY GENERATE INITIAL POINT IN INPUT VARIABLE SPACE
121   CALL GLOBE(COM(L(6)),COM(L(7)),X,N,NVAR)
      DO 122 J=1,N
122   COM(L(2)+J-1)=X(J)
C           IF A LOCAL POLYNOMIAL APPROXIMATION OF THE OBJECTIVE FUNCTION
C                IS TO BE MADE, INITIALIZE SEQSW VARIABLES
12    IF(IS(9).NE.1)GO TO 10
      ISET=1
      CALL SEQSW(X,N,F,ISET,NREG,COM(L(16)),COM(L(17)),COM(L(19)),
     1   COM(L(20)),COM(L(18)))
10    CONTINUE
C           SET SIMULATION INPUT VARIABLES TO INPUT VARIABLE DEFAULT
C                VALUES
      DO 101 J=1,NVAR
101   XIN(J)=XVAR(J)
      K=N+1
      IC=0
      M=N
      CALL RANSET(1.0)
C           CALL COARSE OPTIMIZATION ALGORITHM
      CALL BOX(COM(L(21)),COM(L(2)),N,M,K,ITMAX,IC,IPRINT1,ALPHA,
     1   BETA,GAMMA,DELTA,SALPHA,COM(L(5)),COM(L(7)),COM(L(22)),
     2   COM(L(23)),EF,COM(L(12)),COM(L(8)),XIN,XOUT,IS,NTS,NDPTS,NDP,
     3   NREG,W,INT,NCONST,COM(L(3)),COM(L(4)),COM(L(5)),PENALTY,P,
     4   NPOBJ,TTS,TSAVE,COM(L(16)),COM(L(17)),COM(L(18)),COM(L(19)),
     5   COM(L(20)),NVAR,MOUTS,NONE)
C           OUTPUT NUMBER OF FUNCTION EVALUATIONS COMPLETED IN BOX
      PRINT 1043,IS(20)
C           DOES USER WISH TO EXERCISE DATA INPUT (CHANGE) OR EXIT
C                OPTIONS
      PRINT 1041
      READ 2041,ANS
      IF(ANS.EQ.3HYES) GO TO 1923
      IAX=0
C           IS CONJUGATE SWITCHING BETWEEN THE COMPLEX METHOD AND POWELLS
C                METHOD DESIRED
      IF(IS(10).EQ.0.OR.IS(9).NE.1) GO TO 13
      IAX=1
C           IF A LOCAL POLYNOMIAL APPROXIMATION TO THE OBJECTIVE FUNCTION
C                WAS MADE AND INITIALIZATION OF POWELLS METHOD WITH
C                CONJUGATE SEARCH DIRECTIONS IS DESIRED CALL SWITCH
      CALL SWITCH(COM(L(16)),N,NREG,COM(L(14)),COM(L(12)),COM(L(15)))
13    CONTINUE
C           CALL FINE OPTIMIZATION ALGORITHM
      CALL BOTM(COM(L(22)),COM(L(9)),N,EF,ESCALE,IPRINT2,MAXIT,
     1   COM(L(14)),COM(L(13)),IAX,COM(L(8)),XIN,XOUT,IS,NTS,NDPTS,
     2   NDP,NREG,W,INT,NCONST,COM(L(3)),COM(L(4)),COM(L(5)),PENALTY,
     3   P,NPOBJ,TTS,TSAVE,COM(L(16)),COM(L(17)),COM(L(18)),COM(L(19)),
     3   COM(L(20)),NVAR,MOUTS,NW)
      DO 135 J=1,N
135   X(J)=COM(L(22)+J-1)
C           IF DESIRED OUTPUT POLYNOMIAL APPROXIMATION COEFFICIENTS
      PRINT 10102
      IF(IS(9).EQ.1.AND.IS(2).EQ.0)PRINT 1011,(COM(L(16)+J-1),J=1,NREG)
C           OUTPUT LOCAL OPTIMUM INFORMATION
      DO 131 JJ=1,N
      CALL CONV(COM(L(8)+JJ-1),II)
131   PRINT 1012,JJ,NAME(II),X(JJ)
      IF(NCONST.EQ.0) GO TO 132
      PRINT 1013
      DO 133J=1,NCONST
      CALL CONV(COM(L(3)+ J-1),II)
      CALL CONV(COM(L(8)+II-1),JJ)
133   PRINT 10131,NAME(JJ),COM(L(4)+J-1),COM(L(5)+J-1)
132   CONTINUE
      PRINT 1014,EF
C           ENDLOTM IS THE CPU TIME AT THE END OF THE LOCAL SEARCH
```

```
      ENDLOTM=SECOND(A)
      TM=ENDLOTM-BEGLOTM
      PRINT 1015,TM
C           CALL COMP TO CALCULATE ACCUMULATED CPU, MEMORY, CORE STORAGE,
C                AND TELETYPE COSTS AND USAGE
      CALL COMP(COST,CPCOST,PPCOST,CMCOST,CTCOST,CPTIME,PPTIME,CTTIME,PG
     1   ,COSTMP)
      PRINT 1016,COST,CPCOST,PPCOST,CMCOST,CTCOST
C           ONLY IF THIS IS FIRST LOCAL OPTIMUM SEARCH DO NOT UPDATE
C                CURRENT GLOBAL OPTIMUM
      IF(IGL.NE.1) GO TO 14
      FMIN=EF
      DO 15 J=1,N
15    COM(L(24)+J-1)=X(J)
      GO TO 161
14    IF(EF.GT.FMIN) GO TO 161
C           UPDATE CURRENT GLOBAL OPTIMUM WITH THIS LOCAL OPTIMUM
      FMIN=EF
      DO 16 J=1,N
16    COM(L(24)+J-1)=X(J)
161   CONTINUE
C           OUTPUT CURRENT GLOBAL OPTIMUM INFORMATION
      PRINT 10171
      DO 162 JJ=1,N
      CALL CONV(COM(L(8)+JJ-1),II)
162   PRINT 1017,JJ,NAME(II),COM(L(24)+JJ-1)
      PRINT 1018,FMIN
C           END GLOBAL SEARCH LOOP
11    CONTINUE
C           INCREASE NUMBER OF GLOBAL SEARCHES CONDUCTED BY ONE
      NGLOBE=NGLOBE+1
C           X CONTAINS CURRENT GLOBAL OPTIMUM
      DO 111J=1,N
111   X(J)=COM(L(24)+J-1)
C           IF IS(5).NE.0 POWELLS SPEED UP METHOD FOR IMPROVING GLOBAL
C                OPTIMUM APPROXIMATIONS IS DESIRED AND IS EXECUTED BY
C                A CALL TO SPEED
      IF(IS(5).EQ.0) GO TO 17
      CALL SPEED(X,N,FMIN,ERST,COM(L(12)),FXG,ADD,COM(L(9)),ESCALE,
     1   IPRINT2,MAXIT,COM(L(14)),COM(L(13)),COM(L(8)),XIN,XOUT,IS,NTS,
     2   NDPTS,NDP,NREG,W,INT,NCONST,COM(L( 3)),COM(L(4)),COM(L(5)),
     3   PENALTY,P,NPOBJ,TTS,TSAVE,COM(L(16)),COM(L(17)),COM(L(18)),
     4   COM(L(19)),COM(L(20)),NVAR,MOUTS,NW,S,Y,DEV,NSUB,COM(L(1)))
      GO TO 18
17    DO 19 J=1,N
19    COM(L(12)+J-1)=COM(L(24)+J-1)
      FXG=FMIN
18    CONTINUE
C           OUTPUT GLOBAL OPTIMUM INFORMATION
      PRINT 10181
C           ENDGOTM IS THE CPU TIME AT THE END OF THE GLOBAL SEARCH
      ENDGOTM=SECOND(A)
      TM=ENDGOTM-BEGGOTM
C           IF DESIRED OUTPUT GLOBAL POLYNOMIAL APPROXIMATION
C                COEFFICIENTS
      IF(IS(9).EQ.2.AND.IS(2).EQ.0)PRINT 1019,(COM(L(16)+J-1),J=1,NREG)
      DO 191 JJ=1,N
      CALL CONV(COM(L(8)+JJ-1),II)
191   PRINT 1020,JJ,NAME(II),COM(L(12)+JJ-1)
      IF(NCONST.EQ.0) GO TO 196
      PRINT 1013
      DO 1910 J=1,NCONST
      CALL CONV(COM(L(3)+ J-1),II)
      CALL CONV(COM(L(8)+II-1),JJ)
1910  PRINT 10131,NAME(JJ),COM(L(4)+J-1),COM(L(5)+J-1)
196   CONTINUE
      PRINT 1022,FXG
C           COMPUTE ACCUMULATED CPU, PP, CM, TTY COSTS AND USAGE
      CALL COMP(COST,CPCOST,PPCOST,CMCOST,CTCOST,CPTIME,PPTIME,CTTIME,PG
     1   ,COSTMP)
      PRINT 1023,TM
      PRINT 1016,COST,CPCOST,PPCOST,CMCOST,CTCOST
      IGL=IGL-1
      PRINT 1025,IGL,NGLOBE
      DO 1911 J=1,NVAR
1911  COM(L(22)+J-1)=XVAR(J)
      DO 1912 J=1,N
```

```
      CALL CONV(COM(L(8)+J-1),II)
1912  COM(L(22)+II-1)=COM(L(12)+J-1)
C           IS SENSITIVITY ANALYSIS TO BE PERFORMED ON THE GLOBAL OPTIMUM
      PRINT 1045
      READ 2041,ANS
      IF(ANS.EQ.3HYES) GO TO 1923
C           PERFORM SENSITIVITY COMPUTATIONS ON ALL INPUT VARIABLES
      NSUB(1)=1
      NSENS=NVAR
      CALL SENS(COM(L(22)),NSENS,S,Y,DEV,NSUB,NVAR,COM(L(8)),XIN,XOUT,IS
     1  ,NTS,
     1NDPTS,NDP,NREG,W,INT,J,COM(L( 3)),COM(L(4)),COM(L(5)),PENALTY,
     2   P,NPOBJ,TTS,TSAVE,COM(L(16)),COM(L(17)),COM(L(18)),COM(L(19)),
     3   COM(L(20)),MOUTS,N)
C           OUTPUT SENSITIVITY ANALYSIS RESULTS
      PRINT 1006
      DO 192 J=1,NSENS
      NJ=NSUB(J)
192   PRINT 1026,NJ,NAME(NJ),COM(L(22)+J-1),S(J)
1923  CONTINUE
C
C           EXIT FROM PROGRAM
      PRINT 1027
      READ 2027,ANS
      IF(ANS.EQ.3HYES) GO TO 99
C
C           CHANGE DEFAULT VALUES OF INPUT VARIABLES
      PRINT 1028
      READ 2028,ANS
      IF(ANS.EQ.2HNO) GO TO 194
C           INPUT NEW VARIABLE VALUES
      PRINT 1029
      READ 2029,NCHG
      DO 193 J=1,NCHG
      PRINT 1030
193   READ 2030,JJ,XVAR(JJ)
194   CONTINUE
C
C           RETURN TO DATA1 INPUT SUBROUTINE
      PRINT 1031
      READ 2031,ANS
      IF(ANS.EQ.3HYES)GO TO 1
C
C           RETURN TO DATA2 INPUT SUBROUTINE
      PRINT 1032
      READ 2032,ANS
      IF(ANS.EQ.3HYES) GO TO 2
C
C           RETURN TO DATA3 INPUT SUBROUTINE
      PRINT 1033
      READ 2033,ANS
      IF(ANS.EQ.3HYES) GO TO 3
C
C           RETURN TO SENSITIVITY TESTING
      PRINT 1034
      READ 2034,ANS
      IF(ANS.EQ.3HYES) GO TO 4
C
C           RETURN TO DATA4 INPUT SUBROUTINE
      PRINT 1051
      READ 2041,ANS
      IF(ANS.EQ.3HYES) GO TO 810
C           BY DEFAULT RETAIN SEARCH VARIABLE SET AND RETURN TO GLOBAL
C                SEARCH
      GO TO 9
99    CONTINUE
      PRINT 2035,NGLOBE
      PRINT 2036
      RETURN
1000  FORMAT(* TYPE YES TO RETAIN VARIABLE SEARCH SET AND SKIP SENS. TES
     1TING (A6)*)
2000  FORMAT(A6)
1002  FORMAT(* TYPE IN THE NUMBER OF DEFAULT VALUES TO BE CHANGED (I3)*)
2002  FORMAT(I3)
1003  FORMAT(* TYPE IN THE NUMBER AND DEFAULT VALUE FOR A VARIABLE (I3,F
     1 10.5)*)
2003  FORMAT(I3,F10.5)
```

```
1004  FORMAT(* TYPE IN THE NUMBER OF VARIABLES FOR SENS TESTING (I3)*)
2004  FORMAT(I3)
1005  FORMAT(* TYPE IN THE VARIABLE NUMBERS FOR SENS TESTING (10I3)*)
2005  FORMAT(10I3)
1006  FORMAT(* ...SENSITIVITY TESTING OUTPUT...*/* NO.   NAME   DEF. VAL
     1UE    SENSITIVITY *)
10061 FORMAT(I4,3X,A8,2F12.5)
1007  FORMAT(* TYPE YES TO REPEAT SENS TESTING (A6)*)
2007  FORMAT(A6)
1008  FORMAT(* TYPE YES TO EXIT FROM PROGRAM (A6)*)
2008  FORMAT(A6)
1009  FORMAT(* TYPE IN THE NUMBER OF VARIABLES TO BE SEARCHED-N (I3)*)
2009  FORMAT(I3)
999   FORMAT(* TYPE IN THE NUMBER OF DT UNITS FOR OUTPUT SPACING (I5)*)
1999  FORMAT(I5)
1010  FORMAT(* TYPE YES TO INPUT INITIAL POINT FOR THIS GLOBAL SEARCH (A
     16)*)
2010  FORMAT(A6)
10101 FORMAT(* TYPE IN INITIAL POINT FOR GLOBAL SEARCH (7F10.5)*)
20101 FORMAT(7F10.5)
1011  FORMAT(* ..POLYNOMIAL COEF. FOR THE LOCAL APPROX. ARE..*/(1X,5E14.
     1  5))
1012  FORMAT(3X,I5,2X,A6,2X,F10.5)
10102 FORMAT(* .....LOCAL OPTIMUM OUTPUT.....*)
1013  FORMAT(* ...THE CONSTRAINTS ARE...*)
10131 FORMAT(2X,A6,2X,F10.5,2X,F10.5)
1014  FORMAT(*.. THE LOCAL OPTIMUM FUNCTION VALUE=*,E15.5)
1015  FORMAT(* ..ELAPSED TIME FOR THIS LOCAL OPT. CALC.=*,F10.5)
1016  FORMAT(* THE PRESENT COSTS ARE (DOLLARS) TOTAL,CP,PP,CM,CT*/(2X,5F
     110.5))
10171 FORMAT(* ..THE PRESENT MINIMUM OF LOCAL OPT. IS..*)
1017  FORMAT(2X,I5,2X,A6,2X,F10.5)
1018  FORMAT(* THE PRESENT MINIMUM VALUE =*,E15.5)
10181 FORMAT(* .....GLOBAL OPTIMUM OUTPUT.....*)
1019  FORMAT(* ..POLYNOMIAL COEF. FOR THE GLOBAL APPROX. ARE..*/(1X,5E14
     1  .5))
1020  FORMAT(3X,I5,2X,A6,2X,F10.5)
1021  FORMAT(* ...THE CONSTRAINTS ARE...*/(2X,A6,2X,F10.5,2X,F10.5))
1022  FORMAT(* THE GLOBAL OPTIMU FUNCTION VALUE=*,E15.5)
1023  FORMAT(* THE ELAPSED TIME FOR THIS GLOBAL SEARCH=*,F10.5)
1025  FORMAT(* THE NO. OF LOCAL SEARCHES CONDUCTED FOR THIS GLOBAL SEARC
     1H=*,I4/* THE NO. OF PROBLEMS INVESTIGATED SO FAR=*,I4)
1921  FORMAT(* ..SENS.DATA AT GLOBAL OPTIMUMARE VARIABLE NO.,NAME,VALUE,
     1 AND SENSITIVITY*)
1026  FORMAT(2X,I4,2X,A6,2X,F10.5,2X,F10.5)
1027  FORMAT(* TYPE YES TO EXIT FROM OPT PROGRAM (A6)*)
2027  FORMAT(A6)
1028  FORMAT(* TYPE YES TO CHANGE DEFAULT VARIABLE VALUES (A6)*)
2028  FORMAT(A6)
1029  FORMAT(* TYPE IN THE NUMBER OF DEFAULT VALUES TO BE CHANGED (I3)*)
2029  FORMAT(I3)
1030  FORMAT(* TYPE IN A VARIABLE NO. AND NEW DEFAULT VALUE (I4,F10.5)*)
2030  FORMAT(I4,F10.5)
1031  FORMAT(* TYPE YES TO RETURN TO DATA1 (A6)*)
2031  FORMAT(A6)
1032  FORMAT(* TYPE YES TO RETURN TO DATA2 (A6)*)
2032  FORMAT(A6)
1033  FORMAT(* TYPE YES TO RETURN TO DATA3 (A6)*)
2033  FORMAT(A6)
1034  FORMAT(* TYPE YES TO RETURN TO SENS. TESTING (A6)*)
2034  FORMAT(A6)
2035  FORMAT(2X,I4,*  PROBLEMS INVESTIGATED*)
2036  FORMAT(* .......END OF SUBOPT.......*)
1040  FORMAT(* TYPE IN THE RATE GROUP FOR THIS EXECUTION (F1.0)*)
2040  FORMAT(F1.0)
1041  FORMAT(* TYPE YES TO OBTAIN DATA RETURN AND EXIT OPTIONS (A6)*)
2041  FORMAT(A6)
1043  FORMAT(* FUNCTION EVALUATIONS IN BOX=*,I10)
1045  FORMAT(* TYPE YES TO SKIP SENS TESTING (A6)*)
1051  FORMAT(* TYPE YES TO RETURN TO DATA4 (A6)*)
      END
```

Subroutines DATA1, DATA2, DATA3, and DATA4

The four subroutines DATA1, DATA2, DATA3, and DATA4 form a data input hierarchy within the optimization sub-component. Different sets of data necessary to execute the component are read in each subroutine. With this hierarchical subroutine structure, it is possible to investigate different problems of interest to the user, to build optimization methods for the most efficient solution convergence, and to perform a wide variety of useful computational system analysis functions during the execution of the component. Interactions with the component, through the DATA subroutines, is accomplished with a teletype terminal having a high priority of entrance into the computer. This subroutine hierarchy provides for a flexible and convenient, but powerful, format for changing variable values in the component.

At various points during the execution of the optimization sub-component of SYSOPT, the user is given opportunity to return to any of the DATA input subroutines. All interaction with the component, through a teletype terminal, is initiated in response to SYSOPT output statements of the form:

TYPE <u>variable description</u> (input format)

In response to these statements, the user inputs, in the specified format, the desired value of the described variable. Anytime a "TYPE" statement appears in the output, it is necessary for the user to respond so that execution of the component continues. In order to facilitate interaction with the component, the user should have the computer execute any data input designation programs. These programs alert the user, by output of a fixed character on the teletype terminal, that the computer is waiting for input data. On the CDC 6500 computer

the data input designation program is PROMPT which is executed by the control statement;

PROMPT,ON.

DATA1 is the highest level of the data input structure for the optimization sub-component. Input to this subroutine consists of the variable sets that define the basic problem under investigation and are least likely to change during program execution. These are comprised of two variable sets: (1) model variables, and (2) primary interaction definition variables. The model variables include any input or output variables of the simulation being studied that are necessary for calculation of the objective function. Included in the interactive variable set are those variables specifying the form, type, and limits of optimizations performed with the component. All input to this subroutine is performed through a teletype terminal.

Subroutine Argument List

NDPTS(J)	contains the number of data points in the Jth time series for a parameter estimation problem J = 1,NTS (NTS defined later)
TTS(J,JJ)	contains the JJ^{th} element of the Jth time series for a parameter estimation problem JJ = 1,NTS JJ = 1,NDPTS(JJ)
NAME(J)	contains the name of the Jth input parameter or control variable (maximum of six characters) J = 1,NVAR (NVAR defined later)
XVAR(J)	contains the default value of the Jth input variable J = 1,NVAR
ONAMES(J)	contains the name of the Jth output variable J = 1,MOUTS (MOUTS defined later)

IS	is an integer array containing component control variables which affect the path of optimization algorithm execution. (Individual elements are defined later.)
NTS	contains the number of variables represented by the time series TTS in a parameter estimation problem. In an optimal control problem, NTS is set equal to 1.
NVAR	contains the number of input, parameter or control, variables XIN(1), XIN(2), ..., XIN(NVAR) in the simulation model being studied (subroutine SIM)
MOUTS	contains the number of output variables XOUT(1), XOUT(2), ..., XOUT(MOUTS) in the simulation being studied (subroutine SIM)
NDP	contains the maximum number of data points in the NTS time series

Additional Significant Component Variables

IS(1)	= 0 for a parameter estimation problem
	= 1 for an optimal control problem
KEY	= 0 if there was no smoothing of the real world time series in the pattern recognition sub-component of SYSOPT
	≠ 0 if there was smoothing of the time series
NTSDES	= 0 if the original real world time series are to be used for least squares error evaluation
	= 1 if the smoothed time series from the linear filter is desired
	= 2 if the smoothed time series from the quadratic filter is desired
IS(3)	contains the maximum number of local searches to be conducted for the global optimization search
IS(2)	= 0 to print the coefficients of the polynomial approximation to the objective function and
	= 1 not to print the coefficients

IS(5) = 0 for no speed-up of Powells method along sharp ridges in the objective function and

= 1 for execution of the speed-up method at any presumed global optima

IS(7) = 0 if sensitivity testing of the objective function in the entire input variable set is desired and

= 1 for sensitivity testing only on subsets of the input variable set

IS(9) = 0 for no polynomial approximation to the objective function

= 1 for local polynomial approximations (about local optima)

= 2 for global polynomial approximations

IS(8) = 0 for no interaction (selecting initial points for local searches) in global searches

= 1 for optional interaction in global searches

IS(10) = 0 for regular switching between the Complex method and Powell's method

= 1 for switching with the construction of approximate initial conjugate search directions for Powell's method

IS(12) = 0 for switching interaction between the Complex method and Powell's method after every iteration of the Complex method

= +n for switching interaction after every n iterations of the Complex method

= -n for automatic switching after n iterations of the Complex method

IS(11) = IS(5)

```
      SUBROUTINE DATA1(NDPTS,TTS,NAME,XVAR,ONAMES,IS,NTS,NVAR,MOUTS,NDP)
C
C           TITLE - FIRST LEVEL OF DATA INPUT TO THE OPTIMIZATION SUB-
C                COMPONENT
C
C           PURPOSE - TO INPUT VARIABLE SETS THAT DEFINE THE BASIC
C                PROBLEM UNDER INVESTIGATION AND THAT ARE LEAST LIKELY
C                TO CHANGE DURING PROGRAM EXECUTION
C
C           FUNCTIONS AND SUBROUTINES CALLED - NONE
      DIMENSION NDPTS(NTS),TTS(NTS,NDP),NAME(NVAR),XVAR(NVAR),
     1  ONAMES(MOUTS),IS(20)
C           READ INFORMATION STORED ON TAPE98 THAT WAS GENERATED IN THE
C                PATTERN RECOGNITION SUB-COMPONENT
      PRINT 1000
      READ 2000,ANS
      IF(ANS.EQ.2HNO) GO TO 1
C           READ DATA ON TAPE98
      READ(98)IS(1)
      READ(98)
      READ(98)(NAME(J),J=1,NVAR)
      READ(98)
      READ(98)(ONAMES(J),J=1,MOUTS)
      IF(IS(1).NE.0) GO TO 5
      READ(98)(NDPTS(J),J=1,NTS)
      READ(98)KEY
      IF(KEY.NE.0) GO TO 2
      NTSDES=0
      GO TO 3
2     CONTINUE
C           INPUT 0 FOR USE OF ACTUAL REAL WORLD TIME SERIES AND 1 FOR
C                USE OF SMOOTHED REAL WORLD TIME SERIES
      PRINT 1001
      READ 2001,NTSDES
3     NTSDES=NTSDES+1
      DO 4 J=1,NTSDES
      DO 4 JJ=1,NTS
      ND=NDPTS(JJ)
4     READ(98)(TTS(JJ,JJJ),JJJ=1,ND)
5     CONTINUE
C           INPUT DEFAULT VARIABLE VALUES
      PRINT 1003
      READ 2003,(XVAR(J),J=1,NVAR)
      GO TO 10
1     CONTINUE
C           INPUT FLAG = 0 FOR A PARAMETER ESTIMATION PROBLEM AND
C                      = 1 FOR AN OPTIMAL CONTROL PROBLEM
      PRINT 999
      READ 899,IS(1)
      IF(IS(1).NE.0) GO TO 6
C           INPUT TIME SERIES FOR PARAMETER ESTIMATION PROBLEM
      DO 7 J=1,NTS
      PRINT 997,J
      READ 897,NDPTS(J)
      PRINT 996
      ND=NDPTS(J)
7     READ 896,(TTS(J,JJ),JJ=1,ND)
6     CONTINUE
C           INPUT INPUT VARIABLE NAMES
      DO 8 J=1,NVAR
      PRINT 990,J
8     READ 890,NAME(J),XVAR(J)
C           INPUT OUTPUT VARIABLE NAMES
      PRINT 987
      READ 887,(ONAMES(J),J=1,MOUTS)
C           INPUT INSTRUCTION ARRAY ELEMENTS TO STRUCTURE PROBLEM
C                SOLUTION
10    PRINT 992
      READ 892,IS(3)
      PRINT 993
      READ 893,IS(2)
      PRINT 1004
      READ 2004,IS(5)
      PRINT 1005
      READ 2005,IS(7)
      PRINT 1006
      READ 2006,IS(9)
```

```
      PRINT 1007
      READ 2007,IS(8)
      PRINT 1008
      READ 2008,IS(10)
      PRINT 1009
      READ 2009,IS(12)
      IS(11)=IS(5)
      REWIND 98
      RETURN
1000  FORMAT(* TYPE YES TO READ INFORMATION FROM TAPE98 (A6)*)
2000  FORMAT(A6)
1001  FORMAT(* TYPE 0 FOR REAL WORLD TIME SERIES, OR 1 FOR SMOOTHED TIME
     1 SERIES (I1)*)
2001  FORMAT(I1)
1003  FORMAT(* TYPE IN THE DEFAULT VALUES FOR THE VARIABLES (7F10.5)*
     1 (I4,2X,A6))
2003  FORMAT(7F10.5)
1004  FORMAT(* TYPE 0 FOR NO SPEED-UP OF POWELLS METHOD (I1)*)
2004  FORMAT(I1)
1005  FORMAT(* TYPE 0 FOR DEFAULT SENS. TESTING ON WHOLE VARIABLE SET (I
     1I)*)
2005  FORMAT(I1)
1006  FORMAT(* TYPE 0 FOR NO POLY APPROX., 1 FOR LOCAL APPROX., AND 2 F
     1OR GLOBAL APPROX. (I1)*)
2006  FORMAT(I1)
1007  FORMAT(* TYPE 0 FOR NO INTERACTION IN GLOBAL SEARCH (I1)*)
2007  FORMAT(I1)
1008  FORMAT(* TYPE 0 FOR REGULAR SWITCHING (I1)*)
2008  FORMAT(I1)
1009  FORMAT(* TYPE 0 FOR MANUAL TRANS, +N FOR INTERACTIVE TRANS AFTER N
     1 ITERATIONS OF BOX,*/* -N FOR AUTOMATIC SWITCHING AFTER N ITERATIO
     1NS  (I3)*)
2009  FORMAT(I3)
999   FORMAT(* TYPE 0 FOR A P E PROBLEM AND 1 IF O C PROBLEM (I1)*)
899   FORMAT(I1)
997   FORMAT(* TYPE THE NUMBER OF DATA POINTS IN THE *,I4,* TIME SERIES
     1(I3)*)
897   FORMAT(I3)
996   FORMAT(* TYPE IN THE DATA POINTS ONE BY ONE (7F10.5)*)
896   FORMAT(7F10.5)
990   FORMAT(* THE*,I3,* VARIABLE NAMEIS(A6)AND DEFAULT VALUE(F10.5)*)
890   FORMAT(A6,F10.5)
987   FORMAT(* TYPE IN OUTPUT NAMES (10A6)*)
887   FORMAT(10A6)
992   FORMAT(* TYPE MAX. NUMBER OF GLOBAL ITERATIONS (I2)*)
892   FORMAT(I2)
993   FORMAT(* TYPE 0 TO PRINT POLY. APPROX. COEF (I1)*)
893   FORMAT(I1)
      END
```

DATA2, the second level of data subroutines in the optimization sub-component, inputs variable values that fix the specific optimization algorithm parameters. These variables critically affect the convergence properties of the complex-Powell, and Powell's speed-up methods. With this subroutine the user can "tune" the optimization sub-component to most efficiently solve the class of problems under investigation.

Subroutine Argument List

SALPHA	contains a scaling factor (≥ 1) that determines the size of the initial complex in the complex method with vertices randomly generated around an initial point
ITMAX	contains the maximum number of iterations allowed in the complex method
IPRINT1	is a parameter controlling the printing of intermediate iteration results in the complex method = 1 for intermediate values to print on each iteration = 0 for suppression of printing until the final solution is obtained
ALPHA	is a reflection factor of the complex
BETA	is a convergence parameter between successive iterations of the complex method
GAMMA	contains a complex convergence parameter for variables along constraint boundaries
IPRINT2	contains a parameter controlling printing in Powell's method = 2 for printing of intermediate values after every linear search = 1 for printing of intermediate values after every iteration = 0 suppresses printing until the final solution is obtained

MAXIT	contains the maximum number of iterations in Powell's method
ESCALE	contains a scaling parameter controlling the search step size
ERST	contains a parameter that determines whether the speed-up procedure of Powell's method will be used to attempt to improve the global optimum approximation
IS	defined in DATA1 variable lists
ADD	contains the basic interval for stepping along the most sensitive variable values along a sharp valley in the input variable space
DEV	contains a parameter controlling the percent change in the independent variables for the sensitivity testing
DELTA	contains the explicit constraint violation correction in the complex method

Additional Significant Component Variables

IS(5)	defined in DATA1 variable lists
IS(15)	= +n for an output interval (printing) of n iterations in the complex method
IS(16)	= 0 for no interaction in Powell's method
	= 1 for interaction (to exit) after every iteration
	= 2 for interaction after every linear search
	= 3 for interaction after every function evaluation
IS(17)	= +n for an exit from the complex method after a vertex remains a minimum for n function evaluations
IS(13)	= 0 for the execution of the optimization sub-component during the first numerical optimization
	$\neq$ 0 afterwards

```
      SUBROUTINE DATA2(SALPHA,ITMAX,IPRINT1,ALPHA,BETA,GAMMA,IPRINT2,
     1 MAXIT,ESCALE,ERST,IS,ADD,DEV,DELTA)
C
C           TITLE - DATA2 IS THE SECOND LEVEL IN THE DATA INPUT
C                 SUBROUTINE HIERARCHY IN THE OPTIMIZATION SUB-COMPONENT
C
C           PURPOSE - TO INPUT OPTIMIZATION METHOD (POWELLS, COMPLEX,
C                 POWELL SPEED-UP) PARAMETERS
C
C           FUNCTIONS OR SUBROUTINES CALLED - NONE
C
      DIMENSION IS(1)
C           IF THIS IS FIRST PASS THROUGH SUPOPT - SKIP INTERROGATIVE TO
C                 RETAIN DEFAULT PARAMETER VALUES (COMPLEX METHOD)
      IF(IS(13).EQ.0) GO TO 4
      PRINT 999
      READ 899,ANS
      IF(ANS.EQ.3HYES) GO TO 1
4     CONTINUE
C           INPUT COMPLEX METHOD PARAMETERS
      PRINT 998
      READ 898,SALPHA,ITMAX,IPRINT1,ALPHA,BETA,GAMMA,DELTA
1     CONTINUE
      IF(IS(13).EQ.0) GO TO 5
C           RETAIN DEFAULT PARAMETERS IN POWELLS METHOD
      PRINT 997
      READ 897,ANS
      IF(ANS.EQ.3HYES) GO TO 2
5     CONTINUE
C           INPUT PARAMETER VALUES FOR POWELLS METHOD
      PRINT 996
      READ 896,IPRINT2,MAXIT,ESCALE
2     CONTINUE
      IF(IS(5).EQ.0) GO TO 3
C           INPUT PARAMETER VALUES FOR POWELLS SPEED-UP
      PRINT 994
      READ 894,ERST
      PRINT 993
      READ 893,ADD
3     CONTINUE
C           INPUT DEV - PERCENT CHANGE IN INDEPENDENT VARIABLES - FOR
C                 SENSITIVITY ANALYSIS
      PRINT 992
      READ 892,DEV
C           INPUT INSTRUCTION ARRAY ELEMENTS TO STRUCTURE OPTIMIZATION
C                 ALGORITHMS
      PRINT 1040
      READ 2040,IS(15)
      PRINT 1041
      READ 2041,IS(16)
      PRINT 1042
      READ 2042,IS(17)
      RETURN
1040  FORMAT(* TYPE THE OUTPUT ITERATION INTERVAL FOR COMPLEX (I3)*)
2040  FORMAT(I3)
1041  FORMAT(* TYPE 0, 1, 2, OR 3 FOR INTERACTION IN POWELL (I1)*)
2041  FORMAT(I1)
1042  FORMAT(* TYPE IN COMPLEX MAX. NO. LOW REPEATER (I2)*)
2042  FORMAT(I2)
999   FORMAT(* TYPE YES FOR DEFAULT PARAMETERS IN COMPLEX METHOD(A6)*)
899   FORMAT(A6)
998   FORMAT(* TYPE SALPHA(F8.5), ITMAX(I4), IPRINT1(I2), ALPHA, BETA, G
     1AMMA,DELTA(4F8.5)*)
898   FORMAT(F8.5,I4,I2,4F8.5)
997   FORMAT(* TYPE YES FOR DEFAULT PARAMETERS IN POWELLS METHOD(A6)*)
897   FORMAT(A6)
996   FORMAT(* TYPE IPRINT2(I2), MAXIT(I4), ESCALE(F8.5)*)
896   FORMAT(I2,I4,F8.5)
994   FORMAT(* TYPE CONVERGENCE LIMIT FOR SPEED UP(F8.5)*)
894   FORMAT(F8.5)
993   FORMAT(* TYPE ADD (F8.5) FOR SPEED UP OF POWELLS METHOD*)
893   FORMAT(F8.5)
992   FORMAT(* TYPE DEV (F8.5) FOR SENSITIVITY TESTING *)
892   FORMAT(F8.5)
      END
```

Parameter values that specify the objective function are input in the subroutine DATA3. This includes weighting factors of residuals in least squares minimization problems and weighting components of an optimal control performance index. In many problems of interest, there are multiple performance indices; however, either by constructing a goal programming problem or by combining the multiple indices into a single function through a set of weights, optimal system performance can be examined with the component. Both small changes in objective function parameters and changes of entire objective functions can be accomplished with DATA2 during SYSOPT execution.

Subroutine Argument List

IS	defined in DATA1 variable lists
NDPTS	defined in DATA1 variable lists
W(J,JJ)	contains the weighting factor for the JJ^{th} element of the Jth time series for the least squares objective function in a parameter estimation problem. By default: $$W(J,JJ) = 1.0$$ J = 1,NTS JJ = 1,NDPTS(J)
P(J)	contains the parameter value for the Jth parameter of the objective function in an optimal control problem J = 1,NPOBJ (defined later)
PENALTY	contains a parameter that is added to the objective function each time a constraint is violated by a search variable value
NTS	defined in DATA1 variable lists
NDP	defined in DATA1 variable lists
NPOBJ	contains the number of parameters specifying the form of the objective function in an optimal control problem

Additional Significant Component Variables

IS(18) = 0 for a least squares error criterion in a parameter estimation problem

= 1 for a normalized least squares error criterion in a parameter estimation problem

```
      SUBROUTINE DATA3(IS,NOPTS,W,P,PENALTY,NTS,NDP,NPOBJ)
C
C           TITLE - DATA3 IS THE THIRD LEVEL OF DATA INPUT SUBROUTINES IN
C                 THE OPTIMIZATION SUB-COMPONENT
C
C           PURPOSE - TO INPUT PARAMETERS OF THE OBJECTIVE FUNCTION
C                 EITHER IN A PARAMETER ESTIMATION PROBLEM OR AN OPTIMAL
C                 CONTROL PROBLEM
C
C           FUNCTIONS OR SUBROUTINES CALLED - NONE
      DIMENSION NDPTS(1),W(NTS,1),P(1),IS(20)
      IF(IS(13).EQ.0) GO TO 5
C           RETAIN OBJECTIVE PARAMETER DEFAULT VALUES
      PRINT 1000
      READ 2000,ANS
      IF(ANS.EQ.3HYES) GO TO 99
5     CONTINUE
C           INPUT INSTRUCTION ARRAY ELEMENT TO DETERMINE WEIGHTED OR
C                 NORMALIZED LEAST SQUARES ERROR CRITERION IN A PARAMETER
C                 ESTIMATION PROBLEM
      IF(IS(1).EQ.0) PRINT 1001
      IF(IS(1).EQ.0) READ 2001,IS(13)
C           INPUT CONSTRAINT PENALTY
      PRINT 995
      READ 895,PENALTY
      IF(IS(1).NE.0) GO TO 1
C           USE DEFAULT WEIGHTS FOR LEAST SQUARES ERROR CRITERION
      PRINT 999
      READ 899,ANS
      IF(ANS.EQ.2HNO) GO TO 2
      DO 3 K=1,NTS
      KK=NDPTS(K)
      DO 3 J=1,KK
3     W(K,J)=1.
      RETURN
2     CONTINUE
C           INPUT WEIGHTS FOR LEAST SQUARES ERROR CRITERION
      DO 4 K=1,NTS
      PRINT 998,K
      KK=NDPTS(K)
      READ 898,(W(K,J),J=1,KK)
4     CONTINUE
      RETURN
1     CONTINUE
C           INPUT PARAMETERS OF OPTIMAL CONTROL PERFORMANCE INDEX
      PRINT 997
      READ 897,NP
      IF(NP.EQ.0) GO TO 99
      PRINT 996
      READ 896,(P(K),K=1,NP)
99    CONTINUE
      RETURN
1001  FORMAT(* TYPE 0 FOR LS OBJ FUNC AND 1 FOR WLS OBJ FUNC (I1)*)
2001  FORMAT(I1)
1000  FORMAT(* TYPE YES TO SKIP DATA3-READING OBJ. FUNC. PAR. (A6)*)
2000  FORMAT(A6)
999   FORMAT(* TYPE YES OR NO FOR DEFAULT WEIGHTING ON TIME SERIES(A6)*)
899   FORMAT(A6)
998   FORMAT(* FOR THE*,I3,* TIME SERIES TYPE WEIGHTING COEFFICIENTS OVE
     1R TIME (10F8.5)*)
898   FORMAT(10F8.5)
997   FORMAT(* TYPE NUMBER OF PARAMETERS IN OBJECTIVE FUNCTION (I3)*)
897   FORMAT(I3)
996   FORMAT(* TYPE PARAMETERS OF OBJECTIVE FUNCTION (10F8.5)*)
896   FORMAT(10F8.5)
995   FORMAT(* TYPE PENALTY PARAMETER FOR CONSTRAINT VIOLATION (F8.5)*)
895   FORMAT(F8.5)
      END
```

Subroutine DATA4, which is generally executed more often than the higher level DATA subroutines, inputs search variable names, search variable limits, and search variable initial values. In addition, convergence parameters of Powell's method, specific for each search variable set, are input in DATA4. Any subset of the input variable set can be used to construct a search variable set. With this subroutine, the user can change the variable set over which the objective function is being optimized without halting the execution of the component for reprogramming. This flexibility represents one of the more powerful capabilities of the component.

Subroutine Argument List

IS	defined in DATA1 variable lists
NAMV(J)	contains the name of the Jth search variable J = 1,N (where N is the number of variables in the current search variable set under investigation
XZ(J)	contains the initial value of the Jth search variable J = 1,N
CONS(J)	contains the number of the variable in the search variable set to which the Jth constraint corresponds J = 1,N
CONSTL(J)	contains the lower constraint value of the X(CONS(J)) variable J = 1,NCONST (NCONST defined later)
CONSTU(J)	contains the upper constraint value of the X(CONS(J)) variable J = 1,NCONST
BOUNDSL(J)	contains the lower boundary value for the Jth search variable in the complex method J = 1,N
BOUNDSU(J)	contains the upper boundary value for the Jth search variable in the complex method J = 1,N

N	contains the number of variables in the current search variable set
NCONST	contains the number of variables that are constrained to lie within specified limits in Powell's method
E(J)	contains the successive iteration convergence limit for Powell's method J = 1,N

```
      SUBROUTINE DATA4(IS,NAMV,XZ,CONS,CONSTL,CONSTU,BOUNDSL,BOUNDSU,N,
     1 NCONST,E)
C
C           TITLE-DATA4 IS THE LOWEST LEVEL OF DATA INPUT SUBROUTINES IN
C               THE OPTIMIZATION SUB-COMPONENT
C
C           PURPOSE-TO INPUT VARIABLE VALUES THAT CHANGE MOST FREQUENTLY
C               DURING PROGRAM EXECUTION PARTICULARLY SEARCH VARIABLE
C               INFORMATION
C
C           FUNCTIONS OR SUBROUTINES CALLED-NONE
C
      INTEGER CONS
      DIMENSION IS(20),NAMV(N),XZ(N),CONS(N),CONSTL(N),CONSTU(N),
     1 BOUNDSL(N),BOUNDSU(N),E(N)
C           READ SEARCH VARIABLE NAMES, INITIAL VALUES AND CONVERGENCE
C               PARAMETERS FOR POWELL'S METHOD
      DO 1 J=1,N
      PRINT 998
      READ 898,NAMV(J),XZ(J),E(J)
1     CONTINUE
C           READ ANY CONSTRAINTS ON THE SEARCH VARIABLES
      PRINT 997
      READ 897,NCONST
      IF(NCONST.EQ.0) GO TO 3
      DO 2 J=1,NCONST
      PRINT 995
      READ 895,CONS(J)
      PRINT 996
      READ 896,CONSTL(J),CONSTU(J)
2     CONTINUE
3     CONTINUE
C           READ BOUNDING VALUES OF SEARCH VARIABLES FOR BOTH GLOBAL
C               SEARCHES AND THE COMPLEX METHOD
      DO 4 J=1,N
      PRINT 999,NAMV(J)
4     READ899,BOUNDSL(J),BOUNDSU(J)
      RETURN
999   FORMAT(* TYPE IN THE LOWER AND UPPER BOUNDS FOR *,A8,* (2F10.5)*)
899   FORMAT(2F10.5)
998   FORMAT(* TYPE IN A VARIABLE SEARCH NAME,INITIAL VALUE, AND CONV. L
     1IMIT FOR BOTH*/* (A6,2F10.5)*)
997   FORMAT(* TYPE IN THE NUMBER OF CONSTRAINTS (I2)*)
996   FORMAT(* TYPE IN THE LOWER AND UPPER CONSTRAINTS (2F10.5)*)
995   FORMAT(* TYPE IN THE VARIABLE NUMBER THE CONSTRAINT CORRESPONDS TO
     1 (I2)*)
898   FORMAT(A6,2F10.5)
897   FORMAT(I2)
896   FORMAT(2F10.5)
895   FORMAT(I2)
      END
```

Subroutine VAR

Subroutine VAR is an important part of the data input hierarchy of subroutines. The subroutine verifies that the search variables that are input to DATA4 are proper elements of the input variable set. Executed immediately after DATA4, VAR searches through the entire input variable set once for each search variable. If a match is not found, then an error message is printed on the teletype and the input of another variable name is permitted. After three unsuccessful attempts, an additional error message is printed and the program halts execution. However, once a match is found, the array IVAR provides a link between the search variable set and the input variable set. The use of VAR facilitates the user's input of search variable names into the optimization sub-component.

Subroutine Argument List

NAME(J)	defined in DATA1 variable lists.
NAMV(J)	defined in DATA4 variable lists.
IVAR(J)	contains the number of the input variable to which the Jth search variable corresponds. J = 1,N.
NVAR	defined in DATA1 variable lists.
N	defined in DATA variable lists.

Additional Significant Component Variables

IT	is a counter for the number of times an incorrect variable name is input as a search variable. When IT is greater than or equal to three, an exit from the component is made.

```
      SUBROUTINE VAR(NAME,NAMV,IVAR,NVAR,N)
C
C           TITLE-SUBROUTINE VAR VERIFIES THAT A PROPER SEARCH VARIABLE
C               SET HAS BEEN INPUT IN DATA4
C
C           PURPOSE-TO SEARCH THROUGH THE INPUT VARIABLE SET AND NOTE TO
C               THE USER ANY DISCREPENCIES IN THE SEARCH VARIABLE SET
C
C           FUNCTIONS OR SUBROUTINES CALLED-NONE
C
      DIMENSION NAME(NVAR),NAMV(N),IVAR(N)
C           SEARCH THROUGH THE INPUT VARIABLE SET FOR EACH SEARCH
C               VARIABLE NAME
      DO 1 J=1,N
C           IT IS THE COUNTER FOR THE NUMBER OF UNSUCCESSFUL ATTEMPTS AT
C               CORRECTING AN INVALID SEARCH VARIABLE NAME
      IT=0
C           IVAR(J) DENOTES THE NUMBER OF THE INPUT VARIABLE
C               CORRESPONDING TO THE SEARCH VARIABLE
      IVAR(J)=0
4     CONTINUE
      DO 2 JJ=1,NVAR
      IF(NAMV(J).NE.NAME(JJ)) GOTO 2
C           PROPER MATCH HAS BEEN FOUND
      IVAR(J)=JJ
      GO TO 1
2     CONTINUE
C           TEST FOR WHETHER ANY MATCH HAS BEEN FOUND
      IF(IVAR(J).NE.0) GO TO 3
      PRINT 999,NAMV(J)
      READ 899,NAMV(J)
      IT=IT+1
      IF(IT.LT.3) GO TO 4
      PRINT998
      CALL EXIT
3     CONTINUE
1     CONTINUE
899   FORMAT(A6)
999   FORMAT(//1H ,*..VARIABLE NAME *,A6,* NOT FOUND IN VARIABLE SET..*/
     1/1H ,*  TYPE IN A6 FORMAT CORRECTED VARIABLE NAME    *)
998   FORMAT(///1H ,*.......NO VARIABLE FOUND WITH GIVEN NAME.......*//)
      RETURN
      END
```

Subroutine FUNC

The subroutine FUNC performs the necessary calculations and subroutine calls to evaluate the objective function for a given set of input (search) variable values.

For an optimal control problem FUNC calculates an objective function of the form:

$$J[\underline{a},\underline{u},T,\underline{p},\underline{c}] = M\left(\underline{y}(\underline{a},\underline{u},L\Delta t),LDt,\underline{p}\right) + \Delta t \sum_{i=0}^{k-1} \sum_{j=0}^{m} N\left(\underline{y}(\underline{a},\underline{u},it_u + \frac{jt_u}{m}),\right.$$

$$\left.\underline{u}(it_u),\ it_u + \frac{jt_u}{m}\right) + \sum_{i=1}^{NCONST} P_{penalty} \cdot P_{mult,i}$$

where

$\underline{a}$	is the set of parameters of the model
$\underline{u}$	is the set of control variables of the model
$[0,T]$	is the time horizon
$\underline{p}$	is a set of variables capable of specifying the form of J desired by the user
$\underline{c}$	is the set of constraints on the variables
	$\underline{c}(\underline{x},\underline{u},T,\underline{y},\underline{a}) \geq 0$
$P_{penalty}$	is a penalty parameter added to the objective function each time one of the constraints is violated
$P_{mult,i}$	= 1 if the ith constraint is violated Ø otherwise
m	$= \left[\frac{t_u}{\Delta t}\right]$
L	$= \left[\frac{T}{\Delta t}\right]$

$$\underline{u}(t) = \underline{u}_k \text{ for } (k-1)t_u < t \leq kt_u$$

$$1 \leq k \leq K \quad \text{with} \quad K = \left[\frac{T}{t_u}\right] + 1$$

and $\underline{y}$ is the set of outputs of the model

In the evaluation of J, M penalizes model behavior at the end of the time horizon, N penalizes behavior of the models solution in the time interval [0,T], and the penalty term is added for constraint violation.

If the problem of interest is formulated as an parameter estimation problem, then FUNC calculates a weighted sum of squares error criterion between the model outputs and the real world time series. In this case,

$$J[\underline{a},\underline{w},\underline{y},T] = \sum_{j=1}^{MOUTS} \sum_{i=1}^{L} W_i \left[\frac{y_j(i\Delta t) - Y_j(i\Delta t)}{Y_j^{\gamma}(i\Delta t)}\right]^2$$

where

Y_j is the jth real world (or smoothed real world) time series data point

y_j is the corresponding output of the simulation

MOUTS is the number of model outputs or time series

W_i is the set of weightings on the errors

and γ = 1 for a normalized error criterion

= 0 for a unnormalized error criterion

The subroutine FUNC calls the subroutines MMM and NNN to calculate J, and also calls TRANSFE to insure proper compatability between the search variable value array and the input variable value array. Finally, it should be noted that constraints on the input variables can be handled with constraint violation penalties added to the objective function.

Thus, constrained optimization problems can be converted into unconstrainec problems.

Subroutine Argument List

X(J)	contains the value of the Jth search variable for function evaluation. If IS(14) = 0 J = 1,N If IS(14) ≠ 0 J = 1,NVAR.
N	defined in DATA4 variable lists.
F	contains the objective function value on return from FUNC.
IVAR	defined in VAR variable lists.
XIN(J)	contains the value for the Jth input variable of the simulation. J = 1,NVAR.
XOUT(J)	contains the value for the Jth output variable of the simulation. XOUT is updated every DT time step of the simulation. J = 1,MOUTS.
IS	defined in DATA1 variable lists.
NTS	defined in DATA1 variable lists.
NDPTS	defined in DATA1 variable lists.
NDP	defined in DATA1 variable lists.
NREG	= 2*N+1+N*(N-1)/2 is the number of coefficients in a quadratic approximation of a function in N independent variables.
W	defined in DATA3 variable lists.
INT	contains the number of DT steps corresponding to the real world data sampling interval in a parameter estimation problem.
NCONST	defined in DATA4 variable lists.
CONS	defined in DATA4 variable lists.
CONSTL	defined in DATA4 variable lists.
CONSTU	defined in DATA4 variable lists.

PENALTY	defined in DATA3 variable lists.
P	defined in DATA3 variable lists.
NPOBJ	defined in DATA3 variable lists.
TTS	defined in DATA1 variable lists.
TSAVE(J,JJ)	contains the model output corresponding to the JJ^{th} data point of the Jth time series. J = 1,NTS JJ = 1,NDPTS(J)
AOLD(J)	contains the previous (from last update) J^{th} coefficient of the quadratic approximation to the objective function. J = 1,NREG.
ANEW(J)	contains the new (updated) J^{th} coefficient of the quadratic approximation to the objective function. J = 1,NREG.
UU(J)	contains the value of the quadratic term corresponding to the Jth coefficient. J = 1,NREG.
POLD,PNEW	are arrays containing information necessary for the sequential update of the polynomial approximation to the objective function, both are dimensioned to (NREG,NREG).
NVAR	defined in DATA1 variable lists.
MOUTS	defined in DATA1 variable lists.

Additional Significant Component Variables

NSET	= 0 to initialize the integral component of the optimal control objective function
	= 1 after initialization
IS(14)	= 1 if FUNC was called from the sensitivity testing subroutine SENS
	≠ 1 otherwise
IS(1)	defined in DATA1 variable lists
ISWITCH	= 1 if the simulation model is to be used for function evaluation or plotting only (pattern recognition subcomponent)
	≠ 1 for normal execution of the simulation model (printout, interaction, etc.)

IPLOT	= 0 for function evaluation
	≠ 0 for plotting of model outputs (see SUBPR)
ISET	= 0 to allow updating of quadratic approximation coefficients
	= 1 to initialize the arrays AOLD, POLD for sequential regression calculations
IS(18)	defined in DATA3 variable lists

```
      SUBROUTINE FUNC (X,N,F,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,INT,
     1NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,
     2POLD,PNEW,NVAR,MOUTS)
C
C           TITLE-FUNC CALCULATES OBJECTIVE FUNCTION VALUES
C
C           PURPOSE-TO CALCULATE NORMALIZED OR UNNORMALIZED SUM OF
C                SQUARES ERROR CRITERION FOR A PARAMETER ESTIMATION
C                PROBLEM OR AN INTEGRAL-FINAL TIME CRITERION FOR AN
C                OPTIMAL CONTROL PROBLEM
C
C           FUNCTIONS OR SUBROUTINES CALLED-NNN,TRANSFE,SIM,MMM,SEQSW
C
      DIMENSION X(NVAR),IVAR(N),XIN(NVAR),XOUT(MOUTS),IS(20),NDPTS(NTS),
     1W(NTS,NDP),CONS(N),CONSTL(N),CONSTU(N),P(NPOBJ),TTS(NTS,NDP),TSAVE
     2(NTS,NDP),AOLD(NREG),ANEW(NREG),UU(NREG),POLD(NREG,NREG),PNEW(NREG
     1,NREG)
      INTEGER CONS
      NSET=0
C           IF AN OPTIMAL CONTROL PROBLEM IS BEING SOLVED-INITIALIZE THE
C                INTEGRAL PART OF THE OBJECTIVE FUNCTION
      IF(IS(1).NE.0)CALLNNN(T,DT,P2,XIN,XOUT,P,NVAR,MOUTS,NPOBJ,NSET)
C           ASSIGN SWITCHING AND CAPABILITY VARIABLES
      NSET=1
      ISWITCH=1
      IPLOT=0
      ISET=0
      F=0.
C           IF FUNC IS BEING CALLED FROM SENS ENTIRE INPUT VARIABLE SET
C                VALUES ARE TRANSFERRED TO XIN
C           IF NOT ONLY TRANSFER SEARCH VARIABLE VALUES
      IF(IS(14).EQ.1) CALL TRANSFE(XIN,X,NVAR,IVAR,IS)
      IF(IS(14).NE.1) CALL TRANSFE(XIN,X,N,IVAR,IS)
C           CALL SIMULATION TO COMPUTE MODEL OUTPUTS FOR OBJECTIVE
C                FUNCTION EVALUATION
      CALLSIM(XIN,XOUT,T,DT,P,INT,IPLOT,TSAVE,NTS,NSET,ISWITCH,NVAR,
     1MOUTS,NPOBJ,P2,NTIMES,IS)
      IF(IS(1).NE.0) GO TO 10
C           CALCULATE LEAST SQUARES ERROR CRITERION
      DO 1 J=1,NTS
      ND=NDPTS(J)
      DO 1 JJ=1,ND
      IF(IS(18).NE.0) GO TO 5
C           CALCULATE WEIGHTED-NOT NORMALIZED-LEAST SQUARES ERROR
C                CRITERION
      F=F+W(J,JJ)*(TTS(J,JJ)-TSAVE(J,JJ))**2
      GO TO 1
5     CONTINUE
      IF(ABS(TTS(J,JJ)).GT.1.E-10) GO TO 2
      IF(ABS(TSAVE(J,JJ)).GT.1.E-10) GO TO 3
      GO TO 1
C           CALCULATE WEIGHTED-NORMALIZED WITH TSAVE-LEAST SQUARES ERROR
C                CRITERION
3     F=F+W(J,JJ)*((TTS(J,JJ)-TSAVE(J,JJ))/TSAVE(J,JJ))**2
      GO TO 1
C           CALCULATE WEIGHTED-NORMALIZED WITH TTS-LEAST SQUARES ERROR
C                CRITERION
2     F=F+W(J,JJ)*((TTS(J,JJ)-TSAVE(J,JJ))/TTS(J,JJ))**2
1     CONTINUE
      GO TO 11
 10   CONTINUE
C           CALCULATE OPTIMAL CONTROL OBJECTIVE FUNCTION
C           FIRST COMPUTE P1-THE FINAL TIME BEHAVIOR PENALTY
      CALLMMM(T,DT,P1,XIN,XOUT,P,NVAR,MOUTS,NPOBJ)
C           P2 IS THE INTEGRAL BEHAVIOR PENALTY CALCULATED IN SIM
      F=P1+P2
 11   CONTINUE
      IF(IS(14).EQ.1.OR.IS(9).EQ.0)GO TO 14
C           UPDATE POLYNOMIAL APPROXIMATING COEFFICIENTS
      CALL SEQSW(X,N,F,ISET,NREG,AOLD,ANEW,POLD,PNEW,UU)
 14   CONTINUE
C           ADD ANY PENALTY CONSTRAINTS IF THERE ARE VIOLATIONS
      IF (NCONST.EQ.0) GO TO 13
      DO 12 J=1,NCONST
      PMULT=0.
      IF(X(CONS(J)).LT.CONSTL(J).OR.X(CONS(J)).GT.CONSTU(J)) PMULT=1.
      F=F+PENALTY*PMULT
```

```
12    CONTINUE
13    CONTINUE
      RETURN
      END
```

Subroutine GLOBE

Subroutine GLOBE contains an algorithm that generates initial points in the search variable space for conducting local searches. For an n-dimensional optimization problem, the initial points are computed as

X(J) = BOUNDSL(J) + R(J)*(BOUNDSU(J) - BOUNDSL(J))

J = 1,N

where

X = (X(1), X(2), ..., X(N)) is the initial point in the search variable space and BOUNDSL, BOUNDSU are arrays containing upper and lower bounds for the search variables

R(J) is a random variable uniformly distributed on [0,1]

On the other hand, with the appropriate selection of interactive variable values, it is possible to externally specify initial points for local searches.

Subroutine Argument List

BOUNDSL	defined in DATA4 variable lists.
BOUNDSU	defined in DATA4 variable lists.
X	defined in FUNC variable lists.
N	defined in DATA4 variable lists.
NVAR	defined in DATA1 variable lists.

```
      SUBROUTINE GLOBE(BOUNDSL,BOUNDSU,X,N,NVAR)
C
C           TITLE-SUBROUTINE TO GENERATE INITIAL POINTS FOR GLOBAL SEARCH
C
C           PURPOSE-TO GENERATE RANDOM POINTS IN THE SEARCH VARIABLE
C               SPACE THAT ARE TO BE USED AS INITIAL POINTS IN LOCAL
C               SEARCHES IN A GLOBAL SEARCH
C
C           FUNCTIONS OR SUBROUTINES CALLED-RANF
C
      DIMENSION BOUNDSL(N),BOUNDSU(N),X(NVAR)
C
C           FOR EACH COMPONENT OF THE SEARCH VARIABLE GENERATE A VALUE
C               RANDOMLY WITHIN THE INTERVAL [BOUNDSL(J),BOUNDSU(J)]
C
      DO 1 I=1,N
    1 X(I)=BOUNDSL(I)+RANF(1.)*(BOUNDSU(I)-BOUNDSL(I))
      RETURN
      END
```

Subroutine CONV

CONV is a type conversion subroutine used in the optimization subcomponent because the variable dimensioning scheme used to extend or reduce the amount of core storage needed for proper program execution incorporates a real array COM, in BLOCK COMMON. As a result, integers are stored as integer representations in real variables. Thus there is a need for converting these real expressions into equivalent integer expressions in integer variables. Subroutine CONV accomplishes this by equating an integer variable in a subroutine argument list to a real variable in a subroutine call.

Subroutine Argument List	
I	is an integer variable that is equated to a real variable containing an integer expression.
II	contains the integer expression of the real variable, passed to I, on return from the subroutine.

```
      SUBROUTINE CONV(I,II)
C
C          TITLE-CONV IS USED TO CONVERT AN INTEGER EXPRESSION STORED IN
C              A REAL VARIABLE TO AN INTEGER EXPRESSION STORED IN AN
C              INTEGER VARIABLE
C
C          PURPOSE-IF SUBROUTINE CONV IS CALLED WITH ≠CALL SUBROUTINE
C              CONV(A,J),≠ THEN THE EXPRESSION IN A WILL BE CONVERTED
C              INTO AN INTEGER AND STORED IN J ON RETURN FROM CONV
C
C          FUNCTIONS OR SUBROUTINES CALLED-NONE
C
C          THE INTEGER STORED IN I MUST BE ASSIGNED TO ANOTHER INTEGER
C              VARIABLE SO THAT THE ORIGINAL EXPRESSION IS NOT DESTROYED
C
      II=I
      RETURN
      END
```

Subroutine TRANSFE

Matching of search variable values with proper input variable values is accomplished with the use of subroutine TRANSFE. This subroutine insures correlation betwen the two variable sets, input and search, by usinq the array IVAR which is calculated in subroutine VAR. The input variable values to the simulation are set from both the base (default) values of the input variables read in subroutine DATA1 and the search variable values passed from subroutine FUNC.

Subroutine Argument List

XIN	defined in FUNC variable lists.
X	defined in FUNC variable lists.
N	defined in DATA4 variable lists.
IVAR	defined in VAR variable lists.
IS	defined in DATA1 variable lists.

```
      SUBROUTINE TRANSFE(XIN,X,N,IVAR,IS)
C
C          TITLE-TRANSFE TRANSFERS DATA VALUES FROM ARRAY X TO ARRAY XIN
C
C          PURPOSE-TO PROPERLY CORRELATE SEARCH VARIABLES, INPUT
C              VARIABLE VALUES, AND SIMULATION VARIABLE VALUES
C
C          FUNCTIONS OR SUBROUTINES CALLED-NONE
C
      DIMENSION X(1), XIN(1), IVAR(1),IS(20)
C
C          IF TRANSFE IS BEING CALLED FROM SENSITIVITY TESTING, ASSIGN
C              ALL INPUT DEFAULT VALUES (AS SPECIFIED IN SUBROUTINE
C              SENS) TO THE SIMULATION VARIABLE VALUES
C
      IF(IS(14).EQ.1) GO TO 2
C
C          ASSIGN SEARCH VARIABLE VALUES TO APPROPRIATE SIMULATION
C              VARIABLES VALUES AS SPECIFIED BY THE ARRAY IVAR
C
      DO 1 J=1,N
      I= IVAR(J)
    1 XIN(I)=  X(J)
      RETURN
    2 CONTINUE
      DO 3 J=1,N
    3 XIN(J)=X(J)
      RETURN
      END
```

Subroutine SENS

Sensitivity testing on the objective function is performed in subroutine SENS. If the objective function is given by $J[\underline{a},\underline{u},T]$, then the sensitivity of J with respect to an $x \in \{\underline{a},\underline{u},T\}$ is given by:

$$S_x = \frac{\text{\% change in J due to a change in x}}{\text{\% change in x}}$$

$$= \frac{\Delta J/J}{\Delta x/x}$$

In addition, if the sensitivity of J at x_0 is desired and $J(x_0) = 0.$, then S_{x0} is calculated in SYSOPT AS $\Delta J/\Delta x$. For particular problems, a possible improvement would be to approximate

$$S_x = \frac{\Delta J/J}{\Delta x/x} \quad \text{by}$$

$$S_x \simeq \frac{\Delta x}{J(\Delta x)} \left[\frac{J(\Delta x + \Delta x^2) - J(\Delta x - \Delta x^2)}{2\Delta x^2}\right]$$

$$= \frac{J(\Delta x + \Delta x^2) - J(\Delta x - \Delta x^2)}{2J(\Delta x)\Delta x}$$

when $J(0) = 0$, $x = 0$ and $J(\Delta x) \neq 0$.

This approximation can easily be programmed into SYSOPT within the subroutine SENS.

In the subroutine, the sensitivity of the objective function can be calculated for all the input variables (requiring n+1 simulation runs with n input variables) or for user specified subsets of the input variable set; thus, unnecessary simulation runs are eliminated. The desired mode of execution can be specified by the input of an interactive variable in DATA1.

Sensitivity tests on the objective function identify those variables that most strongly affect changes in the performance or behavior of a model. As a result, these tests are important in the solution of parameter

estimation and policy decision problems for at least four reasons. First, the relative sensitivities of the variables may indicate search variable subsets that could allow for rapid and large improvements in the objective function. Second, the input variable sensitivities may indicate where a good initial point for a search is located. Third, by evaluating the sensitivities at different points near a base set of values, some interactions in the variables may be isolated. This would allow a reduction in the size of the search variable sets necessary to find an optimum of the objective function. As a result, a corresponding reduction would occur in the number of simulation runs needed for the optimization. Last, a sensitivity analysis at the optimum in a parameter estimation problem can assist in determining whether the model being studied is a valid representation of the real world.

Subroutine Argument List

X	defined in FUNC variable lists.
NSENS	contains the number of input variables for which sensitivity testing is to be performed.
S(J)	contains the sensitivity of the objective function with respect to the NSUB(J) 'th input variable J = 1,NSENS.
Y	is an array storing temporary calculations.
DEV	defined in DATA2 variable lists.
NSUB(J)	contains the number of the input variable for which sensitivity testing is desired J = 1,NSENS. In addition if NSUB(1) = 0, then the sensitivity of all the input variables will be calculated.
NVAR	defined in DATA1 variable lists.
IVAR	defined in VAR variable lists.
XIN	defined in FUNC variable lists.
XOUT	defined in FUNC variable lists.
IS	defined in DATA1 variable lists.
NTS	defined in DATA1 variable lists.
NDPTS	defined in DATA1 variable lists.
NDP	defined in DATA1 variable lists.

NREG	defined in FUNC variable lists.
W	defined in DATA3 variable lists.
INT	defined in FUNC variable lists.
NCONST	defined in DATA4 variable lists.
CONS	defined in DATA4 variable lists.
CONSTL	defined in DATA4 variable lists.
CONSTU	defined in DATA4 variable lists.
PENALTY	defined in DATA3 variable lists.
P	defined in DATA3 variable lists.
NPOBJ	defined in DATA3 variable lists.
TTS	defined in DATA1 variable lists.
TSAVE	defined in FUNC variable lists.
AOLD	defined in FUNC variable lists.
ANEW	defined in FUNC variable lists.
UU	defined in FUNC variable lists.
POLD	defined in FUNC variable lists.
PNEW	defined in FUNC variable lists.
MOUTS	defined in DATA1 variable lists.
N	defined in DATA4 variable lists.
IS(14)	defined in FUNC variable lists.
ZBASE	contains the objective function value at the base point for which the sensitivities are desired.
Z	is a real variable containing the intermediate function values corresponding to deviations from the base point.

```
      SUBROUTINE SENS(X,NSENS,S,Y,DEV,NSUB,NVAR,IVAR,XIN,XOUT,IS,NTS,
     1  NDPTS,NDP,NREG,W,INT,NCONST,CONS,CONSTL,CONSTU,PENALTY,P,
     1  NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,PNEW,MOUTS,N)
C
C           TITLE-SENS COMPUTES OBJECTIVE FUNCTION SENSITIVITIES TO INPUT
C               VARIABLE CHANGES
C
C           PURPOSE-TO IDENTIFY INPUT VARIABLES THAT MOST STRONGLY AFFECT
C               CHANGES IN THE PERFORMANCE OR BEHAVIOR OF A MODEL
C
C           FUNCTIONS OR SUBROUTINES CALLED-FUNC,ABS
C
      DIMENSION X(NVAR),S(NVAR),IVAR(N),XIN(NVAR),XOUT(MOUTS),IS(20),
     1  NDPTS(NTS),W(NTS,NDP),CONS(N),CONSTL(N),CONSTU(N),P(NPOBJ),
     2  TTS(NTS,NDP),TSAVE(NTS,NDP),AOLD(NREG),ANEW(NREG),UU(NREG),
     1  POLD(NREG,NREG),PNEW(NREG,NREG),Y(NVAR),NSUB(NVAR)
      INTEGER CONS
C
C           SET SWITCH TO INDICATE CALLS TO FUNC ARE MADE FROM SENS
      IS(14)=1
C
C           CALL FUNC TO COMPUTE ZBASE-THE OBJECTIVE FUNCTION VALUE
C               CORRESPONDING TO THE BASE VARIABLE VALUES-X
      CALL FUNC(X,N,ZBASE,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,INT,
     1  NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,
     1  UU,POLD,PNEW,NVAR,MOUTS)
C
C           TEST TO DETERMINE IF ENTIRE VARIABLE SET IS TO BE USED FOR
C               SENSITIVITY TESTING
      IF(NSUB(1).NE.0) GO TO 1
C
C           CONSTRUCT NSUB ARRAY FOR TESTING ENTIRE INPUT VARIABLE SET
      DO 2 J=1,NVAR
2     NSUB(J)=J
C
C           ASSIGN X ARRAY TO Y ARRAY
1     DO 3 J=1,NVAR
3     Y(J)=X(J)
C
C           ASSIGN TO THE Y ELEMENT,THE BASE VARIABLE VALUE PLUS PERCENT
C               DEVIATION FOR SENSITIVITY TESTING
C           IF THE BASE VARIABLE VALUE IS NEAR 0., USE ABSOLUTE DEVIATION
C               FOR SENSITIVITY TESTING
      DO 4 J=1,NSENS
      NS=NSUB(J)
      IF(ABS(X(NS)).LT.1.E-8) GO TO 5
      Y(NS)=(1.0+DEV)*X(NS)
      GO TO 6
5     Y(NS)=X(NS)+DEV
6     CONTINUE
C
C           COMPUTE Z, THE OBJECTIVE FUNCTION VALUE CORRESPONDING TO THE
C               DISPLACED VARIABLE VALUE
      CALL FUNC(Y,N,Z     ,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,INT,
     1  NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,
     1  UU,POLD,PNEW,NVAR,MOUTS)
C           COMPUTE SENSITIVITY-IF THE BASE VARIABLE VALUE IS NEAR
C               COMPUTE APPROXIMATION TO DERIVATIVE AT BASE POINT
      IF(ABS(X(NS)).LT.1.E-8) GO TO 7
      S(J)=((Z-ZBASE)/ZBASE)/((Y(NS)-X(NS))/X(NS))
      GO TO 4
7     S(J)=(Z-ZBASE)/DEV
4     Y(NS)=X(NS)
C           RESET SWITCH INDICATING FROM WHICH SUBROUTINE FUNC WAS CALLED
      IS(14)=0
      RETURN
      END
```

Subroutines BOX, CONSX, CENTR, CHECK, and CONST

The set of four subroutines BOX, CONSX, CENTR, and CHECK form an integrated algorithm for performing the Complex optimization search developed by G. Box [1]. This algorithm is primarily used to examine broad areas of the search variable space. There are two primary reasons why the Complex method was chosen for inclusion in SYSOPT: (1) in the Complex method regions of the search variable space containing local optima are rapidly isolated and (2) convergence of the Complex method is relatively independent of objective function evaluation noise. Thus, the use of this method provides a base for structuring an efficient two-level search algorithm that is effective both far from and in the vicinity of local optima.

All of the subroutines, BOX, CONSX, CENTR, CHECK, and CONST, are slightly modified versions of programs available from a commercial optimization package [3]. Changes made to the commercial package consist of additional print and interaction options, automatic generation of the initial complex in the search variable space, scaling of the initial complex and providing proper links between these subroutines and the remainder of the component.

The four subroutines, BOX, CONSX, CENTR, and CHECK perform significantly different functions in the program. While BOX is called into execution first, the primary subroutine for executing logic decisions and numerical calculations is CONSX. BOX is mainly used to initialize variables, print final results of the complex search procedure, and print error messages. On the other hand, CONSX generates the initial complex, calculates the coordinates of the new complexes, calls for objective function evaluation, controls the paths of execution of the

method, tests for variable convergence, and checks for cyclical behavior of the complex. To assist CONSX in performing these functions, the subroutine CHECK tests if explicit or implicit constraints of the variables are violated; and if there are any vertex constraint violations, then this subroutine creates a new feasible vertex to replace the infeasible one. In addition, the subroutine CENTR calculates the centroid of the current complex which is used in the generation of new complexes.

Finally, the subroutine CONST is needed to insure proper execution of the component. This subroutine performs no functions in SYSOPT, but must be included in programmed form as:

```
SUBROUTINE CONST(N,M,K,X,G,H,I,NONE)
DIMENSION X(NONE,N),G(N),H(N)
RETURN
END
```

If implicit constraints in the variables or explicit constraints with upper and lower bounds dependent upon the search variable values are to be included in the problem formulation, then this subroutine must be programmed as detailed in reference [3].

For further information concerning the subroutines, including logic flow charts, detailed variable descriptions, an example of use, and programming considerations, see the commercial optimization package literature.

Subroutine BOX Argument List

X(I,J)	contains the Jth coordinate of the Ith vertex in the current complex.
XZ(J)	contains the Jth coordinate of the first vertex of the initial complex.
N	contains the number of variables in the input variable search set.
M	contains the number of sets of constraints.
K	contains the number of vertices in the complex.
ITMAX	contains the maximum number of iterations of the complex method.
IC	contains the number of implicit variables (= 0 during SYSOPT execution)
IPRINT	contains a variable to control printing of intermediate results = 1 for print of intermediate results after each iteration = 0 for suppression of printing until a final solution is obtained (= IPRINT1 in DATA2)
ALPHA	contains a reflection factor for the generation of new complexes in the search.

BETA	contains a variable convergence parameter.
GAMMA	contains an iteration convergence parameter.
DELTA	contains an explicit constraint violation parameter.
SALPHA	contains a parameter for scaling the initial complex.
G	contains lower bounds for the search variables (= BOUNDSL).
H	contains upper bounds for the search variables (= BOUNDSU).
XXX	contains intermediate results of calculations, and input variable values for objective function evaluation.
F(J)	contains the objective function value corresponding to the Jth complex vertex. J = 1,K
EF	contains the minimum value of objective function when convergence is obtained.
XC(J)	contains the Jth coordinate of the centroid of the current complex. J = 1,N
IVAR	defined in VAR variable lists.
XIN	defined in FUNC variable lists.
XOUT	defined in FUNC variable lists.
IS	defined in DATA1 variable lists.
NTS	defined in DATA1 variable lists.
NDPTS	defined in DATA1 variable lists.
NDP	defined in DATA1 variable lists.
NREG	defined in FUNC variable lists.
W	defined in DATA3 variable lists.
INT	defined in FUNC variable lists.
NCONST	defined in DATA4 variable lists.
CONS	defined in DATA4 variable lists.

CONSTL	defined in DATA4 variable lists.
CONSTU	defined in DATA4 variable lists.
PENALTY	defined in DATA3 variable lists.
P	defined in DATA3 variable lists.
NPOBJ	defined in DATA3 variable lists.
TTS	defined in DATA1 variable lists.
TSAVE	defined in FUNC variable lists.
AOLD	defined in FUNC variable lists.
ANEW	defined in FUNC variable lists.
UU	defined in FUNC variable lists.
POLD	defined in FUNC variable lists.
PNEW	defined in FUNC variable lists.
NVAR	defined in DATA1 variable lists.
MOUTS	defined in DATA1 variable lists.
NONE	= N + 1, used for variable dimensioning capability.

Additional Significant Component Variables

IS(20)	contains a variable to count objective function evaluations for switching between the Complex method and Powell's method.

```
      SUBROUTINE BOX(X,XZ,N,M,K,ITMAX,IC,IPRINT,ALPHA,BETA,GAMMA,DELTA,
     1  SALPHA,G,H,XXX,F,EF,XC,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,
     2  INT,NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,
     3  ANEW,UU,POLD,PNEW,NVAR,MOUTS,NONE)
      DIMENSION X(NONE,N),XXX(NVAR),F(NONE),XC(N),H(N), G(N),XIN(NVAR)
     1  ,XOUT(MOUTS),IS(20),W(NTS,NDP),CONS(N),CONSTL(N),CONSTU(N),
     2  P(NPOBJ),TTS(NTS,NDP),TSAVE(NTS,NDP),AOLD(NREG),ANEW(NREG),
     3  UU(NREG),POLD(NREG,NREG),PNEW(NREG,NREG)
      DIMENSION XZ(N),IVAR(N),NDPTS(NTS)
      INTEGER CONS
      IS(20)=0
      IAMMA=GAMMA+.000.1
      DO 1 J=1,N
1     X(1,J)=XZ(J)
      IF(ITMAX.GT.J) GO TO 2
      DO 3 J=1,N
3     XXX(J)=X(1,J)
      RETURN
2     CONTINUE
      PRINT 10
10    FORMAT(1H , /,18X,24HCOMPLEX PROCEDURE OF BOX)
      PRINT 18
18    FORMAT(    2X,1HPARAMETERS )
      PRINT 11,N,M,K,ITMAX,IC,ALPHA,BETA,IAMMA,DELTA
  011 FORMAT (    2X,4HN = ,I2,3X,4HM = ,I2,3X,4HK = ,I2,2X,8HITMAX = ,
     1I4,2X,5HIC = ,I2, /,2X,8HALPHA = ,F5.2,5X,7HBETA = ,F10.5,3X,
     28HGAMMA = ,I2,3X,8HDELTA = ,F6.5)
50    CALL CONSX(N,M,K,ITMAX,ALPHA,BETA,IAMMA,X,F,IT,IEV2,NO,G,H,XC,
     1    IPRINT,XXX,EF,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,INT,NCONST,
     2    CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,
     3    PNEW,NVAR,MOUTS,NONE,DELTA,SALPHA)
      IF (IT-ITMAX) 20,20,30
   20 PRINT 14, F(IEV2)
  014 FORMAT (  /,2X,30HFINAL VALUE OF THE FUNCTION = ,E20.8)
      PRINT 15
  015 FORMAT (    2X,14HFINAL X VALUES)
      DO 300 J=1,N
      PRINT 16, J, X(IEV2,J)
  016 FORMAT (  2X,2HX(,I2,4H) = ,E20.8)
  300 CONTINUE
      PRINT 9, IT
9     FORMAT (  * THE NUMBER OF ITERATIONS IS    *, I5)
      GO TO 999
   30 PRINT 17, ITMAX
  017 FORMAT (    2X,38HTHE NUMBER OF ITERATIONS HAS EXCEEDED ,I4,10X,
     118HPROGRAM TERMINATED)
999   RETURN
      END
```

Subroutine CONSX Argument List

N	defined in DATA4 variable list.
M	defined in BOX variable lists.
K	defined in BOX variable lists.
ITMAX	defined in BOX variable lists.
ALPHA	defined in BOX variable lists.
BETA	defined in BOX variable lists.
GAMMA	defined in BOX variable lists.
X	defined in BOX variable lists.
F	defined in BOX variable lists.
IT	contains a variable to count Complex iterations.
IEV2	contains the number of the complex vertex corresponding to the current minimum value of the objective function.
NO	not used.
G	defined in BOX variable lists.
H	defined in BOX variable lists.
XC	defined in BOX variable lists.
IPRINT	defined in BOX variable lists.
XXX	defined in BOX variable lists.
EF	defined in BOX variable lists.
IVAR	defined in VAR variable lists.
XIN	defined in FUNC variable lists.
XOUT	defined in FUNC variable lists.
IS	defined in DATA1 variable lists.
NTS	defined in DATA1 variable lists.
NDPTS	defined in DATA1 variable lists.

NDP	defined in DATA1 variable lists.
NREG	defined in FUNC variable lists.
W	defined in DATA3 variable lists.
INT	defined in FUNC variable lists.
NCONST	defined in DATA4 variable lists.
CONS	defined in DATA4 variable lists.
CONSTL	defined in DATA4 variable lists.
CONSTU	defined in DATA4 variable lists.
PENALTY	defined in DATA3 variable lists.
P	defined in DATA3 variable lists.
NPOBJ	defined in DATA3 variable lists.
TTS	defined in DATA1 variable lists.
TSAVE	defined in FUNC variable lists.
AOLD	defined in FUNC variable lists.
ANEW	defined in FUNC variable lists.
UU	defined in FUNC variable lists.
POLD	defined in FUNC variable lists.
PNEW	defined in FUNC variable lists.
NVAR	defined in DATA1 variable lists.
MOUTS	defined in DATA1 variable lists.
NONE	defined in BOX variable lists.
DELTA	defined in BOX variable lists.
SALPHA	defined in BOX variable lists.

Additional Significant Component Variables

KNTLOW	contains the number of times a single vertex repeats in having the minimum value of the objective function.

```
      SUBROUTINE CONSX(N,M,K,ITMAX,ALPHA,BETA,GAMMA,X,F,IT,IEV2,NO,G,H,
     1   XC,IPRINT,XXX,EF,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,INT,
     2   NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,
     3   UU,POLD,PNEW,NVAR,MOUTS,NONE,DELTA,SALPHA)
      DIMENSION X(NONE,N),F(NONE),G(N),H(N),XC(N),XXX(NVAR),IVAR(N),
     1   XIN(NVAR),XOUT(MOUTS),IS(20),NDPTS(NTS),W(NTS,NDP),CONS(N),
     2   CONSTL(N),CONSTU(N),P(NPOBJ),TTS(NTS,NDP),TSAVE(NTS,NDP),
     3   AOLD(NREG),ANEW(NREG),UU(NREG),POLD(NREG,NREG),PNEW(NREG,NREG)
      INTEGER CONS
      INTEGER GAMMA
      IT = 1
      KODE = 0
      IF (M-N) 20,20,10
   10 KODE = 1
   20 CONTINUE
      DO 40 II=2,K
      DO 30 J=1,N
   30 X(II,J) = 0.0
   40 CONTINUE
      DO 65 II=2,K
      DO 50 J=1,N
      I = II
      CALL CONST (N,M,K,X,G,H,I,NONE)
      X(II,J)=X(1,J)+(RANF(1.)-0.5)*(H(J)-G(J))/SALPHA
   50 CONTINUE
      K1 = II
      CALL CHECK (N,M,K,X,G,H,I,KODE,XC,DELTA,K1,NONE)
      IF (II-2) 51, 51, 55
   51 IF (IPRINT) 52,65,52
   52 PRINT 18
  018 FORMAT (   2X,30HCOORDINATES OF INITIAL COMPLEX)
      IO = 1
      PRINT 19, (IO, J, X(IO,J), J=1,N)
  019 FORMAT (  2(2X,2HX(,I2,1H,,I2,4H) = ,1PE13.6))
   55 IF (IPRINT) 56,65,56
   56 PRINT 19, (II, J, X(II,J), J=1,N)
   65 CONTINUE
      K1 = K
      DO 70 I=1,K
      DO 651 J=1,N
651   XXX(J)=X(I,J)
      IS(20)=IS(20)+1
      CALL FUNC(XXX,N,Z,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,INT,
     1  NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,
     2  UU,POLD,PNEW,NVAR,MOUTS)
      F(I)=-Z
70      CONTINUE
       KOUNT=1
       IA=0
      IF(IPRINT) 72,80,72
72    PRINT 21
21    FORMAT(  2X,22HVALUES OF THE FUNCTION  )
      PRINT 22 , (J,F(J), J=1,K)
22    FORMAT(  2(2X,2HF(,I2,4H) = ,1PE13.6))
80    IEV1=1
      DO 100 ICM=2,K
       IF(F(IEV1)-F(ICM)) 100,100,90
90    IEV1=ICM
100    CONTINUE
      IEV2=1
      DO 120 ICM=2,K
      IF(F(IEV2)-F(ICM)) 110,110,120
 110    IEV2=ICM
120   CONTINUE
      IF(F(IEV2)-(F(IEV1)+BETA)) 140,130,130
130    KOUNT=1
      GO TO 150
140     KOUNT=KOUNT+1
      IF(KOUNT-GAMMA)150,240,240
150    CALL CENTR(N,M,K,IEV1,I,XC,X,K1,NONE)
      DO 160 JJ=1,N
160    X(IEV1,JJ)=(1.0+ALPHA)*(XC(JJ))-ALPHA*(X(IEV1,JJ))
      I=IEV1
      CALL CHECK(N,M,K,X,G,H,I,KODE,XC,DELTA,K1,NONE)
      DO 161 J=1,N
161   XXX(J)=X(I,J)
      IS(20)=IS(20)+1
```

```
      CALL FUNC(XXX,N,Z,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,INT,
     1  NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,
     2  UU,POLD,PNEW,NVAR,MOUTS)
      F(I)=-Z
      KNTLOW=0
      IIEV2=IEV2
170    IEV2=1
      KNTLOW=KNTLOW+1
      IF(KNTLOW.LT.IS(17)) GO TO 171
      IEV2=IIEV2
      GO TO 240
171   CONTINUE
      DO 190  ICM=2,K
      IF(F(IEV2)-F(ICM)) 190,190,180
180   IEV2=ICM
190   CONTINUE
      IF(IEV2-IEV1) 220,200,220
200   DO 210 JJ=1,N
      X(IEV1,JJ)=(X(IEV1,JJ)+XC(JJ))/2.0
210    CONTINUE
      I=IEV1
      CALL CHECK(N,M,K,X,G,H,I,KODE,XC,DELTA,K1,NONE)
      DO 211 J=1,N
211   XXX(J)=X(I,J)
      IS(20)=IS(20)+1
      CALL FUNC(XXX,N,Z,IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,W,INT,
     1  NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,
     2  UU,POLD,PNEW,NVAR,MOUTS)
      F(I)=-Z
       GO TO 170
220    CONTINUE
      IITIS=IT/IS(15)*IS(15)
      IF(IITIS.NE.IT) GO TO 228
      IF(IPRINT) 230,228,230
230   PRINT 23,IT
23    FORMAT(   2X,17HITERATION NUMBER ,I5)
      PRINT 24
24    FORMAT(  2X,30HCOORDINATES OF CORRECTED POINT)
      PRINT 19, (IEV1,JC,X(IEV1,JC), JC=1,N)
      PRINT 21
      PRINT 22, (I,F(I),I=1,K)
      PRINT 25
25    FORMAT(  2X,27HCOORDINATES OF THE CENTROID)
      PRINT 26, (JC,XC(JC),JC=1,N)
26    FORMAT(  2(2X,2HX(,I2,6H,C) = ,1PE14.6,4X))
228    IT=IT+1
      IEV2=1
      DO 227 ICM=2,K
      IF(F(IEV2)-F(ICM))229,229,227
229   IEV2=ICM
227   CONTINUE
      CALL TRANS(X,N,NONE,EF,IPRINT,F,XC,IT,IEV2,XXX,ITEST,IS)
      IF(ITEST.EQ.1)RETURN
      IF(IT-ITMAX) 80,80,240
240   CONTINUE
      EF=F(IEV2)
      DO 2411 J=1,N
2411  XXX(J)=X(IEV2,J)
      RETURN
      END
```

Subroutine CENTR Argument List

N	defined in DATA4 variable lists.
M	defined in BOX variable lists.
K	defined in BOX variable lists.
IEV1	contains the number of the complex vertex corresponding to the current maximum value of the objective function.
I	contains a point index to the vetrices of the complex.
XC	defined in BOX variable lists.
X	defined in BOX variable lists.
K1	contains a DO loop limit defined in CONSX.
NONE	defined in BOX variable lists.

```
      SUBROUTINE CENTR (N,M,K,IEV1,I,XC,X,K1,NONE)
      DIMENSION X(NONE,N),XC(N)
      DO 20 J=1,N
      XC(J) = 0.0
      DO 10 IL=1,K1
   10 XC(J) = XC(J) + X(IL,J)
      RK=K1
   20 XC(J) = (XC(J)-X(IEV1,J))/(RK-1.0)
      RETURN
      END
```

Subroutine CHECK Argument List

N	defined in DATA4 variable lists.
M	defined in BOX variable lists.
K	defined in BOX variable lists.
X	defined in BOX variable lists.
G	defined in BOX variable lists.
H	defined in BOX variable lists.
I	defined in BOX variable lists.
KODE	contains a key used to determine if implicit constraints are provided by the user.
XC	defined in BOX variable lists.
DELTA	defined in BOX variable lists.
K1	defined in CENTR variable lists.
NONE	defined in BOX variable lists.

```
      SUBROUTINE CHECK(N,M,K,X,G,H,I,KODE,XC,DELTA,K1,NONE)
      DIMENSION X(NONE,N),G(N),H(N),XC(N)
 10    KT=0
      CALL CONST(N,M,K,X,G,H,I,NONE)
      DO 50 J=1,N
      IF(X(I,J)-G(J)) 20,20,30
 20    X(I,J)= G(J)+DELTA
      GO TO 50
 30   IF(H(J)-X(I,J)) 40,40,50
 40   X(I,J)=H(J)-DELTA
 50   CONTINUE
      IF(KODE)110,110,60
 60   NN=N+1
      DO 100 J=NN,M
      CALL CONST(N,M,K,X,G,H,I,NONE)
      IF(X(I,J)-G(J)) 80,70,70
 70   IF(H(J)-X(I,J))80,100,100
 80   IEV1=I
      KT=1
      CALL CENTR (N,M,K,IEV1,I,XC,X,K1,NONE)
      DO 90 JJ=1,N
      X(I,JJ) = (X(I,JJ) +XC (JJ))/2.0
 90   CONTINUE
 100  CONTINUE
      IF (KT) 110, 110, 10
 110  RETURN
      END
```

Subroutine COMP

During the execution of the optimization sub-component of SYSOPT, accured computer usage parameters are made available to the user through the subroutine COMP. This subroutine utilizes the CDC 6500 library function CPSTAT to obtain running totals of accumulated central processor time, core storage usage, accumulated peripheral processor time, and teletype connect time. These data are then converted, with published pricing factors, to computer processor costs, central money costs, peripheral processor costs, and teletype connect time costs, which are passed to SUBOPT and then output to the user. The operation of the CDC function CPSTAT is detailed in the CDC computer manual reference [7].

Subroutine COMP Argument List

COST	contains the total computer costs that have accumulated at the time COMP is called into execution.
CPCOST	contains the accumulated central processor costs.
PPCOST	contains the accumulated peripheral processor costs.
CMCOST	contains the accumulated central memory costs.
CPTIME	contains the accumulated central processor time.
CTTIME	contains the accumulated connect time.
RG	contains the rate group of program execution. This is a CDC cost scaling factor for computer usage.
COSTMP	= 1.0

```
      SUBROUTINE COMP(COST,CPCOST,PPCOST,CMCOST,CTCOST,CPTIME,PPTIME,
     1  CTTIME,RG,COSTMP)
C
C           TITLE-COMP COMPUTES COMPUTER USAGE COSTS
C
C           PURPOSE-TO COMPUTE CENTRAL PROCESSOR, CENTRAL MEMORY,
C                PERIPHERAL PROCESSOR, AND TELETYPE COSTS AND USAGE
C                PARAMETERS
C
C           FUNCTIONS OR SUBROUTINES CALLED-CPSTAT (A CDC 6500
C                SUBROUTINE), SHIFT, FLOAT
C
      DIMENSION X(16)
C
C           CPSTAT ASSIGNS COMPUTER USAGE PARAMETERS TO THE SIXTEEN
C                STORAGE LOCATIONS IN X.  THE PARAMETER VALUES ARE STORED
C                BIT BY BIT AS THE INTEGER PARTS AND DECIMAL PARTS FOR
C                ACCUMULATED CENTRAL PROCESSOR AND PERIPHERAL PROCESSOR
C                TIMES, AS A REAL VARIABLE FOR CENTRAL MEMORY USAGE, AND
C                AS AN INTEGER FOR TELETYPE CONNECT TIME IN SECONDS.
C
      CALL CPSTAT(X)
C
C           FOR CENTRAL PROCESSOR AND FOR PERIPHERAL PROCESSOR
C                ACCUMULATED TIMES, THE INTEGER AND DECIMAL PORTIONS MUST
C                BE EXTRACTED FROM WORDS 9 AND 10 IN X.
C
      JJ=SHIFT((X(10).A.77777777000B),48)
      KK=SHIFT((X( 9).A.77777777000B),48)
      JJJ=X(10).A.77773
      KKK=X(9).A.77773
      CPTIME=FLOAT(KK)+(FLOAT(KKK)*10E-4)
      PPTIME=FLOAT(JJ)+(FLOAT(JJJ)*10E-4)
C
C           COSTS ARE CALCULATED FROM PUBLISHED PRICING RATES
C
      PPCOST=PPTIME*10./3600.
      CPCOST=CPTIME*160./3600.
      CMCOST=X(12)*64.*.0042/(3600.*1000.)
C
C           TELETYPE CONNECT TIME IS EXTRACTED FROM WORD 7 OF X
      LLL=X(7).A.777777B
      CTTIME=FLOAT(LLL)
      CTTIME=0.0
      CTTIME=CTTIME/3600.
      CTCOST=CTTIME*1.50
C
C           ALL COSTS ARE SCALED BY RECOST-THE RATE GROUP SCALING FACTOR
C
      RGCOST=RG*COSTMP/2.
      PPCOST=PPCOST*RGCOST
      CPCOST=CPCOST*RGCOST
      CMCOST=CMCOST*RGCOST
      CTCOST=CTCOST*RGCOST
C
C           COMPUTE TOTAL ACCUMULATED COSTS
C
      COST=PPCOST+CPCOST+CMCOST+CTCOST
      RETURN
      END
```

Subroutine BOTM

BOTM is a modified version of a commercially available program that executes Powell's unconstrained optimization algorithm [1,6,3]. There are three major modifications that have been made to the program: (1) the program has been changed to properly interact with the remainder of SYSOPT, (2) additional printing and human interaction options have been included in the subroutine, and (3) BOTM has been programmed to allow the input of arbitrary initial search directions. Because of the efficiency of Powell's method in the vicinity of a local optimum and because initial search directions can be computed from the data resulting from the use of the Complex method that speeds convergence of Powell's method, BOTM was incorporated into SYSOPT to complement the Complex method. Thus a two-level search is employed in the component to efficiently locate optima both for from and in the vicinity of local optima.

Within the component, interaction and printing of intermediate results can be specified for after every function evaluation, after every linear search, or at the end of each complete local search. In addition, constraints on the variables (search variables or output variables) can be incorporated through the use of penalty functions added to the objective function and accomplished by proper specification of component interaction variables. Finally convergence parameters of BOTM can be "tuned" during the execution of the program.

For further information concerning BOTM, including logic flow charts, detailed variable descriptions, an example of use, and programming considerations see [3] in the commercial optimization package literature.

Subroutine Argument List

X	defined in FUNC variable lists.
E	contains the iteration convergence limit for the search variables.
N	contains the number of search variables.
EF	contains the local minimum objective function value on return from the subroutine.
ESCALE	contains a scaling factor that determines the maximum step size multiplier in the linear variable searches.
IPRINT	contains a parameter that controls printing. = 1 for printing intermediate results after every single variable search. = 2 for printing intermediate results only after every iteration. = 3 for only printing the final search results. (= IPRINT2 in DATA2).
MAXIT	contains the maximum number of iterations allowed in Powell's method.
U(J,JJ)	contains the JJ^{th} coordinate of the Jth direction for initializing the search vectors in Powell's method. By default U(J,J) = 1.0 and U(J,JJ) = 0. if J ≠ JJ.
W	contains intermediate values of computations internal to the BOTM subroutine.
IAX	= 0 for default values of the initial search directions. ≠ 0 for values of the initial search directions input to the subroutine.
IVAR	defined in VAR variable lists.
XIN	defined in FUNC variable lists.
XOUT	defined in FUNC variable lists.
IX	(= IS array in remainder of component) defined in DATA1 variable lists.

NTS	defined in DATA1 variable lists.
NDPTS	defined in DATA1 variable lists.
NDP	defined in DATA1 variable lists.
NREG	defined in FUNC variable lists.
WW	defined in DATA3 variable lists as W.
INT	defined in FUNC variable lists.
NCONST	defined in DATA4 variable lists.
CONS	defined in DATA4 variable lists.
CONSTL	defined in DATA4 variable lists.
CONSTU	defined in DATA4 variable lists.
PENALTY	defined in DATA3 variable lists.
P	defined in DATA3 variable lists.
NPOBJ	defined in DATA3 variable lists.
TTS	defined in DATA1 variable lists.
TSAVE	defined in FUNC variable lists.
AOLD	defined in FUNC variable lists.
ANEW	defined in FUNC variable lists.
UU	defined in FUNC variable lists.
POLD	defined in FUNC variable lists.
PNEW	defined in FUNC variable lists.
NVAR	defined in DATA1 variable lists.
MOUTS	defined in DATA1 variable lists.
NW	contains the value of N*(N+3) for dimensioning W in BOTM.

```
      SUBROUTINE BOTM(X,E,N,EF,ESCALE,IPRINT,MAXIT,U,W,IAX,IVAR,XIN,XOUT
     1  ,IX,NTS,NDPTS,NDP,NREG,WW,INT,NCONST,CONS,CONSTL,CONSTU,PENALTY,
     2  P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,PNEW,NVAR,MOUTS,NW)
      DIMENSION X(NVAR),E(N),U(N,N),W(NW),IVAR(N),XIN(NVAR),XOUT(MOUTS),
     1  IX(20),NDPTS(NTS),WW(NTS,NDP),CONS(N),CONSTL(N),CONSTU(N),
     2  P(NPOBJ),TTS(NTS,NDP),TSAVE(NTS,NDP),AOLD(NREG),ANEW(NREG),
     3  UU(NREG),POLD(NREG,NREG),PNEW(NREG,NREG)
      INTEGER CONS
      IF(IX(16).EQ.2) IPRINT=1
      PRINT 1
1     FORMAT(1H0,10X,32HPOWELL BOTM OPTIMIZATION ROUTINE    )
      PRINT 2,N,MAXIT,ESCALE,(I,X(I),I=1,N)
2     FORMAT(2X,10HPARAMETERS,/,2X,4HN =  ,I2,4X,8HMAXIT =  ,I4,4X,
     1  9HESCALE = ,F7.2,/,2X,15HINITIAL GUESSES,/,2(2X,2HX(,I2,4H) = ,
     2  1PE16.8)   )
      PRINT 3, (J,E(J),J=1,N)
3     FORMAT(2X,31HACCURACY REQUIRED FOR VARIABLES    ,/,2(2X,2HE(,I2,
     1  4H) = ,E16.8))
      DDMAG=0.1*ESCALE
      SCER=0.05/ESCALE
      JJ=N*(N+1)
      JJJ=JJ+N
      K=N+1
      NFCC=1
      IND=1
      INN=1
      DO 4 I=1,N
      IF(IAX.EQ.0) U(I,I)=1.
      W(I)=ESCALE
      USCALE=0.
      DO 444 J=1,N
      IF(IAX.EQ.0.AND.I.NE.J) U(I,J)=0.
444   USCALE=USCALE+U(I,J)**2
      USCALE=SQRT(USCALE)
      DO 4 J=1,N
      W(K)=U(I,J)*ABS(E(I))/USCALE
4     K=K+1
      ITERC=1
      ISGRAD=2
      CALL FUNC(X,N,F,IVAR,XIN,XOUT,IX,NTS,NDPTS,NDP,NREG,WW,INT,NCONST,
     1  CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,
     2  PNEW,NVAR,MOUTS)
      PRINT 999,(X(IIII),IIII=1,N)
      PRINT 998,F
      FKEEP=2.0*ABS(F)
5     ITONE=1
      FP=F
      SUM=0.
      IXP=JJ
      DO 6 I=1,N
      IXP=IXP+1
6     W(IXP)=X(I)
      IDIRN=N+1
      ILINE=1
7     DMAX=W(ILINE)
      DACC=DMAX*SCER
      DMAG=AMIN1 (DDMAG,0.1*DMAX)
      DMAG=AMAX1 (DMAG,20.*DACC)
      DDMAX=10.*DMAG
      GO TO (70,71),ITONE
70    DL=0.
      D=DMAG
      FPREV=F
      IS=5
      FA=FPREV
      DA=DL
8     DD=D-DL
      DL=D
58    K=IDIRN
      DO 9 I=1,N
      X(I)=X(I)+DD*W(K)
9     K=K+1
      CALL FUNC(X,N,F,IVAR,XIN,XOUT,IX,NTS,NDPTS,NDP,NREG,WW,INT,NCONST,
     1  CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,
     2  PNEW,NVAR,MOUTS)
      IF(IX(16).NE.3) GO TO 997
      PRINT 999,(X(IIII),IIII=1,N)
```

```
      PRINT 998,F
      PRINT 990
      READ 890,ANS
      IF(ANS.EQ.3HYES) RETURN
997   CONTINUE
999   FORMAT(* X=*,/,(4E15.5))
998   FORMAT(* F=*,E15.5)
990   FORMAT(* TYPE YES TO EXIT (A6)*)
890   FORMAT(A6)
      NFCC=NFCC+1
      GO TO (10,11,12,13,14,96),IS
14    IF(F-FA)15,16,24
16    IF(ABS(D)-DMAX) 17,17,18
17    D=D+D
      GO TO 8
18    PRINT 19
19    FORMAT(5X,38HMAXIMUM CHANGE DOES NOT ALTER FUNCTION)
      GO TO 20
15    FB=F
      DB=D
      GO TO 21
24    FB=FA
      DB=DA
      FA=F
      DA=D
21    GO TO (83,23),ISGRAD
23    D=DB+DB-DA
      IS=1
      GO TO 8
83    D=0.5*(DA+DB-(FA-FB)/(DA-DB))
      IS=4
      IF((DA-D)*(D-DB))25,8,8
25    IS=1
      IF(ABS(D-DB)-DDMAX)8,8,26
26    D=DB+SIGN(DDMAX,DB-DA)
      IS=1
      DDMAX=DDMAX+DDMAX
      DDMAG=DDMAG+DDMAG
      IF(DDMAG.GE.1.0E+6.) DDMAG=1.0E+60
      IF(DDMAX-DMAX)8,8,27
27    DDMAX=DMAX
      GO TO 8
13    IF(F-FA)28,23,23
28    FC=FB
      DC=DB
29    FB=F
      DB=D
      GO TO 30
12    IF(F-FB)28,28,31
31    FA=F
      DA=D
      GO TO 30
11    IF(F-FB)32,10,10
32    FA=FB
      DA=DB
      GO TO 29
71     DL=1.
      DDMAX=5.
      FA=FP
       DA=-1.0
      FB=FHOLD
      DB=0.
      D=1.
10    FC=F
      DC=D
30     A=(DB-DC)*(FA-FC)
      B=(DC-DA)*(FB-FC)
      IF((A+B)*(DA-DC))33,33,34
33     FA=FB
      DA=DB
      FB=FC
      DB=DC
      GO TO 26
34    D=0.5*(A*(DB+DC)+B*(DA+DC))/(A+B)
      DI=DB
      FI=FB
      IF(FB-FC)44,44,43
```

```
43      DI=DC
        FI=FC
44      GO TO (86,86,85),ITONE
85      ITONE=2
        GO TO 45
86      IF(ABS(D-DI)-DACC)41,41,93
93      IF(ABS(D-DI)-0.03*ABS(D))41,41,45
45      IF((DA-DC)*(DC-D))47,46,46
46      FA=FB
        DA=DB
        FB=FC
        DB=DC
         GO TO 25
47      IS=2
        IF((DB-D)*(D-DC))48,8,8
48      IS=3
        GO TO 8
41       F=FI
        D=DI-DL
        DD=SQRT((DC-DB)*(DC-DA)*(DA-DB)/(A+B))
        DO 49 I=1,N
        X(I)=X(I)+D*W(IDIRN)
        W(IDIRN)=DD*W(IDIRN)
49      IDIRN=IDIRN+1
        W(ILINE)=W(ILINE)/DD
        ILINE=ILINE+1
        IF(IPRINT-1)51,50,51
50      PRINT 52,ITERC,NFCC,F,(X(I),I=1,N)
52      FORMAT( 10H ITERATION,I5,I10,16H FUNCTION VALUES,10X,3HF =,E15.8/
       +2(E16.8,2X))
        GO TO (51,53),IPRINT
51      IF(IX(16).NE.2) GO TO 510
        PRINT 990
        READ 890,ANS
        IF(ANS.EQ.3HYES) EF=F
        IF(ANS.EQ.3HYES) RETURN
510     CONTINUE
        GO TO (55,38),ITONE
55      IF(FPREV-F-SUM) 94,95,95
95      SUM=FPREV-F
        JIL=ILINE
94      IF(IDIRN-JJ)7,7,84
84      GO TO (92,72),IND
92      FHOLD=F
        IS=6
        IXP=JJ
        DO 59 I=1,N
        IXP=IXP+1
59      W(IXP)=X(I)-W(IXP)
        DD=1.
         GO TO 58
96      GO TO (112,37),IND
112     IF(FP-F)37,37,91
91      D=2.0*(FP+F-2.0*FHOLD)/(FP-F)**2
        IF(D*(FP-FHOLD-SUM)**2-SUM) 87,37,37
87      J=JIL*N+1
        IF(J-JJ)60,60,61
60      DO 62 I=J,JJ
        K=I-N
62      W(K)=W(I)
        DO 97 I=JIL,N
97      W(I-1)=W(I)
61      IDIRN=IDIRN-N
        ITONE=3
        K=IDIRN
        IXP=JJ
        AAA=0.
        DO 67 I=1,N
        IXP=IXP+1
        W(K)=W(IXP)
        IF(AAA-ABS(W(K)/E(I)))66,67,67
66      AAA=ABS(W(K)/E(I))
67      K=K+1
        DDMAG=1.0
        W(N)=ESCALE/AAA
        ILINE=N
        GO TO 7
```

```
37      IXP=JJ
        AAA=0.
        F=FHOLD
        DO 99 I=1,N
        IXP=IXP+1
        X(I)=X(I)-W(IXP)
        IF(AAA*ABS(E(I))-ABS(W(IXP))) 98,99,99
98      AAA=ABS (W(IXP)/E(I))
99      CONTINUE
        GO TO 72
38      AAA=AAA*(1.0+DI)
        GO TO  (72,106),IND
72      IF(IPRINT-2)53,50,50
53       GO TO (109,88),IND
109      IF(AAA-0.1) 20,20,76
76      IF(F-FP)35,78,78
78      PRINT 80
80      FORMAT(5X,31HACCURACY LIMITED BY ERRORS IN F)
        GO TO 20
88       IND=1
35      DDMAG=0.4*SQRT(ABS(FP-F))
        IF(DDMAG.GE.1.0E+60)  DDMAG= 1.0E+60
        ISGRAD=1
108     ITERC=ITERC+1
        IF(IX(16).NE.1) GO TO 810
        PRINT 990
        READ 890,ANS
        IF(ANS.EQ.3HYES) GO TO 81
810     CONTINUE
        IF(ITERC-MAXIT)5,5,81
81      PRINT 82, MAXIT
82      FORMAT(I5,29H ITERATIONS COMPLETED BY BOTM)
        IF(F-FKEEP)20,20,110
110      F=FKEEP
        DO 111 I=1,N
        JJJ=JJJ+1
111     X(I)=W(JJJ)
        GO TO 20
101     JIL=1
        FP=FKEEP
        IF(F-FKEEP)105,78,104
104     JIL=2
        FP=F
        F=FKEEP
105      IXP=JJ
        DO 113 I=1,N
        IXP=IXP+1
        K=IXP+N
        GO TO (114,115),JIL
114     W(IXP)=W(K)
        GO TO 113
115     W(IXP)=X(I)
        X(I)=W(K)
113      CONTINUE
        JIL=2
        GO TO 92
 106    IF(AAA-0.1) 20,20,107
20      EF=F
        RETURN
107       INN=1
        GO TO 35
        END
```

Function NEXT

NEXT is a function in SYSOPT that simplifies the calculation of the amount of core storage needed for component arrays which have variable dimensions. In the program, NEXT is called from MAINPR, MAINOPT, and SUBOPT to update a storage requirement variable, LAST, each time that core locations are needed for an array (of arbitrary size N). While NEXT is not necessary for component operation, this function appreciably simplified programming of the sub-components.

Function Argument List*

LAST	contains an integer pointer to core locations in BLANK COMMON for storing component arrays.
NEXT	contains, on return from the function NEXT, the reference location for storing the array of size N in the array COM of BLANK COMMON.
	contains the size of an array that is to be stored as part of the COM array in BLANK COMMON.

---

*See the section of the variable dimensioning capability for further explanations of these variables.

```
      FUNCTION NEXT(LAST,N)
C
C           TITLE-NEXT CALCULATES THE BLANK COMMON LOCATION OF COMPONENT
C               ARRAYS
C
C           PURPOSE-TO ALLOCATE A FIRST ELEMENT OF BLANK COMMON TO AN
C               INITIAL ELEMENT OF AN ARRAY AND TO RESERVE A SECTION OF
C               BLANK COMMON FOR STORAGE OF THE REMAINDER OF AN ARRAY
C
C           FUNCTIONS OR SUBROUTINES CALLED-NONE
C
C           SET NEXT EQUAL TO THE LAST RESERVED ELEMENT OF THE CORE
C               LOCATION IN BLANK COMMON
C           THE LOCATIONS OF BLANK COMMON THAT ARE RESERVED FOR THE ARRAY
C               CORRESPONDING TO THIS CALL ARE LAST+1, LAST+2, ....,
C               LAST+N.
C
      NEXT=LAST
C
C           SET LAST EQUAL TO THE LAST POSITION (CORE LOCATION REFERENCE)
C               THAT WILL BE FILLED BY THE ARRAY CORRESPONDING TO THIS
C               CALL
C           THE FOLLOWING CALL TO NEXT WILL POINT TO CORE LOCATIONS IN
C               BLANK COMMON LAST+N+1, LAST+N+2, ...
C
      LAST=LAST+N
      RETURN
      END
```

Subroutine TRANS

The determination of when to transfer from the Complex method to Powell's method in the two-level optimization search algorithm in SYSOPT is made with subroutine TRANS. As specified by IS instruction element IS(12) input in the data subroutine hierarchy, TRANS either calls for a human decision of whether to switch or not after a fixed number of Complex iterations, or automatically causes switching from the Complex method to Powell's method after a preset number of Complex iterations. Proper transfer of the coordinates and function value of the vertex with minimum function value, as obtained from the Complex method, into SUBOPT is also performed in TRANS.

More complex, or structured, switching algorithms can easily be programming into TRANS, if desired by the user. Switching from the Complex search algorithm to Powell's search algorithm at the proper time can yield large potential savings of computer usage as the number of objective function evaluations can be reduced significantly by proper switching.

Subroutine Argument List

X	defined in FUNC variable lists.
N	defined in DATA4 variable lists.
NONE	(= N + 1) defined in BOX variable lists.
EF	contains the minimum value of the objective function as obtained from the Complex method.
IPRINT	(= IPRINT1) defined in BOX variable lists.
F	defined in BOX variable lists.
XC	defined in BOX variable lists.
IT	defined in BOX variable lists
IEV2	defined in BOX variable lists.
XXX	contains the search variable coordinate values corresponding to EF on return from TRANS.
ITEST	contains a variable that determines if a transfer back to SUBOPT (and thus to BOTM) should occur. = 0 if execution of the component should continue with Complex iterations. = 1 if execution of the component should transfer to SUBOPT.
IS	defined in DATA1 variable lists.

Additional Significant Component Variables

IS(12)	= 0 for switching interaction between the Complex method and Powell's method after every iteration of the complex method. = +n for switching interaction after n iterations of the Complex method. = -n for automatic switching after n iterations of the Complex method.

```
      SUBROUTINE TRANS(X,N,NONE,EF,IPRINT,F,XC,IT,IEV2,XXX,ITEST,IS)
C
C           TITLE-TRANS TRANSFERS CONTROL OF EXECUTION BACK TO SUBOPT TO
C               CONTINUE THE TWO LEVEL SEARCH WITH POWELL'S METHOD
C
C           PURPOSE-TO EITHER ALLOW INTERACTIVE SWITCHING BETWEEN THE
C               COMPLEX METHOD AND POWELLS METHOD OR TO ALLOW AUTOMATIC
C               SWITCHING BETWEEN THESE ALGORITHMS
C
C           FUNCTIONS OR SUBROUTINES CALLED-NONE
C
      DIMENSION X(NONE,N),XC(N),XXX(1),IS(26),F(NONE)
C
C           IF ITEST = 0, COMPLEX ITERATIONS WILL CONTINUE, IF ITEST = 1,
C               COMPLEX ITERATIONS WILL STOP AND POWELL'S ITERATIONS WILL
C               BEGIN
C           INITIALLY, SET ITEST = 0
C
      ITEST=0
C
C           DETERMINE WHETHER HUMAN INTERACTION (AFTER EVERY ITERATION
C               OR AFTER THE FIXED ITERATION INTERVAL) OR AUTOMATIC
C               SWITCHING IS DESIRED
C
      ITT=IT/IS(12)*IS(12)
      IF(IS(12))3,2,1
C
C           TEST TO DETERMINE IF THIS IS THE PROPER ITERATION FOR HUMAN
C               INTERACTION
C
1     IF(ITT.EQ.IT) GOTO 2
      RETURN
C
C           QUERY USER IF SWITCH IS DESIRED
C
2     PRINT 1000,IT,(X(IEV2,J),J=1,N)
      PRINT 1001,F(IEV2)
      PRINT 1002
      READ 2000,ANS
      IF(ANS.EQ.3HYES) RETURN
C
C           SWITCH TO POWELLS METHOD IS DESIRED
C
      GO TO 4
C
3     CONTINUE
C
C           TEST IF AUTOMATIC SWITCHING ITERATION LIMIT HAS BEEN REACHED,
C               IF SO PERFORM SWITCH
C
      IF(IT.GE.-IS(12)) GO TO 4
      RETURN
4     CONTINUE
C
C           RETURN WITH ITEST = 1, MINIMUM VERTEX COORDINATES= AND
C               MINIMUM VERTEX OBJECTIVE FUNCTION VALUE
C
      ITEST=1
      EF=F(IEV2)
      DO 5 J=1,N
5     XXX(J)=X(IEV2,J)
      RETURN
1000  FORMAT(* ITERATIONS, PRESENT MINIMUM, AND FUNCTION VALUE IS...*/
     1 * IT=*,I4 /* X= */  (7F10.4))
1001  FORMAT(* F=*,F15.5)
1002  FORMAT(* TYPE IN YES TO CONTINUE COMPLEX ITERATIONS (A6)*)
2000  FORMAT(A6)
      END
```

Subroutine SPEED

In parameter estimation and policy decision problems with large complex models, often very sharp ridges and valleys in the objective function can produce difficulties in obtaining proper convergence. The following algorithm, a version of a pattern search superimposed on Powell's method, has been developed to aid in coping with this problem. The basic idea involves finding three points along the sharp valley that straddles the true optimum. A quadratic in the most sensitive variable is then fit to these points. Next, analytic minimization of the quadratic yields the value of the most sensitive variable that is close to its value at the true optimum. A search over the remaining variables, with the most sensitive variable fixed at the value that minimizes the quadratic, results in a point in the full variable space which is then close to the true optimum. Essentially this algorithm produces significant movement along a sharp valley by performing a sequence of searches whose solutions rapidly converge to the valley. The algorithm proceeds as follows:

1. Determine the sensitivities s_{xi} of $J[x_1, x_2, \ldots, x_n]$ at the presumed optimum $x^1_{opt} = (x^1_{1,opt}, x^1_{2,opt}, \ldots, x^1_{n,opt})$

2. Find s_{xm} such that $\left|s_{x_m}\right| \geq \left|s_{x_i}\right|$ for $i \neq m$, $i = 1, 2, \ldots, n$

3. Conduct a linear search on g in y where $g:R \to R$ and

   $$g(y) = \min_{\substack{x_i \\ i \neq m}} J[x_1, x_2, \ldots, x_{m-1}, y, x_{m+1}, \ldots, x_n].$$

   This involves repeated optimization of J in the n-1 variables $x_1, x_2, \ldots, x_{m-1}, x_{m+1}, \ldots, x_n$ with x_m fixed at values near $x^1_{m,opt}$, resulting in y^* such that $g(y^*) \leq g(y)$ for $y \epsilon R$

4. Calculate $x^* = (x^*_{1,opt}, \ldots, x^*_{m-1,opt}, y^*, x^*_{m+1,opt}, \ldots x^*_{n,opt})$ where $g(y^*) = J(x^*)$. This involves an optimization of J in the variables $x_1, x_2, \ldots, x_{m-1}, x_{m+1}, \ldots, x_n$ with $x_n = y^*$

The point x^* is taken to be the local optimum of J along the sharp valley. If in step 3, g(y) does not have a pronounced, or detectable minimum, then this might indicate that J does not have a unique local optimum. In a parameter estimation problem, this would mean there are dependent parameter subsets in the model, or that the data are not adequate to identify the true parameter values. In a policy decision problem, if J does not have a unique optimum, then there could be tradeoffs in the input variables resulting in significant savings in real world costs while retaining the desired system behavior, or the original problem could have not been properly stated.

To complete the discussion of this algorithm, the method of conducting the linear search of g(y) in step 3 will be described.

1. Bracket the minimum of g with three points, y_1, y_2, y_3, such that $g(y_1) \geq g(y_2)$ and $g(y_2) \leq g(y_3)$
2. Calculate the value of y, y*, that minimizes a quadratic passing through the points y_1, $g(y_1)$; i = 1,2,3.

   As can easily be demonstrated:

$$y^* = \frac{(y_2^2-y_2^2)\ g(y_1) + (y_3^2-y_1^2)\ g(y_2) + (y_1^2-y_2^2)\ g(y_3)}{(y_2-y_3)\ g(y_1) + (y_3-y_1)\ g(y_2) + (y_1-y_2)\ g(y_3)}$$

In the component, this procedure of increasing the convergence rate of Powell's method is used only at a presumed global optimum of the objective function. Subroutine SPEED increases the probability of proper and rapid convergence of Powell's method even if the objective function cannot be adequately approximated by a quadratic polynomial.

Subroutine Argument List

X	defined in FUNC variable lists.
N	defined in DATA4 variable lists.
F	contains the objective function value at the presumed global optimum prior to when the speed up algorithm is executed
ERST	contains a variable convergence factor that determines whether SPEED computes an improved approximation to the global optimum.
XG(J)	contains the value of the Jth search variable at the global optimum after SPEED is executed. J = 1,N
FXG	contains the function value of the global optimum of the problem under investigation. FXG corresponds to XG.
ADD	contains a step size parameter for the search variables.
E	defined in BOTM variable lists.
ESCALE	defined in BOTM variable lists.
IPRINT	(= IPRINT2) defined in BOTM variable lists.
MAXIT	defined in BOTM variable lists.
U	defined in BOTM variable lists.
W	defined in BOTM variable lists.
IVAR	defined in VAR variable lists.
XIN	defined in FUNC variable lists.
XOUT	defined in FUNC variable lists.
IS	defined in DATA1 variable lists.
NTS	defined in DATA1 variable lists.
NDPTS	defined in DATA1 variable lists.
NDP	defined in DATA1 variable lists.
NREG	defined in FUNC variable lists.

WW	defined in DATA3 variable lists.
INT	defined in FUNC variable lists.
NCONST	defined in DATA4 variable lists.
CONS	defined in DATA4 variable lists.
CONSTL	defined in DATA4 variable lists.
CONSTU	defined in DATA4 variable lists.
PENALTY	defined in DATA3 variable lists.
P	defined in DATA3 variable lists.
NPOBJ	defined in DATA3 variable lists.
TTS	defined in DATA1 variable lists.
TSAVE	defined in FUNC variable lists.
AOLD	defined in FUNC variable lists.
ANEW	defined in FUNC variable lists.
UU	defined in FUNC variable lists.
POLD	defined in FUNC variable lists.
PNEW	defined in FUNC variable lists.
NVAR	defined in DATA1 variable lists.
MOUTS	defined in DATA1 variable lists.
NW	= N*(N+3), defined in BOTM variable lists.
S	defined in SENS variable lists.
Y	defined in SENS variable lists.
DEV	defined in SENS variable lists.
NSUB	defined in SENS variable lists.
NAMV	defined in VAR variable lists.

```
      SUBROUTINE SPEED(X,N,F,ERST,XG,FXG,ADD,E,ESCALE,IPRINT,MAXIT,U,W,
     1   IVAR,XIN,XOUT,IS,NTS,NDPTS,NDP,NREG,WW,INT,NCONST,CONS,CONSTL,
     2   CONSTU,PENALTY,P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,PNEW,NVAR,
     3   MOUTS,NW,S,Y,DEV,NSUB,NAMV)
C
C           TITLE-SPEED EXECUTES POWELL≠S SPEED UP ALGORITHM
C
C           PURPOSE-TO SPEED UP THE CONVERGENCE PROPERTIES OF POWELL≠S
C                METHOD ALONG SHARP RIDGES AND VALLEYS OF THE OBJECTIVE
C                FUNCTION
C
C           FUNCTIONS OR SUBROUTINES CALLED-SENS,BOTM
C
      DIMENSION X(NVAR),XG(N),E(N),U(N,N),W(NTS,NDP),IVAR(N),XIN(NVAR),
     1   XOUT(MOUTS),IS(20),NDPTS(NTS),WW(NW),CONS(N),CONSTL(N),
     3   CONSTU(N),P(NPOBJ),TTS(NTS,NDP),TSAVE(NTS,NDP),AOLD(NREG),
     4   ANEW(NREG),UU(NREG),POLD(NREG,NREG),PNEW(NREG,NREG),S(NVAR),
     5   Y(NVAR),NSUB(NVAR),NAMV(N)
      INTEGER CONS
C
C           SAVE THE VALUE OF THE OBJECTIVE FUNCTION AT THE PRESUMED
C                GLOBAL OPTIMUM
C
      FSAVE=F
C
C           COMPUTE THE OBJECTIVE FUNCTION SENSITIVITIES TO THE SEARCH
C                VARIABLES
C
      IAX=0
      DO 111 J=1,N
      II=IVAR(J)
      XIN(II)=X(J)
111   NSUB(J)=II
      NSENS=N
      CALL SENS(XIN,NSENS,S,Y,DEV,NSUB,NVAR,IVAR,XIN,XOUT,IS,NTS,NDPTS,
     1  NDP,NREG,W,INT,NCONST,CONS,CONSTL,CONSTU,PENALTY,P,NPOBJ,TTS,
     2   TSAVE,AOLD,ANEW,UU,POLD,PNEW,MOUTS,N)
C
C           FIND THE SEARCH VARIABLE RESULTING IN THE GREATEST
C                SENSITIVITY OF THE OBJECTIVE FUNCTION
C
      IMAX=1
      DO 1 K=2,N
      IF(S(K).LT.S(IMAX)) GO TO 1
      IMAX=K
1     CONTINUE
C
C           REARRANGE THE SEARCH VARIABLES, THROUGH THE X, IVAR, E, AND
C                NAMV ARRAYS SO THAT THE VARIABLE WITH MAXIMUM SENSITIVITY
C                IS INTERCHANGED WITH THE NTH SEARCH VARIABLE
C
      SAVE=X(IMAX)
      X(IMAX)=X(N)
      IIN=IVAR(N)
      XIN(IIN)=X(N)
      X(N)=SAVE
      SAVE=E(IMAX)
      E(IMAX)=E(N)
      E(N)=SAVE
      ISAVE=IVAR(IMAX)
      IVAR(IMAX)=IVAR(N)
      IVAR(N)=ISAVE
      SAVE=NAMV(IMAX)
      NAMV(IMAX)=NAMV(N)
      NAMV(N)=SAVE
C
C           CONDUCT A LINEAR SEARCH IN THE SEARCH VARIABLE WITH GREATEST
C                SENSITIVITY USING THE OBJECTIVE FUNCTION VALUE
C                CORRESPONDING TO THE LOCAL OPTIMUM IN THE REMAINING
C                SEARCH VARIABLES
C
C           COMPUTE THE SECOND POINT ALONG THE RIDGE OR VALLEY (ASSUMING)
C                X AS INPUT TO THE SUBROUTINE ALSO LIES ALONG THE RIDGE OR
C                VALLEY)
C
      DO 3 K=1,N
3     Y(K)=X(K)
```

```
      NN=N-1
      ADD=ADD*X(N)
      X2=X(N)+ADD
      II=0
 8    CONTINUE
C
C          COMPUTE LOCAL OPTIMUM IN REMAINING N-1 VARIABLES USING
C              POWELL'S METHOD
C
      CALL BOTM(X,E,NN,FX2,ESCALE,IPRINT,MAXIT,U,W,IAX,IVAR,XIN,XOUT,
     1   IS,NTS,NDPTS,NDP,NREG,WW,INT,NCONST,CONS,CONSTL,CONSTU,PENALTY,
     1   P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,PNEW,NVAR,MOUTS,NW)
C
C          COMPUTE DIRECTION ALONG RIDGE OR VALLEY WHICH OFFERS MOST
C              IMPROVEMENT IN OBJECTIVE FUNCTION VALUES
C
 9    IF(F-FX2)10,12,12
 10   IF(II-1)14,16,16
 14   II=II+1
      ADD=-ADD
      SAVE=FX2
      X2=X(N)+ADD
      GO TO 8
 16   X1=X2
      FX1=FX2
      X2=X(N)
      FX2=F
      X3=X(N)-ADD
      FX3=SAVE
      GO TO 99
 12   X3=X2+ADD
 15   CONTINUE
      IIN=IVAR(N)
      XIN(IIN)=X3
      X(N)=X3
C
C          COMPUTE LOCAL OPTIMUM IN REMAINING N-1 VARIABLES SEARCHING
C              OPPOSITE INITIAL DIRECTION X(N)+ADD
C
      CALL BOTM(X,E,NN,FX3,ESCALE,IPRINT,MAXIT,U,W,IAX,IVAR,XIN,XOUT,
     1   IS,NTS,NDPTS,NDP,NREG,WW,INT,NCONST,CONS,CONSTL,CONSTU,PENALTY,
     1   P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,PNEW,NVAR,MOUTS,NW)
C
C          DETERMINE IF LINEAR SEARCH SHOULD CONTINUE
C
      IF(FX2-FX3) 17,17,13
 13   X(N)=X2
      XIN(IIN)=X2
      F=FX2
      X2=X3
      FX2=FX3
      ADD=2.*ADD
      X3=X2+ADD
      GO TO 15
 17   X1=X(N)
      FX1=F
 99   CONTINUE
C
C          COMPUTE OPTIMUM OF THE MOST SENSITIVE VARIABLE ALONG THE
C              RIDGE OR VALLEY AND CORRESPONDING OBJECTIVE FUNCTION
C              VALUE USING POWELL'S METHOD. THE OPTIMUM VALUE IS
C              BRACKETED BY X1, X2 AND X3
      XNUM=(X2**2-X3**2)*FX1+(X3**2-X1**2)*FX2+(X1**2-X2**2)*FX3
      XDEN=(X2-X3)*FX1+(X3-X1)*FX2+(X1-X2)*FX3
      XOPT=XNUM/(2.*XDEN)
C
C          TEST TO DETERMINE IF A SIGNIFICANT IMPROVEMENT (OR CHANGE) IN
C              THE PRESUMED GLOBAL OPTIMUM IS FOUND
C
      TEST=ABS(XOPT-X(N))
      IF(TEST.LT.ERST) GO TO 98
      XIN(IIN)=XOPT
      X(N)=XOPT
C
C          COMPUTE LOCAL OPTIMUM IN REMAINING N-1 VARIABLES WITH MOST
C              SENSITIVE SEARCH VARIABLE FIXED AT XOPT
C
```

```
      CALL BOTM(X,E,NN,FXG,ESCALE,IPRINT,MAXIT,U,W,IAX,IVAR,XIN,XOUT,
     1   IS,NTS,NOPTS,NOP,NREG,WW,INT,NCONST,CONS,CONSTL,CONSTU,PENALTY,
     1   P,NPOBJ,TTS,TSAVE,AOLD,ANEW,UU,POLD,PNEW,NVAR,MOUTS,NW)
C
C           ASSIGN GLOBAL OPTIMUM VALUES FROM RESULTING LOCAL OPTIMUM
C               CORRESPONDING TO XOPT
C
      DO 2 K=1,N
2     XG(K)=X(K)
      FXG=FSAVE
      RETURN
98    CONTINUE
C
C           IF NO SIGNIFICANT CHANGES IN THE MOST SENSITIVE SEARCH
C               VARIABLE WERE FOUND AT THE GLOBAL OPTIMUM, USE THE
C               PRESUMED GLOBAL OPTIMUM AS THE TRUE GLOBAL OPTIMUM
C
      DO 4 K=1,N
4     XG(K)=Y(K)
      RETURN
      END
```

Subroutines SEQREG, SEQSW, and SWITCH

To insure compatibility between the Complex method and Powell's method, and to obtain a general idea of the behavior of the objective function, the subroutines SEQREG and SEQSW provide the component with the capability of calculating a local or global polynomial approximation to the objective function. To accomplish this, SEQREG calculates a least squares linear regression estimate of the coefficients of the approximation. This regression is performed sequentially in the subroutine meaning that new estimates are calculated from the old estimates and the most recently generated data set. In this manner it is not necessary to retain previously generated data sets in the computer. Calculation of the sequential estimates is rapid while not placing an excessive burden on core storage.

For a problem with search variables $x_1, x_2, \ldots, x_n$ and a quadratic approximating polynomial, the coefficients to be estimated are

$$\underline{a} = (a_1, a_2, \ldots, a_M) \quad \text{where } M = \frac{n(n-1)}{2} + 2n + 1$$

and the approximating polynomial is given by

$$p(x_1, x_2, \ldots, x_n) =$$

$$\sum_{i=1}^{n} a_i x_i + \sum_{i=2}^{n} \sum_{0 \le j \le i} a_{\frac{(i-1)(i-2)}{2}+j+n} x_i x_j + \sum_{i=n+\frac{n(n-1)}{2}+1}^{2n+\frac{n(n-1)}{2}} a_i x_i + a_{\frac{n(n-1)}{2}+2n+1}$$

The coefficients $(a_1, \ldots\ldots, a_M)$ are estimated sequentially as[7]

$$\underline{a}^r = \underline{a}^{r-1} + \underline{P}_r \, \underline{u}_r \, (\underline{y}_r - \underline{u}_r^T \, \underline{a}^{r-1})$$

where $\underline{y}_k = \underline{a} \, \underline{u}_k + \underline{n}_k$

with $\underline{u}_k = (x_1^2, x_2^2, \ldots, x_n^2, x_2x_1, x_3x_1, x_3x_2, x_4x_1, x_4x_2, \ldots x_nx_{n-1}, x_1x_2 \ldots, x_n, 1)$

$\underline{y}_k$ the value of the objective function with input variable values $x_1, x_2, \ldots, x_n$

and $\underline{n}_k$ the residual error between the polynomial approximation and the objective function values.

In addition $\underline{P}_r$ is calculated sequentially from

$$\underline{P}_r = \underline{P}_{r-1} - \underline{P}_{r-1}\underline{u}_r \left(1+\underline{u}_r^T \underline{P}_{r-1} \cdot \underline{u}_r\right)^{-1} \underline{u}_r^t \underline{P}_{r-1}$$

with $\underline{a}^0 = \underline{0}$ and $\underline{P}_0 = 10{,}000\ \underline{I}$ and $\underline{I}$ = an MxM identity matrix.

The subroutine SEQSW is an interface subroutine between the component and the subroutine SEQREG. SEQSW initializes the parameter estimates $\underline{a}$ and the matrix $\underline{P}$, and also calculates the vectors $\underline{u}$. Following this a call is made to SEQREG which performs the regression computation.

The local quadratic polynomial approximation in the variables $x_1, x_2, \ldots, x_n$ calculated during the execution of the Complex method can be described as an nth degree polynomial $p(x_1,\ldots,x_n)$. Therefore the nxn matrix $\underline{Q}$ where

$$q_{ii} = a_i,\ i = 1,\ldots,n$$

$$\text{and}\quad q_{ij} = \frac{a_{\frac{(i-1)(i-2)}{2}+j+n}}{2} \qquad i,j = 1, 2, \ldots, n,\ i \neq j$$

is the Hessian of the approximating polynomial and therefore an approximation to the Hessian of the objective function.

The construction of a vector set that is mutually $\underline{Q}$ conjugate is performated with a generalized Gram-Schmidt Orthogonolization procedure in subroutine switch.

Let $\underline{e}_i^t = (0, 0, \ldots, 1. 0, \ldots, 0)$, with a 1 only in the ith position and zeros elsewhere, be a set of n, 1xn, linearly independent vectors and $\underline{Q}$ be a known symmetric positive definite matrix.

$$\text{If } \underline{u}_1 = \underline{e}_1,\ \underline{u}_{k+1} = \underline{e}_{k+1} + \sum_{j=1}^{k} a_{kj}\underline{u}_j \quad \text{for } k = 2, 3, \ldots, n-1$$

$$\text{and } a_{kj} = \frac{-\underline{e}_{k+1}^t\ \underline{Qu}_j}{\underline{u}_j^t\ \underline{Qu}_j} \qquad 1 \leq j \leq k = 2, 3, \ldots, n-1$$

then $\underline{u}_\ell^t \underline{Qu}_m = 0$ and $\underline{u}_\ell^t \underline{Qu}_\ell \geq 0$ for $1 \leq \ell$, $m \leq n$ and $\ell \neq m$.

These calculations, performed in the subroutine SWITCH which results in a set of vector directions, can be used to initialize Powell's method.

Subroutine SEQREG Argument List*

U	contains the values of the independent variables of the approximating function.
X	contains the function value corresponding to U.
Q	contains a weighting factor for the data as compared to previous estimates.
AOLD	defined in FUNC variable lists.
ANEW	defined in FUNC variable lists.
POLD	defined in FUNC variable lists.
PNEW	defined in FUNC variable lists.
IDIM	(= NREG) defined in FUNC variable lists.

*See reference [8] for further definition of these variables.

```
      SUBROUTINE SEQREG (U,X,Q,AOLD,ANEW,POLD,PNEW,IDIM)
      DIMENSION U(1),AOLD(1),ANEW(1),POLD(IDIM,1),PNEW(IDIM,1)
      B=SQRT(Q)
      DO 3 I=1,IDIM
3       U(I)=U(I)*B
C  CALCULATE DENOMINATOR
      DEN=1.0
      DO 10 I=1,IDIM
      SUM=0.
      DO 5 J=1,IDIM
5     SUM=SUM+U(J)*POLD(J,I)
10    DEN=DEN+U(I)*SUM
C  CALCULATE PNEW(I,J)
      DO 25 I=1,IDIM
      DO25 J=1,IDIM
      SUM=0.
      DO 15 L=1,IDIM
15    SUM=SUM+POLD(I,L)*U(L)
      TOT=0.
      DO 20 K=1,IDIM
20    TOT=TOT+U(K)*POLD(K,J)
25    PNEW(I,J)=POLD(I,J)-TOT*SUM/DEN
C  CALCULATE A(I)
      DO 30 I=1,IDIM
30    U(I)=U(I)/B
      DO 45 I=1,IDIM
      SUM=0.
      DO 35 J=1,IDIM
35    SUM=SUM+PNEW(I,J)*U(J)
      TOT=X
      DO 40 J=1,IDIM
40    TOT=TOT-U(J)*AOLD(J)
45    ANEW(I)=AOLD(I)+SUM*TOT*Q
      RETURN
      END
```

Subroutine SEQSW Argument List

X	defined in FUNC variable lists.
N	defined in DATA4 variable lists.
F	defined in FUNC variable lists.
ISET	$\neq$ 0 for initializing the arrays AOLD and POLD.
IDIM	contains the number of coefficients in the quadratic approximation to the objective function.
AOLD	defined in FUNC variable lists.
ANEW	defined in FUNC variable lists.
POLD	defined in FUNC variable lists.
PNEW	defined in FUNC variable lists.
U	defined in FUNC variable lists.

```
      SUBROUTINE SEQSW(X,N,F,ISET,IDIM,AOLD,ANEW,POLD,PNEW,U)
C
C           TITLE-SEQSW PERFORMS THE CALCULATIONS NECESSARY TO SWITCH
C               BETWEEN THE COMPONENT AND THE SEQUENTIAL REGRESSION
C               SUBROUTINE SEQREG
C
C           PURPOSE-TO PROVIDE AN INTERFACE BETWEEN THE COMPONENT AND THE
C               SEQREG SUBROUTINE BY CONVERTING A NONLINEAR ESTIMATION
C               PROBLEM INTO A LINEAR PROBLEM
C
C           FUNCTIONS OR SUBROUTINES CALLED-SEQREG
C
      DIMENSION X(1),AOLD(IDIM),ANEW(IDIM),POLD(IDIM,IDIM),
     1  PNEW(IDIM,IDIM),U(IDIM)
C
C           IF ISET IS NOT EQUAL TO 0, INITIALIZE AOLD AND POLD TO BEGIN
C               A LINEAR REGRESSION
C
      IF(ISET.EQ.0) GO TO 10
      DO 1 J=1,IDIM
      AOLD(J)=0.
      DO 2 JJ=1,IDIM
      POLD(J,JJ)=0.
      IF(J.EQ.JJ) POLD(J,J)=10L0E .
2     CONTINUE
1     CONTINUE
      RETURN
10    CONTINUE
C
C           THIS SECTION OF SEQSW PERFORMS AN UPDATE OF THE PREVIOUS
C               LEAST SQUARES LINEAR ESTIMATE WITH A CALL TO SEQREG
C
C           SET DATA WEIGHTING FACTOR
C
      Q=1.0
C
C           INPUT TO THE VECTOR U THE INDIVIDUAL TERMS OF THE POLYNOMIAL
C               FUNCTION IN THE INDEPENDENT SEARCH VARIABLES
C
      I=1
      DO 11 J=1,N
      U(I)=X(J)**2
      I=I+1
11    CONTINUE
      DO 12 J=1,N
      DO 12 JJ=1,N
      IF(JJ.LE.J) GO TO 12
      U(I)=X(J)*X(JJ)
      I=I+1
12    CONTINUE
      DO 13 J=1,N
      U(I)=X(J)
      I=I+1
13    CONTINUE
      U(I)=1.0
C
C           CALL SEQREG SUBROUTINE WITH INPUT VECTOR U AND OBJECTIVE
C               FUNCTION VALUE F - THIS RESULTS IN AN UPDATE OF THE LEAST
C               SQUARES ESTIMATE OF AOLD TO ANEW
C
      CALL SEQREG(U,F,Q,AOLD,ANEW,POLD,PNEW,IDIM)
C
C           TRANSFER ANEW AND PNEW INTO AOLD AND POLD FOR THE NEXT UPDATE
C
      DO 20 J=1,IDIM
      AOLD(J)=ANEW(J)
      DO 21 JJ=1,IDIM
      POLD(J,JJ)=PNEW(J,JJ)
21    CONTINUE
20    CONTINUE
      RETURN
      END
```

Subroutine SWITCH Argument List

A — contains the coefficients of the polynomial (quadratic) approximation to the objective function:

$$p(X) = \sum_{I=1}^{N} A(I)*X(I)$$

$$+ \sum_{I=2}^{N} \sum_{0 \le J \le I} A\left[\frac{(I-1)(I-2)}{2} + J + N\right] X(I)*X(J)$$

$$+ \sum_{I=N+\frac{N*(N-1)}{2}+I}^{2*N+\frac{N*(N-1)}{2}} A(I)*X(I)$$

$$+ A\left[\frac{N*(N-1)}{2} + 2*N+1\right]$$

N — contains the number of independent search variable.

NREG — $\left(= \frac{N*(N-1)}{2} + 2*N+1\right)$ contains the number of coefficients in the polynomial approximation.

U — contains the elements of the matrix representation of p(X) i.e., $p(X) = X'UX + CX + D$.

DUM — contains the results of intermediate calculations.

Q — contains the results of intermediate calculations.

```
      SUBROUTINE SWITCH(A,N,NREG,U,DUM,Q)
C
C           TITLE-SWITCH PERFORMS COMPUTATIONS NECESSARY FOR SWITCHING
C               BETWEEN THE COMPLEX METHOD AND POWELL'S METHOD
C
C           PURPOSE-TO GENERATE Q CONJUGATE SEARCH DIRECTIONS FROM THE
C               QUADRATIC APPROXIMATION TO THE OBJECTIVE FUNCTION
C               OBTAINED DURING THE EXECUTION OF THE COMPLEX METHOD.  THE
C               SEARCH DIRECTIONS CAN BE USED TO INITIALIZE POWELLS
C               METHOD
C
C           FUNCTIONS OR SUBROUTINES CALLED-NONE
      DIMENSION A(NREG),U(N,N),DUM(N),Q(N,N)
C
C           FORM THE Q MATRIX FROM THE COEFFICIENTS OF THE APPROXIMATING
C               POLYNOMIAL
      DO 1 I=1,N
      Q(I,I)=A(I)
1     CONTINUE
      K=N+1
      DO 2 I=1,N
      DO 3 J=1,N
      IJ=I+J
      IF(IJ.GT.N) GO TO 2
      Q(I,IJ)=Q(IJ,I)=A(K)/2.
      K=K+1
3     CONTINUE
2     CONTINUE
C
C           FORM THE Q CONJUGATE SEARCH DIRECTIONS USING THE GENERALIZED
C               GRAM SCHMIDT ORTHOGONALIZATION ALGORITHM.  BEGIN BY
C               INITIALIZING THE COLUMNS OF U TO THE COORDINATE AXIS
C               DIRECTIONS
      DO 4 I=1,N
      DO 41 II=1,N
41    U(I,II)=0.
4     U(I,I)=1.
      NONE=N-1
      DO 5 K=1,NONE
      KPONE=K+1
      DO 6 J=1,K
      DO 7 JJ=1,N
      DUM(JJ)=0.
      DO 7 JJJ=1,N
7     DUM(JJ)=DUM(JJ)+Q(JJ,JJJ)*U(J,JJJ)
      RNUM=-DUM(K+1)
      DEN=0.
      DO 8 JJJ=1,N
8     DEN=DEN+U(J,JJJ)*DUM(JJJ)
      AA=RNUM/DEN
      DO 9 JJJ=1,N
9     U(KPONE,JJJ)=U(K+1,JJJ)+AA*U(J,JJJ)
6     CONTINUE
5     CONTINUE
      RETURN
      END
```

Subroutine Control Flow Charts

The subroutine Control Flow Charts in Figures III-3 and III-4 are pictorial representations of the logic flow of execution in SYSOPT. The pattern recognition sub-component, Figure III-3, and the optimization sub-component, Figure III-4, together illustrate the hierarchical structure of the component. With this listing of all the subroutine references in their usual sequential order of execution, the programmer can obtain an understanding of the logic and structure of the program.

These Control Flow Charts can be used in several applications: (1) the charts can acquaint a programmer, unfamiliar with SYSOPT operation, with the execution flow of the program, (2) the charts can aid in debugging programming changes to the component, and (3) the charts can assist in the solution of parameter estimation or optimal control problems by illustrating possible component execution paths. Both the user and programmer can more efficiently use the component by referring to these pictorial representations of the component's execution paths.

Figure III-3

SUBPROGRAM CONTROL FLOW CHART

Pattern Recognition Sub-Component

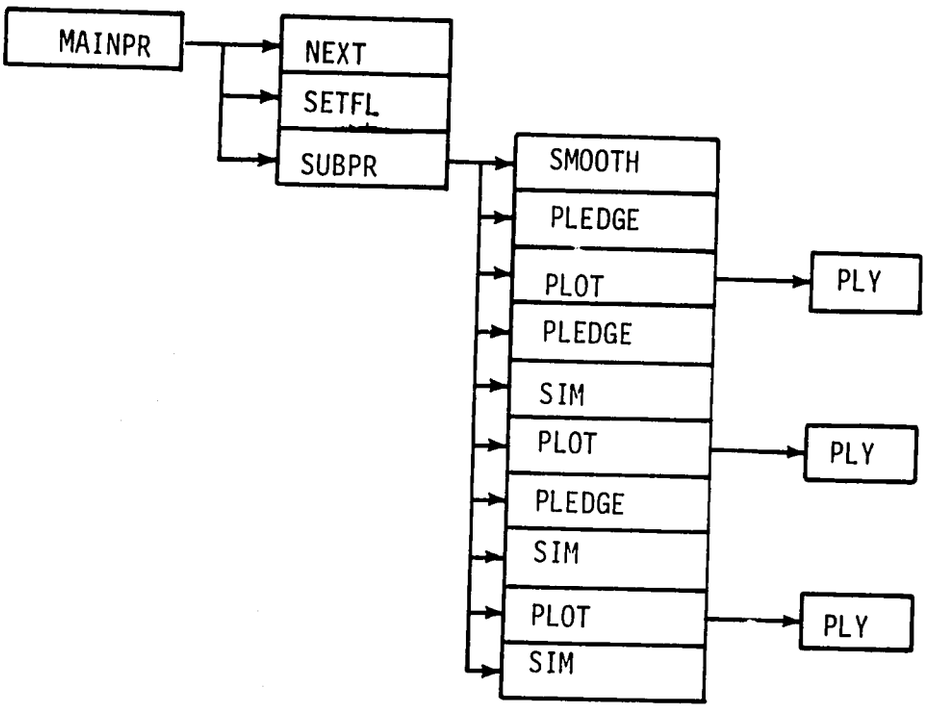


Figure III-4

SUBPROGRAM CONTROL FLOW CHART

Optimization Sub-Component

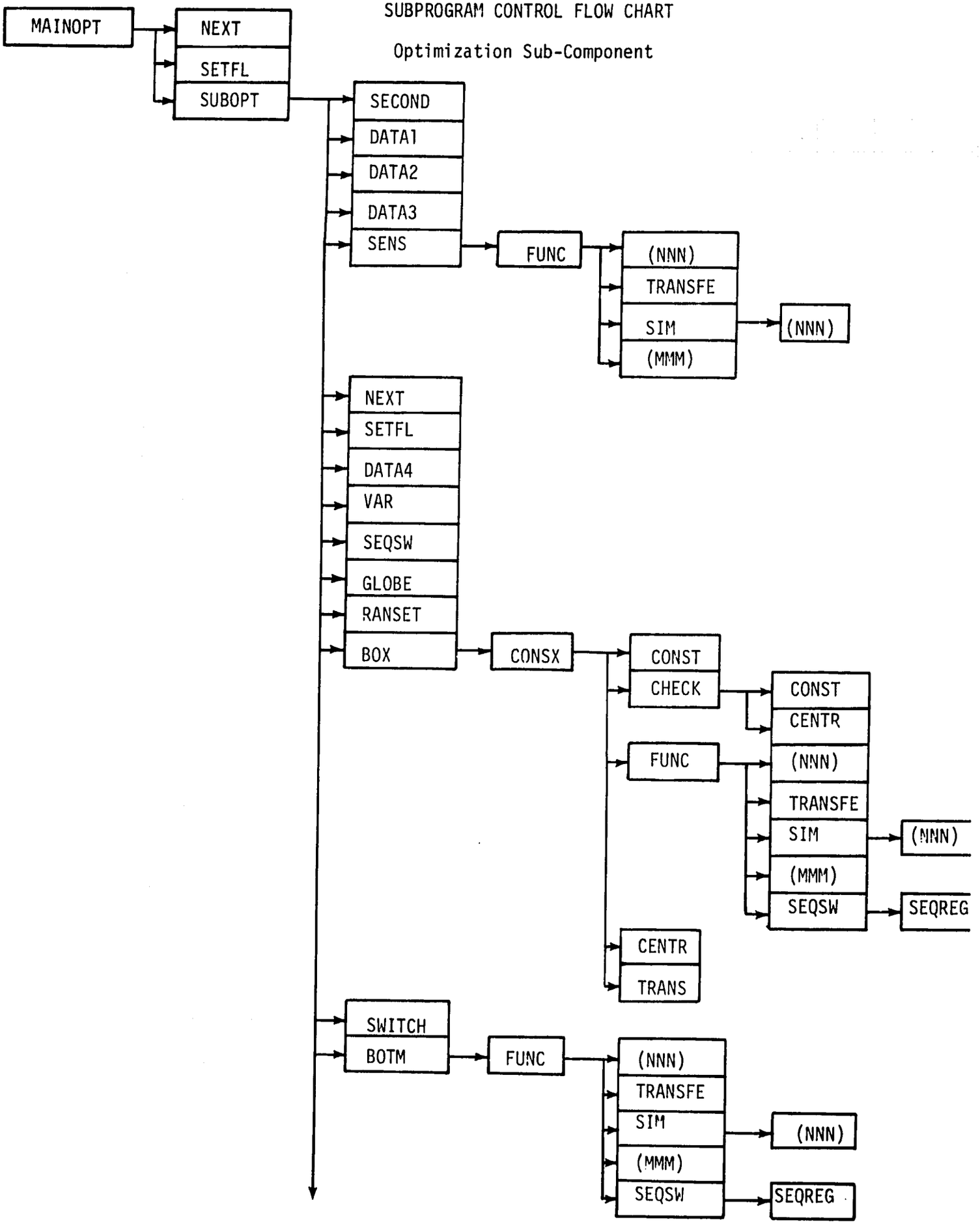
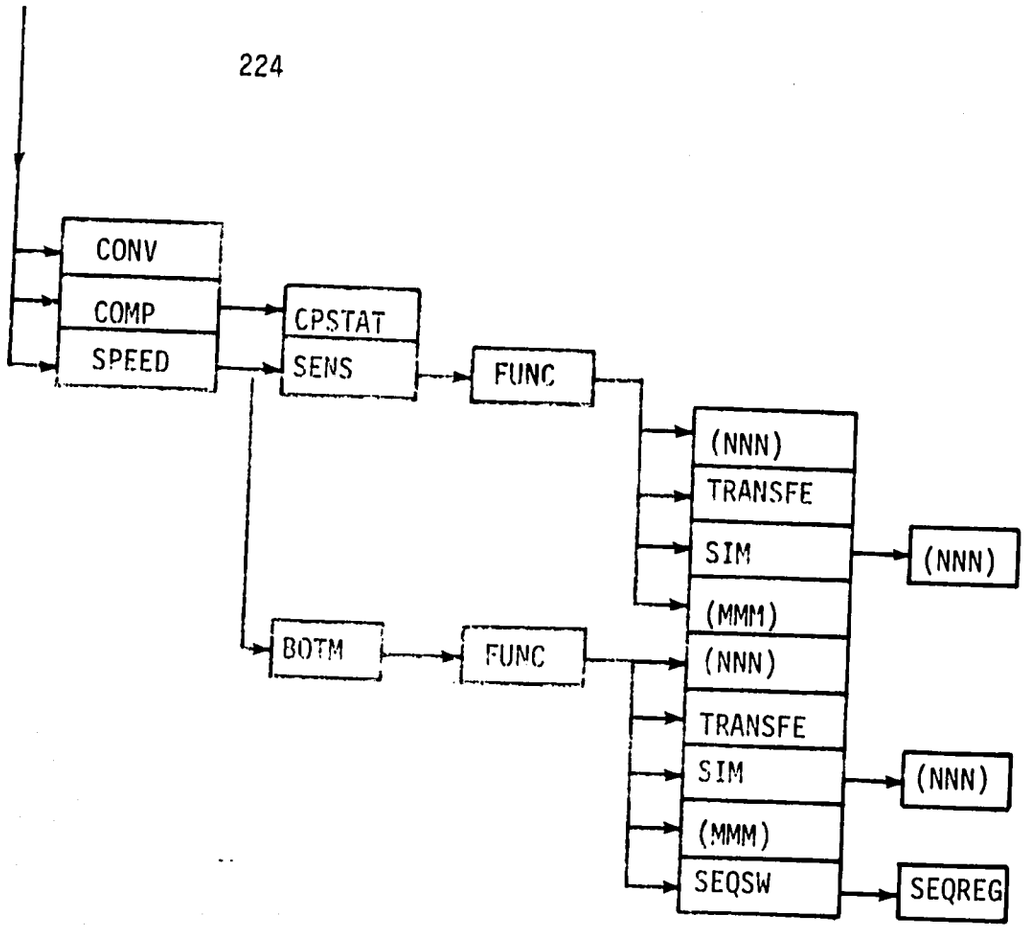


Figure III-4 (cont'd)



SYSOPT Use of Input-Output Tapes

There are three computer tapes (CDC local or permanent files) that SYSOPT employs to simplify the input, output, and transfer of data within the component. Any or all of these tapes can be created and stored in the computer for later input of real world time series to the pattern recognition sub-component, output of large printouts and/or plots to a convenient high speed printer, and transfer of data from the pattern recognition sub-component to the optimization sub-component. Although all the information on the tapes must be in specific formats to correctly interact with the component, those tapes that are generated internal to the program can be directly used. On the other hand, the user also has the option of writing these tapes independently of the component with the programs described in this section.

One tape, TAPE97, is used to simplify input of the real world time series into the pattern recognition sub-component. Because time series generally involve large amounts of data, the use of a computer batch operation to generate the tape from a card deck greatly reduces the difficulty associated with the input of time series in an interactive program execution mode. The format for writing the tape is defined by the FORTRAN statements:

```
      DO 1  J = 1,NDP
1     WRITE(97) (TTS(I,J), I = 1,NTS)
```

where NDP is the number of data points in the time series (assumed all to have the same number of points), NTS is the number of variables represented by the time series, and TTS(I,J) is the (I,J) element of the array TTS and contains the Jth point of the Ith time series.

For an example of the generation of TAPE97, consider an economic parameter estimation problem involving the approximation of the price elasticities with respect to demand and supply for a single product given the values of P, the price of the product; SUP, the supply rate of the product; and DEM, the demand rate for the product. These variables are known for a 12 month period as given in Table III-1.

Table III-1. Time Series Values for Example Problem

Time Units (Months)	P($)	SUP (units/mo.)	DEM (units/mo.)
1	10.0	90.	100.
2	10.5	97.	95.
3	10.4	96.	96.
4	10.6	98.	92.
5	10.2	100.	90.
6	9.8	96.	94.
7	9.7	97.	97.
8	9.9	97.	97.
9	10.2	96.	98.
10	10.4	96.	99.
11	10.6	98.	97.
12	10.4	97.	97.

Therefore NDP = 12, NTS = 3, and TTS(I,J) is the value at the intersection of the Ith column and J^{th} row of the table. A FORTRAN program together with the necessary data cards and CDC 6500 control statements that will properly generate a local file, TAPE97 and catalog it as a permanent file, TIMESERIES, contains the following statements:

```
PNC (user specific)
JOB card (user specific)
FTN.
LGO.
CATALOG, TAPE97, TIMESERIES, ID = xxx, RP = 999.
7
8
9
```

```
      PROGRAM T97 (INPUT,TAPE97)
      DIMENSION TTS(3,12)
      READ 888, NDP
      READ 887, NTS
      DO 2  J = 1,NDP
 2    READ 886, (TTS(I,J), I = 1,NTS)
      DO 1  J = 1,NDP
 1    WRITE(97) (TTS(I,J), I = 1,NTS)
 888  FORMAT(I3)
 887  FORMAT(I3)
 886  FORMAT(3F10.2)
      END
7
8
9
012
ØØ3
10.0      90.       100.
10.5      97.        95.
10.4      96.        96.
10.6      98.        92.
10.2     100.        90.
 9.8      96.        94.
 9.7      97.        97.
 9.9      97.        97.
10.2      96.        98.
10.4      96.        99.
10.6      98.        97.
10.4      97.        97.
6
7
8
9
```

For any parameter estimation problem, program T97 can be used with the given CDC control cards to generate tapes to properly interact with SYSOPT when an appropriate DIMENSION statement and data cards are substituted in the example card deck. As with the other tapes described here, this tape must be attached as a local file (in this case TAPE97) to interact with the component. The CDC control statement to accomplish this on the teletype, or in batch computer operation is:

ATTACH, TAPE97, TIMESERIES.

A second tape is used in the component to store large printout and/or plots of real world or smoothed real world time series, and model outputs generated from the pattern recognition sub-component. This tape, TAPE6, can be transferred to a suitable high speed printer with a DISPOSE command from a teletype terminal

DISPOSE, TAPE6,PR.

With this capability, the teletype is not burdened with the output of large print/plots. In addition, because this tape is generated internally to SYSOPT, there is no need for the tape to be structured by the user.

A third tape, TAPE98, simplifies the input/output functions in the component by transferring information from the pattern recognition sub-component to the optimization sub-component. If desired, this tape can be externally generated without executing the pattern recognition sub-component to reduce the quantity of interactive data input into the optimization sub-component. The number and names of the input variables, the value of a switch indicating whether a parameter estimation or optimal control problem is being investigated and time series sets (real world and smoothed) are all contained on this tape. The following FORTRAN statements define the format of this tape:

```
      WRITE(98) IS(1)
      WRITE(98) NVAR
      WRITE(98) (NAME(J), J = 1,NVAR)
      WRITE(98) MOUTS
      WRITE(98) (ONAMES(J), J = 1,MOUTS)
      IF(IS(1).NE.0) GO TO 9
      WRITE(98) (NDPTS(J), J = 1,NTS)
      WRITE(98) KEY
      DO 1  I = 1,NTS
      JJ = NDPTS(I)
1     WRITE(98) (TTS(I,J), J = 1,JJ)
      IF(KEY.EQ.0) GO TO 9
      DO 2  M = 1,2
```

```
      DO 2  I = 1,NTS
      JJ = NDPTS(I)
 2    WRITE(98) (TS(M,I,J), J = 1,JJ)
 9    CONTINUE
```

where IS(1) = 0 if a parameter estimation problem is being studied and IS(1) = 1 if an optimal control problem is being studied,

NVAR	is the number of input variables in the simulation model; XIN(1), ..., XIN(NVAR)
NAME	is an array containing the names of the input variables
MOUTS	is the number of output variables in the simulation; XOUT(1), ..., XOUT(MOUTS)
ONAMES	is an array containing the names of the output variables
NTS	is the number of time series
NDPTS	is an array containing the number of points in the time series
KEY	= 0 if there was no smoothing of the real world time series
	≠ 0 if there was any smoothing of real world time series
TTS	is an array containing the real world time series where TTS(I,J) contains the Jth point in the Ith time series
TS	is an array containing the smoothed time series calculated in the pattern recognition sub-component from TTS. T(1,I,J) contains the Jth point in the Ith smoothed time series. Note that only TS(1,·,·) is used in the component and if TAPE 98 is externally generated KEY = 0 and TS is not used.

Continuing the previous example of structuring tapes for a problem involving the price elasticities with respect to supply and demand, denoted ELASP and ELASD, respectively, then the following card deck of CDC control statements, FORTRAN program T98, and data card set will generate TAPE98 in the proper format and catalog it as OPTINPUT.

```
PNC (user specific)
JOB card (user specific
FTN.
LGO.
CATALOG, TAPE98, OPTINPUT, ID = xxxx, RP = 999.
7
8
9
      PROGRAM T98(INPUT, TAPE98)
      DIMENSION NAME(2), ONAMES(3), NDPTS(12), TTS(3,12), TS(1,1,1), IS(1)
      READ 888, IS(1)
      READ 887, NVAR
      READ 886, (NAME(J), J = 1,NVAR)
      READ 885, MOUTS, NTS
      READ 884 (ONAMES(J), J = 1,MOUTS)
      IF(IS(1).NE.0) GO TO 19
      READ 883, (NDPTS(J), J = 1,NTS)
      KEY = 0
      DO 2  I = 1,NTS
      JJ = NDPTS(I)
2     READ 882, (TTS(I,J), J = 1,JJ)
19    CONTINUE
      WRITE(98) IS(1)
      WRITE(98) NVAR
      WRITE(98) (NAME(J), J = 1,NVAR)
      WRITE(98) MOUTS
      WRITE(98) (ONAMES(J), J = 1,MOUTS)
      IF(IS(1).NE.0) GO TO 9
      WRITE(98) (NDPTS(J), J = 1,NTS)
      WRITE(98) KEY
      DO 1  I = 1,NTS
      JJ = NDPTS(I)
1     WRITE(98) (TTS(I,J), J = 1,JJ)
9     CONTINUE
888   FORMAT(I1)
887   FORMAT (I3)
886   FORMAT (10A6)
885   FORMAT (2I3)
884   FORMAT (10A6)
883   FORMAT (20I4)
882   FORMAT (8F10.2)
      END
7
8
9
0
002
ELASP ELASD
003   003
P     SUP   DEM
001200120012
```

10.0	10.5	10.4	10.6	10.2	9.8	9.7	9.9
10.2	10.4	10.6	10.4				
90.0	97.	96.	98.	100.	96.	97.	97.
96.	96.	98.	97.				
100.	95.	96.	92.	90.	94.	97.	97.
98.	99.	97.	97.				

6
7
8
9

For either a parameter estimation problem or an optimal control problem, this program and CDC control cards can be used to generate tapes to properly interact with the optimization sub-component of SYSOPT. Only the DIMENSION statement in the program and the data cards need be changed to suit the particular problem under investigation. To place the permanent file OPTINPUT for SYSOPT execution, the following CDC control statement is used.

ATTACH, TAPE98, OPTINPUT.

In summary, the use of tapes can simplify the input, output, and transfer of data in the component. Thus, efficient interaction with SYSOPT is achieved by eliminating many slow man/machine data transfers by either using the component to generate interactive tapes or by generating interactive tapes with the programs presented in this section.

Alphabetical Glossary of Component Variables

A
ADD
ALPHA
ANEW
AOLD
BEGGOTM
BEGLOTM
BEGTIME
BETA
BOUNDSL
BOUNDSU
CMCOST
COM
CONS
CONSTL
CONSTU
COST
CPCOST
CPTIME
CTTIME
DELTA
DEV
DT
DUM
DUR
E
EF
ENDGOTM
ENDLOTM
ERST
ESCALE
ETS
F
FMIN
FXG
G
GAMMA
H
I
IAX
IC
IDIM
IEV1
IEV2
II
INDEX
INDVAR
INT
IPLOT
IPRINT1
IPRINT2
IS(1),IS(2),..
ISET
ISWITCH
IT
ITEST
ITIP
ITMAX
ITT
IVAR
K
KEY
KNTLOW
KODE
K1
L(1),L(2),...,
LA,LB,LC,...,
LAST
LU
LUNIT
M
MAXIT
MOUTS
N
NAME
NAMV
NCONST
NDP
NDPTS
NEXT
NGLOBE
NIND
NO
NPOBJ
NREG
NSENS
NSET
NSUB
NTD
NTIME
NTS
NTSDES
NTSMO
NTW
NTYPE
NVAR
NW
ONAMES
OUT
P
PENALTY
PNAMES
PNEW
POLD
PPCOST
Q
RG
S
SALPHA
STS
TIP
TS
TSAVE
TTS
U
UU
VECTOR
W
X
XC
XG
XIN
XOUT
XVAR
XXX
XZ
Y
Z

Alphabetical Glossary of Component Variables

A — contains the coefficients of the polynomial (quadratic) approximation to the objective function:

$$p(X) = \sum_{I=1}^{N} A(I)*X(I)$$

$$+ \sum_{I=2}^{N} \sum_{0 \le J \le I} A\left[\frac{(I-1)(I-2)}{2}+J+N\right] X(I)*X(J)$$

$$+ \sum_{I=N+\frac{N*(N-1)}{2}+1}^{2*N+\frac{N*(N-1)}{2}} A(I)*X(I)$$

$$+ A\left[\frac{N*(N-1)}{2}+2*N+1\right]$$

ADD — contains the basic interval for stepping along the most sensitive variable values along a sharp valley in the input variable space

ALPHA — contains a reflection factor for the generation of new complexes in the search

ANEW(J) — contains the new (updated) J^{th} coefficient of the quadratic approximation to the objective function J = 1,NREG

AOLD(J) — contains the previous (from last update) J^{th} coefficient of the quadratic approximation to the objective function J = 1,NREG

BEGGOTM — contains the CPU time when the current global search was initiated

BEGLOTM — contains the CPU time when the current local search was initiated

BEGTIME — contains the beginning CPU execution time of the SYSOPT program

BETA — contains a variable convergence parameter

BOUNDSL(J)	contains the lower boundary value for the Jth search variable in the complex method J = 1,N
BOUNDSU(J)	contains the upper boundary value for the Jth search variable in the complex method J = 1,N
CMCOST	contains the accumulated central memory costs
COM	is an array in blank common containing all the program's arrays whose sizes depend on the values of the input variables
CONS(J)	contains the number of the variable in the search variable set to which the Jth constraint corresponds J = 1,N
CONSTL(J)	contains the lower constraint value of the X(CONS(J)) variable J = 1,NCONST (NCONST defined later)
CONSTU(J)	contains the upper constraint value of the X(CONS(J)) variable J = 1,NCONST
COST	contains the total computer costs that have accumulated at the time COMP is called into execution
CPCOST	contains the accumulated central processor costs
CPTIME	contains the accumulated central processor time
CTTIME	contains the accumulated connect time
DELTA	contains the explicit constraint violation correction in the complex method
DEV	contains a parameter controlling the percent change in the independent variables for the sensitivity testing
DT	contains the basic time step size of the simulation model SIM
DUM	contains the results of intermediate calculations
DUR	contains the length of the time horizon of the model SIM
E	contains the iteration convergence limit for the search variables
E(J)	contains the successive iteration convergence limit for Powell's method J = 1,N

EF	contains the minimum value of objective function when convergence is obtained
ENDGOTM	contains the CPU time of the end of the current global search
ENDLOTM	contains the CPU time of the end of the current local search
ERST	contains a parameter that determines whether the speed-up procedure of Powell's method will be used to attempt to improve the global optimum approximation
ESCALE	contains a scaling factor that determines the maximum step size multiplier in the linear variable searches
ETS(J)	contains the Jth point of a smoothed real world time series obtained by smoothing STS in SMOOTH (temporary storage)
F	contains the objective function value at the presumed global optimum prior to when the speed up algorithm is executed
FMIN	contains the current global minimum of the problem under investigation
FXG	contains the function value of the global optimum of the problem under investigation (at the end of a global search FXG = FMIN)
G	contains lower bounds for the search variables (= BOUNDSL)
GAMMA	contains a complex convergence parameter for variables along constraint boundaries
H	contains upper bounds for the search variables (= BOUNDSU)
I	contains a point index to the vetrices of the complex
IAX	= 0 for default values of the initial search directions
	≠ 0 for values of the initial search directions input to the subroutine
IC	contains the number of implicit variables (= 0 during SYSOPT execution)
IDIM	contains the length of the real world time series TS
IEV1	contains the number of the complex vertex corresponding to the current maximum value of the objective function
IEV2	contains the number of the complex vertex corresponding to the current minimum value of the objective function

II — contains the integer expression of the real variable, passed to I, on return from the subroutine

INDEX — defined as INDVAR in SUBPR variable lists

INDVAR — contains the subscripts of output variables that are to be printed/plotted with subroutine PLOT. In particular

INDVAR(1) = m_1

INDVAR(2) = $-m_1$

INDVAR(3) = m_2

INDVAR(4) = $-m_2$

. .

. .

. .

INDVAR(2J-1) = m_J

INDVAR(2J) = $-m_J$

INDVAR(2J+1) = 0

causes the variables corresponding to the $m_1, m_2, \ldots, m_J$ subscripts to be printed and/or plotted. Dimensioned to 4*NVAR+1

INT — contains the number of DT time steps corresponding to the real world data sampling interval in a parameter estimation problem or the desired sampling interval in a behavioral study of the model

IPLOT — = 0 for normal objective function evaluation (see optimization sub-component subroutine FUNC)

≠ 0 for writing model outputs on a computer tape for plotting

IPRINT1 — is a parameter controlling the printing of intermediate iteration results in the complex method

= 1 for intermediate values to print on each iteration

= 0 for suppression of printing until the final solution is obtained

IPRINT2 — contains a parameter controlling printing in Powell's method

= 2 for printing of intermediate values after every linear search

= 1 for printing of intermediate values after every iteration

= 0 suppresses printing until the final solution is obtained

IS	is an integer array containing component control variables which affect the path of optimization algorithm execution
IS(1)	0 for a parameter estimation problem 1 for a optimal control problem
IS(2)	0 for print coefficients of approximating polynomial to the objective function 1 not to print coefficients
IS(3)	maximum number of local searches to be conducted for the global search
IS(5)	0 for no speed up of Powell's methods along a sharp ridge in the objective function 1 for execution of the speed up algorithm
IS(7)	0 if sensitivity testing of the objective function in the entire input variable set 1 if only sensitivity testing on subsets of the input variable set is desired
IS(8)	0 for no interacction (selecting initial points for local searches) in global searches 1 for optional interaction in global searches
IS(9)	0 for no polynomial approximation to the objective function to be calculated 1 for a local polynomial approximation 2 for a global polynomial approximation
IS(10)	0 for regular switching between the Complex and Powell's methods 1 for switching will construction of approximate conjugate initial search directions for Powell's method
IS(12)	0 for switching interaction between the Complex and Powell's methods after every iteration of the Complex method +n for switching interaction after n iterations of the Complex method -n for automatic switching after n iterations of the Complex method
IS(15)	+n for an output interval of n iterations in the Complex method
IS(16)	0 for interaction in Powell's method 1 for interaction (to exit) after every iteration 2 for interaction after every linear search 3 for interaction after every function evaluation

IS(18)	0 for a least squares error criterion in a parameter estimation problem 1 for a normalized least squares error criterion in a parameter estimation problem
ISET	= 0 to allow updating of quadratic approximation coefficients = 1 to initialize the arrays AOLD, POLD for sequential regression calculations
ISWITCH	= 1 if the simulation model is to be used for objective function evaluation or plotting ≠ for normal execution of the simulation model (print interaction, etc.)
IT	contains a variable to count Complex iterations
ITEST	contains a variable that determines if a transfer back to SUBOPT (and thus to BOTM) should occur = 0 if execution of the component should continue with Complex iterations = 1 if execution of the component should transfer to SUBOPT
ITIP	= 1 if an initial value of the time series TS is known with confidence ≠ 1 if no initial value of the time series TS can be relied upon as accurate
ITMAX	contains the maximum number of iterations allowed in the Complex method
ITT	contains a line printer parameter that causes blank lines to be printed so as to match time scales between a real world data study and a behavioral study of the model SIM. ITT = n causes the printer to skip n lines between plot points when plotting the real world or smoothed real world time series
IVAR(J)	contains the number of the input variable to which the Jth search variable corresponds J = 1,N
K	contains the number of vertices in the Complex
KEY	= 0 if there was no smoothing of the real world time series ≠ 0 if there was any smoothing of real world time series

KNTLOW — contains the number of times a single vertex repeats in having the minimum value of the objective function

KODE — contains a key used to determine if implicit constraints are provided by the user

K1 — contains a DO loop limit defined in CONSX

L(J) — contains the reference location in the COM array (see the section on the Variable Dimensioning Capability) for storing the elements of the Jth component array in the following list:

J	Component Array Name
1	NAME
2	XVAR
3	X
4	S
5	Y
6	NSUB
7	TTS
8	TSAVE
9	NDPTS
10	IS
11	ONAMES
12	XOUT
13	P
14	W
15	XIN

LA — contains a pointer (reference location in the COM array) t the NAME array

LB — contains a pointer to the XIN array

LC — contains a pointer to the XOUT array

LD — contains a pointer to the ONAMES array

LE — contains a pointer to the PNAMES array

LF — contains a pointer to the NTSMO array

LG — contains a pointer to the TTS array

LH — contains a pointer to the TSAVE array

LI — contains a pointer to the STS array

LJ — contains a pointer to the ETS array

LK — contains a pointer to the INDVAR array

LL — contains a pointer to the TS array

LM — contains a pointer to the IS array

LN — contains a pointer to the TIP array

LO — contains a pointer to the NDPTS array

LAST	contains an integer pointer to core locations in BLANK COMMON for storing component arrays
LU	contains the input/output unit number on which the following information is to be written 1. Size of variable set to be plotted 2. Output variable names to be plotted
LUNIT	contains the unit input/output number on which the output names, and output data values were printed (In SYSOPT LUNIt = 99)
M	contains the desired degree of the polynomial filter, M = 1 or 2
MAXIT	contains the maximum number of iterations allowed in Powell's method
MOUTS	is the number of output variables in the simulation; XOUT(1), ..., XOUT(MOUTS)
N	contains the number of variables in the input variable search set
NAME(J)	contains the name of the Jth input parameter or control variable (maximum of six characters) J = 1,NVAR
NAMES(J)	contains the Jth name of the output variables J = 1,N
NAMV(J)	contains the name of the Jth search variable J = 1,N (where N is the number of variables in the current search variable set under investigation)
NCONST	contains the number of variables that are constrained to lie within specified limits in Powell's method
NDP	contains the maximum number of data points in the NTS time series. In an optimal control problem set NDP = 1.
NDPTS(J)	contains the number of data points in the Jth time series for a parameter estimation problem J = 1,NTS
NEXT	contains, on return from the function NEXT, the reference location for storing the array of size N in the array COM of BLANK COMMON
NGLOBE	contains the number of global searches that have been completed by the component

NIND	contains 4*NVAR+1
NO	not used
NPOBJ	contains the number of parameters specifying the form of the objective function in an optimal control problem
NREG	= 2*N+1+N*(N-1)/2 is the number of coefficients in a quadratic approximation of a function in N independent variables
NSENS	contains the number of input variables for which sensitivity testing is to be performed
NSET	= 0 to initialize the integral component of the optimal control objective function = 1 after initialization
NSUB(J)	contains the number of the input variable for which sensitivity testing is desired. J = 1,NSENS. If NSUB(1) = 0, then the sensitivities of all the input variables will be calculated
NTD	contains the maximum number of data points in all the NTS time series (primarily used for the variable dimensioning capability)
NTIME	contains the number of data points for printing/plotting from PLOT
NTS	contains the number of variables represented by the time series TTS in a parameter estimation problem. In an optimal control problem, set NTS = MOUTS
NTSDES	= 0 if the original real world time series are to be used for least squares error evaluation = 1 if the smoothed time series from the linear filter is desired = 2 if the smoothed time series from the quadratic filter is desired
NTSMO(J)	contains the number of the model output corresponding to the Jth time series. This array is used for selecting subsets of model outputs and time series for printing/plotting J = 1,MOUTS

NTW	contains MOUTS+NTS
NTYPE	= 1 for range of scales of plots set by actual MAXIMUM and MINIMUM values of the output variables = 3 for printing of time series and for calculated scales of the plots from maximum and minimum values of the output variables = 4 for printing only of the output variables
NVAR	contains the number of input, parameter or control, variables XIN(1), XIN(2), ..., XIN(NVAR) in the simulation model being studied (subroutine SIM)
NW	contains the value of N*(N+3) for dimensioning W in BOTM
ONAMES	is an array containing the names of the output variables
OUT(J)	contains the valued of the m_Jth output variable = VECTOR(m_J) on return from PLY
P(J)	contains the parameter value for the Jth parameter of the objective function in an optimal control problem J = 1,NPOBJ
PENALTY	contains a parameter that is added to the objective function each time a constraint is violated by a search variable value
PNAMES(J)	contains the name of the Jth variable that is to be printed/plotted with the PLOT subroutine. J = 1,NTS+MOUTS Variable names A, B, C, D, ... correspond to the smoothed time series corresponding to XIN(1), XIN(2), XIN(3), XIN(4), ...
PNEW, POLD	are arrays containing information necessary for the sequential update of the polynomial approximation to the objective function, both are dimensioned to (NREG,NRFG)
PPCOST	contains the accumulated peripheral processor costs
Q	contains a weighting factor for the data as compared to previous estimates (subroutine smooth)
RG	contains the rate group of program execution. This is a CDC cost scaling factor for computer usage
S(J)	contains the sensitivity of the objective function with respect to the NSUB(J)th input variable J = 1,NSENS
SALPHA	contains a scaling factor ($\geq$) that determines the size of the initial Complex in the Complex method with vertices randomly generated around an initial point

STS	contains the Jth point of a real world time series that is to be smoothed by the polynomial filter subroutine SMOOTH (temporary storage)
TIP	if ITIP = 1, then TIP contains the initial value of the time series TS if ITIP ≠ 1, then TIP is not used
TS(1,J,JJ)	contains the JJ^{th} element of the Jth time series (smoothed real world or actual real world) that is to be input into the optimization sub-component for solving a parameter estimation problem J = 1,NTS JJ = 1,NDPTS(J)
TSAVE(J,JJ)	contains the model output corresponding to the JJ^{th} data point of the Jth time series, J = 1,NTS and JJ = 1,NDPTS(J)
TTS	is an array containing the real world time series where TTS(I,J) contains the Jth point in the Ith time series
U(J,JJ)	contains the JJ^{th} coordinate of the Jth direction for initializing the search vectors in Powell's method. By default U(J,J) = 1.0 and U(J,JJ) = 0. if J ≠ JJ
UU(J)	contains the value of the quadratic term corresponding to the Jth coefficient J = 1,NREG
VECTOR	contains temporary storage for values of the output variables
W(J,JJ)	contains the weighting factor for the JJ^{th} element of the Jth time series for the least squares objective function in a parameter estimation problem. By default W(J,JJ) = 1.0. J = 1,NTS and JJ = 1,NDPTS(J)
X(J)	contains the value of the Jth search variable for function evaluation J = 1,N
X(I,J)	contains the Jth coordinate of the Ith vertex in the current complex
XC(J)	contains the Jth coordinate of the centroid of the current complex J = 1,N
XG(J)	contains the value of the Jth search variable at the global optimum after SPEED is executed J = 1,N
XIN(J)	contains the current value of the Jth input variable J = 1,NVAR

XOUT(J)	contains the current value of the Jth output variable XOUT is updated every DT time step in the simulation SIM J = 1,MOUTS In addition XOUT is used for temporary storage in the subroutine PLOT J = 1,MOUTS+NTS
XVAR(J)	contains the default value of the Jth input variable J = 1,NVAR
XXX	contains the search variable coordinate values corresponding to EF on return from TRANS
XZ(J)	contains the initial value of the Jth search variable J = 1,N
Y	is an array storing temporary calculations
Z	is a real variable containing the intermediate function values corresponding to deviations from the base point

REFERENCES

1. Beveridge, Fordon S. and Robert S. Schechter, Optimization: Theory and Practice, (New York: McGraw Hill, Inc., 1970).

2. Buchner, Marcus R., An Interactive Optimization Component for Solving Parameter Estimation and Policy Decision Problems in Large Dynamic Models, Ph.D. Dissertation, Michigan State University, 1975.

3. Kuester, James L. and Joc H. Mize, Optimization Techniques, (New York: McGraw Hill, Inc., 1973).

4. Llewellyn, Robert W., FORDYN - An Industrial Dynamics Simulation, (North Carolina State University: 1965).

5. Morrison, Norman, Introduction to Sequential Smoothing and Prediction, (New York: McGraw Hill, Inc., 1969).

6. Powell, M.J.D., "An Interactive Method for Finding Stationary Values of a Function of Several Variables," Computer J., 5:147-151. 1962.

7. User's Guide, Vol. II for the 6500 SCOPE/HUSTLER Operating System, Computer Laboratory, Michigan State University.

8. Winer, Claudia, "SEQREG," Michigan State University, 1973.

9. Wolf, C., et al., "A Fortran Executive Program for Continuous Flow Simulation Models - SIMEX1," CLASS Document 13, (East Lansing: Agricultural Sector Analysis and Simulation Project, Michigan State University, October 1976).

CLASS Documents

Number	Title
1	"Computer Library for Agricultural Systems Simulation: A Progress Report," Michael H. Abkin and Tom W. Carroll. (July 1976)
2	"Software Standards Manual." (July 1976)
3	"Policy Analysis Language, Version 2.3, Programmer's Guide for CDC Cyber Computers," Claudia S. Winer and Chris Wolf. (July 1976)
4	"Policy Analysis Language, Version 2.3, Reference Manual," Claudia S. Winer anc Chris Wolf. (July 1976).
5	"Policy Analysis Language, Version 2.3, Guide to Using a PAL Program," Claudia S. Winer and Chris Wolf. (July 1976)
6	"DEMOGC: Demography with Distributed Age Cohorts," Michael H. Abkin and Chris Wolf. (July 1976)
7	"DEMOGD: Demography with Discrete Age Cohorts," Michael H. Abkin and Chris Wolf. (July 1976)
8	"Distributed Delay Routines: DEL, DELS, DELF, DELLF, DELVF, DELLVF," Michael H. Abkin and Chris Wolf. (July 1976)
9	"Table Functions: TABEL, TABEX, TABUL, TABUX," Michael H. Abkin, Chris Wolf, and Tom W. Carroll. (July 1976)
10	"AGACC: Accounting Routine for the Agricultural Sector," Dennis Pervis and Chris Wolf. (July 1976)
11	"User's Guide for the Beef Cattle Enterprise Simulation Model," Michael R. Jaske. (July 1976)
12	"User's Guide to SYSOPT: An Interactive System Optimization Computer Program," Marcus Buchner. (October 1976)
13	"A FORTRAN Executive Program for Continuous Flow Simulation Models - SIMEX1," Chris Wolf and Thomas J. Manetsch. (October 1976)