COMPUTER-AIDED
DECISION-MAKING
FOR FLIGHT OPERATIONS
TECHNICAL REPORT NO. 3

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS
COMPUTER-AIDED DECISION-MAKING FOR FLIGHT OPERATIONS

Technical Report Number 3
Covering the Period January 1, 1975
to December 31, 1975

Principal Investigator: R.T. Chien

Contributors: P. Biochini
P. Davis
D. Finke
P. Fitzhenry
C. Jacobus
W. Rouse
M. Selander
S. Weissman

This work is supported by the Avionics Laboratory,
U. S. Air Force Systems Command, Wright-Patterson
Air Force Base under Contract AF F33615-73-C-1238.
TABLE OF CONTENTS

1. SUMMARY .................................................................. 1
   1.1. Introduction ................................................ 1
   1.2. The Aircraft Model .......................................... 2
   1.3. Computer Aided Decision Making for Flight Operations.. 4
   1.4. Other Developments ......................................... 5

2. COMPLEX AIRPLANE MODELS .......................................... 7
   2.1. Aerodynamic Simulation ..................................... 7
      2.1.1. Aircraft Being Simulated ......................... 7
      2.1.2. Simulation of the Aircraft ....................... 7
      2.1.3. Simulated Flight Test Methods .................... 8
      2.1.4. Discussion of Results ............................ 12
   2.2. Subsystem Simulation ....................................... 13
      2.2.1. Introduction ....................................... 13
      2.2.2. Fuel/Engine System Simulation ................... 14
      2.2.3. Electrical System Simulation .................... 19
      2.2.4. Hydraulic System Simulation ..................... 23
      2.2.5. Interaction Between Systems ..................... 27
   2.3. Controls and Displays ..................................... 30
   2.4. ARL Computer Operating Environment ....................... 34
      2.4.1. Introduction ....................................... 34
      2.4.2. Hardware Configuration ........................... 34
      2.4.3. PDP-11/40 Functions ............................... 37
      2.4.4. Software Configuration .......................... 40
         2.4.4.1. RT-11 Operating System ................ 40
         2.4.4.2. How MOVER, CORELOG, and RECOV Fit in
                    Simulation ............................... 40
      2.4.5. How MOVER Works ................................... 42
         2.4.5.1. Calling Conventions ..................... 42
         2.4.5.2. Input/Output ............................ 44
         2.4.5.3. Inter-Process Messages ................ 44
      2.4.6. Device Level Input/Output Specifics ............ 46
         2.4.6.1. Message Formats ......................... 46
         2.4.6.2. Plasma-Touch Panel .................... 49
      2.4.7. Examples ........................................... 49
      2.4.8. Conclusions ........................................ 50

3. The CADM System ................................................... 52
   3.1. Introduction and System Goals ........................... 52
      3.1.1. Multiple Failures ................................ 52
      3.1.2. Relations with the Pilot ........................ 52
      3.1.3. Generalized Correction Procedures ............. 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2. The Internal CADM Airplane Model</td>
<td>55</td>
</tr>
<tr>
<td>3.2.1. Structure for Components</td>
<td>55</td>
</tr>
<tr>
<td>3.2.2. Structure for Procedures</td>
<td>58</td>
</tr>
<tr>
<td>3.3. CADM Correction Procedures</td>
<td>58</td>
</tr>
<tr>
<td>3.3.1. The Correction of Failures</td>
<td>58</td>
</tr>
<tr>
<td>3.3.2. Generality of the Correction Procedures</td>
<td>59</td>
</tr>
<tr>
<td>3.3.3. Histories and Error Recovery</td>
<td>62</td>
</tr>
<tr>
<td>3.3.4. Priority, Conflicts and Protection</td>
<td>67</td>
</tr>
<tr>
<td>3.4. Detection Failures and Monitoring Corrections</td>
<td>68</td>
</tr>
<tr>
<td>3.4.1. Correction Monitoring Procedures</td>
<td>68</td>
</tr>
<tr>
<td>3.4.2. Demons and Parallel Processing</td>
<td>70</td>
</tr>
<tr>
<td>3.4.3. Failure Detection</td>
<td>72</td>
</tr>
<tr>
<td>3.4.4. Monitoring the Pilot's Actions</td>
<td>73</td>
</tr>
<tr>
<td>3.4.5. System and Component Reliability</td>
<td>73</td>
</tr>
<tr>
<td>3.5. Conclusions</td>
<td>74</td>
</tr>
<tr>
<td>4. THE SIGNIFICANCE OF CADM - AN OVERVIEW</td>
<td>75</td>
</tr>
<tr>
<td>4.1. The Generality of CADM</td>
<td>75</td>
</tr>
<tr>
<td>4.1.1. CADM's Knowledge Structure</td>
<td>76</td>
</tr>
<tr>
<td>4.1.2. Hierarchies of System Component Classes</td>
<td>77</td>
</tr>
<tr>
<td>4.1.3. System Parametric Definitions of Failures</td>
<td>79</td>
</tr>
<tr>
<td>4.2. The Adaptability of CADM</td>
<td>80</td>
</tr>
<tr>
<td>4.2.1. CADM as an Operator</td>
<td>80</td>
</tr>
<tr>
<td>4.2.2. Adaptability Demonstrated by Program Responses</td>
<td>81</td>
</tr>
<tr>
<td>4.2.2.1. Responses to Failures</td>
<td>81</td>
</tr>
<tr>
<td>4.2.2.2. Responses to Pilot Intervention</td>
<td>82</td>
</tr>
<tr>
<td>4.2.2.3. Responses to Procedure Failures</td>
<td>82</td>
</tr>
<tr>
<td>5. PILOT - CADM INTERACTION</td>
<td>84</td>
</tr>
<tr>
<td>5.1. Introduction</td>
<td>84</td>
</tr>
<tr>
<td>5.2. The Master Monitor Display</td>
<td>84</td>
</tr>
<tr>
<td>5.2.1. General Problems</td>
<td>84</td>
</tr>
<tr>
<td>5.2.2. Design</td>
<td>84</td>
</tr>
<tr>
<td>5.2.2.1. The Touchtone Keyboard</td>
<td>86</td>
</tr>
<tr>
<td>5.2.2.2. Status Information</td>
<td>86</td>
</tr>
<tr>
<td>5.2.2.3. The Multi-Function Display</td>
<td>89</td>
</tr>
<tr>
<td>5.3. Telling the Computer What the Pilot is Doing</td>
<td>89</td>
</tr>
<tr>
<td>5.3.1. The Problem</td>
<td>89</td>
</tr>
<tr>
<td>5.3.2. The Approach</td>
<td>89</td>
</tr>
<tr>
<td>5.3.2.1. Feasibility Studies</td>
<td>90</td>
</tr>
<tr>
<td>5.3.2.2. Results</td>
<td>92</td>
</tr>
<tr>
<td>5.4. Conclusions</td>
<td>92</td>
</tr>
</tbody>
</table>
6. THE COMMUNICATING SYSTEMS ......................................... 94

6.1. PDP-11 System and Communications .................................. 94
   6.1.1. PDP-11 System Improvements ................................ 94
   6.1.2. Alternative PDP-11 Operating Systems .................... 95
          6.1.2.1. The DEC Disk Operating System .................. 95
          6.1.2.2. The DEC Real Time System ....................... 95
          6.1.2.3. The DEC RSX-11 System ......................... 96
          6.1.2.4. The UNIX System ................................ 96

6.2. The CSL PDP-10/PDP-11 Communication Link ......................... 97
   6.2.1. The Channel ............................................ 97
   6.2.2. The Channel Drivers ................................... 98

6.3. The WPAFB/CSL Communication Link .................................. 99
   6.3.1. Implementation ....................................... 99
   6.3.2. Capabilities ......................................... 100
   6.3.3. Protocol ............................................. 101
   6.3.4. Internal Structure .................................. 102
   6.3.5. Testing .............................................. 102

APPENDIX 1. Aircraft Characteristics and Nomenclatures ............... 104
APPENDIX 2. Non-Dimensional Stability Derivatives and Coefficients 112
APPENDIX 3. Simulated Flight Test Conditions .......................... 124
APPENDIX 4. Analysis of the Simulated Aerodynamic Model ............ 126
1. SUMMARY

1.1. Introduction

The goal of this project is to construct a prototype system which demonstrates the feasibility of computer aided decision making during flight operations. The project consists of four phases as follows:

Phase 1. Analysis of flight operations and identification of tasks,
Phase 2. Computer aided decision making with simple tasks,
Phase 3. Complex tasks, and

The work reported here represents progress under Phase 3 of this project. This covers the period January 1, 1975 to December 31, 1975. The technical objective of Phase 3 is to achieve the goal of constructing a system for the purposes of demonstrating the feasibility of computer aided decision making in the realm of complex tasks. This system was completed on schedule and a demonstration was given on February 17, 1976 at the Coordinated Science Laboratory. The purpose of this report is to document the progress achieved in some detail for later reference. During the course of this work, a great deal was learned regarding problem-solving techniques for computer aided decision making, communication problems between various programs in a large software system, and man-machine interface considerations. These insights will undoubtedly prove to be invaluable to any person who is interested in the development of large scale software systems in general, and to people interested in automation in particular. Based on this belief, we have included in this report not
only results and achievements, but also an in-depth analysis of the related areas of investigation.

1.2. The Aircraft Model

In order to create a realistic environment for the development and testing of computer aided decision making system software, we have constructed an integrated software model of the functions of an aircraft and its operating environment. The aircraft simulated is a single seat, twin jet, variable-geometry, fighter plane. The engines develop 15,000 pounds of thrust each and the wings have five sweep settings. The flaps located on the trailing edges of the wings, can be extended to four positions. However, these can only be extended for wing sweep angles of 35° or less. The landing gear have two positions, namely, fully extended and fully retracted.

The aerodynamics simulation programs simulates all aircraft motions except yaw and side force. Also lateral motion is restricted to roll only. Inputs to the aerodynamic simulation come from the control stick, throttles and the subsystem simulation.

The aircraft subsystem model (AIRSYS) is the primary work area for the computer aided decision making system (CADM). Through it the CADM can investigate and act on its environment. AIRSYS provides CADM with numerical data on various systems and components of the aircraft and allows the CADM alternatives for the correcting of a failure when a system or subsystem malfunctions. The aircraft subsystems modeled in AIRSYS are the fuel/engine system, the electrical system and the hydraulic system.

The fuel/engine system is composed of two engines, two fuel tanks,
two drain pumps, one intertank pump, two engine driven fuel pumps, two electric fuel pumps, and four valves. The valve plumbing network allows one fuel tank to feed either or both engines. In addition, the system produces through the electrical system, sensor outputs and failure characteristics for the CADM to evaluate.

The electrical system is composed of two engine driven generators, two fuel pumps, two hydraulic pumps, 48 circuit breakers, all sensors, and all valve actuators. Failures in this system are caused by the incorrect opening of the individual circuit breakers for the AIRSYS components.

The hydraulic system is composed of one hydraulic fluid reservoir, two engine driven hydraulic pumps, two electrically driven hydraulic pumps, one accumulator pressure tank, eight valves, and the mechanisms for activating the wing sweep, the landing gear, and the flaps. Failures of components in this system can cause the landing gear, flaps, and wing sweep to be jammed in their current states. Also, the wing sweep can be jammed because of reduced hydraulic pressure in the accumulator pressure tank. Even with reduced pressure, however, the landing gear and flaps can still be operated hydraulically. Total system failure can result from the loss of the pumps, the loss of hydraulic pressure or the jamming of the valves.

The pilot, the aerodynamic simulation model, AIRSYS, and the CADM all interact through a shared data base (SDB). In addition, a Gremlin routine can set failures in AIRSYS by operating on the SDB. All three AIRSYS systems are heavily interconnected. Failures in one system could cause failures in another, thereby increasing the complexity of any
correction scheme. Cooperation between the systems and the duplication of efforts offers the CADM and the pilot a number of alternatives when responding to a component failure, hence, making the possibility of total system failure more remote, and also increasing the complexity of the solution task.

The hardware of the simulator includes a joystick, two throttles, a touchtone keyboard to input and request data, a monitor display to display present AIRSYS states, and a vertical situation display (VSD).

The pilot utilizes the touchtone keyboard to input changes and corrections into AIRSYS. Through the keyboard, the pilot can change the wing sweep angle, the flap angle, and the gear state. He can also investigate or change the current state of any AIRSYS component and can correct a failure through the use of the appropriate code input to the keyboard.

The VSD displays aircraft pitch and bank angles on an artificial horizon. Heading is displayed on a compass. Altitude, airspeed, rate of climb, and command information are displayed on the VSD by means of bar graphs, alphanumeric symbols, and a moving aircraft symbol.

1.3. Computer Aided Decision Making for Flight Operations

The purpose of this project is to develop the conceptual framework of a computer aided system which relieves the pilot from a high workload during flight operations. Part of this goal was achieved during Phase 2 with the development of the computer aided decision making system (CADM). During Phase 3 work was done to enable CADM to handle a wider and more complex class of problems in a more enriched domain than was the case during Phase 2. The CADM system which was implemented
operates to correct and compensate for failures during normal and degraded mode operations. The system maintains a flexible priority structure which allows it to correct multiple failures and, if necessary, to resolve internal conflicts, including those with the pilot.

The system constructs correction and monitoring procedures to specifically fit the failure, airplane configuration, and component allocation. The general correction procedures make the system more airplane independent.

The flexible DEMON system allows effective monitoring of on-going correction procedures. Using the DEMON system, the pilot's actions can be monitored in order to allow CADM to operate without interfering with the pilot actions. DEMONs also allow the re-evaluation of on-going correction procedures in order to upgrade to more efficient ones, or in the case of failure, to use system maintained histories to perform error recovery to construct alternate correction procedures.

1.4. Other Developments

Although the demonstration in Phase 3 was primarily a demonstration of the system's capability of decision making in degraded mode operations, a number of other advancements have been achieved. These achievements are instrumental to our success and they are quite useful in any large system where the communication between a large number of computer programs become essential.

The success of the system, in real time, depends upon the timely transfer of data and information between the subsystem and the CADM program. The exhibition of the solution depends on the timely transfer of the status
to the display program. These are but two isolated examples of the very high degree of interaction necessary for the coordinated intelligent behavior of the system. The successful implementation of our system and its speedy decision making performance is a direct result of the original organization of the overall system and the solution of the communication problem between programs. We believe that these techniques are of direct usefulness to other systems of similar magnitude.
2. COMPLEX AIRCRAFT MODELS

2.1. Aerodynamic Simulation

2.1.1. Aircraft Being Simulated

The aircraft being simulated is a single seat, twin jet, variable geometry, fighter aircraft. The engines develop 15,000 pounds of thrust each and the wings have five sweep settings; 15, 25, 35, 50, and 70 degrees. The flaps are located on the trailing edges of the wings and can be extended to four positions: 10, 20, 30, and 40 degrees. The flaps can be extended for wing sweep angles of 35 degrees or less, only. The landing gear has two positions; fully extended and fully retracted. A complete list of the aircraft’s characteristics can be found in Appendix 1.

2.1.2. Simulation of the Aircraft

Using standard techniques, as well as a non-dimensional stability derivative representation of a high performance aircraft, an aerodynamic model was derived. Appendix 2 lists the non-dimensional stability derivatives used. However, two important changes were made from the standard development. Lateral motion was reduced to the simplest motion; roll being the only lateral motion allowed. This not only simplified the aerodynamic model but also alleviated the need for rudder controls in the simulator hardware. The other change made was to consider only those derivatives that have a major effect on stability, neglecting those with only minor affect.

The resulting model is both simple enough to be a real-time simulation and complex enough to be a realistic simulation of an actual aircraft.
The aerodynamic model simulation program is written in the FORTRAN computer language. The program flow begins by initializing all the variables in the program. At this point, the non-dimensional stability derivatives are set to zero. After all the variables have been initialized, the program checks the angle of the wing. The program flow is routed to the appropriate section of the program containing the initial variable values for that wing angle. The non-dimensional stability derivatives are initialized to their particular values for the current wing sweep angle.

The first variables to be calculated are the coefficients of lift and drag. The second task of the program is the calculation of the aerodynamic forces and moments. Body-axis accelerations and angular velocities are computed next. Finally, by integrating the accelerations and velocities, the baby-axis velocities and Euler angles are found. Altitude is calculated from an initial altitude and the integration over time of the rate of climb. Velocity and the Euler angles are calculated in a similar manner.

2.1.3. Simulated Flight Test Methods

The simulated flight test of the aerodynamic model had two purposes. First, the test was used to determine that the simulator was responding to control inputs as an actual aircraft should. Also, by analyzing the flight test data, handling qualities were determined.

The stick-fixed flight test method was chosen to evaluate the handling qualities of the aerodynamic model. Although this method does not exactly duplicate the flying qualities of an aircraft controlled by a pilot, the stick-fixed modes of aircraft motion were used as parameters in specifying flying qualities criteria. Thus, through the use of controlled
tests, we obtained data that could be used to determine how the simulator would respond under a pilot's control.

Although, the simplification of the aerodynamic model reduced the number of stick-fixed modes that could be evaluated, the number of possible flight configurations to be tested, due to the inclusion of flaps, landing gear, and wing sweep in the model, was quite large. Appendix 3 is a listing of the conditions tested.

The 25° wing sweep angle was the only configuration for which complete graphical data were collected, i.e., flight tests were run for every possible flap angle, landing gear and c.g. position, and aircraft weight. Graphical data were also collected for elevator and aileron step input responses for the "clean" configuration of the remaining wing sweep positions. A "clean" configuration meant that the flaps and landing gear were not extended, the aircraft was at gross weight, the c.g. was at the reference position, and the aircraft was flying at 5000 feet and as close to 450 feet per second as was practical under the restraints of certain limitations in the testing facilities. The remaining conditions were run and observed on the plasma panel attitude indicator. Here, stability was confirmed if no divergence occurred, and some damping was observed.

Before the simulated flight tests began, the basic dynamic characteristics of the modeled aircraft were calculated. These results were then compared with those determined in the flight test. Some differences between the experimental and the predicted (calculated) data were expected since the predicted data were calculated using approximate methods.

The aerodynamic model was run on the Digital Equipment Corporation PDP 11/40 mini-computer at ARL. Because of limitations in the computer
data collection procedures and data plotting program, two separate methods were used in performing the simulated flight test. One method was used solely to observe the effects of control inputs. The other method was used to collect quantitative data for selected flight conditions. In the first method, control inputs were initiated through the use of a plasma screen touch panel. This panel also included an attitude indicator and numerical readouts for velocity, altitude, rate of climb, pitch angle, and bank angle, giving immediate feedback of results. In the second method, control inputs were initiated from the computer console keyboard. Using these inputs, the simulator program was run. Data were collected from it and stored in the computer. The data were then output to a line printer through a plotting program. Numerical data were also output. Figure 1 shows a typical computer plot. Every variable in each condition had its own plot.

The control inputs were a \(-0.02\) radian (rad) step input to the elevator (forcing the aircraft's nose up) and a \(+0.175\) rad step input to the ailerons (forcing the aircraft into a right roll). For each condition, the controls were activated when the simulator program was started. The starting point for the simulator was the calculated steady-state condition for that particular configuration. Two testing time periods were used: 20 seconds (sec) for the short period mode data and 480 sec for the phugoid and roll mode data. The initial flight condition for the "clean" configuration was the steady-state values of rate of climb and climb angle associated with an altitude of 5000 feet (ft.) and some initial velocity. The initial velocity was either 450 feet per second (fps) or 300 fps, depending on what particular aircraft response was being tested. It was not always possible to start the simulator program exactly at the initial velocity, so there
Figure 1. Typical computer data plot.

\[ u = 450 \text{ fps} \]
\[ \Lambda = 25^\circ \]
\[ \varsigma_e = -0.02 \text{ rad.} \]
\[ T = 0.8240 \text{ lb.} \]
were some small variations in the point where the tests were begun. Some of the flap and landing gear tests were begun at various other initial velocities.

In the testing of the AIRSYS subsystem, it was not possible to collect any quantitative data. However, it was possible to observe the functioning of the subsystems on the Master Monitor Display (MMD). By implementing failures into AIRSYS and observing the actions and reactions of the subsystems and subsystem components, the functioning of those systems and components of AIRSYS was observed and evaluated.

2.1.4. Discussion of Results

A complete analysis of the simulated flight test of the aerodynamic model is given in Appendix 4. The results of these tests show that the model is responding correctly to control inputs and is stable in all aircraft configurations and flight conditions. Analysis of the experimental longitudinal and lateral mode data shows good correlation with the predicted results. Thus, the aerodynamic model of the CADM simulator is a very dynamic and stable simulation of a variable geometry fighter aircraft. Flight experience on the simulator indicates that the flying task is fairly complex and has the ability to apply varying workloads on the pilot.

By observing the response of the AIRSYS subsystems and components to all possible types of failures, we have concluded that the subsystem is performing as designed in all cases.

Thus, the CADM simulator, aerodynamic model and subsystems, are performing as designed. The resulting simulator is a very complex tool for investigating CADM's ability to function in an aircraft environment, and the ability of the pilot and the CADM to work together.
2.2. Subsystem Simulation

2.2.1. Introduction

The aircraft subsystem model (AIRSYS) is the primary work area for the CADM. Through it the CADM can investigate and act on its environment. AIRSYS provides CADM with numerical data on various systems and components of the aircraft and allows the CADM alternatives to the correction of a failure when a system or system malfunction(s). The aircraft subsystems modeled in AIRSYS are the Fuel/Engine System, the Electrical System, and the Hydraulic System. A central program controls the flow through AIRSYS by cycling through a series of calls to subroutines responsible for simulating the various components of the subsystems.

These particular aircraft subsystems were chosen for a number of reasons. Since the general task area of the CADM project was degraded mode operations, AIRSYS had to simulate hardware readily identified with actual aircraft components. The chosen hardware had to be capable of failure modes, and if possible, alternate paths for the failure correction had to be provided. Thus, included in the Fuel/Engine System were two fuel tanks, alternate fuel flow routes, and numerous valves. Mechanically and electrically driven pumps were provided to the Fuel/Engine System and the Hydraulic System. Hydraulic and electric activation of the flaps and landing gear was also included.

Gremlin is the program which sets failures in AIRSYS. Through Gremlin, a failure can occur in any system or component. Failures occur on two levels. A soft failure is one which can be corrected by closing a circuit breaker or some similar action. A hard failure cannot be corrected by the CADM or the pilot. Gremlin, however, can change the
status of any level of failure.

2.2.2. Fuel/Engine System Simulation

The Fuel/Engine System is composed of two engines, two fuel tanks, two drain pumps, one intertank pump, two engine driven fuel pumps, two electric fuel pumps, and four valves. The valve plumbing network allows one fuel tank to feed either or both engines. In addition, the system produces, through the Electrical System, sensor outputs and failure characteristics for the CADM to evaluate. Figure 2 shows the interconnection of this hypothetical hardware.

Any of the components in the system can fail. Pumps and valves can jam in any current state, and since the center of gravity (c.g.) of the aircraft depends on whether the fuel tanks are empty, partially full, or full, the c.g. can be affected by incorrect draining of the fuel tanks. The engines produce thrust proportional to the fuel flow and the altitude. Engine failures are as follows. The engines can catch fire and produce reduced thrust. The engines can be destroyed and produce no thrust, or they can flame out also producing no thrust. Fuel flow, because of jammed valves and pumps can be cut off or continued when it is undesirable to do so.

Sensor outputs from this system are:

1. Fuel available in tanks
2. Fuel flow to engines
3. Measured thrust of each engine
4. Measured temperature of each engine's exhaust
5. Measured vibration modulation of each engine
6. Current states of pumps and valves.
Figure 2. Simulated Fuel/Engine System.
Table 1 summarizes the activities of the Fuel/Engine System simulation. The CADM, the pilot, AIRSYS, Gremlin, and the aerodynamic model interact through a Shared Data Base (SDB). Each reads the data it needs and writes the results of its operations on that data back into the SDB. The method of interaction of all AIRSYS programs and in this case, the Fuel/Engine System, with the SDB is the same. First, the entire SDB is read and any failures specified by Gremlin are acknowledged by setting jam flags for the specific components affected. Control settings set by the CADM and the pilot are implemented if not prevented by a Gremlin set jam. For example, the CADM may ask that a valve be closed. Regardless of whether the valve has previously been closed, the valve is set "closed" if Gremlin has not set a jam in that valve. If Gremlin has set a jam, the valve cannot be changed from its previous state and the CADM is informed that a failure has occurred.

The fuel flow in the plumbing network is a function of four variables. These are the availability of fuel, the state of the valves (open or closed), the state of the fuel pumps (on or off), and the amount of demand thrust. There are 156 possible routes for fuel flow in the network. The Fuel/Engine System simulation determines what amount of fuel flow the current plumbing network will allow, based on the valve states. A route for fuel flow is feasible if there is fuel available, the pump to force the fuel is on, and the valves controlling the path are open.

For the intertank fuel line, the direction of the flow is determined by the state of the intertank pump. The rate of flow is either zero or the maximum rate possible for the intertank pump. The
<table>
<thead>
<tr>
<th>PHASE</th>
<th>FUNCTION</th>
<th>PARAMETER</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUTS</td>
<td>STATES</td>
<td>PUMPS, VALVES, ENGINES</td>
</tr>
<tr>
<td></td>
<td>COMMANDS</td>
<td>PILOT, CADM</td>
</tr>
<tr>
<td></td>
<td>CONSTRAINTS</td>
<td>OTHER SYSTEMS, GREMLIN</td>
</tr>
<tr>
<td>CALCULATES</td>
<td>FUEL SYSTEM</td>
<td>DIRECTION, QUANTITY</td>
</tr>
<tr>
<td></td>
<td>ENGINE SYSTEM</td>
<td>OPERATION, SENSOR DATA</td>
</tr>
<tr>
<td></td>
<td>INFLUENCE</td>
<td>OTHER SYSTEMS, GREMLIN</td>
</tr>
<tr>
<td>OUTPUTS</td>
<td>ENGINE SYSTEM</td>
<td>TEMPERATURE, VIBRATION, THRUST</td>
</tr>
<tr>
<td></td>
<td>FUEL SYSTEM</td>
<td>FLOW, QUANTITY, LOCATION</td>
</tr>
</tbody>
</table>

Table 1. Fuel/Engine System Simulation Functions
state of the pump is under the control of Gremlin, the CADM, and the pilot, or AIRSYS, if the tank auto-level function is on.

The drain fuel lines in the tanks have only one direction of flows overboard from the source tank. The state of the pump is determined by Gremlin, the CADM, and the pilot. The flow rate is the maximum possible for the pump.

The direction of fuel flow to the engines is from the source tanks to the engines. The states of the valves and pumps are determined by Gremlin, the CADM, and the pilot. The flow rate through the engine pump is either zero or the rate specified for the demand thrust by the pilot's throttle position. The fuel flow rate through the plumbing network is either zero or the rate required by the engine pumps. If two pipes have fuel flow into the same pump, each carries half the demand fuel load.

With the preceding fuel network description, we can now describe the engine simulation. The amount of actual thrust produced by each engine, called measured thrust, is equal to the demand thrust if no engine failures exist. If the fuel flow is zero or the engine is destroyed, the measured thrust is zero. A destroyed engine state is non-recoverable. Only Gremlin can set an engine destroyed state. A flameout of the engine, caused by fuel starvation is recoverable. The procedure for recovery is to re-establish fuel flow to the engine and perform an engine restart. Either the pilot or the CADM may issue an engine restart command. AIRSYS responds by attempting ignition of the engine and then resetting the engine restart function. A restart can not be initiated if the Electrical System has failed. An engine flameout state can be set by Gremlin or can occur from fuel starvation brought on by pilot or CADM errors. It can also be caused
by the failure of components in the Fuel/Engine System that in turn leads to fuel starvation.

The measured thrust will be half the demand thrust if an engine is on fire. The engine fire state is set by Gremlin and is reset by turning off the fuel flow to the engine. The engine is now in a flameout state. To obtain thrust again, the pilot or the CADM must start fuel flow back into the engine and issue an engine restart command. In this simulation, no penalty is given for attempting to restart an engine after a fire.

A general cleanup of state flags is performed at the beginning of every AIRSYS "write" into the SDB and specific items for which AIRSYS is responsible are updated. Examples of such items from the Fuel/Engine System are engine temperature, engine vibration, and fuel flow.

Through the SDB, both the pilot and the CADM have access to the information and the means necessary to correct any soft jam caused failures. The number and complexity of failures to this particular system is quite large, as is the number of possible corrections. Through such complex manipulation of this and the other systems of AIRSYS, we can apply varying workloads on the pilot and the CADM.

2.2.3. Electrical System Simulation

The Electrical System is composed of two engine driven generators, two fuel pumps, two hydraulic pumps, 48 circuit breakers, all sensors, and all valve actuators. Failures in this system are caused by the incorrect opening of the individual circuit breakers for the AIRSYS components.

Outputs from this system are as follows:

1. Status of the electrical pumps and valves
2. Status of the circuit breakers

3. Status of the generators


The structure of the Electrical System is shown in Figure 3. Table 2 summarizes the activities of this system.

The Electrical System contains two relay connected generator subsystems, one connected to each engine. An engine driven generator supplies power to a distributing bus through a master circuit breaker. Each bus distributes power to 23 AIRSYS components, each of which is connected to the bus through a circuit breaker. If one of the generators cannot supply power to its bus, either because of engine failure, failure of the generator itself, or the jamming of the master circuit breaker in the open position, the other generator can supply power to both buses with no reduction in current to any component, but only if the interbus relay is closed. If the relay will not close, the components connected to the failed generator's bus will fail.

Failure of the entire Electrical System results in the loss of most of the components in the AIRSYS simulation. However, depending on the status of the components at the time of failure, such a loss need not be fatal to the aircraft.

Again, both the pilot and the CADM have access to all outputs from this system, and the means to correct any failure with the exception of one caused by a hard jam. Gremlin has access to all parts of the Electrical System and can set failures in any component. All components in this system, including the circuit breakers, can be either hard jammed or soft jammed.
Figure 3. Simulated Electrical System
<table>
<thead>
<tr>
<th>PHASE</th>
<th>FUNCTION</th>
<th>PARAMETER</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUTS</td>
<td>STATES</td>
<td>CIRCUIT BREAKER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALVES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUMPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOTORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENERATORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INTERBUS RELAY</td>
</tr>
<tr>
<td></td>
<td>COMMANDS</td>
<td>PILOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CADM</td>
</tr>
<tr>
<td></td>
<td>CONSTRAINTS</td>
<td>GREMLIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OTHER SYSTEMS</td>
</tr>
<tr>
<td>ACTIVATES</td>
<td>FUEL/ENGINE</td>
<td>PUMPS</td>
</tr>
<tr>
<td></td>
<td>SYSTEM</td>
<td>VALVES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SENSORS</td>
</tr>
<tr>
<td></td>
<td>HYDRAULIC</td>
<td>PUMPS</td>
</tr>
<tr>
<td></td>
<td>SYSTEM</td>
<td>VALVES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SENSORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOTORS</td>
</tr>
<tr>
<td></td>
<td>ELECTRICAL</td>
<td>INTERBUS RELAY</td>
</tr>
<tr>
<td></td>
<td>SYSTEM</td>
<td>CIRCUIT BREAKERS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GREMLIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OTHER SYSTEMS</td>
</tr>
<tr>
<td></td>
<td>INFLUENCE</td>
<td>GREMLIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OTHER SYSTEMS</td>
</tr>
<tr>
<td>OUTPUTS</td>
<td>STATES</td>
<td>CIRCUIT BREAKERS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALVES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUMPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOTORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENERATORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INTERBUS RELAY</td>
</tr>
</tbody>
</table>

Table 2. The Simulated Activities of the Electrical System.
2.2.4. Hydraulic System Simulation

The Hydraulic System is composed of one hydraulic fluid reservoir, two engine driven hydraulic pumps, two electrically driven hydraulic pumps, one accumulator pressure tank, eight valves, and the mechanisms for activating the wing sweep, the landing gear, and the flaps. The structure of the Hydraulic System is shown in Figure 4. The activities of this system are listed in Table 3. Failures of components in this system can cause the landing gear, flaps, and wing sweep to be jammed in their current states. Also, the wing sweep can be jammed because of reduced hydraulic pressure in the accumulator pressure tank. Even with reduced pressure, however, the landing gear and flaps can still be operated hydraulically. Total system failure can result from the loss of the pumps, the loss of hydraulic pressure, or the jamming of the valves. Neither the flaps nor the landing gear can operate asymmetrically, although the back-up mechanical actuators can be used to activate one or more of the elements while the others are operated hydraulically. The hydraulic fluid travels through the valves and pumps in a manner similar to the fuel in the Fuel/Engine System. Sensor data from the components of this system are provided through the Electrical System for the use of the pilot and the CADM.

The major component of the Hydraulic System is the accumulator pressure tank. Two engine and two electrically driven pumps pressurize this tank. The electric pumps work only when the engine pumps are off. If both pumps on one side are jammed, the pump on the other side cannot pressurize the tank to maximum pressure. Without maximum pressure in the accumulator tank, the wing sweep mechanism will not function. One pump can, however, supply enough pressure to operate the flaps and landing gear.
Figure 4. The Simulated Hydraulic System.
<table>
<thead>
<tr>
<th>PHASE</th>
<th>FUNCTION</th>
<th>PARAMETER</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUTS</td>
<td>STATES</td>
<td>DEMAND WING SWEEP ANGLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEMAND FLAP ANGLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEMAND LANDING GEAR STATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUMPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALVES</td>
</tr>
<tr>
<td></td>
<td>COMMANDS</td>
<td>PILOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CADM</td>
</tr>
<tr>
<td></td>
<td>CONSTRAINTS</td>
<td>GREMLIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OTHER SYSTEMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AIRSPEED</td>
</tr>
<tr>
<td></td>
<td>CALCULATES</td>
<td>PRESSURE SYSTEM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACCUMULATOR TANK PRESSURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INDIVIDUAL LINE PRESSURES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLUID RESERVOIR LEVEL</td>
</tr>
<tr>
<td></td>
<td>INFLUENCE</td>
<td>GREMLIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LEAKS</td>
</tr>
<tr>
<td></td>
<td>ACTIVATES</td>
<td>HYDRAULIC SYSTEM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WING SWEEP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLAPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LANDING GEAR</td>
</tr>
<tr>
<td></td>
<td>INFLUENCE</td>
<td>GREMLIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OTHER SYSTEMS</td>
</tr>
<tr>
<td></td>
<td>OUTPUTS</td>
<td>STATES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRESENT WING SWEEP ANGLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRESENT FLAP ANGLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRESENT LANDING GEAR STATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACCUMULATOR TANK PRESSURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLUID RESERVOIR LEVEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INDIVIDUAL LINE PRESSURES</td>
</tr>
</tbody>
</table>

Table 3. The Simulated Activities of the Hydraulic System.
In order to activate one of the elements of the Hydraulic System, the valve controlling the flow of fluid to the mechanism is opened. If the valve is jammed closed, the mechanical back-up actuator must be used to activate the element. If both the hydraulic and mechanical actuators are jammed, the subsystem failure flag is set "on" and the subsystem is failed in order to prevent asymmetric activation of elements in that subsystem. Thus, if both actuators of, for example, the nose gear are jammed, the entire landing gear subsystem is failed. The same is true to the flap subsystem.

Total failure of the Hydraulic System does not have the immediately serious consequences of the total failure of the other two systems in AIRSYS. However, total failure of the Hydraulic System can have long range effects on mission success. A wing sweep angle of 70° without flaps and perhaps without landing gear may not be serious in a cruise condition, but later in the mission when a landing is necessary, this aircraft configuration can have a serious effect on safe mission completion.

CADM and the pilot have access to outputs from the Hydraulic System and the means to correct any non-hard jammed failures. Outputs from this system are as follows:

1. Accumulator tank pressure,
2. Status of the flaps, landing gear, and wing sweep subsystems,
3. Status of the pumps and valves,
4. Fluid reservoir level, and
5. Overall Hydraulic System status.
2.2.5. Interaction Between Systems

All three AIRSYS systems are heavily interconnected. Failures in one system could cause failures in another, increasing the complexity of any correction scheme. Also, because of these interactions, recognizing the source of a series of failures becomes a very difficult problem. For example, the loss of an engine, as shown in Figure 5 results in the loss of the engine driven fuel and hydraulic pumps as well as the loss of the electrical generator connected to that engine. The loss of the generator can lead to the loss of a significant number of individual components and possibly the loss of the entire Electric and Hydraulic Systems. A listing of the components that might be lost due to the failure of an engine is given in Table 4. Presenting the CADM with this sizable list of failures forces the CADM to try to find a single component that could cause all the failures and to attempt to repair or by-pass that component. Failing that, the CADM would then have to determine a hierarchy of repair; deciding which components are essential to mission success and should be fixed first, and which are less important and can wait to be fixed. The use of such a complex arrangement taxes the abilities of the pilot and the CADM to function separately and cooperatively.

The interaction between systems applies not only to system failures but also to component back-ups. The electric pumps back-up both the engine driven fuel and hydraulic pumps. Mechanical actuators back-up the hydraulic actuators for the flaps and landing gear subsystems. Cooperation between the systems and the duplication of efforts offers the CADM and the pilot a number of alternatives when responding to a component failure, making the possibility of total system failure more
COMPONENT FAILURE
LEFT ENGINE DESTROYED

FUEL/ENGINE SYSTEM

LEFT MECH. FUEL PUMP STOPS

FUEL TO LEFT ENGINE CUT OFF

A/C THRUST REDUCED

ELECTRICAL SYSTEM

LEFT GENERATOR STOPS

INTERBUS RELAY OPEN?

ALL ELECTRICAL EQUIPMENT POWERED BY LEFT GENERATOR STOPS

HYDRAULIC SYSTEM

LEFT MECH. HYDRAULIC PUMP STOPS

HYDRAULIC PRESSURE REDUCED

WING SWEEP MECHANISM INOPERATIVE

Figure 5. Failure interaction between the engine and hydraulic systems.
JAMMED EQUIPMENT
DUE TO LEFT ENGINE DESTRUCTION

<table>
<thead>
<tr>
<th>FUEL/ENGINE SYSTEM</th>
<th>ELECTRICAL SYSTEM</th>
<th>HYDRAULIC SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>• FUEL VALVES (2)</td>
<td>• LEFT GENERATOR</td>
<td>• HYDRAULIC VALVES</td>
</tr>
<tr>
<td>• INTERTANK PUMP</td>
<td></td>
<td>• LEFT MECHANICAL HYDRAULIC PUMP</td>
</tr>
<tr>
<td>• REAR DRAIN PUMP</td>
<td></td>
<td>• LEFT ELECTRICAL HYDRAULIC PUMP</td>
</tr>
<tr>
<td>• LEFT MECHANICAL FUEL PUMP</td>
<td></td>
<td>• WING SWEEP MECHANISM</td>
</tr>
<tr>
<td>• LEFT ELECTRICAL FUEL PUMP</td>
<td></td>
<td>• FLAPS *</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LANDING GEAR *</td>
</tr>
</tbody>
</table>

* POSSIBLE LOSS - DEPENDING ON VALVE STATES AT TIME OF FAILURE

Table 4. The possible components lost due to engine failure.
remote, and also increasing the complexity of the solution task.

Another aspect of the interactions between the systems is shown in Figure 6. In this Figure, the pilot has initiated a simple, straightforward command: turn on the forward drain pump. But, in forgetting that the auto-level function is on, he has inadvertently caused a failure, the draining of both fuel tanks. This type of failure, one caused by unthinking commands to the subsystem, is another area where the interaction between the systems becomes important. While the CADM is unlikely to make this type of error for the same reasons the pilot would, inattention or haste, the possibility exists that the CADM could make this type of error, or the pilot, in erring in this manner, could interfere with an ongoing CADM solution to a failure.

2.3. Controls and Displays

The hardware of the simulator includes a joystick, two throttles, a touchtone keyboard to input and request data, a monitor display to display current AIRSYS states, and a vertical situation display (VSD). Figure 7 shows the simulator hardware arrangement. The monitor display (MMD) is described elsewhere in this report. The VSD displays aircraft pitch and bank angles on an artificial horizon. Heading is displayed on a compass. Altitude, airspeed, rate of climb, and command information are displayed on the VSD by means of bar graphs, alphanumeric symbols, and a moving aircraft symbol. Figure 8 shows how these devices are arranged on the VSD screen.

The joystick and throttles send signals to the SDB through an A/D converter. The aerodynamic model program reads the SDB and acts on these signals. The results of the reactions to these inputs are returned
SUBSYSTEM ACTIVITY

PILOT

TURN ON
FORWARD
DRAIN PUMP

ELECTRICAL
SYSTEM

FORWARD
DRAIN PUMP
TURNED ON

FUEL / ENGINE
SYSTEM

FORWARD
DRAIN PUMP
ON

AUTO LEVEL ON?

FORWARD TANK BEGINS
TO DRAIN

INTERTANK PUMP
TURNED ON

INTERTANK
PUMP ON

FORWARD FUEL
QUANTITY SENSOR

FORWARD FUEL
QUANTITY INSTRUMENT

AFT FUEL
QUANTITY SENSOR

AFT FUEL
QUANTITY INSTRUMENT

BOTH TANKS
DRAIN

Figure 6. System activity for starting forward drain pump.
Figure 7. The simulator I/O hardware.
Figure 8. The vertical situation display.
to the SDB where the VSD and the MMD read and then display them to the pilot on their screens. The CADM also has access to this data.

The pilot utilizes the touchtone keyboard to input changes and corrections into AIRSYS. Through the keyboard, the pilot can change the wing sweep angle, the flap angle, and the gear state. He can also investigate or change the current state of any AIRSYS component and can correct a failure through the use of the appropriate code input to the keyboard.

2.4. ARL Computer Operating Environment

2.4.1. Introduction

This report deals with work done primarily between March 1975 and March 1976 at the Aviation Research Laboratory at Willard Airport in preparation for phase four of the CADM contract. In this phase, the flight simulation, along with some closely associated computation, will be moved onto hardware at ARL. The CADM programs will still be run on the PDP-10 at CSL, and the communication with the PDP-10, as well as the simulation-related computation will be performed on a PDP-11/40 minicomputer at ARL. In particular, it is the software written for the PDP-11/40 that is the primary subject of this report.

2.4.2. Hardware Configuration

Shown in Figure 9 are the hardware facilities at ARL. Early plans for phase four included connection to CSL through a synchronous 9600-baud modem, a DU-11 synchronous interface, and a KG-11 block check arithmetic device for error detection. Later developments have modified this such that the telecommunications with CSL will be done with a Bell 202-D
Figure 9. The Aviation Research Laboratory hardware configuration.
asynchronous modem and a DC-11 UART.

A smaller version of the PDP-11 (a model 10) will act as an interface between the analog control circuitry of the GAT-2 and the PDP-11/40. Several DR-11 general device interfaces are used to make the connections for data transfer between the PDP-11's and the GAT-2. Each of the DR-11 interfaces consists of enough logic to transfer 16-bit words in two directions, and to handle the protocol of the PDP-11 Unibus. A DL-11 is a UART used to control a local terminal. The link between the PDP-11/40 and the PDP-11/10 will be made with two DL-11's, back-to-back, each interfaced to their respective PDP-11's.

A plasma panel will be one of the primary means of communicating with the pilot. A graphics-oriented controller built at ARL will be used to drive the plasma panel. This controller is capable of reading lists of encoded commands from the core memory of the PDP-11/40, and executes these commands by selectively turning on, or turning off individual points of the 512 x 512 dot matrix of the panel. Closely associated with the plasma panel is the touch panel. This is a 16 x 16 matrix of light beam sources and sensors. The combined effect of the plasma-touch panel is to allow a display to be piece-wise constructed and modified on the plasma panel, and to let a program on the PDP-11/40 sense when the pilot points to, or touches a specific part of the display. As another possible means of communicating with the pilot, we have a Votrax speech synthesizer which generates the phoenemes to pronounce messages to the pilot.

There are a number of other I/O devices connected to the PDP-11/40, most notably a VT05 video console terminal, and a RK11
cartridge disk controller with two disk drives. The disks and the terminal are used mainly as support to the operating system, and for program development. These are also used for loading and initializing programs for the simulation, but their actual use during the course of the simulation is limited. The disk is used in constructing some tables for display to the pilot, and the VT05 is used to notify the operator of error conditions in the simulation.

Finally, a few words may be in order about the architecture of the PDP-11/40. This is shown in a general way in Figure 10. The PDP-11 processor has a 16-bit address field, byte-level addressing, and two 8-bit bytes per word. This gives a virtual 32K word address space. The Unibus, which connects the processor both with core memory and with I/O device interface registers, has an 18-bit address field. The box labeled "MM" is the memory management unit, which allows the 2 - 16 address space of the processor to be mapped onto any part of the 2 - 18 address space of the Unibus. Upon hardware initialization, and unless specifically altered using the memory management hardware, the low 28K word addresses of the processor are mapped onto the low 28K word address of the bus. These are usually core memory. The upper 4K word address of the processor are mapped onto the upper 4K word address of the bus, which are the I/O device registers.

2.4.3. PDP-11/40 Functions

As mentioned in the introduction, one of the major functions of the PDP-11/40 during the simulation will be to maintain communications between the various facilities at ARL and CSL. In some cases, this amounts to sending a continuous stream of a set of current-values along a one-way channel. In other cases, only exceptional data, i.e. a change in a
Figure 10. The PDP-11/40 UNIBUS architecture.
variable, or an error condition, will be sent. In any case, it is of
great advantage to be able to easily alter the specifics of the communica-
tions format, such as the content, or the frequency with which updates
should be sent. Experience has shown that these can vary from one day,
or one hour, to the next during stages of development.

In addition to communication, the PDP-11/40 will be performing
a number of computations closely associated with the simulation itself.
One of these tasks, for instance, will be to provide the CADM-pilot
interface, as MASTER MONITOR has in phases two and three. We expect
to have (and indeed already have had) a number of different people
contributing their efforts to the software system for the 11/40. A
well-defined, proven interface between these sub-tasks will be necessary.
For this reason as well as many of the standard arguments for higher-
level-language programming, we decided to write as much of the software
as possible in FORTRAN. Much of the work that has been done to date has
been in the direction of creating an environment, using assembler language
routines, which will enable FORTRAN program to perform the above functions.

During the course of the simulation, and invisibly to the
FORTRAN programs, a record of the significant events in the simulation
will be kept. This record, or log, contains such information as which
subprogram is accessing which common variables, what I/O operations are
occurring, the order in which events are taking place, and samples of
certain critical variables. A set of the most recent 128 log entries
are kept available at all times, and can be examined at the end of the
simulation. This is especially useful in case the software fails, because
the log can be examined to determine the events leading up to the failure.

2.4.4. Software Configuration

2.4.4.1. RT-11 Operating System

During the simulation itself, very little external software support will be necessary. During program development, however, and for the purposes of loading and starting programs, the aid of a host operating system is next to essential. We are running under RT-11, supplied by Digital Equipment Corporation. RT-11 is a small, fast operating system designed for real-time applications. It is a foreground-background system, that is, two tasks may be supported simultaneously. The foreground task always executes if it is not blocked, otherwise the background task may execute. RT-11 also provides primitives for inter-task communication and synchronization.

2.4.4.2. How MOVER, CORELOG, and RECOV Fit in Simulation

The working environment is provided to FORTRAN primarily through a subroutine called MOVER. Critical variables in the simulation are kept in a Shared Data Area, and may be accessed though calls to MOVER. Likewise, calls to MOVER are used to initiate I/O operations in a device-independent manner (see Figure 11).

Each call to MOVER may result in the generation of a log entry. This depends on whether a switch on the PDP-11/40 console is set, so the log facility of MOVER can be turned on and off at will during the simulation. The log entries are sent to the background as messages. Running in the background is a program called CORLOG which maintains the log. The log is kept in core memory that is accessible only with the memory-
Figure 11. The PDP-11/40 system software.
management hardware, thereby guaranteeing that the log will survive nearly any software crash. Yet another background program, RECOV, can be loaded and run which formats and prints the log on the line-printer.

2.4.5. How MOVER Works

2.4.5.1. Calling Conventions

Figure 12 shows some sample calls to the MOVER module. In part (a) MOVER is called with seven parameters. The first parameter identifies the subprogram and the statement-within-subprogram of the call for purposes of the log. The next five indicate the source, destination, and length of the data is to be moved. The last parameter is a returned value that is a serially assigned transaction number. Each of the source and destination can be specified as local variables. Shared Data Area variables, or an external device. If both the source and destination are internal to the PDP-11/40, the data is moved, a message is constructed and sent to the background, and a return is made to the calling FORTRAN routine, the desired function having been accomplished. If, on the other hand, the transaction involves an external device, the return to the calling FORTRAN routine is made once the I/O transaction has been initiated and entered in the active queue.
INTEGER VAR, DEV, SDA, TI, TO, RI, RO
DATA VAR/0/, DEV/1/, SDA/1/, TI/0/, IDO/0/, IDI/1/
DATA TO/1/, RI/2/, RO/3/
C MAIN PROGRAM FOR GADM 11/40 ROUTINES
C
C INITIALIZATION
C
CALL QINIT
CALL MOVER(ID1, DEV, TI, SDA, 3, 6, ITNUM)
GO TO 50
10 CONTINUE
CALL MOVER(IDO, DEV, TI, SDA, 3, 6, ITNUM1)
C TWO TRANSACTIONS ALWAYS ACTIVE
CALL INTRUD
CALL GAT
CALL WAIT(60)
50 CALL CQUE(ITNUM)
IF (ITNUM .NE. 0) GO TO 50
CALL LOGCOM
CALL FCQE
ITNUM=ITNUM1
GO TO 10
END

Figure 12. An example demonstrating calls to MOVER.
2.4.5.2. Input/Output

While execution of the FORTRAN routine continues, the I/O transaction is serviced by interrupt routines. The FORTRAN routines can check on the completion status of a previously issued transaction as in part c) of Figure 12. The parameter ITNUM is returned as zero if the transaction was completed, and is unchanged otherwise. When the interrupt routine for a device completes a transaction, the queue element for that transaction is removed from the active queue and placed in the completed queue. Two more routines are available to the FORTRAN programs: LOGCOM and FCQE. For each element in the completed queue, a log message is built and sent to the background signifying the completion of the transaction. FCQE returns elements in the completed queue to the free pool (see Figure 13).

It may seem that it would be more straightforward to send the completion message directly to the background task from the interrupt routine that completes the transaction, and thereby remove the need for the intermediate completed queue. The problem here is that the message facility is provided by the RT-11 operating system. Such system requests are not, in general, supported from within interrupt routines, since they are not, themselves either foreground or background tasks, but are instead another class of hardware-scheduled (i.e. interrupt) tasks. Furthermore, it is necessary to act in accordance with the rules set down by RT-11, since RT-11 and the FORTRAN runtime system are still being used to perform some input-output functions to disk and the console terminal.

2.4.5.3. Inter-Process Messages

One of the major objectives in the design of this system was to
Figure 13. I/O interrupt software system.
give the FORTRAN foreground program enough power to exercise the required control over a variety of I/O devices without being unnecessarily concerned about the specifics of the device. In general, an I/O device is considered to be a one-way serial sender or acceptor of data. Two-way devices are logically thought of as two separate devices.

Associated with each device is an initialization routine and an interrupt service routine. The initialization routine is called as each transaction is started. When the interrupt routine completes a transaction, a new one will be started from the active queue if one is pending. Otherwise the interrupt routine will become inactive. Especially for input devices, it is advantageous to keep at least one transaction pending at all times.

Through the initial stages of development, quite a variety of devices are, and have been, in use with MOVER.

2.4.6. Device Level Input/Output Specifics

2.4.6.1. Message Formats

Our first attempt at I/O through MOVER was with the PDP-10 at CSL through a remote-terminal line. The devices were operated by sending and receiving ASCII characters. The two devices were called TI and TO, for input and output, respectively. On the PDP-10, a FORTRAN program did I/O to what looked like a user terminal in fixed field format. In going from the PDP-10 to the PDP-11, each block of data was sent as a synchronizing character followed by a fixed field (15 in FORTRAN) of numerals. The five numerals were sufficient to represent all the values of a 16-bit PDP-11 word. A second special character was used to preface each numeric field, since non-printing filler characters are sometimes sent between lines by
the PDP-10 terminal facilities. Including the non-printing characters, 8 to 10 bytes are sent across the line for each 16-bit value transmitted.

Values were sent to the PDP-10 by converting them to decimal numerals and "typing" them in to a FORTRAN program on the PDP-10. This is essentially the same scheme that was used for communication between the PDP-10 and the PDP-11 as CSL for phase two. At 1200 baud, the effective data transfer rate is less than 14 words per second in each direction.

The original plans for phase four of CADM called for a 9600-baud synchronous link to the PDP-10. Using this hardware, the plan was to continuously send fixed sets of current values of all the relevant variables. Since the synchronous communication facilities would be operated in transparent mode, a data transfer rate of over 500 16-bit words per second could be had in each direction. This would be adequate with a very simple protocol, and with very little data packing and unpacking. Since it has become apparent, now, that the synchronous hardware will not be available, the following scheme has been devised to get improved effective data transfer rates using the asynchronous hardware, all with a reasonably limited investment of time and effort.

First, instead of transmitting ASCII numerals, a set of 64 printable characters will be used, giving, in effect, a radix 64 number system shown in Figure 14. A full 16-bit value can then be transmitted with three ASCII characters, giving an effective data rate of over 35 words per second. Also, considerable packing of data will take place. Since three six-bit values give two more bits of information than necessary for a 16-bit word, the length (one, two, or three characters) of a word-value can be encoded with the data. For instance, four boolean values which fit
SIX-BIT CODE

16 BIT WORD VALUES

FROM
PDP-10

TO
PDP-10

Figure 14. Data packing for transmission.
logically together can be encoded with the length (one) and sent efficiently as a single character. Otherwise, it would be necessary to encode all information into packages of 16 relevant bits. The advantage of this six-bit scheme is that it allows the use of the standard terminal I/O facilities on the PDP-10, accessible to the higher-level programming languages.

The anticipated data transmission requirements are such that a fixed set of values will be transmitted from the PDP-10 to the PDP-11. The set of values will be a list of contiguous values in the PDP-11 Shared Data Area. These will be sent in a block that is begun with an ASCII (line-feed) character, and terminated by an ASCII (carriage-return). Each word-value consists of one to three characters. Data will generally take place only when a change in the values of certain variables takes place. The message block will still be delimited by a (line-feed) and a (carriage-return), but will consist of two characters to specify an index, and one to three characters for a value.

2.4.6.2. Plasma-Touch Panel

Performing output to the plasma panel is quite easily done, since it involves simply copying data into a buffer and sickness the controller on it. A request for data from the touch-panel is also fairly simple, since the controller just indicates that data is available when the screen has been touched, and presents the x- and y-coordinates in a register.

2.4.7. Examples

In the summer of 1975, we set out to construct a demonstration that would exercise the major facilities of the PDP-11/40 system. The following scheme was designed: Positional data for a primary aircraft
and some number of intruder (traffic) aircraft, was generated by the PDP-10 in real time. This positional data was such that the primary aircraft flew around a pre-defined course. The data on the airplanes' positions were sent to ARL on the asynchronous telephone line as ASCII numerals. A map of the course, along with the positions of the moving aircraft was displayed on the plasma panel. Figure 12 is the FORTRAN main-line code used for this demonstration. The subroutine GAT drew the course and the primary aircraft (seen as a triangle). INTRUD drew the intruder aircraft (squares). NORMAL and QINIT performed some system initialization. A picture of the display is shown in Figure 15. The simulation log for this demonstration is included as an appendix.

2.4.8. Conclusions

The overall software system design has, thus far, shown itself to be a good one. The I/O system has survived sweeping changes in the specifications of the data transfer for the phase four demonstration. Our initial plans to have a large number of tasks close to the simulation performed on the PDP-11/40 have been pared somewhat. Given this knowledge, a somewhat simpler system might have been designed. The ideas of keeping important variables in a Shared Data Area, and keeping a log of the simulation have already shown themselves to be valuable debugging tools. The extra time required for these does not seem to be a problem.
Figure 15. The plasma panel display.
3. THE CADM SYSTEM

3.1. Introduction and System Goals

3.1.1. Multiple Failures

During Phase III work has been done to enable CADM to handle a wider and more complex class of problems in a more enriched domain. The CADM system which was implemented and will be described operates to correct and compensate for failures during degraded mode operations. The failures may occur in the fuel and/or electrical subsystems.

Within this domain, certain goals of performance were established. Among these was the ability of CADM to handle multiple failures. This includes the cases in which failures occur simultaneously as well as those in which failures occur while CADM is monitoring the progress of on-going correction procedures. This involves CADM overseeing the allocation of aircraft components. To do this, the system must also have a variable priority structure relating the relative importance of correcting a class of failure to the mission profile and available equipment.

3.1.2. Relations With the Pilot

Another goal is to improve CADM's relationship with the pilot. CADM's overriding concern is that the pilot must be able to have full control over the airplane's capabilities in order to successfully carry out the mission. CADM must be careful not to undo pilot actions. To do this, the pilot's actions are monitored. In order to keep the pilot informed of what CADM is doing and is trying to accomplish, CADM must provide to the pilot information concerning CADM detections, corrections and perceptions of failures within the aircraft. However, the CADM
system should not overburden the pilot with voluminous of extraneous information. The information should be available if the pilot desires it, but it should not be forced upon him.

3.1.3. Generalized Correction Procedures

Because a goal of the CADM project is to investigate and develop a computer system to aid a pilot, an aim is to try to make the system more general. This means that the techniques used should not be geared to correcting a fixed class of errors on a fixed airplane type. To accomplish this, the present system introduces this generality in two areas. First, the airplane model is treated as factual data rather than being explicitly coded into the program. Second, the correction procedures are generalized to model component and instrument types rather than a specific switch or dial, with programmed knowledge of how these component types operate and affect the operation of the airplane.

The overall system outline is depicted in Figure 16. The various portions of the system will be described in detail in the following sections. As new data is received from the common database, the information is automatically processed by special purpose procedures (called DEMONS). Among the tasks accomplished by the DEMONS are the monitoring of on-going correction procedures and pilot actions. A check is made to determine if failures exist and if so, are they already being corrected by the pilot or CADM actions. The failures to be corrected are ordered. The system then attempts to construct a correction procedure (and corresponding monitoring procedure). If successful, actions are taken. The success or failure of the procedure is then monitored as new information is received. During
Figure 16. The CADM software system.
all of the steps, information about CADM actions is relayed to the pilot. The system which will be described is implemented in MACLISP on a PDP-10 computer and allows real time interaction with the airplane models and pilot.

3.2. The Internal CADM Airplane Model

In order to increase CADM's applicability by reducing ad hoc programming of correction procedures for a specific airplane configuration, a more general method of representing the airplane and available system action is employed.

3.2.1. Structure for Components

In this system the models of the physical components present and their properties are stored in a semantic net. Examples of portions of the net are shown in Figures 17 and 18. The component classes are grouped together by common characteristics such as the role in the aircraft or power source. In Figure 17, for example, PUMPS are partitioned by the two separate power sources into ELECTRICAL-PUMP and MECHANICAL-PUMP. There are three usage types of electrical pumps, INTER-TANK-PUMP, DRAIN-PUMP and ELECTRICAL-FUEL-PUMP. ELECTRICAL-FUEL-PUMP and MECHANICAL-FUEL-PUMP are grouped to form the class of FUEL-PUMP. Properties can be associated with any node of the net. All of the links in Figure 17 are of the IS-A type (as in an ELECTRICAL-FUEL-PUMP IS-A type of FUEL-PUMP). None of the nodes shown in Figure 17 are used to represent a specific piece of equipment. Such nodes do exist, an example of one being depicted in Figure 18. This node represents a particular electrical fuel pump. The various links are used to structure attributes about this pump as well as
Figure 17. Semantic net depicting instruments.
Figure 18. Semantic node representing a specific electrical fuel pump.
to relate its role in the entire airplane system. In this case, ELECTRICAL-
FUEL-PUMP3 is an ELECTRICAL-FUEL-PUMP, ELECTRICALLY powered by BUS2 via
circuit breaker BREAKER53. Its function is to pump fuel to ENGINE1 and
has an associate on/off switch, SWITCH 42. (Nodes with labels having numerical
suffixes indicate particular components).

By knowing any specific component, it is possible to climb through
the net to determine any other relevant instruments.

3.2.2. Structure for Procedures

Associated with any node could be procedural information concerning
a component or component class. The two most common cases are PREDICATE
and ACTION attributes. The PREDICATE case points to procedures to deter­
mine the state of the component(s). The ACTION case specifies procedures
to alter the state. Generally, this type of information is associated
with how to work with a component class, rather than with a specific
component.

By associating these procedures with higher levels classes, such
as ELECTRICAL-INSTRUMENTS, for example, a greater degree of generality is
obtained. A non-component dependent procedure is used to indicate how many
specific instruments operate. The automatic generation of this net is
done once before runtime with the resulting net being saved for future use.

3.3. CADM Correction Procedures

3.3.1. The Correction of Failures

One of CADM's main responsibilities while operating in a degraded
mode aircraft is the correction (or compensation) of failures. In order
to broaden the applicability of the CADM system, the correction procedures
are built up by CADM from elemental actions. By investigating cause-effect relationships, relevant actions are grouped together.

3.3.2. Generality of the Correction Procedures

As has been stated, the potentially relevant action procedures are generally associated with a component class. The system searches through the airplane model semantic net by use of the functions MAKE and IS. Basically, MAKE isolates action procedures and is used to construct correction procedures while IS is used to examine the state of the world to determine whether some condition is true or false for a group of components.

To illustrate the operation of the system, a continuing example of a flamed out engine caused by fuel strangulation due to a power failure will be discussed. In order to correct a flame-out which has occurred in ENGINE1, the command

(MAKE (TEMPERATURE NORMAL) ENGINE1)

indicates that a procedure to normalize engine temperature should be used. The system would check the data structure in order to determine how temperature could be observed and altered. It would find the potentially relevant functions on the semantic net under TEMPERATURE:

| ACTION | NORMAL --> TEMP-NORMAL  |
| LOW    --> TEMP-LOW     |
| PREDICATE: NORMAL --> TEMP-NORMAL-P |
| LOW --> TEMP-LOW-P |

Now, function TEMP-NORMAL may be essentially made up of the following steps:
As will be discussed, this nesting could go down to any level.

Whenever a MAKE returns (at any level), several types of information are present:

1. A linear program segment (possibility NIL) indicating what steps and intermediate sensory checks should be made in order to bring about the desired results.

2. A structure called a HISTORY summarizing the results of the construction effort. Also included in the history is information indicating why certain steps are in the plan segment as well as what to do if a continuation of the search is necessary. This is used during error recovery and will be described later.

3. Optional procedures indicating what to do upon successful completion or failure of the procedure. This may indicate such things as inferences which may be made or directions which should be pursued.

As IS call would just return a HISTORY, possibly indicating which sensors were used to determine values. As can be seen from the example, the arguments to a MAKE are a class and desired effect (e.g., (VALVE ON) or (TEMPERATURE NORMAL)) and a list of components for which the effect is to be made true. This second argument could also be a HISTORY returned from a previous MAKE which would cause a different correction procedure to be formulated.

An example of the nesting which could occur is shown in Figure 19. Each indentation represents a lower level function call. So, to make the TEMPERATURE NORMAL, there must be POSITIVE fuel and the restart must be ON. To make the flow POSITIVE, fuel must be AVAILABLE, the valves and fuel pumps which service the downed engine with fuel must be ON. The
(MAKE (TEMPERATURE NORMAL)---)

(MAKE (FUEL-FLOW POSITIVE)---)

(MAKE (FUEL AVAILABLE)---)
(MAKE (VALVE ON)---)

(MAKE (POWER ON)---)

(MAKE (BUS ON)---)
(MAKE (BREAKER ON)---)

(MAKE SWITCH ON)---)

(MAKE (FUEL-PUMP ON)---)

(MAKE (FUEL-PUMP-MECHANICAL ON)---)

(MAKE (FUEL-PUMP-ELECTRICAL ON)---)

(MAKE (POWER ON)---)
(MAKE (SWITCH ON)---))

(MAKE (RESTART ON)---)

Figure 19. Calling sequence for correcting temperature.
power must be ON for any of the electrical instruments to operate.

Whenever a MAKE is used, an explicit IS is done first. If the IS determines that the desired result is already true, then the action function is not evaluated and, therefore, there would be no more nested calls to MAKE. In Figure 19, if the system, when trying to MAKE fuel flow POSITIVE into an engine, determines (using an IS) that it already is POSITIVE, the nested calls to MAKE up to the call to MAKE the restart ON would not be generated. The level of nesting is determined by the failure and the current state of the aircraft. Only actions corresponding to actual anticipated needs would be computed.

When a MAKE returns a program segment, the calling function incorporates and combines it into larger programs. Figure 20 indicates the plan segment which could be returned from function calls of Figure 19. Indicated are the segments corresponding to the MAKE's which created them. How this plan is executed and monitored will be discussed in the following section.

3.3.3. Histories and Error Recovery

It is not reasonable for the system to assume that just because a possible correction procedure could be constructed, that the procedure will always be effective. There must be methods which would allow new and different procedures to be formulated. The reformulation should be able to take into account any new information as well as the knowledge of previously unsuccessful correction attempts. For maximum flexibility, it is not desirable to have the search constrained to such techniques as backtracking or starting over. Rather, some record of the decisions made should be available, allowing the construction procedures latitude in how
(TURN LEFT-BUS-BREAKER ON)

(SENSOR LEFT-BUS ON)

(TURN LEFT-ELECT-FUEL-PUMP-BREAKER ON)

(TURN LEFT-ELECT-FUEL-PUMP ON)

(SENSOR LEFT-FUEL-FLOW POSITIVE)

(TURN LEFT-RESTART ON)

(SENSOR LEFT-TEMPERATURE NORMAL)

Figure 20. Planning segment returned from calls of Figure 19.
to continue. The structure called the HISTORY serves this purpose.

The form of a HISTORY is shown in Figure 21, with an example in Figure 22. The first element is equivalent to the first argument to MAKE a component, component class or sensor type and a desired state (e.g., (VALVE ON), (TEMPERATURE NORMAL)).

The next element is a list of lists indicating the current or expected state of the particular equipment being considered. The first list gives components with the desired state, the second list contains negative cases. The third list contains those components which have been allocated to other users (i.e., other CADM jobs or the pilot) as well as who controls them. The last element of the list are defective components which cannot be put into the desired state because they are inoperable.

In Figure 22, valves V1, V2, and V4 are to be turned ON. This HISTORY indicates that V1 and V2 are already ON while V4 is OFF and cannot be altered because circuit breaker B2 is OFF and is under the control of CADM job MONITOR6.

The next element of a HISTORY contains a list of relevant information such as histories or variables to be saved. The total information necessary to recreate the decision process can be obtained because each HISTORY can include an arbitrary number of histories. The depth of histories saved would generally parallel the depth of MAKES used to construct the correction procedure. In Figure 22, the histories saved include POWER and SWITCH histories for the valves. Note that the POWER HISTORY contains BUS and BREAKER histories of its own.

The last element of a HISTORY is a pointer indicating the continuation point of an ACTION procedure. It is not necessary to restart
FORM OF HISTORY

(((COMPONENT-CLASS-NAME)
  (DESIRED-STATE))
(POSITIVE-CASES)
(NEGATIVE-CASES)
(PROTECTED-CASES)
(JAMMED-CASES))

(HISTORY) --- (HISTORY)
  AND/OR (VARIABLE-LISTS-TO-BE-SAVED))
TAG)

Figure 21. The form of a HISTORY.
A HISTORY of the status of valves V1, V2 and V4.

Figure 22.
from this position if the procedure is reentered. It serves merely as a guide.

3.3.4. Priority, Conflicts and Protection

In order to correct multiple failures the system maintains a flexible priority structure indicating in what order corrections should be made. While a correction is being made, all components which are to be used are allocated by the system to the correction procedure. The affected equipment is said to be protected. Pilot usage of components also causes protections to be instituted. Before any equipment is to be used, a check is made to ensure that no protection violations will occur.

In some cases, an ACTION procedure may return and indicate that no correction procedure is possible because the needed components have been previously allocated to other on-going correction procedures (or the pilot). In this case, an arbitrator program is called to determine if it is possible to reallocate equipment to ensure that the highest priority failure is being corrected. This may necessitate suspending an existing correction procedure in order to free some needed equipment.

Occasionally, a correction procedure for a failure will be constructed which is not the best possible (with respect to expected speed of correction) but is the best available. This may be because the "better" equipment has been allocated for other purposes. In this system, the correction procedure which was determined would be initiated but if the "better" equipment becomes available before the correction has been complete, the procedure would be reevaluated to determine if it is possible to upgrade into one incorporating the "better" equipment. The DEMON mechanism for accomplishing this will be described in the following sections.
By employing the correction approaches described here, it is possible to correct multiple failures, maintain control of allocation of equipment, and construct error correcting procedures from general cause-effect relationships based upon airplane operations.

3.4. Detecting Failures and Monitoring Corrections

Whenever correction procedures are constructed, the plan produced must be executed. It is not realistic to have CADM implement all of the steps and then assume that the correction has taken place. The operation of the airplane must be monitored to ensure that the desired (and expected) effects occur. During correction and monitoring phases, it may be necessary to perform sensor checks which would indicate whether the correction is progressing successfully.

3.4.1. Correction Monitoring Procedures

To allow this type of monitoring, CADM takes the correction steps and creates a monitoring procedure. The monitoring program is automatically constructed for each failure to be corrected. The monitoring procedure allows the correction to be spread out over an arbitrary time period. System actions can be delayed until information verifying the effect of previous steps is obtained.

The general form of the monitoring procedure is shown in Figure 23. As in all LISP CONDITIONALS, the first element of each line is evaluated. If there is a positive response (i.e., non-NIL), the rest of the line is evaluated and the function is exited. The functions in this figure referred to as SENSOR CHECK indicate that some requirement which can be sensed has been met (e.g., POSITIVE fuel flow). The BODY portions may be any steps
CORRECTION MONITOR

(COND ((SENSOR CHECK n) (BODY) SUCCESS)
  ((LOWER LIMIT n)
   (COND ((> TIME (to + 4t)) FAIL)
     (T SUCCEEDING)))
  ((SENSOR CHECK n-1) (BODY)
   (RESET LOWER LIMIT n))
  \ldots
  ((LOWER LIMIT i) ---).
  \ldots
  ((SENSOR CHECK 1) ---)
  (T (BODY) (SET LOWER LIMIT)))

Figure 23. The form of the monitoring procedure.
of procedure which are to be executed at some future time. The LOWER-LIMIT portions are used to restrict the inspection in the conditional to areas above (or before) the set lower limit. Initially, there is no set lower limit. If the evaluation passes to the current LOWER-LIMIT, a check may be made to ensure that reasonable time delays have not been exceeded.

Figure 24 depicts the conditional portion of a monitoring procedure which would be constructed given the plan segment of Figure 20. At first, there is no lower limit. When first evaluated, none of the sensory checks should succeed because it was the absence of positive sensor values which caused additional action to be inserted into the plan. So, the last line (beginning with T) would be executed, initiating the plan and setting the LOWER-LIMIT to 1.

As time proceeds, the monitoring procedure is reentered and the conditions are reexamined. Eventually, the LEFT-BUS would be observed to be ON. At this time further actions (to institute flow and reset the LOWER-LIMIT) can be taken. This would continue until the LEFT TEMPERATURE is observed to be in the normal operating range, at which time a successful correction would be reported.

Note that the sensors are checked from the front of the conditional to the end, while the last actions specified are expected to be executed first. Because of this, positive sensor readings will avoid the execution of steps which have become irrelevant. These procedures are tailored by the system for each specific correction plan. They can have any number of intermediate checks, flexible actions and time delays.

3.4.2. Demons and Parallel Processing

The monitoring procedures which have just been discussed are
(COND
  ((SENSOR LEFT-TEMP NORMAL)
   SUCCESS
  )
  ((LOWER LIMIT 3)---)
  ((SENSOR LEFT-FUEL-FLOW POSITIVE)
   (TURN LEFT-RESTART ON)
   (RESET LOWER-LIMIT 3))
  ((LOWER LIMIT 2)---)
  ((SENSOR LEFT-BUS HOT)
   (TURN LEFT-ELECT-FUEL-PUMP-BR ON)
   (TURN LEFT-ELECT-FUEL-PUMP ON)
   (RESET LOWER LIMIT 2))
  ((LOWER LIMIT 1)---)
  (T (TURN LEFT-BUS-BR ON)
     (RESET LOWER LIMIT 1)))

Figure 24. Monitoring condition procedure for plan segment of Figure 20.
continually reevaluated by including them in the DEMON mechanism. Conceptually, DEMONS allow procedures to be associated with a specific input datum. Any change in data causes the related procedures to be invoked. All data can have any number of procedural DEMONS. The DEMONS can be added or removed easily by a programmer or, as is the case for the monitoring of corrections, by the system itself.

For each datum the order in which the DEMONS are evaluated can be easily modified. Each DEMON has a priority number associated with it. The ones with the higher priority are invoked first. There is a variable lower threshold beneath which DEMONS will not be executed. This DEMON approach in some sense simulates a type of parallel processing for analyzing the new data.

### 3.4.3. Failure Detection

In the CADM system, the DEMON procedures are keyed to be activated by changes in sensor values, switch values and time. These are all quantities provided by the common database. The DEMONS which are sensor driven are used to do such tasks as automatically quantizing values (e.g., determining if an engine temperature is LOW, NORMAL or HIGH) or computing the value of useful system variables such as differences of fuel available in the tanks.

Components and component classes have associated with them modifiable indications of what failures are to be considered as well as the definition of the failures. These definitions are usually in terms of the DEMON-produced quantized variables. With this structure the system is able to detect failures. When a failure is detected, CADM performs
some checks to ensure that the failure is not already being corrected by a 
CADM correction procedure. It also checks to determine whether the pilot 
is expected to make the correction. The responsibility for correction 
will be passed off if CADM is unable to construct a correction procedure. 
Usually the cause for this would be that the pilot is using some equipment 
deemed necessary for correction by CADM.

3.4.4. Monitoring the Pilot's Actions

The DEMONs which are switch driven are used to monitor pilot 
actions. In this implementation no attempt is made to ascribe intentions 
to the pilot actions. All that is maintained is a record of what equipment 
the pilot is using and when he last used it. Using these DEMONs, CADM is 
able to determine whether the pilot is using equipment which CADM has 
allocated for its own correction procedures. If this is the case, CADM 
can suspend the violated correction procedure, and try to construct one 
that does not interfere with the pilot's actions.

3.4.5. System and Component Reliability

The switch actuated DEMONs are also used to monitor airplane 
switch changes such as failed circuit breakers. Using DEMONs, diagnostic 
procedures could be performed to check the reliability of a doubtful 
component.

The time driven DEMONs are used to monitor events which are 
expected to change over a given time interval (as opposed to a switch 
setting which can be altered at any time). This is needed in two cases 
which have been discussed. In the first case, when monitoring and 
executing a correction procedure, the system, during the interval of
correction, needs to observe whether sensor quantities change or not in order to determine whether the procedure is succeeding.

In the second case, when the system desires to upgrade a correction procedure to one which is more efficient, the availability of a component over a fairly short time interval must be monitored.

3.5. Conclusions

In the preceding sections, the current version of the CADM system was described. This system is able to correct or compensate for failure in the fuel and electrical system of a twin engine jet aircraft. The system maintains a flexible priority structure which allows it to correct multiple failures and, if necessary, resolve internal conflicts.

The systems constructs correction and monitoring procedures to specifically fit the failure, airplane configuration, and component allocation. These general correction procedures make the system more airplane independent.

A flexible DEMON system allows effective monitoring of on-going correction procedures. Using the DEMON system, the pilot's actions can be monitored in order to allow CADM to operate without interfering with the pilot actions. DEMONs also allow the re-evaluation of on-going correction procedures in order to upgrade the more efficient ones, or in the case of failure, to use system maintained histories to perform error recovery to construct alternate correction procedures.
4. THE SIGNIFICANCE OF CADM - AN OVERVIEW

Research is ultimately judged both within and without its specific academic and theoretical fields. This means that CADM will be judged not only by its theoretical richness within the field of artificial intelligence, but within more global contexts. In the USAF, the global context consists of the set of prevailing, current, and anticipated management and operational problems. In these contexts the evaluative criterion for research is simply, "Does this work address, offer insight into, or solve any aspect of my problems?"

A prevailing problem in the USAF has been the increasing life-cycle costs of avionics systems [1]. A major portion of these spiraling costs have been attributed to the avionics software. A costly deficiency in software procured by the USAF in the past has been the software's lack of flexibility. Typically the procurement of a new avionics system required the purchase of a unique, ground based support machine and the associated software for maintaining, translation codes for, and interfacing with the on-board machine. The USAF operational environment (aircraft are dispersed over the world, the same avionics system is used in different types of aircraft, one aircraft may have several different avionics systems) has resulted from the purchase of many "unique" support machines and several flavors of support software for each new avionics system.

4.1. The Generality of CADM

The research accomplished during Phase 3 indicates that a portion of the problem of inflexibility need not be a future concern for the USAF. The CADM we have implemented is essentially airplane independent, i.e.,
the software is not written for some specific airplane with a specific number of valves, pumps, engines, fuel tanks, electrical generators, etc. Once the host aircraft configuration is provided, CADM constructs a data structure representing the specific aircraft and uses this one-time computed structure in correcting failures and error conditions on that aircraft.

The CADM design philosophy provides flexibility in two dimensions: across types of airplanes, since the CADM correction procedures would work as well on a B-52 as on an F-15; and across time, since CADM can generate a new representative data structure as a result of modifications to existing aircraft.* Within these bounds, the presently implemented CADM is fully sufficient for handling any fuel and electrical power distribution system. This means that no additional software is required regardless of how such systems are configured. It also means that no conceptual extension is required (though more subsystem specific software is necessary) to provide sufficiency for handling any other subsystem which can be modeled as a network of discrete components.

4.1.1. CADM's Knowledge Structure

Within the field of programmed intelligence, two primary criteria are used to determine how intelligent a particular program is, namely, the generality and the adaptability of the particular implementation. The previous discussion has revealed CADM to be highly general, using no specific aircraft dependent techniques. This high degree of generality is derived from two properties of the structure of CADM's store of knowledge. First, *The degree to which this is possible in limited only by the degree to which CADM's model of the subsystem components (valves, pumps, etc.) is consistent with the actual aircraft components. The present CADM implementation models such components in a general, but somewhat idealized manner.
correction procedures are associated not with individual aircraft components but with classes of components, and second, aircraft subsystem error conditions are defined in terms of aircraft system parameters rather than in terms of component states. CADM's knowledge is stored in a procedural net [2], which is a hybrid data structure resulting from the merging at a technique for producing a semantic net [3] and a technique for the procedural specification of knowledge [4].

4.1.2. Hierarchies of System Component Classes

In CADM, classes of aircraft subsystem components and specific components are nodes in a hierarchy. Links between the nodes are relations between the components and classes of components. Procedures (programs) are associated with the most general nodes in a particular hierarchy (see Figure 25). This type of structure is suitable for modeling classes of components in the most general way. For example, the concept "VALVE" is represented as something that is between an engine and a tank. In our implementation all valves are electrically actuated. Hence, the concept "VALVE" is subordinate to the concept "ELECTRICAL INSTRUMENT". Electrical instruments have switches to control them and circuit breakers to connect them to their source of power. Since the concept of an electrical instrument is very general, procedures to determine whether any electrical instrument is on or off (the switch must be on, the circuit breaker closed, and the power source active if the instrument is to be on) and procedures for turning any electrical instrument on or off are associated with the concept node "ELECTRICAL INSTRUMENT". The advantage in this method is that a specific valve (e.g. VALVE37) only need have a pointer to the concept node "VALVE" to inherit all the properties of valves.
Figure 25. A hierarchy of procedures involving an electrical instrument.
If more valves are added to an existing aircraft, the data structure representing the aircraft subsystem does not have to be completely reconfigured. Rather, one additional node and a pointer from that node to the concept node "VALVE" is all that must be added to ensure that the added valves can be turned on and off.* There is no need for a specific procedure for controlling a specific valve (e.g. VALVE37). Of course, if VALVE37 is an unusual or special valve requiring a specific procedure to control it, that procedure can be associated directly with the node "VALVE37", thereby preventing the more general valve control procedures from affecting VALVE37.

4.1.3. System Parametric Definitions of Failures

Equally important in providing CADM's generality is the definition of error conditions in terms of system parameters (engine temperature, fuel flow, etc.). This means that CADM does not have to continuously monitor the status of every individual component. This method of definition eliminates two problems. First, because the number of components in an aircraft is very large, continuously monitoring each component would consume much of the limited on-board computational capacity. Second, since there is usually considerable redundancy in aircraft systems, knowing the status of a particular component may convey little useful information by itself. An error condition is typically caused by improper relationship existing between several components rather than by the state of any one particular component.

Therefore, error conditions are defined by such entities as ENGINE TEMP LOW and INADEQUATE FUEL FLOW rather than by PUMP1 OFF or VALVE3 ON. This means that the correction procedures can be highly generalized specifications as for instance, MAKE TEMP NORMAL and MAKE FUEL FLOW POSITIVE,

*Additional pointers must be added to the tanks and/or engines involved to ensure that the valves functions in relation to the existing components are understood by the program.
where TEMP NORMAL and FUEL FLOW POSITIVE are defined conditions. For example, FUEL FLOW POSITIVE requires fuel to be available from some tank, a valve between the tank and the appropriate engine to be open, and one of the fuel pumps to the appropriate engine to be on.

4.2. The Adaptability of CADM

The generalized structure of knowledge and the generalized definition of error conditions result in CADM's being a compensator rather than a diagnostician.

4.2.1. CADM as an Operator

Because CADM is motivated to keep the aircraft flying rather than to determine exactly what caused the error condition, its performance is more that of an operator than of a maintainer. It is cognizant of the constraints of operating in real time. There are three distinct reasons for this. First, in an operational aircraft, an error condition is a serious thing that needs correcting immediately. The first priority is to restore the aircraft's capability and the second priority is to identify what caused the problem and then repair it if possible.

Second, CADM must cooperate with other intelligent entities. One of these entities is the pilot. If the pilot turns a particular valve, CADM assumes that the pilot has a good reason for doing so, even if closing the valve causes an error condition. Consequently, rather than competing with the pilot and opening the same valve that the pilot just closed, CADM must compensate by finding another way to keep the aircraft flying, preserving the pilot's independence. A second intelligent entity which CADM must deal with is CADM itself. CADM must be able to solve many problems simultaneously,
i.e., multiple error conditions may occur simultaneously, may be detected simultaneously, and may require simultaneous correction. To accomplish this, CADM creates independent correction procedures, one for each error condition, and lets them operate in parallel. This means that there might be competition between two independent correction procedures for the same component. The losing procedure must find another way of accomplishing its goal even if the desired component is not available for use.

Third, in our implementation the components are idealized and, with few exceptions (circuit breakers can become jammed open or closed), cannot fail. Later versions of CADM will include non-idealized components and the capability to diagnose the fundamental cause of error conditions.

4.2.2. Adaptability Demonstrated by Program Responses

Whereas generality is that property of a program design that ensures broad applicability of the program, adaptability is that property of a program that ensures stability of the program's performance within any one of its applications. In our implementation, adaptability is demonstrated by the response of the program to perturbations in the composition of the aircraft system, to the pilot's desires, or to the failure of one of CADM's generated procedures to correct an error condition.

4.2.2.1. Responses to Failures

If an aircraft engine was turned off in flight because it was damaged, an intelligent system would no longer expect FUEL FLOW POSITIVE to be desirable for the non-functioning engine. CADM recognizes FUEL FLOW POSITIVE into a damaged or destroyed engine as an error condition and corrects it in the same manner that it recognizes FUEL FLOW ZERO as an error condition into a healthy engine.
4.2.2.2. Responses to Pilot Intervention

In response to the pilot's desires, there are three available modes of CADM operation. Normally, CADM will be fully operational, detecting and correcting all error conditions. If for some reason the pilot does not desire CADM's assistance, he may completely disable CADM. In this situation CADM is dormant, sensitive only to the pilot's call to return to duty. Or CADM may be placed in monitor mode where it detects error conditions and advises the pilot of them, but does nothing to correct them. These multiple modes have research as well as operational implications. Such a capability is required if the performance of the man-machine combination in various situations is to be evaluated.

4.2.2.3. Responses to Procedure Failures

A final example of CADM's intelligence and adaptability is its ability to cope with failure. If in the process of investigating an approach to correct an error condition CADM finds a method to be unfruitful, it provides as a continuation point for itself a data structure called a HISTORY. Using this HISTORY, CADM either may continue by investigating other methods within the same approach, or if the approach is found to be unsuccessful, may try other alternate approaches. The important point is that CADM does not have to start all over again when a method or approach does not prove successful. It is able to make use of results of past efforts to solve the problem. Only after all approaches have proved ineffective does CADM pass the error condition to the pilot for correction.
REFERENCES


5. PILOT-CADM INTERACTION

5.1. Introduction

In our last report, we noted that the full potential of CADM would not be realized if the pilot and CADM could not effectively communicate. In this section of this report, we will consider first CADM-to-pilot communication and then, pilot-to-CADM communication.

5.2. The Master Monitor Display

5.2.1. General Problems

Two general considerations were stressed in the development of a display system for CADM. The first involved organizing a considerable amount of information. It was not feasible to put all the sensors, controls, and CADM information on one or two dedicated displays. Instead, we organized the information in a hierarchical structure similar to that espoused by other researchers [1,2]. We will later discuss this aspect of the display in detail.

The second general consideration involved informing the pilot of CADM's decisions. Our approach here included a combination of status information, messages, and a failure monitor. All of these features were aimed at keeping the pilot aware of CADM and thereby avoiding conflicts between the two decision makers.

5.2.2. Design

Now, we will pursue these topics in more detail. Figure 26 illustrates the Master Monitor Display. The top two-thirds of the display is devoted to data pages which are hierarchically arranged with there being six choices including fuel, electrical, hydraulic, and engine subsystems
Figure 26. The Master Monitor Display
as well as top level pages for the flight plan and failure monitor. Most of these top level data pages branch to more specific pages. For example, hydraulic has three branches including the pump system, landing gear, and flaps/wing sweep. Data elements are intensified when they change and thus, draw the pilot's attention. After a short time with no change, the intensity is lowered.

5.2.2.1 The Touchtone Keyboard

The pilot utilizes a touchtone-like keyboard to choose data pages and branches. With so few keys, it is possible to learn how to perform without actually looking at the keyboard. Keyboard entries preceded by a control character are for selection of options on the multi-function display which occupies the lower one-third of the Master Monitor Display. The function currently in use is intensified for easy reference by the pilot. The idea of a multi-function display has been utilized elsewhere in the design of flight-management systems [3].

5.2.2.2 Status Information

The status information on the right side of each data page indicates whether or not CADM has operated that control recently and will also indicate when the subsystem or component indicated has failed. Messages to the pilot from CADM appear on the bottom of the display. Table 5 summarizes the messages which CADM can send to the pilot. In addition, the messages and their times of occurrence are concatenated on the failure monitor as shown in Figure 27. The pilot can access this data page if he desires more detail concerning CADM's actions.
<table>
<thead>
<tr>
<th>Number</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Message (Blank)</td>
</tr>
<tr>
<td>1</td>
<td>Engine System Check</td>
</tr>
<tr>
<td>2</td>
<td>Fuel System Check</td>
</tr>
<tr>
<td>3</td>
<td>Electrical System Check</td>
</tr>
<tr>
<td>4</td>
<td>Hydraulic System Check</td>
</tr>
<tr>
<td>5</td>
<td>Left Engine Flameout</td>
</tr>
<tr>
<td>6</td>
<td>Right Engine Flameout</td>
</tr>
<tr>
<td>7</td>
<td>Left Engine Fire</td>
</tr>
<tr>
<td>8</td>
<td>Right Engine Fire</td>
</tr>
<tr>
<td>9</td>
<td>Left Engine Destroyed</td>
</tr>
<tr>
<td>10</td>
<td>Right Engine Destroyed</td>
</tr>
<tr>
<td>11</td>
<td>Buss Relay Open</td>
</tr>
<tr>
<td>12</td>
<td>Left Buss Down</td>
</tr>
<tr>
<td>13</td>
<td>Right Buss Down</td>
</tr>
<tr>
<td>14</td>
<td>Left Hydraulic Pump Failure</td>
</tr>
<tr>
<td>15</td>
<td>Right Hydraulic Pump Failure</td>
</tr>
<tr>
<td>16</td>
<td>Left Elec Fuel Pump Failure</td>
</tr>
<tr>
<td>17</td>
<td>Right Elec Fuel Pump Failure</td>
</tr>
<tr>
<td>18</td>
<td>Left Mech Fuel Pump Failure</td>
</tr>
<tr>
<td>19</td>
<td>Right Mech Fuel Pump Failure</td>
</tr>
<tr>
<td>20</td>
<td>Left Drain Pump Failure</td>
</tr>
<tr>
<td>21</td>
<td>Right Drain Pump Failure</td>
</tr>
<tr>
<td>22</td>
<td>Inadequate Hydraulic Pressure</td>
</tr>
<tr>
<td>23</td>
<td>Inadequate Hydraulic Fluid</td>
</tr>
<tr>
<td>24</td>
<td>Hydraulic Line valve Jam</td>
</tr>
<tr>
<td>25</td>
<td>Inadequate Fuel Flow to Left Engine</td>
</tr>
<tr>
<td>26</td>
<td>Inadequate Fuel Flow to Right Engine</td>
</tr>
<tr>
<td>27</td>
<td>Inadequate Fuel in Forward Tank</td>
</tr>
<tr>
<td>28</td>
<td>Inadequate Fuel in Rear Tank</td>
</tr>
<tr>
<td>29</td>
<td>Fuel Line Valve Jam</td>
</tr>
<tr>
<td>30</td>
<td>Intertank Flow Control Jam</td>
</tr>
<tr>
<td>31</td>
<td>Left Engine Temperature Low</td>
</tr>
<tr>
<td>32</td>
<td>Left Engine Temperature High</td>
</tr>
<tr>
<td>33</td>
<td>Right Engine Temperature Low</td>
</tr>
<tr>
<td>34</td>
<td>Right Engine Temperature High</td>
</tr>
<tr>
<td>35</td>
<td>Inadequate Left Engine Thrust</td>
</tr>
<tr>
<td>36</td>
<td>Inadequate Right Engine Thrust</td>
</tr>
<tr>
<td>37</td>
<td>Left Engine Vibration Irregular</td>
</tr>
<tr>
<td>38</td>
<td>Right Engine Vibration Irregular</td>
</tr>
<tr>
<td>39</td>
<td>Fuel Line Valve Jam (FT-LE)</td>
</tr>
<tr>
<td>40</td>
<td>Fuel Line Valve Jam (RT-RE)</td>
</tr>
<tr>
<td>41</td>
<td>Fuel Line Valve Jam (FT-RE)</td>
</tr>
<tr>
<td>42</td>
<td>Fuel Line Valve Jam (RT-LE)</td>
</tr>
<tr>
<td>43</td>
<td>Left Engine Restarted</td>
</tr>
<tr>
<td>44</td>
<td>Right Engine Restarted</td>
</tr>
<tr>
<td>45</td>
<td>Left Engine Fire Out</td>
</tr>
<tr>
<td>46</td>
<td>Right Engine Fire Out</td>
</tr>
<tr>
<td>47</td>
<td>Adequate Hydraulic Pressure</td>
</tr>
<tr>
<td>48</td>
<td>Adequate Fuel Flow to Left Engine</td>
</tr>
<tr>
<td>49</td>
<td>Adequate Fuel Flow to Right Engine</td>
</tr>
<tr>
<td>50</td>
<td>Left Engine Temperature Normal</td>
</tr>
<tr>
<td>51</td>
<td>Right Engine Temperature Normal</td>
</tr>
<tr>
<td>52</td>
<td>Left Engine Thrust Adequate</td>
</tr>
<tr>
<td>53</td>
<td>Right Engine Thrust Adequate</td>
</tr>
<tr>
<td>54</td>
<td>Left Engine Vibration Normal</td>
</tr>
<tr>
<td>55</td>
<td>Right Engine Vibration Normal</td>
</tr>
</tbody>
</table>

Table 5. CADM messages.
MMD Failure Monitor

256 RIGHT ENGINE RESTARTED
252 RIGHT ENGINE TEMPERATURE NORMAL
250 RIGHT ENGINE FLOW ADEQUATE
246 RIGHT ENGINE FLAMEOUT
245 RIGHT ENGINE FLOW INADEQUATE
241 RIGHT ENGINE TEMPERATURE LOW
109 LEFT ENGINE RESTARTED
105 LEFT ENGINE TEMPERATURE NORMAL
101 LEFT ENGINE FLAMEOUT
99 LEFT ENGINE TEMPERATURE LOW

Figure 27. MMD Failure Monitor Format
5.2.2.3. The Multi-Function Display

The multi-function display allows the pilot to put CADM into any of three modes: off, monitoring only, and on. In this way, he can easily override the CADM system.

5.3. Telling the Computer What the Pilot is Doing

A more subtle approach to pilot-to-CADM communication will now be considered.

5.3.1. The Problem

If a CADM system is to be successful, the pilot's additional workload due to the necessity of communicating with CADM should be less than the workload absorbed by the CADM system. Otherwise, the initial reason for CADM is, for the most part, subverted. Thus, we need a method of covertly determining what the pilot is doing. The approach to be discussed here is reported elsewhere in detail [4,5] and thus will only summarize the method and not discuss the specifics of its implementation.

5.3.2. The Approach

When flying an aircraft, a pilot has both control and monitoring tasks. When one of his monitoring tasks requires more attention than usual, one might expect the pilot's control performance to be affected. Thus, by looking at the pilot's control performance, it may be possible to determine if his attention is diverted to one of his monitoring tasks. With this information, CADM might be able to determine which monitoring task has attracted the pilot's attention and then, CADM can act so as not to conflict with the pilot. If such an approach is feasible, then CADM can determine what the pilot is doing without explicitly asking him.
5.3.2.1. Feasibility Studies

To study the feasibility of this approach, the aircraft situation was abstracted to include only a pursuit tracking task with randomly occurring arithmetic side tasks. The display employed is shown in Figure 28. Three experiments were performed with subjects being instructed to minimize tracking error while responding to the arithmetic problems as quickly as possible without making any multiplication errors.

Subjects were told to keep tracking at all times and touch-type the answers to the arithmetic problems. To separate the effects of scanning from the effects of thinking about arithmetic and touch-typing, artificial sidetasks were introduced and identified by noting that the operator was "K" and not "X".

Our goal was to develop a method that would utilize the subject's tracking behavior to predict when the subject was concentrating on the arithmetic task even though he was looking at the tracking task. Also, we wanted a method that would not detect mere scanning of the artificial side tasks.

The method which we developed includes two parts. First, we utilize the displayed tracking error and joystick outputs with a fading-memory system identification algorithm [6,7] to find the parameters of an $N$th-order sampled-data linear model of the pilot's tracking behavior. The memory of the algorithm is designed to allow rapid (a few seconds) adaptation to changes in the pilot's tracking behavior.

Second, by using the parameters found by the system identification algorithm, we employed discriminant analysis [8] to classify the tracking model as belonging to either pure tracking or tracking plus mental arithmetic.
Figure 28. The Experimental Situation
5.3.2.2. Results

Based upon three experiments utilizing six subjects, we found that the method could detect approximately 90% of the shifts of attention. The few false alarms could be attributed to the artificial side tasks. The discriminant analysis had to be adjusted when the system dynamics were changed but did not have to be adjusted for different subjects. The approach could easily be utilized in real time since it requires approximately 10 CPU seconds to process 5 minutes of data.

Thus, we showed our method to be feasible for the detection of shifts of pilot attention in a real time environment. However, actual implementation in an aircraft will require various heuristics to handle, for example, times when the pilot completely stops tracking. Multi-class discriminant analysis might be useful in this situation.

5.4. Conclusions

We have considered approaches to interfacing the pilot with CADM. In the CADM-to-pilot direction, we have chosen approaches that appear satisfactory in our limited demonstrations. However, more rigorous investigation of the alternatives should be pursued especially with emphasis on solid empirical studies.

Our approach to covert pilot-to-CADM communication appears promising. The experiments in the abstracted aircraft situation should not be extended to more realistic aircraft situations.

A CADM system cannot be developed with only a perspective of the aircraft. The computer has to realize that the pilot exists and system designers must develop appropriate methods of computer-pilot communication. Otherwise, sophisticated automation schemes may prove worthless or perhaps
even dangerous.

REFERENCES


6. THE COMMUNICATING SYSTEMS

6.1. PDP-11 Systems and Communications

6.1.1. PDP-11 System Improvements

In the past months several improvements have been made to PDP-11 monitor programs. Originally device independent I/O was implemented in such a way as to be inefficient for block transfer devices (like disk) and efficient for character devices (TTYs). This has been remedied by adding an I/O mode which is still device independent, even though it implemented to optimize block transfers.

Several patches have been made to make the multiprocessing system more flexible and more efficient. Support for a software interrupt device has been written to allow fast task swapping to high priority devices that have I/O requests pending. This along with some patches in the interrupt dispatching software decreases the time taken to wake up tasks sleeping on I/O events, thereby making the overall system respond more quickly to real time interrupts.

Several new system programs for faster file transfers through the PDP-11/10 interface have been planned and are partially completed. These programs will increase effective transfer rates by a factor of five. This improvement will allow much greater productivity during the debugging stages of software development, when many recompilations and file transfers are done (compilation is done on the PDP-10 and load files are then shipped to the PDP-11 through the high speed interface).

The last bugs are being found in the PDP-11/10 interface software. This has proven to be a difficult software system to debug due to the asynchronous nature of computer-to-computer interfaces.
6.1.2. Alternative PDP-11 Operating Systems

At this stage of the development of our PDP-11 system we have enough information to make intelligent comments concerning the relative merits of our system versus other systems that have been developed.

6.1.2.1. The DEC Disk Operating System

Digital Equipment Corporation has developed three general software systems for PDP-11's. The first is DOS (Disk Operating System). This system supports a wide variety of system software for our application (that is, real time software running on a PDP-11 with host support from a PDP-10), most of which is not pertinent. The DOS system requires a disk, where as ours will run from disk, Dectape, or even papertape. DOS, by swapping from disk, runs in as small a system as a 12K, PDP-11. It also supports no facilities for accessing memory beyond 28K. Our system is totally core resident (approximately 36K) and separates user programs from each other and the system via the mapping hardware. Real time response is possible because no disk swapping is required. Basically, DOS is a general system for a PDP-11 without memory management options where real time (and or multiprocessing) is not required.

6.1.2.2. The DEC Real Time System

The DEC RT (Real Time, for small PDP-11's) system supports most of the facilities available under DOS for a single user (DOS has a file structure partitioned by user numbers, while RT does not support user numbers). This system, while being faster than DOS, still supports no extended memory access and loads user programs into the same memory partition as the monitor itself. This system runs comfortably from Dectape, Magtape or Disk. It is
an excellent system for small PDP-11's. However, for our applications, it is too simple (note: this system runs in 8K PDP-11's reasonably well).

6.1.2.3. The DEC RSX-11 System

The RSX-11 system supports real time data acquisition tasks and a large body of development aids. In addition it provides a means of using the PDP-11/40 memory management hardware. This system supports multiple tasks that are disk swappable but does not support user level hardware interrupt servicing. Hardware service tasks must run as privileged system tasks.

Because of the cost of this system, because we wanted to be able to service interrupts in user level jobs, because we did not need the extensive support software (because of PDP-10 host support), and because we wanted a system that was less dependent on the hardware I/O system (i.e. could run without a disk) we did not choose to get RSX-11.

6.1.2.4. The UNIX System

We recently obtained a copy of the Bell Labs UNIX-11 system. This is a timesharing-multiprocessing system that is quite flexible and quite non-DEC compatible (i.e. until support programs are written, our PDP-11 host support is minimal). This system does not support a real time user facility of any kind. All real time tasks must be included as part of the running monitor. This can be quite inconvenient if devices require service programs that are not of the data transfer variety. For these devices, none of the monitor facilities are particularly useful, however, the drivers may not be written as outside monitor tasks. We have found in our work on both the PDP-11 and the PDP-10 that, for real time applications, many interrupt service routines are best implemented as tasks outside the operating system.
in order to minimize problems that arise when new software is integrated with old software.

6.2. The CSL PDP-10/PDP-11 Communication Link

Communication between the PDP-11 and the PDP-10 is accomplished by means of a specially built piece of hardware, called the channel. The channel provides high-speed, direct-memory access (DMA), data transferring and transformation capabilities between the two machines. The channel hardware design revolves around the concepts of symmetry and autonomy and attempts to achieve these goals in spite of the fact that the two machines have very different architectures. The channel is operationally symmetric. That is, the same status and control bits are provided to both machines, although not necessarily in the same order in a particular word. Software on both machines would respond in a similar if not identical ways to a particular status or control bit. The channel is autonomous in the sense that it simply does not provide for one machine to force its will upon the other. In other words it is possible to run separate software on both the PDP-11 and PDP-10 without fear of corruption from faulty or malicious software on the other machine. Communication then is accomplished by mutual consent. That is, the software on both machines must agree to a data transfer before it can occur.

6.2.1. The Channel

On the PDP-10 the word length is 36 bits whereas on the PDP-11 it is 16 bits. This creates several problems with respect to the construction of a uniform channel, that is, a channel through which complete words may be transmitted and received on both machines. To overcome these problems the channel provides two modes for mapping the bits from one machine to the other.
In the first mode, two full PDP-11 words are constructed from one PDP-10 word, and in the second mode, three PDP-11 words are used to construct a full PDP-10 word.

As of April 1975 the channel hardware has been completely built and debugged. In the most sophisticated debugging procedures used, a program on the PDP-10 and a specially written channel driver on the PDP-11 were used to simulate on the PDP-11 a small disk for use by the PDP-11. Files were transferred to and from this simulated disk and the accuracy of transmission was verified. Timing checks were also performed and it was determined that, including standard system interference and overhead, a sustained data transfer rate of 2.5 million bits per second could be maintained. Needless to say this test makes extensive use of the DMA, the mapping, and the control hardware in the channel. This test is also important since it is the first time the DATA type interrupts which are used by the channel on the PDP-10 had been successfully used on our system.

6.2.2. The Channel Drivers

On the PDP-11 the channel driver was embedded in the Memory Operating System, called MOS. MOS is an extensively modified version of Digital Equipment Corporation’s DOS, version from operating system for PDP-11s. In addition, to allowing batch processing, dynamic console switching, MOS uses the extended memory available on our PDP-11 as the system device instead of a disk. Now that an RK05 disk-pack is connected to the PDP-11 system, MOS has given way to the M & M and UNIX systems, both of which utilize the RK05 and make more direct use of the extended memory.

For the past year the channel has been extensively used as the primary means for transferring data between the two machines, and the
hardware aspects of the channel have been limited to maintenance only. Until now, using the channel required the use of special programs on both the PDP-11 and the PDP-10 with special 10 privileges to access and manipulate the channel registers directly. At present, the major efforts in the development of the channel are limited to software. Drivers for the channel have been written for both the PDP-11 M + M system and the PDP-10 602 timesharing system. These drivers have been installed into their respective monitors and are presently being debugged. When this debugging is completed the M + M driver will be modified and transported to the UNIX system. This should allow unprivileged user programs such as PIP on both the PDP-10 and the PDP-11 to communicate with each other via the channel using standard monitor 10 calls.

6.3. The WPAFB/CSL Communication Link

In order to facilitate the delivery to WPAFB of software written for WPAFB at CSL, and to debug it on the WPAFB system, a computer to computer communication link was implemented between WPAFB and CSL. Communications take place over a 1200 baud asynchronous communication line between the two sites. Each end of the line is attached to the front end of its respective PDP-10 and it is assigned a TTY line number by the TOPS-10 monitor. Communication is accomplished by OPEN-ing that TTY line from the local system and doing I/O with it as a device. The software uses some of the advanced features of the TOPS-10 monitor including non-blocking I/O, software interrupts, and other features. Because of this it requires the 602 monitor in order to run.

6.3.1. Implementation

Some of the problems encountered and overcome during the implementation phase include bugs to the TOPS-10 monitor, and problems arising from each
PDP-10 assuming that itself was the master of the communication line, thereby causing endless character echoes between the two systems. The latter was fixed by turning off the echo feature of the line at the appropriate times, and slightly changing the command decoder in the monitor so that an error message from the other system would not be interpreted as a new command for the local system.

6.3.2. Capabilities

The software itself is computer independent. It allows the PDP-10 to connect to any asynchronous ASCII line. We have used the software in order to dial out on a data set to a remote computer, which is the reverse of the normal situation where a data set is only used for incoming calls. This has been used successfully in connecting from CSL to a local system which in turn is connected to the ARPANET.

The software is also line speed independent (although the hardware, of course, is not). The software should work at data rates ranging from 110 baud to 9600 baud. It has been verified to work successfully at 110, 300, 600, 1200, and 2400 baud. It includes capabilities for both virtual terminal and file transferring. Virtual terminal capabilities allow a terminal which is attached to one system to act as if it were physically attached to the remote system. The user typing at the terminal would notice no difference, except possibly that character echoing occurs somewhat slower than normal because the characters must go through two systems rather than one. File transferring is supported in one of three modes including BINARY, ASCII and PIP (compatible with the PDP-10 program called PIP). A user can transfer files by running on the remote system the same software as he uses on his local system, but rather
than using it to connect to another teletype line he must give different commands in order to use it in local I/O mode when sending or receiving a file.

6.3.3. Protocol

The file protocol was designed to resemble as closely as possible the standard PDP-10 teletype protocol. It uses the full duplex feature of the line in order to maintain adequate buffer space on the receiving end of the line. When the receiving PDP-10 gets a new character it puts the character into a buffer and checks that the buffer isn't too full. If it is, it sends a control-S character to the transmitting PDP-10, signaling that it should stop sending in order to allow the receiving program a chance to process the characters in the buffer. When the receiving buffer becomes empty, a control-Q character is sent to the transmitter, signaling that it should resume the sending of characters. The end of file is signified by a control-Z character. In ASCII transferring mode, the program sends the full ASCII character set with the exception of the NULL character and the control-Z character. The PIP mode of transferring is much like the ASCII mode with the exception that the line feed character in the carriage return/line feed sequence is not sent because the PIP program automatically fills in this character. The binary mode is the only mode of file transferring that is not machine independent. In this mode, each 36 bit word of the PDP-10 is segmented into six bytes, each six bits long. Each byte is then incremented by the octal constant 40 in order to map the character onto one of 64 ASCII characters (all non-control characters). These characters are transmitted to the receiving side where the reverse process takes place and the characters are decoded and put into the receiving file. Because BINARY mode sends six characters per word as
opposed to the five characters per PDP-10 word in ASCII mode, it is somewhat slower, but has the advantage that the files are transferred in exact image mode.

6.3.4. Internal Structure

The communication link software attaches, through service routines, the input of a user's teletype to the output of the communication line and the input of the communication line to the output of his teletype. It is as if someone had patched a cable from the user's teletype line directly to the communication line, thereby bypassing the local machine entirely.

The program is interrupt driven. When the program is not busy, it remains in the hibernation state until it receives a new character from either the local or the remote system. At that time a software interrupt occurs in the program, flags that an event took place and dismisses the interrupt. This also wakes the main level program from its hibernation and starts a polling loop to determine which event occurred to awaken it. When it finds out what the event was, it performs the appropriate action and then returns to the hibernation state.

In order to maintain fast response times for character echoing, the software has the ability to lock itself into core and to run in a high priority run queue. This occurs when the user specifies the appropriate switches and has the privileges to do these things. By choosing to do this, he can eliminate the problems of long echoing times when running on a heavily loaded system.

6.3.5. Testing

The test case for the software was the transferring of a local
program library (the TELL program) from CSL to WPAFB. This library consists of approximately 768,000 bytes of data and it was transferred in BINARY mode. The file was transmitted in 50 block segments, each block containing 768 bytes, in order to reduce the perturbations caused by line errors. During the entire transfer only one error occurred that required a segment to be transferred again. This represents an error rate of 1 bit in 1,000,000.
APPENDIX 1

Aircraft Characteristics and Nomenclatures
AIRCRAFT CHARACTERISTICS

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>9.16</td>
<td>ft</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T/eng</td>
<td>15,000</td>
<td>lbs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wt, gross</td>
<td>55,000</td>
<td>lbs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wt, empty</td>
<td>35,000</td>
<td>lbs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alt, max</td>
<td>80,000</td>
<td>ft</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mach no, max</td>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>550</td>
<td>ft²</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e₀</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/s, max</td>
<td>100</td>
<td>lb/ft²</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

δf

<table>
<thead>
<tr>
<th>Angle</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Λ</td>
<td>15</td>
<td>25</td>
<td>35</td>
<td>50</td>
<td>70</td>
</tr>
<tr>
<td>AR</td>
<td>6.55</td>
<td>5.90</td>
<td>5.46</td>
<td>4.37</td>
<td>3.28</td>
</tr>
<tr>
<td>Iₓ</td>
<td>70000</td>
<td>70000</td>
<td>70000</td>
<td>45000</td>
<td>45000</td>
</tr>
<tr>
<td>Iᵧ min</td>
<td>260000</td>
<td>260000</td>
<td>260000</td>
<td>275000</td>
<td>275000</td>
</tr>
<tr>
<td>Iᵧ max</td>
<td>320000</td>
<td>320000</td>
<td>320000</td>
<td>335000</td>
<td>335000</td>
</tr>
<tr>
<td>δₑ trim @450 fps</td>
<td>0.0745</td>
<td>0.061</td>
<td>0.045</td>
<td>0.00125</td>
<td>-0.069</td>
</tr>
<tr>
<td>c.g. ref</td>
<td>25</td>
<td>35</td>
<td>37</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>α trim</td>
<td>0.06</td>
<td>0.066</td>
<td>0.0726</td>
<td>0.0985</td>
<td>0.1306</td>
</tr>
<tr>
<td>b</td>
<td>60</td>
<td>54</td>
<td>50</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>$\Lambda$, deg</td>
<td>$C_L \times C_{L_{\text{max}}}$</td>
<td>$V_{\text{stall}}$, fps</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>2.0</td>
<td>205</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>1.8</td>
<td>216</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1.6</td>
<td>229</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>1.3</td>
<td>254</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>1.0</td>
<td>290</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*without flaps. Add .2 to $C_L$ for each 10° flap

<table>
<thead>
<tr>
<th>$\Lambda$, deg</th>
<th>$T_{ss}$, lb @450 fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>9565</td>
</tr>
<tr>
<td>25</td>
<td>8239</td>
</tr>
<tr>
<td>35</td>
<td>8240</td>
</tr>
<tr>
<td>50</td>
<td>6914</td>
</tr>
<tr>
<td>70</td>
<td>5589</td>
</tr>
<tr>
<td>Symbol</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>Alt</td>
<td>reference altitude</td>
</tr>
<tr>
<td>AR</td>
<td>aspect ratio</td>
</tr>
<tr>
<td>$a_1$, $a_2$</td>
<td>time non-dimensionalizing terms</td>
</tr>
<tr>
<td>b</td>
<td>wing span</td>
</tr>
<tr>
<td>C</td>
<td>side force</td>
</tr>
<tr>
<td>$C_D$</td>
<td>drag coefficient</td>
</tr>
<tr>
<td>$C_D^{c}$</td>
<td>drag coefficient due to control inputs</td>
</tr>
<tr>
<td>$C_D^{o}$</td>
<td>drag coefficient for zero lift</td>
</tr>
<tr>
<td>$C_D_\alpha$</td>
<td>drag coefficient due to angle of attack</td>
</tr>
<tr>
<td>$C_D_\alpha$</td>
<td>drag coefficient due to rate of change of angle of attack</td>
</tr>
<tr>
<td>$C_D_\delta_e$</td>
<td>drag coefficient due to elevator deflection</td>
</tr>
<tr>
<td>$C_D_\delta_f$</td>
<td>increment of drag coefficient due to flap extension</td>
</tr>
<tr>
<td>$C_D_\delta_g$</td>
<td>increment of drag coefficient due to landing gear extension</td>
</tr>
<tr>
<td>c.g.</td>
<td>center of gravity location</td>
</tr>
<tr>
<td>c.g. ref</td>
<td>reference center of gravity location</td>
</tr>
<tr>
<td>$C_L$</td>
<td>lift coefficient</td>
</tr>
<tr>
<td>$C_L^{c}$</td>
<td>lift coefficient due to control inputs</td>
</tr>
</tbody>
</table>
\( C_{L_{\text{max}}} \)
maximum lift coefficient

\( C_{L_0} \)
lift coefficient at zero angle of attack

\( C_{L_{\alpha}} \)
lift coefficient due to angle of attack

\( C_{L_{\alpha}} \)
lift coefficient due to rate of change of angle of attack

\( C_{L_{\delta_e}} \)
lift coefficient due to elevator deflection

\( C_{L_{\delta_f}} \)
increment in lift coefficient due to flap extension

\( C_{L_q} \)
lift coefficient due to pitching velocity

\( C_1 \)
rolling moment coefficient

\( C_{1c} \)
rolling moment coefficient due to control deflection

\( C_{1p} \)
rolling moment coefficient due to rolling velocity

\( C_{1\delta_{\alpha}} \)
rolling moment coefficient due to aileron deflection

\( C_m \)
pitching moment coefficient

\( C_{mc} \)
pitching moment coefficient due to control deflection

\( C_{m_0} \)
pitching moment coefficient at zero angle of attack

\( C_{mq} \)
pitching moment coefficient due to pitching velocity

\( C_{m_{\alpha}} \)
pitching moment coefficient due to angle of attack
$C_{m_{\alpha}}$ pitching moment coefficient due to rate of change of angle of attack

$C_{m_{\delta_{e}}}$ pitching moment coefficient due to elevator deflection

$C_{m_{\delta_{f}}}$ increment in pitching moment coefficient due to flap extension

$C_{m_{\delta_{g}}}$ increment on pitching moment coefficient due to landing gear extension

$-c$ wing aerodynamic chord

$C_{x}$ force coefficient along the X axis

$C_{z}$ force coefficient along the Z axis

$D$ drag

$e_{0}$ Wing efficiency factor

$g$ acceleration of gravity

$I_{x}$ moment of inertia about the X axis

$I_{y}$ moment of inertia about the Y axis

$I_{z}$ moment of inertia about the Z axis

$I_{zx}$ product of inertia in the ZX plane

$L$ rolling moment

$L'$ lift

$L'_{c}$ lift due to control inputs

$M$ pitching moment

$m_{, wt}$ aircraft mass, weight

$N$ yawing moment
p  rolling velocity
q  pitching velocity
q∞  dynamic pressure
r  yawing velocity
R/C  rate of climb
S  wing area
T  thrust
u  X velocity
V  airspeed
v  Y velocity
w  Z velocity
X  aircraft axis, positive forward
X'  aerodynamic force in X direction
Y  aircraft axis, positive along right wing
Y'  aerodynamic force in Y direction
Z  aircraft axis, positive down
Z'  aerodynamic force in Z direction
α  angle of attack of the zero-lift line
αabs  absolute angle of attack
αx  angle of attack
dotα  rate of change of angle of attack
β  sideslip angle
δa  aileron deflection angle
δe  elevator deflection angle
\( \delta_f \)  
flap extension angle  

\( \delta_g \)  
gear state; 0 up, 1 down  

\( \gamma \)  
flight path angle  

\( \Lambda \)  
wing sweep angle  

\( \theta \)  
pitch angle  

\( \rho \)  
density  

\( \phi \)  
bank angle  

\( \psi \)  
heading angle  

\( \tau \)  
period  

\( \omega_n \)  
undamped natural frequency  

\( \zeta \)  
damping ratio  

\( ('') \)  
first derivative with respect to time  

\( (''') \)  
second derivative with respect to time  

**Subscripts**  

\( (o)_{0}, (\ast)_{e} \)  
reference condition, steady-state condition  

\( (E) \)  
Earth axis reference frame  

\( (w) \)  
wind axis reference frame  

\( (\ast) \)  
body axis reference frame  

\( (ph) \)  
phugoid mode  

\( (sp) \)  
short period mode  

\( (r) \)  
roll mode
APPENDIX 2

Non-Dimensional Stability Derivatives and Coefficients
<table>
<thead>
<tr>
<th>Derivative</th>
<th>15</th>
<th>25</th>
<th>35</th>
<th>50</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{m_o}$</td>
<td>0.1701</td>
<td>0.1504</td>
<td>0.1401</td>
<td>0.11995</td>
<td>0.10004</td>
</tr>
<tr>
<td>$C_{L_{max}}$</td>
<td>2.0</td>
<td>1.8</td>
<td>1.6</td>
<td>1.3</td>
<td>1.0</td>
</tr>
<tr>
<td>$C_{m_{\alpha}}$</td>
<td>-0.6</td>
<td>-0.8</td>
<td>-1.0</td>
<td>-1.2</td>
<td>-1.4</td>
</tr>
<tr>
<td>$C_{m_{\delta}}$</td>
<td>-1.8</td>
<td>-1.6</td>
<td>-1.5</td>
<td>-1.4</td>
<td>-1.2</td>
</tr>
<tr>
<td>$C_{m_q}$</td>
<td>-20.0</td>
<td>-25.0</td>
<td>-27.5</td>
<td>-20.0</td>
<td>-12.5</td>
</tr>
<tr>
<td>$C_{L_{\alpha}}$</td>
<td>5.7</td>
<td>5.4</td>
<td>5.1</td>
<td>4.2</td>
<td>3.6</td>
</tr>
<tr>
<td>$C_{L_q}$</td>
<td>6.0</td>
<td>7.0</td>
<td>8.0</td>
<td>8.25</td>
<td>5.0</td>
</tr>
<tr>
<td>$C_{L_{\delta}}$</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.9</td>
<td>0.8</td>
</tr>
<tr>
<td>$C_{L_{\alpha}}$</td>
<td>3.0</td>
<td>3.0</td>
<td>3.3</td>
<td>3.0</td>
<td>2.8</td>
</tr>
<tr>
<td>$C_{L_o}$</td>
<td>-0.0015</td>
<td>-0.0024</td>
<td>-0.00026</td>
<td>-0.000175</td>
<td>-0.00004</td>
</tr>
<tr>
<td>$C_{m_{\alpha}}$</td>
<td>-4.0</td>
<td>-4.0</td>
<td>-5.0</td>
<td>-5.5</td>
<td>-6.5</td>
</tr>
<tr>
<td>$C_{D_o}$</td>
<td>0.06</td>
<td>0.05</td>
<td>0.05</td>
<td>0.04</td>
<td>0.03</td>
</tr>
<tr>
<td>$C_{D_{\delta}}$</td>
<td>0.029</td>
<td>0.029</td>
<td>0.029</td>
<td>0.027</td>
<td>0.028</td>
</tr>
<tr>
<td></td>
<td>$C_{l_{\delta a}}$</td>
<td>$C_{l_{p}}$</td>
<td>$C_{L_{\delta_f}}$</td>
<td>$C_{D_{\delta_f}}$</td>
<td>$C_{m_{\delta_f}}$</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------</td>
<td>-------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>-0.085</td>
<td>-0.4</td>
<td>1.1428</td>
<td>0.012</td>
<td>-0.024</td>
</tr>
<tr>
<td></td>
<td>-0.085</td>
<td>-0.4</td>
<td>1.1428</td>
<td>0.011</td>
<td>-0.021</td>
</tr>
<tr>
<td></td>
<td>-0.085</td>
<td>-0.5</td>
<td>1.1428</td>
<td>0.010</td>
<td>-0.02</td>
</tr>
<tr>
<td></td>
<td>-0.0775</td>
<td>-0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-0.0575</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Etkin\cite{1} gives the following set of equations of motion for an aircraft

\begin{align*}
X' - mg\sin \theta &= m(\dot{u} + q\omega - rv) \\
Y' + mg\cos \theta \sin \phi &= m(\dot{v} + ru - pw) \\
Z' + mg\cos \theta \cos \phi &= m(\dot{w} + pv - qu) \\
L &= I_y \dot{p} - I_{zx} (\dot{\tau} + pq) - (I_y - I_z)qr \\
M &= I_y \dot{q} - I_{zx} (\dot{r}^2 - p^2) - (I_z - I_x)rp \\
N &= I_z \dot{r} - I_{zx} (\dot{p} - qr) - (I_x - I_y)pq \\
\dot{\phi} &= p + q\sin \phi \tan \theta + r\cos \phi \tan \theta \\
\dot{\theta} &= q\cos \phi - r\sin \phi
\end{align*}
\[ \psi = (q \sin \phi + r \cos \phi) \sec \theta \]
\[ \dot{\alpha}_x = q - q_w \sec \beta - p \cos \alpha_x \tan \beta - r \sin \alpha_x \tan \beta \]
\[ \dot{\beta} = r_w + p \sin \alpha_x - r \cos \alpha_x \]
\[ \dot{p}_w = p \cos \alpha_x \cos \beta + (q - \dot{\alpha}_x) \sin \beta + r \sin \alpha_x \cos \beta \]
\[ \dot{x}_E = V \cos \theta \cos \phi \]
\[ \dot{y}_E = V \cos \theta \sin \phi \]
\[ \dot{z}_E = -V \sin \theta \]

Figures 29 and 30 define the geometry involved.

The above equations were developed using the following approximations:

1. The Earth is a sphere rotating on an axis fixed in inertial space
2. \( g \) is a radial vector
3. The centripetal acceleration caused by the rotation of the Earth is neglected
4. The aircraft is a rigid body and has a plane of symmetry
5. The atmosphere is at rest
6. The surface of the Earth directly under the aircraft is considered to be flat
7. All rotor effects are neglected
8. \( g \) is a constant.
Figure 29. Aircraft geometry (adapted from [2]).
Figure 30. Relationship between body and wind axes.
As explained above, for the purposes of the simulation, lateral motion is restricted to roll motion only, e.g., v = 0, \psi = \dot{\psi} = 0, \beta = 0, etc. Thus, the Dutch roll and spiral modes are suppressed. With this restriction, the above equations of motion become:

\[
X' = m(\ddot{u} + qw) \\
\text{where } X' = L'\sin \alpha - D\cos \alpha \\
Z' = mg\cos \theta \cos \phi = m(\ddot{w} - qu) \\
\text{where } Z' = -L'\cos \alpha - D\sin \alpha
\]

\[
L = \frac{I\ddot{p}}{x} \\
M = \frac{I\dot{q}}{x} \\
\dot{\phi} = p + q\sin \phi \tan \theta \\
\dot{\theta} = q\cos \phi \\
\dot{\alpha}_x = q - \dot{\gamma} \\
\text{where } \dot{\gamma} = q_w \\
p_w = pc\cos \alpha_x
\]

The \( \dot{\chi}_E \) and \( \dot{Z}_E \) equations have been dropped since they are not needed for the simulation. Principal axes have been assumed.

The above equations of motion are nonlinear. However, linear approximations are used for the calculation of aerodynamic forces and moments, i.e., linear air reactions are assumed. Lift, for example, is considered to be a function of various other variables, i.e.,

\[
L' = f(\alpha, \dot{\alpha}, q, L'_c)
\]

Assuming that \( L' \) is a linear function of these variables,
the equation becomes:

\[ L' = f_1(\alpha) + f_2(\dot{\alpha}) + f_3(q) + \Delta L' \c_c \]

The individual \( f \)'s are defined as the dimensional stability derivatives. Inserting the stability derivatives into the above equation we obtain:

\[ L' = L'\alpha a_{abs} + L'\dot{\alpha} + L'q + \Delta L' \c_c \]

All the components of the aerodynamic forces and moments, \( X', Z', M, L, \) can be expanded in this manner.

The stability derivative data for the aircraft to be simulated are given in non-dimensional form. Therefore, these data must be converted to dimensional form to be used in the equations of motion. However, the aerodynamic moments are proportional to \( \rho V^2 L^3 \), and the aerodynamic forces are proportional to \( \rho V^2 L^2 \), allowing a relatively simple conversion between the dimensional and non-dimensional equations. Thus,

\[ X' = C_{x}\infty \rho V^2 \sin \alpha - C_{D}\infty \cos \alpha \]

The non-dimensional stability derivatives can, with appropriate conversions, be substituted for the dimensional stability derivatives, e.g., the equation for the lift coefficient becomes

\[ C_{L} = C_{L}\alpha a_{abs} + C_{L}\dot{\alpha} + C_{Lq} q\dot{\alpha} + C_{Lc} \]

where \( C_{Lc} = C_{L}\delta e + C_{L}\delta f \)

and \( a_1 \) is the time non-dimensionalizing term.
The equations used in the simulation are summarized below. The equations are dimensional with linear approximations for the aerodynamic forces and moments. Because of the omission of speed derivatives, proper control and dynamics will not be simulated in the transonic speed range.

Accelerations
\[ \dot{u} = -wq + \frac{X}{m} - g\sin \theta \]
\[ \dot{w} = uq + \frac{Z}{m} + g\cos \theta \cos \phi \]
\[ \dot{p} = \frac{L}{I_x} \]
\[ \dot{q} = \frac{M}{I_y} \]

Euler Angle Rates
\[ \dot{\theta} = q\cos \phi \]
\[ \dot{\phi} = p - qs\sin \phi \tan \theta \]

Aerodynamic Forces
\[ X' = C_x q \infty S \]
\[ Z' = C_z q \infty S \]
where \( C_x = C_L \sin \alpha - C_D \cos \alpha \)
and \( C_z = -C_L \cos \alpha - C_D \sin \alpha \)
Lift and Drag Coefficients

\[ C_L = C_{L\alpha} \alpha_{\text{abs}} + C_{L\dot{\alpha}} \dot{\alpha} + \frac{c}{2V} + C_{Lq} \frac{q}{2V} + C_{Lc} \]

where \( C_{Lc} = C_{L\delta_e} \delta_e + C_{L\delta_f} \delta_f \)

and \( \frac{c}{2V} = a_1 \), the time non-dimensionalizing term

for translational or linear velocities.

\[ C_D = C_{D0} + \frac{C_L^2}{\pi e_0 AR} + C_{Dc} \]

where \( C_{Dc} = C_{D\delta_e} \delta_e + C_{D\delta_f} \delta_f + C_{D\delta_g} \delta_g \).

Aerodynamic Moments

\[ M = C_m q_S \frac{s}{c} \]

where \( C_m = C_{m0} + C_m \alpha + C_m \dot{\alpha} + \frac{c}{2V} + C_m q \frac{c}{2V} + C_L (\text{c.g. - c.g. ref}) + C_{mc} \]

and \( C_{mc} = C_{mc\delta_e} \delta_e + C_{mc\delta_f} \delta_f + C_{mc\delta_g} \delta_g \)

\[ L = C_1 q_S \frac{s}{b} \]

where \( C_1 = C_{1p} p \frac{b}{2V} + C_{1c} \)

and \( C_{1c} = C_{1\delta_a} \delta_a \)

and \( \frac{b}{2V} = a_2 \), the time non-dimensionalizing term for the rolling velocity.
Velocities and Angles

\[ u = \int \dot{u} \, dt \]
\[ w = \int \dot{w} \, dt \]
\[ q = \int \dot{q} \, dt \]
\[ p = \int \dot{p} \, dt \]
\[ \theta = \int \dot{\theta} \, dt \]
\[ \phi = \int \dot{\phi} \, dt \]

\[ \alpha = \tan^{-1} \frac{w}{u} = \frac{w}{u} \]

\[ \hat{\alpha} = \frac{\dot{w} \dot{u} - \dot{u} \dot{w}}{u^2 + w^2} \]

\[ V = \sqrt{u^2 + w^2} \]

\[ \text{Mach no.} = \frac{V}{A} \]

where \( A \) is the speed of sound

\[ R/C = V \sin \gamma \]

\[ \gamma = \theta - \alpha \]
APPENDIX 3

Simulated Flight Test Conditions
## LONGITUDINAL MOTION

<table>
<thead>
<tr>
<th>CONDITIONS</th>
<th>VARIABLES</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>$\Lambda$</td>
<td>$\Lambda = 15, 25, 35, 50, 70 \text{ deg}$</td>
</tr>
<tr>
<td>6-9</td>
<td>$\delta f$</td>
<td>$\Lambda = 15^\circ \delta f = 10, 20, 30, 40 \text{ deg}$</td>
</tr>
<tr>
<td>10-13</td>
<td>$\delta f$</td>
<td>$\Lambda = 25^\circ \delta f = 10, 20, 30, 40 \text{ deg}$</td>
</tr>
<tr>
<td>14-17</td>
<td>$\delta f$</td>
<td>$\Lambda = 35^\circ \delta f = 10, 20, 30, 40 \text{ deg}$</td>
</tr>
<tr>
<td>15-19</td>
<td>$\delta f$</td>
<td>$\Lambda = 15, 25, 35, 50, 70 \text{ deg}$ $\delta g = 1$ (extended)</td>
</tr>
<tr>
<td>20-22</td>
<td>c.g.</td>
<td>$\Lambda = 15^\circ \text{ cg = .20, .25, .30 chord}$</td>
</tr>
<tr>
<td>23-25</td>
<td>c.g.</td>
<td>$\Lambda = 25^\circ \text{ cg = .30, .35, .40 chord}$</td>
</tr>
<tr>
<td>26-28</td>
<td>c.g.</td>
<td>$\Lambda = 35^\circ \text{ cg = .32, .37, .42 chord}$</td>
</tr>
<tr>
<td>29-31</td>
<td>c.g.</td>
<td>$\Lambda = 50^\circ \text{ cg = .35, .40, .45 chord}$</td>
</tr>
<tr>
<td>32-34</td>
<td>c.g.</td>
<td>$\Lambda = 70^\circ \text{ cg = .45, .50, .55 chord}$</td>
</tr>
<tr>
<td>35-39</td>
<td>wt, $\Lambda$</td>
<td>$\Lambda = 15, 25, 35, 50, 70 \text{ deg}$ $\text{ wt = 35000 lb}$</td>
</tr>
</tbody>
</table>

## LATERAL MOTION

<table>
<thead>
<tr>
<th>CONDITION</th>
<th>VARIABLE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>$\Lambda$</td>
<td>$\delta a = 10^\circ \Lambda = 15, 25, 35, 50, 70 \text{ deg}$</td>
</tr>
<tr>
<td>6-8</td>
<td>$\delta a$</td>
<td>$\Lambda = 25, \delta a = 1^\circ, 5^\circ, 10^\circ$</td>
</tr>
</tbody>
</table>
APPENDIX 4

Analysis of the Simulated Aerodynamic Model
Before the simulated flight tests began, the basic dynamic characteristics, i.e., $\omega_n$, $\zeta$, and $\tau$ were calculated. These results were then compared with those determined in the flight test. Some differences between the experimental data and the calculated data were expected since the predicted results were calculated using approximate methods. The following equations from Etkin[1] were used to calculate the predicted data.
Phugoid Mode:

\[
\tau_{ph} = \frac{\pi \sqrt{2} V_e}{g}
\]

\[
\omega_{n_{ph}} = \frac{\sqrt{2} g}{V_e}
\]

\[
\zeta_{ph} = \frac{1}{\sqrt{2}} \left( \frac{D}{L'} \right)_e
\]

Short Period Mode:

\[
\tau_{sp} = \frac{2 \pi \omega_{n_{sp}} \sqrt{1 - \zeta_{sp}^2}}{\omega_{n_{sp}}}
\]

\[
\omega_{n_{sp}}^2 = - \left( q \sqrt{S \frac{C_m}{I_y}} \right) - \left( q \sqrt{S \frac{C_m}{I_y}} \frac{c}{2V_e} \right)
\]

\[
2\zeta_{sp} \omega_{n_{sp}} = - \left( q \sqrt{S \frac{C_m}{I_y} 2V_e} \right) - \left( q \sqrt{S \frac{C_m}{I_y} 2V_e} \frac{c}{2V_e} \right)
\]

\[
- \left( q \sqrt{S \frac{C_m}{I_y} c} \right) - \left( q \sqrt{S \frac{C_m}{I_y} c} \frac{c}{2V_e} \right)
\]
Roll Mode:

\[ \tau_r = \frac{4 I_x}{(\rho S b^2 V_e C_1)} \]

The experimental and predicted phugoid mode data for the 25° sweep angle are presented in Table 6. The data show very good agreement, indicating that, at least for the long period motions, the simulator is performing as expected. Figure 31 gives an example of the output for the phugoid mode motions. The short period response has been omitted for clarity.

Figure 32 shows the effect of elevator deflection on equilibrium flight for two initial flight conditions, one well above the speed for minimum thrust required and the other just below the speed for minimum thrust required. For the 25° wing sweep angle configuration, the speed for minimum thrust required is 317 feet per second. The two speeds chosen are 450 feet per second and 300 feet per second. The results of this comparison are presented in Table 7. The graphical data show that the simulator is performing as an aircraft should. The \( \alpha \) and \( q \) variables settle to the same points. The rate of climb for the 450 fps condition shows an increase from the steady-state value while the R/C for the 300 fps condition shows a decrease, even though slight. In both cases, the airspeed decreases as expected.

Figure 33 presents a comparison of the phugoid mode data for various sweep angles. Table 8 is a presentation of the experimental and predicted
$V_e = 450$ fps

$\Lambda = 25^\circ$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Experimental</th>
<th>Calculated</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_{ph}, \text{sec}$</td>
<td>59.5</td>
<td>62.1</td>
</tr>
<tr>
<td>$\zeta_{ph}$</td>
<td>0.0918</td>
<td>0.106</td>
</tr>
<tr>
<td>$\omega_{n_{ph}}, \text{rad/sec}$</td>
<td>0.1058</td>
<td>0.1012</td>
</tr>
</tbody>
</table>

Table 6. Phugoid Mode Results
Figure 31. Phugoid mode results, 25° sweep.
Figure 32. Phugoid mode results, 25° sweep, different velocities.
<table>
<thead>
<tr>
<th>$V_e$, fps</th>
<th>$\tau_{ph}$ Experimental, sec</th>
<th>$\tau_{ph}$ Calculated, sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>450</td>
<td>59.5</td>
<td>62.1</td>
</tr>
<tr>
<td>300</td>
<td>41.92</td>
<td>41.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$V_e$, fps</th>
<th>$\omega_{ph}$ Experimental, rad/sec</th>
<th>$\omega_{ph}$ Calculated, rad/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>450</td>
<td>0.1058</td>
<td>0.1012</td>
</tr>
<tr>
<td>300</td>
<td>0.1502</td>
<td>0.1518</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$V_e$, fps</th>
<th>$\zeta_{ph}$ Experimental</th>
<th>$\zeta_{ph}$ Calculated</th>
</tr>
</thead>
<tbody>
<tr>
<td>450</td>
<td>0.0918</td>
<td>0.106</td>
</tr>
<tr>
<td>300</td>
<td>0.0664</td>
<td>0.0845</td>
</tr>
</tbody>
</table>

Table 7. Phugoid Results, $\Lambda = 25^\circ$
Figure 33. Phugoid mode results, various sweep angles.
### Table 8. Phugoid Results

<table>
<thead>
<tr>
<th>$\Lambda$, deg</th>
<th>$\tau_{ph}$ Experimental, sec</th>
<th>$\tau_{ph}$ Calculated, sec.</th>
<th>$V_e$, fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>58.56</td>
<td>61.69</td>
<td>447</td>
</tr>
<tr>
<td>25</td>
<td>59.5</td>
<td>62.1</td>
<td>450</td>
</tr>
<tr>
<td>35</td>
<td>60.8</td>
<td>61.69</td>
<td>447</td>
</tr>
<tr>
<td>50</td>
<td>60.4</td>
<td>61.41</td>
<td>445</td>
</tr>
<tr>
<td>70</td>
<td>60.4</td>
<td>61.27</td>
<td>444</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Lambda$, deg</th>
<th>$\omega_n$ Experimental, rad/sec</th>
<th>$\omega_n$ Calculated, rad/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.1079</td>
<td>0.1018</td>
</tr>
<tr>
<td>25</td>
<td>0.1058</td>
<td>0.1012</td>
</tr>
<tr>
<td>35</td>
<td>0.1041</td>
<td>0.1018</td>
</tr>
<tr>
<td>50</td>
<td>0.1042</td>
<td>0.1023</td>
</tr>
<tr>
<td>70</td>
<td>0.1041</td>
<td>0.1025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Lambda$, deg</th>
<th>$\zeta_{ph}$ Experimental</th>
<th>$\zeta_{ph}$ Calculated</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.107</td>
<td>0.123</td>
</tr>
<tr>
<td>25</td>
<td>0.0918</td>
<td>0.106</td>
</tr>
<tr>
<td>35</td>
<td>0.0872</td>
<td>0.106</td>
</tr>
<tr>
<td>50</td>
<td>0.0656</td>
<td>0.0889</td>
</tr>
<tr>
<td>70</td>
<td>0.0511</td>
<td>0.0719</td>
</tr>
</tbody>
</table>
Again, good agreement between the two is shown. In all cases, the phase relationships between $\theta$ and $\alpha$, $\theta$ and $q$, and $\theta$ and $u$ are correct. Also, the final values of $\alpha$ correspond to the values calculated from the non-dimensional stability derivatives. For example,

\[
\Delta \alpha = 25^\circ \\
C_{m\delta_e} = -1.6/\text{rad} \\
C_{m\alpha} = -0.8/\text{rad}
\]

where, in the steady-state

\[
\frac{\Delta \alpha}{\Delta \delta_e} = \frac{C_{m\delta_e}}{C_{m\alpha}} = \frac{-(-1.6)}{-0.8} = -2.0
\]

thus, if $\delta_e = -0.02$ rad or (-1.144$^\circ$), the $\alpha$ increase should be

\[
(-1.144) \times (-2.) = 2.288^\circ
\]

The experimental data yields a $\Delta \alpha$ of .04 rad or 2.288$^\circ$. Similar agreement is shown for all the phugoid conditions tested.

Table 9 presents the predicted and experimental results for the other conditions tested for the 25$^\circ$ wing sweep angle. Because no unexpected motions or data appeared in these tests, the data is not presented in graphical form.

The short period mode results for the 25$^\circ$ sweep angle are shown in Figure 34. The agreement between the experimental and predicted data for all wing angles, as shown in Table 10, is very good.
\[ A = 25^\circ \]

<table>
<thead>
<tr>
<th>( \delta f )</th>
<th>( \delta g ) wt, lb</th>
<th>c.g., chord</th>
<th>( \tau_{\text{Experimental, sec}} )</th>
<th>( \tau_{\text{Calculated, sec}} )</th>
<th>( V_e ), fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
<td>55K .35</td>
<td>45.76</td>
<td>43.61</td>
<td>316</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>55K .35</td>
<td>51.52</td>
<td>50.65</td>
<td>367</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>55K .35</td>
<td>59.84</td>
<td>61.65</td>
<td>446.7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>35K .35</td>
<td>58.24</td>
<td>61.93</td>
<td>448.75</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>55K .30</td>
<td>60.48</td>
<td>61.63</td>
<td>446.6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>55K .40</td>
<td>59.20</td>
<td>61.96</td>
<td>449.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \delta f )</th>
<th>( \delta g ) wt, lb</th>
<th>c.g., chord</th>
<th>( \omega_{\text{Experimental, rad/sec}} )</th>
<th>( \omega_{\text{Calculated, rad/sec}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
<td>55K .35</td>
<td>0.1377</td>
<td>0.1441</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>55K .35</td>
<td>0.1223</td>
<td>0.1241</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>55K .35</td>
<td>0.1027</td>
<td>0.1019</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>35K .35</td>
<td>0.1018</td>
<td>0.1012</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>55K .30</td>
<td>0.1002</td>
<td>0.1019</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>55K .40</td>
<td>0.1005</td>
<td>0.1014</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \delta f ), ( \delta g ) deg wt, lb</th>
<th>c.g., %chord</th>
<th>( \zeta_{\text{Experimental}} )</th>
<th>( \zeta_{\text{Calculated}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
<td>55K .35</td>
<td>0.0755</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>55K .35</td>
<td>0.0798</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>55K .35</td>
<td>0.1152</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>35K .35</td>
<td>0.1329</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>55K .30</td>
<td>0.0869</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>55K .40</td>
<td>0.848</td>
</tr>
</tbody>
</table>

Table 9. Phugoid Results, Various Conditions
Figure 34. Short period mode results, 25° sweep.
<table>
<thead>
<tr>
<th>θ, deg</th>
<th>τ&lt;sub&gt;sp&lt;/sub&gt; Experimental, sec</th>
<th>τ&lt;sub&gt;sp&lt;/sub&gt; Calculated, sec</th>
<th>V&lt;sub&gt;e&lt;/sub&gt; fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>3.97</td>
<td>4.54</td>
<td>447</td>
</tr>
<tr>
<td>25</td>
<td>3.72</td>
<td>3.86</td>
<td>450</td>
</tr>
<tr>
<td>35</td>
<td>3.36</td>
<td>3.48</td>
<td>447</td>
</tr>
<tr>
<td>50</td>
<td>3.20</td>
<td>3.30</td>
<td>445</td>
</tr>
<tr>
<td>70</td>
<td>2.92</td>
<td>3.06</td>
<td>444</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>θ, deg</th>
<th>ω&lt;sub&gt;sp&lt;/sub&gt; Experimental, rad/sec</th>
<th>ω&lt;sub&gt;sp&lt;/sub&gt; Calculated, rad/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1.58</td>
<td>1.654</td>
</tr>
<tr>
<td>25</td>
<td>1.79</td>
<td>1.897</td>
</tr>
<tr>
<td>35</td>
<td>1.96</td>
<td>2.064</td>
</tr>
<tr>
<td>50</td>
<td>1.96</td>
<td>2.050</td>
</tr>
<tr>
<td>70</td>
<td>2.14</td>
<td>2.144</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>θ, deg</th>
<th>ξ&lt;sub&gt;sp&lt;/sub&gt; Experimental</th>
<th>ξ&lt;sub&gt;sp&lt;/sub&gt; Calculated</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.535</td>
<td>0.548</td>
</tr>
<tr>
<td>25</td>
<td>0.503</td>
<td>0.516</td>
</tr>
<tr>
<td>35</td>
<td>0.472</td>
<td>0.486</td>
</tr>
<tr>
<td>50</td>
<td>0.364</td>
<td>0.373</td>
</tr>
<tr>
<td>70</td>
<td>0.279</td>
<td>0.283</td>
</tr>
</tbody>
</table>

Table 10. Short period results.
Table 11 gives the results of aircraft response to a +.175 rad step aileron input. This table compares the time to half amplitude for the roll mode. Agreement between the experimental and predicted results is, again, fairly good. The roll response of the simulator, a flying qualities parameter, is given in Table 12. These data indicate a quite fast roll response. This response is, perhaps, not as fast as it should be for a fighter-type aircraft, but it is very acceptable for the purposes of the CADM project.

Figure 35 presents the roll mode results in graphical form. As is to be expected, the \( p \) and \( \dot{\phi} \) data correspond very well. The most interesting result of these tests is that the roll response for the 35° sweep angle is the slowest, followed by that for the 25° sweep and then that for the 15° sweep angle. This would indicate that roll response in cruise (25 and 35 degree angles) is lower than that for slow speed flight (15°) and high speed flight (50 and 70 degrees). This is not, however, an unexpected result since the size of the \( C_{1p} \) derivatives for the various wing sweep angles indicate this result.

Overall, the results of the simulated flight test are quite good. Predicted and experimental data show good agreement in all conditions and modes. From the results of the various tests we can be sure that the simulator is responding as an actual aircraft should. The phugoid, short period, and roll mode data also give enough information to indicate that the simulator has satisfactory handling qualities in all modes.
<table>
<thead>
<tr>
<th>$\Lambda$, deg</th>
<th>$\tau_2$, Experimental</th>
<th>$\tau_2$, Calculated</th>
<th>$V_e$, fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.29</td>
<td>0.332</td>
<td>450</td>
</tr>
<tr>
<td>25</td>
<td>0.35</td>
<td>0.407</td>
<td>450</td>
</tr>
<tr>
<td>35</td>
<td>0.34</td>
<td>0.383</td>
<td>450</td>
</tr>
<tr>
<td>50</td>
<td>0.61</td>
<td>0.644</td>
<td>450</td>
</tr>
<tr>
<td>70</td>
<td>1.44</td>
<td>1.72</td>
<td>450</td>
</tr>
</tbody>
</table>

Table 11. Roll results.
<table>
<thead>
<tr>
<th>$\Lambda$, deg</th>
<th>Time to $30^\circ$, sec</th>
<th>$\Lambda$, deg</th>
<th>Time to $60^\circ$, sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1.09</td>
<td>15</td>
<td>2.05</td>
</tr>
<tr>
<td>25</td>
<td>1.20</td>
<td>25</td>
<td>2.08</td>
</tr>
<tr>
<td>35</td>
<td>1.28</td>
<td>35</td>
<td>2.16</td>
</tr>
<tr>
<td>50</td>
<td>1.57</td>
<td>50</td>
<td>2.59</td>
</tr>
<tr>
<td>70</td>
<td>1.74</td>
<td>70</td>
<td>2.66</td>
</tr>
</tbody>
</table>

Table 12. Flying Qualities Parameters, Roll Mode.
Figure 35. Roll mode results, various sweep angles.
REFERENCES


