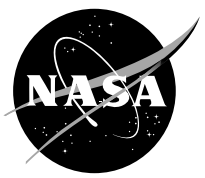
NASA Technical Memorandum 104315

Flight Simulation Software at NASA Dryden Flight Research Center

Ken A. Norlin

October 1995



National Aeronautics and
Space Administration

NASA Technical Memorandum 104315

Flight Simulation Software at NASA Dryden Flight Research Center

Ken A. Norlin
NASA Dryden Flight Research Center
Edwards, California

1995

National Aeronautics and
Space Administration

Dryden Flight Research Center
Edwards, California 93523-0273

FLIGHT SIMULATION SOFTWARE AT NASA DRYDEN FLIGHT RESEARCH CENTER

Ken A. Norlin*
NASA Dryden Flight Research Center
Edwards, California

Abstract

The NASA Dryden Flight Research Center has developed a versatile simulation software package that is applicable to a broad range of fixed-wing aircraft. This package has evolved in support of a variety of flight research programs. The structure is designed to be flexible enough for use in batch-mode, real-time pilot-in-the-loop, and flight hardware-in-the-loop simulation. Current simulations operate on UNIX-based platforms and are coded with a FORTRAN shell and C support routines. This paper discusses the features of the simulation software design and some basic model development techniques. The key capabilities that have been included in the simulation are described. The NASA Dryden simulation software is in use at other NASA centers, within industry, and at several universities. The straightforward but flexible design of this well-validated package makes it especially useful in an engineering environment.

Introduction

Flight simulation has a vital role in the flight research performed at the NASA Dryden Flight Research Center. Simulation benefits all phases of a flight research program: the early conceptual and design phase, systems design and testing, and flight test support and envelope expansion.

A simulation laboratory was established at NASA Dryden in 1957 to support the X-1B flight program.[1] Capability expanded and experience increased while supporting a variety of unique and challenging flight research programs, including the X-15 aircraft and lifting body vehicles. These early simulations used analog computers and varying degrees of flight hardware in the loop.

The first all-digital simulation at NASA Dryden was developed in 1970 for the 37.5 percent–scale unpowered F-15 Remotely Piloted Research Vehicle (subsequently known as the Spin Research Vehicle). A standard simulation structure was established in 1975. Since then, the simulation software has steadily evolved into a versatile and flexible engineering tool. The simulation capability now forms the backbone of the NASA Dryden Integrated Test Facility that opened in 1992 to support ground and flight testing of advanced research aircraft.[2]

The flight research environment at NASA Dryden relies extensively on simulation to support development, modifications, and verification and validation of aircraft control laws. In addition, simulation is used for mission planning, safety-of-flight analysis, pilot training, and evaluation of new vehicle concepts. The simulation for any aircraft continually evolves as the flight research program matures. Changes to the vehicle definition, revisions to the control laws, and updates from the most recent flight test results are incorporated into the simulation as required.

The flight vehicles simulated are usually unique, uniquely modified, or conceptual fixed-wing aircraft. Vehicle types range from small subscale models to large transport aircraft, from sailplanes to hypersonic and orbital vehicles, and from benign to highly agile, high-performance fighters. Some of the currently supported projects include the X-31, X-33, X-34, F-18 HARV, F-15 ACTIVE, F-16XL SLFC, F-18 SRA, Perseus, and APEX programs.

The small simulation staff, large number of projects, and frequent model updates (based on flight results) have

*Aerospace Engineer. Member AIAA

Copyright © 1995 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

driven development of a versatile standard simulation structure. This structure, with both flat- and oblate-Earth versions, has successfully supported more than 50 different aircraft. The software is used in batch-mode, real-time pilot-in-the-loop, and flight hardware-in-the-loop operation.

The goal of the simulation group is to provide a high-fidelity simulation that is exceptionally capable, flexible, and responsive to the needs of the researchers. These simulations are not production training simulations; they are engineering simulations used in a research environment. Fixed-base cockpits are used with modest visuals. Every simulation is similar in terms of software structure, user interface, naming conventions, operating characteristics, and key capabilities. These similarities allow users and developers to be immediately familiar with the simulation regardless of which project they are supporting. The end result is a highly efficient, effective, and productive simulation environment. This paper discusses the simulation software structure, some basic model development techniques, and several capabilities and features.

Nomenclature

ACTIVE	Advanced Control Technology for Integrated Vehicles
GRAM	Global Reference Atmospheric Model
GUI	graphical user interface
HARV	High Alpha Research Vehicle
NASP	National Aerospace Plane
SES	Simulation Electric Stick
SLFC	Supersonic Laminar Flow Control
SRA	Systems Research Aircraft
V&V	verification and validation
A	stability matrix
B	control matrix
C	state observation matrix
D	control observation matrix
u	control vector
x	state vector
$\dot{x}$	state derivative vector
y	output vector

Software Design

The software design philosophy is to be responsive to the needs of the researchers by including as much capability and flexibility as possible in the basic simulation structure. Portability of the software has been proven across numerous platforms as upgrades are made to take advantage of increasing computer performance. Current real-time simulations operate on multiprocessor UNIX-based computers, and batch simulations run on desktop workstations.

The simulation software primarily uses FORTRAN 77 and C languages, but the capability to interface with Ada routines also exists. The majority of the code is written in FORTRAN. This FORTRAN code includes the aircraft models, equations of motion, integration, table look-ups, initialization, and display generation routines. C is used for the graphical user interface (GUI) and system specific routines such as memory mapping, priority boosting, and interrupt handlers. In addition, C is used for the graphics and distributed system functions that are described in the "Distributed Systems" subsection.

FORTRAN continues to be used because of the tremendous amount of FORTRAN code retained from earlier simulation efforts. The balance between FORTRAN and C is shifting toward C as older code becomes obsolete. If onboard aircraft software is written in Ada, then that Ada code is also integrated into a simulation rather than recoding in another language. UNIX "makefiles" are used to compile the simulation. The use of "makefiles" allows minor updates to be made in less than 5 minutes.

Any real-time simulation structure logically lends itself to splitting the user interface from the real-time loop. The NASA Dryden simulation is no different. The simulation is primarily comprised of two main tasks: the background executive and the real-time loop (fig. 1).

The main simulation program is referred to as the background executive. This FORTRAN task is used to initialize the simulation database and provide a display and command line interface for monitoring and controlling the simulation. Once this main program has performed its initialization function, it starts the real-time loop as a separate task. The real-time loop may contain one or more programs run in parallel. This parallelism is used as needed to meet frame-time requirements. For batch mode, only one pseudoreal-time loop exists that runs the models in series.

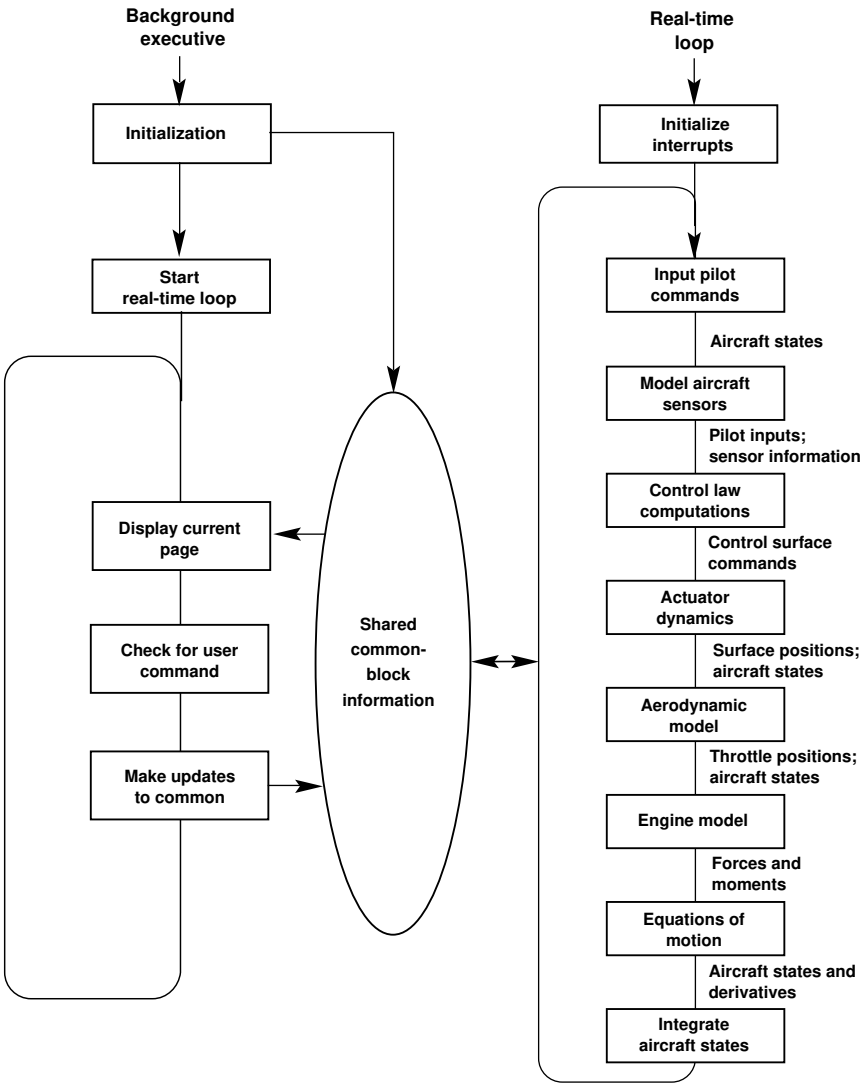


Figure 1. Simulation diagram.

These multiple programs communicate through shared FORTRAN common blocks. Although common-block sharing between separate programs is nonstandard FORTRAN, most of the UNIX-based machines tested by NASA Dryden support this scheme using some combination of compiler and linker options.

Sharing common blocks between tasks provides an extremely flexible simulation environment. All flags, switches, data tables, intermediate variables, and any other information that might prove useful is placed in FORTRAN common blocks. The high-priority real-time loop runs at its required frame rate and simply uses the values in this common-block region. Meanwhile, the background task runs at a slower rate and provides an interactive interface with the user to monitor and modify the common-block information.

An additional benefit of structuring the code with shared common blocks is the ease with which simulation variable information can be generated and accessed. All the common blocks that are to be shared are put into one or more common-block files. These files are then included in the declaration region of the main background and real-time FORTRAN programs.

A simple common-block parser program has been created to read these common-block files and generate variable list and variable address files. The variable list file is read by the background executive during initialization and includes the variable name, type, byte size, and array dimension. The background executive performs a FORTRAN "INCLUDE" of the variable address file to determine the variable addresses at the time of the run. All the variable information is then stored in a common block.

Figure 2 shows the process for variable list generation and implementation in the background executive. This process is mechanized in the simulation "makefiles" to run automatically whenever the common-block files are modified.

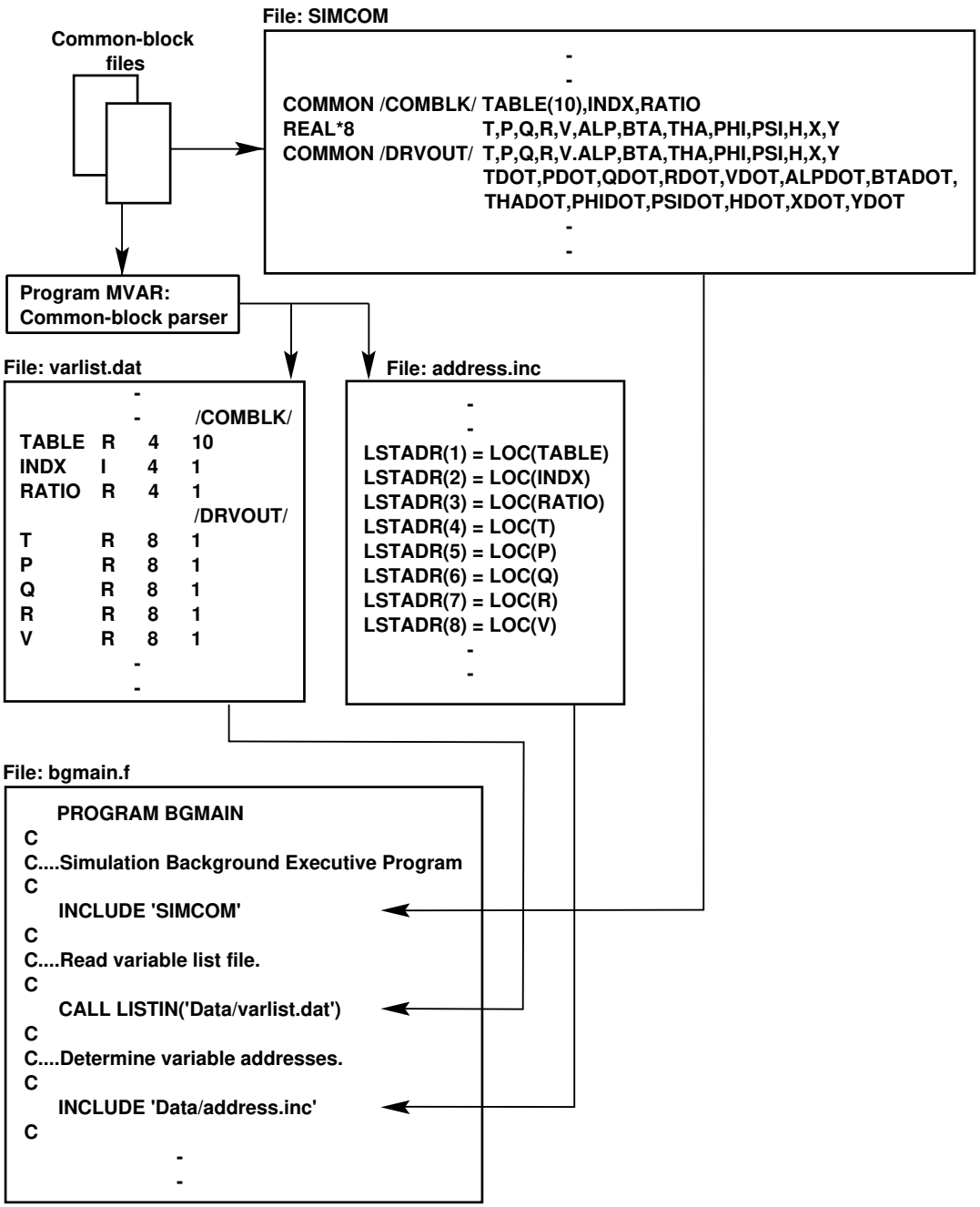


Figure 2. Variable list generation and implementation.

Background Executive

The purpose of the background executive is to provide for simulation initialization, control, and display. The background executive includes many features that enhance its utility and flexibility, such as data file reinitialization during run time, command line interface with scripts, interactive display pages, and easy access to all variables.

When the background executive is first started, it initializes the data in the shared common-block region. The initial conditions, aircraft geometry, and mass properties are read from data files. All the tables that comprise the simulation database for the various models are also read at this time. Tables for the aerodynamic coefficient data, engine model values, and control system gains are normally grouped into separate data files.

Using data files to initialize this information allows the initial setup and database to be updated without requiring recompilation. In addition, all software switches, failure flags, and control system modes are initialized at this time. These switches are normally hard-wired in the code. However, sets of logical and floating-point arrays called user variables exist that are initialized from data files and provide quick and flexible model modification capability.

After the initialization is finished, the background program starts the real-time program. From this point, the background executive loops continuously on a time-available basis. The current display page is updated on the screen each frame of the loop, and the command line is checked for user input.

The display interface uses a standard 24-line-by-80-character output. These dimensions have been maintained to retain code commonality with pre-UNIX computer systems that are still in use. When these older systems are phased out, the display output dimensions can easily be modified. The display output buffer is generated using FORTRAN code. However, on the UNIX systems, a C-language GUI has been created to display the information.

A flexible command line interface is provided for the simulation user. Commands can be typed in using the keyboard, or a script interface can be used. Scripts are files that contain multiple simulation commands. These script files can be read directly by the simulation instead of requiring the user to type in all the commands. Repeatability is thus guaranteed because typing errors are prevented. Currently, nesting scripts to one level is possible, allowing complex tasks to be performed by a series of simple scripts.

Because all pertinent data is included in the shared common region and can be accessed from this command line interface, the simulation becomes a powerful tool for the user. If a need exists to alter the center of gravity, increase a control system gain, or modify the directional stability term, the background executive provides a mechanism to quickly make these modifications during a simulation session while the program is running.

Dozens of interactive display pages exist that are updated continually and contain virtually all parameters of interest (fig. 3). The user can go from one display to another by simply typing in the new display name. Each display is designed for a specific purpose, and depending on which display is currently being viewed, only certain simulation options can be modified. Project-specific displays and a set of generic displays are available to all simulations. The project-specific displays usually include customized pages for displaying and modifying information regarding the various aircraft models. Custom displays usually exist for the aerodynamic, engine, and actuator models as well as the control system.

Generic displays are available in every simulation. One generic display is the initial condition page that allows the user to vary the initial aircraft states. For example, the user can select the initial airspeed, altitude, or angle of attack. Another generic display is a trim display that allows specification of trim type and trim constraints. The default trim type adjusts the angle of attack and stick inputs to trim the aircraft for level flight at the selected airspeed. Other generic displays allow the user to select customized pilot inputs, set up disturbance models, monitor cockpit input/output, look at timing, or initialize data recording and playback.

Two flexible displays are the strip-chart and multivariable displays. These display pages allow the user to view or modify any simulation parameter from the variable list. The strip-chart page was designed specifically to provide a user interface to strip-chart recorders (fig. 4). The variables to be output to the strip charts can be selected on this page; their values can be changed; and the minimum and maximum values for scaling can be specified. The multivariable page is a generic display that allows a maximum of 36 simulation parameters to be specified on one page. These displays contain a feature that allows the user to save the setup to a file so it can be recalled during later simulation sessions.

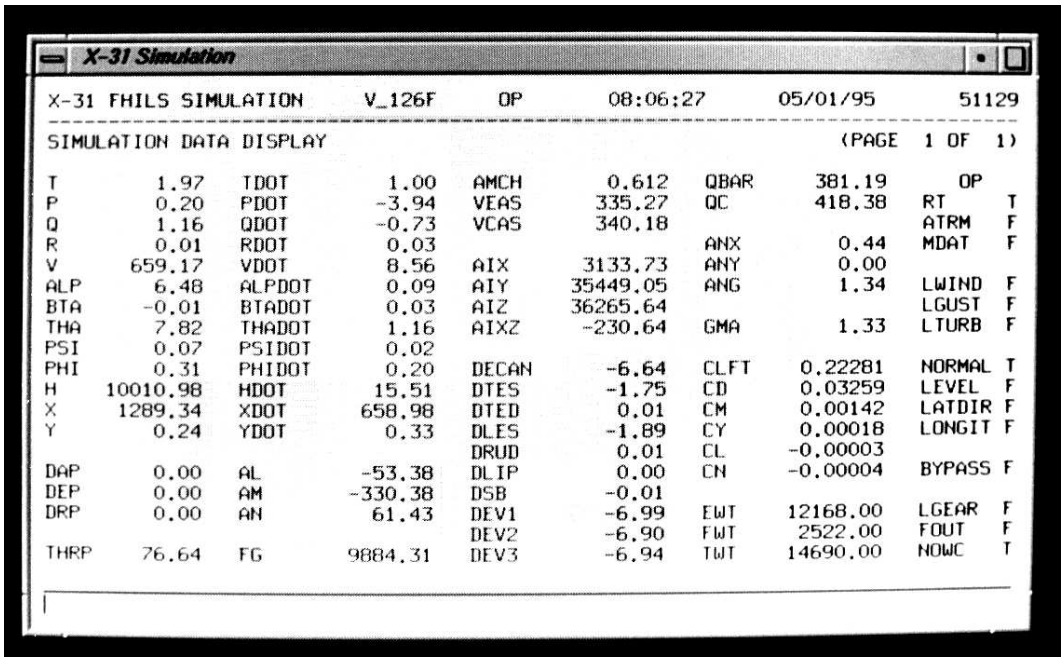


Figure 3. Sample simulation display.

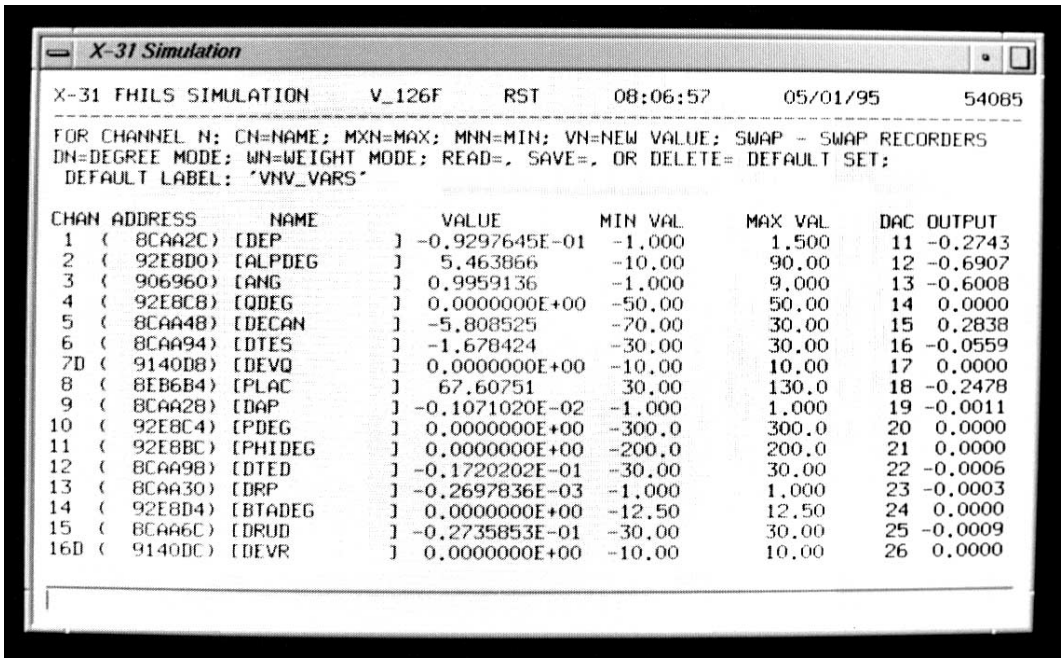


Figure 4. Simulation strip-chart display.

In addition to display and scripting commands, other commands can be entered from any display page. The simulation mode can be changed by entering the reset, hold, or operate commands. A “dump” or “copy” command sends the current display to a line printer or a file, respectively.

In conjunction with the FORTRAN background executive program, a C-language GUI is used. The toolkit for this GUI was developed at NASA Dryden in 1989 and is based on low-level X-library calls. Currently, an effort is being made to use a more standardized GUI to take advantage of the commercial GUI builders that are available.

Figure 5 shows a representative GUI interface for the X-31 simulation. The main simulation display window is shown on the bottom left with the command line along the bottom. The window shown on the bottom right provides point-and-click user control and a mouse-driven cockpit for pilot-stick, rudder-pedal, and throttle inputs. A log window, shown on the upper left, was recently added to make the simulation user friendly. User commands are echoed in the log window, and errors, warnings, and notices are posted to provide feedback to the user. The window shown on the upper right shows the graphics capability that will be described in the "Three-Dimensional Solid Model Graphics" subsection.

The code for the background executive is usually the same for batch-mode, real-time pilot-in-the-loop, and flight hardware-in-the-loop simulation; however, various displays may be inactive in some cases. For example, during hardware-in-the-loop simulation using the flight control computers, the control system display is not used when internal control system information is not available.

Real-Time Loop

All the simulation model calculations and integration of the equations of motion are performed in the real-time loop. For pilot-in-the-loop or hardware-in-the-loop simulation, the main real-time task is interrupt-driven at the highest allowable system priority. Any additional real-time programs may be synchronized with memory flags or run at their own required frame rate. Computers with multiple central processing units are used for real-time operation to ensure timing constraints are met. In batch-mode simulation, the real-time loop normally runs as fast as it can with round-robin scheduling performed by the UNIX kernel.

As in most simulations, three modes exist: reset, hold (or freeze), and operate. One aspect of the NASA Dryden simulation that should be emphasized is that all the model code is executed even while in reset mode. Some reinitialization is performed, but the simulation runs through the dynamics and control system code as if the simulation were in the operate mode (fig. 6). The main

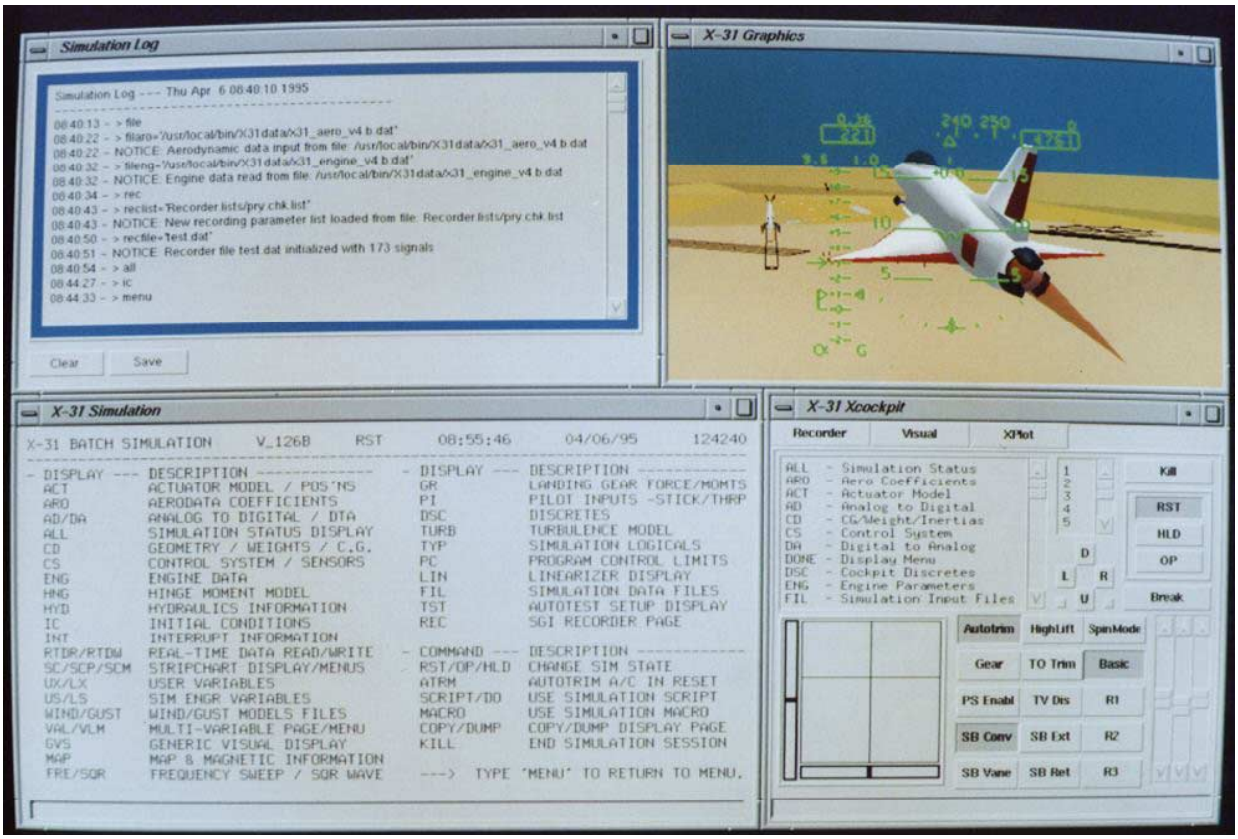


Figure 5. Simulation graphical user interface.

difference is the aircraft states are not integrated. This method has proved valuable in debugging. Execution in reset mode also allows the aircraft, in a sense, to fly to new initial conditions as they are selected on the initial conditions page. For the X-31 hardware-in-the-loop simulation, the aircraft states were ramped to the new initial conditions to prevent tripping the redundancy management. The simulation may be placed in the hold mode to freeze the current conditions. This feature facilitates debugging and analysis.

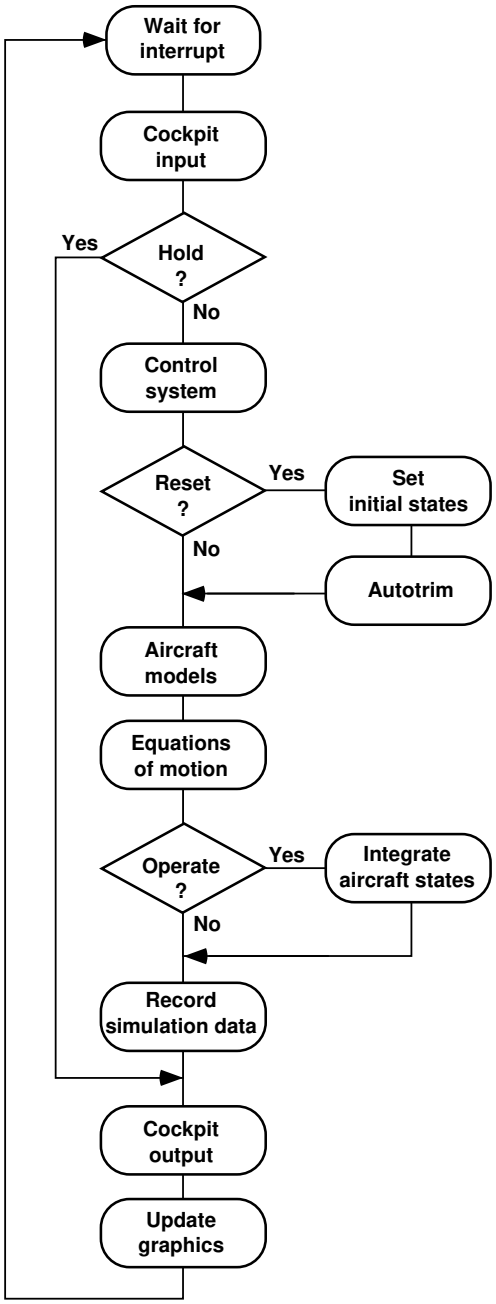


Figure 6. Real-time loop flow chart.

The NASA Dryden simulation uses an integration scheme that has been optimized for real-time operation. This method differs from the classical second-order Runge-Kutta method in that the derivatives are only calculated once for each frame. Therefore, the simulation is able to run at high frame rates, and the performance is still superior to first-order methods. As shown in the integration code (appendix A), a weighted average of the midpoint and previous frame derivative is used to predict the derivative at the end of the frame.

Most NASA Dryden simulations contain the ability to vary the integration interval while the simulation is running. The dynamic filter and rate calculations are coded to incorporate this ability. This feature was originally included to allow the developer to easily adjust the simulation frequency to prevent frame overruns. The feature also facilitates faster-than-real-time simulation. Research has shown that faster-than-real-time simulation practice can improve pilot performance during high gain tasks.[3] Varying the frame rate also proved useful during NASP simulation studies, allowing the aircraft to quickly fly a mission profile by accelerating the operation of the least dynamic phases.

In most cases, flat-Earth six-degree-of-freedom equations of motion are used. Oblate-Earth equations of motion were developed for the space shuttle simulation and later used in the NASP and follow-on simulation studies. The flat- and oblate-Earth equations of motion use a hybrid axis system that allows forces and moments to be added in the axis systems for which they are commonly computed, thus reducing axis transformations.

Generally, these equations of motion are simplified by assuming aircraft symmetry about the x-z plane so that the inertial terms "I_{xy}" and "I_{yz}" are 0 slug-ft^2. However, this assumption is invalid for some aircraft, such as the AD-1 Oblique Wing airplane and the F-16XL airplane with the supersonic laminar flow glove on the left wing. In these aircraft, all inertial terms are included. Additional flexibility is provided with flags that can be set to restrict some of the degrees of freedom. For example, the simulation can be set up to allow only longitudinal or lateral–directional motion.

Like the background executive, the code for the real-time loop is kept consistent between batch-mode, real-time, and hardware-in-the-loop simulation. Flags are set during initialization to indicate in which mode the simulation is to operate. These flags are then used to determine which sections of code are executed in the real-time loop.

Model Development

Using the software structure described in the preceding section, implementation of a new aircraft simulation is a rapid process. Depending on model availability, simple simulations (such as Perseus) can be ready for the user in as little as 2 weeks. More complex simulations with flight hardware in the loop (such as the X-31) require approximately 9 months to develop.[4]

Development usually begins by copying the most current simulation software and replacing all the aircraft-specific models. Normally, this process requires math models for the aerodynamics, propulsion system, control laws, and actuator dynamics. In most cases, additional models are required. For example, the F-18 HARV and X-31 aircraft require thrust-vectoring models to determine moments provided by the thrust-vectoring systems.

Data Files

Models that require large data tables are initialized by reading data files at startup. This method allows the database to be changed without requiring recompilation. An added feature is a file display page that allows the database to be modified during a simulation session. For example, the user can type in the new file name for the aerodynamic database, and the background executive will read the new file and copy that data into the appropriate common block. Because the real-time loop is using the data values in the shared common region, this change is incorporated immediately.

Because the data for different simulations and different models come from a variety of sources and disciplines, the data formats will vary significantly. Therefore, a unique input subroutine is written for each data file.

Table Look-Ups

Many simulation models, especially most aerodynamic models, include large, multidimensional table look-ups. A common approach is to develop a generic set of library routines to perform the table look-up function. The disadvantage to this approach is that the overhead required to make these subroutine calls can be unacceptable in a real-time environment. Therefore, NASA Dryden usually uses custom-coded table look-up routines for each project.

Because this approach could be prohibitive in the cost of development time, a table look-up code generator was developed in the late 1970s to support the space shuttle simulation. This program produces complete FORTRAN table look-up code for functions of a maximum of 9 independent variables by reading files that describe the variable dependencies and breakpoint structure. Considering the number of simulations and modifications that are supported at NASA Dryden, this table look-up code generator is a crucial time saver. For example, the table look-up code for the X-31 simulation (225,000 data points and 200 dependent variables of a maximum of 5 dimensions) was generated and verified in 2 work weeks.

The table look-up code generator output is maximized for performance. When several parameters are functions of the same variable with identical breakpoint values, the table indices and interpolation ratios are calculated once and shared. Breakpoint values and deltas are hard-wired in the code to minimize array accessing. The style of the code and the type of array indexing can be selected by the user to optimize performance on a given platform. This approach results in more efficient code than using a generic set of library routines to perform each table look-up.

One innovation incorporated in the table look-up code generator is an especially efficient means of handling unevenly spaced breakpoints. Evenly spaced breakpoints offer speed of execution because table indices can be directly computed and the breakpoint interval is fixed and known. Unevenly spaced breakpoint table look-ups are not as straightforward. Classical approaches include consecutive “IF” tests, binary searches, and schemes that retain the last position in the table or assume that no more than one breakpoint boundary will be crossed for each simulation frame. The logic involved in determining the indices and the calculation of the breakpoint intervals all act to slow down execution.

The NASA Dryden approach for table look-ups with unevenly spaced breakpoints is to create an evenly spaced breakpoint array that is a superset of the unevenly spaced points (fig. 7). The index into this evenly spaced array can be directly computed. This index is then used in an equivalent breakpoint index array that provides pointers to the appropriate interpolation equation. Admittedly, the evenly spaced array may contain many elements, but overall memory requirements are little changed because the actual data tables are not affected. Execution speed almost matches that for an evenly spaced breakpoint table look-up. Appendix B shows sample code produced by the code generator for a one-dimensional table look-up using this method.

Block Diagram Coding

Aircraft models, particularly control systems, are commonly described and documented using block diagrams. NASA Dryden developed a standard method for implementation of block diagrams so that the resulting simulation code is easy to read, verify, and modify. A standard naming convention is used, and each block is uniquely named so that the code and diagram can be cross-referenced.

To implement block diagrams, each block is numbered sequentially. Then, the relationship between the input(s) and output(s) for each block is defined. All gains, limits, filter values, and other constants are initialized once from the background executive and stored in common blocks. This method allows these values to be modified by the user during simulation execution without recompilation.

Normally, the block inputs and outputs are also stored in common blocks. This method is extremely useful for diagnosing problems because each intermediate value can then be monitored or recorded. The naming convention for these variables begins with a one- or two-character identifier (for example, "P" for the pitch axis), is followed by the block number, and ends with a label indicating the variable type (input, output, gain, filter coefficient, and so forth). Coding the block diagrams in a consistent manner facilitates development and debugging.

Because one main NASA Dryden simulation activity involves the design and testing of control systems, this implementation approach has proved invaluable. The controls engineer is able to modify portions of the control system (gains, filters, limits) while in real time, make test runs, and evaluate results without having to stop the simulation and recompile. The intermediate values from any point in the block diagram can be monitored and recorded to help quickly identify problems in the system. This capability makes the most efficient use of simulator time.

Because block diagrams are currently implemented by the simulation engineer by hand, this implementation step is more efficient with a block-diagram code generator. Although many commercial products are available for this purpose, maintaining the NASA Dryden coding method was desirable. Therefore, the method used in the NASA Dryden simulations has recently been incorporated into a block-diagram code generator developed under contract. This code generator also automatically provides system documentation as well as block-level testing. For example, gain schedules and filter frequency responses are plotted.

Simulation Capabilities

Several capabilities have been developed for the NASA Dryden simulation software to enhance its ability to support flight research. These capabilities are implemented in each simulation based on project requirements.

Real-Time Recording and Monitoring

Real-time recording of time history data is an important feature of the simulation for debugging and analysis. Any variable in FORTRAN common can be selected and recorded at the simulation frame rate. The signal list for recording is specified in an ASCII data file that can be modified and reinitialized during a simulation session. A set of standard NASA Dryden data formats referred to as "GetData" formats exists.[5] These ASCII and binary formats are used for storing simulation data and flight data. Using common formats allows a common set of plotting and analysis tools to be used for simulation and postflight analysis.

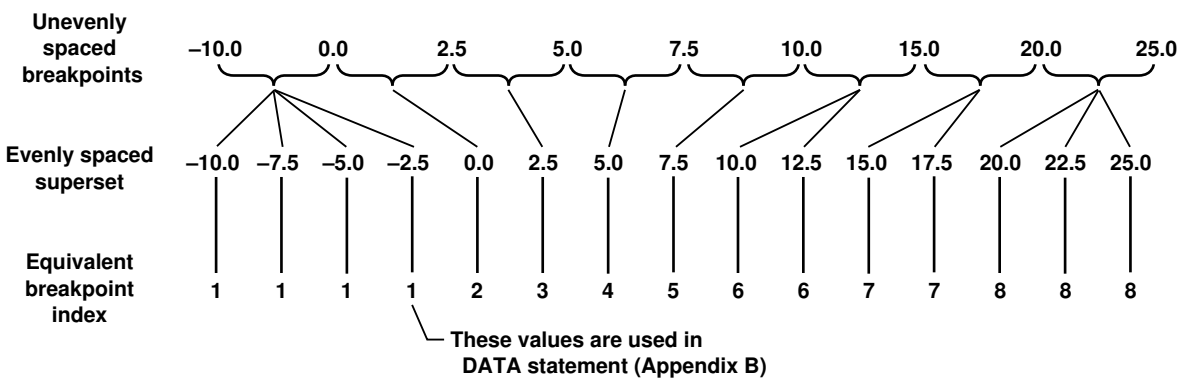


Figure 7. Unevenly spaced breakpoint scheme.

Monitoring of simulation data varies. Various simulation display pages are continually updated; however, more specialized and graphical displays are often needed. This requirement led to the development of a software toolkit for monitoring the simulation. This library allows a variety of stand-alone programs to access simulation data and provide custom displays. Variables are selected in a monitoring list and output in a buffering scheme implemented in shared memory. These stand-alone programs range from software strip charts to real-time displays of a three-dimensional aircraft and its flight trajectory. Many of the displays that run in the control room during an actual flight can also be run in the simulation laboratory during a simulation session.

Simulation Control Panel

The NASA Dryden real-time simulations are designed for single-user operation. For this purpose, a convenient simulation control panel with an array of lighted pushbuttons is provided with each simulation cockpit. Figure 8 shows a typical fixed-base cockpit with the adjacent simulation control panel. This control panel allows the pilot or engineer to select simulation options from within the cockpit and provides feedback with lights to indicate mode selection. The values for all the options that can be controlled from the panel are retained in software. A toggle switch is also provided.

Using the simulation control panel buttons in conjunction with the toggle switch allows the pilot to alter any of the initial conditions for the aircraft state. For example, the pilot can push the "reset" button to select reset mode, then push the "altitude" button and adjust the initial altitude by moving the toggle switch up or down and holding it until the desired altitude has been reached. Then the pilot pushes the "operate" button to fly from the new condition.

In addition to modifying initial conditions and changing the simulation mode, other options can be controlled from the simulation control panel. One of the pushbuttons is used to start and stop the strip-chart recorders. Another pushbutton is used in conjunction with the toggle switch to adjust the volume for the cockpit sound system. Buttons are provided to modify the graphics viewpoint, inject customized pilot inputs, change control system modes, and insert system failures.

Alternate Pilot Inputs

For testing purposes in batch-mode and real-time simulation, the user can specify substitutes for pilot inputs by setting up customized inputs on one of the generic simulation displays. A data file with the pilot input information can also be provided.

Several signals can be inserted in the pilot-command path. A generic display is provided in each simulation, allowing the user to select the shape, duration, and amplitude of the signal. For example, a square, sawtooth, or ramped pulse or doublet can be used for the pitch-stick, roll-stick, rudder-pedal, or throttle inputs. The user can specify the time to start the input, change direction

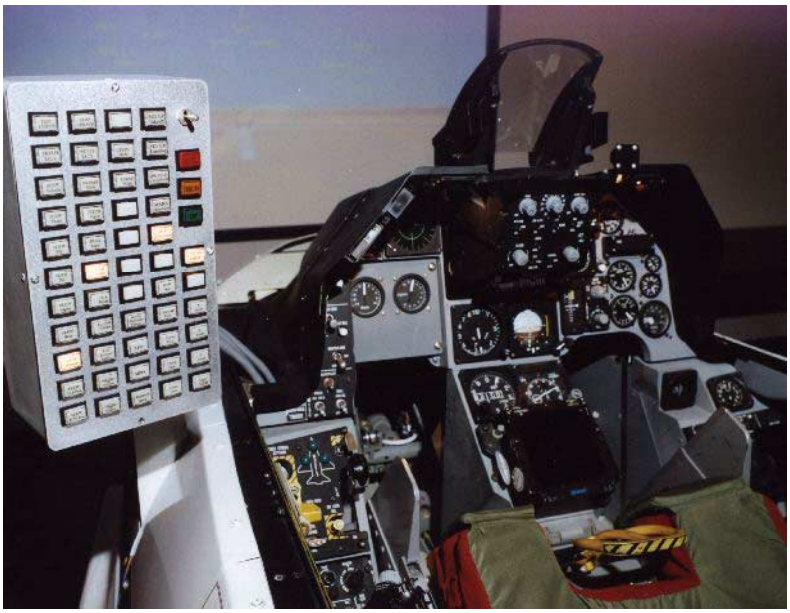


Figure 8. Simulation control panel.

for the doublet, and remove the input. Standard frequency sweeps are selected with the sine wave option.

Specialized Schroeder or Chirp functions are also available.[6,7] These functions are frequency sweeps that use different methods to compute the sweep based on the frequency content specified by the user. As an added feature, any of the above inputs can be superimposed on the control law commands or surface positions to facilitate frequency response or failure analysis.

A time history file can also be used to substitute for pilot inputs to the simulation. Because the pilot inputs during an actual flight are normally recorded, these same inputs can be played back in the simulation. Therefore, a comparison can be made between simulation and flight data for identical maneuvers. This method is frequently used to provide clues to improving the fidelity of the simulation models.

Automated Testing

Because verification and validation normally involve a lot of repetition, most of the NASA Dryden simulations include an automated testing capability. This feature was developed to allow repeatable tests to be performed without requiring a pilot or engineer. Script files are generated to perform multiple tests in succession and record the data.

The automated test capability includes a suite of test functions such as frequency sweeps, steps, biases, and multipliers. The type, shape, amplitude, and timing for these functions can be selected on the automated testing display. Several hooks are provided in the software to specify where these functions can be injected. For example, the input/output to the cockpit or control laws can be specified. This capability provides a consistent mechanism for failure insertion and mode switching during testing.

Atmospheric Disturbance Models

Disturbance models for winds, gusts, and turbulence are included in the simulation structure. These basic models are normally adequate for most simulation studies. The gust and turbulence models are implemented to calculate body-axis velocities. The gust model allows the user to include a gust along any body axis. The amplitude and frequency can be changed as required. The gust has a standard shape of 1 – cos. The simulation uses the Dryden form of the turbulence model defined in military specifications.[8] These models provide the researcher with the ability to assess the basic aircraft response to gusts and turbulence.

The wind model is implemented as a table look-up for wind magnitude and direction based on altitude. A user-defined time to ramp in the winds also exists. The wind velocities are added to the inertial velocities. The dynamic effects on the aircraft are modeled by adding an increment to angle of attack, angle of sideslip, and velocity based on the change in wind velocities for successive simulation frames. For example, the effect of flying the aircraft into a wind shear can be observed by ramping in a vertical wind component.

The more comprehensive Global Reference Atmospheric Model (GRAM) was implemented in real time for the NASP project.[9] However, the computational requirements are currently too high to include this model in every simulation.

Aerodynamic Model Modification

To facilitate improving the simulation fidelity, a feature referred to as the “delta aero” capability is included in several NASA Dryden simulations. This feature gives the user the ability to include increments or multipliers on the various components of the aerodynamic model. These modifiers are implemented as multidimensional tables with varying breakpoints. Normally, this capability is used in conjunction with flight data played back through the simulation. When the simulation response differs from the aircraft response to the same inputs, this difference can often be traced to inaccuracies in the aerodynamic modeling.

The “delta aero” capability allows quick changes to be made to the aerodynamic model in the simulation. This capability was used to perform parametric studies with the space shuttle simulation to determine how much uncertainty could be tolerated in the aerodynamic modeling. Recently, the “delta aero” tables were used to adjust the X-31 aerodynamics and simulate a quasi-tailless aircraft to ensure the control laws could correct for the reduced directional stability.

Three-Dimensional Solid Model Graphics

Each of the piloted simulations has a visual scene that is displayed on either a cathode ray tube display or large projection screen in front of the cockpit. In addition to an out-the-window view of the local Edwards Air Force Base runways, a three-dimensional solid-model view of the aircraft is included (fig. 9). This graphics option is selectable from the simulation control panel and allows the researcher to view a dynamic model of the aircraft while flying the simulation. The solid model includes moving control surfaces as well as a three-dimensional velocity vector representation. The velocity vector

originates from the aircraft center of gravity, and its length gives a relative indication of the aircraft velocity. A two-dimensional plane is displayed at the end of the velocity vector to indicate angles of attack and sideslip. Different colors are used for positive and negative angles of sideslip.

The ability to display the three-dimensional solid model was originally developed to allow visualization of aircraft motion during high-angle-of-attack maneuvers. This capability has also proven valuable for analyzing flight incidents. Researchers were able to gain insight into the X-29 departure from controlled flight by replaying flight data through this graphics program.

Distributed Systems

The NASA Dryden real-time simulations make extensive use of distributed systems. The simulations are designed to operate on most UNIX-based computers. Separate VME-based equipment is used for the analog and discrete interfaces to the simulation cockpit or to flight hardware. The simulation communicates to these systems using high-speed VME-to-VME shared memory devices.

The fixed-base simulation cockpits use an electromechanical stick and rudder pedal assembly to provide the required force-feel characteristics for the pilot. This interface is controlled by the Simulation Electric Stick (SES) software. A cockpit interface unit has been developed to process the remaining analog and discrete information going to and from the simulation cockpit. These programs are configured using software. A single ASCII file is used to set up memory locations for the data and to provide scaling information about the cockpit instruments, control stick, rudder pedals, and throttle(s). An additional central processing unit and sound board can be included in the VME card cage to provide cockpit sounds.

A simulation interface device was developed for interfacing the simulation to flight hardware. In addition to signal conditioning, this system provides circuit protection in both directions to prevent damage to either the flight hardware or the simulation computer. The simulation interface device was used for both the X-29 and F-18 HARV hardware-in-the-loop simulations.

Linear Model Output

Inclusion of a linearization capability has greatly enhanced the simulation as a powerful tool for flight research. Several output options are available that write the linearized matrices to files that are directly compatible with the various linear analysis tools in use at NASA Dryden. The linearization generates matrices for the state equation $\dot{x} = [A]x + [B]u$ and output equation $y = [C]x + [D]u$, where x and u are the state vector and control vector, respectively. In addition,

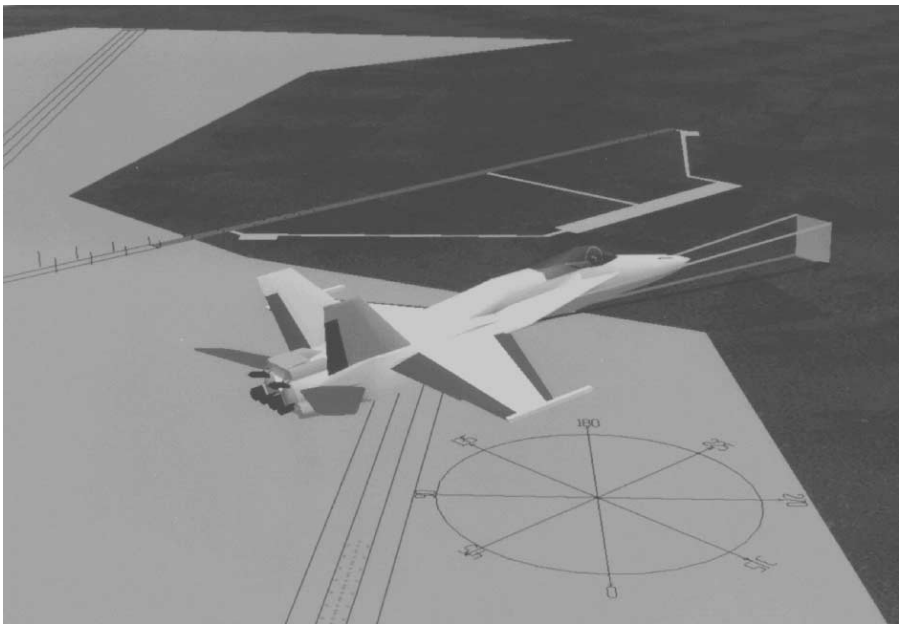


Figure 9. Three-dimensional solid model.

derivatives for the nondimensional aerodynamic coefficients are generated automatically for the same states and control surfaces.

The linearization will generate linearized matrices for any flight condition. The user can trim the aircraft to any initial condition in reset mode or place the simulation in hold during maneuvering. When the linearization option is selected from the linearization display page, the simulation immediately calculates the linearized matrices for the current aircraft state. These matrices are displayed on the linearization page and can be written to a file if desired, or the display pages can be fed directly to the printer. Scripts can be used to rapidly generate linear matrices for a large number of conditions. These matrices can then be used by linear analysis tools to provide quick safety-of-flight analysis.

Concluding Remarks

The NASA Dryden Flight Research Center has developed a flexible simulation software package that is capable of quickly and efficiently supporting flight research requirements and conceptual vehicle studies. The software and its inherent features have evolved through decades of experience with fixed-wing aircraft, including X aircraft, high-performance fighters, transport aircraft, sailplanes, and hypersonic vehicles. The same structure supports batch-mode, real-time pilot-in-the-loop, and flight hardware-in-the-loop simulation.

Capabilities such as linearization, the "delta aero" model, and automated testing with scripts provide researchers with the tools needed to quickly assess flight safety issues, make parametric and sensitivity studies, and perform verification and validation testing. The simulation graphics provide the pilot with visual cues and include a three-dimensional model to give researchers a keen understanding of the aircraft motion.

The NASA Dryden simulation is effective, capable, and easy to use. The system can be operated by a single individual through the use of the simulation control panel, keyboard entry, and mouse input. Ready access to simulation parameters and a flexible command line interface enhance the productivity of a typical simulation session. This simulation structure is used at other NASA centers, within industry, and at several universities and is available to qualified organizations.

APPENDIX A

Simulation Integration Method

```
      SUBROUTINE INTG
C
C.... ROUTINE FOR NUMERICAL INTEGRATION.  THIS ALOGRITHM IS THE
C     RESULT OF MODIFYING A 2ND ORDER RUNGE-KUTTA ALGORITHM BY
C     REMOVING THE SECOND DERIVATIVE EVALUATION AND REPLACING IT
C     WITH A PREDICTION OF THE DERIVATIVE BY A WEIGHTED AVERAGE
C     OF THE PREVIOUS END-FRAME AND MID-FRAME VALUES.
C
      IMPLICIT NONE

      INTEGER          I
      REAL*4           DF(13)
      REAL*4           DQQ(13)
      REAL*8           F(13)
      REAL*8           QQ(13)
      LOGICAL          LWEIGHT  /.TRUE./

      REAL*8           T,P,Q,R,V,ALP,BTA,THA,PHI,PSI,H,X,Y
      REAL             TDOT,PDOT,QDOT,RDOT,VDOT,ALPDOT,BTADOT,
     .                 THADOT,PHIDOT,PSIDOT,HDOT,XDOT,YDOT
      COMMON /DRVOUT/T,P,Q,R,V,ALP,BTA,THA,PHI,PSI,H,X,Y,
     .                 TDOT,PDOT,QDOT,RDOT,VDOT,ALPDOT,BTADOT,
     .                 THADOT,PHIDOT,PSIDOT,HDOT,XDOT,YDOT
      INTEGER          NEQN              ! NUMBER OF INTEGRANDS
      REAL                   HI,H2       ! HI = SIM FRAME TIME
      COMMON /INGDAT/NEQN,HI,H2          ! H2 = HI/2
C
      EQUIVALENCE      (F(1),T),    (DF(1),TDOT)
C
C.... SAVE CURRENT INTEGRAND AND DERIVATIVE VALUES.  INTEGRATE TO
C     FRAME MID-POINT USING DERIVATIVE VALUES FROM END OF PREVIOUS
C     FRAME.
C
      DO I=1,NEQN
         QQ(I) = F(I)
         DQQ(I) = DF(I)
         F(I) = F(I) + H2*DF(I)
      ENDDO
C
C.... EVALUATE DERIVATIVES AT MID-POINT.
C
      CALL DERIVC
C
C.... INTEGRATE FROM BEGINNING TO END OF FRAME USING DERIVATIVE
C     VALUE DETERMINED AT MID-POINT.  PREDICT DERIVATIVE VALUE
C     AT END OF FRAME USING WEIGHTED AVERAGE OF PREVIOUS END-POINT
C     AND MID-FRAME DERIVATIVE VALUES.
C
      DO I=1,NEQN
         F(I) = QQ(I) + HI*DF(I)
         IF(LWEIGHT)  DF(I) = 1.5*DF(I) - 0.5*DQQ(I)
      ENDDO
C
      RETURN
      END
```

APPENDIX B

Table Look-Up Code for Uneven Breakpoints

```
      SUBROUTINE TLU
C
C....  WRITTEN  03/10/95   15:55:30  BY THE CODE GENERATOR
C      PROGRAM 'LOOKUP, REV 1.9  '   NASA/DFRC
C
      DIMENSION            NALP9( 15)
C
      REAL                 ALPDEG
C
      REAL                 CLA,CLAA
      COMMON / ARODAT/     CLA,CLAA(9)
      REAL*8               T,P,Q,R,V,ALP,BTA,THA,PHI,PSI,H,X,Y
      COMMON / DRVOUT/     T,P,Q,R,V,ALP,BTA,THA,PHI,PSI,H,X,Y,
     .                     TDOT,PDOT,QDOT,RDOT,VDOT,ALPDOT,BTADOT,
     .                     THADOT,PHIDOT,PSIDOT,HDOT,XDOT,YDOT
C
      DATA NALP9 / 1, 1, 1, 1, 2, 3, 4, 5, 6, 6, 7, 7, 8, 8, 8     /
C
      ALPDEG = ALP * 57.29578   ! ANGLE OF ATTACK IN DEGREES
C
C....  COMPUTE INDEX OF GREATEST LOWER BOUND BREAK-POINT
C      AND INTERPOLATION RATIO FOR EACH INDEPENDENT VARIABLE.
C
C....  ALP9   : UNEVENLY SPACED, 'COMPUTED-GO-TO'
C                LIMIT WITHIN TABLE BOUNDARY
C      BRK PTS:-10.0    0.0      2.5      5.0      7.5
C               10.0    15.0     20.0     25.0
C      DEP VAR:CLA
C
      IF(ALPDEG.LE.-.100000E+02) THEN
      IALP9=  1
      RALP9=0.0
C
      ELSEIF(ALPDEG.GE.0.250000E+02) THEN
      IALP9=  8
      RALP9=1.0
C
      ELSE
      ALP9X=ALPDEG
C
      IB   =ALP9X*0.400000E+00 +0.500000E+01
      IALP9=NALP9(IB)
C
      GO TO ( 101, 102, 103, 104, 105, 106, 107, 108),IALP9
C
      101  RALP9=(ALP9X+ .100000E+02)*0.100000E+00
           GO TO  109
      102  RALP9=(ALP9X                  )*0.400000E+00
           GO TO  109
      103  RALP9=(ALP9X-0.250000E+01)*0.400000E+00
           GO TO  109
      104  RALP9=(ALP9X-0.500000E+01)*0.400000E+00
           GO TO  109
      105  RALP9=(ALP9X-0.750000E+01)*0.400000E+00
           GO TO  109
```

```
      106  RALP9=(ALP9X-0.100000E+02)*0.200000E+00
           GO TO  109
      107  RALP9=(ALP9X-0.150000E+02)*0.200000E+00
           GO TO  109
      108  RALP9=(ALP9X-0.200000E+02)*0.200000E+00
C
      109  CONTINUE
      ENDIF
C
C....  CLA    --> FCT(ALP9)
C
      CLA = RALP9*(CLAA(IALP9+1)-CLAA(IALP9)) + CLAA(IALP9)
C
      RETURN
      END
```

References

[1] Smith, John P., Schilling, Lawrence J., and Wagner, Charles A., *Simulation at Dryden Flight Research Facility From 1957 to 1982*, NASA TM-101695, 1989.

[2] Binkley, Robert L. and Mackall, Dale, *System Overview of the NASA Dryden Integrated Test Facility*, NASA TM-104250, 1992.

[3] Guckenberger, Dutch, Uliano, Kevin C., Lane, Norman E., and Stanney, Kay, "The Effects of Above Real-Time Training (ARTT) on Three Tasks in an F-16 Part-Task Simulator," 15th Interservice/Industry Training Systems and Educational Conference Proceedings, Nov. 1993, pp. 99–108.

[4] Mackall, Dale, Norlin, Kenneth, Cohen, Dorothea, Kellogg, Gary, Schilling, Lawrence, and Sheen, John, *Rapid Development of the X-31 Simulation to Support Flight-Testing*, NASA TM-104256, 1992.

[5] Maine, Richard E., *Manual for* GetData *Version 3.1: A FORTRAN Utility Program for Time History Data*, NASA TM-88288, 1987.

[6] Young, Peter and Patton, Ronald J., "Comparison of Test Signals for Aircraft Frequency Domain Identification," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 3, May–June 1990, pp. 430–438.

[7] Rabiner, L. R., "Chirp z-Transform Algorithm Program," *Programs for Digital Signal Processing*, IEEE Acoustics, Speech, and Signal Processing Society, 1979, pp. 1.6-1–1.6-13.

[8] U. S. Government, *Flying Qualities of Piloted Airplanes*, MIL-F-8785C, Nov. 1980.

[9] Justus, C. G., Alyea, F. N., Cunnold, D. M., Jeffries, W. R. III, and Johnson, D. L., *The NASA/MSFC Global Reference Atmospheric Model—1990 Version (GRAM-90), Part I: Technical/Users Manual*, NASA TM-4268, 1991.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1995	Technical Memorandum

4. TITLE AND SUBTITLE

Flight Simulation Software at NASA Dryden Flight Research Center

5. FUNDING NUMBERS

WU 505-68-81

6. AUTHOR(S)

Ken A. Norlin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Dryden Flight Research Center
P.O. Box 273
Edwards, California 93523-0273

8. PERFORMING ORGANIZATION REPORT NUMBER

H-2052

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-104315

11. SUPPLEMENTARY NOTES

Prepared for the American Institute of Aeronautics and Astronautics Flight Simulation Technologies Conference, Baltimore, Maryland, Aug. 7–10, 1995.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified—Unlimited
Subject Category 05

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The NASA Dryden Flight Research Center has developed a versatile simulation software package that is applicable to a broad range of fixed-wing aircraft. This package has evolved in support of a variety of flight research programs. The structure is designed to be flexible enough for use in batch-mode, real-time pilot-in-the-loop, and flight hardware-in-the-loop simulation. Current simulations operate on UNIX-based platforms and are coded with a FORTRAN shell and C support routines. This paper discusses the features of the simulation software design and some basic model development techniques. The key capabilities that have been included in the simulation are described. The NASA Dryden simulation software is in use at other NASA centers, within industry, and at several universities. The straightforward but flexible design of this well-validated package makes it especially useful in an engineering environment.

14. SUBJECT TERMS

Flight simulation; Simulation software

15. NUMBER OF PAGES

21

16. PRICE CODE

AO3

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited