Space vehicle control, guidance, and navigation require onboard computers. Mission safety and success demand high program reliability without preliminary in-flight testing.

Interactive Saturn flight simulation discussed in this paper tests all normal and perturbed launch vehicle interactions with the mission computer and programs to find and correct programming problems. Using a graphics console, flight analysts execute mission programs, make programming changes, and observe and document the simulated reactions of the launch vehicle.

Interactive Saturn flight program simulator

by J. H. Jacobs and T. J. Dillon

An interactive programming and flight-control system has been developed[1] for simulating the action of a launch vehicle with all of the complex and stringent requirements of space flight. The simulation program exercises a Saturn mission program by interactively simulating all normal and perturbed space vehicle operations required for navigation, guidance, engine control, event sequencing, and vehicle-ground intercommunication. Flight analysts, the specialists who develop the mission programs, have real-time control of the progress of the flight simulation at a display console. The analysts can also modify the onboard mission program and test the real-time effects of such modifications.

In this paper, we discuss the interactive Saturn program simulation system by which flight analysts "fly" all phases of launch vehicle missions in real time and through which they make programming changes and analyses. A basic system component discussed is the onboard digital computer, which holds and executes the mission program. Also presented are system components through which the analyst interacts with the onboard program and computer: interactive graphics system, laboratory computer and programming, and Saturn simulation programs. Interactive Saturn simulation is illustrated by a typical mission.

If space flight simply involved a ballistic trajectory or preplanned phases without the possibility of variations, no onboard flight computer and no program simulator would be needed. However, launch vehicles and space missions have become so complex that onboard flight computers are required for mission safety, success, and flexibility. The objective of our simulation system is to establish the reliability of the onboard program in performing the mission of injecting a manned spacecraft into a trajectory toward the moon. Mission program simulation is the only way of producing and correcting programming failures, because no opportunity exists prior to a mission for exercising flight programs in the actual space environment.

Several simulation techniques have been tried in the Saturn project:

- Interpretive digital
- Analog
- Analog-digital
- Multiple digital
- Digital, with interactive graphics

Interpretive digital simulation tends to be inefficient. Analog and multiple-computer approaches are difficult for the flight analyst to use. We implemented the last approach because of its ease of use, expandability, and realistic operation.

In order to fulfill the needs of the typical engineering-oriented flight analyst and to provide realistic Saturn launch vehicle simulation, we embodied the following capabilities into a simulation laboratory:

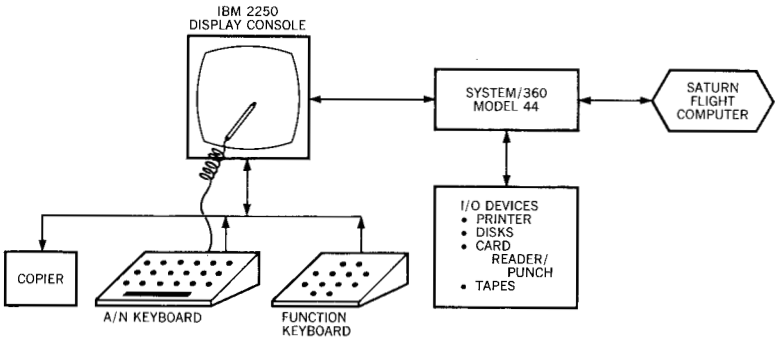
- Operational simplicity
- Real-time operation and display response
- Realistic launch control by the analyst
- Validation of flight simulation entries
- Real-time error indication and correction

The real-time, interactive simulation laboratory[2] is shown in Figure 1. It consists of an IBM System/360 Model 44, the Saturn flight computer, and an IBM 2250 display console.[3] A simulation program operates within the Model 44. The display console is the key element of this system because it provides the real-time, man-in-the-loop control of all phases of the Saturn launch vehicle simulation.

Saturn computing system

Our overall objective is to test Saturn mission programming by simulation in the environment of an actual flight computer with realistic input/output signals. Elements of the simulation are illustrated in the right portion of Figure 2 with a pointer to their places

Figure 1 Interactive simulation laboratory



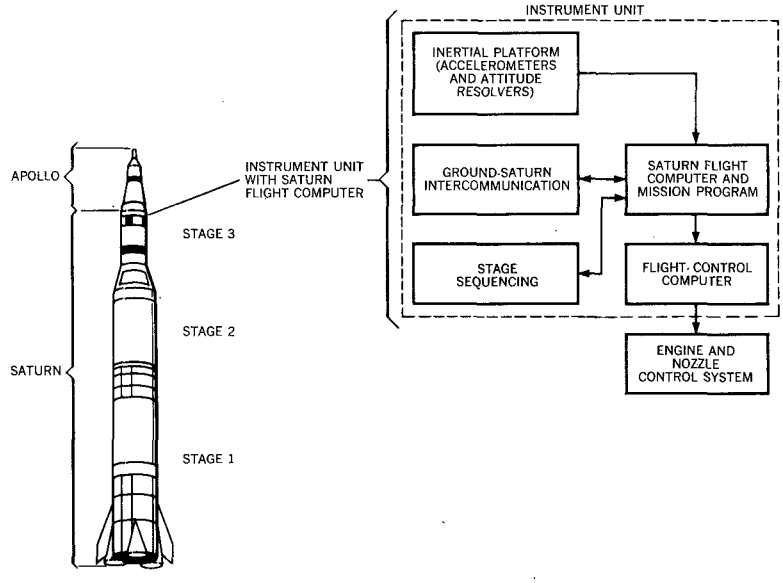
in the instrument unit of a simplified Saturn configuration to the left of that figure. The launch vehicle consists of the three powered stages shown plus the instrument unit. The Apollo spacecraft rides at the top of stage three and the instrument unit. Residing in the instrument unit, the flight computer and the mission program together control the powered stages with minimum interaction with the payload. In the Apollo configuration, the flight computer performs the general functions of prelaunch checkout, liftoff, guidance into earth orbit, orbital checkout, and guidance into a lunar transfer trajectory.

Using a stored program in its random-access, magnetic core storage, the flight computer operates in a binary-serial mode. This is the first onboard digital computer in which there has been sufficient confidence to entrust it to control the powered-flight phase of a manned mission. To achieve its high reliability, a two-out-of-three "voting" system involving triple-modular redundancy is used for the computer logic circuits. Also, duplexed storage modules detect and correct errors in the storage through the use of two parity bits on each computer word. Although the flight computer is designed for a special application, the following major system characteristics provide a high degree of mission flexibility within that application area:

- Twenty-six bit, fixed-length words
- Thirty-two thousand word storage
- Two instructions per word
- Eighteen assembly-language instructions
- Fixed-point arithmetic
- Interruption-oriented system

The flight computer occupies a two-and-a-half cubic foot package weighing seventy-two pounds. Through the use of triple-modular

Figure 2 Elements in a Saturn flight simulation



redundant logic, this operational computer has never had a failure that hampered program execution.

input/output

As shown in Figure 2, the flight computer is an integral part of the instrument unit, from which the computer receives its inputs and to which outputs are directed. Some input signals, such as attitude angles, are received by the computer in analog form and are converted into digital quantities before storing. On the other hand, digital inputs, such as those produced by the integrating accelerometers, are periodically received by the computer. These particular signals are immediately integrated to produce the launch vehicle state vectors, i.e., velocity and displacement. Typically, outputs that are produced by the computer in digital form are the event sequence commands (engines "on," "off," stage separation, etc.). Returned signals indicating that the commanded action has occurred form an additional class of input signals. These stage-sequencing computer outputs and inputs are received (and transmitted) by relays in the propulsion stages. Ground-Saturn intercommunication is handled digitally under computer control, whereas engine nozzle deflection signals are digitally produced by the flight computer and immediately converted to analog signals by internally contained adapters and transmitted to the engine control system.

mission program

Modularly structured, the mission program consists of two basic program packages totaling about 28K words in the flight computer: application modules and a control program. These are the programs

that the simulation system is designed to exercise. (Mission simulation will be discussed in detail later in this paper.) Application modules are a collection of relatively independent, closed flight programs, each of which performs a single or a number of related functions, e.g., process accelerometer data, compute guidance commands, or perform vehicle sequencing. As new requirements are defined, new application modules are added.

The application modules are divided into distinct segments according to vehicle modes of operation. A segment includes all functions required within a given phase of a mission, e.g., powered flight or orbital operation. Independently of the control program application modules are scheduled as a function of time and are assigned definite priorities. Beginning with such engineering requirements as guidance equations, mission sequencing logic, limits and tolerances, and performance data, mission program development extends to the specification of methods by which the mission program detects and compensates for equipment malfunctions in both the computer and space vehicle. As the mission program becomes more precisely defined, it is optimized to meet the requirements with minimum storage and execution time.

control program

Operational linkage, sequencing, and execution of the application modules for a given mission are governed by the control program through the use of an executive program and a common communication area. Whereas the latter allows for centralization of intermodule communications, common data, and mission or vehicle-dependent parameters, the executive program controls the execution of application modules, services interruptions, routes control to the appropriate application module on a priority basis, and provides utility operations. The control program is a group of interacting subprograms whose primary purpose is the sequencing of the execution of individual application modules. It examines interruptions and priority control tables to determine the order of module execution and provides a means of transferring operational control to and from these modules. Through a queuing process, these priority control tables are modified in real time. This enables the control program to execute application programs necessary to perform the functions required in the various stages of flight.

Saturn simulation system

To provide the analyst with facilities for effectively developing and checking out mission programs, a System/360 Model 44 with priority-interruption and high-resolution timing capabilities are linked with a Saturn flight computer. These features provide quick response to actions initiated by the flight computer and allow simulation in a multiprogramming mode. Another essential element is to provide interactive, real-time input controls to efficiently integrate

the analyst into the simulation process. Referring to Figure 1, this is accomplished by interconnecting a 2250 display console equipped with a light pen, alphameric keyboard, and variable function keyboard to the Model 44. An IBM 2285 display copier provides permanent records of display information. The analyst interacts with the Saturn flight computer via the 2250 and three major programs that execute in the Model 44:

- Control system
- Interactive graphics program
- Real-time simulation programs

control system

The simulator control system is the Model 44 programming system (44PS) with additions and modifications to convert it from a sequential batch job processor to a real-time, multiprogramming processor. All the original functions and features of 44PS have been retained; and programs not requiring the elements of a real-time, multiprogramming system operate as though the additional facilities were not present. The control system consists of a supervisor, assembler, FORTRAN compiler, linkage editor, and system support programs. Provision is made for both FORTRAN and assembly language processing. Program execution in a monitored environment with automatic job-to-job transition, interruption handling, and input/output supervision are embodied in the control system. Facilities for the creation and maintenance of libraries (both program and data) and for the manipulation of their contents are also provided by the control system. This system similarly provides extensive job control and program segmentation capabilities for flexibility and versatility in the preparation of programs for execution.

One of the required functions of the control system is the ability to activate various computing operations at precise time intervals. These operations are selected for execution by a time-sequencing priority scheme. Other operations are designed to execute as a result of interruptions induced outside of the central processor by the flight computer. These interruptions are generally of such importance that their priorities are higher than operations initiated as a result of timing. Multiprogramming functions operate through a scheme of priority interruptions, with the requirement of real-time operation being the principal decision criterion for the control system priority algorithm. To satisfy these requirements, capabilities in three primary areas have been added to 44PS: multiprogram scheduling, real-time input/output scheduling, and application program overlay control. Also added is a posting function associated with I/O termination, the purpose of which is to allow higher priority routines to request I/O, to relinquish control to a lower priority level, and then regain control when the higher I/O priority is completed. A comprehensive statistics gathering capability is used to evaluate the multiprogramming system performance.

It is in the supervisor that additions and changes have principally been made to the 44PS. Control of the entire simulation system and a common interface to all mission programs are provided by supervisor management of input/output devices, data sets, and mission programs. Modules are loaded from the system program library as required by job control or as requested by other programs being executed. The supervisor handles input/output requirements including error recovery procedures and the execution of input/output without reference to a particular device type by the analyst. All classes of interruptions are serviced with transfers to the appropriate simulation system or application module for processing, with input/output channels scheduled for overlapping in these operations.

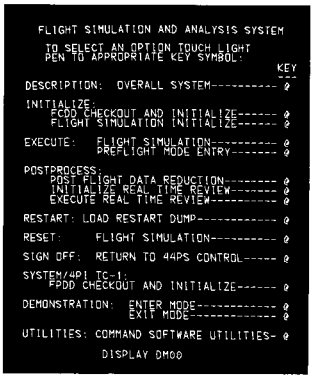
interactive graphics program

The interactive graphics program is the real-time, interactive link between the flight analyst and the mission program, whereby he executes the flight simulation. Facilities are provided for the complete display of real-time flight information from both the simulator and the flight program and for stopping, analyzing, and perturbing the flight action as it is taking place. Flight setup and post-flight analysis are also provided.

Because the graphics program is based on a tree-structure organization, the analyst has available at each stage of the simulation a hierarchy of options that he selects either through light pen or the alphameric keyboard action. These actions select programs for Model 44 processing from the tree structure, thereby allowing the flight analyst to selectively step through many levels of flight simulation detail.

Illustrated in Figure 3 is the first tier of branches (options) in the graphics program. The next level of options is called and displayed by touching the areas labeled "key" in the column to the right in the figure. By a system of program linkages, successive levels of display and program execution are initiated. Currently there are approximately three hundred different displays available to the flight analyst. To facilitate rapid display transition and to minimize the CPU time required to move from one display to the next, the set of displays is maintained on direct-access storage in the form of graphic orders ready for immediate transmission to the 2250 display buffer.

Figure 3 Graphics program initial options



Each tier of options contains all relevant information for that level of the graphics program hierarchy, including orders necessary to display information contained in the 2250 display buffer, activate light-pen-sensitive areas, and to enable inputs from the alphameric keyboard and the light pen to select other branches of the graphics program. Internally associated with each graphics program level is control information for indicating the next display to present following a particular light pen or keyboard action. An indicator also specifies the simulation program to be activated following

an operator action. Typical of the simulation programs specified are those for: vehicle simulation setup and execution, data reduction, and hardware diagnostic testing.

The graphics program is linked to the simulation programs through an internal transfer table of address constants. The graphics program is resident in the control system supervisor and is responsible for initiating all flight-analyst requests made through the display console. After system initialization, the graphics program presents the initial options on the display screen, as shown in Figure 3, to allow the flight analyst to begin the simulation. Since the graphics program is operating in a multiprogram environment, it may be requested to concurrently processes information from several sources. Although the graphics program is not reentrant, it accepts multiple flight-analyst and simulation-program requests by placing them in request queues and processing them in a first-in/first-out manner

Another input to the interactive graphics program is function-key action, which is initiated differently from that of the light pen and alphameric keyboard. The function keyboard is used primarily for analyst requests that are independent of the current display, such as, for example, the freezing of the current graphic display to produce a hardcopy output. Function keyboard activation is initiated by dynamically attaching a simulation program to a particular function key with a special call to the graphics program. In the same way, the simulation program is detached from a function key.

Although the light pen is the preferred option-selecting device, we often find it necessary to enter large quantitites of symbolic and numeric data (e.g., the specification of engine thrust perturbations, sequencing failures, etc.) that cannot be conveniently entered by light pen. Such information is entered through the alphameric keyboard. Since errors frequently occur when entering alphameric information, the graphics program provides a validity check service. As a part of this check, additional information is associated with the displays in the graphics program hierarchy that indicate which alphameric input fields are allowable. The added information specifies the length of the fields and the type of information that may be accepted; entries not meeting these specifications signal error conditions to the flight analyst. Having determined the validity of the alphermeric input information, the graphics program converts the input data from the System/360 EBCDIC code to a format appropriate to the purpose of the particular data entered (i.e., decimal, floating point, or binary). Similarly, data format conversion occurs in the "reverse direction" during a simulation exercise when display plotting is required for evaluating the progress of a flight. Through requests to the graphics program, numerical information may be plotted on the display screen. Data reduction programs collect the necessary information and formats it in tables to be passed to the graphics program, which translates the resultant tables into 2250 graphic

orders and plots the information in either point or vector form. An arbitrary number of plots may be superimposed on a single display grid.

At any time during the exercising of an application module, status or error messages may immediately be presented to the flight analyst. In signaling an error, the simulation program activates the display program with a request indicating the status or error message to be presented. Normally, this message immediately overrides the existing display. If, however, another status or error message is already being displayed, further requests are placed in a message-request queue and a function key is activated so that the flight analyst may step through the message queue. Thus neither can status and error messages flash rapidly on the display screen, nor can the analyst proceed with the simulation without examining all such messages awaiting his attention in the queue.

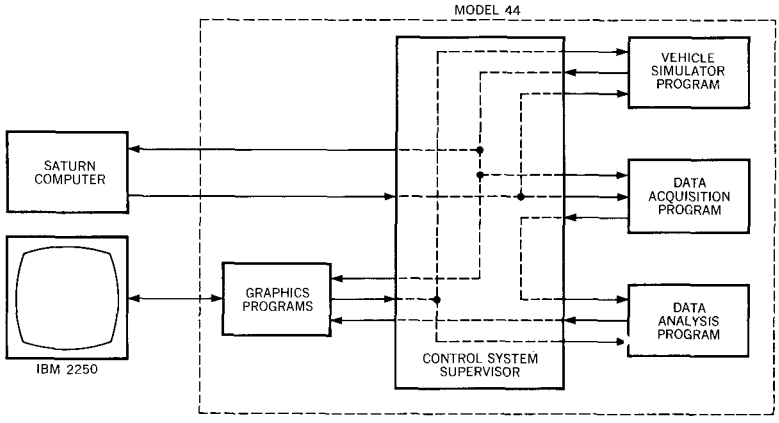
For more routine operations, the graphics system has been designed to operate with an automatic display sequence program, which automatically selects options, executes the simulation, and records results. To do this, the flight analyst records the display option selection sequence information on punched cards or in direct-access storage, and the simulation proceeds according to this preplanned selection. Should the analyst later rerun the same or a similar job, the system provides for calling up the information previously recorded and then executing the entire simulation exercise in a manual or automatic mode, depending on whether the analyst plans to modify the previous simulation information. To provide the flight analyst with a complete record of the display actions taken during his run, the system is designed to record all significant display events on the printer.

vehicle simulator

Up to this point we have introduced the programs and the simulation laboratory equipment that the flight analyst uses in simulating the flight of a Saturn launch vehicle. Recall that the objective of this simulation is to test mission programs with all the realism necessary to find any programming and engineering errors or anomalies. Underlying mission programming is the phenomenon of the launch vehicle's motion, which is described in terms of the vehicle's six degrees of freedom: roll, pitch, yaw, and the three spatial coordinates x, y, and z. These motions produce the signals in the accelerometers and attitude resolvers in the inertial platform that are transmitted to the flight computer as previously discussed. We now discuss the program that simulates the input to the flight computer and accepts the resultant flight computer output commands.

Relationships among programs and computing system elements stored and executed by the Model 44 are shown in Figure 4: the vehicle simulator (which operates in a real-time, multiprogramming mode with the data acquisition and analysis programs), the graphics

Figure 4 Saturn simulation programming and equipment



programs, and the supervisor part of the control system. The major portion of the vehicle simulator program is the six-degree-of-freedom, vehicle-response model that closes the loop between engine command signals from the Saturn flight computer and inertial platform input data back to the flight computer. The 6-D model also simulates stage sequencing and the astronaut and ground command systems.

Data from the flight computer intended for transmission to ground are collected continuously by the data acquisition program. In parallel with flight computer data, vehicle simulation data are also picked up by the data acquisition program from the Model 44. Both kinds of information are stored on direct-access devices for immediate processing by the real-time data analysis program; these data are also logged on tape for off-line processing by a post-flight data analysis program not shown in the figure.

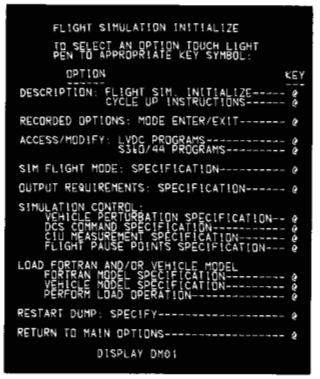
Data analysis routines are designed to access results of all flight computer and vehicle simulation calculations as they are saved by the acquisition program for immediate processing, scaling, and presentation in real time during the simulation. Such presentations enable the analyst to monitor the progress of the simulation and perform the mission program modification in real time. Numerical data are processed by the data analysis program for presentation in either tabular or graphic-plot form. Expected parameter values from the simulation programs and computed flight parameters are differenced, and the differences are displayed primarily for use as off-course and out-of-bounds indicators for the Saturn launch vehicle.

In the initialization phase of a simulation exercise, the flight analyst may specify the required set of real-time presentations. The data acquisition routines then save all information necessary for analysis and presentation during the exercise. While the simulation is in

progress, the system is designed to give the flight analyst the option of changing from one presentation to another. Of course, flight status information (velocity, acceleration, and displacement) is available for display at all times, having been placed in continuously updated storage by the acquisition program. Data reduction routines process the status information for display in a predefined format to inform the flight analyst of the progress of the mission.

Each component of the real-time launch vehicle simulator shown in Figure 4 is assigned a priority within this multiprogramming environment. Thus the simulator is implemented according to a real-time process optimizing algorithm. In general, priorities are assigned according to the importance of the function being performed, whereby human safety factors have highest priority. Normally the six-degree-of-freedom functions and hardware indications occupy the higher priority levels. These are followed in order of priority by the graphics communication functions. Data analysis programs use the lowest priority levels. Additionally, high priority factors have been designed for rapid execution in order to create a highly responsive system. A higher priority factor preempts one of lower priority that is being executed; the lower priority factor resumes execution at the point at which processing was preempted when all higher priority programs have been executed. Thus the optimization algorithm embodies value judgments of the relative importance of the calculations performed by the mission program while at the same time reckoning with the fact that all factors must be calculated. For speed, therefore, at each priority level only factors essential to that level are computed; other items are computed at the appropriate level. Routines at the highest priority levels require less than three milliseconds for execution.

Figure 5 Flight simulation initialization



Flight simulation

To demonstrate the system principles discussed, we now step through a hypothetical mission simulation as an analyst "flies" it. Before executing the flight simulation, certain initialization procedures illustrated in Figure 3 must be performed. This figure is a photograph of the display screen presenting the major simulation system functions, one of which is "flight simulation initialize." Using the light pen, the analyst calls for the initialization display shown in Figure 5. At this point, his primary actions are loading the program into the flight computer, entering launch vehicle specifications, and specifying the required real-time outputs. These are the factors that change for each simulation, or that are exercised, or that are expected to yield new information.

Figure 6 Begin simulation exercise

The flight simulation begins executing when the start/resume option in Figure 6 is selected. With this action, the flight computer begins active navigation, guidance, and control according to the mission

Figure 7 Tabular display of velocity components

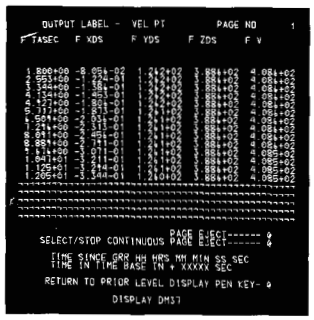


Figure 8 Graphic display of velocity components

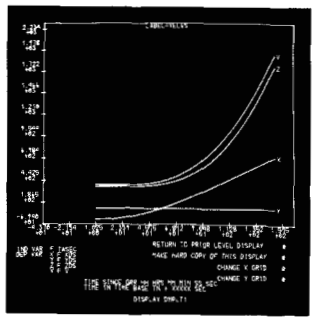
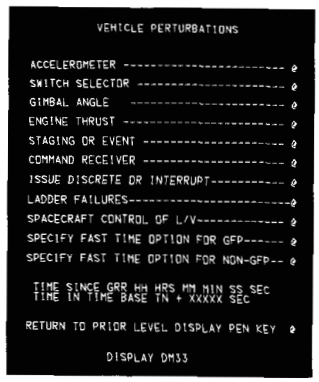


Figure 9 Vehicle perturbation selection display



specifications, and all the programs previously discussed are running in a multiprogramming mode. The flight computer requests mission data and executes its program using the data asynchronously with the vehicle simulator in the Model 44. The flight computer also transmits data to the simulator as it is calculated by the mission program.

While the simulated flight is taking place, the analyst usually monitors the launch vehicle's progress by means of tables and plotted outputs. An example of tabular data are the vector components of velocity shortly after launch shown in Figure 7. Notice that at the time of the display the table is only partially filled; lines of information are added as the flight progresses. The same information may also be dynamically displayed in graphic form as shown in Figure 8. Here the three velocity components and the resultant magnitude are shown superimposed as they are calculated later in the same launch as that of Figure 7. If the flight analyst retains this display he observes the motion of the vehicle by the dynamically changing configuration of the velocity curves. At any time in the flight, restart information can be captured in the form of restart dumps. This capability allows the flight analyst to resume the flight at any restart point rather than starting from launch. Information is captured by touching the "take restart dump" key (shown in Figure 6.) with the light pen. The analyst can conveniently check mission program modifications that affect the flight after several hours into the run without having to begin the run from liftoff each time a problem is discovered.

During the flight, the analyst may wish to inject one of the vehicle perturbations illustrated in Figure 9. Such perturbations simulate performance failures in any vehicle function that interfaces with the flight computer. The effects of these failures can be viewed in tables and graphs like those shown in Figures 7 and 8.

If the mission program does not respond effectively, programming malfunction is apparent in the graphic displays. Unlike actual launch and flight, the progress of the Saturn space vehicle can be frozen in space and time by pointing to the "stop" option. At this point the analyst can review all data so far collected. If necessary, he may view the flight execution on an instruction-by-instruction basis by entering a trace mode of operation, whereby the flight computer is stepped through the program. The system is designed so that programming and program modifications may be done on-line by the analyst's entering instructions or data modifications through the 2250. Such an instruction modification is shown being made in Figures 10 and 11, where Figure 10 shows the new instruction just entered through the keyboard and Figure 11 shows the previous contents of a storage location and the new contents of that location. The flight may then be resumed at the point where it was stopped, or it may be backed up either to a restart point, or it may be re-

launched. Using these techniques, the entire mission can be executed. At any point, execution may be suspended and the recorded information may be analyzed by post-flight data reduction programs.

Concluding remarks

The graphic simulation system discussed in this paper has been used extensively in support of the Apollo 8, 9, 10, 11, and 12 flight program developments, and it has met all design objectives. By providing a new dimension of program visibility and man-computer interaction, the system has had the effect of stimulating higher levels of interest and creativity by flight analysts.

Problems of designing a mission program simulation system and ensuring that the analyst has the capability he requires has been a complex task. Our experience has shown that simulation techniques, selection of equipment, and methods of man-computer and computer-computer interaction must be carefully balanced to meet overall design objectives. Requirements for high precision and safety of the manned mission introduce elements of realism and urgency into the abstract problem of space-vehicle control, guidance, and navigation. The resultant simulation system that combines these factors in a real-time multiprogramming environment has proved to be a valuable tool for the development and checkout of mission programs. The system has thus made possible the development of mission programs that have been relied upon to a much greater extent than previous mission programs and has reduced the amount of time necessary to produce them.

Programming concepts we have used may certainly be applied in related areas such as process control and other real-time-oriented technologies. The operating system[4] and graphics program have direct conceptual applications in other airborne and space vehicle simulators.

Figure 10 Entering a new instruction in the flight computer

Figure 11 Old and new instruction for the flight computer

ACKNOWLEDGMENT

The interactive simulation laboratory is based on the concepts and original development work of T. H. Witzel, J. O. Thompson, and J. S. Hughes.

CITED REFERENCES AND FOOTNOTES

1. The work described in this paper was performed under a contract with the National Aeronautics and Space Administration for the George C. Marshall Space Flight Center at Huntsville, Alabama.
2. J. S. Hughes and T. H. Witzel, "On-line software checkout facility for special purpose computers," *AFIPS Conference Proceedings, Fall Joint Computer Conference,* **35,** 789–800 (1969).

3. A. Appel, T. P. Dankowski, and R. L. Dougherty, "Interactive graphics in data processing: Aspects of display technology," *IBM Systems Journal,* **7,** 3 and 4, 176–187 (1968).
4. J. L. Johnstone, "RTOS—extending OS/360 for real time space flight control," *AFIPS Conference Proceedings, Spring Joint Computer Conference,* **34,** 15–27 (1969).