AGENCY FOR INTERNATIONAL DEVELOPMENT
WASHINGTON, D. C. 20523

BIBLIOGRAPHIC INPUT SHEET

FOR AID USE ONLY

1. SUBJECT CLASSIFICATION

A. PRIMARY: ECONOMICS

B. SECONDARY: AGRICULTURAL ECONOMICS

2. TITLE AND SUBTITLE

A SIMULATION LANGUAGE TO FACILITATE THE DEVELOPMENT AND USE OF SIMULATION MODELS IN POLICY, PROGRAM AND PROJECT ANALYSIS--PRELIMINARY OUTLINES

3. AUTHOR(S)

T.J. MANETSCH

4. DOCUMENT DATE: MARCH 1973

5. NUMBER OF PAGES: 24 PAGES

6. ARC NUMBER: ARC 309.20016-M275

7. REFERENCE ORGANIZATION NAME AND ADDRESS

MICHIGAN STATE UNIVERSITY
DEPT. OF AGRICULTURAL ECONOMICS
EAST LANSING, MICHIGAN 48824

8. SUPPLEMENTARY NOTES *(Sponsoring Organization, Publishers, Availability)*

9. ABSTRACT

Simulation models are finding increasing applications as aids to decision makers in exploring the likely consequences of alternative actions and in formulating "good" policies, programs and projects based on a creative interaction process involving decision makers, analysts and the results of successive simulation "runs".

One purpose of this paper is to sketch necessary requirements for and some features that might be useful in a simulation language designed particularly to suit the needs of decision makers in LDCs concerned with policy, program and project formulation This is based largely on the experience of the MSU team to date in building and using models in Nigerian, Brazil and Korea. The discussion is a first attempt at "defining the problem" and it is hoped that the paper will stimulate further discussion leading to a more complete answer to the question "what should a computing system do in order to meet the needs of decision makers at the policy, program and project levels in LDC's? A second purpose of the paper is to present some initial thoughts on how a simulation language might be constructed to accomplish the kinds of objectives that have been set fourth.

10. CONTROL NUMBER: PN-AAB-996

11. PRICE OF DOCUMENT:

12. DESCRIPTORS

LDC's, NIGERIA, BRAZIL, KOREA. DECISION MAKERS. ANALYSTS, SIMULATION RUNS

13. PROJECT NUMBER: 931-17-140-536

14. CONTRACT NUMBER: AID/CSD-2975

15. TYPE OF DOCUMENT: SECTOR ANALYSIS STUDY

AID 590-1 (4-74)

A Simulation Language to Facilitate the Development and Use of Simulation Models in Policy, Program and Project Analysis--Preliminary Outlines

Working Paper 73-1[1/]

T. J. Manetsch

Introduction

Simulation models are finding increasing applications as aids to decision makers in exploring the likely consequences of alternative actions and in formulating "good" policies, programs and projects based on a creative interaction process involving decision makers, analysts and the results of successive simulation "runs". ("Simulation" is used here in a broad sense to mean a problem-solving approach which applies a broad range of quantitative methods in a coordinated resolution of planning and management questions (8).) Experience has shown that simulation "languages" (collections of special purpose computer programs) can greatly facilitate the construction of these decision making models and enhance the effectiveness with which decision makers can interact with them.

To date a number of simulation langauges have been developed and some of them are finding extensive use as aids to decision making (1, 2, 3, 4, 7, 10, 11). In the writer's opinion none of these fully exploits the potential that exists for furthering the overall effectiveness of planning and management in LDC's. Most are, in fact, inappropriate in this context

---

[1/] This paper was developed as part of Contract AID/csd-2975 Agricultural Sector Analysis and Simulation Projects.

due to the rather unique constraints imposed by operation in many countries on many sizes and makes of computers. Most existing simulation languages are simply not compatible with the range of computers available in LDC's.

One purpose of this paper is to sketch necessary requirements for and some features that might be useful in a simulation language designed particularly to suit the needs of decision makers in LDC's concerned with policy, program and project formulation. This is based largely on the experience of the MSU team to date in building and using models in Nigerian, Brazil and Korea. The discussion is a first attempt at "defining the problem" and it is hoped that the paper will stimulate further discussion leading to a more complete answer to the question "what should a computing system do in order to meet the needs of decision makers at the policy, program and project levels in LDC's?" A second purpose of the paper is to present some initial thoughts on how a simulation language might be constructed to accomplish the kinds of objectives that have been set forth.

Some Requirements for a Simulation Language (Computing System) for Aiding Decision Making in LDC's

A fundamental requirement is compatibility--to be widely useful the system must be operable on a wide range of sizes and makes of computers that are encountered in countries around the world, developed and developing. This strongly suggests that the system should be compatible with FORTRAN--the closest thing to a universal computer language. Further, the system should be scalable so that it can address meaningful problems on both large and medium to small computers. A related requirement is that the system should be capable of incorporating the wide range of software components

likely to be stored in the kinds of software libraries under current discussion. This requirement is automatically satisfied if the entire system, language and software library elements, are based on a general purpose programming language such as FORTRAN.

Other important requirements relate to the need to provide close and effective interaction with decision makers. Key data inputs specifying policies to be explored and model outputs describing consequences of decisions taken must be in problem related formats which are meaningful to decision makers and easily learned. The system should provide for on-line interaction between decision makers and models. Decision makers should be able to set policies, etc. and review results and revise decisions, if desired, during the course of a simulation run. Decision makers might also profit from knowledge of the specific factors the model has included in producing the outputs being evaluated. In some cases models should produce statistics which display the ranges of outputs that are possible given certain policy actions and the uncertainties inherent in the system and data. (This involves operating models in the so-called Monte Carlo mode). In other cases it may be useful to operate models in an optimization mode in which a criterion selected by the decision makers is maximized or minimized.

There are a number of other features a formal computing system can provide which can greatly facilitate the development of decision-making models and improve their accuracy and reliability. These are listed below along with short descriptive comments.

1. Pre-programmed input and output formats. These can provide decision makers with an effective means of communicating with models without special programming for each application in each country. These might include special table formats, summary tables, graphs, histograms, etc.

2. Ability to conduct sensitivity analysis on any model coefficient. These tests permit the user to examine the consequences of changes in the values assigned to model coefficients, including those which specify policy inputs (investment levels, tax rates, price policies, etc.). For a change in a specified coefficient the model computes the likely changes in system behavior that will result. These tests can serve at least two purposes: they can indicate the priority policies which are likely to have the greatest impact on system performance and they can indicate where errors in available data (as they affect values assigned to model coefficients) are most serious in terms of affects on criteria used in decision making. The latter information can lead enlightened data gathering strategies aimed at improving information in areas which are likely to have high payoff in terms of improving the quality of decision-making models.

3. Time series tracking. As part of the model testing and validation process it is often helpful to test the model against the past behavior of the real system. This is done by comparing model results with historical results and computing some measure of performance or "fit" such as weighted sum-of-squared errors. It is sometimes appropriate to use such tests as means of determining values for unknown model coefficients, i.e., values for certain coefficients are adjusted so that a good or best fit with historical data is attained. Using modern programming techniques it is often possible to automate the process of optimally adjusting a complex model so it fits historical data taken from the real world. Automated time series tracking is another feature that should be seriously considered as a part of a simulation language in the present context.

4. The computer language should readily incorporate a host of different components, dynamic simulation, linear programming, critical path analysis (for study of project implementation), project appraisal, etc.

5. The system should provide for ease in the use of overlays to extend the model size that can be accommodated on a given computer.

6. The system should readily permit a large model to be operated in segments, i.e., separate operation of regional models within a larger sector model.

7. Special error diagnostics (not available in the base language, i.e., FORTRAN, etc.). The system should provide special error diagnostics which aid model testing and validation. Included might be tests for:

a. model variables of incorrect sign

b. model variables of inappropriate magnitude

c. flow conservation constraints violated

d. circuit constraints violated

e. incorrect computing sequence

f. numerical errors (integration, etc.) out of bounds

g. others?

The language should provide printouts which indicate the sections of the model in which specific errors have been located.

8. Collection of data from models for use in constructing response surfaces.

9. Estimation of "good" response surfaces to approximate models using recursive least squares, etc.

10. Automatic documentation of model changes

a. values assigned to coefficients

b. changes in model equations

11. Others?

The above constitutes the beginning of a "shopping list" of desirable features to be considered for inclusion in a decision-oriented language. Some of these features are much easier to implement than others. A number (sensitivity analysis, Monte Carlo analysis, time series tracking, output formats, etc.) have been included, at least in rudimentary form, as peripheral features in simulation models developed at MSU (8). A great deal of additional work remains, however, to generalize and extend the approach to include as many priority features as are feasible within budget and computer constraints. At this point interaction is needed among decision makers, analysts and computer specialists to determine appropriate detailed specifications for such a computing system. Suggestions and comments along these lines are solicited.

The next section of the paper moves to a more technical level to begin to address questions relating to how such computing systems might be implemented. It is expected that this discussion will be of more interest to analysts and computer specialists than to persons interested principally in decision making applications. The latter group of readers may wish to, at most, skim the remainder of the paper (not forgetting that the enterprize needs their inputs regarding the features they feel should be incorporated in the computing system!)

On the Implementation of a Computing System to Facilitate the Construction and Use of Decision-Making Models

We will proceed by examining in moderate detail how some of the features described above might be incorporated into an organized structure (language or system) which permits them to be readily applied to a wide range of specific decision-making problems. Specifically we will consider as a starter a system which includes:

1. ordinary (non-interactive) policy, program or project analysis
2. decision analysis with interaction which permits decision makers to insert decisions "on line"
3. optimal decision analysis wherein the system seeks a set of decisions which will optimize some specified criterion
4. sensitivity analysis conducted on any specified model coefficient or set of coefficients
5. Monte Carlo analysis to compute statistics for likely model outcomes
6. ordinary tracking of models against historical data generated by the "real world"
7. automatic model tracking to adjust unknown model coefficients so the model gives a "best fit" to historical data.

The approach described here is an extension of that used in programming the Nigerian simulation model (9).

We will discuss the system using a series of flow charts. It will be seen that a key to the development of such a system is the appropriate definition of <u>sets of model variables</u> (and coefficients)--that various specific functions are implemented by using and operating on specific variable sets. It is this feature which makes it possible to develop a general computing

system capable of performing desired functions on a wide range of specific models. In what follows these various variable sets will be written in brackets as [•].

The Executive Program

A flow chart for the overall executive program for the system is shown in Figure 1. The executive program begins by reading a set of control parameters [CP] specified by the user. These parameters specify which function is to be carried out on the specific computer run or series of runs (Monte Carlo, sensitivity analysis, policy analysis, etc.). They also specify the characteristics of the run or series of runs--the time horizon the model is to operate over, the number of runs, the time increment used in simulation, the type of output desired (detailed or summary, tables, graphs, histograms, and the time interval between printouts of model output), etc.

Next the executive program reads or prepares for the reading (from mass storage) of the input data required to implement the function that has been specified for the given run or runs. This will include all variable sets required as input and equations describing models to be executed. Included will be default values for all model parameters. They may be altered later, as in sensitivity analysis, but all relevant parameters are assigned values. Note that only data sets and models needed for execution of the specified function are read into core memory at this point. This feature adds to the scalability of the system vis-a-vis various size computers.

The executive program then transfers control to the one of seven functions that has been specified by the user for the given series of computer runs. These functions correspond to the seven branch points in Figure 1. as follows:

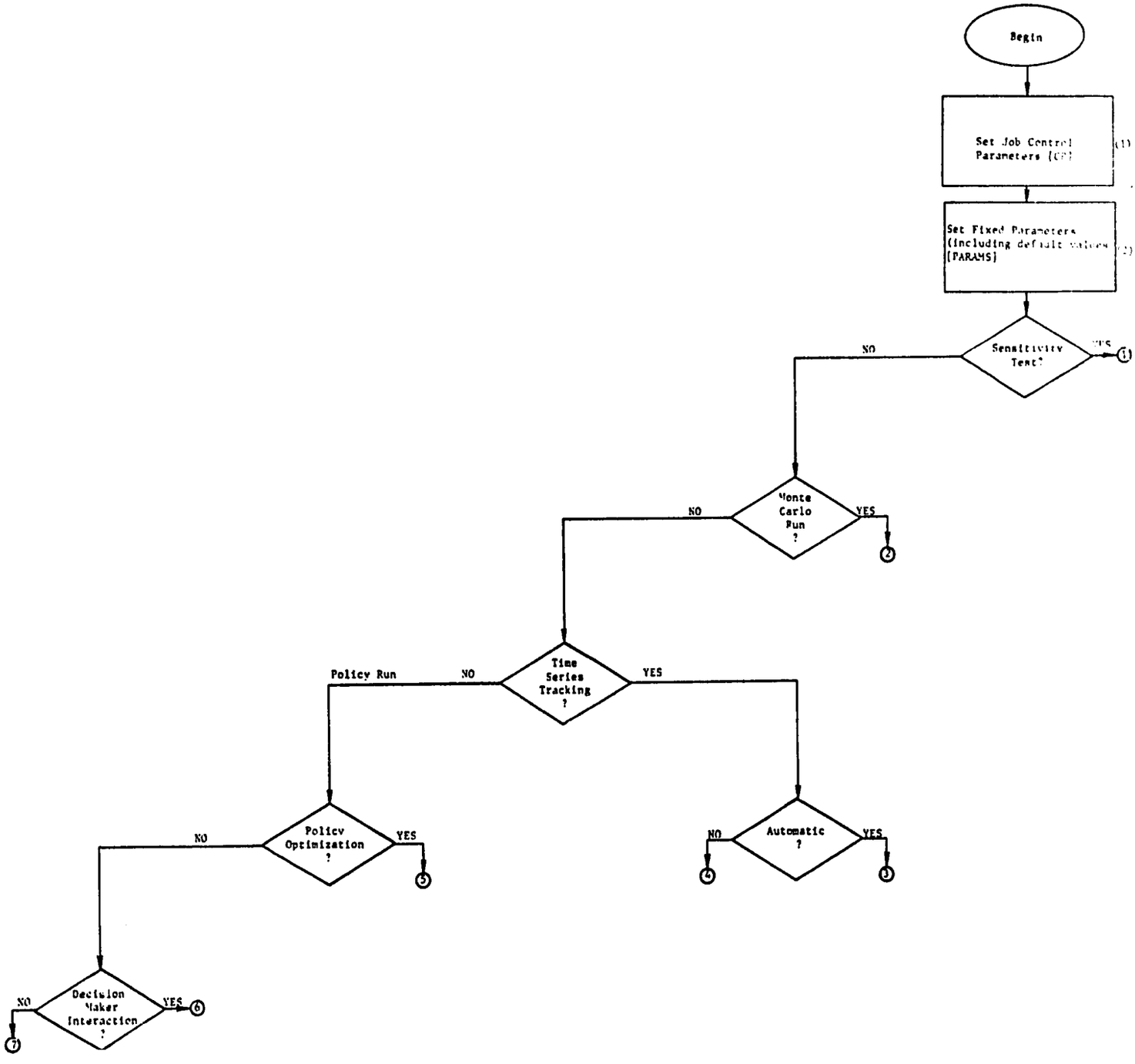


Figure 1. Major Branches in Simulation Language Executive Program

1. sensitivity analysis
2. Monte Carlo analysis
3. automatic time series tracking
4. manual time series tracking
5. policy runs[1/] with a criterion optimized
6. policy runs[1/] with on line decision maker interaction
7. standard policy runs[1/] (no optimization, no Monte Carlo, and no decision maker interaction.

We will present and discuss flow charts which describe the execution of each of these functions. In so doing we will define basic variable sets needed by a general purpose simulation language capable of executing the various functions.

A flow chart for sensitivity analysis is shown in Figure 2. Operation begins by setting a new value for the parameter(s) to be changed on the current run, a member of the set [SENPAR], and by specifying non-default values for the policy variables [POL]. The set of variables [SENPAR] is supplied by the user of the model in the following form for all parameters to be changed in a set of runs: (variable set name; name of variable, new value or percent increase/decrease, ...). The program next sets all initia conditions for the model--the variable set [IC]. These variables determine the initial state of the model at the beginning of a simulation run and include the stock (state) variables and other variables that must be reset between successive simulation runs. The model is next executed through one complete simulation run ((3) and (4)) and parameters are reset to default

---

[1/] Runs to examine consequences of specific policies, programs and projects.

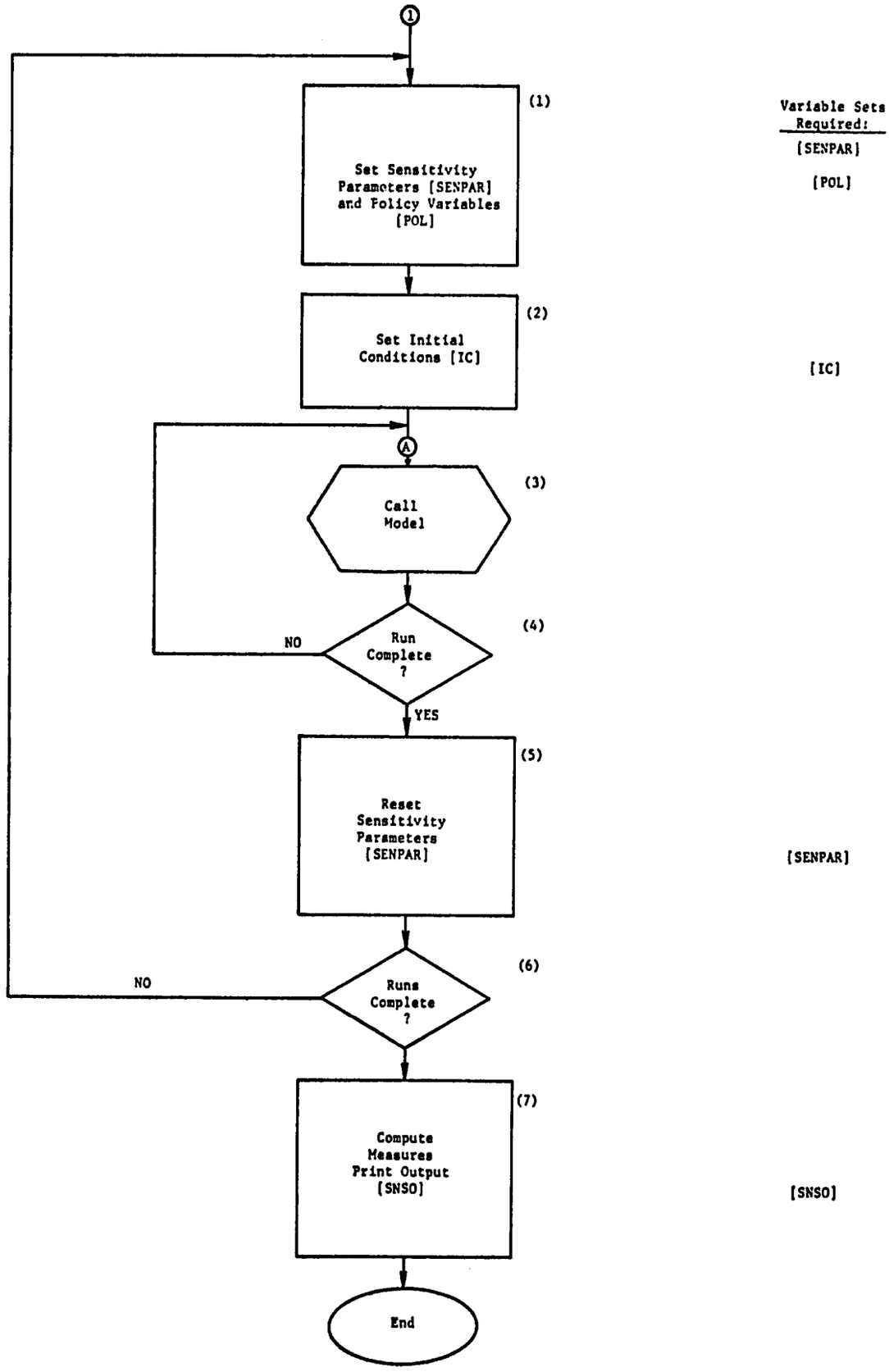


Figure 2. Sensitivity Runs--Flow Chart

values (5). The loop (1) through (6) is executed once for each sensitivity run that has been specified. After all sensitivity runs have been executed, output is provided. Specifically, output is provided for each run for the variable set [SNSO] specified by the user. Output format can also be determined by user specified control parameters (probably elements of [CP]).

Monte Carlo Runs

A flow chart for execution of Monte Carlo Runs is shown in Figure 3. Policy variables for the series of Monte Carlo runs [POL], are set in (1). Only values which are departures from default values need be specified at this point. These new values, if any, are specified by the user and enter the system in card format. At (2) and (3) data are acquired which specify which variables are to take on random values and their statistical properties. The user provided parameter set [RPW] provides the following information for each parameter that varies randomly <u>between</u> simulation runs:

i. name of variable

ii. parameters which specify the probability distribution for the variable (mean and variance; smallest value, most likely value, maximum value; etc.)

Similar information is provided for the parameter set [RPW] which includes parameters which vary randomly <u>within</u> simulation runs. In addition a correlation coefficient is specified. After the execution of a series of simulation runs (steps 4-9 in Figure 3.) statistics are computed and printed for the members of user specified parameter set [SMC].

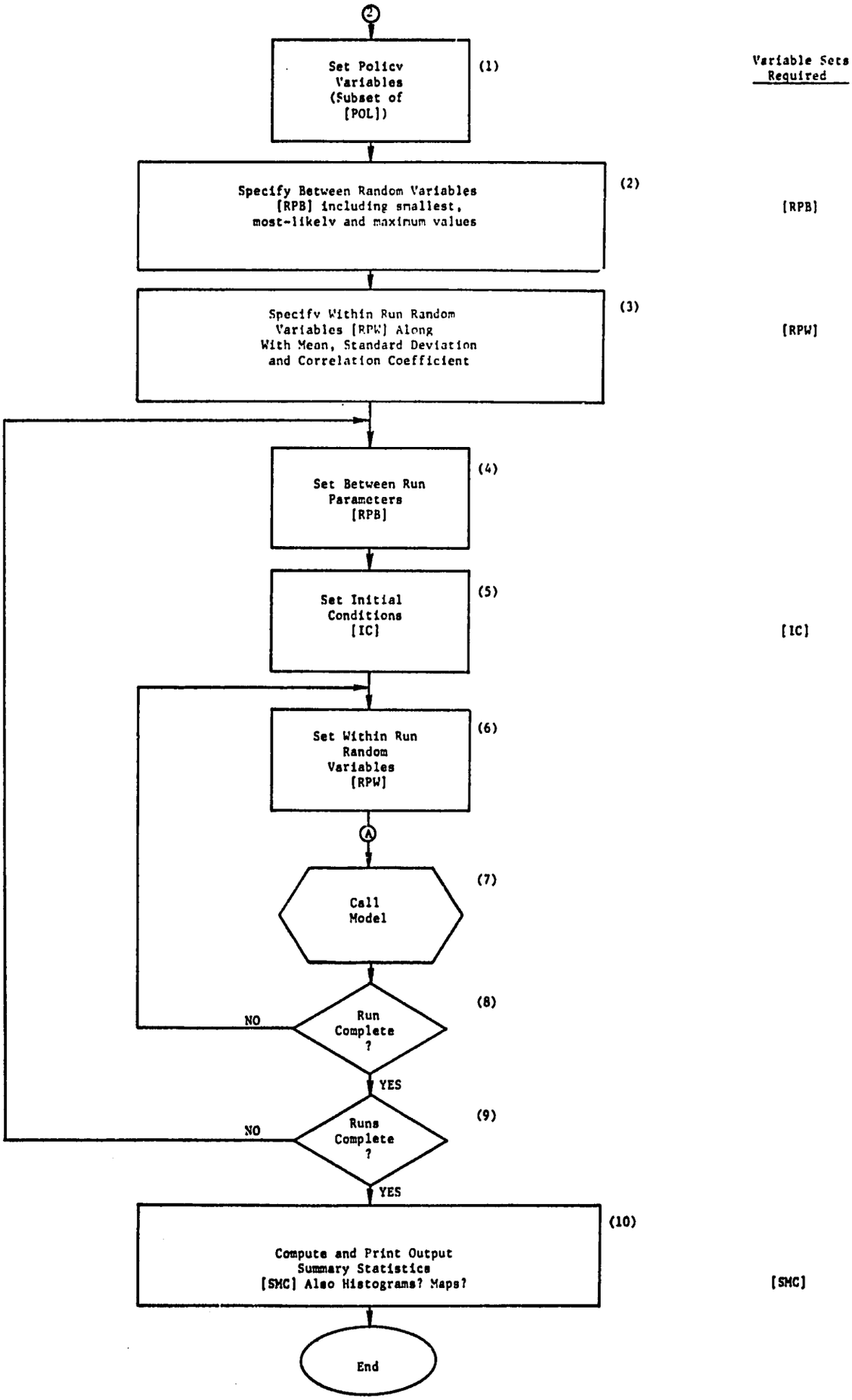


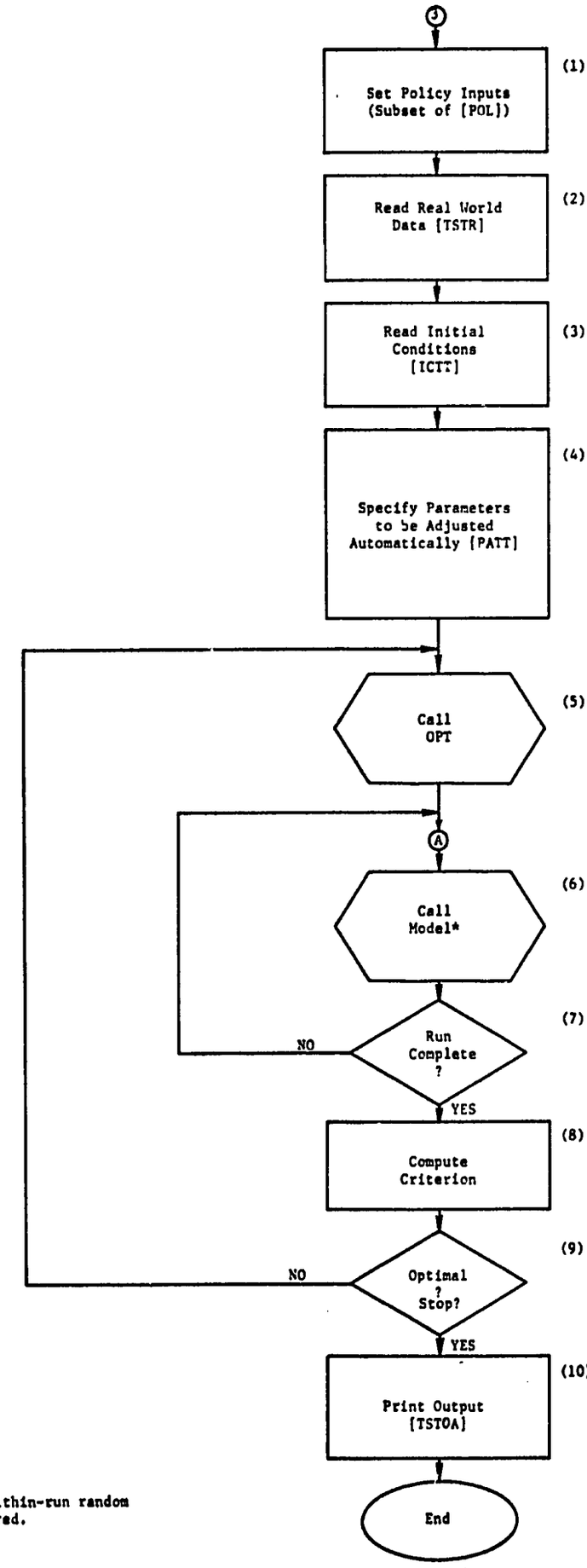
Figure 3. Monte Carlo Runs--Flow Chart

Automatic Time Series Tracking

Operation begins by setting any members of policy variable set [POL] which are different from default values (see Figure 4.). Next the real world time series the model is being compared with, [TSTR], is read into core memory along with the specific set of initial conditions [ICTT] that apply. In general [ICTT] will be different from [IC] because models will usually be started at different real world times in the tracking and policy analysis modes. In (4) the model is given the user-specified set of parameters [PATT] that are to be automatically adjusted in the tracking and tuning process. [PATT] also includes any constraints that apply to its members. Operations (5)-(9) optimize the fit between the model and historical data, [TSTR], by adjusting the parameter set [PATT] subject to specified constraints on members of [PATT]. The program OPT in block (5) is a collection of optimization routines (simplex, response surface, search, etc.) designed to efficiently locate optima for large scale dynamic simulation models. Finally, output is provided for the user specified set of output variables [TSTOA].

Ordinary Time Series Tracking
(Without Automatic Parameter Adjustment)

Operation in this mode involves essentially the same variable sets as automatic time series tracking discussed above. See Figure 5. Since parameters are being adjusted manually the user may wish to evaluate internal model behavior in detail before making decisions about further model changes. For this reason it may be useful to have the system provide a detailed



Variable Sets Required

[POL]

[TSTR]

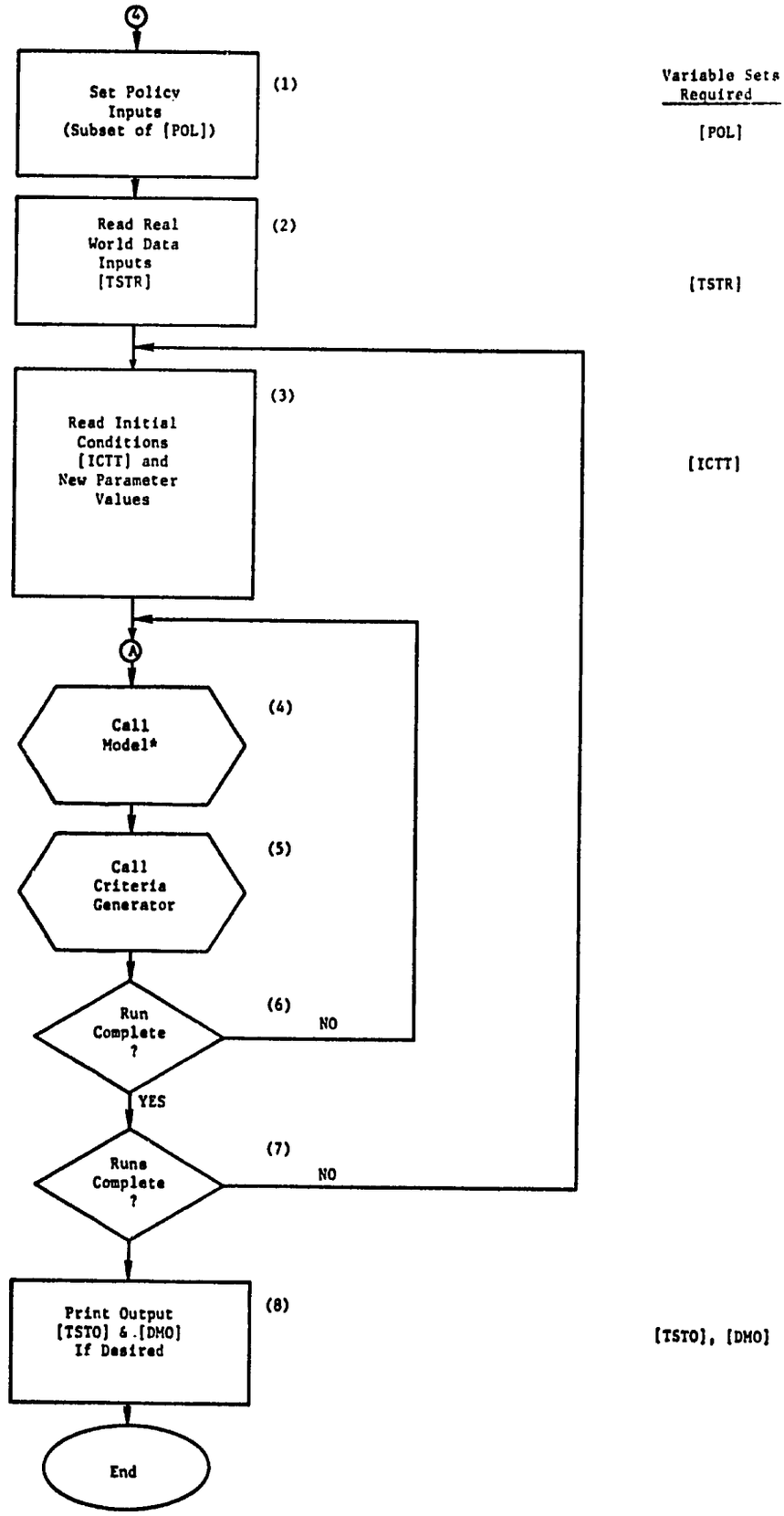
[ICTT]

[PATT]

[TSTOA]

*Should include within-run random variables if desired.

Figure 4. Automatic Time Series Tracking--Flow Chart



*Should include within-run random variables if desired.

Figure 5. Time Series Tracking (Without Automatic Parameter Adjustment)--Flow Chart

output of model output (the variable set [DMO] in 8).[1/]

Policy Optimization

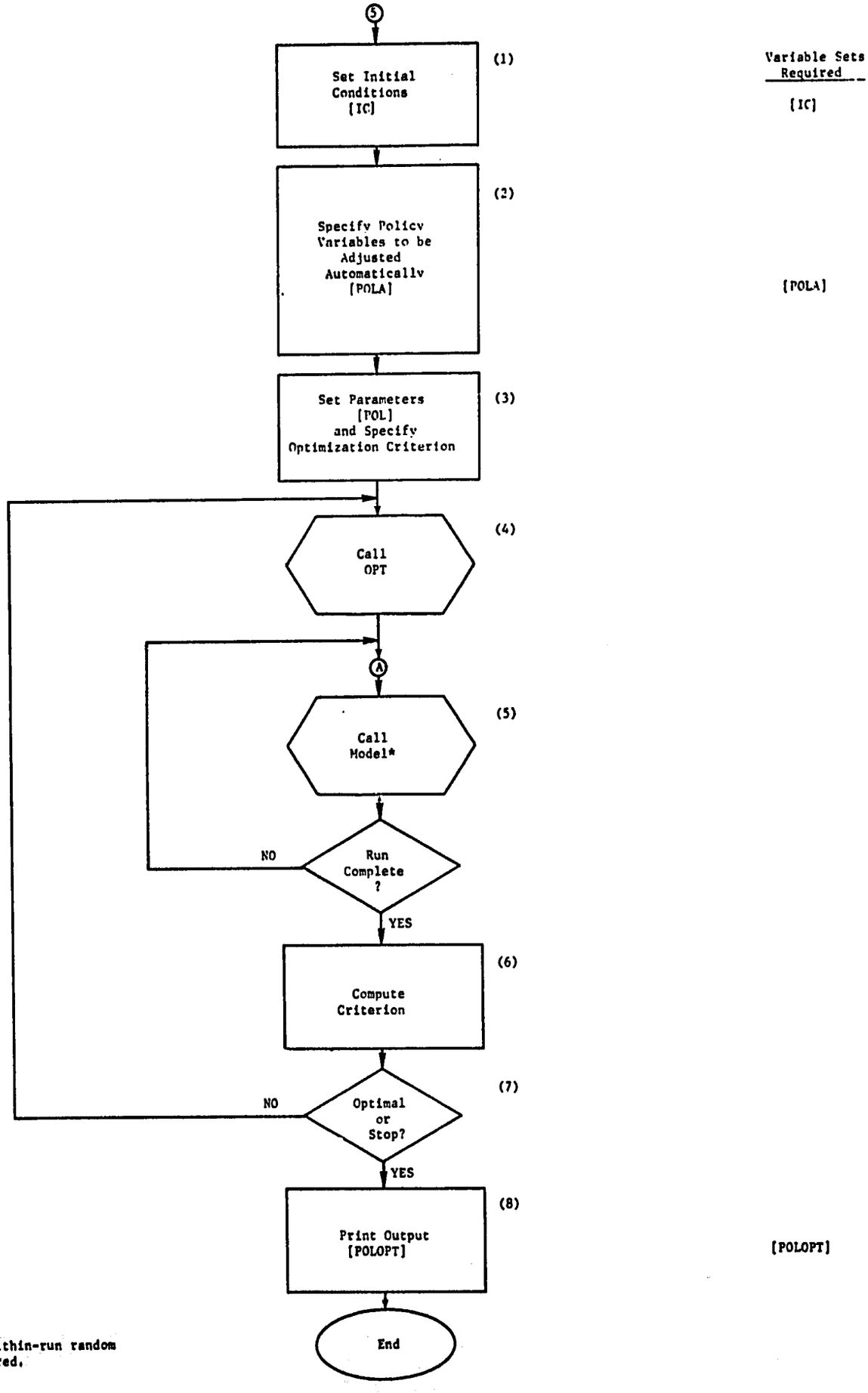
A flow chart for the operation of models in a policy optimization mode is shown in Figure 6. (The word "policy" here is used in a broad sense and "policy optimization" means finding the specific set of decisions that optimizes the selected decision making criterion or performance variable). The set of model initial conditions [ICC] is specified in (1). Next the model is given the set of policy variables to be automatically adjusted during optimization, [POLA]. (A great deal of preliminary work, sensitivity tests, etc. is necessary before this set can be specified rationally. Careful specification will also help reduce computational costs which tend to be high for this function). In (3) non-default policy variables are set, [POL], and the user-specified optimization criterion is supplied. The system then enters an optimization loop (4)-(7). The block "OPT" in (4) can, in all probability, be the same collection of optimization routines used in automatic time series tracking discussed earlier. Operation concludes by printing the output variables desired by the user, [POLOPT].

Standard Execution of Policy Runs
(No Optimization or Decision Maker Interaction)

This operation is very straight forward requiring the specification of initial conditions [IC], desired values for policy variables, [POL] and the

---

[1/] This kind of information might also be useful when operating in the automatic tracking mode. It could serve as a means of checking the reasonableness of an "optimal" parameter set.



*Should include within-run random variables if desired.

Figure 6. Policy Optimization--Flow Chart

set of output variables to be used in evaluating system performance, [POLO]. Figure 7. describes this function.

Execution of Policy Runs with Decision Maker Interaction

A flow chart for the execution of policy runs with decision maker interaction is shown in Figure 8. In this mode decision makers can interact with models "on line" changing policy inputs at any desired point in simulated time. The decision maker, in essence, sets non-default values for [POL] as he interacts with the model through time. At interaction times the model presents the current status of the variables [DMIO] which are used to evaluate the impact of policies upon the system. This variable set is also specified by the decision maker and should be readily changeable as interaction proceeds. A system control parameter (TNI) determines the simulated time of the next interaction. This parameter is set at each interaction time to schedule the next interaction. Table I summarizes the variable sets required to implement the seven functions just described.

Concluding Remarks

This concludes, in rather broad-brush fashion, description of a computing system designed to implement a number of functions useful in the development and application of decision-making models. Much more detailed work is needed to implement the kind of system described. The development of such a system would logically follow a study to determine the most important features to be included from the standpoint of practical usefulness. Detailed system development would be a sizable undertaking involving 2-10 man-years of effort depending upon the scope of objectives. From this

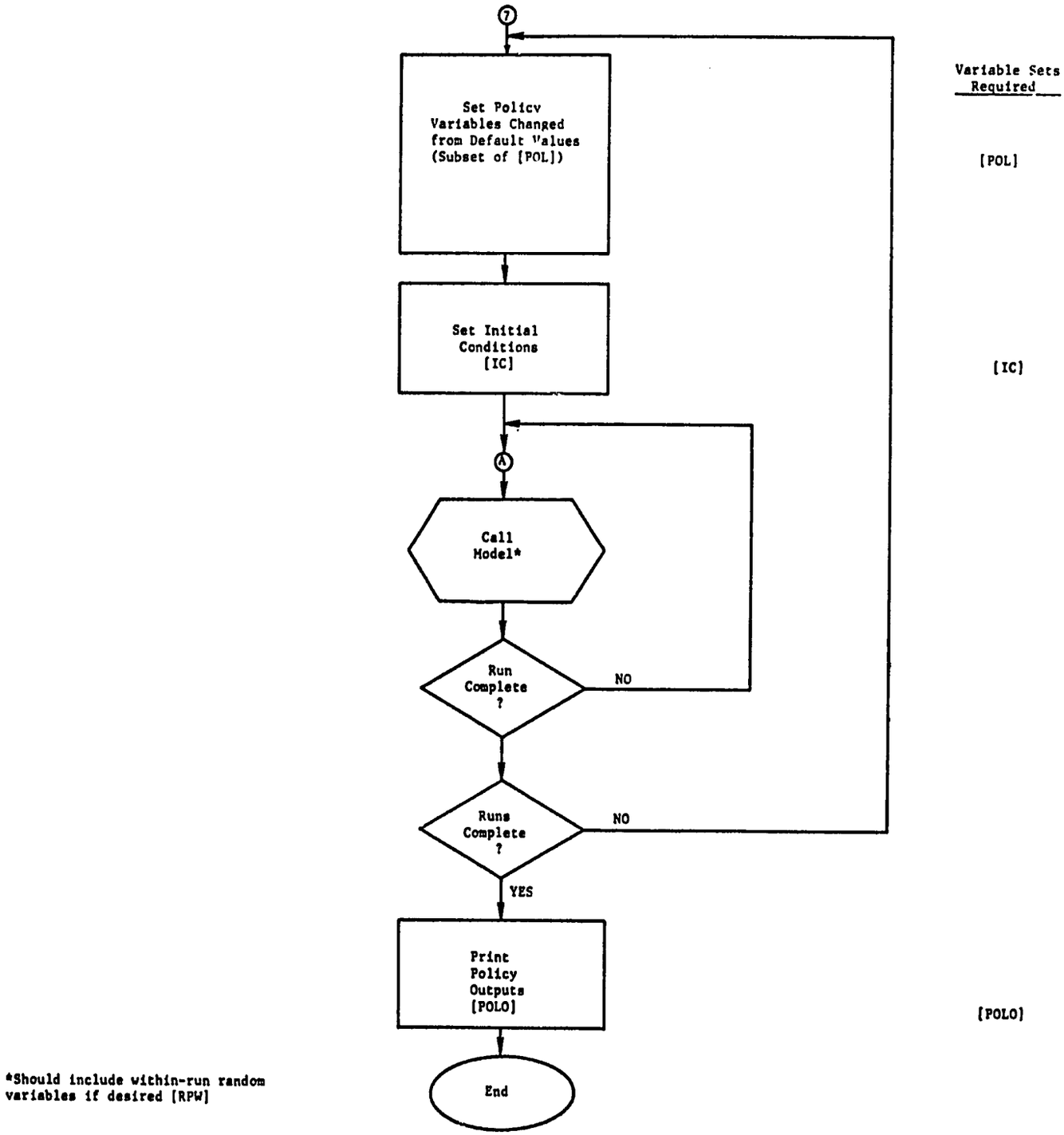
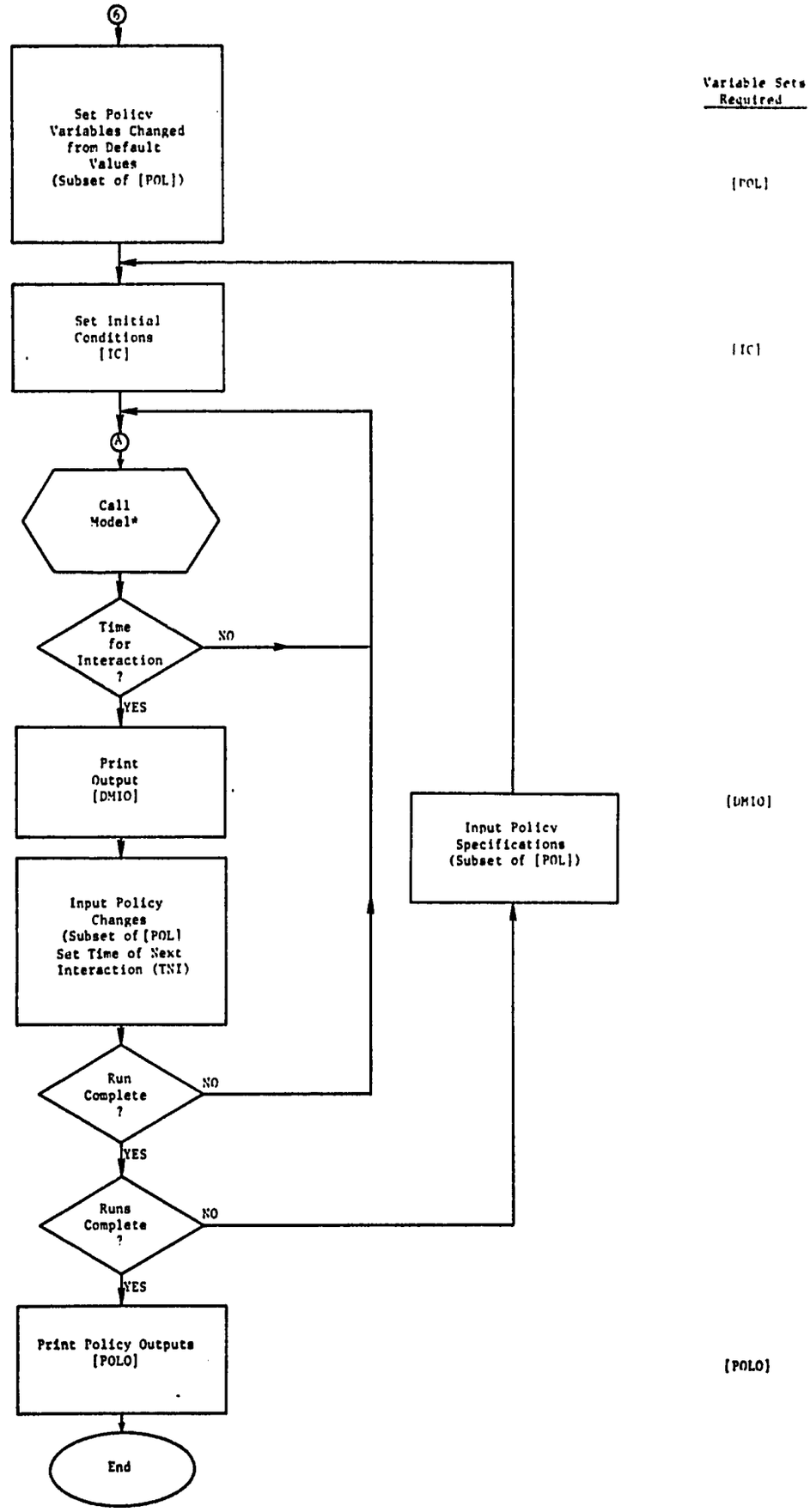


Figure 7. Standard Execution of Policy Runs (No Decision Maker Interaction or Optimization)



*Should include within-run random variables if desired [RPW]

Figure 8. Execution of Policy Runs (With Decision Maker Interactions)--Flow Chart

Table I

Variable Sets Required to Implement Seven Basic Functions of a Simulation Language for Decision Analysis

Variable Set	Description	Use
[CP]	Control parameters which specify number, type and characteristics of computer runs.	Executive Program
[DMO]	Those output variables which constitute a detailed printout of model variables (of primary use to programmers and analysts in evaluating validity of model performance).	Various Functions
[DMIO]	Those output variables used by decision makers to evaluate results of decisions taken in an interactive mode.	Policy Runs With Decision Maker Interaction
[IC]	The set of values assigned to model variables whose values change during runs and which must be initialized at the beginning of each simulation run. These typically include price levels, capital stocks, population levels and other stock variables. (This set provides initial values for the "state" variables of the system.)	All Functions Except Time Series Tracking
[ICTT]	The set of initial values assigned to state variables when model is run in a time series tracking mode.	Time Series Tracking
[PATT]	The set of parameters that is adjusted automatically during time series tracking.	Time Series Tracking
[POL]	The set of model variables which is set by decision makers to define specific policy, program or project alternatives to be explored.	All Runs to Explore Consequences of Alternative decisions
[POLA]	The set of variables (a subset of [POL]) which is adjusted automatically during runs to optimize a decision-making criterion.	Policy Runs in an Optimization Mode
[RPB]	The set of model parameters that takes on random values between model runs. Each element of [RPB] is an array that contains the variable name and sub-parameters that describe its probability distribution.	Monte Carlo Runs
[RPW]	The set of model parameters that varys randomly within a given simulation run. Each element of [RPW] is also an array that contains the variable name and sub-parameters that define its statistical properties--mean, variance and auto-correlation coefficient.	Any Runs

Table I (continued)

Variable Set	Description	Use
[SENPAR]	The set of parameters whose values are to be changed during sensitivity runs. Each element of [SENPAR] is an array that contains the parameter name, its nominal value and information defining the change to be made in the parameter value during a sensitivity run.	Sensitivity Runs
[SMC]	The set of summary statistics to be printed out after a series of sensitivity runs. Each element of [SMC] is an array containing the variable name, and the specific statistics required, i.e., means, standard deviations, number of observations exceeding a prescribed level, etc.	Monte Carlo Runs
[SNSO]	The set of variables to be printed out to evaluate the results of sensitivity runs. Each element of [SNSO] is a variable name.	Sensitivity Runs
[TSTO]	The set of variables to be printed out to evaluate a time series tracking run. Each element of [TSTO] is also a variable name.	Time Series Tracking Run
[TSTR]	The set of variables that represent the real historical behavior of the system under study. Each element of [TSTR] is an array that contains a variable name, K values that correspond to K past values of the variable named, the time corresponding to the earliest historical data value and the time interval between historical values.	Time Series Tracking

vantage point it appears feasible to develop such a system in modular fashion--beginning with certain selected functions and supplementing these as it becomes expedient to do so.

Experience has shown that formal computing systems (simulation languages) can pay great dividends in terms of reduced model development costs and increased effectiveness in formulating decisions. We have every reason to expect that these advantages can be further extended in the area of decision making in LDC's by the development of a simulation language designed to meet the unique set of needs that exist in this particular area of application. Such a language should be viewed as an integral part of a software library of models and model components.

Bibliography

1. IBM Corporation, General Purpose Systems Simulator II. White Plains, New York: IBM Corporation, 1963.

2. IBM Corporation, System/360 Continuous System Modeling Program. Application Description 360A-CX-16X, GH20-0240-3, 1972.

3. IBM Corporation, System/360 Continuous System Modeling Program User's Manual. Program Number 360A-CX-16X, GH20-0367-4, 1972.

4. Kiviat, Phillip J.; Villanueva, R.; and Markowitz, H. The SIMSCRIPT II Programming Language, Englewood Cliffs: Prentice Hall, 1969.

5. Krasnow, Howard S., and Merikallio, Reino A. "The Past, Present and Future of General Simulation Languages. Management Science, 11 (Nov. 1964), 236-367.

6. Krasnow, Howard S. "Computer Languages for System Simulation." Digital Computer User's Handbook. Edited by Melvin Klerer and Granino Korn, New York: McGraw-Hill Book Company, 1967.

7. Llewellyn, Robert W. FORDYN--An Industrial Dynamics Simulator. Raleigh: North Carolina State University, 1965.

8. Manetsch, T. J., et al., A Generalized Simulation Approach to Agricultural Sector Analysis with Special Reference to Nigeria. East Lansing: Michigan State University, 1971.

9. Page, Gloria; Turnquist, Mark and Carroll, Tom W. Technical Reference Manual for the Nigerian Simulation Model Version 7, Agricultural Sector Analysis and Simulation Projects, November 1971.

10. Pritsker, A. A. B., and Kiviat, Phillip J. Simulation with GASP II. Englewood Cliffs: Prentice Hall, 1969.

11. Pugh, Alexander L. Dynamo User's Manual. Cambridge: The M.I.T. Press, 1963.

12. Teichrow, Daniel, and Lubin, John F. "Computer Simulation: Discussion of Techniques and Comparison of Languages." Communications of A.C.M. 9, 10 (Oct. 1966), 723-741.

13. Tocher, K. D. "Review of Simulation Languages." Operations Research Quarterly, 16 (June 1965), 189-218.