A COMPLETE INTERACTIVE SIMULATION ENVIRONMENT GPSS/360-NORDEN

Julian Reitman, Donald Ingerman*, Jerry Katzke,
Jon Shapiro, Kenneth Simon**, Burton Smith

Norden Division
United Aircraft Corporation
Norwalk, Connecticut 06856

Abstract

Norden's expanded and interactive version of GPSS/360 has been extensively improved through new data entry and storage capabilities, a new report generator, selective output display, and HELP blocks to enable model manipulation and to provide a "window" to view the simulation during its progress.

The data entry and manipulation system allows the user to input and display matrix savevalues through an IBM 2250 Display Unit. Titles may be placed on rows and columns to simplify the location of data items. A less elaborate system was developed for remote terminal and batch mode data entry.

The user can witness the progress of the simulation during the actual model execution. He then has the option to stop the simulation at any time, change parameters controlling operation of the model, and resume running of the model. This interactive feature opens vast new horizons for the world of simulation since with the appropriate equipment moving displays can be produced.

BACKGROUND ON DEVELOPMENT AND REQUIREMENTS OF SIMULATION LANGUAGES

Simulation is the newest and most conceptually sophisticated Operations Research technique. It must operate along two opposing major fronts. These are involvement in technique development as contrasted with application to specific problem areas. Unfortunately, the parallel technique development of simulation concepts and sufficiently powerful simulation languages has detracted from the building up of interfaces with those fields of potential applications. Furthermore, the resultant conflict of this duality has inhibited adequate recognition of simulation as evidenced by the lack of noticeable experimentation and application by industrial O.R. departments, university O.R. departments, computer mainframers, and management.

Responsibility for the lack of acceptance thus caused lies largely within the field of simulation. Though long-term evolution has resulted in the existence of the necessary techniques, the process has been too slow, and has left too many discouragements in its path. This is due to the fact that special interests internally have not devoted the time and resources necessary for systematic development. This evolution, resulting in simulation languages, has chronologically occurred as follows:

- special purpose languages written for unique applications, such as network and economic simulators.
- semi-general languages composed of simulation subroutines written in programming languages such as Fortran or Algol.
- truly general purpose user languages, such as GPSS, Simscript and Simula.
- creation of a simulation environment exhibiting real world capabilities enabling a system expert to create, modify and experiment dynamically with a model of his system in a manner consistent with the normal operation of the system.

Orginally, the general purpose simulation languages were not very powerful due to lack of adaptability to problem areas and poor com-

*Donald Ingerman, Currently with Bell Labs., New Brunswick, New Jersey
**Kenneth Simon, Currently with IBM, White Plains, New York

pilers. Attempts to do the simulation in then currently available languages would end up in failure or exasperation. Frequently, a person undertaking a simulation modeling effort could more quickly obtain a useful result by developing a specialized simulator. However, this occurred at high cost and slow production of answers. The high percent of failure could be attributed to the combination of modeler ineptness, inadequate funds, and lengthy delays in obtaining results.

Fortunately evolution continued and useful general purpose languages have emerged. Certainly some specialized simulators will always be needed for specialized application areas, such as computer system simulations. But as the general purpose languages achieve greater capability, it becomes expedient to make use of them. This is due to the excessive manpower and financial resources required for a simulation done with inadequate tools. The poorer a language is, the more effort required to prepare a working model from the design. And as simulation usage increases, greater model building capability is needed to enable production of models in greater quantities.

Even though improvements have been shown in the three most widely used general purpose simulators (GPSS, Simscript, Simula), it still has been apart from the work in systems analysis concepts, and application of simulation concepts. There is a tradeoff between technical capabilities and application considerations that has to be made in language development. However, there has been an imbalance favoring technical improvements. Included in this category would be increasing the ability of the programmer/user to specify his own data or language structures. But such added flexibilities increase modeling time, and thus cost.

Simulation has a reputation for being one of the most expensive Operations Research techniques. Even though the cost reductions and increases in efficiency possible through simulation studies can be great, they are often indirectly achieved and not easily verifiable. Thus, confidence in the technique follows completion of a successful project. However, approval for initial funding requires that some amount of confidence must exist prior to the project. It is therefore necessary to show the potential user a proposal which he can understand, and perceive its practicality. Relating this to language development, a potential user will be more confident in attempting to commit himself to a language that he can visualize and interpret than one in which he has to provide the entire structure. He will likewise be interested in the specific simulation techniques offering greatest reduction in modeling and study expenses.

Thus, cost is an ever present nemesis for simulation. It becomes evident that perhaps emphasis on decreasing project time and cost is most important. Methods to accomplish this include:

- decreasing computer run time
- moving computer usage into the realm of professionals in other fields, reducing the need to train computer personnel in these fields.
- improving computer turnaround time, and thus total project elapsed time.
- isolating modeler from simulation housekeeping chores.
- removing modeler from formats and programming techniques related to computer operation.
- improving debugging information--reduce debugging time.

Of the numerous simulation languages available, GPSS comes closest to achieving these goals. Thus, GPSS was selected as a language base for construction of the desired simulation environment.

Some of the advantages of GPSS are:

- GPSS is forgiving of poor understanding of large problems. Large system problems, as previously stated, involve significant costs to define, let alone guarantee success. Large scale models may be constructed using a "top-down" approach through the use of a coarse overview model to be later detailed as needed through patches and modifications.
- GPSS, being a "flow-chart" language, is easy to transcribe from a system logic flow chart.
- Its structured format allows for quick replacement or modification of code, and easy reference to the operations of the code.
- Being an interpretive language, diagnostic messages relate to GPSS instruction blocks and not compiler generated object code.
- In addition, the interpreter allows statistics to be kept on each line of code, which is not practical in a compiler language.

However, design of the environment must reflect several basic functions common to simulation studies which are not fully considered in GPSS. These include:

- fund acquisition
- problem definition
- study supervision
- model creation
- model usage and experimentation
- output data application

GPSS/360-NORDEN was designed to create an environment serving these functions, whether they are performed by one or many people. Thus, it provides a precise course for the successful application of simulation by increasing the probability of fulfilling these functions.

Goals for the simulation environment must also be based on the characteristics of large, and usually poorly defined system problems. Studies of these problems involve:

- huge quantities of data, requiring the ability to quickly initialize and change the data.
- computer system limitations
- long execution times
- need for flexible report generator output to enable output formating for all classes of users.
- need for interaction between modeler, his model and the computer, to cut down turnaround time.
- desirability of enabling a team of system specialists to interact with the system model as they would interact with the real system.

It remains to be shown what additions and modifications were made in GPSS/360-NORDEN to converge on the desired environment.

GPSS/360-NORDEN

Descriptions of GPSS/360-NORDEN have been made in previous papers.[1] Since the most recent of these papers, development has continued and additional concepts have been implemented. The goal of making simulation techniques available for more applications on a more economical basis will be discussed in terms of both previous and new capabilities. Emphasis will be given to the reasons for following these directions as derived from our simulation efforts in production, weapon system design, test equipment utilization, finances, resource allocation, transportation, total weapon system applications.[2]

Norden's convergence towards the desired simulation environment will be elaborated on in the following order:

- man-machine interaction
- data storage
- data handling
- core requirements
- output report generator NRG (Norden Report Generator)

MAN-MACHINE INTERACTION

An important previously reported feature was the ability to construct, edit, debug input and assembly errors, execute, and partially view model output using the IBM 2250 Display Unit. Extensions in this area have followed two courses. First the continuation of interactive feature development of the 2250, and second, conversion of a subset of the 2250 feature to a time-sharing environment through a 2741 typewriter, CRT display or teletype terminal. Changes in 2250 features have been directed at increasing data set manipulation capabilities. Previous capabilities included INSERT, DELETE and REPLACE functions. Recent additions include cursor positioning, MOVE, COPY and ALTER capabilities. The preposition of the cursor operates during the insertion or replacement of consecutive lines, and is indicated by the user light pen detecting on the desired cursor position. The MOVE and COPY function keys allow the modeler to MOVE or COPY sections of code or subroutine simultaneously from one location in the model to another. The ALTER function allows the modeler to similarly change consecutive lines of code simultaneously. If thirty components of a radar system are defined by unique matrices of the same size, then one ALTER command could change the size defined on all thirty MATRIX cards. Figure A shows the options available through the function keys. Figure B shows the source code display during INSERT mode. Figure C illustrates the displaying of errors, which lead to display of the faulty source code by light pen detection on the line number.

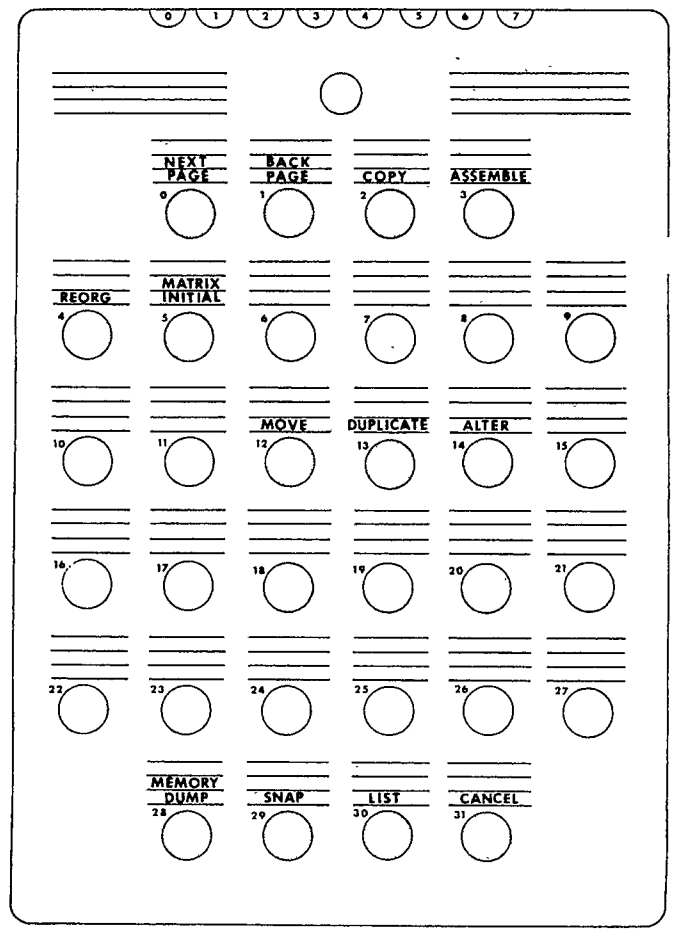

FIGURE A - TEMPLATE - EDIT

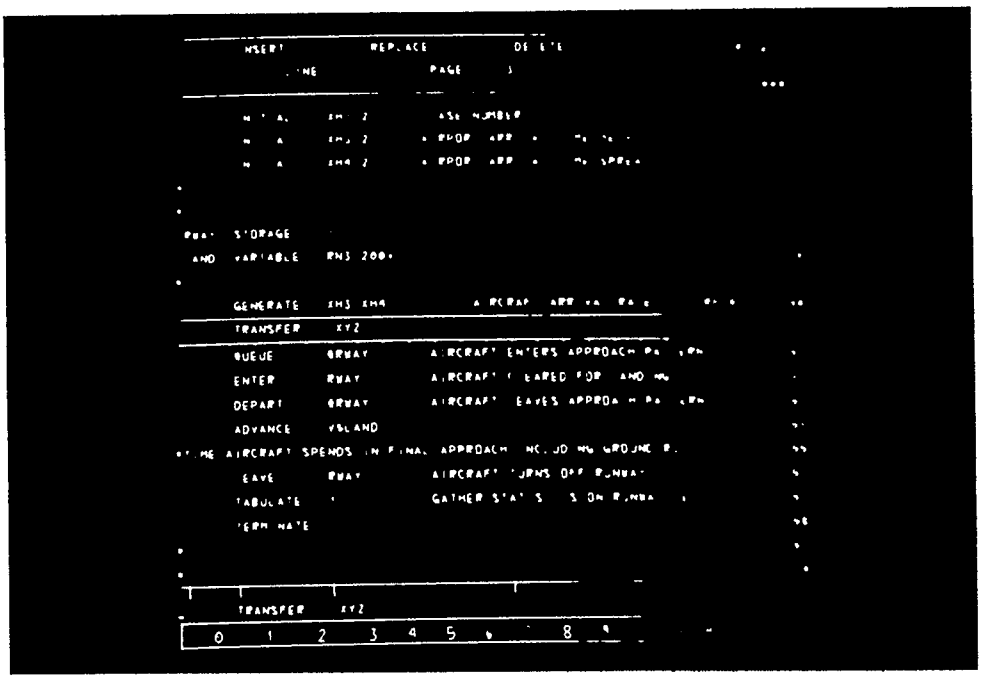

FIGURE B - EDIT MODE - INSERTION

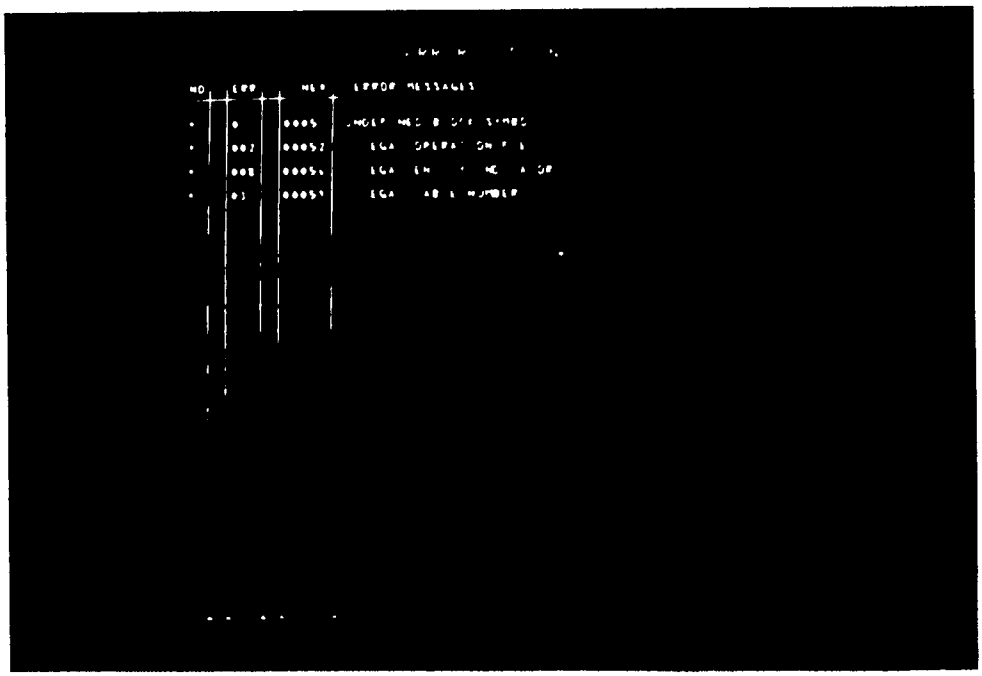

FIGURE C - ASSEMBLY ERROR LISTING

While displayed output is still limited to TEXT, COMMENT cards and graphic output, one can now not only flip forward through the report generator "pages", but also return to page 1 of the report generator, return to the source program in edit mode, or continue execution if the report generator was arrived at from SNAP (interruption of execution to check statistics). Figure D shows these options.

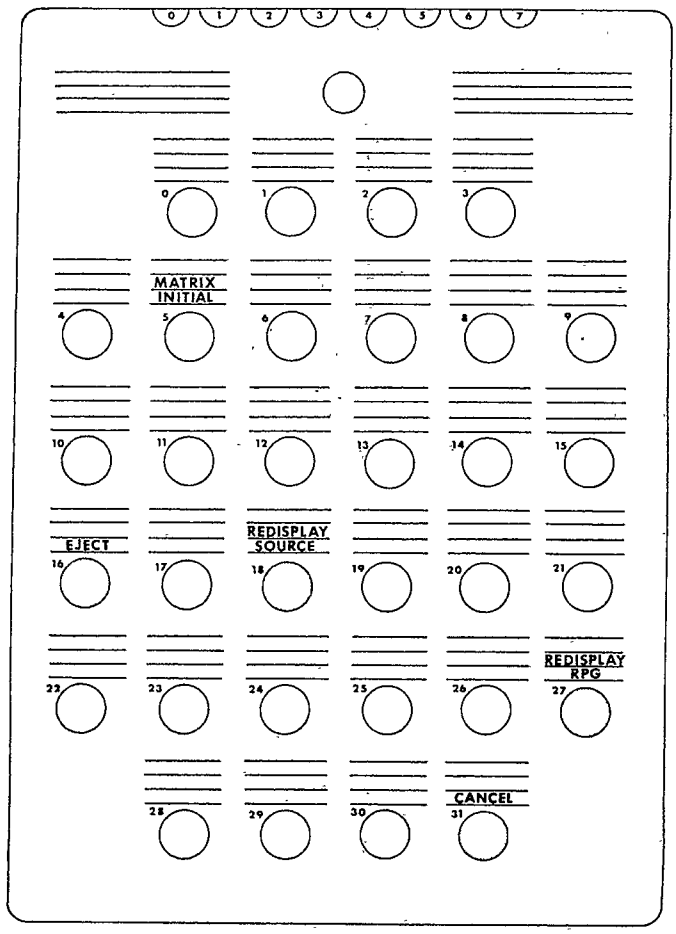

FIGURE D - REPORT GENERATOR OPTIONS

The two newest improvements in interactive capabilities concern output display. The new NRG is capable of full viewing on the 2250, not just the few cards indicated above for standard report generator. An output package is in use on the 2250 Display Unit and 2741 typewriter terminal that allows "menu" selection and display of any GPSS SNA at the user's request. This is entered after model execution, or during a SNAP. Thus, all system characteristics are quickly accessible during and after execution.

The behavior of a model during simulated time may be of great interest to the user. Therefore, it is obvious that if an on-line terminal is being used, the greatest utilization can be achieved only by allowing the user to be constantly in touch with the simulation during model execution. There are several applications for this capability. First, the termination of long simulations which are not performing correctly. Second, the ability to understand the dynamic characteristics of the model. Third, entering data and/or commands to the model to guide its performance. Norden has developed 27 generic HELP blocks which allow the user to pass values generated during the simulation to assembler language routines which display the results dynamically. Naturally, the display capabilities on a graphics terminal greatly outclass those of a typewriter terminal because of the ability to create pictures with vectors, and to show entire pictures at one time. The complexity of designing pictorial representations of a model has been practically eliminated through the usage of the set of graphic subprograms imbedded in GPSS/360-NORDEN through the HELP blocks.

The 2250 implementation allows design of pictures with orientation towards the system being simulated. Airport runways (Figure E), aircraft carrier decks, and financial graphs can be displayed with equal ease. These pictures may then be dynamically altered either in part or whole as the model execution progresses. The modeler can symbolically display any model occurrences in a manner of his choosing. Characters and numbers may be displayed along with, or instead of, the computer generated vector pictures. Interaction is achieved with the model by allowing the user to enter commands and/or data to the model with the light pen (Figure F), function keys, or keyboard. Rather than just allowing the user to interact with the simulation, a new usage has grown out of these capabilities. The simulation model may be used as a subprogram in the GPSS/360-NORDEN program. For example, the user can interact with the program to determine and specify certain system input parameters. Once satisfied with the model specifications, the user may initiate entrance into the simulation portion of the program. An example of this would be the user's selection of an evaluation of a company's financial activities. The user could specify accounts, calendar intervals, view the results, and repeat for different conditions.

In the 2741 typewriter terminal environment, characteristics and numbers may be outputted by the 2741, and are displayed one line at a time. Input is accomplished through the keyboard, and both characters and numbers are accepted. Input and output can be combined on the same printed line. This environment allows the desired interaction with the model. For example, the simulation of financial activities could be implemented interactively as mentioned for the 2250, expect for display of the financial graphs.

DATA STORAGE

The library feature, previously existing in GPSS/360-NORDEN, enables the storage of GPSS matrix savevalues (data arrays) on disk, using their symbolic names as keys. Thus, once matrix savevalues have been initialized they can be stored and used in the current run or in any subsequent runs of this or any other model. The user can bring into core a matrix

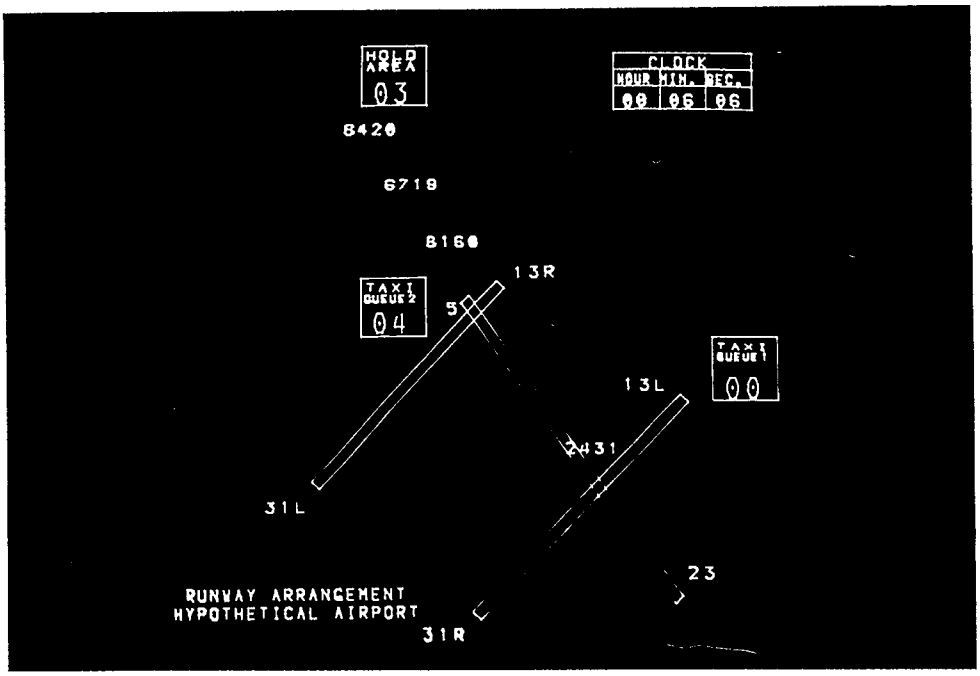

FIGURE E - AIRCRAFT USING RUNWAY 5 FOR TAKEOFF AND LANDING

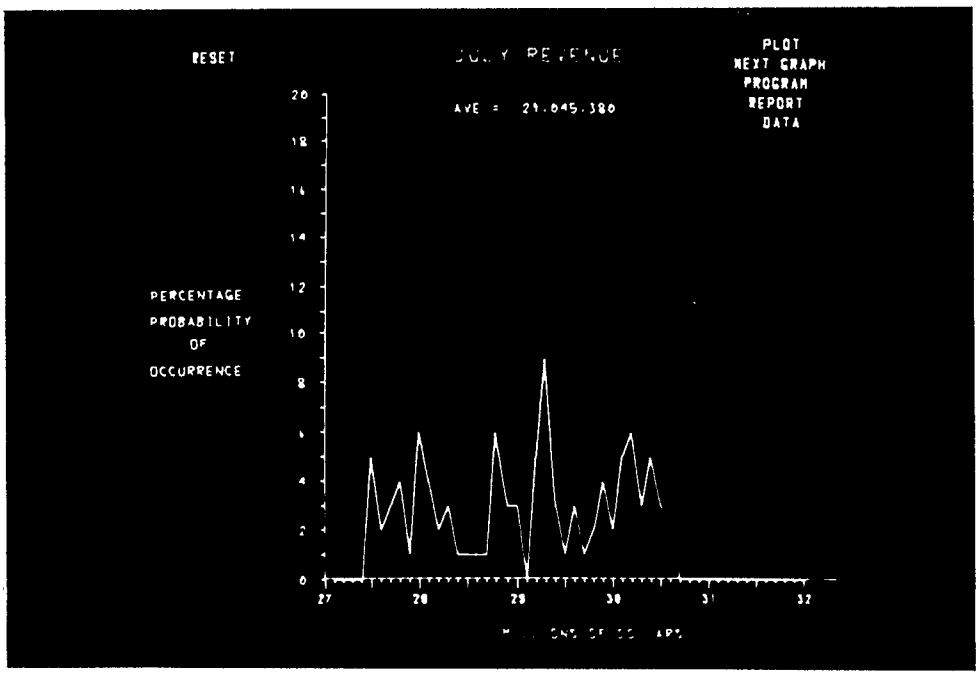

FIGURE F - INTERACTIVE GRAPHIC DISPLAY UNDER LIGHT PEN CONTROL

from a specified library, use and/or modify it during execution, and replace the old version on disk. Thus, a user can have stored on disk a data bank as the base of his simulation environment. The recent improvements allow the size (m x n) of a matrix stored on disk to be changed, either temporarily or permanently, by a model or by a separate program. Furthermore, every matrix (data) library can have a matrix title library associated with it and printed out when desired. Thus, a matrix can exist with unique titles for each of its rows and columns, as shown in Figure G.

DATA HANDLING

Though data in a matrix can be added or changed through a model, this becomes impractical when large amounts of data are used. Routines have been developed for use on the 2250 and 2741 terminals, and also for batch-mode processing. Because of the tie-in between the model and the data, MINITIAL (the 2250 implementation) is an effective data entry technique for the user. It may be called from the edit routine, during model execution, from the report generator display, or may be used independently. Large amounts of data can be quickly changed. Thus, if matrices are used to define a simulated system, the system definition can be cheaply and conveniently altered.

The 2741 terminal and batch mode versions have been provided with similar "data-entry" routines. The user specifies the data and/or title changes he wishes to make in what is basically a simplified, but more powerful, GPSS INITIAL card format. The data-entry routine is part of an extensive utility package which, through simple instructions, allows manipulation of data, titles and entire matrices between other matrices and libraries. The utility package can also be used with matrix libraries maintained through MINITIAL on the 2250.

The efforts in data manipulation have not been applied solely to non-execution areas. Movement of data in any model can often require extensive block usage and execution time. This may be an unfortunate circumstance of the basic GPSS conception. However, Norden has developed HELP block routines to solve much of this problem. Norden's data manipulation HELP blocks are written in assembler language, and greatly reduce execution time of certain operations.

The HELP PLOAD (Parameter LOAD) block rapidly transfers data from a matrix savevalue to transaction parameters. There are two main usages of this feature. One, it facilitates indirect addressing in a model. Two, it renders efficient the operation of transferring data into parameters before loading onto JOBTAPES. In fact, it was found that due to its speed it seldom becomes necessary to use a JOBTAPE, as transactions can be quickly loaded with repetitive data during each occurrence of model execution. The complement to PLOAD is HELP PSAVE (Parameter SAVE). This block transfers data in the exact opposite manner of PLOAD; that is, from transaction parameters to a matrix savevalue.

Increased dynamic definition of GPSS functions is the purpose of HELP FUNCUT (FUNCtion

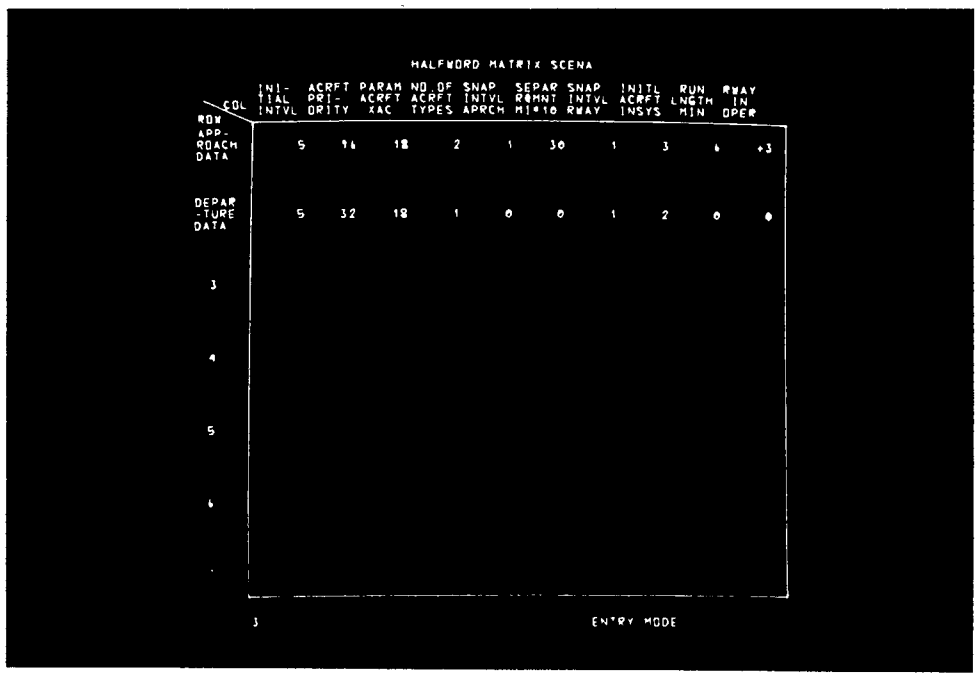

FIGURE G - AIRPORT MODEL DATA MATRIX WITH TITLES

UTilization). Normally, GPSS functions may have SNA Y values, but the X values must be constants. This HELP block allows replacement of X and/or Y values in a function by values in a matrix savevalue. Thus, a function's domain and range may be entirely controlled by model dynamic characteristics.

CORE REQUIREMENTS

Due to the increasing complexity and size of models, and increasing usage of "partitions" and "regions" within core, more often than ever before it becomes a problem to construct models which can fit into the available core. Experience shows that the main factors determining the amount of core required are blocks, matrix savevalues and transactions. Thus, there is greater need than ever for the previously reported Load Section feature. This allows temporary placement on disk during model execution of matrix savevalues and blocks. Using this overlay feature certain blocks and matrix savevalues remain core resident while the others, previously selected by the user and placed on disk are brought into an overlay area of core. While proper block and matrix savevalue organization insures minimal I/O time, the only required user statements to use Load Sections are a few new REALLOCATE cards. The user may reference any block, block SNA, or matrix SNA whether it is in core or on disk. Thus, the load section operations are entirely transparent to him.

OUTPUT REPORT GENERATOR

Output is perhaps the most underrated function in creating a simulation environment. It is certainly the most prominent in contact with the customer. One most cumbersome and inflexible area of GPSS is the special report generator (Output Editor). Although it was previously mentioned that a highly structured language is necessary in simulation, such a language style can be inflexible for user readable output. Therefore, Norden has designed and implemented an improved PL/I style, free-format GPSS report generator. The user may use either the normal GPSS report generator or the Norden Report Generator. This new feature makes available all GPSS entities (SNAs), and allows greater control of output format with less programming and coding effort. Instead of writing one line of code for each line of output, it is now possible to use loops, conditional statements, repeated headings, and other report oriented controls. However, complicated format control as is found in Fortran is not required of the user. An English-keyword instruction set enables a non-programmer to quickly develop competence in the full usage of the Norden Report Generator features.

ADDITIONAL FEATURES

A significant new feature of GPSS/NORDEN solves the problem of obtaining output from a model which appears to be looping. Normally, a modeler would hesitate to cancel an apparently looping model for fear of losing all output, and not being able to determine where, if at all, a loop occurred. Depressing a specific function key on the 2250 now causes a SNAP to occur, which produces a full GPSS statistical printout. The user can then elect either to continue execution or terminate. This feature also allows interim examination of long, complex simulations, and has also helped detect subtle logic errors.

Additional features to support the 2250 environment have been developed. The display on the 2250 may be entirely plotted on a Cal-Comp plotter through special HELP blocks. During execution, the user may transfer to the MINITIAL data routine, view or change any matrix in his library, and return to execution to use the new data.

The most successful method of approaching the desired simulation environment is through usage of a graphic terminal with 2250 capabilities, such as easy cursor control and vector and character capabilities. Norden has strived to make the environment available to 2741 terminal and batch-mode users as close to that available to the graphic terminal user. However, it must be realized that graphic hardware is required for the most effective implementation. Of course, success with a more confined environment may lead to eventual acquisition of a better one.

APPLICATION OF ENVIRONMENT

The new and old capabilities of GPSS/360-NORDEN have been described. To show how their utilization constructs the desired environment a modeler's usage of this simulation environment will be described. Figure H shows a flow chart of the process this usage will follow. The system being constructed is a financial risk forecasting model. The numbered steps below correspond to the flow chart.

1) Determine main financial items and their effect on revenue, expense and profit values.

2) Code model to coordinate and accumulate revenue, expense and profit computations over a time period.

3) Initial modeling effort using 2250 Environment.

   a) Switch to MINITIAL, create and store matrices of desired names and sizes (Figure I), with financial data and risk levels.

   b) Switch mode to construct model sections from code and previously existing elements.

   c) Now, or later, Load Sections may be required to fit model into available core. Add necessary REALLOCATE cards.

   d) Assemble model. Assembly and input error sheets displayed as needed. Iterative interaction between model source code and assembly and input error sheets enables correction of errors.

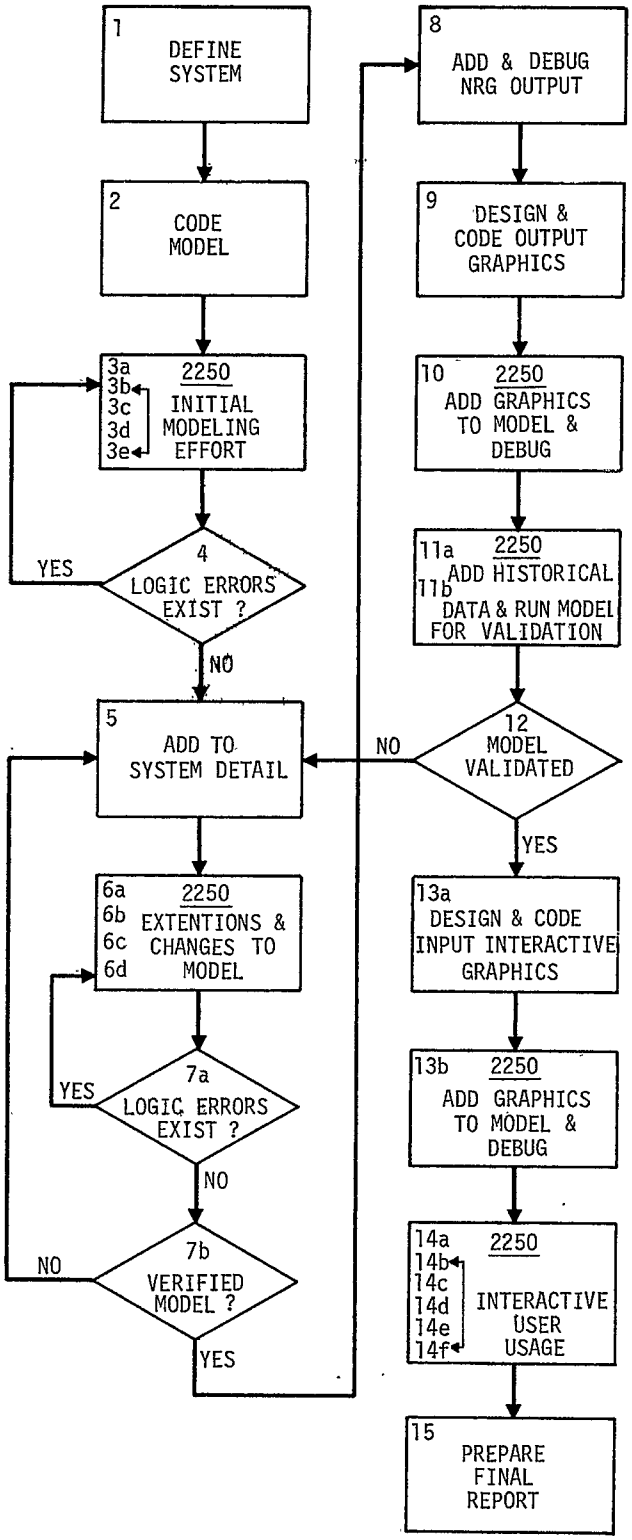

FIGURE H - FLOW CHART FOR GPSS/360-NORDEN USAGE

e) Successful assembly, and execution to completion or running error. For running error return to step 3b, otherwise, continue to step 4.

4) Use hard copy of model output to verify model results. Return to step 3b to correct logic errors. For extension or different detailing of the model continue to step 5.

5) If model verification indicates insufficient detail, extend system definition downward to greater detail. Further breakdown and/or change of major financial items into subaccounts may be required.

6) Extensions and changes to the model using the 2250 environment.

   a) Transfer to Utility mode, copy and rename, matrices to maintain historical copies.

   b) Transfer to MINITIAL, change size of matrices, add subaccounts' test data, and titles.

   c) In MINITIAL, print matrices for record.

   d) The model construction and debugging procedure described in steps 3b, d and e should be utilized with additions from step 5.

7) Use hard copy of model output to verify model results.

   a) Return to step 6d to correct logic errors.

   b) If further model detail is required, return to step 5.

8) Write a report generator using Norden Report Generator for periodic revenue, expense and profit results. Use 2250 environment to add NRG, run model with NRG, and terminate. If NRG errors appear on hard copy, repeat this process. It is possible to run and debug NRG independently, thus saving model run time.

9) Design and code financial risk graphs for 2250 display using graphic HELP blocks.

10) Using the 2250 Environment, add graphs to model and debug using displayed Norden HELP block error messages.

11) Validation run using 2250 Environment.

   a) Transfer to MINITIAL, add historical data for validation run, print matrices.

   b) Switch mode, Assemble model. Terminate on 2250.

12) From hard copy determine if model reflects historical financial performance. If not, return to step 5. If so, continue.

13) Construction of user environment interactive procedures.

a) Design and code input capability for model to inquire and accept user instructions as to time periods, intervals and financial categories to be displayed.

b) On 2250 Environment, add interactive input capability to model and debug using HELP block error messages (Figure F).

14) Interactive user usage on 2250 Environment.

a) Assemble model to enter interactive execution.

b) Select financial risk graphs and resultant calculations desired (Figure J), and view their values through displayed graphs, NRG and SNA menu selection.

c) Plot displayed graphs for hard copy record.

d) Is model experimentation finished? If so, terminate 2250 and go to step 15. Otherwise, continue.

e) Transfer, while in execution, to MINITIAL. Enter new experimentation data.

f) Print matrices. Return to step 14b.

15) Prepare final report from matrix printouts, NRG output, plotted graphs. Flow chart of model for documentation can easily be made due to GPSS structured format. If desired, model can be SAVEd, and different NRG report generators used on the READ to obtain different reports for different users (i.e., Income Statements, Balance sheets).

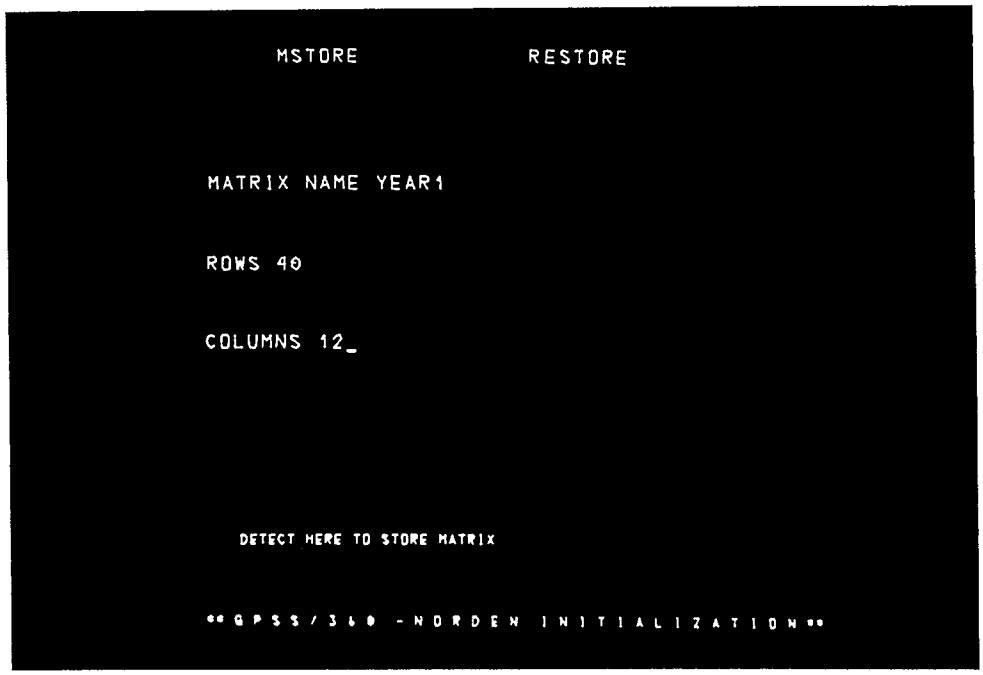

FIGURE I - MATRIX SPECIFICATION FOR MINITIAL

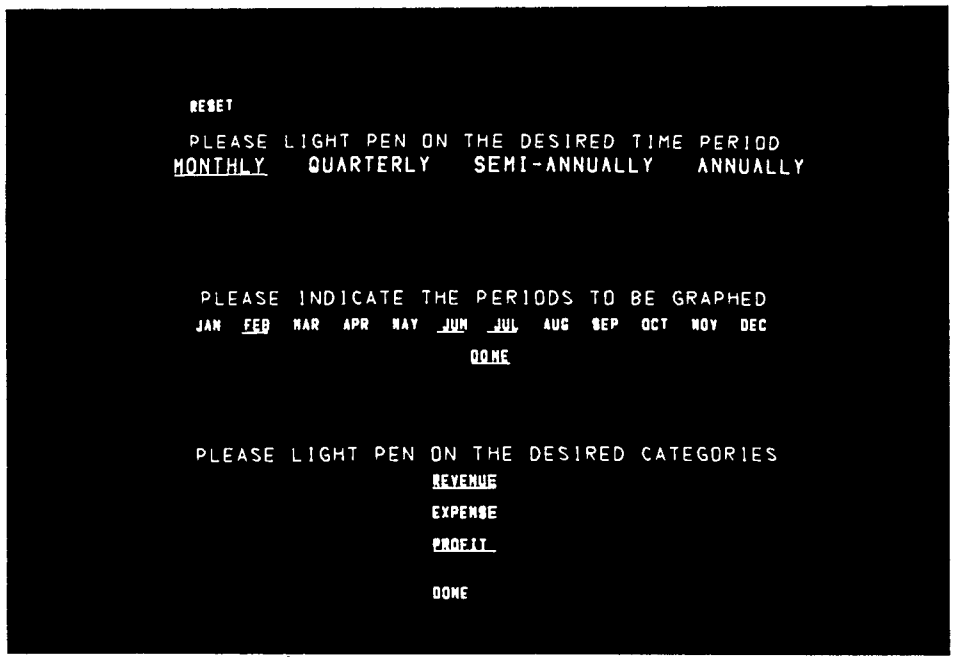

FIGURE J - INTERACTIVE GRAPHICS FROM GPSS/360-NORDEN PROGRAM

REFERENCES

1. William A. Walde, David Eig and Sherman R. Hunter, "GPSS/360-NORDEN, An Improved System Analysis Tool", IEEE Transactions on Systems Science and Cybernetics, November 1968.

   Sherman Hunter and Julian Reitman, "GPSS/360-NORDEN, A Partial Conversational GPSS", Digest of The Second Conference on Applications of Simulation, December 1968.

   User's Guide to Conversational GPSS (GPSS/360-NORDEN), Norden Report #4269 R 0003, December 1969.

2. J. P. Carstens, R. C. Baxter and Julian Reitman, "Economic Models for Rail Systems", IEEE Transactions on Systems Science and Cybernetics, December 1966.

   Shipboard Support System Computer Logic Model, Norden Report #1212 R 0024, August 1968.

   IEEE Transactions in Systems Science and Cybernetics, November 1968

   Richard C. Baxter, "Prediction of a Naval Vessel's Performance"

   W. Wallace Fenn, "An Antisubmarine Warfare Model for Convoy Protection"

   Julian Reitman and Thomas J. Burke, "Production Management Information System"

   Donald Ingerman, "Simulation of a Railed Automated Highway", Digest of The Second Conference on Applications of Simulation, December 1968.

   Air Traffic Control Simulation Model Exploratory Study, United Aircraft Report #FAA-RD-70-2, December 1969.

   S. W. Seidler, The CASEE (Carrier Aircraft Squadron Effectiveness Evaluation) Simulation Model, Norden Report #1222 R 0001, July 1970.