Combined Multidimensional Simulation Language, Database Manager and Sensitivity/Confidence Analysis Package For System Dynamics Modeling

George A. Backus, President
Policy Assessment Corporation

Jeffrey S. Amlin, President
Systematic Solutions, Inc

ABSTRACT

A fast, interactive, large-scale, database-oriented simulation language (LAMDA) for IBM compatible microcomputers has been developed for System Dynamics' applications. Sophisticated output features include high-resolution graphics, full report generation capabilities with textual explanation, and user defined screen menu options. An integrated sensitivity/confidence package allows parameters or structures to be evaluated as a function of time and as a function of all other parameters for their impact on model results. Confidence intervals are determined along with the time and circumstances where certain information is critical. An integrated statistical package can be used to estimate model parameters based on historical information or, in combination with the data base, used to test model hypotheses and statistical inferences. LAMDA can also interface with FORTRAN applications. Extensive dialogue capabilities allow the model builder to make the model user-friendly and fit the need/sophistication of the client.

LAMDA SIMULATION SOFTWARE SYSTEM

LAMDA -- Language for Advanced Modeling and Data Analysis -- is a general purpose, structured simulation system that can be used for a variety of multidimensional applications, including information management, simulation, and mathematical modeling in business, engineering and the sciences. A modified version called LAMDA-SIM was especially designed for System Dynamics applications. (Backus 1985)

LAMDA is fast, interactive, database-oriented and transportable to all IBM-compatible microcomputers. It can easily simulate models containing over 32,000 variables.

LAMDA has sophisticated output features including character, medium- and high-resolution color graphics which provide multivariable line plots, histograms, and bar graphs with titles, labels and scaling. Plots do not have to relate a variable to time but can relate one or more variables to another.

LAMDA provides complete report generation capabilities. Tabular output can be formatted for column or row labels with explanatory text interspersed. With LAMDA's logical expression capabilities, model results can be analysed and the conclusions stated in prose rather than tables. The user can interact with LAMDA via a dialogue rather than only through PLOT or PRINT statements. The input or output of any model can be obtained from user-defined menu screens. Menus can present user choices through the process of highlighting predefined options. Other menus can provide a structured format for users to enter input in a manner specific to their needs and desires. Tutorials and help screens can also easily be added to any model. LAMDA itself has an on-line tutorial available whenever needed. Additional output variables or equations can be mathematically defined interactively in "rerun" mode. Input and output can also be put in to "electronic spreadsheet" files for use with other popular programs.

LAMDA operates interchangeably between two modes: direct and indirect. In the direct or command mode, LAMDA interactively accepts a statement, converts it to executable instructions and proceeds to the next statement. In the indirect or compile mode, LAMDA compiles a group of statements called procedures which may be executed later as a single unit. A procedure may be called for execution by other procedures including itself. Procedures can require arguments and contain local variables. From a limited view, procedures can be thought of as DYNAMO macros which can call other macros. Also from a limited view, LAMDA's interactive mode can be thought of as DYNAMO's rerun mode; constants, tables, and output can be modified interactively. New procedures and equations, however, can also be defined interactively. Any LAMDA compatible syntax can be entered in interactive or compile mode. LAMDA can be executed from a batch file when multiple runs and extended input data/changes are required. Models can be productized, compiled, and incorporated into LAMDA's RUN-TIME package. With the RUN-TIME package, models can be executed as a standard application by users with limited computer experience and no knowledge of LAMDA.

The information in a LAMDA program is composed of variables, sets, and structures. Variables store the information and sets classify it. Variables are multidimensional arrays whose dimensions (up to 10) or subscripts are sets. Because of the correspondence between sets and variables, LAMDA can operate on variables without the need for a complicated subscript reference.

The statements of LAMDA are organized into well-defined complete, causal structures. A LAMDA statement is not simply a line of code; it contains built-in internal structure which the user does not have to program. LAMDA statements have no line numbers and no GOTO statements. With LAMDA's structural syntax they are not needed. Full 8087 numerical processing is automatically supported.

LAMDA contains a full screen editor, a data manager, and a program manager. The full screen editor enables the user to develop and modify models and organize the resultant files on mass storage. It also enables the user to correct programming errors as they occur during the compilation.

The database manager can reduce the memory requirements of data storage or allow all data to be stored for later review. Results from multiple runs can be stored in such a way as to focus on critical portions of the analysis. Model generated data can be manipulated and reviewed without "rerunning" the model. The database functions also allow the user to "resume" a simulation from any time period. The data manager can be used interactively and independently from the program code. Data files can be textual, random access, or in LAMDA array format for use with none LAMDA pre- or post-processors. With the database, multiple run can be stored for later comparison analyses.

The program manager allows extremely large programs to be executed on the microcomputer by dividing the code into segments and organizing them into a hierarchical tree structure. To date, even programs that could not fit on large mainframes without virtual memory fit easily on the microcomputer without the need for program management. Additionally, the existing applications execute in only twice the time required on the mainframe.

An integrated sensitivity/confidence package (HYPERSENS) allows parameters or structures to be evaluated as a function of time and other parameters for their impact on model results. (Ford 1983, Amlin 1985) The approach is particularly useful for determining the robustness of policies tested with the models. The package can show the need or lack of need for additional data. Confidence intervals are determined along with the time and circumstances where certain information is critical. Errors in model logic and coding become obvious. All probable behavior modes of the model are concomitantly simulated as part of the sensitivity analysis process. HYPERSENS uses Latin-Hypercube sampling methods to vary all parameters simultaneously to insure full sensitivity analysis of all variables over the complete time horizon of the model.

An integrated statistical package can be used to estimate model parameters based on historical information, or in combination with the database, used to test model hypotheses and statistical inferences. Both a multiple (and step-wise) linear regression (MLR) and an adaptive process regression (APR) package are available with LAMDA. Transformation capabilities and all standard statistics are calculated. The complexity of the problem is only limited by the capacity of the microcomputer hardware. The APR package focuses on fuzzy-set and state-space analyses. With it, the best choice of several possible alternative hypotheses can be evaluated. These packages are developed in cooperation with Ohio State University faculty.

LAMDA has been adopted by the American Public Power Association as the the host decision support system (DSS) for financial, systems, regulatory and enginering analyses. The Electric Power Research Institute is currently considering LAMDA as one of the DSS packages it recommends. Several utilities and utility commissions currently use LAMDA with System Dynamics applications on a daily basis. (Backus 1985) All clients whose original applications were developed with DYNAMO have since converted to LAMDA for less than the licensing cost of DYNAMO. Clients are also converting their DSS applications to LAMDA in order to take advantage of its integration capabilities.

SPECIFICS FOR SYSTEM DYNAMICS APPLICATIONS

LAMDA forces structure upon the System Dynamics model. It requires that all variables be defined before they are used. Variables are also "weakly typed" as real, integers, money, time, strings or code. Because LAMDA allows a model to be built from separately compiled procedures, it forces the modeler to think of the system as interconnected subsystems. The behavior of each subsystem can be tested separately before integration into the larger model. The input, outputs and interfaces of each part must be recognized. LAMDA will allow simulation with a haphazard assembly of equations but more easily simulates an orderly progression of causal thought as exemplified by the causal flow of equations in causally related procedures. This relationship is analogous to sentences forming related paragraphs as a complete explanation of a process. Carrying this analogy further, LAMDA has a syntax composed of only nouns and verbs. LAMDA is structured to DO, SELECT, AUDIT, EDIT, COMPARE, DEFINE, READ, WRITE, PLOT, etc. nouns (options, data, variables, and equations).

All of the basic DYNAMO macros (SMOOTH, DELAY, STEP,CLIP, etc.) are implemented in LAMDA in the standard syntax. LAMDA, however, uses no ".K"s or equation typing. All functions such as MAX, ABS, or SUM are performed on vectors or arrays. For example, the equation:

PEAK(CLASS)=MAX(HOUR)(DEMAND(CLASS,HOUR))

will find the maximum energy demand by rate classification (CLASS) from hourly demand data. The functions IMAX and IMIN are identical to the DYNAMO MAX and MIN functions.

LAMDA does not order the equations as does DYNAMO. This drawback is offset by the extensive conditional branching structure capabilities LAMDA provided at the expense of "self-ordering." The value of this branching capability will become clear later. In terms of equation ordering, LAMDA requires a separate initialization procedure for all levels. This procedure can be as simple as assigning initial values from the database to the levels, to very complicated equilibrium condition setting (even if the initial condition requires the solution to simultaneous equations). Equations need only be ordered within the subsections of procedures. The normal style of writing the level equation first, then the rate equations, then the auxilaries are simply reversed. The logical segregation of the model into subsystems (procedures) almost automatically insures that information is calculated before it is used. The modeler must, nonetheless, manually deal with the "order of calculation" concerns. The need for all equations to use the same "level value" for a given time period is achieved by actually having two values for the level. (Similar to the concept of a ".J" and a ".K" value in DYNAMO.) One value is always used on the right side of the equation even if a new value has been calculated on the left side of the defining equation. At the end of a time iteration the "right side" value is set equal to the "left side" value.

As noted earlier, LAMDA operates on each equation as an array process. There is seldom the need for the equivalent of a DYNAMO "FOR" statement in LAMDA. A level equation:

```
POP=POP+DT*(BIRTHS-DEATHS+NETMIGR)
```

would be performed over all indices automatically, unless a set of indices were selected. For example, the BIRTH equation could be preceded by a:

```
SELECT AGE (TEENAGER,ADULT),SEX (FEMALE)
BIRTHS=POP*FERTILITY
```

BIRTHS would only be calculated from females for these two age groups (and all remaining dimensional indices). This selection remains in effect until a new select statement is encountered. The statement, "SELECT AGE*,SEX*", would reset the model to use all age and sex indices. Note that only the form of a level equation distinguishes it from all others. There is no distinction between rates and auxiliaries. Constants and tables are defined in the same manner as DYNAMO and/or stored on the database.

Table functions are defined differently in LAMDA but can be of any dimension up to 10. The name of the table function must be defined. For instance if fertility is a function of food per capita (FPC) then a function called TABFR could be developed that relates FPC to FERTILITY as:

```
FERTILITY=TABFR(FPC)
```

The user defines both the x and "y" values of the variable. If the function were just FERTILITY as a function of FPC then the range of FPC could be from 0 to 10000 calories per day. The intervals between steps don not need to be constant. Thus the user can increase the resolution in critical areas. The FPC range table could be FPCR. The corresponding FERTILITY range table (the output) could be FRT. In LAMDA the "set-up" of the table would be:

```
READ FPCT
0 500 750 1000 1100 1200 1500 2000 2500 3000 10000
READ FRT
0 1E-5 2E-5 1E-4 2E-4 5E-4 7.5E-3 1E-2 3E-2 5E-2 5E-2
```

to change one entry in the table interactively (in rerun mode) the user could type "FRT(6)=3E-4". To change several entries the user could type a SELECT statement and then "READ FRT." The range can also be changed interactively.

Occasionally models must be developed which contain a wide range of time constants. The short-term behavior may come to equilibrium quickly relative to the long-term mechanisms. In LAMDA these short-term phenomena can be treated as simultaneous equations which are solved for their equilibrium solution. The user simply defines a set of equations as a "system." Whenever the "system" is called, LAMDA will solve the equilibrium conditions for all output variables, based on the input variables.

LAMDA has extensive conditional capabilities. These are primarily of the form:

```
DO IF condition
statement(s).
ELSE condition
statement(s)
ELSE condition
statement(s)
ELSE ...
.
.
END
```

LAMDA supports comparison of both numerical values and alphanumeric strings (characters). Conditional statements can involve multiple logical operators connected by AND, OR, and NOT. This comparison can be used for "clipping" in different structures or policies at the user's discretion, performing unique operations on selected variables or interpreting output to tell the user what is happening. For example, conditional statements can be used for error checking, to show when a variable has exceeded critical values, or that oscillations are occuring and spreading to other subsystems. LAMDA can also request user input via the "ASK" statement whenever necessary.

There is also a SELECT OPTION statement which is quite similar to the DYNAMO OPTION statment. It allows the user to set-up the printer, change the color of the video output, set margins, cursor/printer head location, generate page ejects, and set graphics resolution.

As yet, every known use of DYNAMO can be duplicated and enhanced using LAMDA. LAMDA's higher-level structured approach also makes it flexible enough to perform most functions available to more general languages such as PASCAL with minimal conceptual or programming difficulty. LAMDA has many other capabilities not presented here. A complete user's manual and demo diskette are available.

An example of a very simple but complete model is shown below. Other examples are available free from the authors.

```
*                              *
*      *    *    *    *       *
*   Simple Population Model    *
*             *                *
*    LAMDA Example Program     *
*      *    *    *    *       *
*                              *
**********
DEFINE SET
**********
*
YEAR(40)
RANGE(11)
*
```

```
END DEFINE SET
*
***************
DEFINE VARIABLE
***************
*
BR(YEAR)     'Birth Rate (Persons/Yr)'                          ,TYPE=REAL(8,10)
DR(YEAR)     'Death Rate (Persons/Yr)'                          ,TYPE=REAL(8,10)
FIR(YEAR)    'Food Increase Rate (Calories/Yr)'                 ,TYPE=REAL(8,10)
FOOD(YEAR)   'Food (Calories)'                                  ,TYPE=REAL(8,10)
FOODI        'Initial Food (Calories)'                          ,TYPE=REAL(8,10)
FOODV        'Food Value(Calories)'                             ,TYPE=REAL(8,10)
FPC(YEAR)    'Food per Capita (Calories/Person)'                ,TYPE=REAL(8,10)
FPCR(RANGE)  'FPC Range per Capita (Calories/Person)',TYPE=REAL(8,10)
FR(YEAR)     'Fertility Rate ((Persons/Person)/Yr)'             ,TYPE=REAL(8,10)
FRR(YEAR)    'Food Regen. Rate ((Cal./Cal.)/Yr)'                ,TYPE=REAL(8,10)
FRT(RANGE)   'FR Table ((Persons/Person)/Yr)'                   ,TYPE=REAL(8,10)
FUR(YEAR)    'Food Usage Rate (Calories/Yr)'                    ,TYPE=REAL(8,10)
MR           'Mortality Rate ((Persons/Person)/Yr)'             ,TYPE=REAL(8,10)
POP(YEAR)    'Population (Persons)'                             ,TYPE=REAL(8,10)
POPI         'Initial Population (Persons)'                     ,TYPE=REAL(8,10)
POPV         'Population Value (Persons)'                       ,TYPE=REAL(8,10)
YRV(YEAR)    'Year'                                             ,TYPE=INTEGER(2)
*
END DEFINE VARIABLE
*
DEFINE RELATION
TIME(YEAR,YRV)
END DEFINE RELATION
*
DEFINE FUNCTION
TABFR(FRT,FPCR)
END DEFINE FUNCTION
*
DT=.25
ENDING=40
YRV(1)=1
SELECT YEAR(1-39)
YRV(I+1)=YRV(I)+1
SELECT YEAR*
*
READ FPCR
0 500 750 1000 1100 1200 1500 2000 2500 3000 10000
READ FRT
0 1E-5 2E-5 1E-4 2E-4 5E-4 7.5E-4 1E-3 3E-3 5E-3 5E-3
*
FRR=.05
FOODI=1E9
MR=.015
POPI=10000
*
```

```
************************
DEFINE PROCEDURE INITIAL
************************
*
TIME=0
FOODV=FOODI
POPV=POPI
END PROCEDURE INITIAL
*
*********************
DEFINE PROCEDURE RUN
*********************
*
DO UNTIL TIME GT ENDING
SELECT YEAR(TIME)
*
* FOOD
*
FOOD=FOODV
FPC=FOOD/POP
FUR=IMIN(FPC,3500)*POP
FIR=FOOD*FRR
FOODV=FOODV+DT*(FIR-FUR)
*
* POPULATION
*
POP=POPV
FR=TABFR(FPC)
BR=POP*FR
DR=POP*MR
POPV=POPV+DT*(BR-DR)
*
WRITE (TIME,POP,FOOD)
TIME=TIME+DT
END DO UNTIL
SELECT YEAR*
SELECT GRAPHICS=HIGH
PLOT LINES (YRV,FOOD,POP)
PLOT LINES (FOOD,POP)
*
END PROCEDURE RUN
*
INITIAL
RUN
WRITE ("Rerun Changes?")
*
```

REFERENCES

Amlin, J. S., A. Ford and G. A. Backus. "A Practical Approach to Sensitivity Testing of System Dynamics Models," Proceedings of the 1983 International System Dynamics Conference, Chestnut Hill, MA, July 27-30, 1983.

Backus, G.A. and J. S. Amlin. "Integrated Utility Planning and Consumer Response Model." National Regulatory Research Institute Papers, Columbus, OH. September 14-17, 1983

Amlin, J. S. and G. A. Backus. "Interactive Sensitivity/Confidence Analysis of Large Scale Simulation Models on Microcomputers." 1985 Summer Computer Simulation Conference, Chicago, IL, July 22-26, 1985

Backus, G.A. and J. S. Amlin. "Comprehensive Corporate Policy Planning Using Large-Scale Simulation on Microcomputers." 1985 Summer Computer Simulation Conference, Chicago, IL, July 22-26, 1985