Implementation of an Adaptive Controller System from Concept to Flight Test

Richard R. Larson*, John J. Burken†, and Bradley S. Butler‡
NASA Dryden Flight Research Center, Edwards, California, 93523

and

Steve Yokum§
West Virginia High Technology Consortium Foundation, Fairmont, West Virginia, 26554

The National Aeronautics and Space Administration Dryden Flight Research Center (Edwards, California) is conducting ongoing flight research using adaptive controller algorithms. A highly modified McDonnell-Douglas NF-15B airplane called the F-15 Intelligent Flight Control System (IFCS) was used for these algorithms. This airplane has been modified by the addition of canards and by changing the flight control systems to interface a single-string research controller processor for neural network algorithms. Research goals included demonstration of revolutionary control approaches that can efficiently optimize aircraft performance for both normal and failure conditions, and to advance neural-network-based flight control technology for new aerospace systems designs. Before the NF-15B IFCS airplane was certified for flight test, however, certain processes needed to be completed.

This paper presents an overview of these processes, including a description of the initial adaptive controller concepts followed by a discussion of modeling formulation and performance testing. Upon design finalization, the next steps are: integration with the system interfaces, verification of the software, validation of the hardware to the requirements, design of failure detection, development of safety limiters to minimize the effect of erroneous neural network commands, and creation of flight test control room displays to maximize human situational awareness.

Nomenclature

A	=	aircraft stability derivatives
ADI	=	attitude displacement indicator
AIU	=	avionics interface unit
API	=	application program interface
ARTS	=	Airborne Research Test System
B	=	aircraft control derivatives
B_a	=	basis function
C	=	neural network input category
CAT	=	Choose-a-Test
CC	=	central computer
CIU	=	cockpit interface unit
Cmd	=	command
DAG	=	Dial-a-Gain
DFRC	=	Dryden Flight Research Center
DLL	=	design limit load
DR	=	discrepancy report

* Systems Engineer, Flight Systems Branch, P.O. Box 273/4840D, AIAA Member.
† Controls Engineer, Controls and Dynamics Branch, P.O. Box 273/4840D, AIAA Member.
‡ Aerostructures Engineer, Aerostructures Branch, P.O. Box 273/48202A.
§ Chief Engineer, Airborne Research Test Systems, 1000 Technology Drive/Suite 1000.

FC = flight computer
FCC = flight control computers
G = adaptation gain
HW = hardware
IADS = Interactive Analysis and Display System
IFCS = Intelligent Flight Control System
I/O = input/output
K_d = derivative gain
K_i = integral gains
K_p = proportional gain
K_z = learning gains
K_{z_1} = learning gains integral
L = error-modification term
MCC = Mission Control Center
MPD = multi-purpose display
MPDP = multi-purpose display processor
NASA = National Aeronautics and Space Administration
NN = neural network
NVRAM = non-volatile random access memory
N_y = y-axis normal load factor
N_z = z-axis normal load factor
OFP = operational flight program
PAL = Pick-a-Limit
PID = parameter identification
p = roll rate
pid = proportional-integral-derivative
PIO = pilot-induced oscillation
PTC = portable test computer
PVI = pilot-vehicle interface
q = pitch rate
r = yaw rate
SES = simulation electric stick
SW = software
U_{ad} = augmentation command
U_{dd} = error compensation commands
u = surface positions
u_c = surface deflections commands
V&V = verification and validation
W = neural network weights
$\dot{W}$ = neural network weight derivative
W^T = neural network weight transpose
x = aircraft states
$\dot{x}$ = aircraft state derivative
$\dot{x}_e$ = aircraft state derivative command
x_e = state error
x_{ref} = angular body axis rate
$\dot{x}_{ref}$ = angular body axis acceleration
Z = neural net learning signal

I. Introduction

There is interest in investigating possible future technologies by performing flight-test research using neural network (NN) adaptive controllers. The use of NN and similar adaptive technologies in the design of highly fault- and damage-tolerant flight control systems shows promise toward increasing the survivability of future aircraft by maintaining the integrity of the flying qualities of the aircraft as much as possible in the presence of certain failures. To address this challenge, the National Aeronautics and Space Administration Dryden Flight Research Center (DFRC) at Edwards Air Force Base (AFB) (Edwards, California) formed the F-15 Intelligent Flight Control System (IFCS) program using a highly modified McDonnell-Douglas NF-15B test airplane, tail number 837, (Fig. 1), to conduct this research. The IFCS F-15 airplane incorporates a quadraplex all-digital flight control system that interfaces with the NN processor; the addition of canards; and thrust-vectoring nozzles.

Early in the F-15 IFCS program, a design decision was made to interface the NN controller in a single processor with the existing flight control computers (FCC). This single-string approach rendered the software and hardware less complicated, but added risk due to the lack of redundancy. This paper presents the processes used in developing adaptive controller algorithms throughout the progression of a flight-test program. There were many challenges toward making this research cost-effective, timely, efficient, and, most importantly, safe; an overview of the related tasks is presented. Some of the approaches taken may be applied to future flight-test activities.

II. Simulink® Algorithms

The neural adaptive flight control system, shown in Fig. 2, is based on the augmented model inversion controller developed by Calise, Lee, and Sharma,[1] and Rysdyk and Calise.[2] An explicit model-following scheme is used to achieve the desired handling qualities. This direct adaptive approach integrates feedback linearization theory with on-line learning sigma-pi NNs. These networks generate command augmentation signals to compensate for errors in the model inversion. A Lyapunov stability proof guarantees boundedness of the tracking error and network weights.

The pilot generates flight commands through longitudinal and lateral stick deflections and use of the rudder pedals. The flight commands are then filtered through reference models that produce angular body axis rate (x_{ref}) and acceleration ($\dot{x}_{ref}$) signals with the desired frequency and damping characteristics.[3] Dynamic inversion is then used to compute the necessary surface deflections commands (u_c) to achieve the desired accelerations. If the dynamic inversion and aircraft dynamics behaved together as a perfect integrator, then the response of the aircraft would match the reference models. Since errors are introduced by inaccuracies in the inversion, however, proportional-integral-derivative (pid) controllers are necessary to generate error compensation commands (U_{dd}), as shown in Eq. (1),

$$U_{dd} = K_i \int x_e + K_p x_e + K_d \dot{x}_e \tag{1}$$

where x_e is the state error, K_p is the proportional gain, and K_d is the derivative gain.

The NNs work in conjunction with the pid controllers by recognizing error patterns and learning how to compensate for them through the generation of augmentation commands (U_{ad}). The selected NN inputs consist of sensor feedback (for *A*-matrix failures), control commands (for *B*-matrix failures), and bias terms (for out-of-trim conditions). These inputs are normally separated into different categories (C), which are then combined together to form a basis function (B_a) using nested Kronecker products, as shown in Eq. (2).

$$B_a = kron\left(kron\left(C_1, C_2\right), C_3\right) \ldots \tag{2}$$

This basis function can then be used to update the NN weights (W) with an adaptation law, as shown in Eq. (3), and to compute the augmentation command (U_{ad}), as shown in Eq. (4).

$$\dot{W} = -G\left(Z B_a + L|Z|W\right) \tag{3}$$

$$U_{ad} = W^T B_a \tag{4}$$

The adaptation gain (G) specifies how fast the weights will adapt. The error-modification term (L) helps to contain the growth of the weights. The learning signal ($\square$) is computed as a function of error, as shown in Eq. (5), with learning gains (K_z) that are conditionally based upon the dynamic inversion pid controller gains. In this equation, the subscripts p and d refer to proportional and derivative, respectively.

$$Z = K_{z_i} \int x_e + K_{z_p} x_e + K_{z_d} \dot{x}_e \tag{5}$$

The role of dynamic inversion is provided by a pseudo-inverting Versatile Control Augmentation System (VCAS), developed by the Boeing Phantom Works (Saint Louis, Missouri) (The Boeing Company, Chicago, Illinois). The objectives of this improved adaptive controller algorithm, called Gen 2a, will now be discussed.

A. Generation 2a Enhancements

The performance objectives of the F-15 neural adaptive flight control tests include restoring model-following tracking performance, reducing cross-coupling effects, and reducing initial transients caused by failure insertion. For piloted aircraft applications, the NNs must adapt quickly enough to assist pilots in controlling a damaged aircraft, yet avoid learning transients that could interfere with the pilot's ability to control the aircraft during adaptation. The Gen 2a enhancements improved the performance of the adaptive system.[4] The design team processed Eqs. (1) through (5) into MATLAB® and Simulink® (both registered trademarks of The MathWorks, Inc., Natick, Massachusetts) block diagrams. Figure 3 shows the top level of the control system with NNs that were processed by Simulink® AutoCode into the C computer language.

From figure 3, the block diagram is broken down into a smaller AutoCode subsystem shown in Fig. 4. At this time the AutoCode is generated which is used in the six-degrees-of-freedom simulation and eventually in the aircraft FCC in the form of an operational flight program (OFP). The OFP is the final step after many tests have been performed in the six-degrees-of-freedom simulation, including pilot-in-the-loop testing.

III. Airborne Research Test System Software Development

The airborne research test system (ARTS) is a flight-ruggedized processor system dedicated to the execution of flight research software, in this case NN adaptive control. The ARTS was developed by the West Virginia High Technology Consortium (WVHTC) Foundation (Fairmont, West Virginia) in support of the IFCS. The architecture is a system of three single-board computers with a variety of input/output (I/O) options to support a large set of potential research initiatives. Each computer is a 400Mhz PowerPC with 256 MB of RAM. Two of the computers are dedicated to the execution of research; one computer handles the system interfaces. Inter-board communication is performed using the VME backplane. All single-board computers run the VxWorks® (Wind River Systems, Inc., Alameda, California) real-time operating system.

The baseline ARTS software load has no pre-knowledge of the research algorithms that it will be executing. An executive framework is in place to handle the communication with the aircraft data buses and an application program interface (API) mechanism allows research experiments to register their existence and communicate, via the I/O computer, with the aircraft flight controls. This architecture keeps the ARTS generic while providing an interface to the aircraft for the research experiments.

Research flown on the ARTS to date has been developed using the MathWorks Simulink® software development environment. The process of integrating the Simulink® model into the ARTS involves AutoCoding the model to the C computer language and later importing to a PowerPC. The model conversion process is shown in Fig. 5. A wrapper of C-language hand code is then written around the I/O of the model to receive and forward the aircraft state data to the AutoCoded model as well as transmit command augmentations from the model back to the aircraft flight controls. This is achieved using the APIs provided by the flight executive. Additionally, APIs are provided to record any required state data of the model to non-volatile random access memory (NVRAM) solid-state disk drives within the ARTS. These may be retrieved post-flight for additional analysis.

Validating proper integration of the Simulink® model is a three-phase process. The developer of the model provides a check-case with a set of inputs and a set of expected outputs. That check-case is first executed within the native Simulink® environment to validate that it matches on the machine that will perform the AutoCoding. The Simulink® model is then AutoCoded to the C language. The C code and the check-case are then executed again on a host processor (typically a UNIX derivative) to validate that the check-case still matches. The check-case validation

architecture is shown in Fig. 6. Once confidence is established in the C code, the code is compiled for VxWorks® and integrated with the ARTS APIs. The check-case is then executed using a hardware-in-the-loop simulation (HILS). This HILS is executed on a PC with 1553 I/O capabilities that simulate the 1553 buses on the aircraft. The results of the check-case are compared against the original for an acceptable tolerance. The check-case will never match exactly due to the fact that 64-bit floating point signals are scaled to 16 bits for insertion on the 1553 bus.

IV. Airborne Research Test System Design

The ARTS architecture showing its interface with the F-15 IFCS is shown in Fig. 7. The ARTS is a remote terminal (RT) to the FCC and the central computer (CC). It is a bus controller to the instrumentation systems and also features an analog multiplexer (AMUX) card to interface with analog signals. The NN experiment options are controlled by the pilot using the existing multi-purpose display (MPD) panel in the cockpit. This display panel allows a pilot vehicle interface (PVI) with the ARTS. The NN algorithm engagement options, along with variable gain sets for each function, can be controlled by the pilot. The gain sets are used to alter the states within the model and provide a means to configure the software without a recompile.

A. Pilot-Vehicle Interface

Three functions are used to select the experiment configuration and options: Pick-a-Limit (PAL), Dial-a-Gain (DAG), and Choose-a-Test (CAT). An image of the MPD showing these options is presented in Fig. 8.

The PAL is used to define the flight limits. There are two possible flight envelope ranges, depending on the PAL number selected, as shown in Table 1. If any of the parameters shown in Table 1 are outside of the flight envelope range, the system will downmode to the conventional flight control laws. A unique downmode flag is set on the 1553 bus and monitored in the mission control center (MCC).

For every PAL selected there is a DAG option which can be engaged to either create a particular type of failure in the FCC (stabilator lock or canard gain multiplier) or a surface excitation signal for parameter identification (PID) testing. There are eight DAG sets, one of which is shown in Table 2. In this example, various stabilator lock values or canard multiplier failures may be set by the FCC. An excitation signal may also be activated after a failure is set, or the excitation signal can be commanded independently.

The CAT page is used to engage a particular NN for flight-testing, as shown in Table 3. There were three NNs available in the ARTS OFP with possible beta sources as an input to the NN. The option to select multiple beta sources was added to the NN gain set files because of the uncertainty of the beta signal quality. This design allowed for particular configuration changes to be made without having to create and compile a new OFP. These configuration files called, CONFIG files, included unique options within the NN for each CAT.

B. CONFIG files

The configuration files for the NNs included options to modify various parameters such as gains, limits, deadzones for each feedback signal, upper and lower weight limits, surface limits, and flags. A software tool was developed to: ftp the existing configuration files from the ARTS OFP, replace the ones that had been modified, recompute the checksums, and create a new zipped TAR file for import to the ARTS OFP.

C. Non-volatile Random Access Memory

The ARTS OFP provided for a set of NN signals to be written to NVRAM within the processor. Each time the NN was engaged, a predefined data set began writing to the NVRAM and continued until the NN was disengaged. The data file was retrieved from the ARTS after the flight using a portable test computer (PTC) that interfaced with the ARTS using an Ethernet connection.

D. Log Files

In addition to the NVRAM files the ARTS wrote log files, which provided status information such as modes transitions, errors, file opening or closing status, et cetera. The log file was retrieved after each flight and was particularly useful for troubleshooting. An excerpt from a log file is shown immediately below. The log file segment shown has flagged transition events within the ARTS of an NN engagement (CAT 51).

01:29:14, 321483, Severity:SUCCESS, Task:exec, Procedure:state_change.c, Details:Exec transitioned from the DISENGAGED to the ENGAGED state because FC requested Engagement
01:29:52, 323816, Severity:INFORMATIONAL, Task:exec, Procedure:process_cats.c, Details:CAT value of 51 coupled
01:29:58, 324169, Severity:INFORMATIONAL, Task:exec, Procedure:mode.c, Details:CAT mode changed from NOT_COUPLED to COUPLED
01:30:09, 324802, Severity:INFORMATIONAL, Task:exec, Procedure:mode.c, Details:CAT mode changed from COUPLED to CL_NN

V. Formal Verification and Validation Testing

The verification and validation (V&V) of the new NN software release was performed in-house by NASA IFCS project personnel. An overview of the DFRC simulation architecture is shown in Fig. 9. The simulation featured a cockpit for piloted simulations with a projection screen for situational awareness, a full six degrees of freedom, hardware interfaces for the ARTS and the CC, and a functional MPD in the cockpit. A PTC was used to load and retrieve files in the ARTS. A PASS 3200 1553 bus analyzer was part of the PTC and was used for bus testing. All of the NN OFP V&V was performed with this system.

The software V&V process is shown in Fig. 10. A formal V&V test plan was written, which included absolute requirements that are referred to as "shall statements." Test procedures that contained simulation scripts for the test inputs and recorded output files were written for each requirement. This method produced more repeatability in the results and a faster, more automated execution of the procedures. All results referenced the test requirement "shall statement number." A PASS or FAIL criteria was applied to the test results by both the test conductor and a witness. A test results document containing subsets of the recorded testing data was written. All failed tests were documented in discrepancy reports (DR). The DRs were categorized as a required fix, a minor limitation, or a procedure work-around. When a failed test was encountered, the tests continued to completion. The only exception was if the failed test was unacceptable and required a new OFP; for these tests there were no DRs in this category, however, there were DRs that were corrected at the conclusion of the V&V by using a new configuration file that did not require a new OFP. A selected subset of retests were performed for regression testing to complete the final V&V. Other discrepancies were found that would have required a new OFP but were considered minor and consequently carried as DRs against the system and not fixed.

A. Verification and Validation Testers

Verification and validation tests were divided among the IFCS team according to the team members' area of expertise. The testing included: corrected DRs; new system requirements; comparisons of software versus hardware for no NN; NN(123) without failures; NN(123) with failures; 1553 bus testing; NVRAM; configuration files; piloted evaluations of the NN performanace and NN disengagement transients; simulated structural loads; OFP path coverage; and failure modes and effects testing (FMET).

B. Sample Verification and Validation Test Results

A sample test verification test result is shown in Figs. 11. Attitudes, rates and accelerations are shown in Figs. 11(a) and 11(b). The surface positions comparisons are shown in Fig. 11(c). The NN commands are shown in Fig. 11(d). These time histories show the comparisons of the all-software version of the NN versus the NN interfaced with the simulation from the ARTS hardware. A simulation script was used to generate pitch, roll, and yaw stick and rudder pedal doublets.

A sample of a piloted validation evaluation test card is shown in Fig. 12. This test was a handling qualities evaluation using one of the NN designs; the NN performed as expected. There were numerous cards flown to evaluate each NN for nominal, with induced DAG-type failures, various piloted maneuvers and flight conditions, disengage transients, and system-type fault insertions. These tests were all acceptable.

VI. Safety Monitors

The NN controller is a single-string interface to the FCC, therefore, the primary concern was that the worst-case failure would likely be an erroneous NN command hardover (rapid and sustained displacement of an aircraft aerodynamic control surface) to the FCC control laws. This type of command could result in excessive structural loads and *g* acceleration excursions to the aircraft. The pilot also imposed an additional requirement of no more than $\pm$ 0.5 *g* lateral and $\pm$ 2 *g* normal accelerations from trimmed flight for any disengagement transient. Safety monitors,

which would disengage the NN and revert to the conventional control laws for most failures, were added in the FCC processors. These safety monitors will now be described.

A. Pick-a-Limit Monitor

The PAL monitor generates a downmode from the NN when any limit is exceeded, as mentioned in the "Pilot-Vehicle Interface" section above. If any of these limits were exceeded, the system would revert to the conventional control system. Unfortunately, this monitor did not prevent the *g* limits from being exceeded for a NN hardover. A graphical representation of this problem is shown in Fig. 13. An NN hardover was inserted in the pitch-up direction, causing the PAL limiter to trigger a disengagement at 6 *g*. Due to the momentum of the aircraft, however, the response peaked at approximately 10.5 *g*. A similar result for the roll axis is shown in Fig. 14. In this case, the PAL safety monitor was never set because the lateral acceleration never hit the limit. These responses are both unacceptable, thus, a new monitor, called a floating limiter, was developed and added to the FCC software. The floating limiter proved to be a much better concept for controlling both the structural loads and *g* excursions. As shown in these Figs. 13 and 14, this new monitor limited the *g* excursion to approximately 2.3 *g*, which was far more effective in producing the desired results.

B. Floating Limiter

The floating limiter safety monitor[5] was required to allow full surface authority at the maximum actuator rates if required by the NN control laws while still protecting the pilot and aircraft from excessive forces in the case of a commanded hardover.

The floating limiter monitor automatically engages when the NN is activated, and works by computing the rate of change for each NN command and applying a moving upper and lower boundary limits window. The floating limiter structure is shown in Fig. 15. Full surface authority at the maximum actuator rates (within the moving window) is achieved but with the limitation that the stick must be pulled back no faster than the drift rate of the floating limiter. This limiter and procedure still allow for normal piloted maneuvers such as wind-up turns, which is a reasonable compromise. The window constantly tries to center about the NN command rate, but is programmed to move at a much slower drift than is possible from NN command rates. This algorithm provides the desired protection from a commanded hardover. The NN command rates within the window boundaries are allowed to change at maximum rates since the commands are not to any one direction long enough to cause problems. If the NN command does persist in one direction, however, it will "catch up" with the floating limiter boundary and will be temporally rate-limited to that value. When rate-limiting occurs, a persistence counter starts and a signal is sent to the NN control laws to stop learning. When the limit is reached for the persistence counter, a downmode command is generated. A maximum and minimum range limiter was also included that would cause a disengage.

One of the problems in implementing the floating limiter was the desire to make the monitor very tight when the NN is first engaged but no failures have been induced (a very small NN command), then open the limits after the failure is in a transition phase, and finally change the limits again after the failure has stabilized so the NN is allowed to generate the necessary commands to compensate for the failure without causing a downmode. This was solved by developing three floating limiter regions shown in Fig. 16. The set of constants used for the floating limiter is shown in Table 4. These parameters were tuned using the DFRC simulator to meet the $\pm$ 0.5 *g* lateral, $\pm$ 2 *g* normal accelerations, $\leq$ 80% design limit load (DLL) requirements. Drift rates are independently set according to the axis, failure type and magnitude. The drift, persistence, downmode, and range flags were added to a 1553 message from the FCC to the ARTS; these signals were also available on the instrumentation system for monitoring in the MCC.

VII. Loads Model

The adaptive control algorithms were tested against the allowable load envelope of NF-15B 837 by running batch simulations of the loads clearance maneuvers with the analytical loads model in the loop (1 *g* half-stick 360° degree rolls, 4 *g* wind-up turns, and 3 *g* loaded roll reversals). These tests caused the floating limiter to detect an excessive rate of change in the NN commands, which resulted in a disengagement and return to the baseline conventional control laws, hopefully before the loads exceeded 80% design limit load (DLL). Each batch simulation run consisted of taking a loads clearance maneuver and inserting the hardover command at one time frame in the simulation; this was repeated at successive 0.5-s intervals for the duration of the maneuver. Thus, a 20-s maneuver generated 40 ARTS hardover runs. Any test conditions that showed a simulated ARTS failure to generate over 80% DLL on any of the load stations were not flown unless further analysis indicated that the pre-programmed stick inputs were partially responsible for high accelerations or surface positions. In these cases, additional runs were

conducted using recorded pilot inputs to drive the batch simulation. If these showed less than or equal to 80% DLL the suspect test conditions were flown. It should also be noted that in a few cases the floating limiter was too efficient; it would downmode within a few time frames of engagement, or during the maneuver. This was typically due to exceeding the pitch rate floating limiter, and occurred on the largest stabilator-lock and canard multiplier cases. These difficulties were surmounted by using pre-recorded pilot input files to drive the batch simulation; the human pilot was able to compensate for the reduced stability at the higher canard multipliers and the pilot-induced oscillation tendency with the largest stabilator-lock. The 80% DLL cutoff was selected to give the airframe extra margin to compensate for uncertainty in the loads model and the adaptive controller algorithm.

The floating limiter algorithm prevents ARTS hardover commands from propagating to the FCC, thus preventing catastrophic load levels from high-rate hardover surface commands and accelerations. This is illustrated in Figs. 17 through 20. Figure 17 shows the N_z and N_y response due to a nominal 1 *g* trim condition with a canard failure multiplier of -0.5 inserted at 5 s and a hardover inserted at 15 s. With the floating limiter engaged, the peak N_z was approximately 2.5 *g* and the peak N_y was smaller than -0.25 *g*, without the limiter the N_z peaks at 11 *g* and the N_y at -2.5 *g*. Figure 18 shows the predicted loads on the forward fuselage station FS400. With the floating limiter engaged the loads are well below the 80% DLL limit; without the limiter the loads build to over 200% DLL for vertical shear and the vertical bending to lateral bending strength envelope. Figures 17 and 18 show a nominal 1 *g* trim condition with a left stabilator lock at trim deflection inserted at -0.5 s and a hardover inserted at 15 s. Figure 19 shows the N_z and N_y response. With the floating limiter engaged, the peak N_z was approximately 2.5 *g* and the peak N_y was smaller than -0.25 *g*; without the limiter the N_z peaks at 8.5 *g* and the N_y at -1.75 *g*. Figure 20 shows the predicted loads on the forward fuselage station FS400. With the floating limiter engaged, the loads are well below the 80% DLL limit; without the limiter the loads build to over -150% DLL for vertical shear and the vertical bending to lateral bending strength envelope peaked at almost -125% DLL.

VIII. Interactive Analysis and Display System Displays

The NASA DFRC decided to replace the MCC computers with PCs so that a real-time display system called the Interactive Analysis and Display System (IADS) developed by Symvionics (Arcadia, California)[8] could be used. Some of the MCC displays developed by the systems engineers are shown in Figs. 21 through 23. The basic F-15 systems display page is shown in Fig. 21. Signals such as aircraft health, mode, configuration, surface positions, and fuel quantities are shown. Failures were programmed to display in red. The very user-friendly IADS design environment enables quick prototyping of powerful display pages including derived parameters using C-type expressions. For example, when a BLIN code is set, it is programmed to blink to capture the attention of the systems engineer so another page may be selected to see the code description. The primary NN display page is shown in Fig. 22. This page shows the PVI parameters (PAL, DAG, and CAT) that have been selected. If any PAL limits have been exceeded, then that particular limit will turn red and latch until the pilot cycles the mode switch in the cockpit. The NN weights display page is shown in Fig. 23. The weights for each axis are displayed in both strip chart and tabular form. These simple examples show the ease with which a systems engineer can develop a user-friendly but sophisticated IADS display. Some of the IADS display builder widgets will now be discussed.

A. Alphanumeric Widgets

These display types were used heavily because derived parameters could easily be created, permitting multiple test strings to be displayed depending on the value of that signal. String colors were added depending on the signal values. A large amount of information could easily be processed and displayed using this data structure.

B. Strip Charts

Multiple parameters with unique colors were also used in a single strip chart. If a particular event was missed, the scroll bar on the far right was used to go back in time to see the event.

C. Slider Bars

A slider bar was implemented to help visualize the control surface motion in relation to the other surfaces. The slider bar is shown in Fig. 24.

D. Latched Functions

This feature was created as a derived parameter. For example, peak maximum and minimum NN commands were programmed in IADS to show the limit excursions. Latching reset logic was also performed in the IADS configuration files.

E. Flasher Functions

A flasher function was used on the IADS display to enhance human awareness of any serious problem requiring immediate response. Flashers were used to a F-15 BLIN code situation and if the display pages were not getting new data.

F. Built-in Displays

One of the built-in IADS displays for the ADI is shown in Fig. 25. This display provided good situational awareness to the control room of the airplane attitude.

IX. Flight-Test Results

This section discusses the comparison between the flight-test results and the simulation. The stick and rudder positions from flight were recorded and used as inputs to drive the simulation. The flight results are from the Gen 2a adaptive controller with a left stabilator failed at trim 7.5 s into the maneuver. The flight conditions were Mach .75 at an altitude of 20,000 ft. The flight response data (pitch and roll stick, rudder pedals, and throttle commands) were recorded from flight and back-driven into the simulation to see how closely the responses would match. Figure 26(a) shows the angle of attack (alpha) and bank angle (phi) from flight (the blue lines) and the simulation (the red lines). Figure 26(b) shows the flight roll rate, p, the pitch rate, q, and the yaw rate, r, (blue lines) and the simulation p, q, and r (red lines). Neural network outputs, which are the roll axis signal, the pitch axis signal, and the yaw axis signal, are shown in Fig. 26(c). The blue lines are from flight and the red lines are simulation results.

The flight-to-simulation comparison time histories match very well. The results show that the simulation did a very good job of predicting the behavior of the actual aircraft response. The only possible exception is the yaw axis NN signal shown in Fig. 26(c); however, upon closer examination of this signal the size or magnitude of the yaw axis signal can be observed to be very small. The yaw axis response is very difficult to match even in a healthy airplane.

Conclusion

The approach taken to investigate experimental adaptive control algorithms proved to be both cost-effective and safe. It was cost effective because of the simplified, single-string ARTS interface with the existing F-15 flight control computers and the pilot-vehicle interface allowed for flexibility of various options during the flight tests. All Gen 2a software was verified and validated using the Dryden Flight Research Center F-15 simulation. The pilot-vehicle interface design allowed for considerable flexibility to select neural network algorithms, failure combinations, and excitations systems as required for flight test.

The verification and validation of the neural network software was thorough, but not overly time-consuming. Some of the problems that were identified were corrected with new configuration files that did not require a new compile of the operational flight program. Other discrepancy reports were considered minor and were not fixed; these were handled using procedures to work around the problem.

Safety was a primary consideration to get to flight test. Any problem with the neural network would result in a downmode back to the conventional flight control computer control laws. The worst case that was considered was if the neural network commands were to go hard over (command a rapid and sustained displacement of an aircraft aerodynamic control surface). A floating limiter was designed to protect the structure forces of the airplane and to meet the pilot-imposed maximum *g* excursions. A loads model in the Dryden simulator was used extensively to validate the floating limiter disengage metrics.

Interactive analysis and display system display pages, developed by the flight-test engineers for monitoring in the Mission Control Center, proved to be easy to build and very effective. The derived functions capability within the interactive analysis and display system helped immensely to convert raw signal constructs into user-friendly formats that provided insight to both the neural network and the basic F-15 airplane systems.

References

[1]Calise, A. J., Lee, S., and Sharma, M., "Direct Adaptive Reconfigurable Control of a Tailless Fighter Aircraft," AIAA-98-4018, 1998.

[2]Rysdyk, R. T., and Calise, A. J., "Fault Tolerant Flight Control Via Adaptive Neural Network Augmentation," AIAA-98-4483, 1998.

[3]*Flying Qualities of Piloted Vehicles*, U.S. Department of Defense, MIL-STD-1797, March 31, 1987.

[4]Kaneshige, J., and Burken, J. J., "Enhancements to a Neural Adaptive Flight Control System for a Modified F-15 Aircraft," AIAA 2008-6986, 2008.

[5]Perhinschi, M. G., Napolitano, M. R., Campa, G., Seanor, B., Burken, J., Larson, R., "Design of Safety Monitor Schemes for a Fault Tolerant Flight Control System," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 1, No. 2., p. 562.

Tables

Table 1. Pick-a-Limit flight envelope limits.

Parameter	Envelope #1 (PAL 0-7)		Envelope #2 (PAL 8-15)	
	Lower limit	Upper limit	Lower limit	Upper limit
Angle of Attack	-4.0 deg	12.0 deg	-4.0 deg	12.0 deg
Sideslip Angle	-5 deg	5 deg	-5 deg	5 deg
Pitch Angle	-180 deg	180 deg	-180 deg	180 deg
Bank Angle	-90 deg	90 deg	-180 deg	180 deg
Pitch Rate	-45 deg/s	45 deg/s	-60 deg/s	60 deg/s
Roll Rate	-75 deg/s	75 deg/s	-300 deg/s	300 deg/s
Yaw Rate	-15 deg/s	15 deg/s	-60 deg/s	60 deg/s
Normal Acceleration[1]	-1 g	2.1 g	-2.0 g	5.0 g
Lateral Acceleration	-0.5 g	0.5 g	-1.0 g	1.0 g
Mach	0.55	0.95	0.55	0.95
Qbar	253 psf	733 or 550 psf2	253 psf	733 or 550 psf2
Altitude	15000 ft	35000 ft	15000 ft	35000 ft
Pitch Stick	-3.1 in	5.46 in	-3.1 in	5.46 in
Roll Stick	-4.0 in	4.0 in	-4.0 in	4.0 in
Yaw Pedal	-3.25 in	3.25 in	-3.25 in	3.25 in
Throttle (PLA)	16.5 deg	130 deg	16.5 deg	130 deg
Discrete Switches				PAL 15 only
Flap Up Mode	1U		1U	2 DM
Landing Gear Up	2 DM		2 DM	0 LGD
Throttle Switch Up	2 DM		2 DM	2 DM
Spin Mode	0 NS		0 NS	0 NS
Weight on Wheels	0 NGC		0 NGC	1 GC

Legend for Discrete Switches

U = up	NGC = no ground contact	LGD = landing gear down
DM = doesn't matter	NS = no spin	GC = ground contact

Table 2. Dial-a-Gain options.

PAL	DAG	Flying Qualities	Failure	Excitation	Qbar Limit
	20	Baseline	None	None	733 psf
	21	Baseline	4 deg Left Stabilator Lock	None	550 psf
	22	Baseline	2 deg Left Stabilator Lock	None	733 psf
	23	Baseline	0 deg Left Stabilator Lock	None	733 psf
	24	Baseline	-2 deg Left Stabilator Lock	None	733 psf
	25	Baseline	-4 deg Left Stabilator Lock	None	550 psf
	26	Baseline	-0.5 Canard Failure Multiplier	None	550 psf
1 or 8	27	Baseline	-0.2 Canard Failure Multiplier	None	733 psf
	28	Baseline	0 Canard Failure Multiplier	None	733 psf
	29	Baseline	0.1 Canard Failure Multiplier	None	733 psf
	30	Baseline	0.2 Canard Failure Multiplier	None	733 psf
	31	Baseline	0.4 Canard Failure Multiplier	None	733 psf
	32	Baseline	0.6 Canard Failure Multiplier	None	733 psf
	33	Baseline	0.8 Canard Failure Multiplier	None	733 psf
	34	Baseline	None	None	733 psf
	35	Baseline	None	None	733 psf

Table 3. Choose-a-Test options.

CAT #	Neural Network Type	Beta source
40	None	
41	NN1 Sigma Pi Baseline	NA
42	NN1 Sigma Pi Option 1	NA
43	NN1 Sigma Pi Option 2	NA
44	NN1 Sigma Pi Option 3	NA
45	NN1 Sigma Pi Option 4	NA
46	NN2 Scalar Baseline	Nose boom beta
47	NN2 Scalar Option 1	Filtered nose boom beta
48	NN2 Scalar Option 2	CC beta
49	NN2 Scalar Option 3	Estimated beta
50	NN3 Scalar Baseline	Nose boom beta
51	NN3 Scalar Option 1	Filtered nose boom beta
52	NN3 Scalar Option 2	CC beta
53	NN3 Scalar Option 3	Estimated beta
54	None	
55	None	

Table 4. Floating limiter constants metrics.

Rt Stab Failure From Trim fl_drift_table_conf_file[][][]	P axis	Q axis	R axis
0 deg and no fail, dps^3 transition final	150 500	50 90	0.03 0.01
+2 deg, dps^3 transition final	230 700	60 60	0.03 0.02
+ 4 deg, dps^3 transition final	430 850	60 60	0.03 0.09
-2 deg, dps^3 transition final	230 525	60 60	0.03 0.02
-4 deg, dps^3 transition final	430 550	60 60	0.03 0.09

Canard AOA Fails			
Set 1, dps^3 transition final	100 100	1 1	0.03 0.03
Set 2, dps^3 transition final	100 100	20 20	0.03 0.03

Metrics			
Initial drift, dps^3 fl_init_drift_conf_file[]	1.0	1.0	0.01
Delta, dps^2 fl_delta_conf_file[]	200	52	0.10
Range limit, dps^2 fl_hard_range_conf_file[]	775	300	0.2
Persistence time, s fl_persistence_time_conf_file[]	0.25	0.10	0.25
Transition time, s fl_transition_time_conf_file	3		

Figures

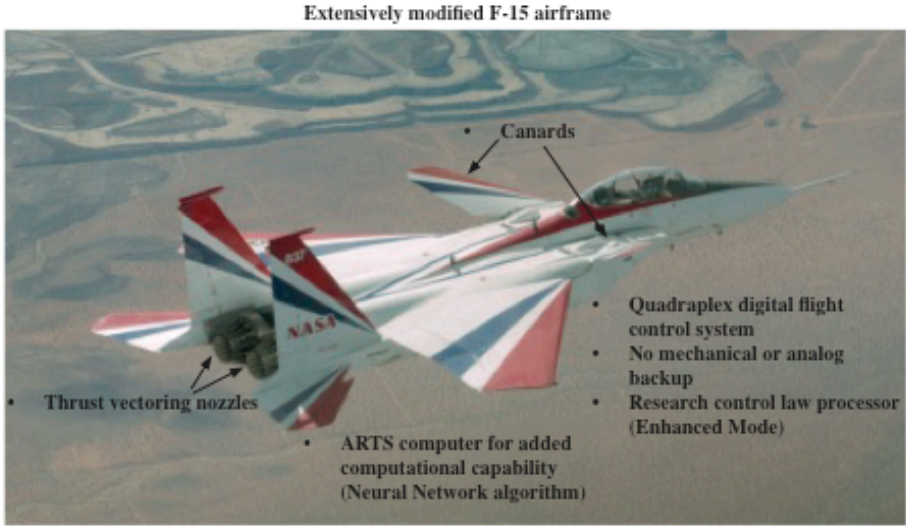


Figure 1. The F-15 Intelligent Flight Control System airplane, tail number 837.

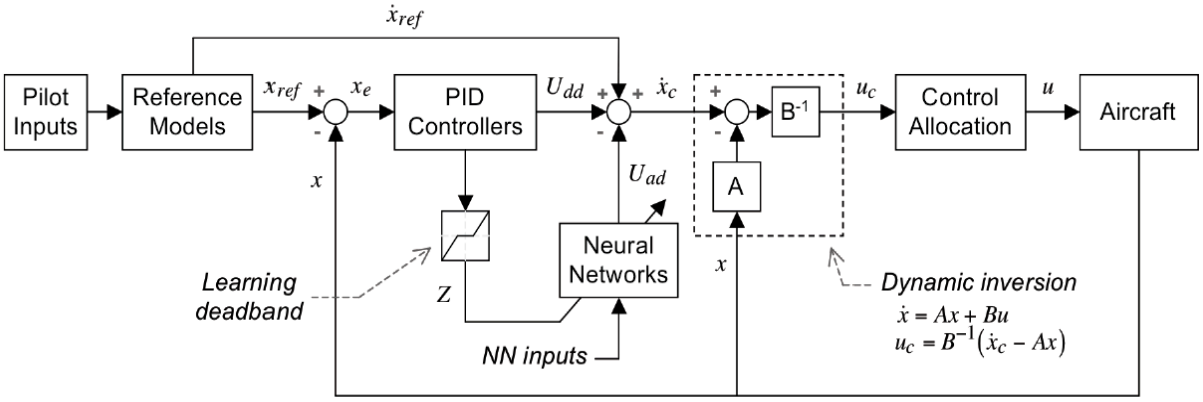


Figure 2. The neural adaptive flight control system architecture.

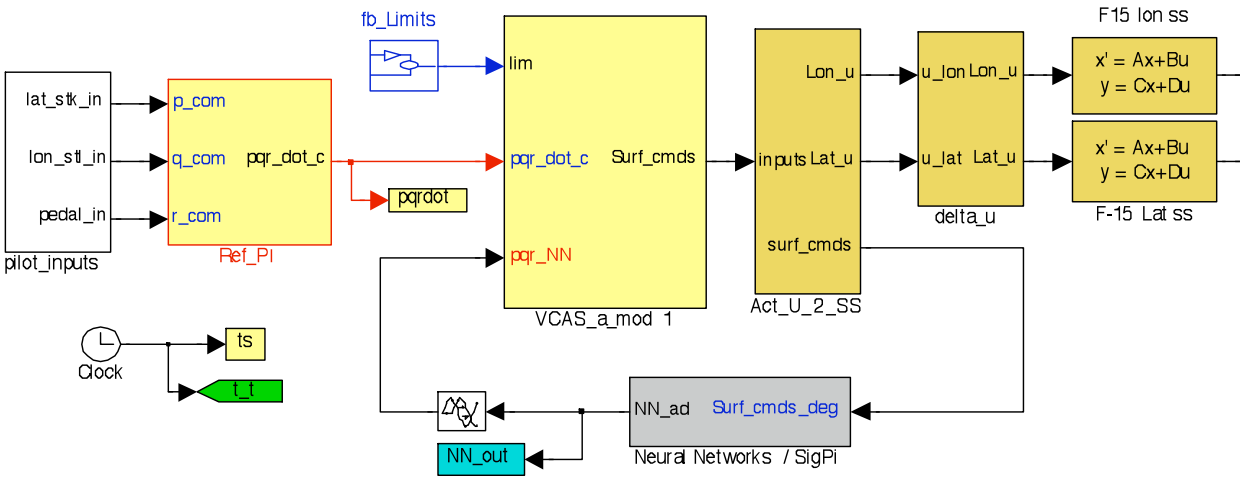


Figure 3. The top level of the flight control system with neural networks.

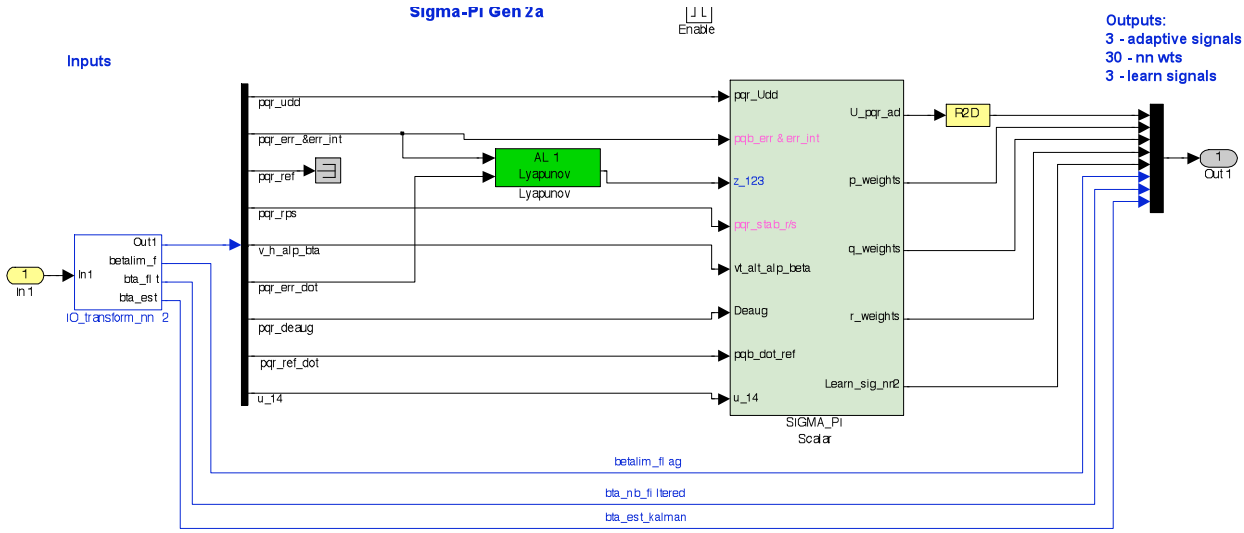


Figure 4. The Simulink® AutoCode block diagram.

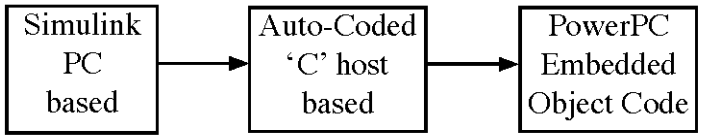


Figure 5. The Simulink® model conversion process.

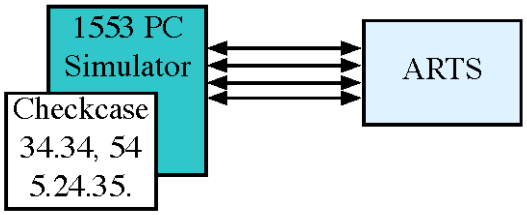


Figure 6. The check-case validation architecture.

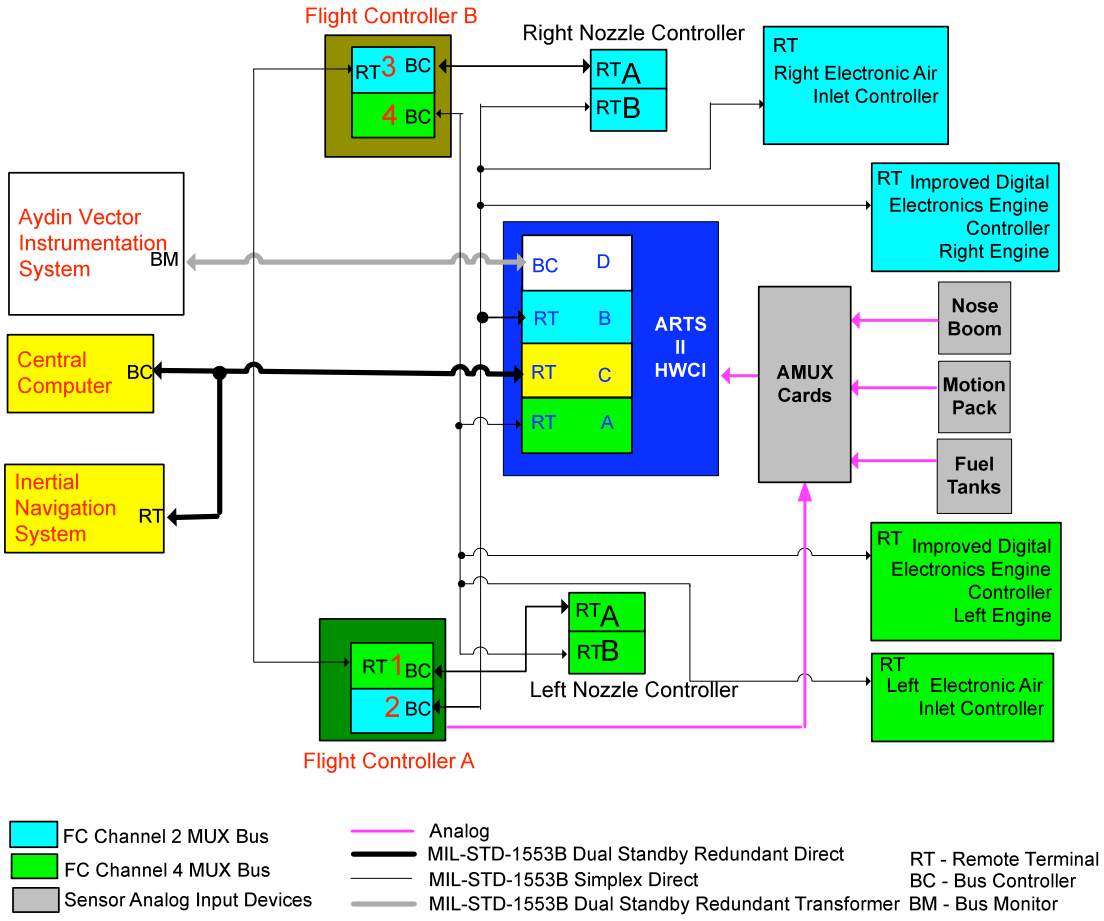


Figure 7. The F-15 Intelligent Flight Control System architecture.

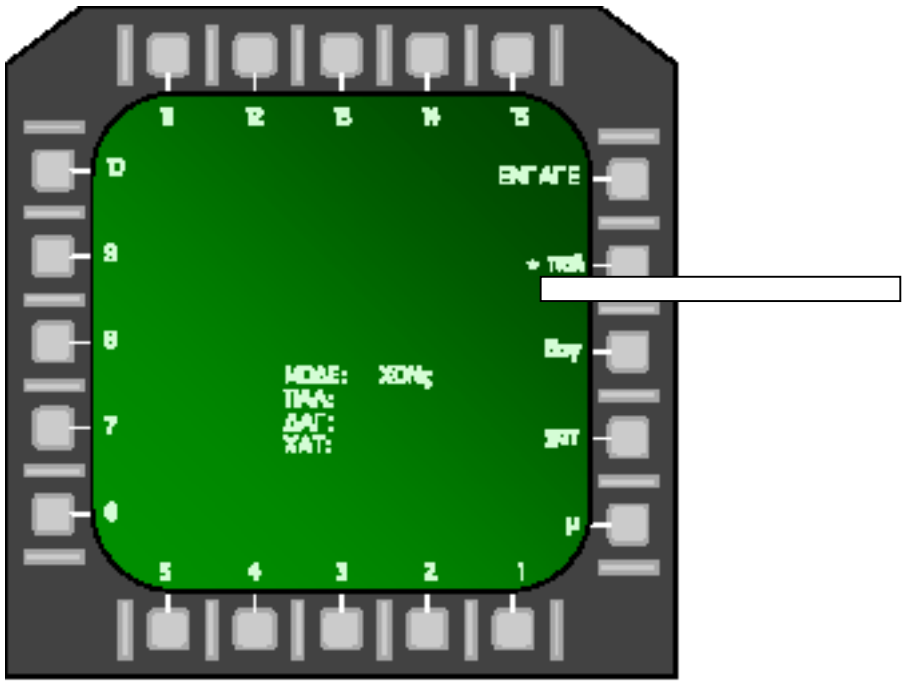


Figure 8. The multi-purpose display Intelligent Flight Control System functions.

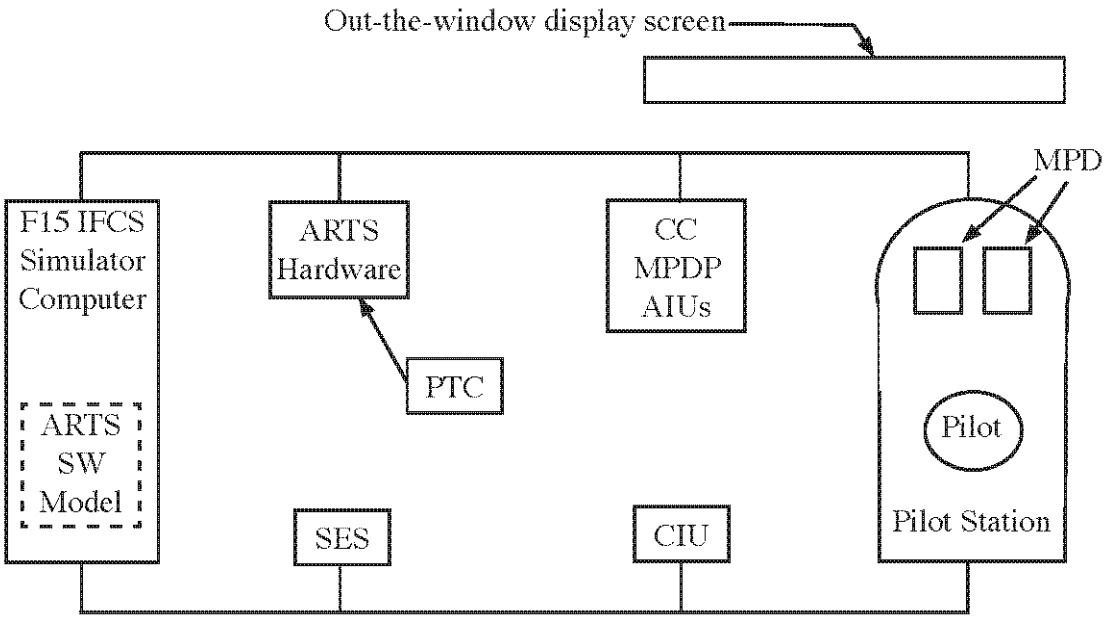


Figure 9. The F-15 Intelligent Flight Control System NASA Dryden simulation architecture.

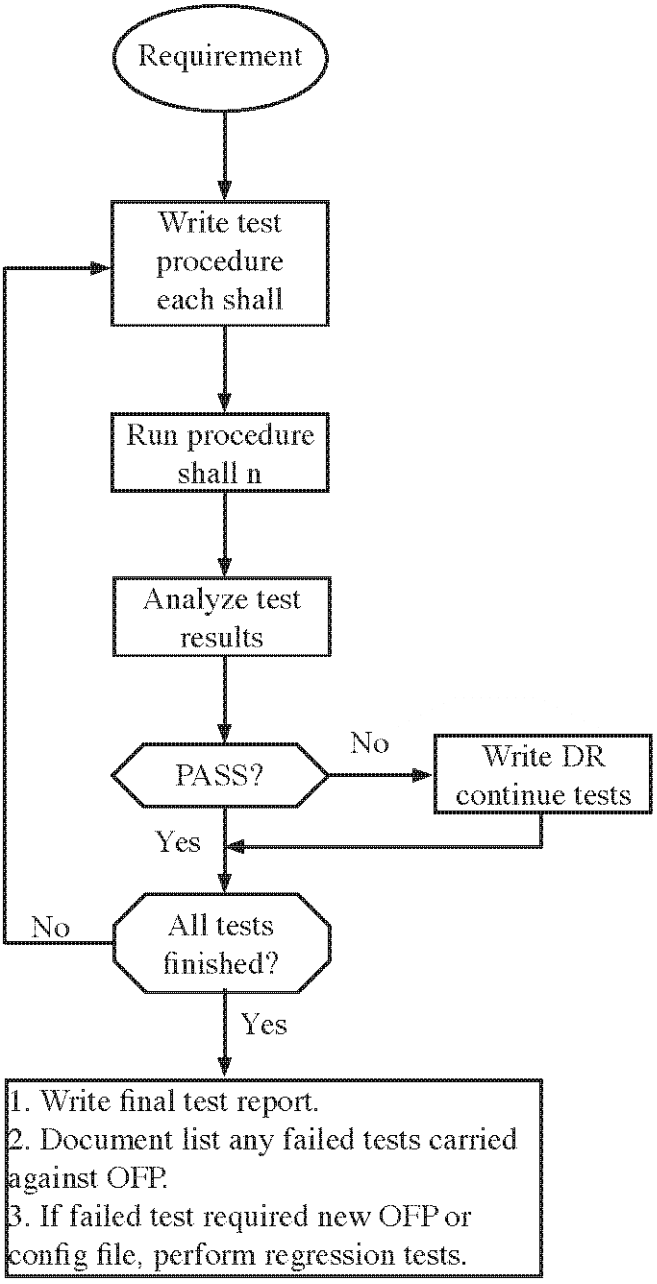
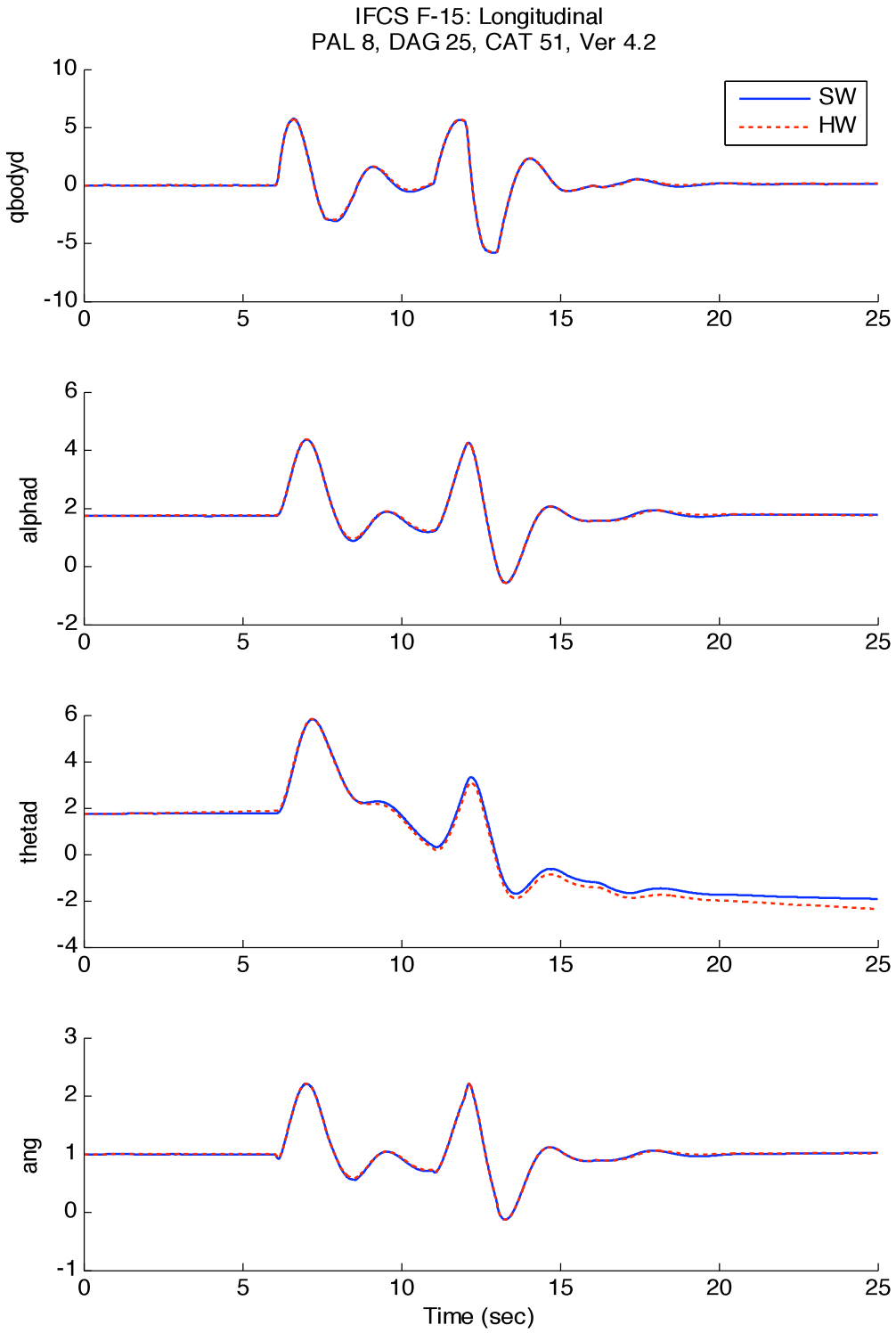
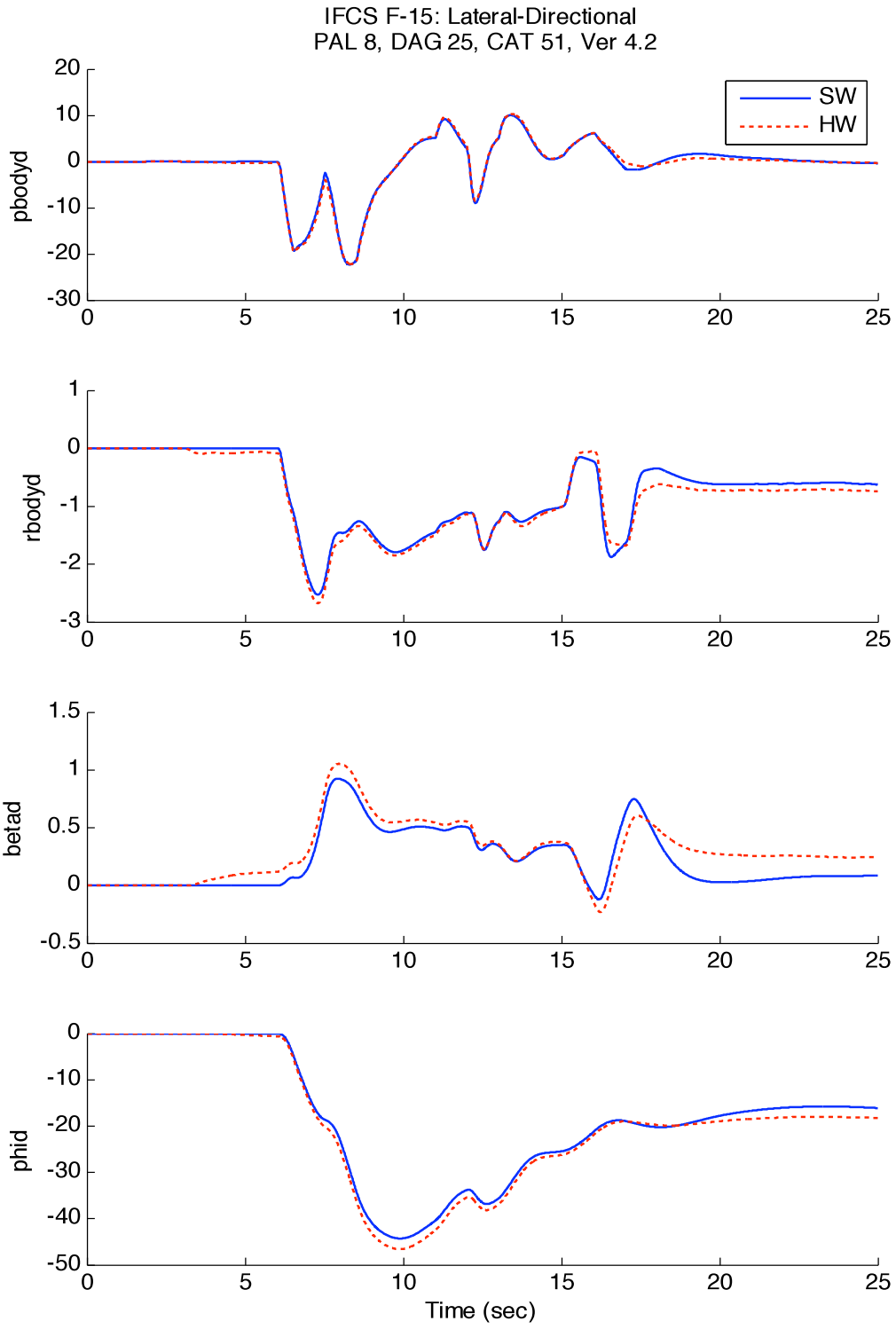


Figure 10. The software verification and validation process.



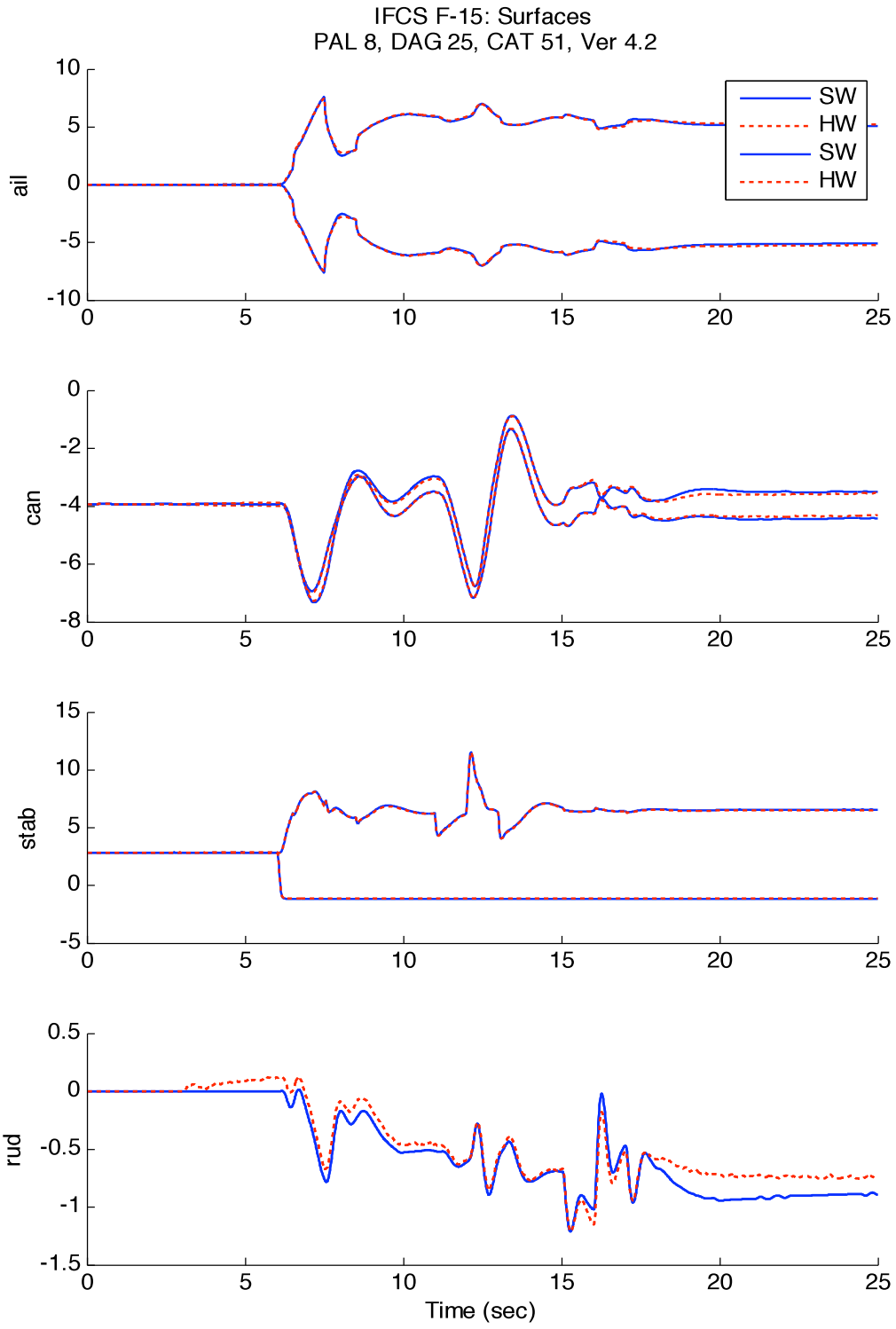
sw_arts_w_cockpit_p8_d25_c51_sw-hw73.cmp4
hw_arts_p8_d25_c51_sw-hw73.cmp4

Figure 11(a). Simulation comparisons of software versus hardware, Airborne Research Test System, set 1.



sw_arts_w_cockpit_p8_d25_c51_sw-hw73.cmp4
hw_arts_p8_d25_c51_sw-hw73.cmp4

Figure 11(b). Simulation comparisons of software versus hardware, Airborne Research Test System, set 2.



sw_arts_w_cockpit_p8_d25_c51_sw-hw73.cmp4
hw_arts_p8_d25_c51_sw-hw73.cmp4

Figure 11(c). Simulation comparisons of software versus hardware, Airborne Research Test System, set 3.

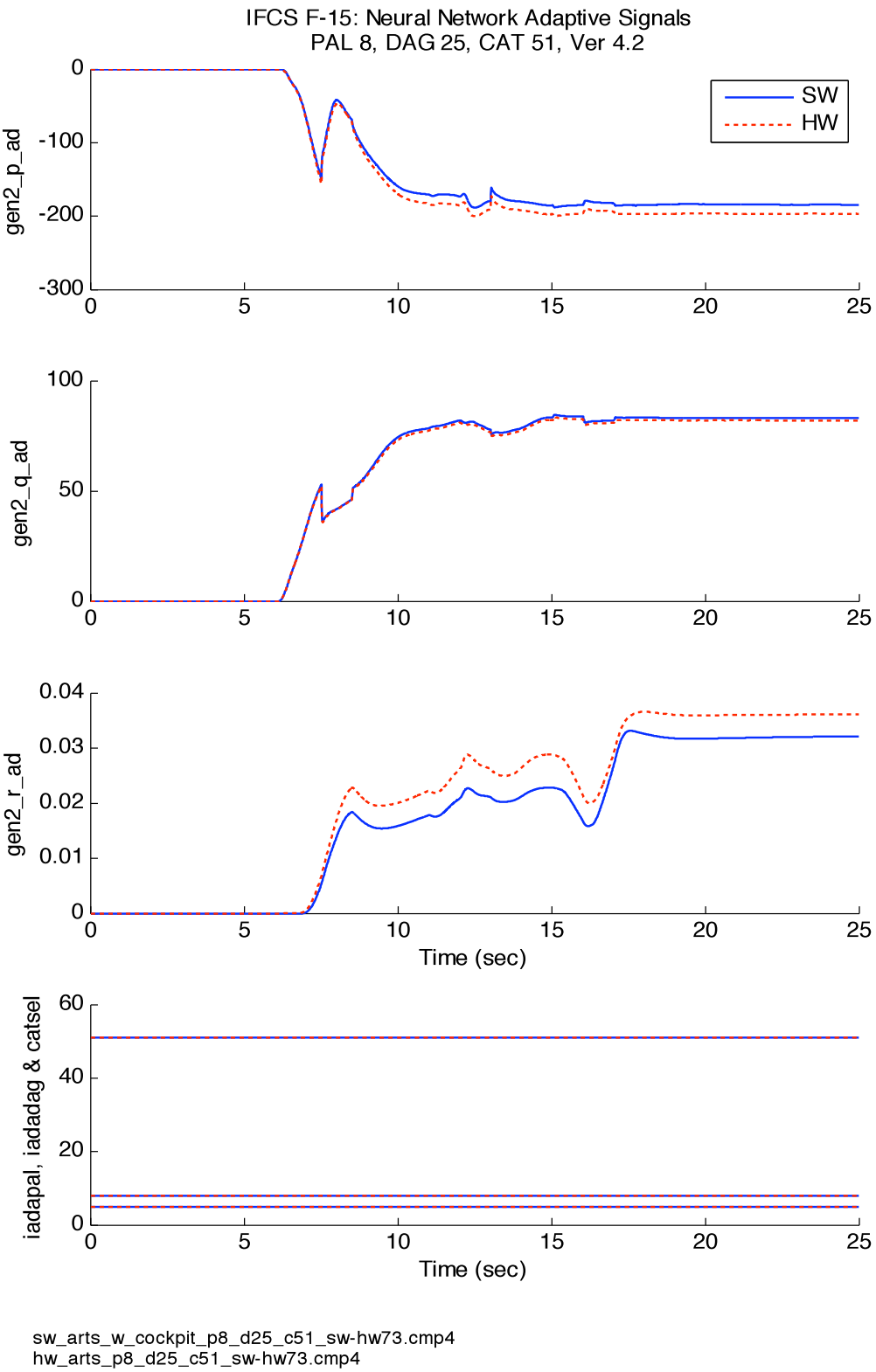


Figure 11(d). Simulation comparisons of software versus hardware, Airborne Research Test System, set 4.

Figure 11. Simulation comparisons of software versus hardware, Airborne Research Test System, sets 1 through 4.

TEST: **ENVELOPE TOLERANCE TEST WITH GEN-2A ENGAGED AND NO FAILURE** CARD: **HQ-02**

TEST PURPOSE: VERIFY THAT THE GEN-2A ADAPTIVE SYSTEM EXHIBITS ACCEPTABLE HANDLING QUALITIES FOR THE OFF-NOMINAL FLIGHT CONDITION, NO FAILURES

TEST NOTES: THIS TEST WAS PERFORMED BY AN ENGINEERING PILOT.

REQUIREMENTS: 6.7.1-1.

TEST RESULTS: ALL PAL, DAG AND CAT SELECTIONS WORKED AS EXPECTED, AND THE AIRPLANE ENGAGED AND DISENGAGED PROPERLY. AT THE BOTH THE HIGH-QBAR AND LOW-QBAR TEST CONDITIONS, THE AIRCRAFT EXHIBITED GOOD FLYING CHARACTERISTICS FOR THETA CAPTURES, BANK-TO-BANK TURNS AND 3-G WINDUP TURNS.

ENGAGE TRANSIENTS LEVELS: NONE

DISENGAGE TRANSIENTS LEVELS: (NZ AND NY IN G'S, R IN DEG/SEC)

Manuever	**High Qbar**			**Low Qbar**		
	nz	**ny**	**r**	**nz**	**ny**	**r**
theta captures	0.0	0.08	0.7	0.0	0.03	0.25
bank-to-bank turns	0.0	0.06	0.3	0.0	0.03	0.25
3-g wind-up turns	0.0	0.07	0.35	0.0	0.03	0.35

During the high- and low-qbar tests, yaw-axis weights #1, 3, 4 and 6 continued to grow throughout each test, producing a growing gen2_r_ad command. However, their values remained small (much less than one) and resulted in minor yaw rate disengage transients.

Figure 12. A sample piloted evaluation flight card.

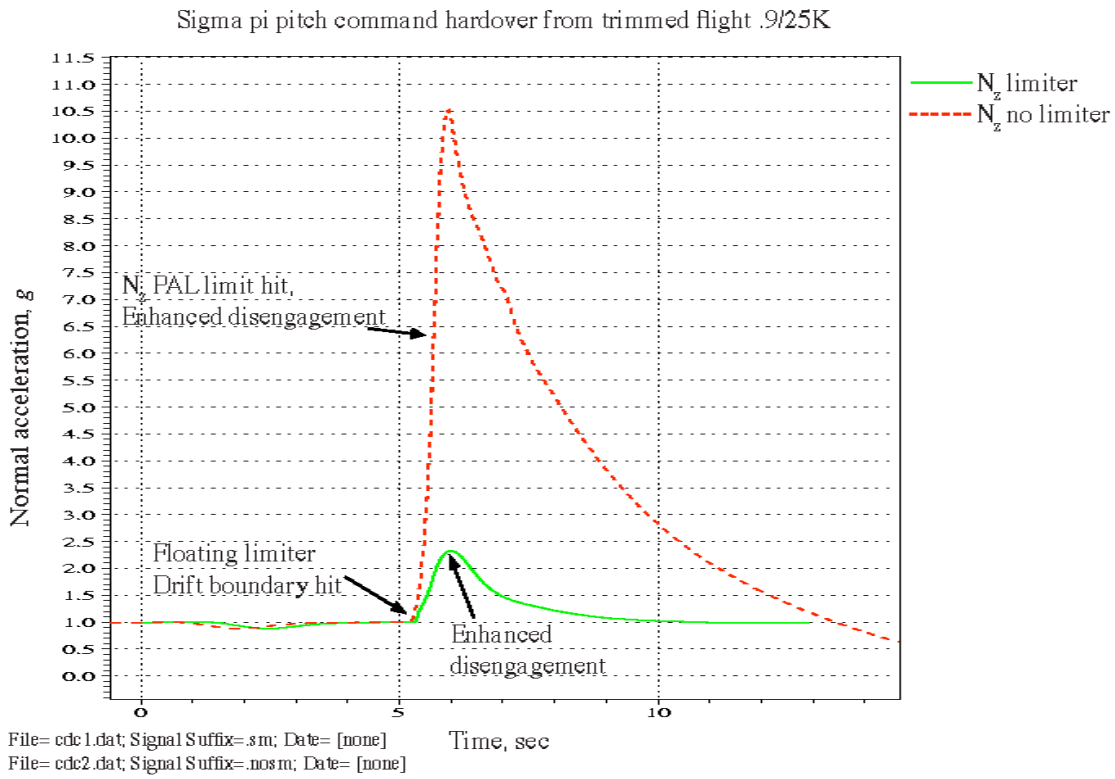


Figure 13. Normal acceleration response due to an Airborne Research Test System pitch hardover.

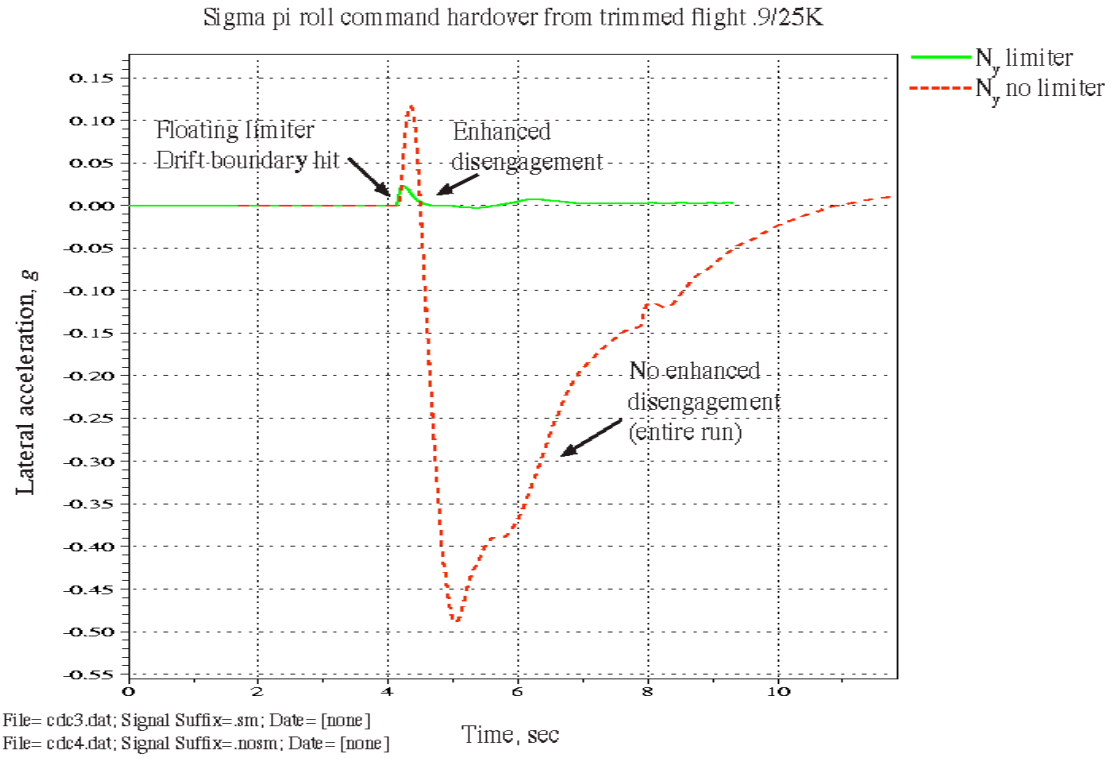


Figure 14. Lateral acceleration response due to an Airborne Research Test System roll hardover.

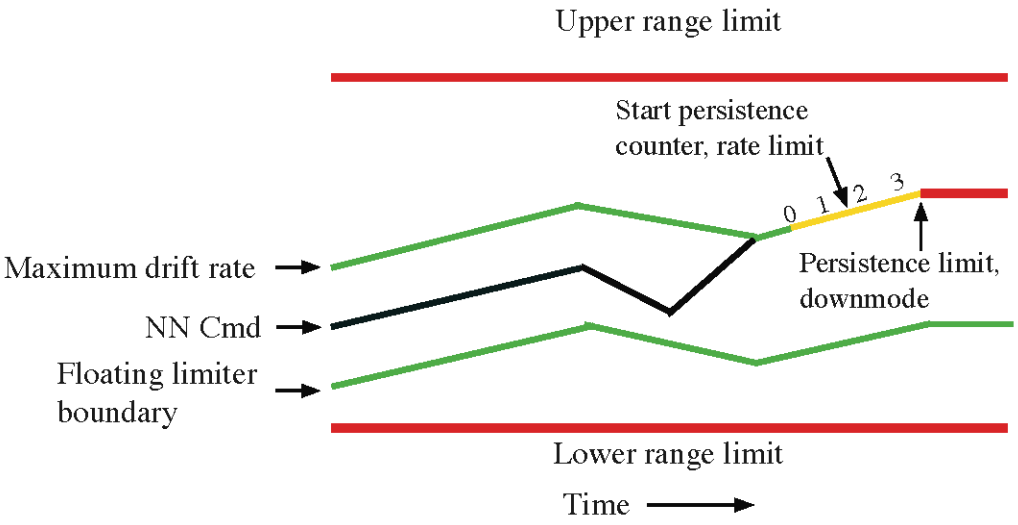


Figure 15. The floating limiter structure.

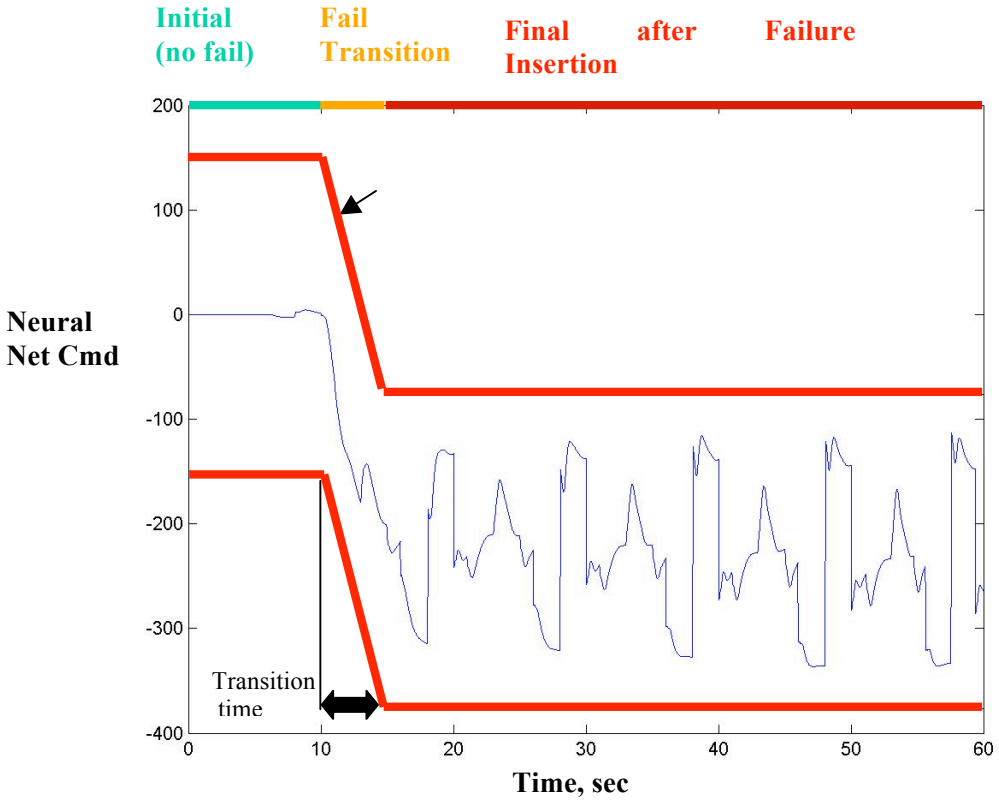


Figure 16. The floating limiter regions.

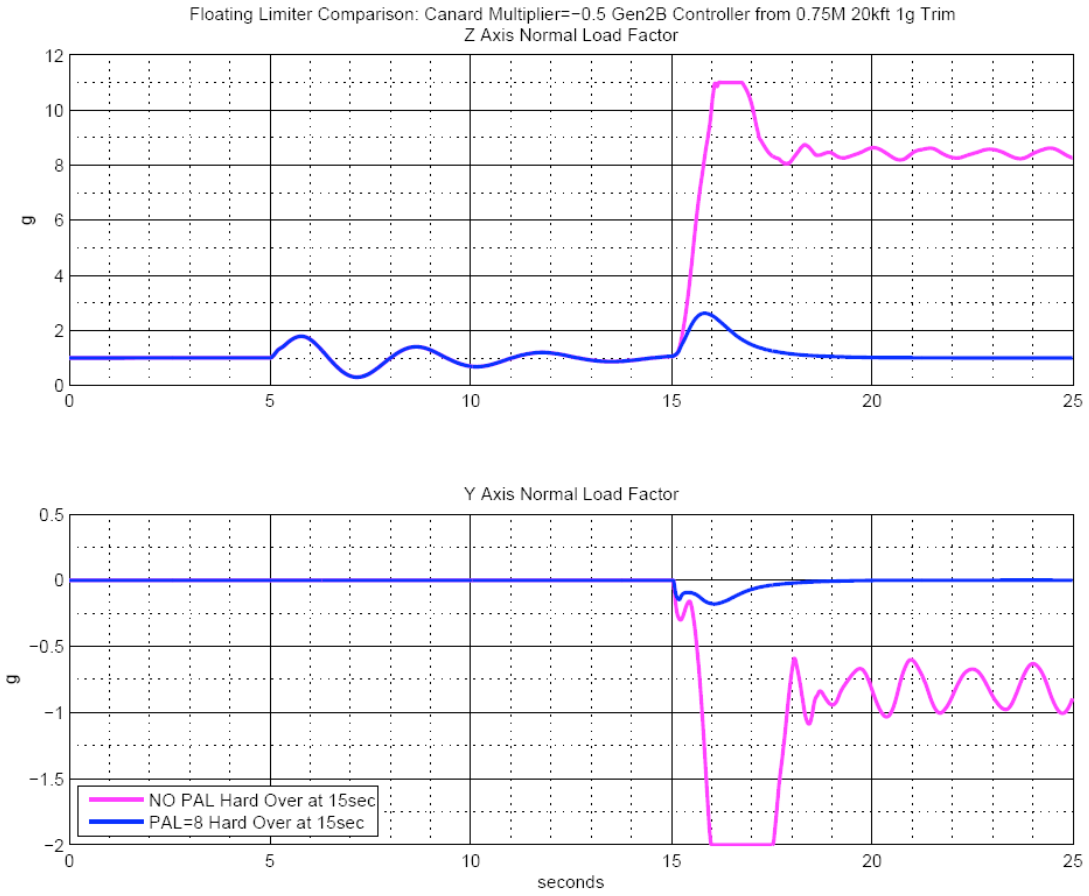


Figure 17. Simulated normal acceleration response, 1 *g* trim, canard multiplier of -0.5, with Airborne Research Test System hardover inserted at 15 s (PAL = 8, DAG = 28, CAT = 51).

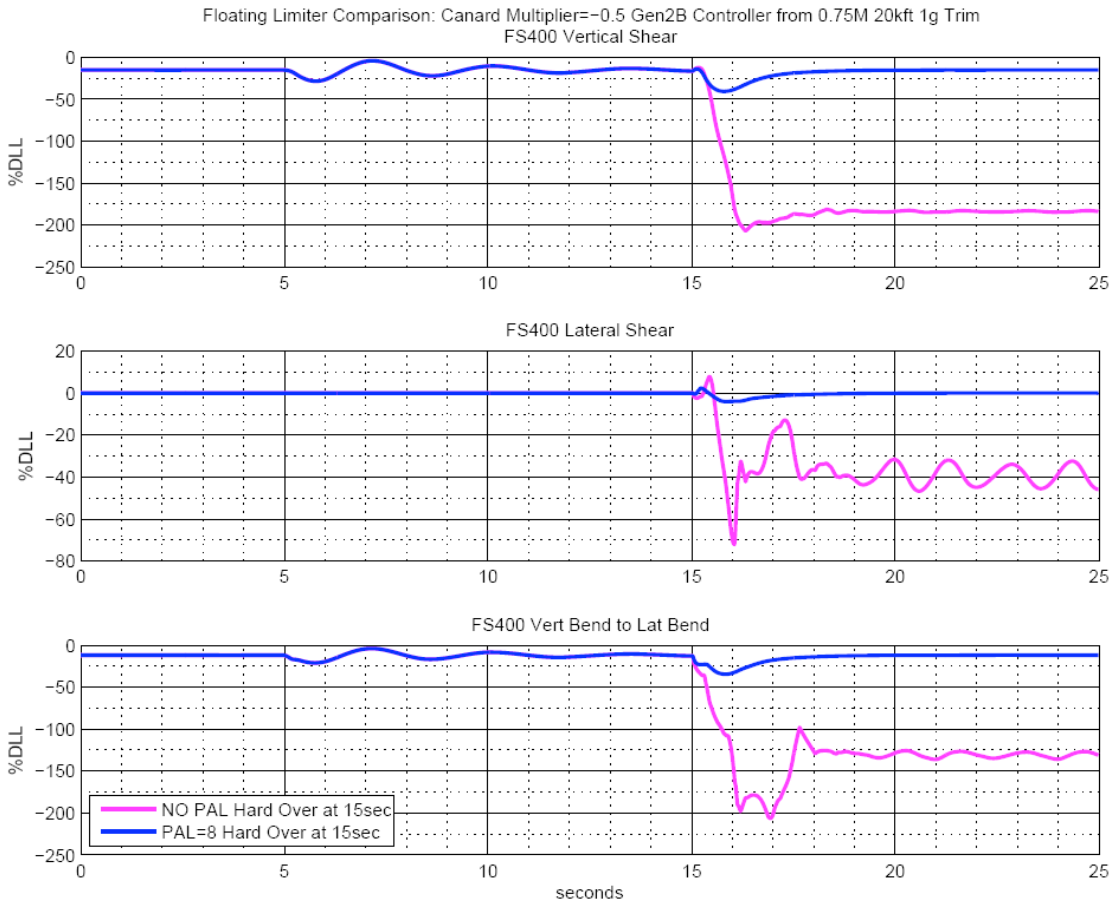


Figure 18. Simulated forward fuselage loads, 1 *g* trim, canard multiplier of -0.5, with Airborne Research Test System hardover inserted at 15 s (PAL = 8, DAG = 28, CAT = 51).

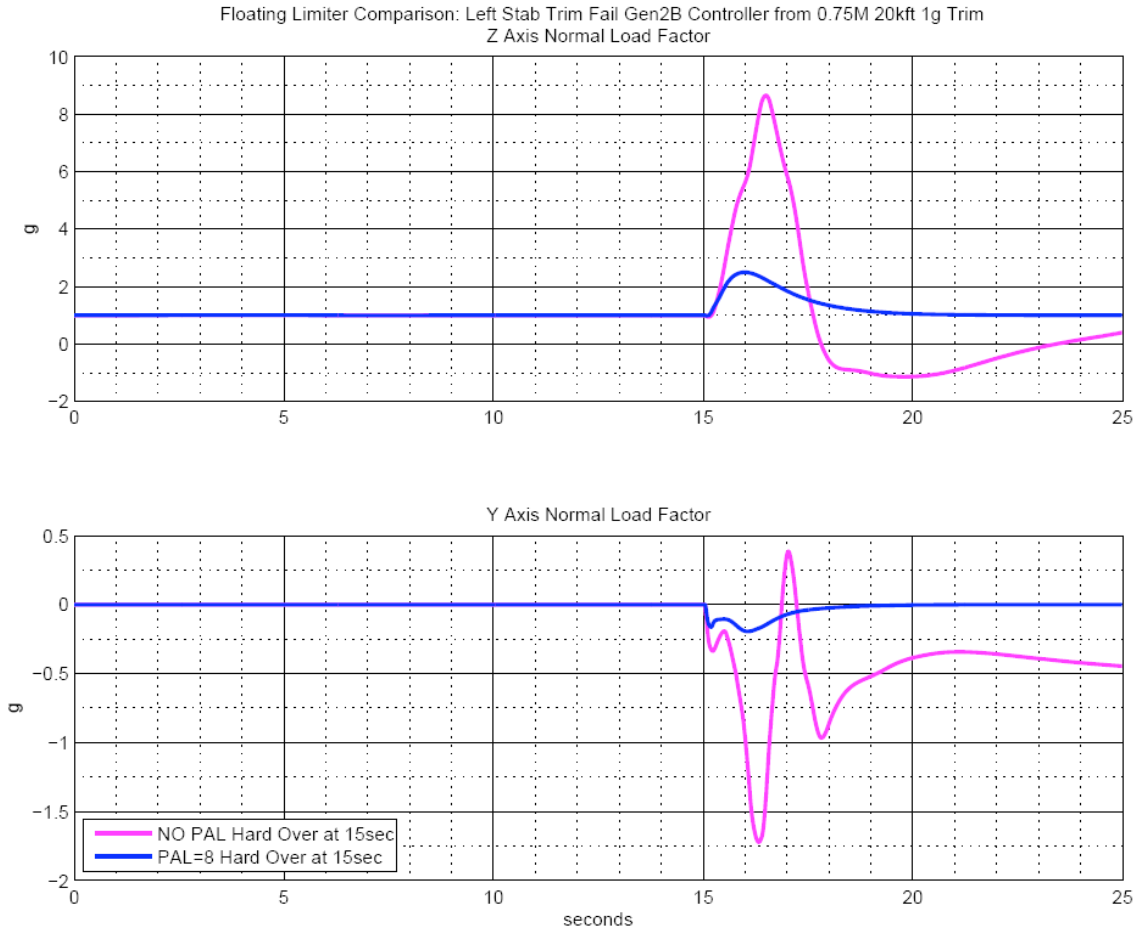


Figure 19. Simulated normal acceleration response, 1 *g* trim, left stabilator locked at trim deflection, with Airborne Research Test System hardover inserted at 15 s (PAL = 8, DAG = 23, CAT = 51).

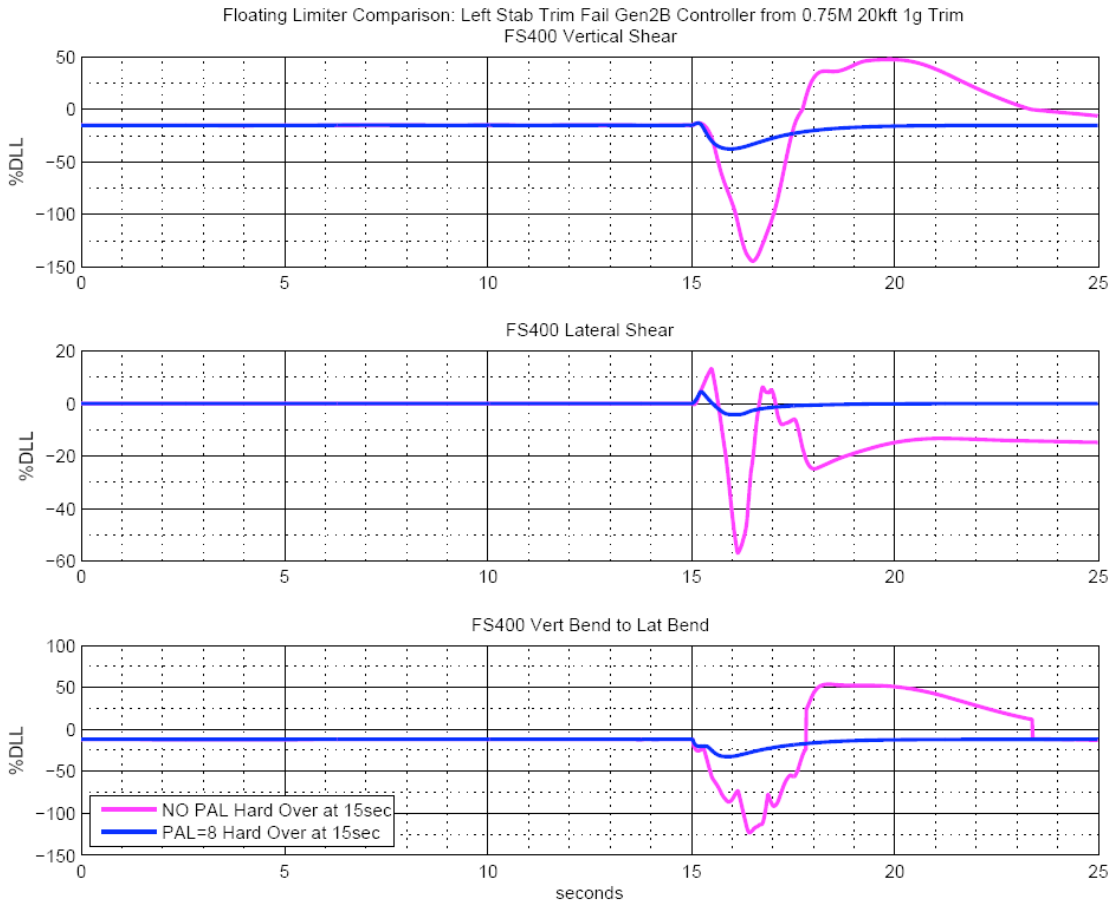


Figure 20. Simulated forward fuselage loads, 1 *g* trim, left stabilator locked at trim deflection, with Airborne Research Test System hardover inserted at 15 s (PAL = 8, DAG = 23, CAT = 51).

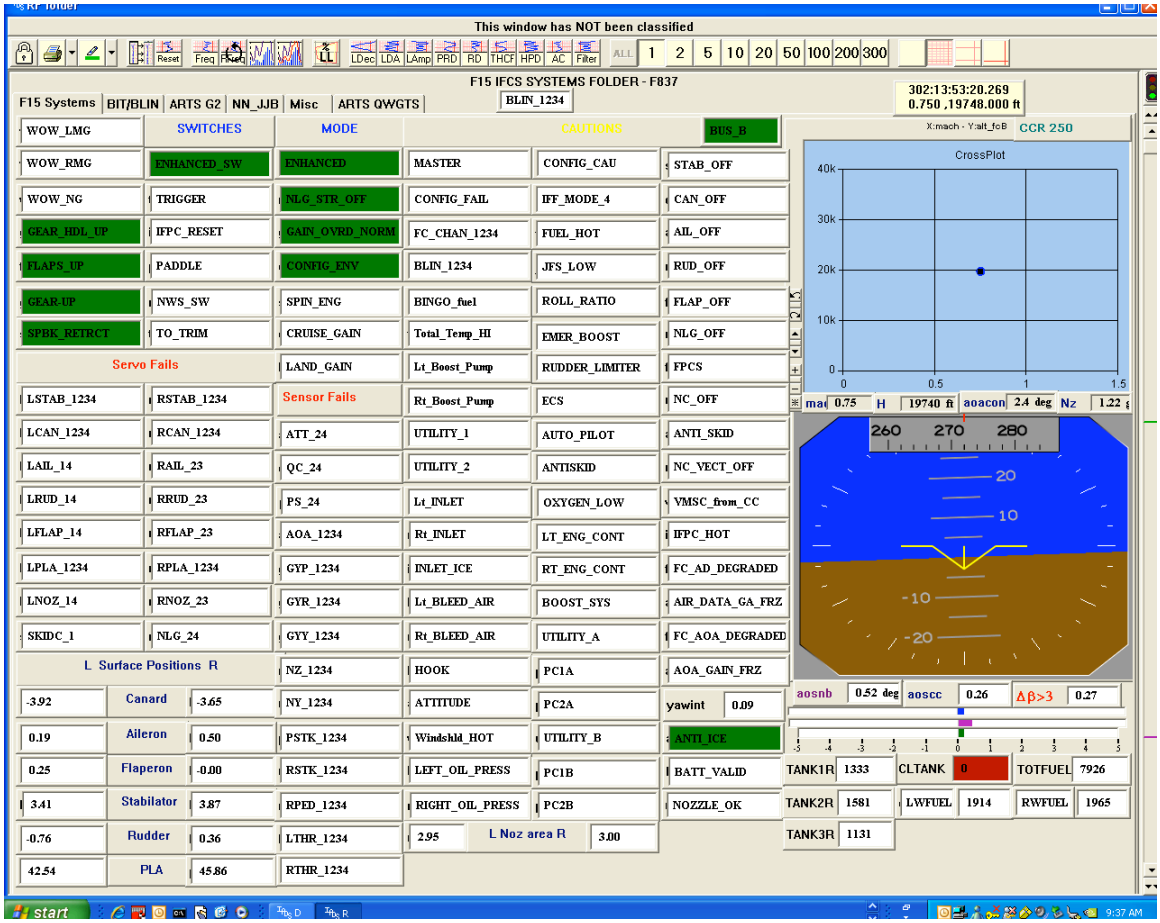


Figure 21. The interactive analysis and display system: the basic F-15 systems display page.

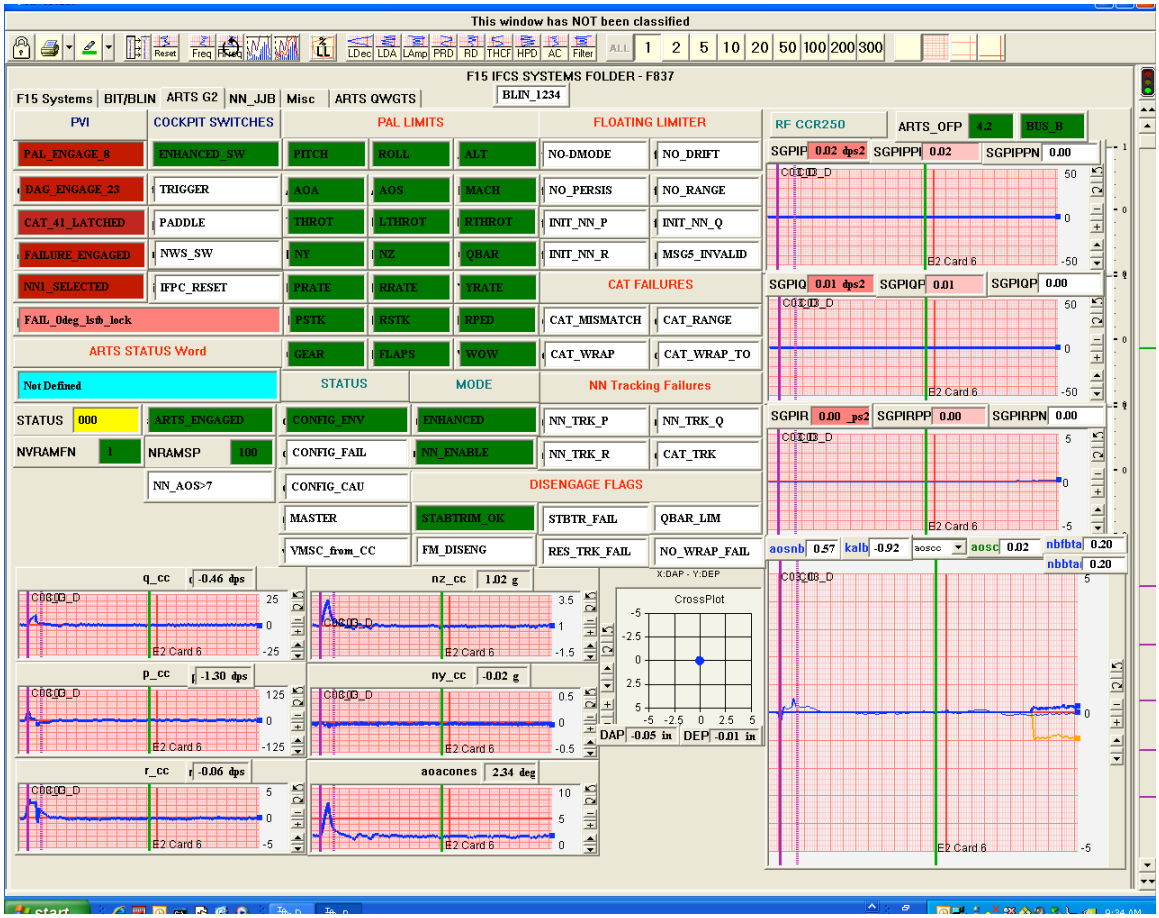


Figure 22. The interactive analysis and display system: neural network systems page 1.

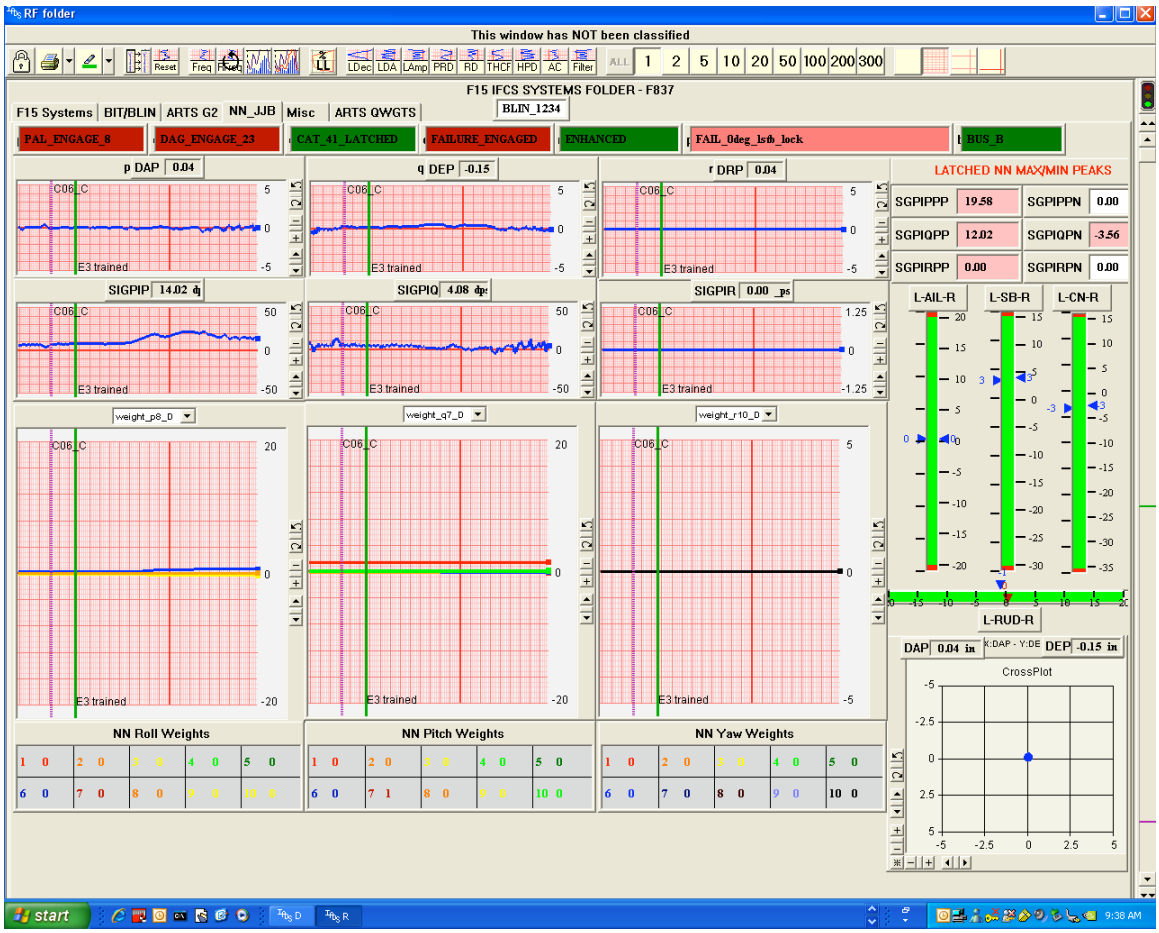


Figure 23. The interactive analysis and display system: neural network systems page 2.

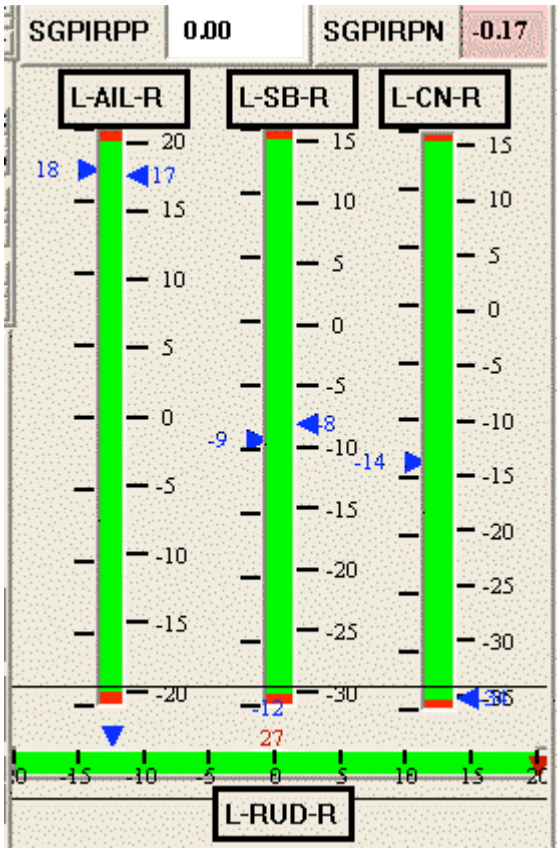


Figure 24. The slider bar display.

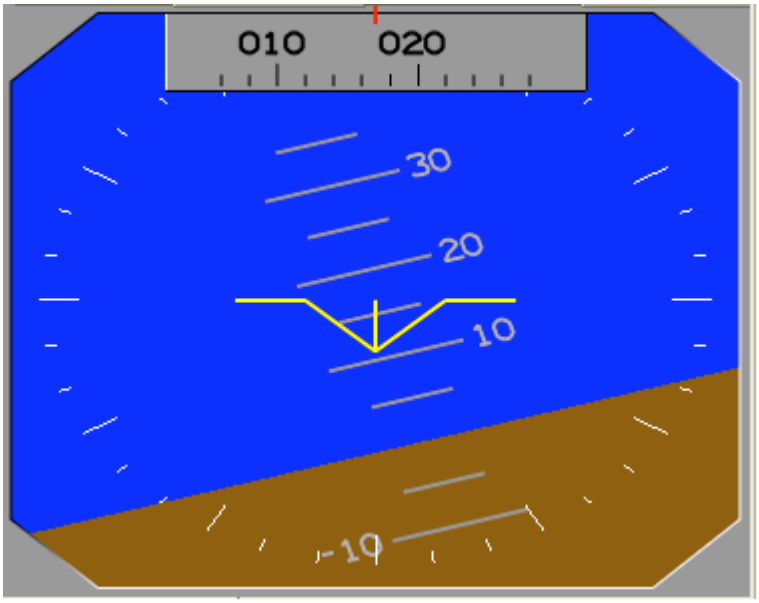


Figure 25. The attitude displacement indicator display.

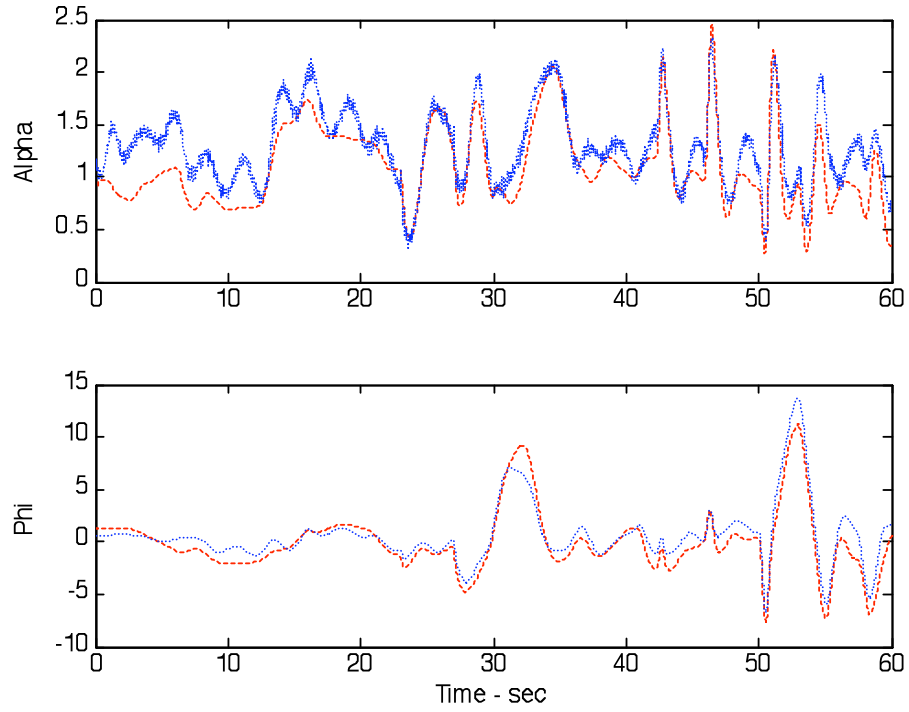


Figure 26(a). Flight (blue lines) to simulation (red lines) comparison of alpha and phi for left; the stabilator jammed at 7.5 s.

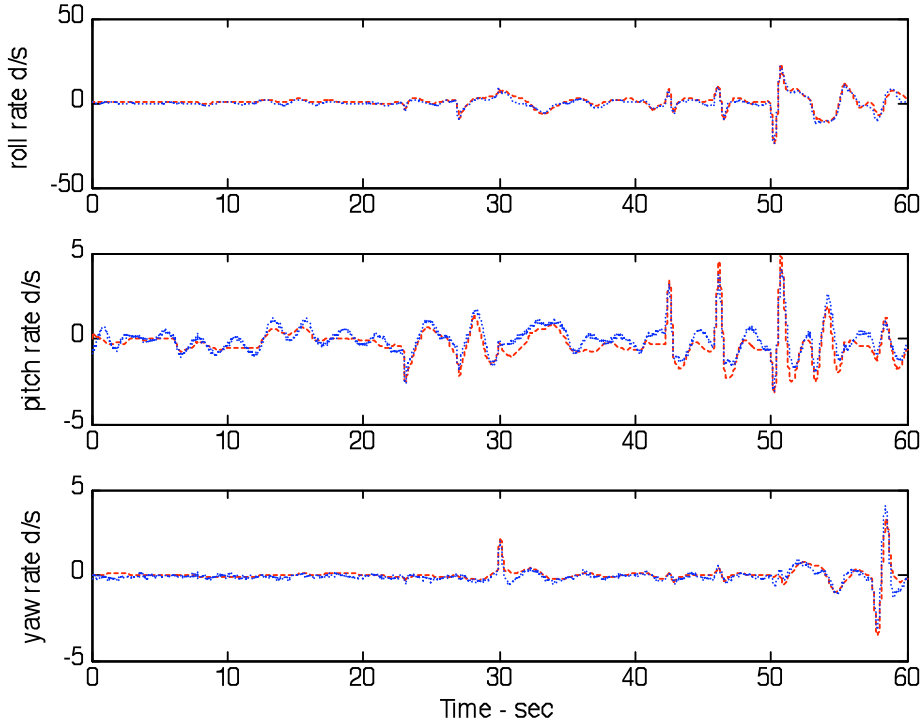


Figure 26(b). Flight (blue lines) to simulation (red lines) comparison of p, q, and r for left; the stabilator jammed at 7.5 s.

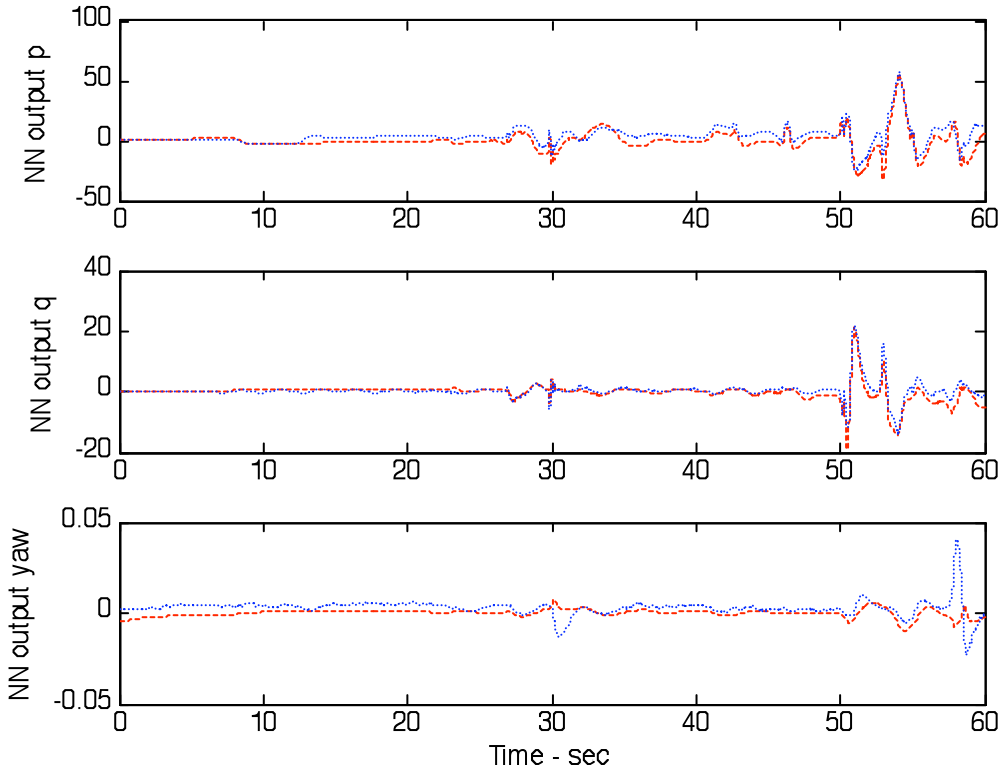


Figure 26(c). Flight (blue lines) to simulation (red lines) comparison of the neural network signals for left; the stabilator jammed at 7.5 s.

Figure 26. Comparisons of flight to simulation.