REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-02-

0103

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
03-04-2002	Paper/PDF files	12-01-1997 - 4-30-2001

4. TITLE AND SUBTITLE
YF22 MODEL WITH ON-BOARD ON-LINE LEARNING MICROPORCESSORS-BASED NEURAL ALGORITHMS FOR AUTOPILOT AND FAULT-TOLERANT FLIGHT CONTROL SYSTEMS

5a. CONTRACT NUMBER

5b. GRANT NUMBER
F49620-98-1-0136

5c. PROGRAM ELEMENT NUMBER

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

6. AUTHOR(S)

Marcello R. Napolitano, PI, Professor
Mechanical and Aerospace Engineering Department – PO Box 6106
West Virginia University, Morgantown, WV 26506/6106
Tel. (304) 2934111 Ext. 2346 – E-mail: *napolit@cemr.wvu.edu*

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

West Virginia University Research Corporation
886 Chestnut Hill Road
P.O. Box 6845
Morgantown, WV 26506
Tel. (304) 2933998

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dr. Belinda B. King, Program Manager, Dynamics and Control
AFOSR/NM
801 N. Randolph St., Room 732
Arlington, VA 22203
[illegible].(703)696-8409, Fax(703)696-8450

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT

12. DISTRIBUTION / AVAILABILITY STATEMENT

Unclassified - Unlimited

20020401 060

13. SUPPLEMENTARY NOTES
This document was prepared for the AFOSR, Dynamics and Control Program.

14. ABSTRACT
This project focused on investigating the potential of on-line learning "hardware-based" neural approximators and controllers to provide fault tolerance capabilities following sensor and actuator failures. Following a phase of simulation studies a set of selected architectures for neural estimators and neural controllers were flown on a semi-scale YF-22 aircraft model. The YF-22 model was designed, built, and flown at research facilities at West Virginia University. Additionally, a customized electronic payload featuring these fault tolerant schemes was designed, built, tested and interfaced with the YF-22 flight control system. A series of 33 flight tests were conducted with the aircraft; the flight data confirmed the potential of neural estimators and controllers for fault tolerance purposes. Another research objective was to start addressing system requirements leading to the problem of software validation and verification for this new class of algorithms for fault tolerant flight control systems.

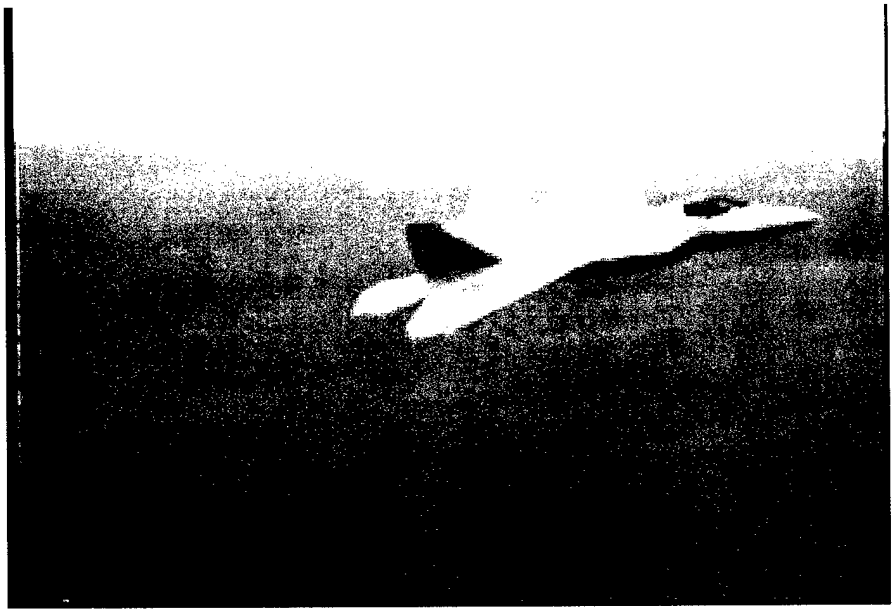
15. SUBJECT TERMS
Fault-Tolerant Flight Control System, Neural Networks, Sensor Failure, Actuator Failure, Flight Testing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Marcello R. Napolitano
Unclassified	Unclassified	Unclassified	none	154 + Appendix	19b. TELEPHONE NUMBER (include area code) (304)293-4111 Ext.2346

Final Report

YF22 MODEL WITH ON-BOARD ON-LINE LEARNING MICROPROCESSORS-BASED NEURAL ALGORITHMS FOR AUTOPILOT AND FAULT-TOLERANT FLIGHT CONTROL SYSTEMS

GRANT F49620-98-1-0136
Total Funding: $230,000



submitted by:

Marcello R. Napolitano, PI, Professor
Department of Mechanical and Aerospace Engineering
West Virginia University, Morgantown, WV 26506/6106
Tel. (304) 2934111 Ext. 2346, Fax (304) 2936689
E_mail: *napolit@cemr.wvu.edu*

submitted to:

Dr. Belinda B. King
Program Manager, Dynamics and Control
AFOSR/NM
801 N. Randolph St., Room 732
Arlington, VA 22203
Tel. 703-6968409, FAX 703-6968450
E-mail: *belinda.king@afosr.af.mil*

March 2002

ABSTRACT

This project focused on investigating the potential of on-line learning "hardware-based" neural approximators and controllers to provide fault tolerance capabilities following sensor and actuator failures.

Following a phase of simulation studies a set of selected architectures for neural estimators and neural controllers were flown on a semi-scale YF-22 aircraft model. The YF-22 model was designed, built, and flown at research facilities at West Virginia University. Additionally, a customized electronic payload featuring these fault tolerant schemes was also designed, built, tested and interfaced with the YF-22 flight control system. A series of 33 flight tests were conducted with the aircraft; the flight data confirmed the potential of neural estimators and controllers for fault tolerance purposes.

Another research objective was to start addressing system requirements leading to the problem of software validation and verification for this new class of algorithms for fault tolerant flight control systems.

Table of Content

data", Paper 99-4236, Proceedings of the '99 AIAA Guidance Navigation & Control Conference, Portland, OR, August 1999

A.6 - Del Gobbo, D., Napolitano, M.R.., Callahan, J., Cukic, B. "Experience in Developing System Requirements Specification for a Sensor Failure Detection and Identification Scheme", Proceedings of the 3rd IEEE High-Assurance Systems Engineering Symposium, Washington DC, November 1998

INTRODUCTION

This document presents a summary of the results of a research effort on the design, construction, and flight-testing of a semi-scale YF-22 flying research model. The model is equipped with a specific electronic payload to demonstrate the capabilities of on-line learning neural networks for fault tolerant flight control systems. An additional objective was to start to address software validation and verification issues for this specific class of control systems.

The report is divided in the following sections highlighting different results and accomplishments:

Section #1: Design and construction of the WVU YF-22 model.

Section #2: Design and construction of the electronic payload.

Section #3: Flight-testing activities.

Section #4: Results of a parameter identification study for the determination of the mathematical model of the WVU YF-22 aircraft followed by the development of Matlab/Simulink simulation codes featuring on-line learning neural networks for fault tolerant flight control systems (sensor and actuator failures).

Section #5: Formal specification of requirements for analytical redundancy-based fault tolerant flight control systems.

Section #6: Flight testing results of the application of the fault tolerant schemes.

Appendix A contains a copy of the technical publications from this effort.

Project Personnel

Several researchers were fully or partially supported throughout the duration of the project (3 ½ years). A list of the researchers involved in the project includes:

- Marcello Napolitano, Professor, PI;
- David Martinelli, Associate Professor, Co-PI;
- Brad Seanor, Aerospace Engineering Graduate Research Assistant, current PhD student;
- Diego Del Gobbo, Aerospace Engineering Graduate Research Assistant, December 2000 WVU graduate with a PhD in Aerospace Engineering (later partially involved as Research Assistant Professor);
- Gu Yu, Aerospace Engineering Graduate Research Assistant (PhD student)
- Srikanth Gururajan, Aerospace Engineering Graduate Research Assistant (PhD student);
- Peter Cooke, Aerospace Engineering Undergraduate Research Assistant, Pilot;
- Ben Reid, Aerospace Engineering Undergraduate Research Assistant.

A list of former WVU researchers (who graduated or left WVU before the end of the project) includes:

- Stelios Pispistos, Aerospace Engineering Graduate Research Assistant, August 1999 WVU graduate with a MS in Aerospace Engineering;
- Younghwan An, Aerospace Engineering Graduate Research Assistant, December 2000 WVU graduate with a PhD in Aerospace Engineering;
- Francesco Nasuti, Research Assistant Professor.

The project also involved the collaboration of:

- Giampiero Campa, Research Assistant Professor;
- Mario Luca Fravolini, Visiting Research Assistant Professor.

Craig Aviation was a Sub-Contractor in this project.

Section #1

Design and construction of the WVU YF-22 model

Section #1 - Table of Contents

1.1. – Design and construction of the WVU YF-22 Model

The design and the construction of the YF-22 model started in 1998 with the collaboration of several graduate students and research personnel at West Virginia University (WVU) along with Craig Aviation, an external sub-contractor.

The adopted philosophy for the design of the WVU YF-22 model was quite simple and straightforward. Unlike equivalent recreational models, research aircraft models cannot be exact scale models due to the need of carrying a research payload whose weight could be a substantial ratio of the take-off weight. Thus, the WVU YF-22 aircraft was designed around its payload. Therefore, as soon as a preliminary estimate of the size and weight of the electronic payload (described in Section #2) was available, two critical design parameters - the "Weight/Wing Surface" ratio (also known as the wing load W/S) and the "Thrust/Weight" ratio - were set to suitable ranges to achieve satisfactory handling qualities and dynamic characteristics. In turn these design parameters dictated the necessary installed thrust and the size of the wing surface, which, in turn, allowed the sizing and the design of the entire aircraft model.

Including the construction of the plug and mold of the model, the design and the construction of the WVU YF-22 model – excluding the electronic payload - required approximately 1,400 working hours by WVU graduate students and Craig Aviation over a 10-month period. The WVU YF-22 aircraft model is shown in Figures 1.1 and 1.2.

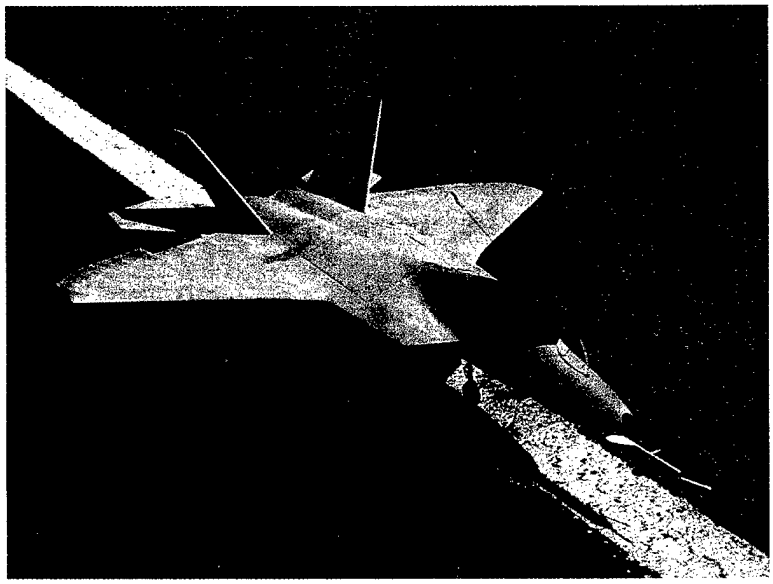


Figure 1.1 – The WVU-YF-22 Model

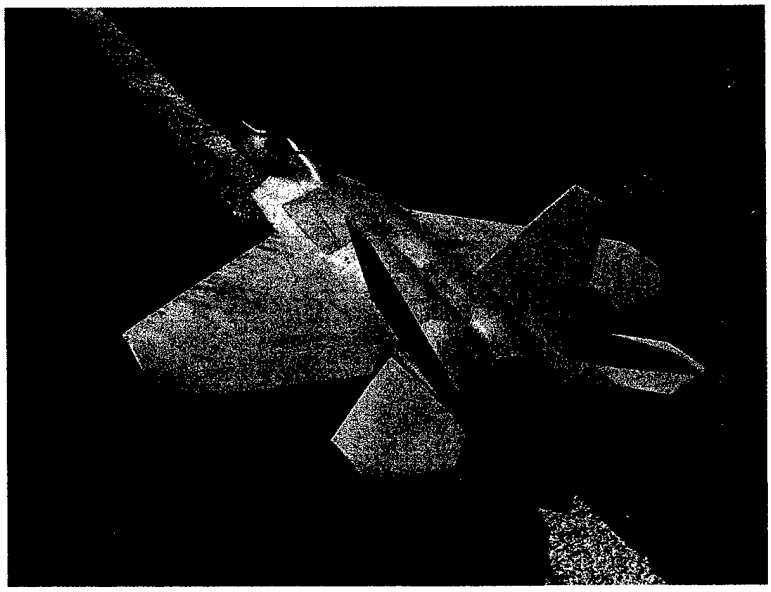


Figure 1.2 – The WVU-YF-22 Model

A summary of the main characteristics and geometric data of the WVU YF-22 model are shown in Table 1.1. Table 1.2 lists the raw materials used for the construction of the model while Table 1.3 lists more detailed information.

Scale	approx. $1/8^{th}$
Length	84 in
Wing span	67.5 in
Remote control (RC)	8 channel programmable radio Channel at nominal conditions: L/R Horizontal Stablilator, L/R ailerons, L/R rudders, L/R flaps, engine throttle, nose gear and retracts, brakes)

Table 1.1 –Geometric data and basic characteristics of the WVU YF-22 mdel

Fuselage	Fiberglass, carbon fiber, plywood reinforcement
Wing	Fiberglass, foam, balsa, plywood
Front access hatch	RC/computer interface, front gear, probe sensors
Middle access hatch	computer payload, aircraft sensors, main gear, fuel
Rear access hatch	propulsion system, control surface sensors, servos
Landing gear	retractable gear w/brake system
Propulsion system	28 lbs. thrust miniature turbine (RAM 1000)

Table 1.2 – Construction details of the WVU YF-22 Model

Wing surface	approx. 11.0 ft^2
Chord (wing root)	38.59 in
Chord (wing tip)	11.79 in
Wing tip ratio	0.3
Aspect ratio	2.87
Wing sweep angle	approx. 47°
Single stablilator surface	138.3 in^2
Single aileron surface	27.6 in^2
Single vertical tail surface	180.0 in^2
Single rudder surface	40.8 in^2
Thrust (cruise conditions)	approx. 25 lbs
Weight (configuration #1)	36.8 lbs Configuration #1: with electronic payload, without fuel
Weight (configuration #2)	43.3 lbs. (with 104 oz. ~ 6.5 lbs. fuel) Configuration #2; with electronic payload, with full fuel
Thrust/Weight (configuration #1)	approx. 0.68
Thrust/Weight (configuration #2)	approx. 0.58
Wing load (W/S) (configuration #1)	approx. 3.34 lbs/ ft^2
Wing load (W/S) (configuration #2)	approx. 3.94 lbs/ ft^2

Table 1.3 – Additional characteristics of the WVU YF-22 model

1.2. – Selection and interface of the propulsion System

R.A. Microjets, Inc. produces the RAM 1000 engine installed on the WVU YF-22 model. The maximum available thrust range from this engine is 28 lbs. The RAM 1000 is a single shaft turbojet with an annular combustor and with a fuel consumption of 12 oz./min at maximum throttle. A single axial flow turbine wheel drives a single stage centrifugal compressor. The shaft is supported by two lubricated, pre-loaded angular contact bearings. An electronic control unit (ECU) monitors exhaust gas temperature, engine compressor pressure, and controls the pump drive voltage. The miniature pump in turn controls turbine speed by varying its RPM and, conversely, the fuel supplied to the turbine.

Throughout a typical flight in Configuration #2 the throttle setting is at approximately "½ maximum thrust" with maximum throttle settings necessary only for take-offs. The mounting of the jet engine on the model is shown in Figure 1.3. A metal screen between the cargo bay and the engine bay was introduced to avoid the ingestion of small objects and/or pebbles by the engine (with potentially catastrophic consequences). The specifications of the RAM 1000 engine are summarized in Table 1.4.

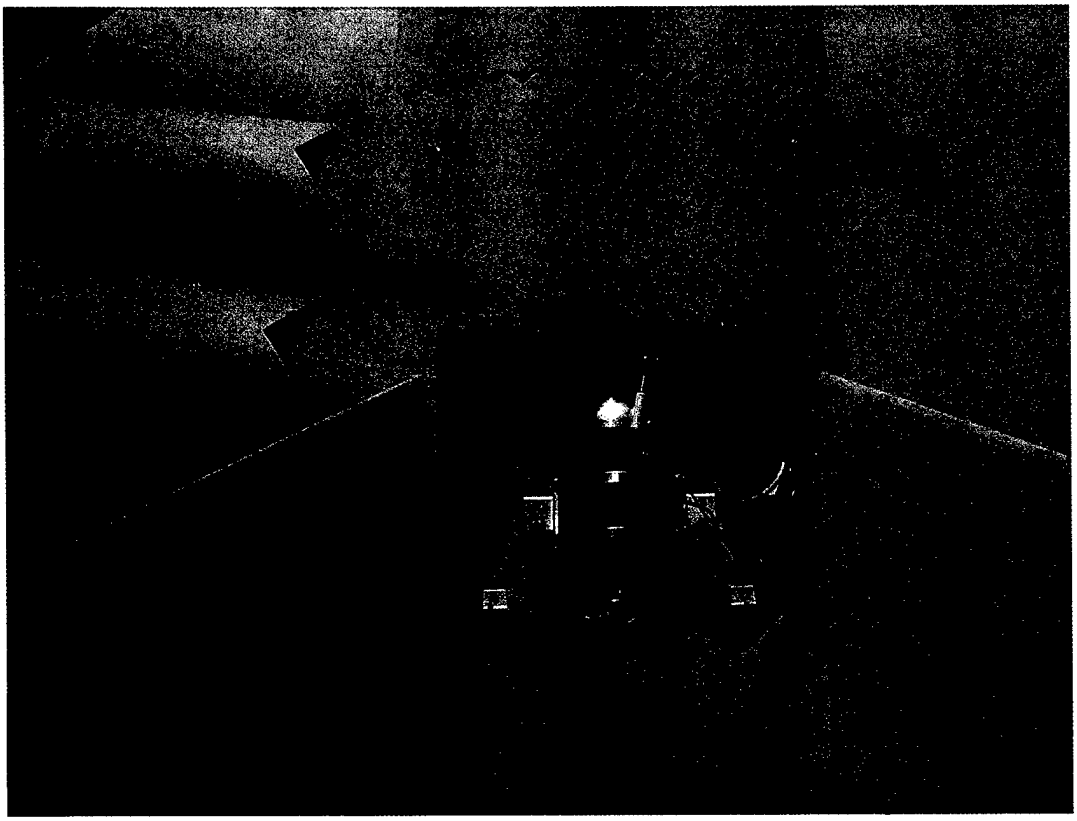


Figure 1.3 – Installation of the RAM 1000 engine on the YF-22 model

Maximum thrust	125 Newtons (28 lbs.)
Thrust range	5 N. (1.0 lbs.) - 125 N. (28 lbs.)
Fuel consumption	12 oz. / min. @ Max R.P.M
RPM range	36,000 @ Idle - 126,000 Max
Pressure ratio	3:1
Exhaust Gas Temperature (EGT)	650 Degree C. (Max)
Weight:	2.5 lbs
Max diameter	108 mm (4.25")
Total length	216 mm (9.5")

Table 1.4 – Specifications of the RAM 1000 turbine

Section #2

Design and construction of the electronic payload of the WVU YF-22 model

Section #2 - Table of Contents

List of Symbols (Section #2)

Greek Symbols

α	Longitudinal aerodynamic angle of attack (deg)
β	Lateral sideslip angle (deg)

Acronyms

AFDIA	Actuator Failure Detection, Identification, and Accommodation
CG	Center of Gravity
DAS	Data Acquisition Software
ECU	Engine Control Unit
EMI	Electro-Magnetic Interference
GCU	Ground Control Unit
IMU	Inertial Measurement Unit
OBC	On-Board Computer
PWM	Pulse Width Modulation
RC	Remote Control
SFDIA	Sensor Failure Detection, Identification, and Accommodation

2.1 – Design Approach for the Electronic Payload of the WVU YF-22 Model

The electronic payload of the WVU YF-22 was designed to perform in the following modes:

MODE #1 : manual flight with data acquisition.

The ground pilot has direct control of the aircraft. Flight data from a network of sensors are collected and stored in the on-board computer.

MODE #2 : automatic flight with actuator/sensor failure with manual control in the background

The design and the development of the electronic payload of the WVU YF-22 model required approximately 2,800 hours by two WVU graduate students and a research associate over a 16-month period.

The on-board computer (OBC) drives the control surfaces and "injects" actuator/sensor failures. The pilot can switch from manual flight to automatic flight and vice versa anytime. At any time in Mode #2 the pilot can regain direct control of the aircraft. The hardware/software requirements for the two modes are shown in Table 2.1.

Mode	Hardware Requirements	Software Requirements	Other Requirements
Manual flight with data acquisition	▪ RC electronics ▪ Instrumentation electronics ▪ Data recorder	▪ Data Acquisition Software	-
Automatic flight With sensor failure (for SFDIA) OR actuator failure (for AFDIA)	▪ RC electronics ▪ Instrumentation electronics ▪ Data recorder ▪ Control computer ▪ Switch module (manual/automatic) ▪ Servo driver module ▪ SFDIA hardware OR ▪ AFDIA hardware	▪ Data Acquisition Software ▪ SFDIA software OR ▪ AFDIA software	▪ Design of guidance/control laws ▪ Switch procedure (manual/automatic) ▪ SFDIA failure procedure OR AFDIA failure procedure

Table 2.1 – Hardware/software requirements for each of the modes

The main components of the WVU YF-22 electronic payload are:

- the remote control (RC) system,
- the on-board computer (OBC),
- the network of sensors,

- the control/switch board,
- the power components.

The development was divided into the following phases.

- Phase 1
  Design of the overall architecture.
- Phase 2
  Manual RC mode.
- Phase 3
  Manual RC mode with data acquisition.
- Phase 4
  Manual RC mode with data acquisition and "injections" of actuator and sensor failures.

2.2 – Description of the remote control (RC) components.

The components of the RC system for the WVU YF-22 model are:

- 1 receiver,
- 11 servos,
- 1 microprocessor controlling the engine.

A brief description on each of these components is provided below. The RC receiver has been mounted on the control/switch board (described in Section 2.5). The receiver has 8 channels that can be driven coupled or with different mixing strategies depending on the operating modes described above. On the receiver there are 8 connectors for servos, 1 extra connector (not used), and 1 connector for power supply (connected to a battery).

There are 11 servos and three additional servo signals for engine and computer use, as outlined below in Table 2.2. Note that multiple servo signals can be driven with the same channel.

Receiver channel	Numbers of servo signal	Cable	Servos
1	2	L+R	Ailerons
2	3	L+R	Elevators Brake
3	1	R	Throttle
4	3	L+R+N	Rudders Nose wheel
5	1	N	Retracts
6	2	L+R	Flaps
7	1	Computer	Computer Use
8	1		Not used
Total	**13**		

Table 2.2 – List of the receiver channels

Each servo connector (Z-connector), shown in Figure 2.1, has 3 pins with the following signals:

Pin	Signal
1	Command (PWM)
2	GND
3	V+

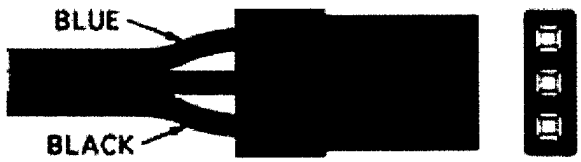


Figure 2.1 - Z-connector

Note that all the servos can share the same V+/GND signals. To minimize the interference between receiver and servos, two separate battery packs are used for the receiver and two battery packs are used for the servos. DB9 (serial) connectors (with 9 pins) have been used to transfer servo signals. There are two cables connected to each side of the plane. Each main cable is connected to the control/switch board (outlined in Section 2.5 below) with a DB9 connector and ends with 5 RC connectors connected to the relative rudder, elevator, aileron, and flap servos and the Engine Control Unit (ECU).

2.3 – Description of the on-board computer (OBC).

The on-board computer (OBC) is based on a PC/104 format to minimize weight and power consumption. The OBC performs the basic functions of acquiring and recording flight data as well as hosting the software featuring the control laws. The entire assembly is shown in Figure 2.2.

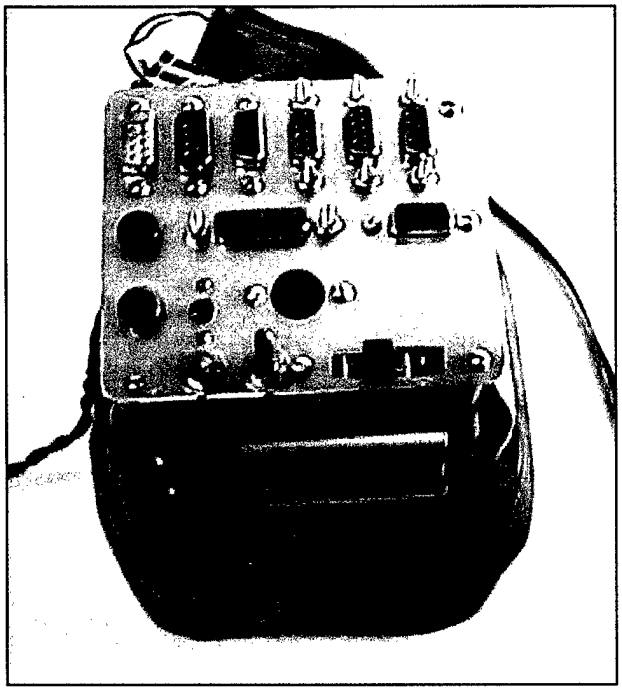


Figure 2.2 –OBC unit (upper view)

A more detailed side-view showing all the different components is provided in Figure 2.3.

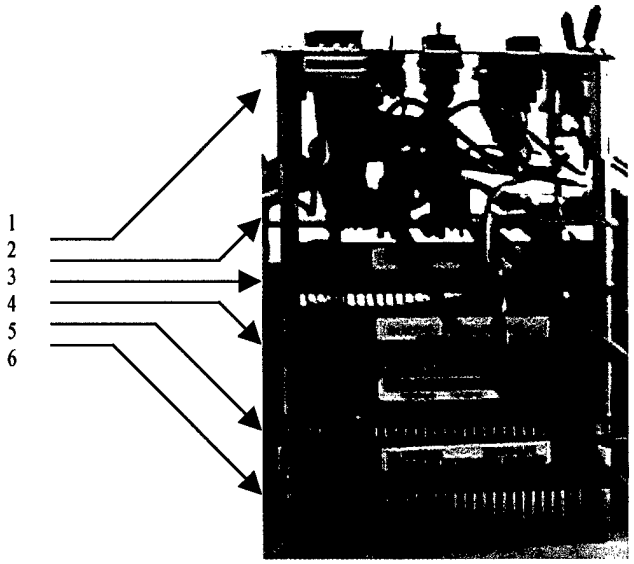


Figure 2.3 – OBC unit (side view)

with:
1: connection and control panel,
2: custom adapter board,
3: data acquisition module (Diamond-MM-32),
4: video module (Minimodule /SVG-II),
5: CPU module (Coremodule /P5i)
6: power supply module (Jupiter-MM).

The CPU module, which is the critical component of the OBC, is the AMPRO Coremodule /P5I shown in Figure 2.4.

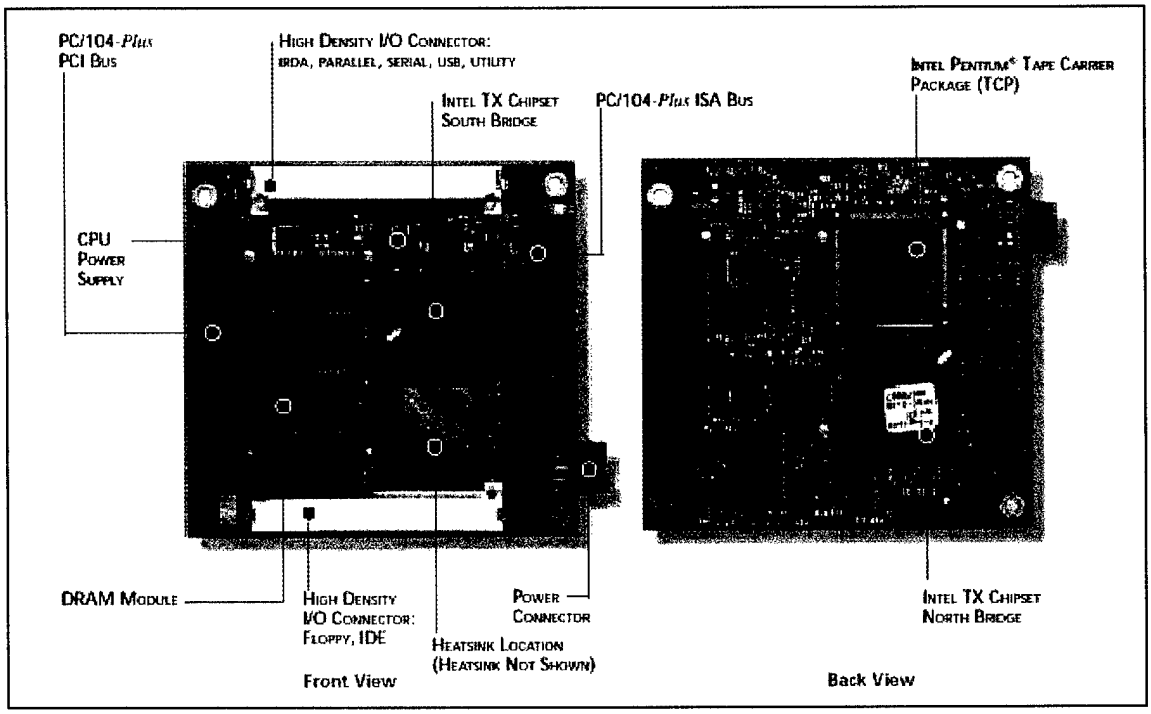


Figure 2.4 – AMPRO Coremodule /P5I CPU module

The specifications of the CPU card are provided in Table 2.3 below.

CPU	• Pentium CPU, 133 MHz VRT internal clock rate,
CHIPSET	• Intel 82439TX North Bridge/82371AB South Bridge
MEMORY	• 64 Mbytes
DMA INTERNAL	• 7 DMA channels (8237 equivalent), • 15 interrupt channels (8259 equivalent)
COUNTER TIMER	• 3 programmable counter/timers (8254 equivalent)
KEYBOARD	• PC/AT-compatible keyboard port
REAL TIME CLOCK	• Real time clock with CMOS RAM (MC146818 equivalent); requiring external 3.0-3.6V battery
BIOS	• Award ROM-BIOS with Ampro enhancements
SERIAL	• 2 RS232C serial ports with full handshaking, both ports implemented using 16C550 equivalent, with 16 byte data FIFOs, • Serial port 2 supports TTL mode
PARALLEL	• IEEE-1284 compatible enhanced parallel printer port with bi-directional data lines
FAST IDE	• Supports Ultra DMA/33 mode transfers for throughput up to 33 MB/sec
FLOPPY	• Supports 1 or 2 drives
PS/2 MOUSE PORT	• Usable with PS/2 mouse devices
USB	• Two USB ports
IRDA	• Infrared interface port • Normal mode supports up to 115.2K Baud • Fast IR mode supports up to 4M bits per sec
ON BOARD DISKONCHIP™	• 4 or 8 MB storage capacity • Real-time operating system support
CONFIG EEPROM	• Supports battery-free boot capability • 512 bits available for OEM use
WATCHDOG TIMER	• Utilizes real-time clock alarm function • Timeout triggers hardware reset or non-maskable interrupt
SIZE	• 3.6 x 3.8 x 0.9 in. (90 x 96 x 23mm)
BUS	• Compatible with 16-bit PC/104 ISA bus • Compatible with 32-bit PC/104-Plus PCI bus
ENVIRONMENTAL	• 0 to 70 deg. C standard temperature • Weight: 4.1 oz. (116 gm)

Table 2.3 – Specifications of the CPU card

The video card in the OBC, shown in Figure 2.5, is the AMPRO Minimodule/SVG-II.

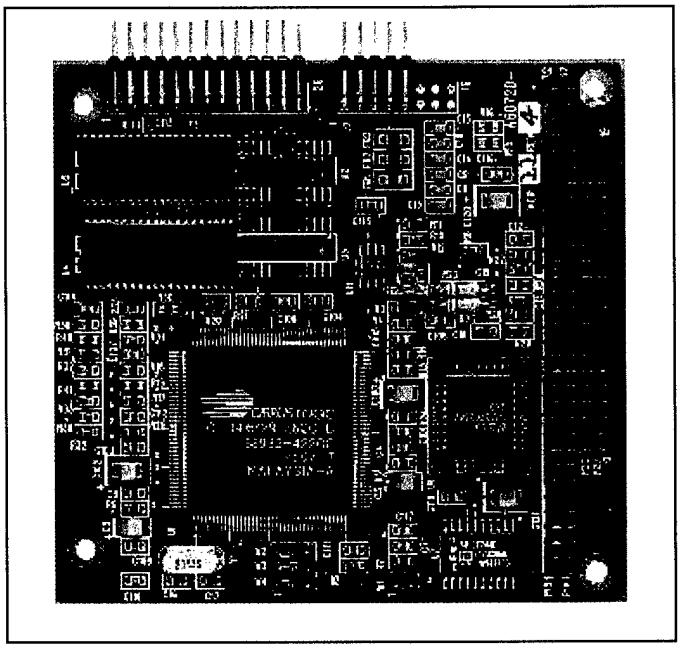


Figure 2.5 – AMPRO Minimodule/SVG-II video card

The DC power supply card, shown in Figure 2.6, is the Diamond Jupiter-MM card.

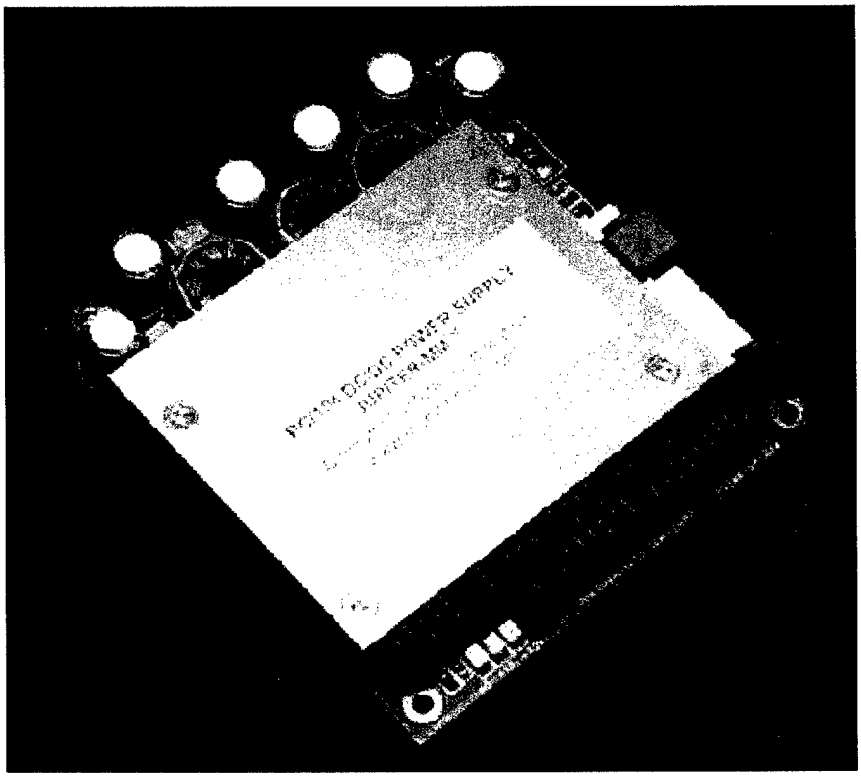


Figure 2.6 - Diamond Jupiter-MM power supply card.

The data acquisition card, shown in Figure 2.7, is the Diamond MM-32 card. The performance of the data acquisition card is particularly critical. In fact this card not only

collects signals from each sensors, but also send commands to drive servos through the digital output.

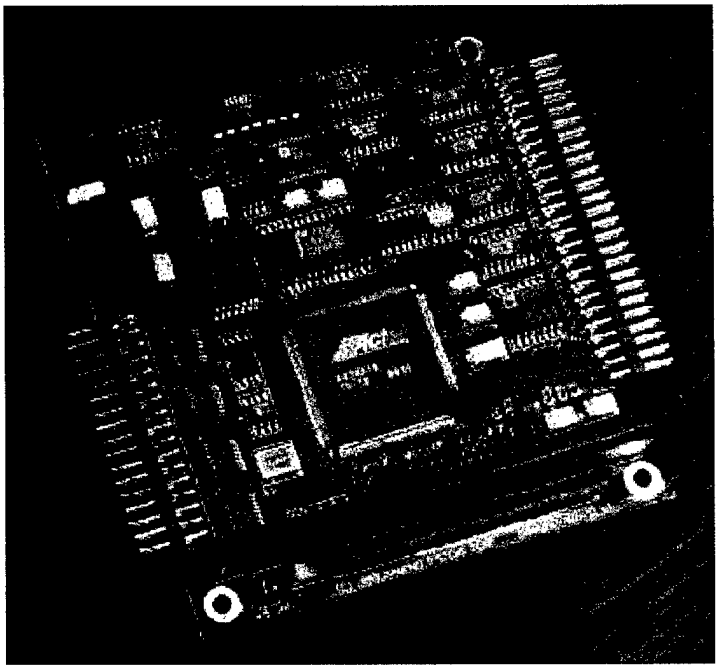


Figure 2.7 - Diamond MM-32 data acquisition card.

Through the data acquisition card the OVB processes 22 (out of 32) analog inputs, that is:

- α and β aerodynamic angles,
- static and dynamic pressure,
- left and right elevator deflections,
- deflections of the left and right rudders,
- deflections of the left and right ailerons,
- linear accelerations along the X, Y, and Z body axes,
- angular velocities around the X,Y, and Z body axes,
- the pitch and roll Euler angles,
- the control switch position (described below),
- the command switch position (described below),
- the battery's voltage.

Another OCB component is the adapter board, shown in Figure 2.8.

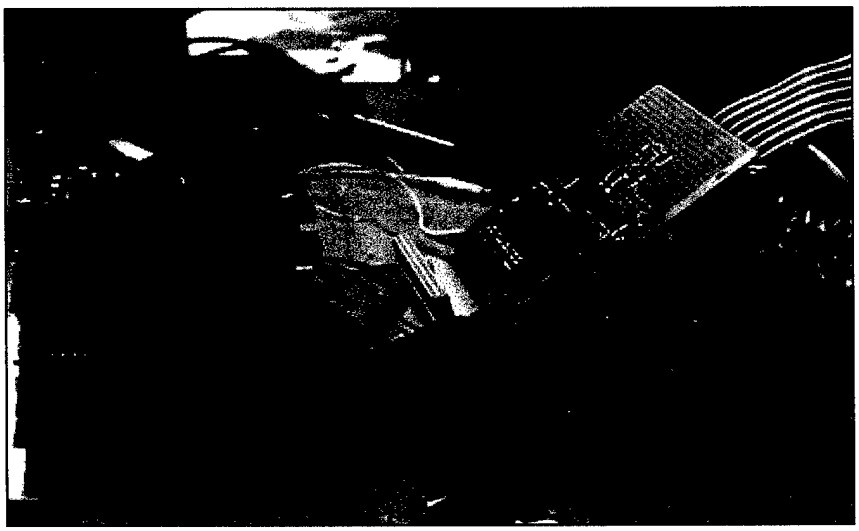


Figure 2.8 – Custom-built adapter card.

The task of the adapter card is to connect the serial port, the video port, and the keyboard signals to the panel for input/output and communication. The board also connects the battery and external power supply and provides different voltage levels for the panel. Furthermore, the board connects the sensor signals between the panel and the data acquisiton card. The adapter card was custom-designed and built at WVU; the design of the interface with the signals from the sensors is shown in Figure 2.9.

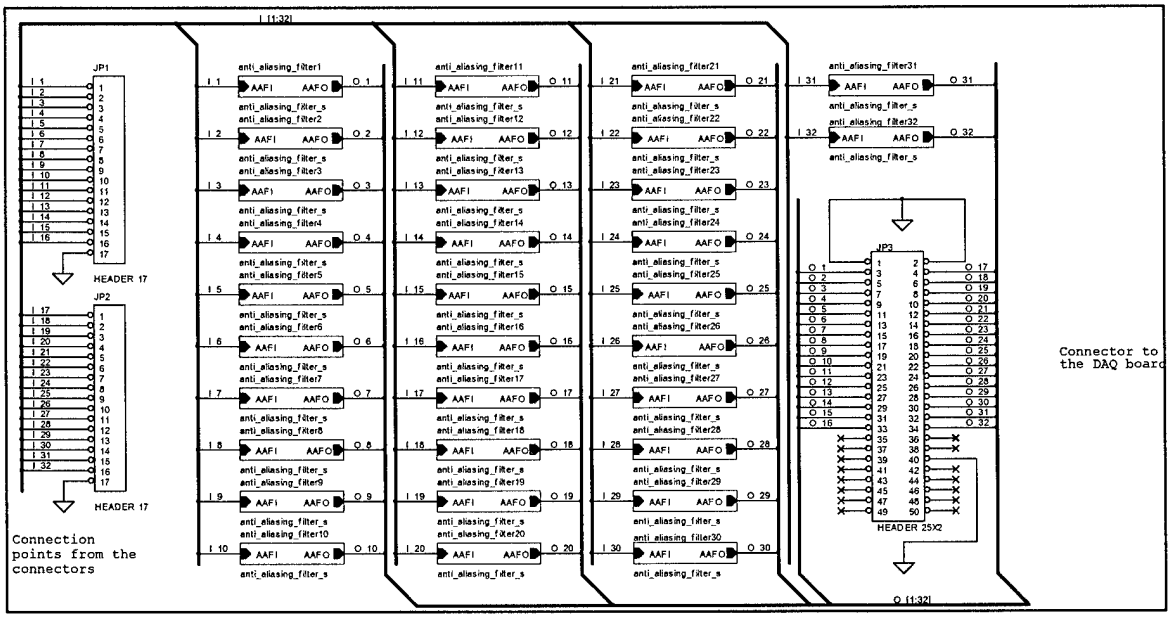


Figure 2.9 – Interface between the signals from the sensors and the adapter board

Finally, the last OCB component is the interface panel, shown in Figure 2.10.

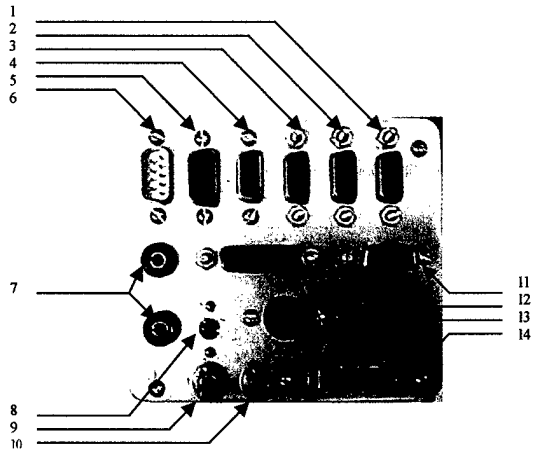


Figure 2.10 – Interface panel

where:

1. DB9 connector for the signals from the sensors on the aircraft nose,
2. DB9 connector for the signals from the potentioments on the control surfaces on the left side of the aircraft,
3. DB9 connector for the signals from the potentioments on the control surfaces on the right side of the aircraft,
4. DB9 connector for resetting the IMU,
5. DB9 connector for Serial port #1,
6. DB9 connector for Serial port #2,
7. 2 power connectors for the external power supply (for ground operations),
8. power switch for the RF modem,
9. pwer switch between on-board battery and the external power supply (for ground operations),
10. power switch for the OCB,
11. video connector (used for on-ground operations),
12. DB15 Connector for IMU,
13. PS2 Connector for keyboard (used for on-ground opeartions),
14. position switch between different operation modes.

Finally, there are two additional separate connectors:

- a power connector for connecting with the on-board battery,
- a 4-pin connector for the use of a mouse (for software developing and ground testing purposes).

The OCB unit is placed in a carbon fiber package for protection and shielding. The whole package is installed on two carbon fiber rails; this setup allows the OCB to slide on the rail so the aircraft can be balanced at the desired CG position. Given the limited space of the cargo bay of the WVU YF-22 model, special attention was given to the electromagnetic interference (EMI) problem for the OCB unit. In fact, the EMI issue could have been potentially critical for the accuracy of the flight data as well as the safety of the flight-testing operations. Despite the mounting of the OCB inside a carbon fiber package, some EMI problems became evident during an initial ground-testing phase. In particular, it was noted that the EM field from the OCB decreased the range of the RC system. Using a spectrum analyzer a wide range of EM was discovered from the OCB.

The peak frequency of this EM was found to be at 66MHz, which was almost coincident with the CPU on the motherboard with additional smaller peaks around 72Mhz, which is the frequency of the RC system. This second peak was due to the fact that several cables were working as antenna, especially the main power cable.

To overcome these EMI issues, the following countermeasures were introduced:
1- Addition of a thin steel plate at the bottom the OCB's box to enhance the shielding of the OCB.
2 – Introduction of chokes on selected cables of the power line and sensors lines. The specific cables were identified based on the EMI induced using the spectrum analyzer.
3 – Shift of the RC receiver toward the nose of the aircraft, allowing an increase of the distance from the OCB to the receiver antenna.

The combination of these countermeasures allowed decreasing the noise level by approximately 6 db. Furthermore, to increase the safety of the flight-testing operations, a 50 Mhz RC transmitter/receiver system was selected in lieu of a 72 Mhz system.

2.4 – Description of the sensors.

The WVU YF-22 electronic payload features a network of sensors allowing measuring all the most important dynamic parameters; furthermore, the angular deflections of each of the control surfaces are recorded. The list of on-board sensors includes:

- A nose probe with a Pitot tube and vanes with potentiometers for measuring the aerodynamic angles (α & β),
- An Inertial Measurement Unit (IMU) measuring linear accelerations, angular velocities, along with the pitch and roll Euler angles,
- A set of potentiometers for measuring surface deflections for left/right elevators, left/right ailerons, and left/right rudders.

A brief description of the characteristics of these sensors system is provided below. The nose probe was designed and custom-built at WVU. Commercially available products with excellent performance were considered but found to be too expensive for the given budget. The nose probe on the WVU YF-22 model is shown in Figure 2.11.

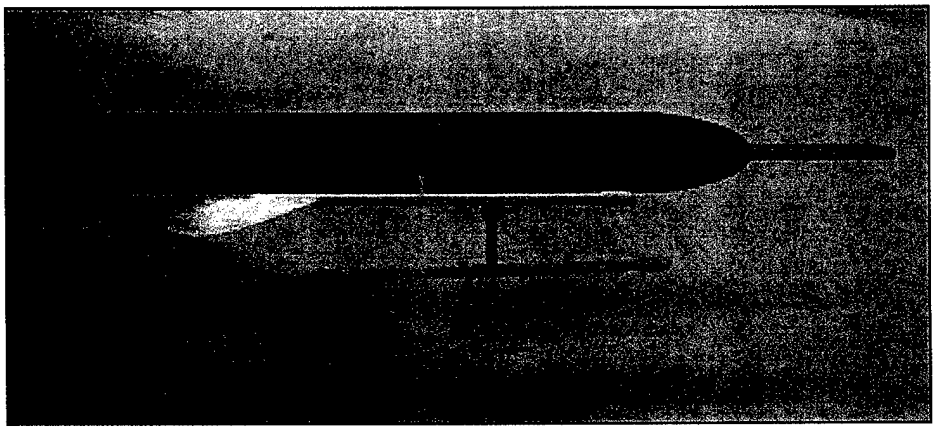


Figure 2.11 – Nose probe for the WVU YF-22 model

The α & β vanes were manufactured with a brass tube with the wings made of aluminum. The vanes were connected with two customized 10k potentiometers. The characteristics of the potentiometers were selected so that they are functional after the airspeed reaches approximately 35 mph and works smoothly with the airspeed above 40 mph. A specific procedure was designed for ground calibration of these potentiometers prior to the airctraft take-off. The pitot tube was manufactured with two layers of brass tubes. The dynamic pressure is measured directly from the front of the inner tube while the static pressure is measured through 4 small holes on the side of the outer tube. Of particular interest is the differential pressure sensor measuring the difference between the static pressure and the dynamic pressure. Using these pressure measurements, the measurements of airspeed and altitude were deduced using standard atmospheric air relationships. Two SenSym pressure sensors were used for this task; one of the sensors is shown in Figure 2.12.

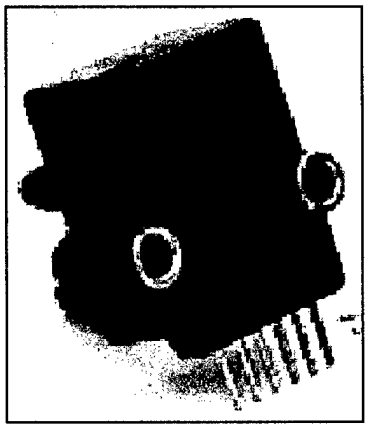


Figure 2.12 – Pressure sensor for the nose probe

A SenSym ASCX15AN sensor was used for the static pressure measurement – allowing the altitude measurement – with a range between [0-15] psi. A SenSym ASCX01DN sensor was used for the differential pressure – allowing the airspeed measurement – with a range between [0-1] psi.

The IMU, shown in Figure 2.13, is the Crossbow DMU-VGX unit.

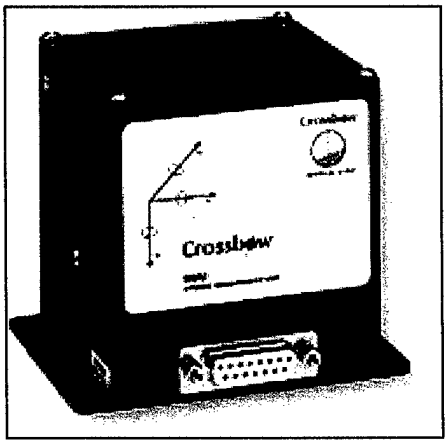


Figure 2.13 – DMU-VGX IMU unit

The DMU-VGX unit is a solid-state, 6-axis inertial measurement unit measuring linear accelerations, angular velocities, pitch and roll Euler's angles. Although fairly expansive this unit was preferred over other products because of its performance/price ratio. The DMU-VGX provides stabilized roll and pitch angles through signal processing of the rate and acceleration sensors. Conventional tilt sensors use earth gravitational field to measure angles. These sensors only measure tilt accurately when the object whose motion is being measured is not accelerating. In dynamic environments since gravity-based tilt sensors cannot distinguish between tilt and acceleration. The DMU-VGX uses a combination of angular rate and acceleration signals to solve this problem. Tilt is calculated by integrating the angular rate sensor outputs to angle. Then the tilt response of the accelerometers corrects for error due to angular rate sensor drift. Drift corrections are applied continuously and the frequency of the updates can be set via the serial command 'T' <0-255>. Roll and pitch angle, angular rate, and acceleration outputs are available in the DMU-VGX packet. This data can be requested via a serial command or set for continuous transfer. A list of the DMU-VGX specifications is provided in Table 2.4 below.

Performance	• Roll, Pitch Angle: Dynamic Accuracy	1° RMS	Application Dependent
	• Resolution	0.1°	Typical
	• Linearity	< 1 % of FS	FS Roll 180°, FS Pitch = 90°
	• Full Scale Span (analog outputs)	± 4.096 VDC	FS Roll 90°, FS Pitch = 90°
Power	• Input Supply Voltage	8 - 30 VDC	
	• Input Supply Current	100 mA (max)	
Environmental	• Operating Temperature Range	-40 to 85°	
	• Storage Temperature Range	-55 to 85° C	
	• Package	Aluminum housing	
	• Weight	475 grams	
	• Mechanical Shock	1000 G	(1 ms half sine wave)
	• Vibration	10 G RMS	
Data Output Rate	• Digital Voltage Mode	166 Hz	
	• Digital Scaled Sensor Mode	156 Hz	
	• Digital Angle Mode	100 Hz	
	• Analog Data Update Rate	400 Hz	

Table 2.4 – Specifications of the DMU-VGX IMU unit

The final components of the network of sensors are the potentiometers used for the measurements of the deflections of the control surfaces. These potentiometers were fairly expansive since their size had to be compatible with the small size of the metal hinges of the different control surfaces of the WVU YF-22 model. For an acceptable trade off between high signal/noise ratio and lower power consumption 10k potentiometers were selected.

Following a description of the on-board sensors an overview of the sensor wiring is provided below. There are four major cables connected between sensors and the OBC:

- nose cable,
- left main cable,
- right main cable,
- IMU cable.

The nose cable connects the nose probe to the OBC. It features 6 wires measuring:

1. Ground,
2. +5 Volt,

3. voltage signal from the α vane,
4. voltage signal from the β vane,
5. voltage signal from the dynamic pressure sensor,
6. voltage signal from the static pressure sensor.

The IMU Cable connects the DMU-VGX IMU unit to the OCB. It features 15 wires measuring:

1. RS-232 Transmit Data
2. RS-232 Receive Data
3. Vcc
4. Ground,
5. X-axis acceleration (analog voltage),
6. Y-axis acceleration (analog voltage),
7. Z-axis acceleration (analog voltage),
8. Roll rate (analog voltage),
9. Pitch rate (analog voltage),
10. Yaw rate (analog voltage),
11. Timing pulse,
12. Roll Euler angle (analog voltage),
13. Pitch Euler angle (analog voltage),
14. Unused,
15. Unused.

Each of the two left and right main cables connects the potentiometers from each side to the OCB. Both cable feature 7 wires measuring:

1. ground,
2. ground,
3. +12 Volt,
4. Voltage signal from the potentiometer embedded in the elevator,
5. Voltage signal from the potentiometer embedded in the rudder,
6. Voltage signal from the potentiometer embedded in the aileron,
7. Voltage signal from the potentiometer embedded in the flap.

The 4 cables are shielded and have a DB9 or DB15 connector at one end. In both main cables, the wire for the +12 Volt and an extra ground wire went from the outside of the shielded cable to minimize the interference caused by the current change.

2.5 – Description of the control/switch board.

The control/switch box, shown in Figure 2.14, is a critical component of the WVU YF-22 payload.

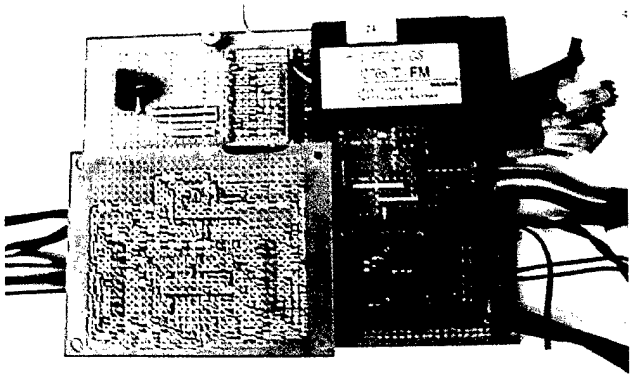


Figure 2.14 – Custom-designed control/switch board

The functions of the control/switch board are:

- to receive control signals from the pilot on the ground,
- to receive control signals from the OBC,
- to transfer the commands from the pilot into PWM signals,
- to transfer the commands from the OBC into PWM signals,
- to select the current operation mode of the flight control system (manual or automatic)
- to determine the channel/signal mapping for each flight control mode,
- to distribute control signals to each of the servos.

As discussed above, the control/switch box is designed so that the aircraft can be controlled in manual mode and automatic mode. In manual mode the receiver receives the control signal from the pilot on the ground and controls the aerodynamic surfaces and thrust mechanisms on the aircraft. In automatic mode, the OBC sends the commands to the control/switch box through the serial port. According to the specific mode (with or without failure) the control/switch board can either control all the surfaces or a subset of the surfaces. The pilot can use Channel 7 of the transmitter to enter the automatic mode and can decide to reverse to manual RC mode at any time. Given the unique characteristics of the WVU YF-22 flight control system, there were not commercially available boards. Thus, this item had to be designed and built by WVU researchers. The components of the control/switch board are:

- the receiver,
- the mother board,
- the command module,
- the mixer,
- the PWM to TTL board,
- the switch board.

The block diagram of the control/switch board is shown in Figure 2.15.

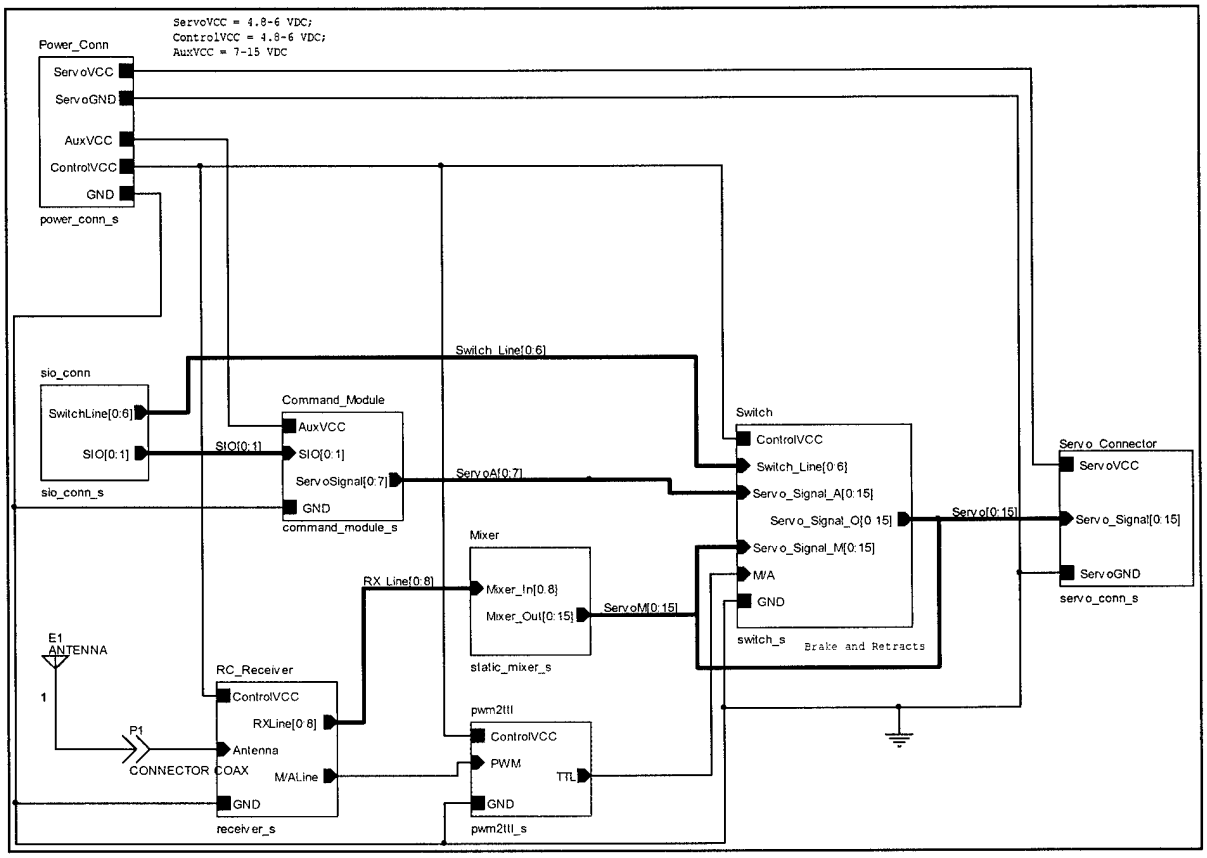


Figure 2.15 – Block diagram of the control/switch board

2.6 – Description of the on-board power system.

Prior to the design of the electronic payload a detailed power budget was formulated. An estimate of the power consumption of each of the components is shown below in Tabl 2.5 while Figure 2.16 shows a simple block diagram of the power distribution.

Components	Voltage(V)	Current(A)	Imax(A)	Power(W)
CPU card	5		2.370	11.85
Video card	5	0.35		1.75
DAQ Card	5	0.35		1.75
DMU VGX	12		0.1	1.2
Potentiometers	12		0.012	0.144
Total				**16.69**

Table 2.5 – Estimate of the power consumption for the WVU YF-22 payload

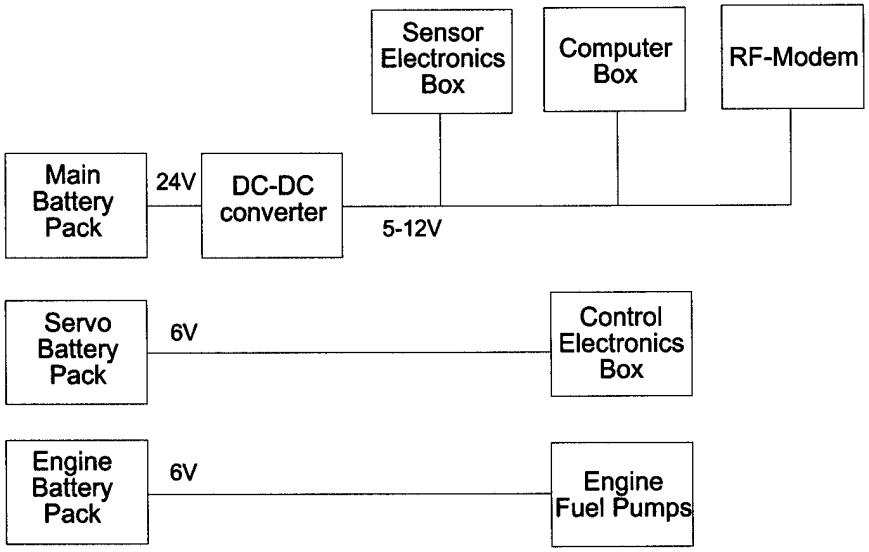


Figure 2.16 – Block diagram of the power scheme

2.7 – Interfacing of the components on the WVU YF-22 model.

The interfacing of the components of the electronic payload on the WVU YF-22 was particularly challenging due to the following reasons:

- limited volume in the cargo bay;
- need to minimize EMI issues (as described above);
- need to accommodate a specific range for the CG of the model for handlinq qualities purposes.

The total weight of the components is summarized below in Table 2.6.

Item	Weight(lbs)
OBC	2 lbs.
OBC Battery Pack	1.9 lbs.
OBC Enclosure	0.2 lbs.
IMU Unit	1.1 lbs.
Nose Extension	0.6 lbs.
Cabling & Wiring	0.5 lbs.
Rail System	0.5 lbs.
Control/switch board	0.4 lbs.
Potentiometers	0.2 lbs.
Miscellanea	0.5 lbs.
Total	**7.9 lbs.**

Table 2.6 – Estimate of the weight for the WVU YF-22 payload

The most appropriate configuration to guarantee desirable handling qualities (see Section #3) was found to be a configuration with all the heaviest components (that is the OBC + IMU) located approximately at the CG. Thus, these two items were mounted on the sliding rail to allow flexibility in the balancing of the aircraft. These components are shown in Figure 2.17.

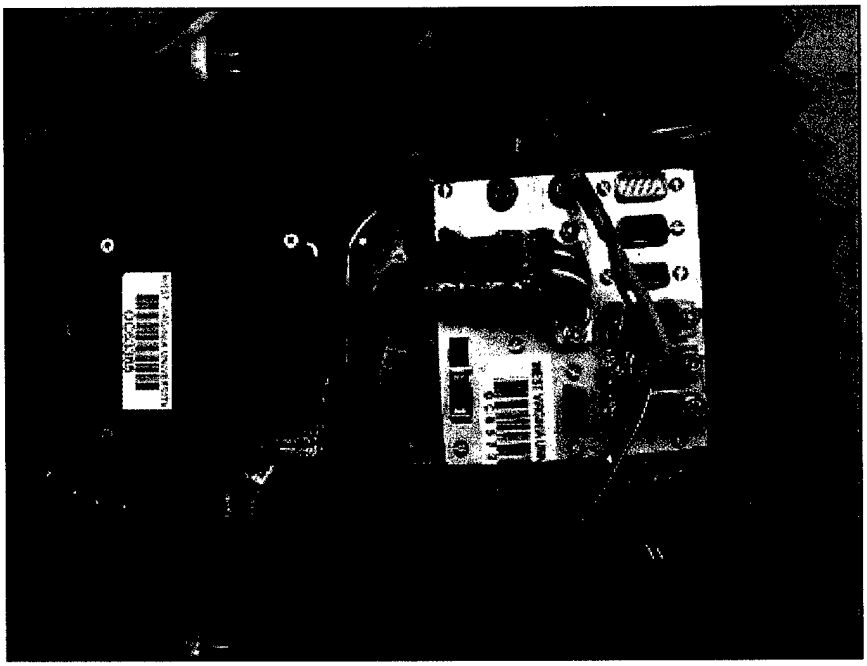


Figure 2.17 – Top view of the main cargo bay

The control/switch board, shown in Figure 2.18, was instead installed on a small platform under the canopy, immediately in front of the OBC box in the main cargo bay.

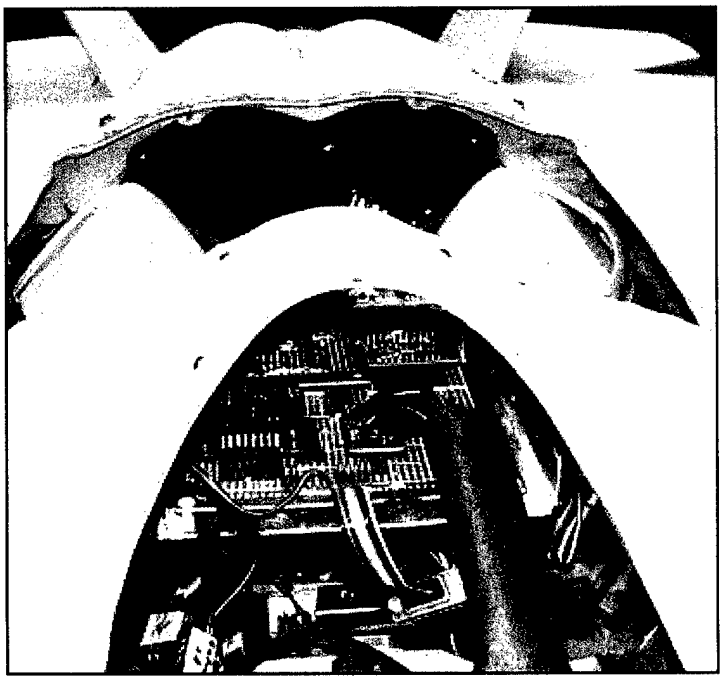


Figure 2.18 – View of the control/switch board

The potentiometers are installed at the hinge axis of each control surfaces. They were practically bonded to the structure during the construction of the model to avoid

slippage problems. Prior to each flight they are calibrated through a customized calibration procedure so that accurate measurements of the angular deflections are provided to the OBC. The remaing components of the electronic payload are installed in the nose section of the aircraft model accessable through the removable canopy. These components are shown in Figure 2.19.

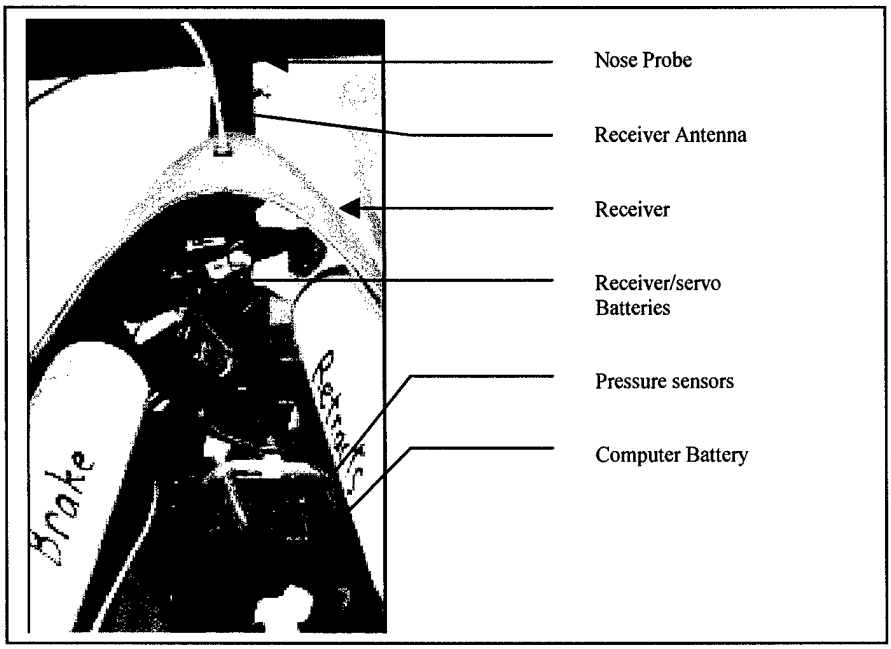


Figure 2.19 – View of the payload components in the nose section

Particularly, the nose section houses the following components:

- the nose probe, which is directly attached to the tip of the model;
- the RC receiver (installed on the nose to avoid/minimize EMI issues);
- the RC antenna (mounted on the top of the RC receiver);
- the RC receiver/servo batteries (four identical batteries: two for the receiver and two for servos);
- the pressure sensors (for the measurements of static and dynamic pressure);
- the battery for the OCB (2000mAhr model);
- the pressurized air tanks for the pneumatics of the front wheel brake and retractable landing gears.

2.8 – Description of the Data Acquisition Software

The software for the OBC of the WVU YF-22 model was written as a SIMULINK S-function compiled to be an executable file using the Real-Time-Workshop toolbox in Matlab. The SIMULINK environment was found to be ideal for the development of the flight control software in separate modules. The design of the on-board software was based on the following tasks:

Task #1 - Data acquisition at nominal flight conditions;
Task #2 - Actuator Failure Detection, Identification, and Accommodation (AFDIA);
Task #3 - Sensor Failure Detection, Identification, and Accommodation (SFDIA).

The software for the SFDIA and AFDIA schemes will be described in the Section #4 and Section #6 of this document. The main function of the Data Acquisition Software (DAS) is to read, convert, send, and store sensor readings for the electronic payload. The development of this software was tailored on the performance of the Diamond-MM-32 PC/104 Format 16-bit Analog I/O Module. Below the main characteristics of the DAS are described:

Analog Input Channels

The Diamond-MM-32 has the capacity of 32 analog I/O channels. 22 analog I/O channels – listed below in Table 2.7 - were actually used within this effort.

Channel Number	Channel Name	Notes
1	Alpha	
2	Beta	
3	Static Pressure	
4	Dynamic Pressure	
5	Left Elevator	
6	Left Rudder	
7	Left Aileron	
8	Control Switch	Manual/Automatic control
9	Right Elevator	
10	Right Rudder	
11	Right Aileron	
12	Reserved	
13	Acceleration -X	
14	Acceleration -Y	
15	Acceleration -Z	
16	P	
17	Q	
18	R	
19	Pitch angle	
20	Roll angle	
21	Command switch	Data Acquisition/Download
22	Battery Voltage	

Table 2.7 – List of the I/O analog channels on the Diamond-MM-32 card

Input Ranges and Resolution

All the sensors were powered with +12V or +5V (nose probe). Therefore, all the analog I/O channels were configured to accept 0-10V unipolar inputs with resolution of 153 μV.

A/D Conversion Formulas

The 16-bit value returned by the A/D converter is always an integer complement number ranging from –32768 to 32767 regardless of the original input range. The range of the input to the A/D is fixed at ±10V. Actually, the input signal from the different components is actually magnified and shifted to match this ±10V range before it reaches the A/D. For example, for an input range of 0-10V the signal is first shifted down by '*5V*' to ±5V and then amplified by '*2*' to be in the ±10V range.

A/D Conversion

There are seven steps involved in performing an A/D conversion:

Step #1 - Select the input channel or input channel range

The Diamond-MM-32 contains a channel counter circuit that controls which channel is sampled on each A/D conversion command. The circuit uses two channel numbers called the '*low*' and the '*high*' channel respectively. These are stored in registers at Base +2 and Base +3 (that is, base address of the Diamond-MM-32 card). The circuit starts at the '*low*' channel and automatically increments after each A/D conversion until the '*high*' channel is reached. When an A/D conversion is performed on the '*high*' channel, the circuit resets to the '*low*' channel and starts over again. To read from a series of consecutively numbered channels, the user writes the starting channel to Base+2 and the ending channel to Base+3. In the DAS, the '*low*' channel is set to '0' and the '*high*' channel is set to '21'. The data acquisition card scans the whole range once at each sampling time.

Step #2 - Select the analog input range (Range, Polarity, and Gain codes)

The desired input range can be selected by writing to the analog I/O control register at Base+11. The user only needs to write to this register if he/she wishes to select a different input range from the one used for the previous conversion. As stated above, the analog input range is 0-10V.

Step #3 - Wait for analog input circuit to settle

After changing either the input channel or the input range, the user must allow the circuit to settle on the new value before performing an A/D conversion. The settling time is fairly long compared to software execution times; therefore, a timer is provided on board to indicate when it is safe to precede with A/D sampling. The WAIT bit at Base+11 indicates when the circuit is settling and when it is safe to sample the input.

Step #4 - Start an A/D conversion on the current channel

To generate an A/D conversion, the software simply writes to Base+0 to start the conversion.

Step #5 - Wait for the conversion to be concluded

The A/D converter takes about 4 microseconds to complete a conversion. If a reading of the A/D converter data is attempted immediately after starting a conversion, an invalid data is obtained. Thus, the A/D converter provides a status signal to indicate whether it is busy or idle. This signal can be read back as the STS bit in the status register at Base+8. When the A/D converter is busy (performing an A/D conversion), this bit is '1'; else, when the A/D converter is idle (conversion is done and data is available), this bit is '0'.

Step #6 - Read the A/D data

Once the conversion is complete, the data can be read back from the A/D converter. The data is 16 bits wide and is read back in two 8-bit bytes at Base+0 and Base+1. The following pseudo-code illustrates how to construct the 16-bit A/D value from these two bytes:

```
LSB = read (base);        : 'Get low 8 bits'
MSB = read (base+1);      : 'Get high 8 bits'
Data = MSB*256+LSB ;      : 'Combine the 2 bytes into a 16-bit value'
```

The final data ranges from 0 to 65535 (0 to 2^{16}- 1) as an integer (without the sign). This value must be interpreted as an integer ranging from -32768 to +32767.

Step #7 - Convert the numerical data to a meaningful value

Once the A/D value is available, it needs to be converted to a meaningful value using the formula shown above.

Sampling Rate

The sampling rate of the DAS has been designed to be adjustable. For nominal flight conditions in "pure" data acquisition mode (that is, without the occurrence of a sensor or an actuator failure and without on-line learning for the neural networks – as described in the next sections) the sampling rate is set to 100Hz. For SFDIA and AFDIA flight testing (that is, with on-line learning for the neural networks as described in the next sections) the sampling rate is set to 50Hz.

The block diagram of the WVU YF-22 DAS software is shown in Figure 2.20.

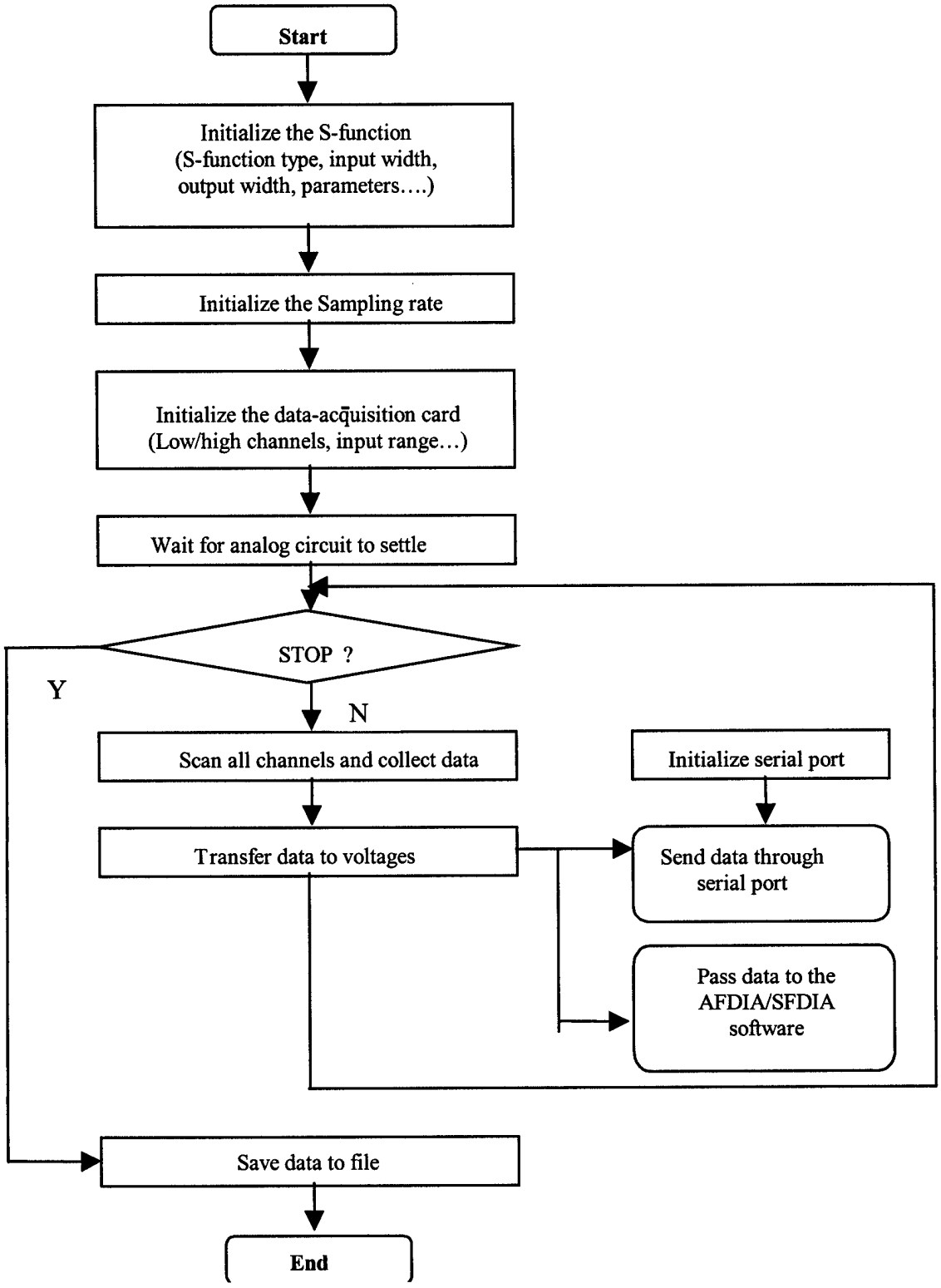


Figure 2.20 – General Diagram of the Data Acquisition Software

Section #3

Flight testing activities.

Section #3 - Table of Contents

List of Symbols (Section #3)

English

a	Acceleration (ft/sec^2)
k	Discrete time index
p	Aircraft angular velocity around the x body axis (roll rate), rad/sec
q	Aircraft angular velocity around the y body axis (pitch rate), rad/sec
r	Aircraft angular velocity around the z body axis (yaw rate), rad/sec
t	Time, sec

Greek

α	Angle of attack, rad or deg
β	Angle of sideslip, rad or deg
θ	Pitch Euler angle, rad or deg
δ	Control surface deflection, rad or deg
ϕ	Roll Euler angle, rad or deg
ψ	Yaw Euler angle, rad or deg

Subscripts

A	Aileron
E	Elevator
L	Left side
R	Right side
R	Rudder

Acronyms

BLS	Batch Least Square
CG	Center of Gravity
FTR	Fourier Transform Regression
GCU	Ground Control Unit
LS	Least Square
LWR	Locally Weighted Regression
MLP	Multi-Layer Perceptron
NN	Neural Networks
OBC	On-Board Computer
PID	Parameter Identification
RC	Remote Control
RLS	Recursive Least Square

3.1. – WVU Flight Testing Facility

The research team at WVU has access to a dedicated facility for the flight-testing activities. The facility is located approx. 70 miles south with respect to the WVU main campus in Morgantown, WV, and it is part of the WVU campus at Jackson Mill, WV. The facility is named "Louis Bennett Field" and it features a semi-paved 3,400 ft, approx. 50 ft wide runway.

Given the experimental nature of the research the remoteness of the facility is ideal for the purpose. The only drawbacks are related to the general weather conditions in West Virginia; on average there are only 80-90 days/year with atmospheric conditions ideal for flying research aircraft models. For our purposes we define as ideal the following atmospheric conditions:

- absence of precipitation;
- temperatures between 45°F and 80°F;
- humidity ratio < 65%;
- wind < 12 mph.

A customized wood platform was designed and built for the transportation of the model to the flight testing facilities on a WVU vehicle, as shown in Figures 3.1 and 3.2. A second vehicle with all the ground equipment and the flight testing personnel was also used. Details of the runway at the WVU Jackson Mill facilities are shown in Figure 3.3 and 3.4.

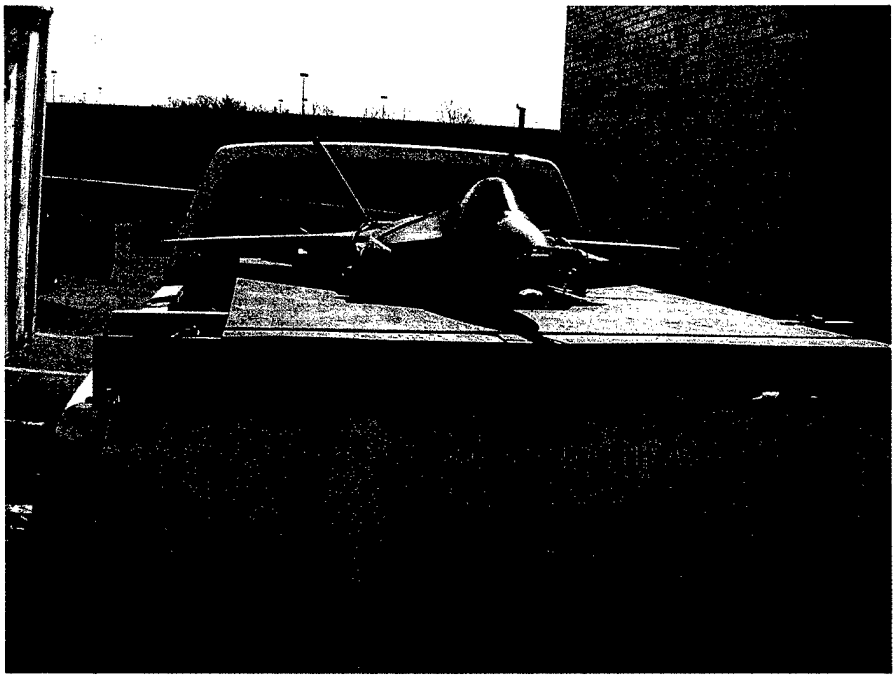


Figure 3.1 – Details of the transportation set up of the WVU YF-22 model

Figure 3.2 – Details of the transportation set up of the WVU YF-22 model

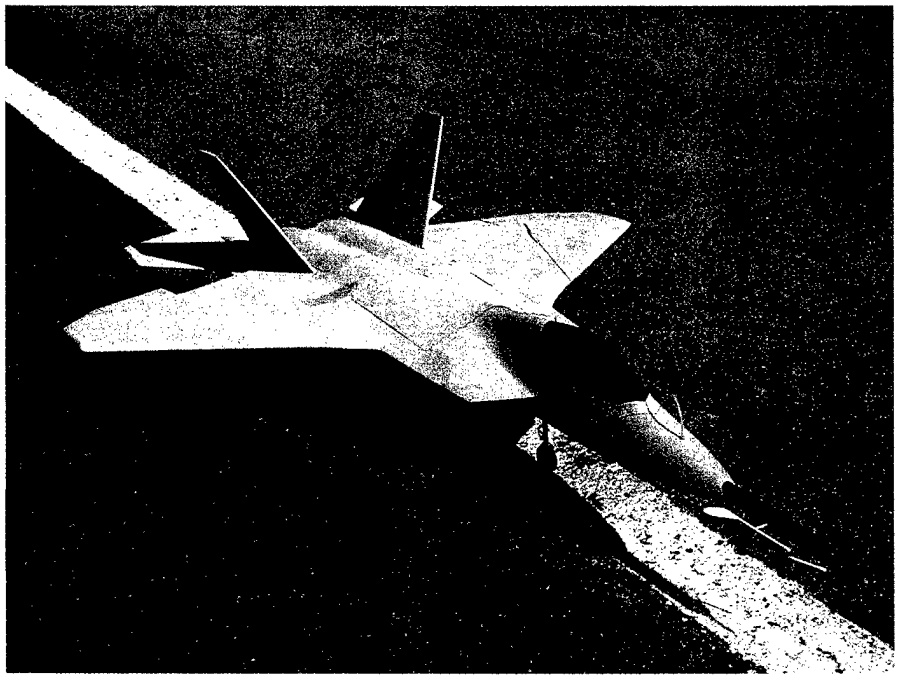


Figure 3.3 – WVU YF-22 model at the Jackson Mill facility

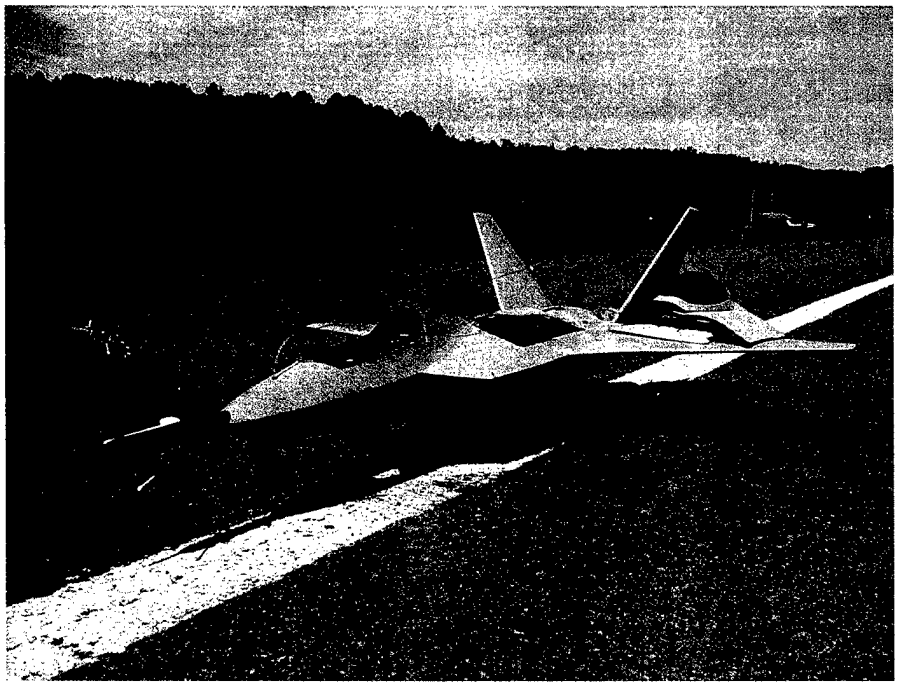


Figure 3.4 – WVU YF-22 model at the Jackson Mill facility

3.2. – Flight Testing of the WVU YF-22 Model

A total of 37 flights have been conducted within the activities of the project from Summer 1999 until late Fall 2001.

The flight activities have been divided in the following phases:

PHASE #1: Ground/taxi tests of the WVU YF-22 aircraft model.
These tests were conducted for the following purposes:
- assessment of the aircraft handling qualities on the runway;
- system check of the RC radio system;
- assessment of the fuel consumption.

No particular problems were detected in this phase. Minor adjustments and/or refinements were introduced on the model and the RC system throughout this phase.

PHASE #2 (9 flights) : Flight testing of the WVU YF-22 model in RC mode only.
These tests were conducted for the following purposes:
- assessment of the propulsion performance;
- assessment of the longitudinal and lateral directional handling qualities;

These flights allowed the pilot and the flight crew to perform:
- accurate balancing and selection of an optimal range for the aircraft CG (without electronic payload);
- evaluation of the trimming characteristics and selection of the actuator gains for the RC system.

PHASE #3 (6 flights): Introduction of "dummy" weight (to simulate the electronic payload).

These tests were conducted for the purpose of evaluating the modifications in handling qualities and propulsion performance for the model following the installation of the payload (without the risks of loosing the payload in the event of an accident). Dummy weights were progressively introduced with 1½ lbs increment until the weight with the electronic payload was reached. Emphasis was placed in ensuring that the addition of the dummy weight did not imply a shift from the desirable range for the aircraft CG. No substantial deteriorations of the handling qualities were observed for altitude flight in addition to the expected deterioration of the propulsion performance during the climb phase. However, a substantial increase of the pilot workload was observed during the landing and approach phases. A partial use of the flaps was found to be necessary to avoid high values for the longitudinal angle of attack during the final approach. Overall, with minor adjustments to the trim configurations, the aircraft was found to be an ideal platform for the following flight tests.

PHASE #4 (9 flights) : Installation and performance assessment of the OBC (with data acquisition software) and the electronic payload

These tests were conducted for the purpose of evaluating the overall performance of the electronic payload. The OBC was designed to collect data from all sensors and stores them in the 16 MB flash card for post flight downloading. In particular the purpose of this phase was for testing the following subsystems:

- the network of sensors;
- the OBC;
- the data acquisition system;
- the power system.

A few problems – ranging from minor to substantial - were experienced throughout this phase leading to modifications and/or improvements in the electronic payload. Particularly, the occurrence of significant electronic/magnetic interference (EMI) problems was noticed; these problems were accurately analyzed with a spectrum analyzer and solved with the modifications described in Section#2. Another modification was the replacement of the potentiometers for the α and β vanes on the nose of the YF-22 model with more sensitive types since the original potentiometers were found to be not accurate in the low speed range. Overall, except for the EMI issues, this flight testing phase was fairly smooth and confirmed the reliability of the electronic payload.

PHASE #5 (7 flights): Acquisition of flight data for a parameter identification (PID) study (for the determination of a mathematical model of the YF-22 model) and off-line training for the neural networks of the SFDIA and AFDIA schemes

The objective of this phase was to collect a large set of flight data containing data from typical PID (Parameter Identification) maneuvers, that is longitudinal and lateral-directional doublets. An average of approximately 11 maneuvers were performed in each flight with a total of 76 doublets, that is 27 elevator doublets, 23 aileron doublets, and 26 rudder doublets. These PID maneuvers, spanning over a total of approximately 52 minutes of flight time (excluding take off and landing phases) were also ideal for the purpose of performing an initial off-line training for the different Multi Layer Perceptron

(MLP)-type neural networks (NNs) of the SFDIA and AFDIA schemes. The results of this study are outlined in the next sections of this document.

PHASE #6 (3 flights): Flight testing of the SFDIA scheme
The objective of this phase was to collect flight data showing the performance of the SFDIA scheme. A total of 6 sensor failures were artificially "injected" on the pitch, roll, and yaw rate sensors in the flight control system by the pilot through the GCU. The numerical results relative to the performance of the SFDIA scheme are reported in Section #6.

PHASE #7 (3 flights): Flight testing of the AFDIA scheme
The objective of this phase was to collect flight data showing the performance of the AFDIA scheme. Special modifications – described in Section #6 - were introduced on the electronic payload to install the failure trigger hardware/software and doing flight tests with aileron failure. The failure trigger scheme proved to be quite efficient and reliable. A total of 4 actuator failures were triggered by the pilot during the flight with the GCU with the pilot able to "remove" the failure at any moment during the flight. The numerical results relative to the performance of the AFDIA scheme are reported in Section #6.

3.3. - Sample of Flight Testing Data

Samples of flight test data are enclosed below. The flight data are relative to Flight #3 of Phase #5. The OBC recorded 600 seconds of data including 310 seconds of flight time excluding take off, climbing, approach, and landing. Four of each of three types of maneuvers (elevator, lateral, and directional doublets) were performed throughout this flight.

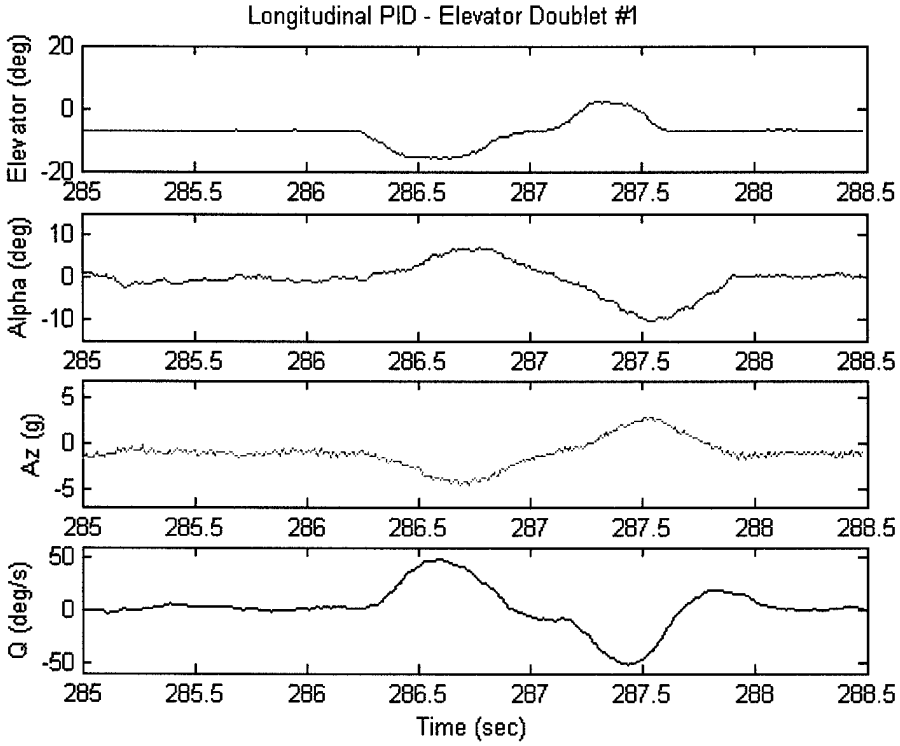


Figure 3.5 – Flight data of the 1st longitudinal maneuver (Flight #1-Phase #5)

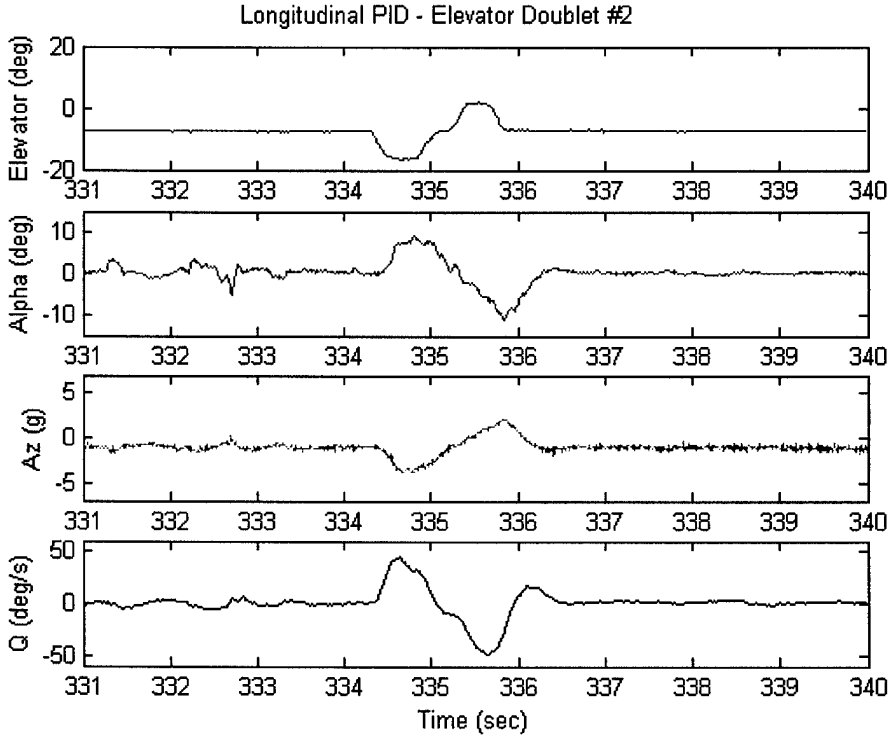


Figure 3.6 – Flight data of the 2nd longitudinal maneuver (Flight #1-Phase #5)

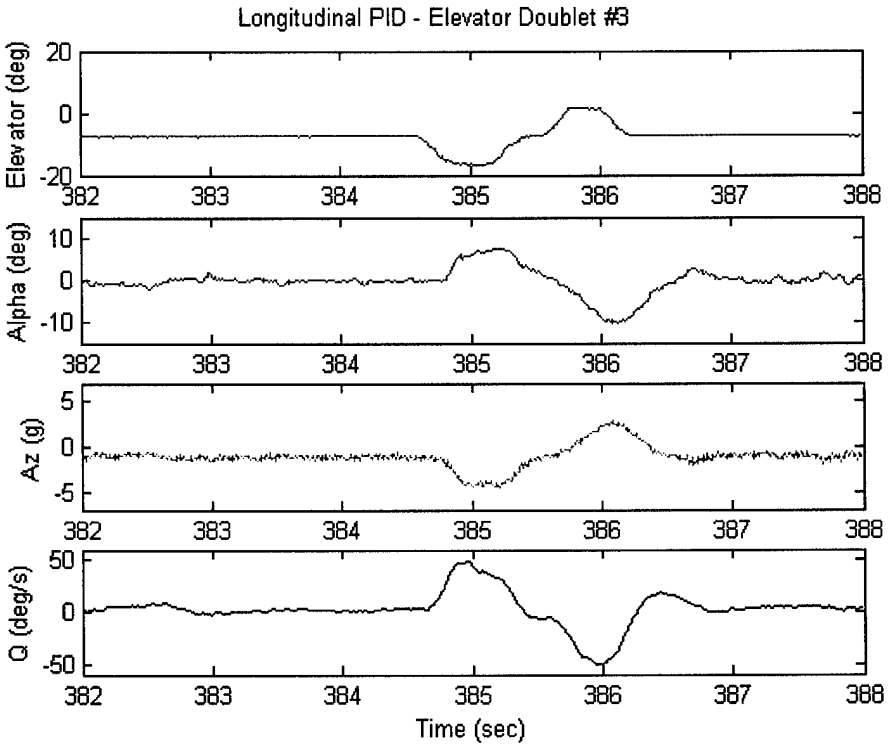


Figure 3.7 – Flight data of the 3rd longitudinal maneuver (Flight #1-Phase #5)

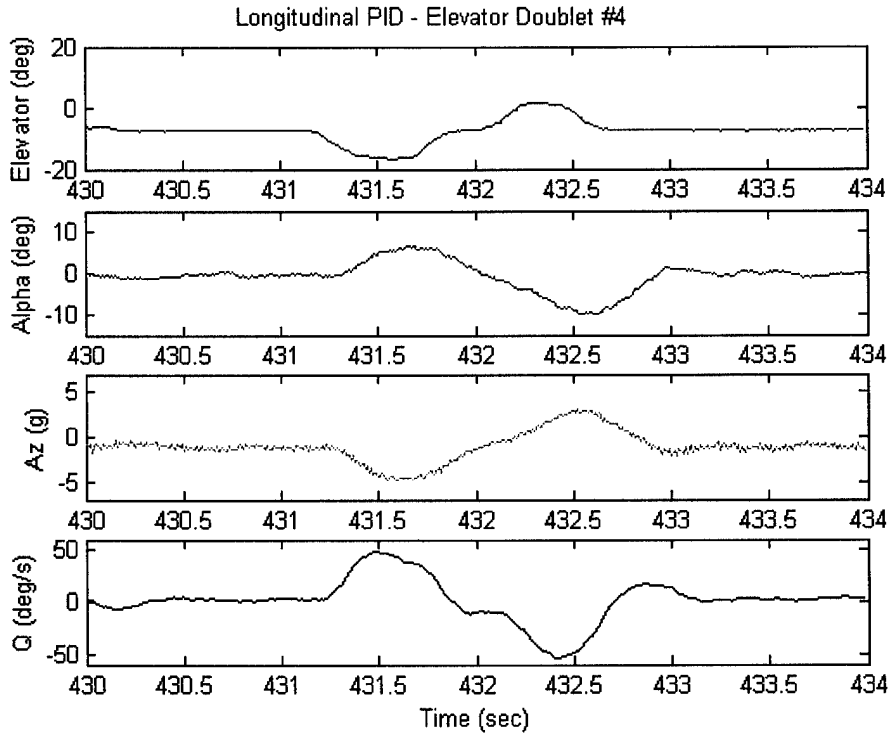


Figure 3.8 – Flight data of the 4th longitudinal maneuver (Flight #1-Phase #5)

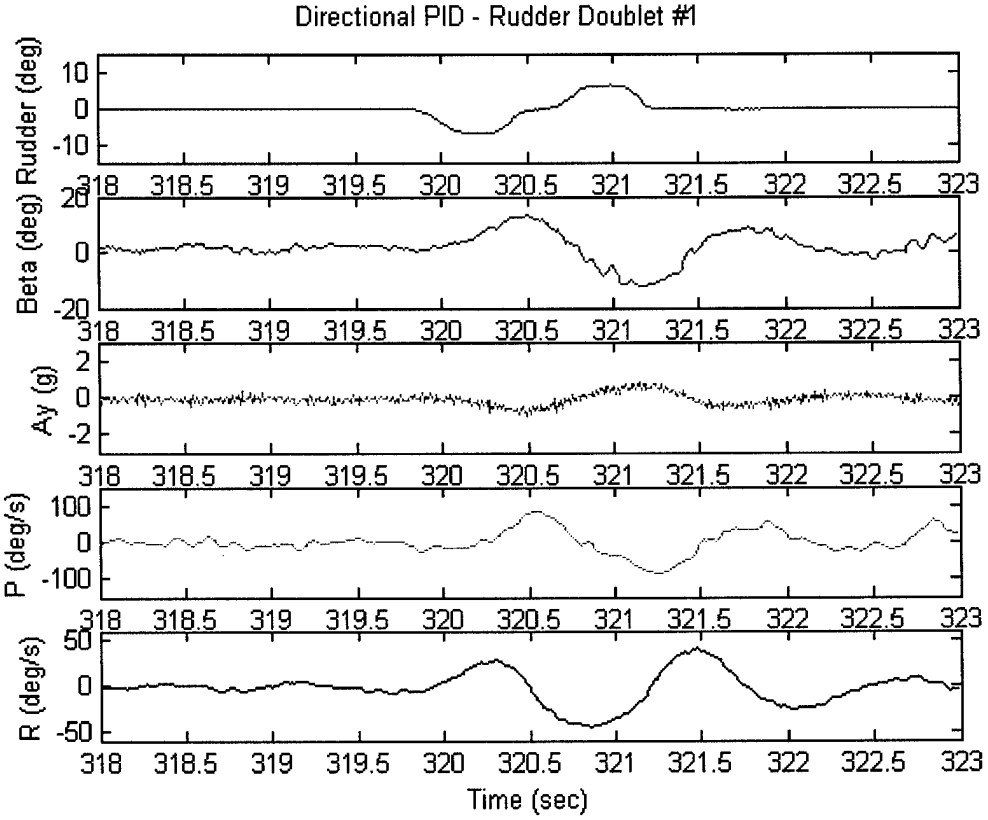


Figure 3.9 – Flight data of the 1st directional maneuver (Flight #1-Phase #5)

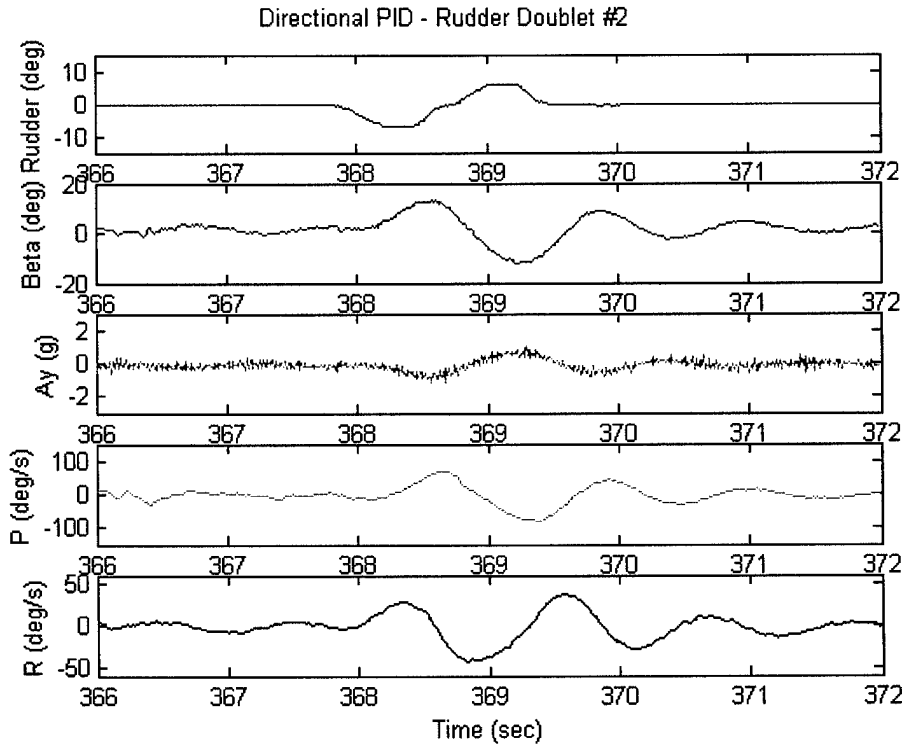


Figure 3.10 – Flight data of the 2nd directional maneuver (Flight #1-Phase #5)

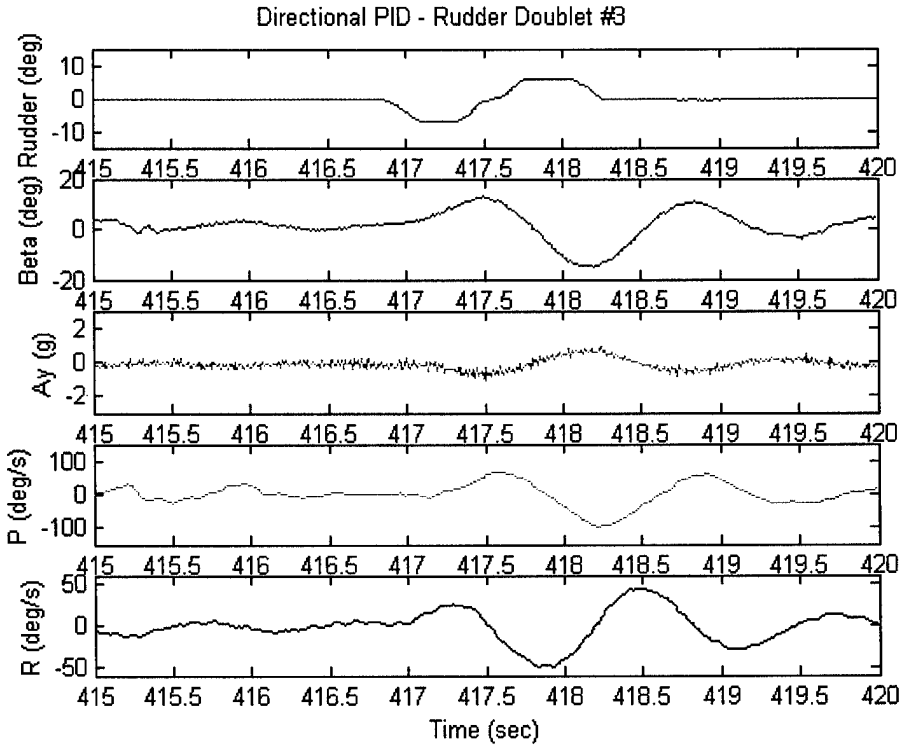


Figure 3.11 – Flight data of the 3rd directional maneuver (Flight #1-Phase #5)

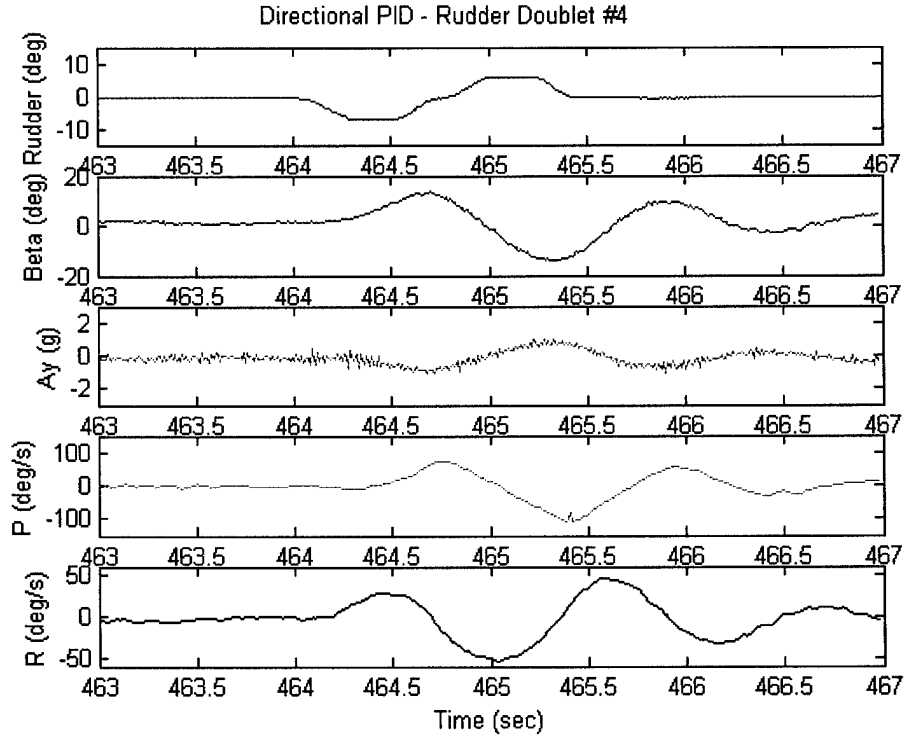


Figure 3.12 – Flight data of the 4th directional maneuver (Flight #1-Phase #5)

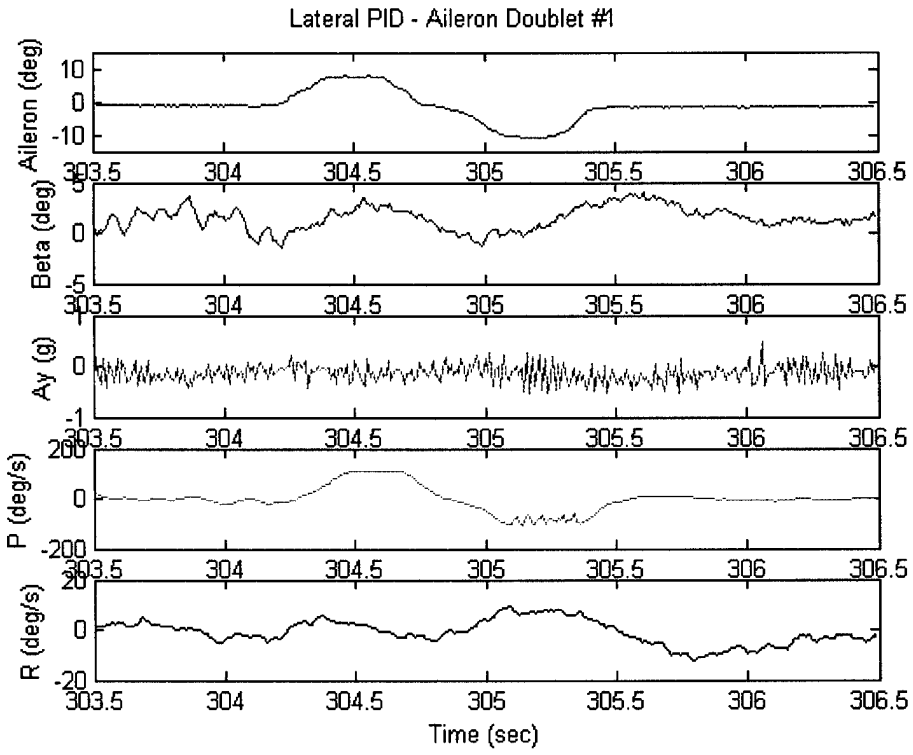


Figure 3.13 – Flight data of the 1st lateral maneuver (Flight #1-Phase #5)

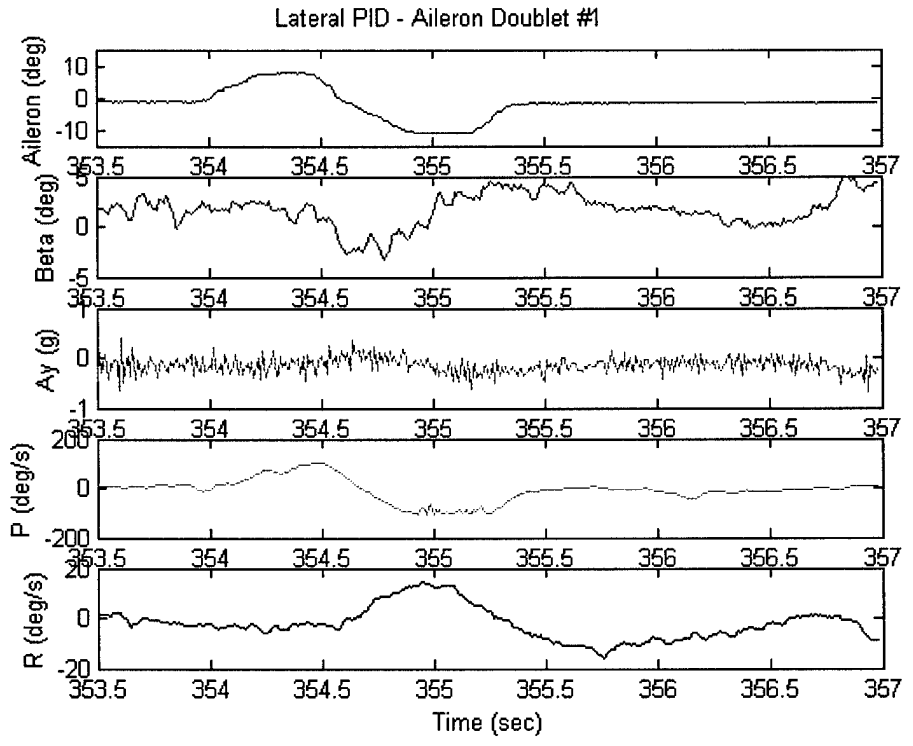


Figure 3.14 – Flight data of the 2nd lateral maneuver (Flight #1-Phase #5)

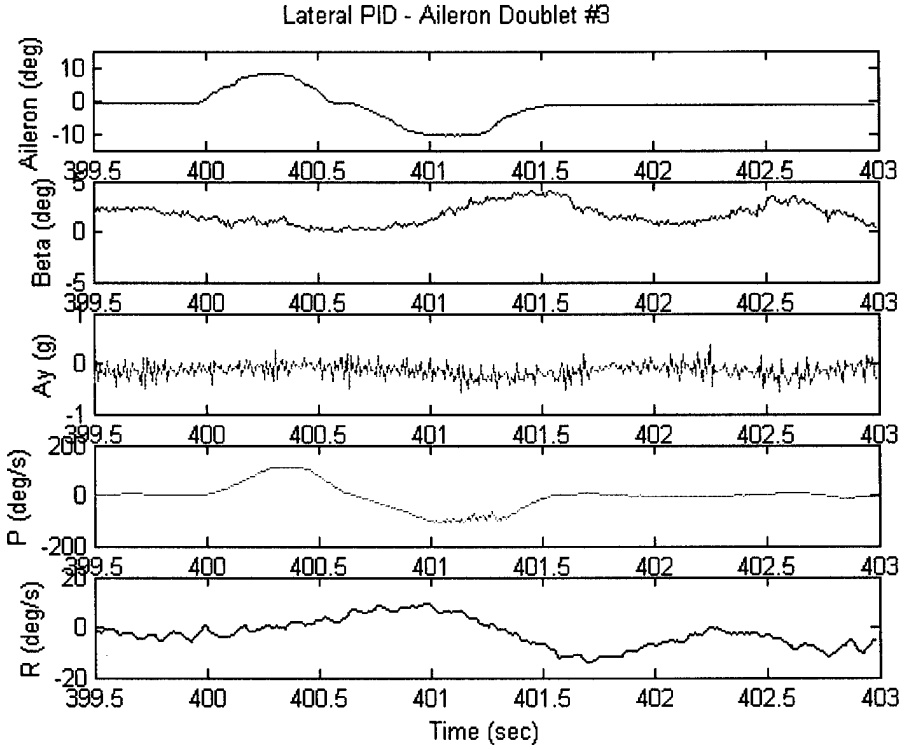


Figure 3.15 – Flight data of the 3rd lateral maneuver (Flight #1-Phase #5)

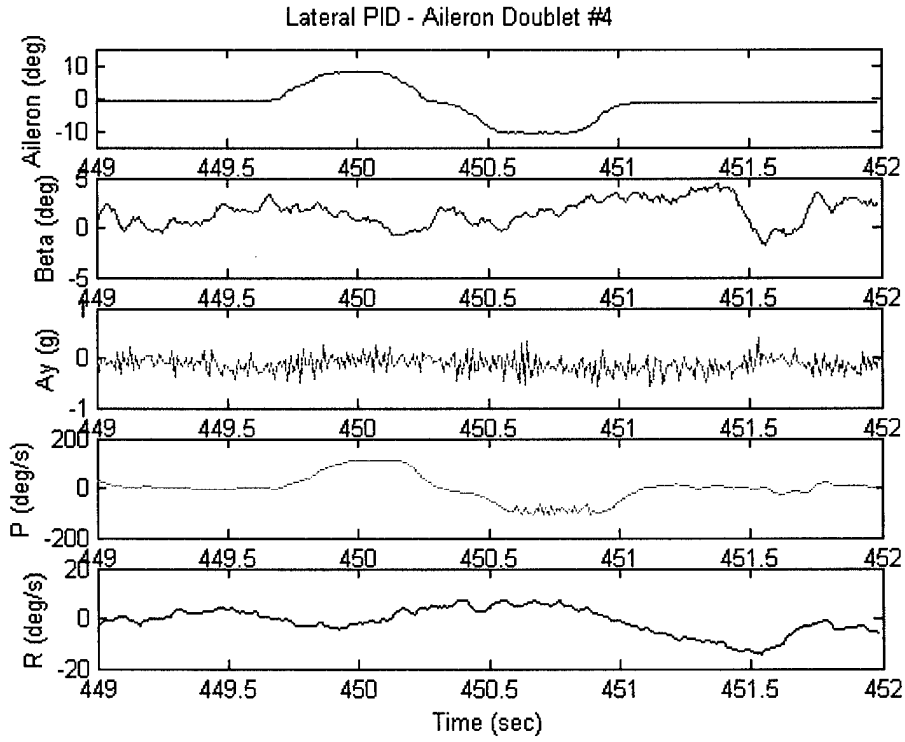


Figure 3.16 – Flight data of the 4th lateral maneuver (Flight #1-Phase #5)

Section #4 - Table of Contents

List of Symbols (Section #4)

English

a	Acceleration (ft/sec^2)
k	Discrete time index
J	Cost function
m	Pattern for the neural network input data
p	Aircraft angular velocity around the x body axis (roll rate), rad/sec
q	Aircraft angular velocity around the y body axis (pitch rate), rad/sec
r	Aircraft angular velocity around the z body axis (yaw rate), rad/sec
t	Time, sec

Greek

α	Angle of attack, rad or deg
β	Angle of sideslip, rad or deg
β	Unknown coefficients (for PID)
ε	Random variables (for PID)
θ	Pitch Euler angle, rad or deg
δ	Control surface deflection, rad or deg
ϕ	Roll Euler angle, rad or deg
ψ	Yaw Euler angle, rad or deg

Subscripts

A	Aileron
E	Elevator
L	Left side
R	Right side
R	Rudder

Vectors & Matrices

A	System state matrix (for PID)
B	Input system matrix (for PID)
C	Output system matrix (for PID)
D	Input/Output system matrix (for PID)
e	Error (for PID)
O	Neural network output
R	Auto or cross correlation function
W	Weight matrix (for PID)
X	Known system inputs (for PID)
Y	Parameters to be estimated
Y	Known system outputs (for PID)

Acronyms

AFA	Actuator Failure Accommodation
AFDI	Actuator Failure Detection and Identification
AFDIA	Actuator Failure Detection, Identification, and Accommodation
BLS	Batch Least Square

BPA	Back Propagation Algorithm
DTFT	Discrete Time Fourier Transform
DNN	Decentralized Neural Network
DQEE	Decentralized Quadratic Estimation Error
EBPA	Extended Back Propagation Algorithm
FDI	Failure Detection and Identification
FTR	Fourier Transform Regression
LS	Least Square
LWR	Locally Weighted Regression
ML	Maximum Likelihood
MNN	Main Neural Network
MQEE	Main Quadratic Estimation Error
NNC	Neural Network Controller
OBC	On-Board Computer
OQEE	Output (of NN) Quadratic Estimation Error
PID	Parameter Identification
PTBU	(Neural) Parameters To Be Updated (at each computational step)
RLS	Recursive Least Squares
RMS	Root Mean Square
SFA	Sensor Failure Accommodation
SFDI	Sensor Failure Detection and Identification
SFDIA	Sensor Failure Detection, Identification, and Accommodation

4.1 - The Parameter Identification (PID) Problem

4.1.1 – Introduction to the PID Problem

While one of the advantages of neural estimators and neural controllers within a fault tolerant flight control system is the on-line learning capability, for a safer approach the on-line neural learning could start in flight from previously off-line trained neural architectures within the fault tolerant schemes. Thus, there was a need to determine an approximate mathematical model of the WVU YF-22 aircraft in terms of the conventional state matrices (A,B,C,D) so that a simulation code could be developed for implementing the off-line training. In this section this study will be referred to as a parameter identification (PID) study although technically it is a model identification study. However, in the technical literature the acronym 'PID' is often used for both "parameter identification" and "model identification" with the rational that "model identification" implies the determination of the coefficients (parameters) of the matrices of the state variable model.

The research group coordinated by the Principal Investigator has had experience with off-line batch PID, mainly using the well-known Maximum Likelihood (ML) method ([Iliff1972], [Iliff1976], and [Maine1986]). In addition, recently a research effort was conducted at WVU toward the selection of an on-line real-time PID scheme to be used within the NASA Intelligent Flight Control System (IFCS) F-15 program. Thus, the codes for several algorithms suitable for on-line PID applications were developed for the IFCS project and already available to the research team for the purpose of this project.

The "deliverable" of this PID study was an estimated mathematical model of the WVU YF-22 aircraft – in terms of the matrices of the state variable model - on which the off-line training for the NNs of the fault tolerant schemes was based.

The next paragraphs of this section will review 3 methods used for this PID study. The first method is the well-known Batch Least Square (BLS), which is the most common PID technique for batch off-line applications. For on-line real time applications typically there are two main categories of PID methods, that is on-line PID methods in the time domain and in the frequency domain. Thus, the other two PID methods used in this study (the Locally Weighted Regression (LWR) method and the Fourier Transform Regression (FTR) method) are techniques from these two different categories. Several additional PID methods have been described in the technical literature; however, an even more extensive PID analysis using additional PID methods was deemed to be outside the scope of this research effort whose emphasis is on NN-based fault tolerant schemes. Another paragraph will provide a review and a comparison of the PID results using the WVU F-22 flight data acquired in Phase 5 (as described in Section 3). Although 2 of the 3 PID algorithms outlined below are suitable for on-line applications, the PID analysis for the WVU YF-22 aircraft was intended to be off-line and the implementation of the PID software on the on-board computer (OBC) was of no interest.

4.1.2 – The Batch Least Squares (BLS) PID Method

The Batch Least Squares (BLS) technique consists essentially in solving an over-determined linear system in the least squares sense ([Mendel1973], [Ljung1987], and [Neter1996]). It is one of the most widely used approaches for the estimation of a vector

of parameters from a collection of "almost-linearly" related input-output data. Being based on linear algebra, this approach leads to an elegant formulation and a straightforward analysis, allowing the use well known algorithms. In essence, the reliability of this method comes from the propriety that a pseudo-inverse solution for a linear system with more equations than unknowns is optimal in the least squares sense. The general linear regression model is given by:

$$Y = X\beta + \varepsilon \tag{4.1}$$

where Y is a $(n\times 1)$ vector of known responses of the system, X is a $(n\times p)$ matrix of known inputs to the system (note that the last column of this matrix is usually a column of ones allowing for a "bias" - namely a constant input to the system – to be introduced), β in the $(p\times 1)$ vector of parameters to be estimated, and ε is a $(n\times 1)$ vector of independent normal random variables, with zero mean ($E\{\varepsilon\} = 0$) and unknown diagonal variance-covariance matrix. This matrix is generally assumed to be a multiple of the $(n\times n)$ identity matrix: ($\sigma^2\{\varepsilon\} = \sigma^2 I$). Therefore we have that $E\{Y\} = X\beta$ and $\sigma^2\{Y\} = \sigma^2 I$. The problem is to find the vector β such that $X\beta$ (which is the expected value of Y) is as close as possible (in the least squares sense) to Y, so that $\sigma^2$is minimized. Particularly, the objective is to find the value of β that minimizes the following quadratic index:

$$Q = \varepsilon^T \varepsilon = (Y - X\beta)^T (Y - X\beta) \tag{4.2}$$

The solution to this problem is given by:

$$b = X^+ Y = (X^T X)^{-1} X^T Y \tag{4.3}$$

It can be shown - using the Gauss-Markov theorem - that this solution is such that the error vector:

$$e = Y - Xb \tag{4.4}$$

has zero mean – meaning unbiased estimation - and minimum variance among all the possible linear unbiased solutions; thus, the relative estimation is known as *BLUE* (Best Linear Unbiased Estimation). Furthermore, it can be shown that the resulting estimation for σ^2 is the MSE value (Mean Square Error):

$$MSE = \frac{e^T e}{n - p} \tag{4.5}$$

The covariance of the solution is:

$$\begin{aligned} \sigma^2\{b\} &= E\{(b - E\{b\})(b - E\{b\})^T\} = (X^T X)^{-1} X^T \sigma^2\{Y\} X (X^T X)^{-1} \\ &= \sigma^2 (X^T X)^{-1} \end{aligned} \tag{4.6}$$

Substituting the MSE in lieu of σ^2 in equation *(1.6)* we obtain:

$$\sigma^2\{b\} = \frac{e^T e}{n-p}(X^T X)^{-1} \tag{4.7}$$

Since the problem of interest consists in the identification of a linear system of the form:

$$\begin{bmatrix} \dot{x}(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{4.8}$$

by transposing (4.7) the PID problem can be set up as in the following:

$$\begin{aligned} Y &= \begin{bmatrix} \dot{x}^T(t) & y^T(t) \end{bmatrix} \\ X &= \begin{bmatrix} x^T(t) & u^T(t) \end{bmatrix} \\ \beta &= \begin{bmatrix} A & B \\ C & D \end{bmatrix}^T \end{aligned} \tag{4.9}$$

The BLS PID method is not suitable for on-line real time applications within flight control system due to the complexity of the calculations associated with the relationships described by Eqs. (4.3), (4.8), and (4.9).

4.1.3 - The Locally Weighted Regression (LWR) PID Method

The Locally Weighted Regression (LWR) is a particular time domain PID method ([Atkeson1997a], [Atkeson1997b]). The LWR PID algorithm is characterized by featuring both a weighting of the associated linear system by a diagonal matrix and a retention and deletion algorithm. The weighting allows selecting the *"most important"* data while, at the same time, avoiding to completely forgetting the (past) information enclosed in the rest of the available data. This particular feature could be precious for on-line estimation problems where the system to be estimated might be time varying.

In the LWR method the equation (4.1) is weighted with a diagonal matrix W to express the major/minor importance of a particular row of data. Therefore:

$$WY = WX\beta + W\varepsilon \tag{4.10}$$

The goal is to find the value of β that minimizes the following quadratic index:

$$Q = \varepsilon^T W^2 \varepsilon = (WY - WX\beta)^T (WY - WX\beta) \tag{4.11}$$

The solution is provided by:

$$b = (WX)^{+} WY = (X^T W^2 X)^{-1} X^T W^2 Y \tag{4.12}$$

where $W^2 = W^T W$. This solution is such that the weighted error vector:

$$We = WY - WXb \tag{4.13}$$

has zero mean - unbiased estimation - and minimum variance among all the possible linear unbiased solutions. The covariance matrix of the solution is given by:

$$\begin{aligned} \sigma^2\{b\} &= E\{(b - E\{b\})(b - E\{b\})^T\} = (X^T W^2 X)^{-1} X^T W^2 \sigma^2\{Y\} W^2 X (X^T W^2 X)^{-1} \\ &= \sigma^2 (X^T W^2 X)^{-1} X^T W^4 X (X^T W^2 X)^{-1} \end{aligned} \tag{4.14}$$

Substituting the MSE in lieu of σ^2 in equation we obtain:

$$\sigma^2\{b\} = \frac{e^T e}{n - p} (X^T W^2 X)^{-1} X^T W^4 X (X^T W^2 X)^{-1} \tag{4.15}$$

If $W \approx I$, we can also consider $X^T W^4 X (X^T W^2 X)^{-1} \approx I$.

For on-line estimation problems, the size of $D=WX$ needs to be constrained (depending on the available computational power). Therefore, once the number of rows in D reaches its maximum user-selected predefined value, the problem of which rows in D should be replaced by the new data needs to be faced. A simple strategy would be to delete the oldest rows, or the ones that are in the least squares sense "*most different*" from the current one. However, this strategy could cause the deletion of rows that bring precious information to the estimation, causing, therefore, ill conditioning and increases in the estimation variances for the relative parameters. A better strategy (the one used in the implemented algorithm) would be to delete the rows of D that "*bring less information*" for the estimation process. Thus, when those rows will be replaced, is it likely that the new rows will bring more information than the ones just deleted. This would decrease the estimation variances for the parameters. This strategy is accomplished by replacing the rows of D whose deletion cause the trace of $(D^T D)^{-1}$ to increase the least, since the lower is this trace, the most well-conditioned is $(D^T D)^{-1}$ and the lower is the variance of the estimated parameters.

4.1.4 - The Fourier Transform Regression (FTR) PID Method

The Fourier Transform Regression (FTR) is a particular frequency domain method suitable for on-line PID ([Morelli1997], [Morelli1998], and [Morelli1999]). Consider a linear system of conventional continuous-time aircraft model given by:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{4.16}$$

For a generic signal $x(t)$, the Discrete Time Fourier Transform (DTFT) of a finite number of samples $x(i\Delta t)$ is given by ([Klein1978]):

$$\tilde{x}(\omega) = \Delta t \sum_{0}^{N-1} x(i\Delta t) e^{-j\omega i\Delta t} = X(\omega)\Delta t \tag{4.17}$$

Applying the DTFT to the samples of the previous linear model will provide:

$$\begin{aligned} j\omega\tilde{x}(\omega) &= A\tilde{x}(\omega) + B\tilde{u}(\omega) \\ \tilde{y}(\omega) &= C\tilde{x}(\omega) + D\tilde{u}(\omega) \end{aligned} \tag{4.18}$$

As for the LS and BLS regression methods, the measurements of the vector x, u, and y can be used to set up a cost function having the coefficients of A, B, as argument. In particular, consider the generic k-th state equation over a set of frequency points $[\omega_1, \omega_2, \cdots, \omega_m]$:

$$\begin{bmatrix} j\omega_1\tilde{x}_k(\omega_1) \\ j\omega_2\tilde{x}_k(\omega_2) \\ \cdots \\ \cdots \\ j\omega_m\tilde{x}_k(\omega_m) \end{bmatrix} = \begin{bmatrix} \tilde{x}^T(\omega_1), \tilde{u}^T(\omega_1) \\ \tilde{x}^T(\omega_2), \tilde{u}^T(\omega_2) \\ \cdots \\ \cdots \\ \tilde{x}^T(\omega_m), \tilde{u}^T(\omega_m) \end{bmatrix} \begin{bmatrix} A_k^T \\ B_k^T \end{bmatrix} \tag{4.19}$$

where A_k and B_k are the k-th row of the state and control matrices A and B, respectively; $\tilde{x}_k(\omega_n), \tilde{u}(\omega_n)$ are the DTFT of the k-th state and control variable relative to the frequency ω_n. If we denote the above equation as $Y = X\Theta + \varepsilon$ with complex equation error, ε, and obvious definitions for Y, X, and Θ, the problem can be formulated as a standard LS regression problem with complex data. Thus with the following cost function

$$\begin{aligned} J_k &= \frac{1}{2}\sum_{n=1}^{m} \left| j\omega_n\tilde{x}_k(\omega_n) - A_k\tilde{x}(\omega_n) - B_k\tilde{u}(\omega_n) \right|^2 \\ &= \frac{1}{2}(Y - X\Theta)^*(Y - X\Theta) \end{aligned} \tag{4.20}$$

the parameter vector minimizing the above function is immediately given by:

$$\hat{\Theta} = \left[\mathrm{Re}(X^*X)\right]^{-1} \mathrm{Re}(X^*Y) \tag{4.21}$$

where * indicates a complex conjugate transpose. It should be emphasized that the cost function is made of a summation over m frequencies in a range of frequencies of interest.

In addition, the covariance matrix of the estimates of Θ is computed as:

$$\mathrm{cov}(\hat{\Theta}) = E\left\{(\hat{\Theta}-\Theta)(\hat{\Theta}-\Theta)^*\right\} = \sigma^2 \cdot \left[\mathrm{Re}(X^*X)\right]^{-1} \tag{4.22}$$

where σ^2 is the equation error variance and can be estimated on-line using

$$\hat{\sigma}^2 = \frac{1}{(m-p)}[(Y - X\hat{\Theta})^*(Y - X\hat{\Theta})] \tag{4.23}$$

Furthermore, the standard deviation of the estimation error for the l-th unknown of the p parameters in Θ can be evaluated as the square root of the l,l-th coefficient (main-diagonal coefficient) of the covariance matrix. This standard deviation allows for an on-line assessment of the accuracy of the estimates of the parameter. The type of required on-line calculations should also be described for an assessment of the computational effort. For a given frequency, ω_n, the DTFT at the i-th time step is related to the DTFT at the (i-1)-th time step as follows:

$$X_i(\omega_n) = X_{i-1}(\omega_n) + x_i e^{-j\omega_n i\Delta t} \tag{4.24}$$

where:

$$e^{-j\omega_n i\Delta t} = e^{-j\omega_n \Delta t} e^{-j\omega_n (i-1)\Delta t} \tag{4.25}$$

since the quantity $e^{-j\omega_n \Delta t}$ is constant for a given ω_n and a given Δt, the on-line computation of $X_i(\omega_n)$ requires a reasonably low computational effort. In addition, the scheme requires only a fixed memory space for $X(\omega)$ even if $X(\omega)$ is updated at every step. Furthermore, a very important characteristic of this technique is that the time domain data from previous flight maneuvers – containing "good information" for PID purposes – can still be used by simply iterating the calculation of the DTFT. Thus, this DTFT approach allows for retaining all the PID results from previous time steps and, at the same time, provides the necessary flexibility to follow changes in the system dynamics. In terms of frequency range, the m frequencies over which the cost function is evaluated can be selected as evenly spaced between $\omega_{\min}$ and $\omega_{\max}$. Typically, the rigid body dynamics frequency range for the considered aircraft can be selected allowing, therefore, filtering out higher frequency noise and/or structural interference. The size of the frequency range to be selected is related to the rigid body dynamics for the considered aircraft; clearly a smaller range of frequency would decrease the computational effort. Since the DTFT is recursively computed, the part of the algorithm requiring the most computational effort is the inversion of the matrix $\mathrm{Re}(X^TX)$. This inverse is computed using the Singular Value Decomposition (SVD) for robustness and convergence issues. In more details, for each vector Θ of parameters to be estimated, one SVD ($O(n^3)$ flops) of the matrix $\mathrm{Re}(X^TX)$ (average size 6 by 6) has to be performed for each computational step.

4.1.5 – PID Results

Throughout the flights of Phase #5 (as described in Section #3) 76 PID maneuvers were executed; in particular flight data from 27 elevator doublets, 23 aileron doublets, and 26 rudder doublets were collected by the OBC (as described in Section #2).

The PID analysis is performed in a Matlab/Simulink environment. As a first step, flight data from a specific PID maneuver from a selected PID flight are fed into a Simulink-based filter to remove sensor noise. The filter cut-off frequencies were carefully selected to avoid removing significant portions of the WVU YF-22 rigid body dynamics. For the LWR and FTR PID analysis the filtered data are fed into two Simulink executable schemes. The general main Simulink PID scheme is shown in Figure 4.1.

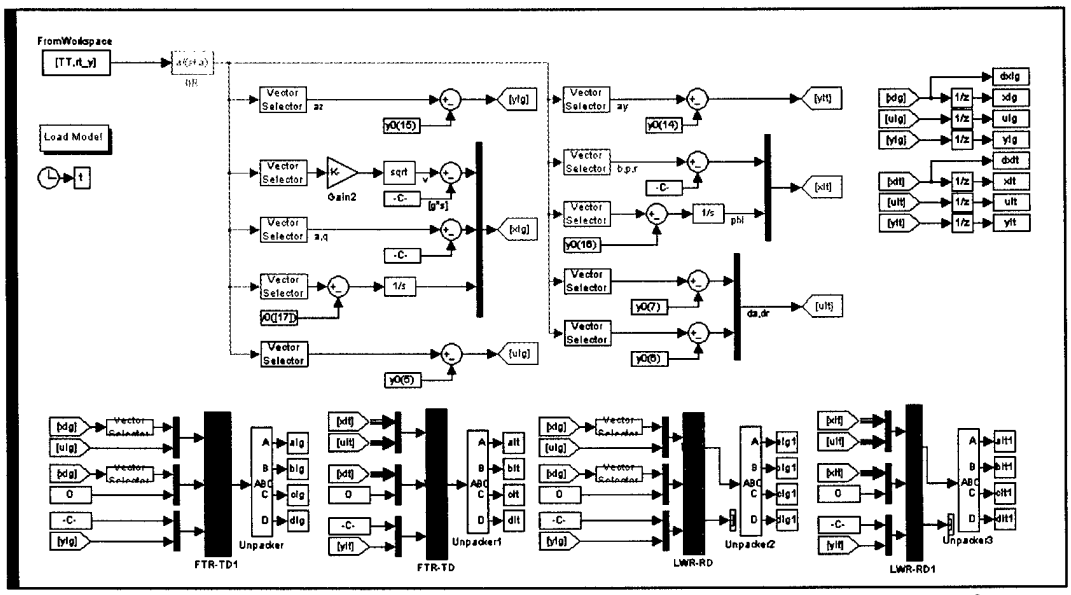


Figure 4.1 - Main Simulink PID scheme for LWR and FTR analysis

In particular the LWR and FTR PID methods receive as PID inputs the following input and state variables: α, q, δ_e (for the longitudinal PID), β, p, r, Φ, δ_a, δ_r (for the lateral and directional PID). A similar Simulink scheme is used for the BLS PID analysis using the same filtered data. Two slightly different version of the BLS algorithm are used; for the 1st version ("BLS_0") the selected state and input variables are V, α, q, δ_e and θ for the longitudinal PID analysis and β, p, r, Φ, δ_a, δ_r for the lateral directional PID analysis. Within a 2nd version of the BLS method ("BLS_1") the variables V, θ and Φ are not used. A key difference between the two on-line methods (the LWR and the FTR methods – although they are used off-line in this effort) and the BLS is in the computation of the time derivative of the state vector. In fact, for the LWR and FTR methods this derivative is either approximated by '*(x(t)-x(t-T))/T* ' (as in the LWR case) or evaluated in the frequency domain. The BLS method does not attempt to directly compute this derivative, but, instead, directly identifies the discrete time equivalent of the system, and therefore only needs the state at the next sampling time x_{k+1}. This issue makes the PID algorithm available in batch mode only; the system is then converted to continuous time.

Using all the different PID maneuvers the statistics of the results of the PID analysis in terms of estimates of the (A,B,C,D) matrices are shown below.

Longitudinal PID Results (in terms of state variable matrices)

PID Method #1: BLS_0

NOTE: The BLS_0 method uses measurements of V and θ.

$$\begin{bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.1183 & -0.0036 & 0.0014 & 0.0000 \\ -15.4990 & -2.2636 & 0.7204 & 0.0099 \\ -99.0799 & -20.0915 & -1.1617 & -0.0121 \\ 0.4991 & 0.1010 & 1.0057 & 0.0001 \end{bmatrix} \begin{bmatrix} V \\ \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.0078 \\ 1.0984 \\ -19.0852 \\ 0.0954 \end{bmatrix} \delta_e \qquad (4.26)$$

$$Az = \begin{bmatrix} -3.0907 & -0.2253 & -0.0274 & 0.0004 \end{bmatrix} \begin{bmatrix} V \\ \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} -0.0438 \end{bmatrix} \delta_e \qquad (4.27)$$

PID Method #2: BLS_1

NOTE: The BLS_1 method does not use measurements of V and θ.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -1.7259 & 0.5199 \\ -17.9653 & -1.5262 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} 0.3823 \\ -20.1494 \end{bmatrix} \delta_e \qquad (4.28)$$

$$Az = \begin{bmatrix} -0.1450 & -0.0473 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -0.1091 \end{bmatrix} \delta_e \qquad (4.29)$$

PID Method #3: FTR

NOTE: The FTR method does not use measurements of V and θ.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.8764 & 0.7982 \\ -13.9989 & -1.2866 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} 2.0277 \\ -16.2285 \end{bmatrix} \delta_e \qquad (4.30)$$

$$Az = \begin{bmatrix} -0.0947 & -0.0495 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -0.1099 \end{bmatrix} \delta_e \qquad (4.31)$$

PID Method #4: LWR

NOTE: The LWR method does not use measurements of V and θ.

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\end{bmatrix}=\begin{bmatrix}1.4737 & -0.3432\\ -5.3702 & -3.7541\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix}+\begin{bmatrix}-1.7452\\ -14.6898\end{bmatrix}\delta_e \tag{4.32}$$

$$Az=\begin{bmatrix}0.2498 & -0.0754\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix}+\begin{bmatrix}-0.0660\end{bmatrix}\delta_e \tag{4.33}$$

Note that the LWR PID longitudinal results were considered to be unacceptable since the eigenvalues analysis of the linear model estimated by the LWR methods provided an unstable eigenvalue, which is inconsistent with the stable longitudinal dynamic exhibited by the WVU YF-22 aircraft.

Lateral-directional PID Results (in terms of state variable matrices)

PID Method #1: BLS_0

NOTE: The BLS_0 method uses measurements of Φ.

$$\begin{bmatrix}\dot{\beta}\\ \dot{p}\\ \dot{r}\\ \dot{\Phi}\end{bmatrix}=\begin{bmatrix}-0.0528 & -0.1133 & 0.8380 & -0.0599\\ 45.3713 & -5.1260 & 4.2451 & -0.5651\\ -16.5618 & -0.3595 & -1.0507 & 0.0861\\ -0.2294 & 1.0258 & -0.0211 & 0.0028\end{bmatrix}\begin{bmatrix}\beta\\ p\\ r\\ \Phi\end{bmatrix}+\begin{bmatrix}0.8535 & -1.5509\\ 62.1884 & 24.4214\\ 2.4649 & -20.2276\\ -0.3132 & -0.1245\end{bmatrix}\begin{bmatrix}\delta_a\\ \delta_r\end{bmatrix} \tag{4.34}$$

$$Ay=\begin{bmatrix}-0.0565 & 0.0005 & -0.0042 & -0.0001\end{bmatrix}\begin{bmatrix}\beta\\ p\\ r\\ \Phi\end{bmatrix}+\begin{bmatrix}-0.0044 & -0.0229\end{bmatrix}\begin{bmatrix}\delta_a\\ \delta_r\end{bmatrix} \tag{4.35}$$

PID Method #2: BLS_1

NOTE: The BLS_1 method does not use measurements of Φ.

$$\begin{bmatrix}\beta\\ p\\ r\end{bmatrix}=\begin{bmatrix}-0.0302 & -0.1391 & 0.7184\\ 45.5843 & -5.3692 & 3.1168\\ -16.5943 & -0.3225 & -0.8788\end{bmatrix}\begin{bmatrix}\beta\\ p\\ r\end{bmatrix}+\begin{bmatrix}1.1188 & -2.0754\\ 64.6908 & 19.4748\\ 2.0841 & -19.4745\end{bmatrix}\begin{bmatrix}\delta_a\\ \delta_r\end{bmatrix} \tag{4.36}$$

$$Ay=\begin{bmatrix}-0.0565 & 0.0005 & -0.0044\end{bmatrix}\begin{bmatrix}\beta\\ p\\ r\end{bmatrix}+\begin{bmatrix}-0.0039 & -0.0239\end{bmatrix}\begin{bmatrix}\delta_a\\ \delta_r\end{bmatrix} \tag{4.37}$$

PID Method #3: FTR

NOTE: The FTR method uses measurements of Φ.

$$\begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} 1.2876 & -0.1449 & 1.0814 & -0.1078 \\ 52.0084 & -5.3617 & 7.9742 & -1.4111 \\ -17.9250 & -0.3459 & -0.8854 & 0.1128 \\ -0.5201 & 1.0536 & -0.0797 & 0.0141 \end{bmatrix} \begin{bmatrix} \beta \\ p \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} 1.1336 & -0.2903 \\ 62.0252 & 12.3251 \\ 2.2747 & -20.4663 \\ -0.6203 & -0.1233 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \quad (4.38)$$

$$Ay = \begin{bmatrix} -0.0571 & 0.0007 & -0.0045 & -0.0001 \end{bmatrix} \begin{bmatrix} \beta \\ p \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} -0.0077 & -0.0286 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \quad (4.39)$$

PID Method #4: LWR

NOTE: The LWR method uses measurements of Φ.

$$\begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} -1.3462 & 0.2277 & 0.3646 & 0.2081 \\ 12.7416 & -15.6300 & 26.4494 & -4.7652 \\ -11.9008 & -3.2008 & 3.3423 & -1.0534 \\ -0.1278 & 1.1564 & -0.2657 & 0.0481 \end{bmatrix} \begin{bmatrix} \beta \\ p \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} -3.2500 & 0.3439 \\ 163.5521 & -278.4519 \\ 35.0312 & 12.4106 \\ -1.6364 & 3.1458 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \quad (4.40)$$

$$Ay = \begin{bmatrix} -0.0402 & 0.0001 & -0.0027 & 0.0000 \end{bmatrix} \begin{bmatrix} \beta \\ p \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} 0.0011 & 0.4152 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \quad (4.41)$$

Note that the LWR lateral directional PID results were also found to be less accurate than the results from the other methods.

Using the above results in terms of state variable matrices, the PID performances for each of the different approaches were assessed in terms of differences between "actual" and "estimated" time histories. For this task the Simulink scheme shown below in Figure 4.2 was used. Note that the comparison was not performed for the LWR longitudinal dynamics.

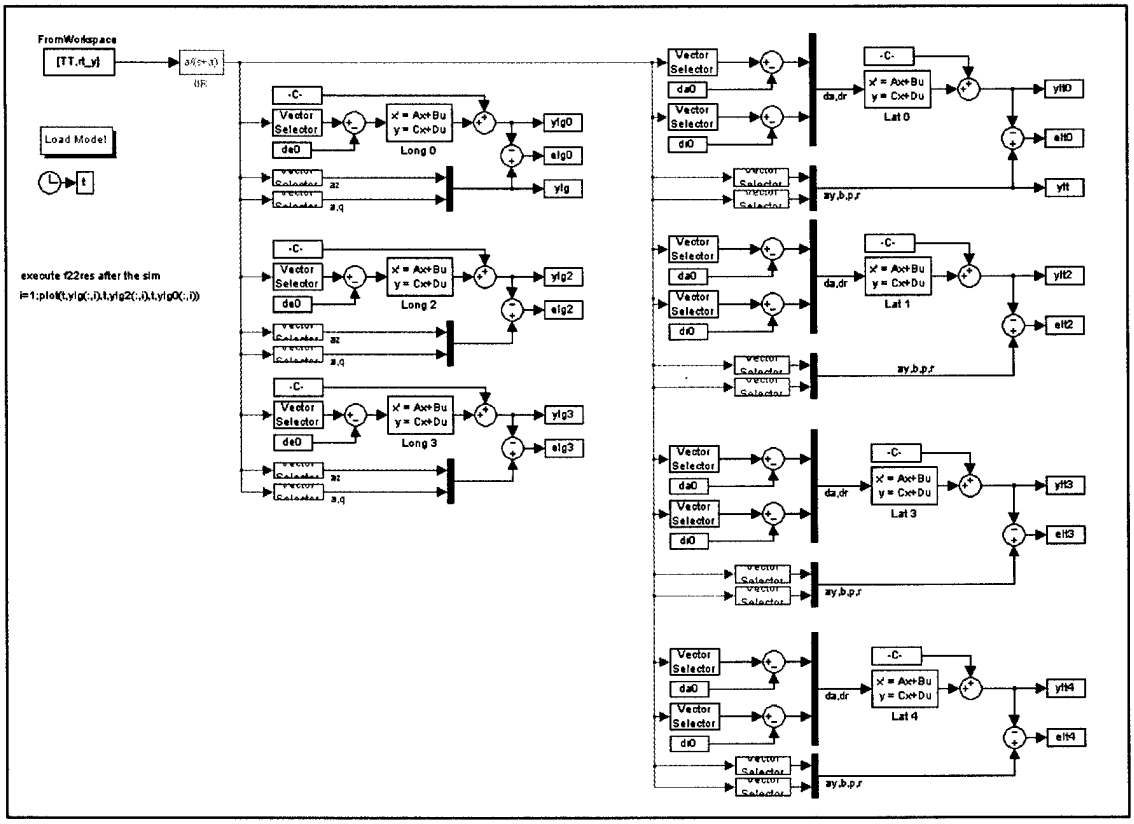


Figure 4.2 – Simulink block diagram for time histories comparison with different PID methods

The scheme above features longitudinal linear state variable models from the application of the BLS_0, BLS_1, and FTR methods along with lateral-directional state variable models from the application of the BLS_0, BLS_1, FTR and LWR methods. The same aircraft inputs (that is deflection of the control surfaces) were provided as inputs to each of the linear models with the relative outputs being compared with the actual outputs recorded from the flight test data. The results of this comparison phase - called '*validation phase*' - are shown in the Figures 4.3 – 4.16 below in terms of comparison of time histories for different longitudinal and lateral directional parameters for given PID input-maneuvers (for elevators, rudders, and ailerons). Note that the LWR results were not plotted for the longitudinal PID.

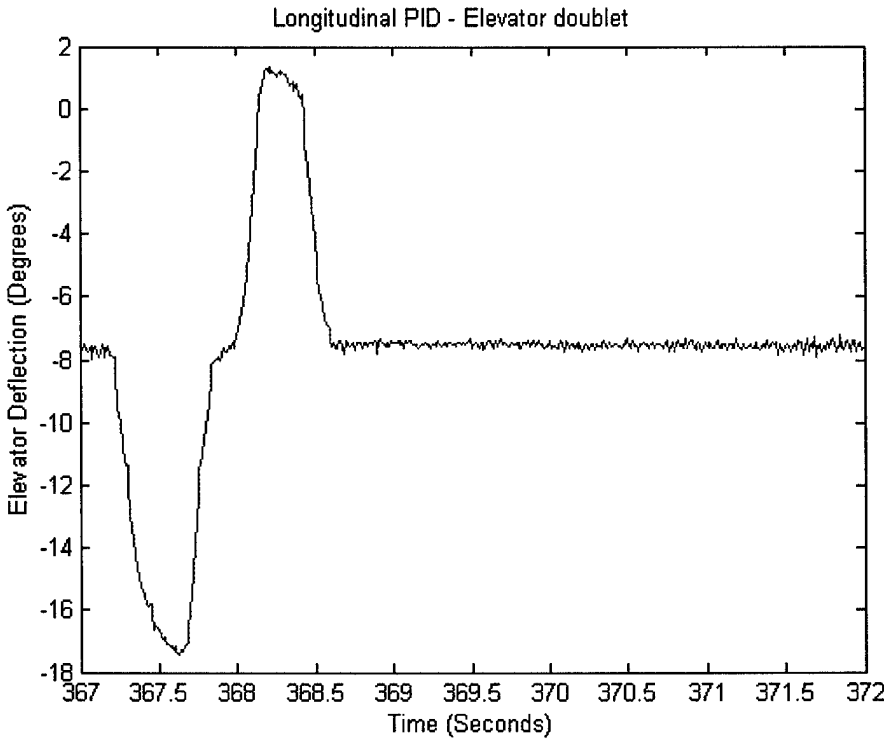


Figure 4.3 – Longitudinal PID input

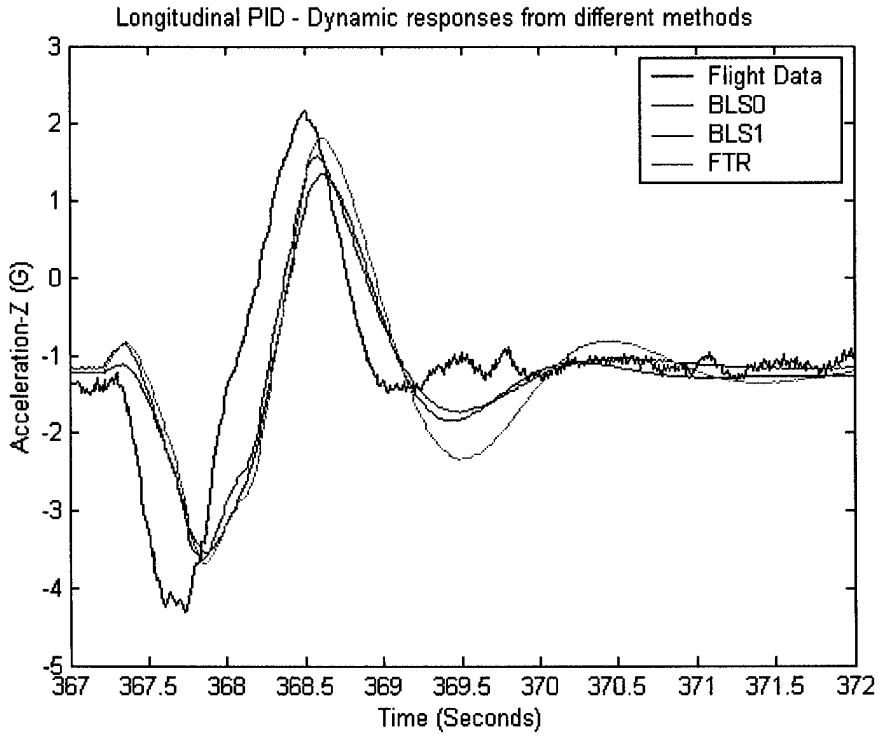


Figure 4.4 – Comparison of 'a_z' responses

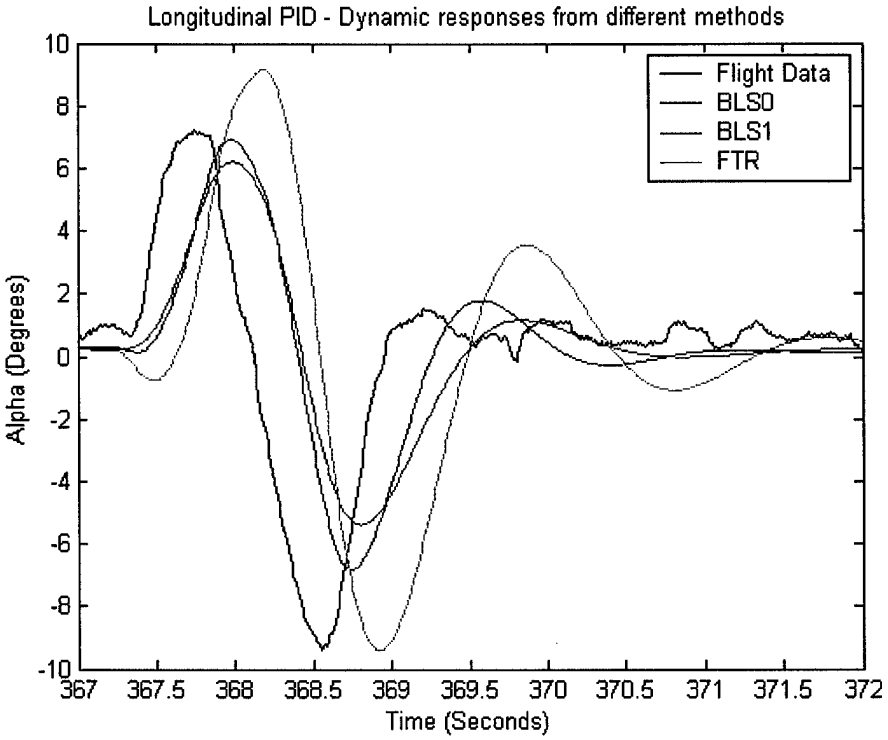


Figure 4.5 – Comparison of 'α' responses

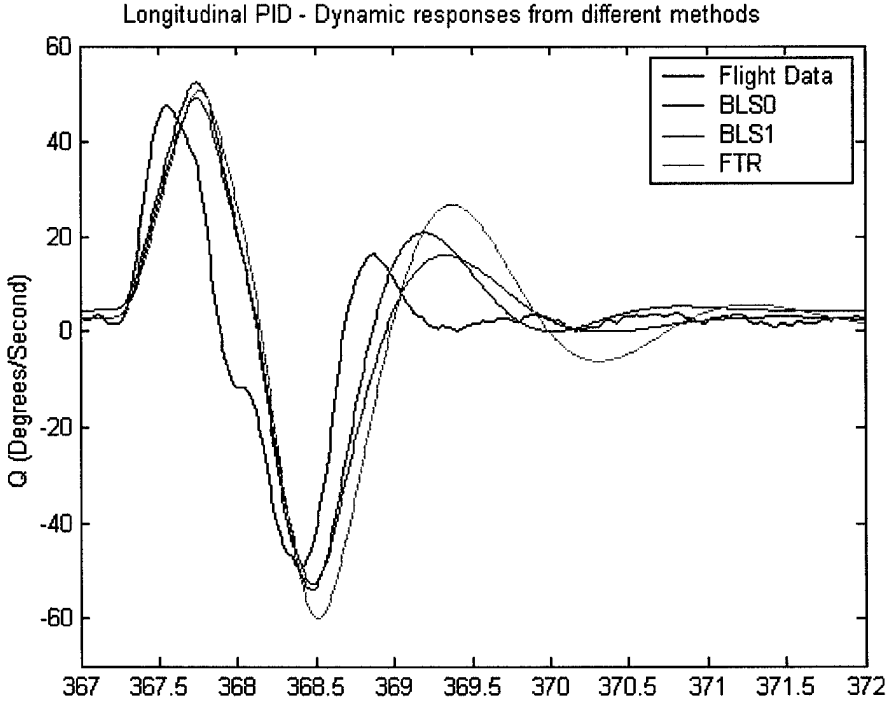


Figure 4.6 – Comparison of 'q'responses

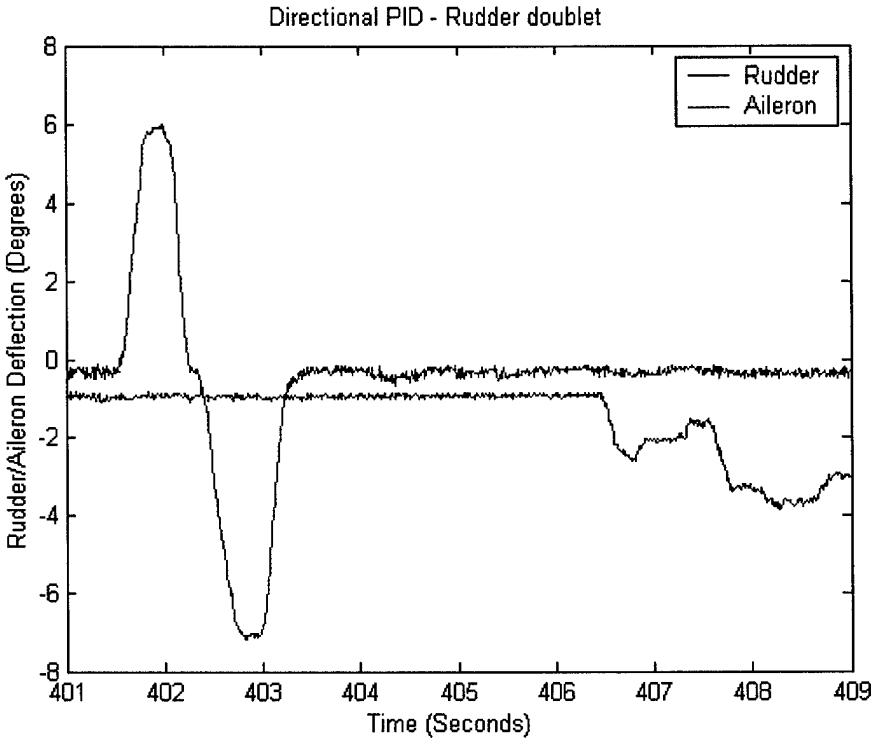


Figure 4.7 – Directional PID input

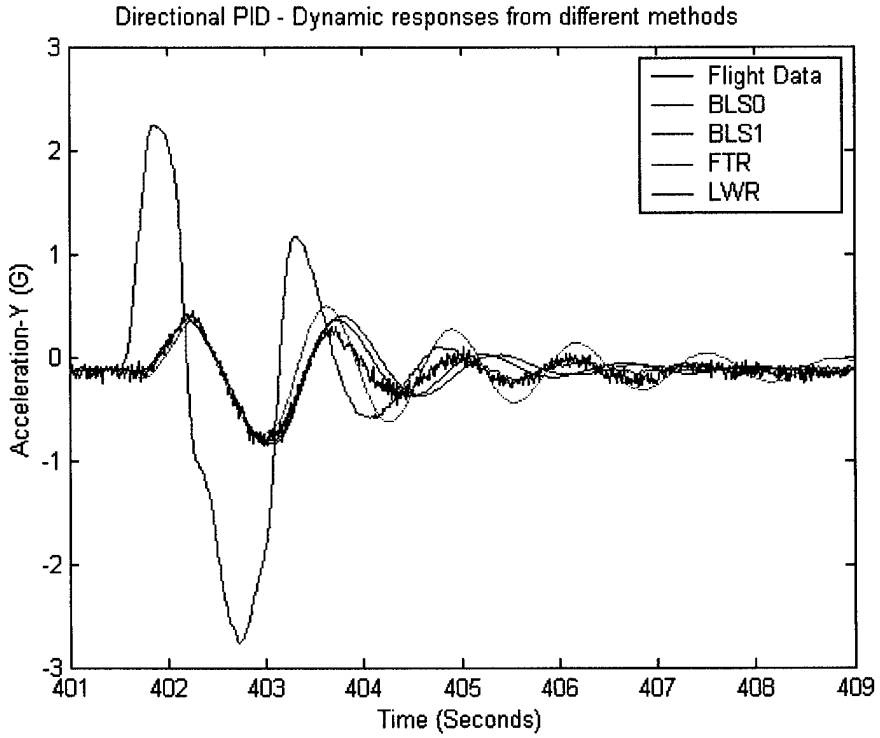


Figure 4.8 – Comparison of 'a_y'responses

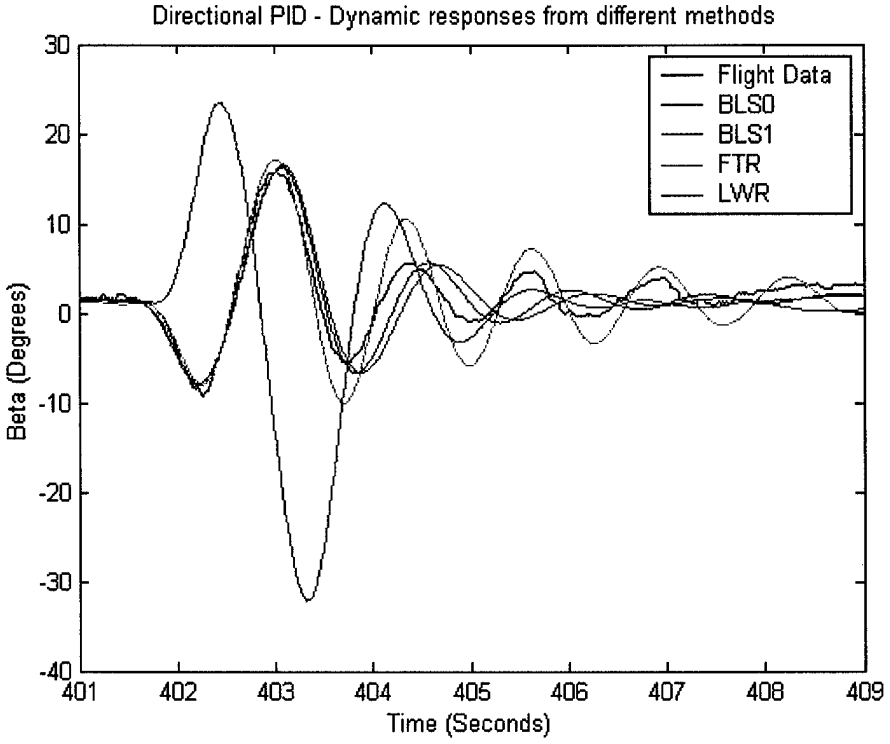


Figure 4.9 – Comparison of 'β'responses

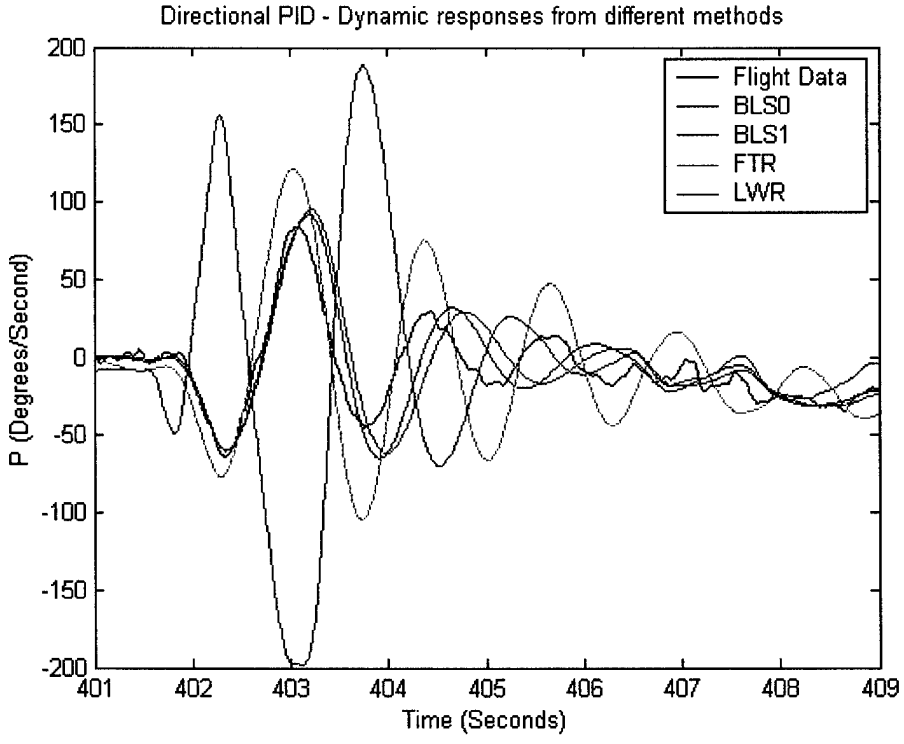


Figure 4.10 – Comparison of 'p'responses

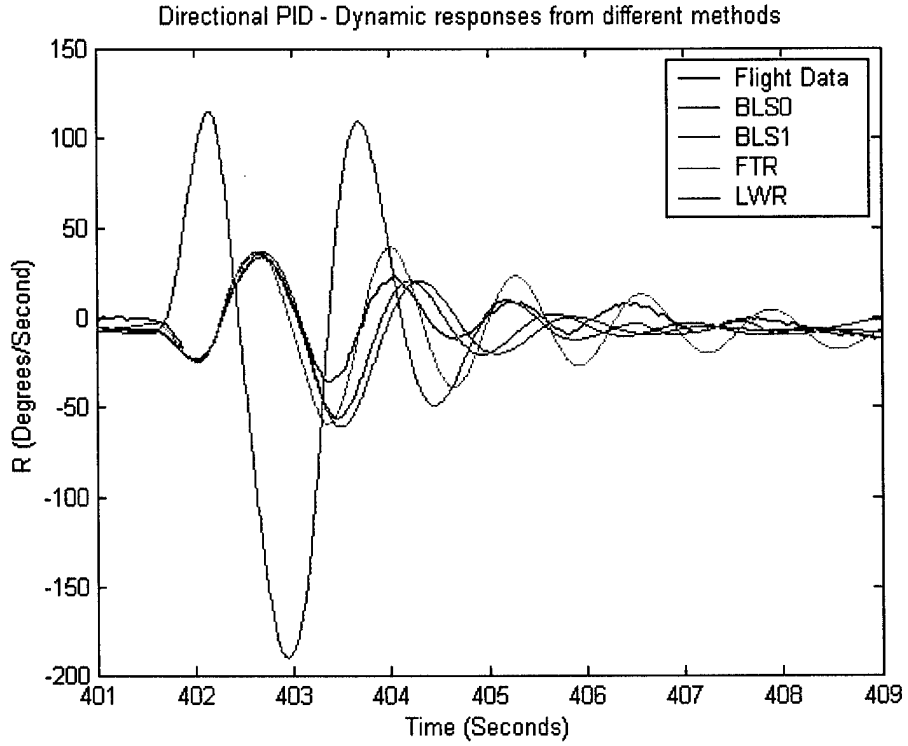


Figure 4.11 – Comparison of 'r'responses

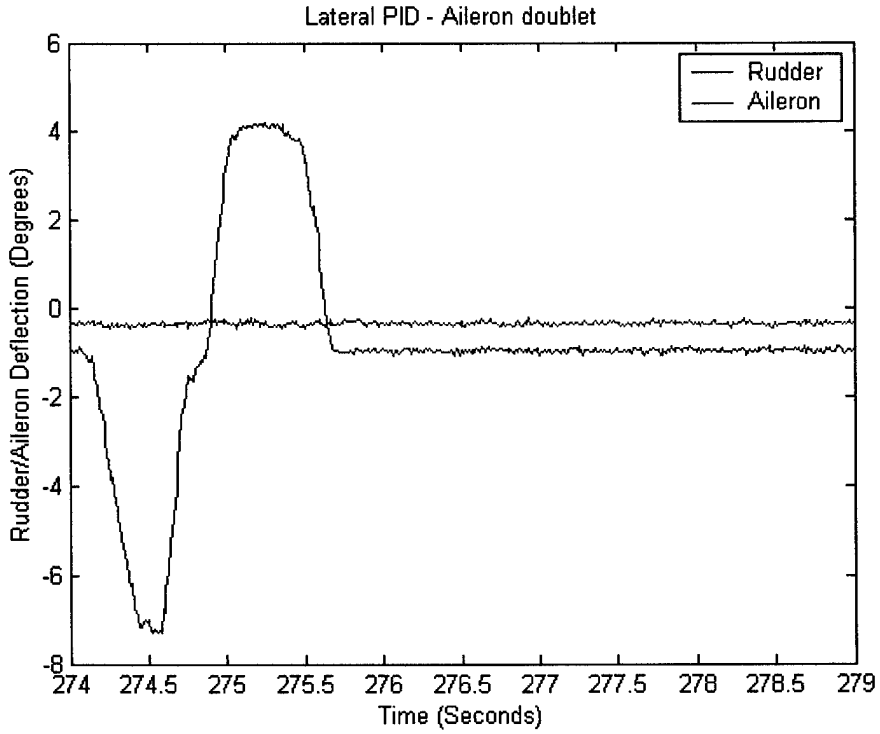


Figure 4.12 – Lateral PID input

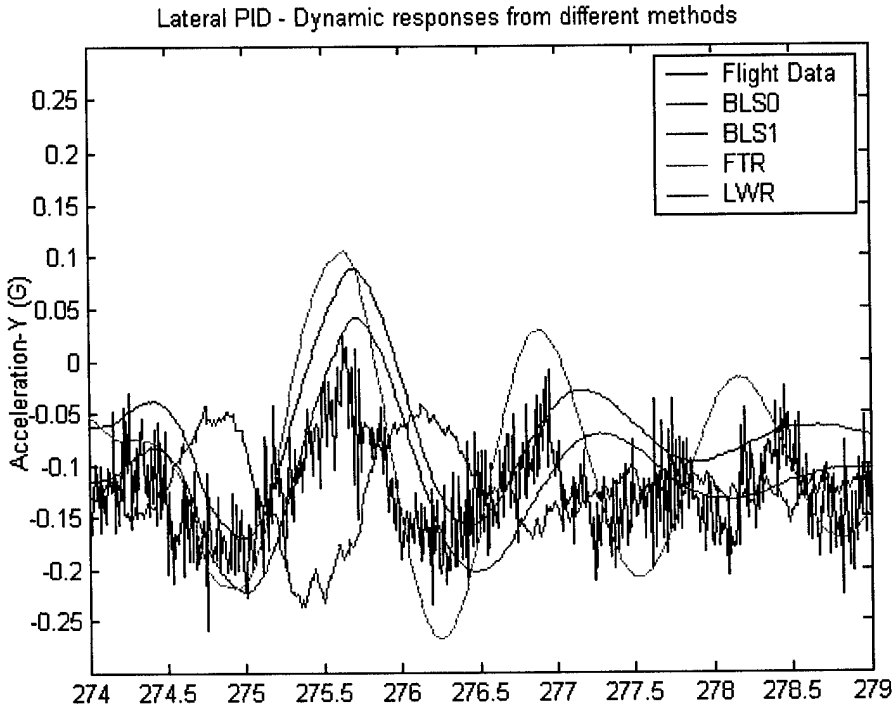


Figure 4.13 – Comparison of 'a_y' responses

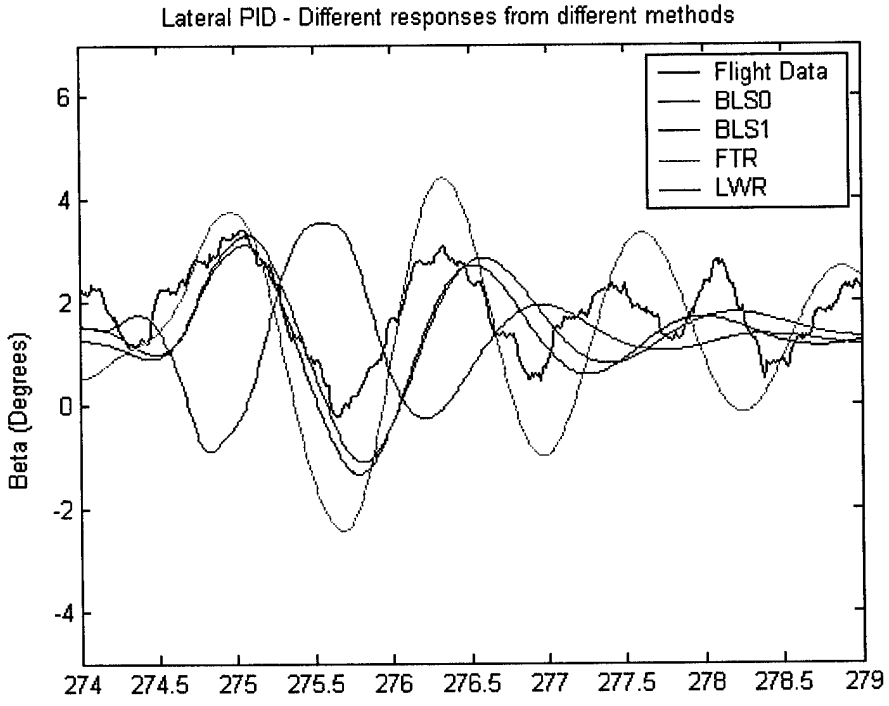


Figure 4.14 – Comparison of 'β' responses

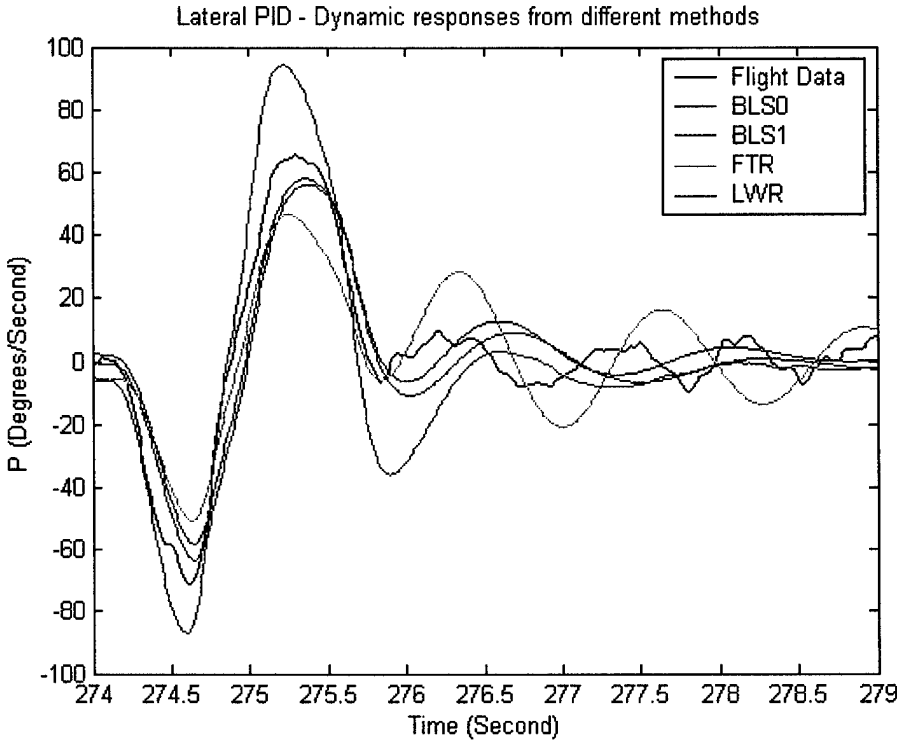


Figure 4.15 – Comparison of 'p' responses

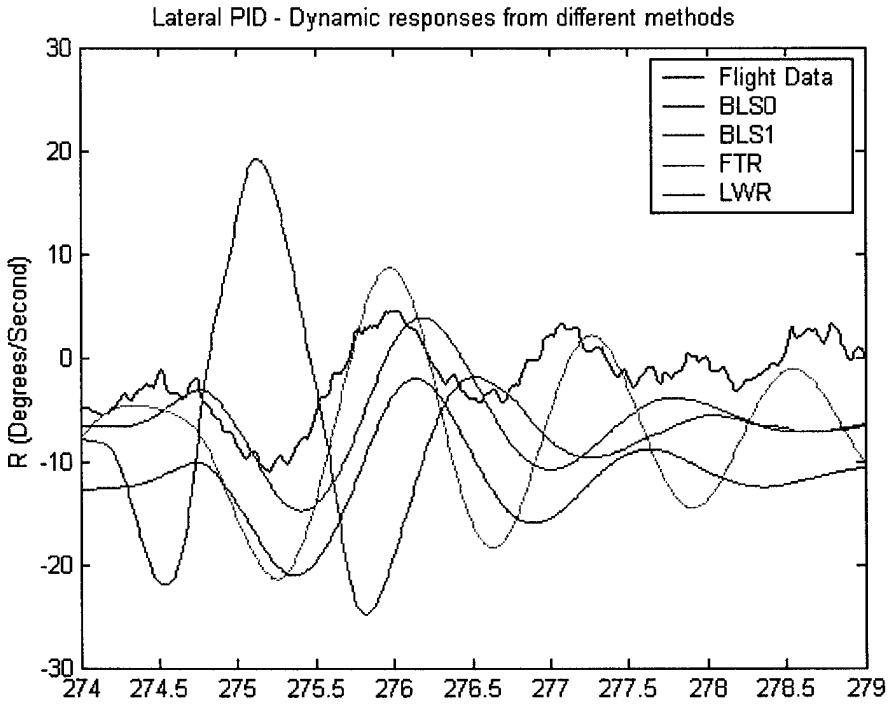


Figure 4.16 – Comparison of 'r' responses

The next phase of the PID analysis consisted in the statistical analysis of the difference (defined as '*error*') between the estimated responses and the actual flight data (shown in Figures 4.4-4.6, Figures 4.8-4.11, and Figures 4.13-4.16). The results of this statistical analysis have been summarized in the Tables 4.1 and 4.2 below. Once again, the LWR results for the longitudinal PID study have not been included.

Error Statistics	BLS_0	BLS_1	FTR	LWR
a_z Mean	0.1609	0.1046	0.1090	-
a_z Standard Deviation	0.3664	0.3746	0.3978	-
α Mean	0.5200	0.5189	0.5225	-
α Standard Deviation	1.2012	1.2465	1.8340	-
q Mean	-2.3162	-1.1427	-1.1438	-
q Standard Deviation	6.0219	6.8101	7.9220	-

Table 4.1 - Statistical analysis of the longitudinal PID results

Error Statistics	BLS_0	BLS_1	FTR	LWR
a_y Mean	-0.0212	-0.0042	-0.0122	0.0026
a_y Standard Deviation	0.0762	0.0727	0.0903	0.2678
β Mean	0.5315	0.4695	0.5166	0.6719
β Standard Deviation	1.0969	1.1634	1.4369	4.2687
p Mean	-0.8158	0.4606	-0.5650	-0.4063
p Standard Deviation	10.6370	11.8932	13.8609	33.4578
r Mean	2.6997	0.5134	1.2415	0.5046
r Standard Deviation	6.9107	6.8191	6.9920	22.9703

Table 4.2 - Statistical analysis of the lateral-directional PID results

The data in the above tables can be used to calculate the RMS (Root Mean Square) values of the error for both the lateral and longitudinal variables for the models determined from the application of the different PID methods. The following table displays in percentage the RMS of the error divided by the range of variation of for each variable.

Variable	BLS_0	BLS_1	FTR	LWR
a_z	6.19%	6.01%	6.38%	-
α	6.90%	7.12%	10.05%	-
q	6.59%	7.06%	8.18%	-
a_y	5.63%	5.18%	6.49%	19.07%
β	4.56%	4.69%	5.71%	16.17%
p	5.02%	5.60%	6.53%	15.76%
r	9.31%	8.58%	8.91%	28.84%

Table 4.3 – PID error percentage for the different parameters

From an evaluation of the results in the above table it can be said that the accuracy of the PID the quality of the estimation is in general moderately good with the BLS_1 PID method providing the best results. The yaw rate estimation shows the largest

error, and this is probably due to low input excitation for the directional PID compared to the longitudinal and lateral PID. Also it can be noticed that the accuracy of the results of the FTR method – which is suitable for on-line real time applications – is somewhat comparable to the accuracy of the both BLS PID methods – used for batch off-line applications - with the LWR PID method providing by far the worst performance.

Although the PID analysis was outside the main technical scope of this project, the PID study provided a fairly accurate state variable model of the WVU YF-22 aircraft as a starting point for the design of the neural network fault tolerant control laws described in the next paragraphs.

4.2 – Description of the Fault Tolerant Schemes

According to the research objectives, a fault tolerant flight control system needs to perform:

- sensor failure detection, identification, and accommodation (SFDIA);
- actuator failure detection, identification, and accommodation (AFDIA).

Furthermore, the SFDIA task can be divided into:

- sensor failure detection, identification (SFDI), which monitors the degree of deterioration in the accuracy of the sensors.
- sensor failure accommodation (SFA), which replaces the faulty sensor with an appropriate estimation.

Similarly, the AFDIA task can be divided into:

- actuator failure detection and identification (AFDI), which detects significant abnormalities and searches for the cause or for a set of probable causes;
- actuator failure accommodation (AFA), which determines on-line what actions should be taken to recover the impaired aircraft.

Sensor failure detection and identification (SFDI) is critical when the measurements from a failed sensor are used in the feedback loop of a control law. Since the aircraft control laws need sensor feedback to set the current dynamic state of the airplane, even slight sensor inaccuracies, if left undetected and un-accommodated for, can lead to closed-loop instability, potentially leading to unrecoverable flight conditions. For SFA purposes, most of today's high performance military aircraft as well as commercial jetliners implement a triple physical redundancy in their sensor capabilities. However, when reduced complexity, lower cost, and weight optimization are of concern, an analytical sensor redundancy approach is appealing.

In terms of the AFDIA problem, an actuator failure may imply a locked surface, a missing part of the control surface, or a combination of both. In increasing order of severity, an actuator failure modifies trim conditions and induces a dynamic coupling followed by deterioration of dynamic stability potentially leading to unrecoverable flight conditions. The final objective of the AFDIA is to achieve, in increasing order of importance, a lower failure-induced handling qualities degradation rate, a lower mission abort rate, and a lower aircraft loss rate. In recent years neural networks have been proposed for identification and control of linear and non-linear dynamic systems. The code developed by the WVU team features a SFDIA scheme based on on-line learning neural networks along with an AFDIA scheme with an approach combining on-line learning Multi-Layer Perceptron (MLP) neural networks and cross-correlation analysis.

The code also features a logic allowing distinguishing between the occurrence of a sensor failure or an actuator failure. The problem of the integration between the SFDIA and the AFDIA schemes is actually a very original topic in the fault tolerance literature.

The following properties of Multi Layer Perceptron Neural Networks (MLP NNs -or- NNs) are critical for the design of these SFDIA and AFDIA schemes:

- *Learning and adaptation*: NNs can be trained using past recorded or simulated data (off-line training) or current data (on-line learning).
- *Applicability to non-linear systems*: The applicability of NNs to non-linear systems originates from their demonstrated mapping capabilities.
- *Application to multivariable systems*: NNs are multi-input, multi-output (MIMO) entities and this, naturally, leads to their application to multivariable systems.
- *Parallel distributed processing and hardware implementation*: NNs have an inherent parallel architecture, which, naturally, leads to high-speed parallel hardware implementations.

To date, for MLP NNs the Back-Propagation algorithm (BPA), a gradient-based optimization method, has been widely used as a training algorithm for the NN architecture. However, limited learning speed and local minimum points are well-known drawbacks of the BPA. Different researchers have proposed several algorithms and/or procedures for dealing with these problems. The approach here used consists in using a heterogeneous network, meaning that each neuron in the hidden and output layers of the NN has the capability of updating the output range (upper and lower bounds U, L) and the slope of the sigmoid activation function (T) as given:

$$f(x,U,L,T)=\frac{U-L}{1+e^{-x/T}}+L \tag{4.42}$$

where 'x' is the same argument as in the BPA sigmoid activation function. The relative learning algorithm has been named the Extended Back-Propagation algorithm (EBPA) and has demonstrated substantial performance improvements with respect to the BPA in terms of accuracy and learning speed (Napolitano1995). It should be emphasized that different neural paradigms and/or other algorithms could be used (and have been used at WVU) within the SFDIA and the AFDIA schemes.

The next sub-sections review the SFDIA and AFDIA schemes.

4.2.1- Neural Network-Based Sensor Failure Detection, Identification, and Accommodation (SFDIA)

Using on-line learning NN estimators, the SFDIA problem can be approached by introducing multiple feed-forward NNs trained on-line with the EBPA. Particularly, the scheme consists of a main NN (MNN) and a set of *n* decentralized NNs (DNNs), where *n* is the number of the sensors in the flight control system without physical redundancy. The outputs of the MNN are the estimates of the same parameters measured by the *n* sensors at time *'k'*, using measurements from time instant *'k-1'* to *'k-m'*; these estimates are compared with the actual measurements at time *'k'*. For the i-th of the *n* DNNs, the output is the estimate of the measurement of the i-th sensor, that is, the prediction of the state at time *'k'*, using measurements from *'k-1'* to *'k-m'* to be compared with the actual

measurement at time *'k'*. The inputs to the i-th DNN are the measurements from any number to up *'n-1'* sensors, in other words, all the *n* sensors excluding the i-th sensor.

For SFD purposes, when a quadratic estimation error parameter from the MNN exceeds some predefined threshold at a certain time instant, the scheme deduces that a sensor failure may be occurring or has already occurred. Following the positive sensor failure detection, the learning for each DNN is halted; then, a quadratic estimation error parameter from the DNNs exceeding, at the same time instant, another threshold provides the identification. For the accommodation phase, the i-th DNN output is used to replace the measurement from the faulty sensor; i-th DNN output is also used as input to the MNN for the purpose of allowing the MNN to provide detection capabilities until the end of the flight. This output is also passed to all other DNNs using the i-th sensor as an input parameter. This 'double trigger' approach using both MNN and DNNs has the purpose of reducing the rate of false alarms in the FDI process. Several options can be added to this scheme to add robustness for noisy measurements and/or intermittent sensor failures. For example, a lower and a higher threshold level can be introduced for the DNNs. If the estimation error for the i-th DNN exceeds the lower threshold once, the status of the corresponding i-th sensor is declared "suspect" and the numerical architecture of the i-th DNN is not updated. Should this status continue for a certain number of time instants and/or the estimation error in successive time instants exceeds the higher threshold, then the sensor is declared failed and is, therefore, replaced by the output from the i-th DNN. Figure 1 shows a block diagram of the SFDIA process.

The code features failures for the pitch rate, the roll rate, and the yaw rate gyros. The approach can be extended to handle failures to other sensors without any loss of generality. As for any other FDIA approach, the following distinct capabilities are critical:

- failure detectability and false alarm rate for SFDI purposes;
- estimation error for SFA purposes.

Two distinct parameters are used for sensor failure detection (SFD) purposes: MQEE and OQEE. The first SFD parameter, MQEE, is defined as:

$$MQEE(k) = \frac{1}{2}\sum_{i=1}^{Num.of\,DNNs}(Y_i(k)-O_{i,MNN}(k))^2 \qquad (4.43)$$

$$= \frac{1}{2}[(p(k)-\hat{p}_{MNN}(k))^2+(q(k)-\hat{q}_{MNN}(k))^2+(r(k)-\hat{r}_{MNN}(k))^2]$$

The sensor failure identification (SFI) can instead be achieved by monitoring the absolute value of the estimation error of each DNN, defined as:

$$DQEE_x(k)=\frac{1}{2}(x(k)-\hat{x}(k))^2 \quad \text{where x = p, q, and r} \qquad (4.44)$$

For sensor failure accommodation (SFA) purposes, the following classic parameters for the estimation error are instead evaluated:

$$DAEE_x = \frac{1}{N}\sum_{k=1}^{N}(x(k)-\hat{x}(k)) \tag{4.45}$$

$$DVEE_x = \frac{1}{N}\sum_{k=1}^{N}[(x(k)-\hat{x}(k))-DAEE_x]^2 \quad \text{where x = p, q, and r} \tag{4.46}$$

where DAEE and DVEE represent the DNN estimation error mean and variance respectively. The 'N' refers to the number of time steps from the instant when failure of sensor is declared to the end of the simulation.

In the case of a step-type sensor failure, the desirable detection capabilities are provided by the peak value of the MQEE parameter regardless of bias magnitude. However, in the case of soft sensor failures, particularly for ramp-type sensor failures, the MQEE-criterion has not shown reliable detection capabilities. Therefore, an additional parameter has been introduced to ensure desirable performance under conditions resulting from any type of failures. The second SFD parameter, OQEE, is defined by:

$$OQEE(K) = \frac{1}{2}\sum_{i=1}^{Num.of\,DNNs}(O_{i,MNN}(k)-O_{i,DNN}(k))^2 \tag{4.47}$$

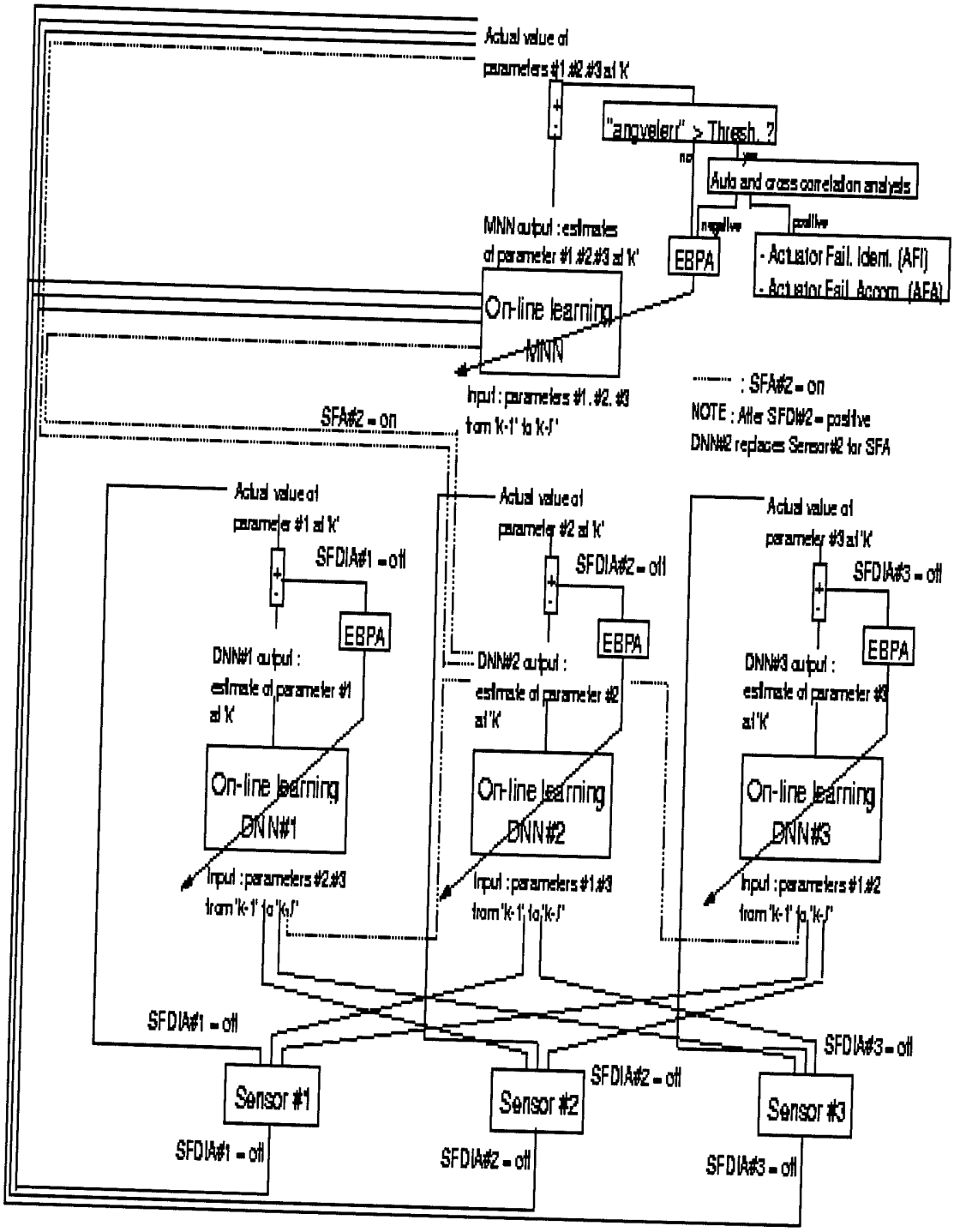
$$= \frac{1}{2}[(\hat{p}_{MNN}(k)-\hat{p}_{DNN}(k))^2+(\hat{q}_{MNN}(k)-\hat{q}_{DNN}(k))^2+(\hat{r}_{MNN}(k)-\hat{r}_{DNN}(k))^2]$$

where $\hat{p}_{DNN}$, $\hat{q}_{DNN}$, and $\hat{r}_{DNN}$ are the estimates of p, q, and r from the respective DNNs.

The need for the use of the parameter OQEE for SFD purposes is better described by an analysis of a typical sensor failure. For example, when a pitch rate gyro fails, the three parameters q(k), $\hat{q}_{MNN}(k)$, and $\hat{q}_{DNN}(k)$ within the 'MQEE' and 'OQEE' are considered. At nominal conditions $\hat{q}_{MNN}(k)$ and $\hat{q}_{DNN}(k)$ are estimated by the MNN and q-DNN respectively to emulate the actual pitch rate gyro. In the event of a ramp-type q-gyro failure, the $\hat{q}_{MNN}$ tends to resemble the corrupted signals from the gyro. In fact, since the pitch rate gyro is typically included as the input parameter in MNN, the MNN architecture is updated with the failed q-gyro values during the on-line learning. As a result, MQEE does not provide an accurate detection because the difference between q(k) and $\hat{q}_{MNN}(k)$ is relatively small despite the gyro failure. However, the output of the q-DNN ($\hat{q}_{DNN}(k)$) in Eq.(4.47) follows the nominal 'q' value (that is the value as it would be without a failure) relatively well despite the sensor failure. This is because the q-DNN does not receive, as input data, measurements from the faulty sensor. The discrepancy between $\hat{q}_{MNN}$ and $\hat{q}_{DNN}$ in Eq.(4.47) causes, therefore, the peak of 'OQEE'. This problem does not occur for the step-type sensor failures. In fact, for these failures $\hat{q}_{MNN}(k)$ is not consistent with the measurement from the failed sensor, which induces therefore a peak for MQEE.

In general, it can be concluded that the MQEE provides better performance for step-type sensor failures whereas the OQEE performs better for ramp-type of sensor failures. The SFDIA Matlab code features a combined detection scheme using both 'MQEE' and 'OQEE'. This provides more reliable detection capability for the SFDIA for any type of failures.

A block diagram of the SFDIA scheme is shown in Figure 4.17 (Napolitano2000); a more detailed flow chart is given in Figure 4.18 (Napolitano2000). Figure 4.18 shows also the interaction with the AFDIA scheme described in the next paragraph.



NOTE : For simplicity purposes, only 3 sensors are considered.

NOTE: Figure modified from 'Portrait' format

Figure 4.17 – General Block Diagram of the SFDIA Scheme

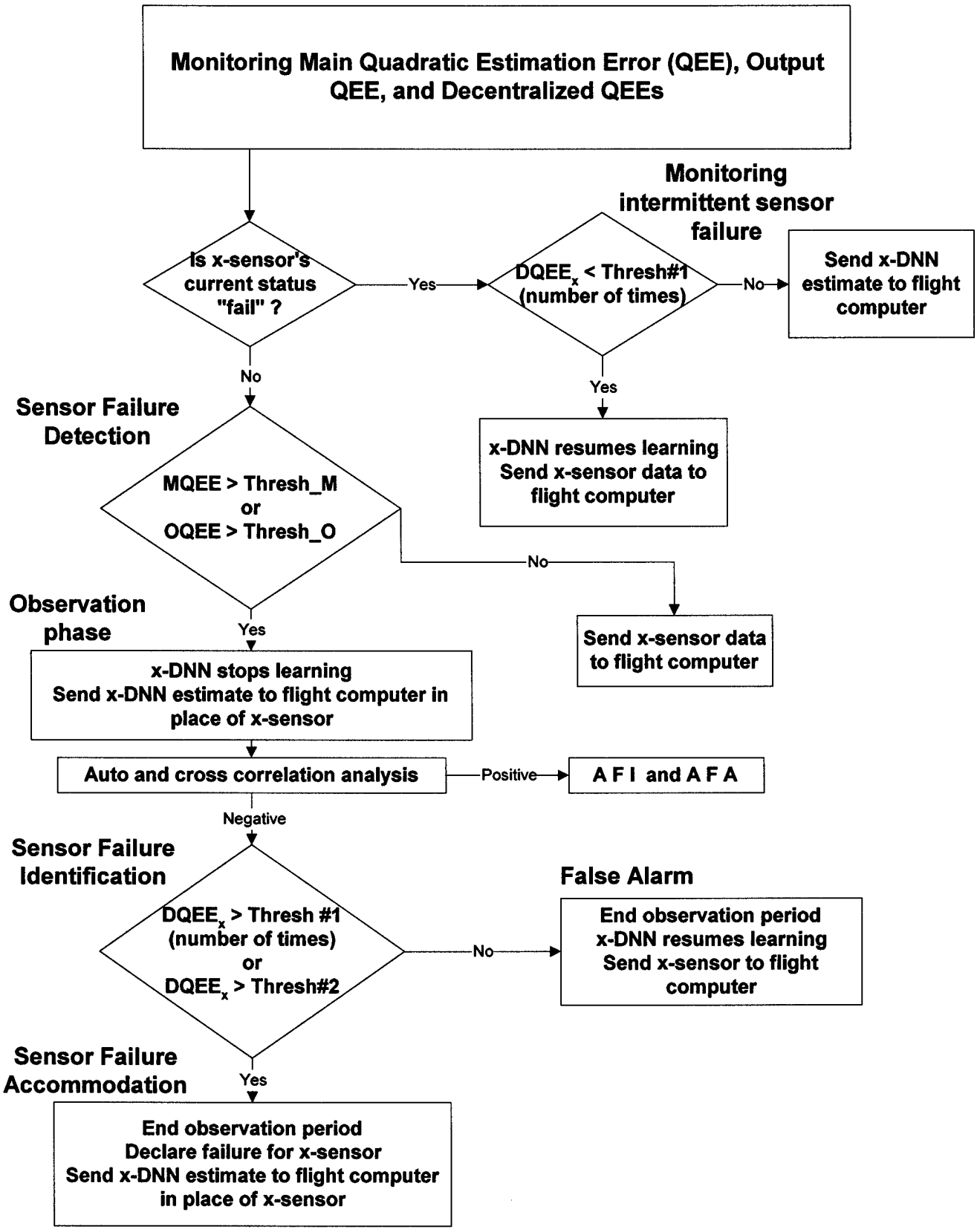


Figure 4.18 – Functional flow chart of the SFDIA scheme

4.2.2. - Neural network and cross/auto correlation-based Actuator Failure Detection, Identification, and Accommodation (AFDIA)

The occurrence of any actuator failure also implies that the parameter 'MQEE' defined above exceeds a selected threshold. Thus, the actuator failure detection (AFD) can be achieved by spotting substantial changes in the aircraft angular velocities following any type of actuator failure. Next, the actuator failure identification (AFI) can be performed by analyzing specific cross-correlation functions. In general, for two random processes Y(k) and X(k), the cross-correlation and the autocorrelation functions are defined by:

$$R_{YX}(n)=E[Y(k)X(k+n)], R_{XX}=E[X(k)X(k+n)] \tag{4.48}$$

For AFI purposes different sets of cross-correlation functions can be used, taking advantage of the fact that any type of actuator failure on any aircraft control surface involves a loss of symmetry. A dynamic coupling between longitudinal and lateral-directional aircraft dynamics then follows this loss of symmetry. The use of cross-correlation functions relative to the angular velocities (R_{pq}, R_{pr}, R_{qr}) and to the Euler's angles ($R_{\Theta\Phi}, R_{\Phi\Psi}, R_{\Theta\Psi}$) has been tested with success. Furthermore, different auto-correlation functions, in particular $R_{\Theta\Theta}, R_{\Phi\Phi}$, have shown capabilities as identification tools. It should be underlined that for each of these cross and auto-correlation functions the selection of detection thresholds is very critical for a desirable trade-off between false alarm and failure detectability.

Following a positive AFD and AFI, the immediate objective is to regain equilibrium and to compensate for the pitching, rolling, and yawing moments induced by the failure. Toward this goal, three separate NN controllers are introduced: a NN pitch controller, a NN roll controller, and a NN yaw controller ([Napolitano1995]).

The output of the NN pitch controller (NNC pitch) is the compensating deflection for the remaining healthy elevator (elevator failure case) or the symmetric elevators (aileron and/or rudder failure case). The on-line learning for the NN pitch controller is initiated as the simulation starts. Under nominal conditions, the controller is trained to emulate the actual control deflections for the symmetric elevators. Therefore it minimizes the cost function:

$$J_{pitch_{nom}}=(\delta_{H_{L,R}}-\hat{\delta}_{H_{L,R}}) \tag{4.49}$$

Following a positive AFD and AFI showing the need for a longitudinal AFA, the on-line learning NN pitch controller switches its target to minimize the cost function:

$$J_{pitch_{AFA}}=k_{Long\_1}(q-q_{ref})+k_{Long\_2}(\theta-\theta_{ref})+k_{Long\_3}(a_n-a_{n_{ref}}) \tag{4.50}$$

where $q_{ref}=0$, $\theta_{ref}=\theta_{trim}$, and $a_{n_{ref}}=1$.

It is interesting to note that this cost function resembles a controller with a PID error formulation. It should also be mentioned that the on-line learning at nominal

conditions (that is with the AFDIA scheme inactive) has no physical meaning; in fact, the NN pitch controller is just "emulating" control deflections at nominal conditions. However, this procedure has shown the benefit of improving the transient response by having the NN output within the same order of magnitude of the NN output necessary when the AFA process is turned on following a positive AFD and AFI.

Within the AFDIA scheme the NN roll and yaw controllers operate in a similar fashion. Under nominal conditions, these controllers learn to replicate the actual control deflections for the ailerons and the rudder by minimizing the cost functions:

$$J_{roll_{nom}} = (\delta_{A_{L,R}} - \hat{\delta}_{A_{L,R}}) \tag{4.51}$$

$$J_{yaw_{nom}} = (\delta_R - \hat{\delta}_R) \tag{4.52}$$

Following positive AFD and AFI, the on-line learning NN roll and yaw controllers switch their targets to minimize the cost functions:

$$J_{roll_{AFA}} = k_{Lat\_1}(p - p_{ref}) + k_{Lat\_2}(\phi - \phi_{ref}) \tag{4.53}$$

$$J_{yaw_{AFA}} = k_{Dir\_1}(r - r_{ref}) + k_{Dir\_2}(\psi - \psi_{ref}) \tag{4.54}$$

where the $p_{ref} = \phi_{ref} = r_{ref} = 0$ at trim conditions.

Note that different cost functions could be selected at post-failure conditions. Also, it should be emphasized that time-varying coefficients $k_{long\_i's}$, $k_{lat\_i's}$, $k_{dir\_i's}$ could be introduced for an adaptive formulation of the cost functions at post-failure conditions related to the outcome of the EBPA computations. The complexity of the neural algorithms would then substantially increase.

Figure 4.19 shows a block diagram of the AFDIA process while an overall functional block diagram of the SFDIA/AFDIA software is shown in Figure 4.20.

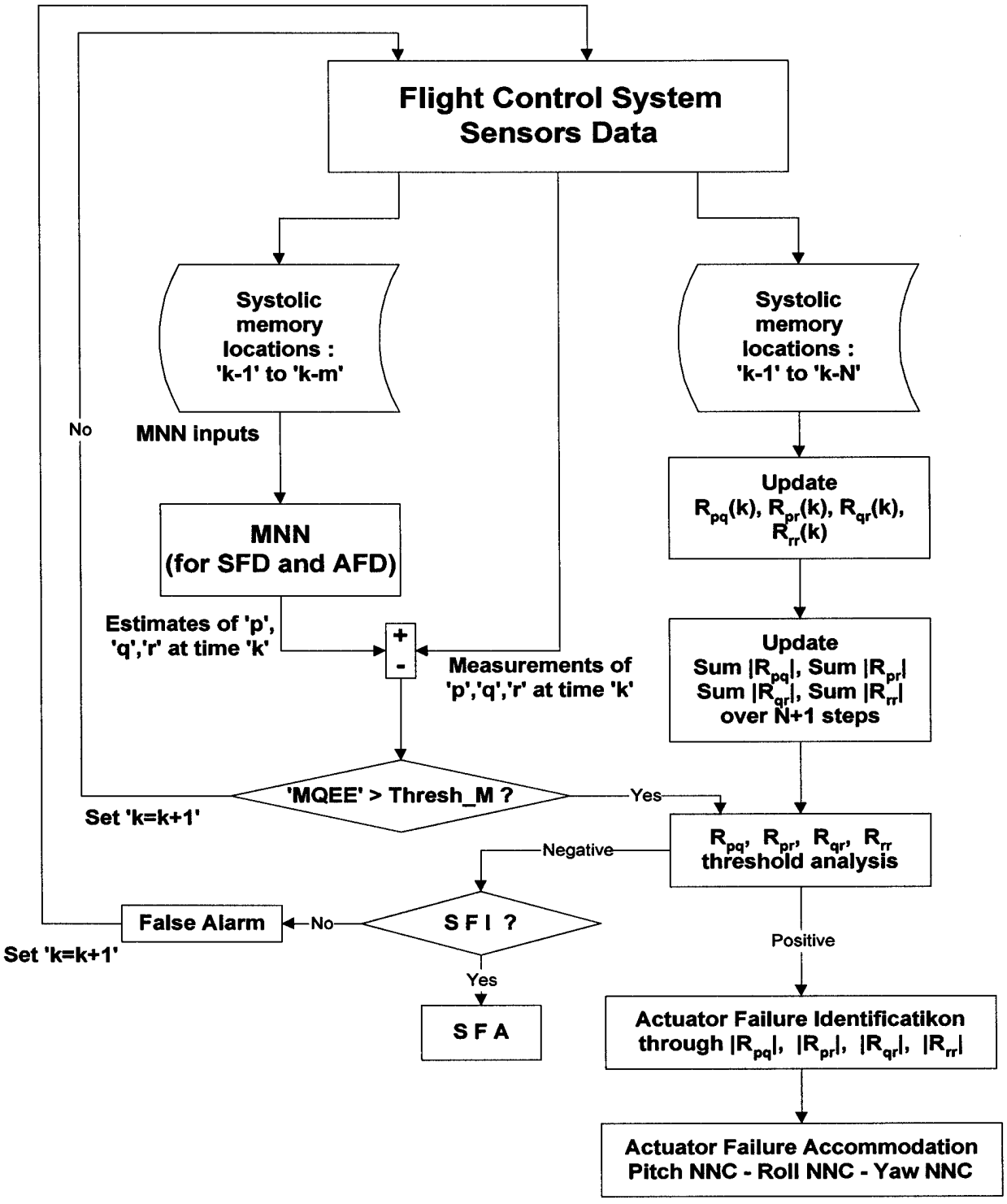


Figure 4.19 – Block Diagram of the AFDIA Scheme

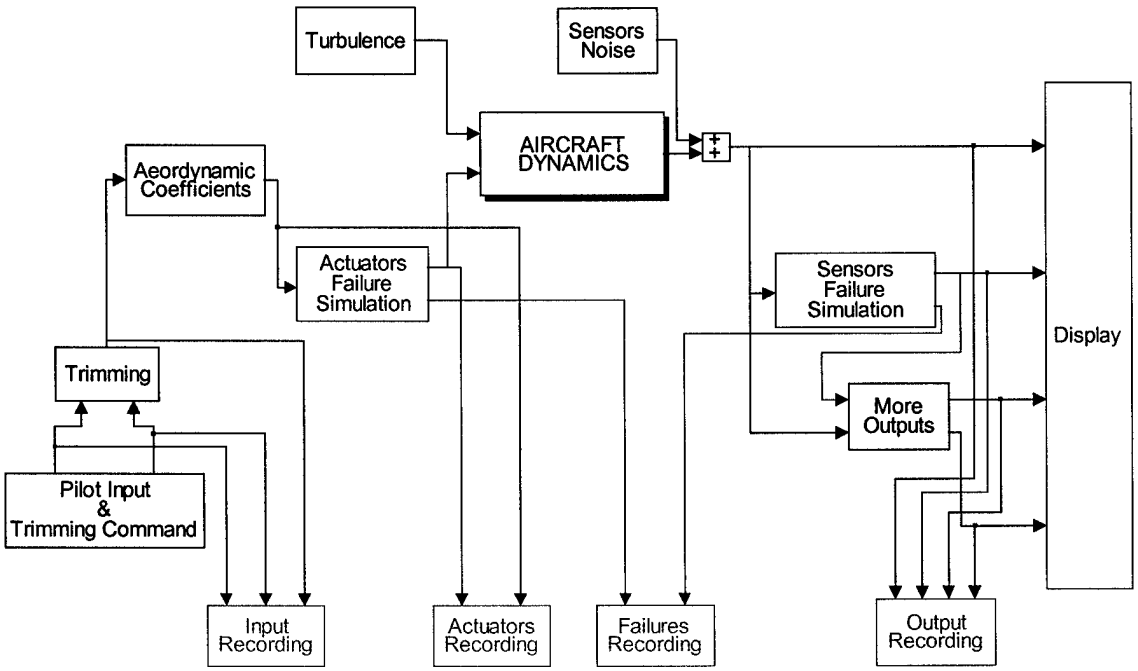


Figure 4.20 – Functional Block Diagram of the SFDIA/AFDIA Simulation Software

4.2.3 - Mathematical modeling of the SFDIA and the AFDIA conditions

As stated in a previous section, the SFDIA scheme is simulated for failures of the pitch, roll, and yaw rate gyros. Therefore, the NN-based SFDIA scheme uses one main NN (MNN) with three decentralized NNs (DNNs). Note that input parameters of q-DNN do not include pitch rate measurements. Similarly, the p-DNN and the r-DNN do not use 'p' and 'r' as input parameters, respectively.

A generic sensor failure for the parameter x_i can be modeled as:

$$x_{failure,i} = x_{nom,i} + \rho n_i \tag{4.55}$$

where n_i is the direction vector for the *i-th* faulty sensor, and ρ is the magnitude of the failure which can be positive or negative :

- for step-type sensor failures, $n_i = 1$;
- for ramp-type sensor failures, $n_i = \dfrac{t - t_{f1}}{t_{f2} - t_{f1}}$ for $\left(t_{f1} \le t \le t_{f2}\right)$, $n_i = 1$ for $\left(t \ge t_{f2}\right)$

where t_{f1} and t_{f2} indicate the initial and final time instant of ramp-type sensor failure, respectively. The different types of sensor failures considered in this study based on the above formula are represented by:

Type #1 - large instantaneous bias;

Type #2 - small instantaneous bias;
Type #3 - large drifting bias with slow transient period;
Type #4 - large drifting bias with fast transient period;
Type #5 - small drifting bias with fast transient period;
Type #6 - small drifting bias with slow transient period.

The AFDIA scheme can be simulated for failures of the actuators of stabilators (elevators), ailerons, and rudder. The AFDIA scheme consists of three NN controllers (a longitudinal controller, a directional controller, and a lateral controller) and shares the MNN with the SFDIA. The maximum control surface deflections in the simulation code are ± 20 degrees for the stabilators and ailerons and ± 30 degrees for the rudder. The code also models the actuation rate for each of the control surfaces with a maximum deflection rate of ± 20 deg/sec.

The considered actuator failures are given by:

- stuck R/L stabilator at trimmed/untrimmed angular deflections;
- stuck R/L ailerons at trimmed/untrimmed angular deflections;
- stuck R/L rudder at trimmed/untrimmed angular deflections.

4.3 - The SFDIA/AFDIA Matlab code

A Matlab/Simulink code was developed for the simulation of the fault tolerant schemes described above. For the execution of the code the user is required to have the following Matlab toolboxes:

- Control Toolbox;
- Signal Processing and DSP Blockset toolboxes;
- Simulink.

It is also required by the user to install all the software on the hard-drive in the directory C:\S_AFDIA. The code is run by executing the *'main.m'* Matlab file. The code was designed to perform the following tasks.

Task #1 – Dynamic simulation only (at nominal and actuator failure conditions) – The neural learning is turned off for the SFDIA and AFDIA neural networks.

Task #2 – Dynamic simulation – Simulated on-line learning for the SFDIA and AFDIA neural networks. Flight at nominal conditions (without sensor and/or actuator failures)

Task #3 – SFDIA simulation. "Injection" of sensor failures within user-defined scenarios.

Task #4 – AFDIA simulation. "Injection" of actuator failures within user-defined scenarios.

Task #5 – Simulation of the SFDIA/AFDIA integration. "Injection" of a sequence of sensor and actuator failures within a variety of user-defined scenarios.

The software is organized with the following sub-directories:

S_AFDIA
    TASK_1
        SAVED_SIMULATIONS

NEW_SIMULATION
TASK_2
SAVED_SIMULATIONS
PRE_DESIGNED
NEW_SIMULATION
TASK_3
SAVED_SIMULATIONS
PRE_DESIGNED
NEW_SIMULATION
TASK_4
SAVED_SIMULATIONS
PRE_DESIGNED
NEW_SIMULATION
TASK_5
SAVED_SIMULATIONS
PRE_DESIGNED
NEW_SIMULATION

4.3.1 - Task #1 – Dynamic simulation only (at nominal and actuator failure conditions)

The simulation of the WVU YF-22 linearized dynamics is provided at a given point in the (limited) flight envelope – typically airspeed=50 ft/sec and altitude=1,000 ft. In this task the simulations can be run at both nominal and actuator failure conditions, with the user able to select specific failure scenarios.

First, the user is asked to enter the total simulation time with a range between 0 and 2000 sec. Next the user is prompted with the following menu:

> MANEUVER TYPE
> 1 – Pre-designed Longitudinal Maneuver (at nominal conditions)
> 2 – Pre-designed Lateral/Directional Maneuver (at nominal conditions)
> 3 – Pre-designed Maneuver with Actuator Failure

Should the user select option #1 or #2 generic doublet-type maneuvers are performed. Should option #3 be selected the user is prompted the menu:

> ACTUATOR FAILURE SIMULATION
> 1 – Pre-designed Failure
> 2 – User-defined Failure

If option #1 is selected the user can choose between four different failure scenarios.
If option #2 is selected the user is prompted to provide additional information on the time and the type of actuator failure he/she wishes to simulate. Note that to monitor the failure induced dynamic coupling the user has the option of entering the size of the window for the auto correlation functions $R_{\Phi\Phi}$ and $R_{\Theta\Theta}$. By introducing fairly simple modifications

in the code the user has the option of using different identification schemes based on the use of the cross-correlation functions (R_{pq} , R_{pr} , R_{qr}).

Prior to the simulation the user is also asked whether he/she wishes to add system and/or measurement noise modeled with zero mean and the standard deviation provided in Table 4.4 and Table 4.5. In particular, the user is prompted for a percentage coefficient for the noise level with a mid-range value associated with a nominal level of noise. Should the user select a zero percentage coefficient the simulation will be noise free.

Parameter	Standard Deviation
Airspeed indicator	4 ft/s
Roll rate gyro	0.1 deg/s
Pitch rate gyro	0.1 deg/s
Yaw rate gyro	0.1 deg/s
Longitudinal accelerometer	0.5 ft/s^2
Lateral accelerometer	0.5 ft/s^2
Directional accelerometer	0.5 ft/s^2
Attitude pitch gyro	0.4 deg
Attitude roll gyro	0.4 deg
Attitude yaw gyro	0.4 deg
Altitude indicator	10 ft
Angle of attack	0.6 deg
Sideslip angle	0.6 deg

Table 4.4 – Standard deviation for the measurement noise

Parameter	Standard Deviation
α_{gust}	0.1 deg
β_{gust}	0.1 deg

Table 4.5 – Standard deviation of system noise

Following these selections the dynamic simulation is performed with results shown through different subplots with the following grouping:

- control surface deflections (deg);
- angular velocities (deg/sec);
- Euler angles (deg);
- airspeed (ft/sec), alpha (deg), beta (deg), and altitude (ft);
- trend of the auto correlation functions $R_{\Phi\Phi}$, $R_{\Theta\Theta}$ with the user defined window size.

At the end the user is asked whether to save the simulation data in the sub-directory C:\S_AFDIA\TASK_1\SAVED_SIMULATIONS in the default file 'simulation_all.mat'. If the user wishes to save different simulations he/she should change the name of the default file in the code (at the end of the file '*simulation_only.m*') to avoid overwriting.

4.3.2 - Task #2 – Dynamic simulation – Learning on for the SFDIA and AFDIA neural networks.

The objective for the code for this task is to allow the learning of the different NNs used for the SFDIA and AFDIA schemes at nominal flight conditions, that is without the occurrence of actuator and/or sensor failures.

First, the user is asked to enter the total simulation time with a range between 0 and 2000 sec with a range between 0 and 2000 sec. As in Task #1 the user is also asked whether he/she wishes to add system and/or measurement noise modeled as described above. Should the user select a zero percentage coefficient for both the system and the measurement noise, the simulation will be noise free.

Next, the user is shown a list of the different SFDIA and AFDIA neural networks (MNN, p_DNN, q_DNN, r_DNN, roll NN controller, pitch NN controller, and yaw NN controller). These NNs are stored in the C:\S_AFDIA\TASK_2\PRE_TRAINED directory and have undergone an extensive period of training. The user should be aware that the training phase, especially for large size NNs , such as those listed above, can take several hours and/or days.

Next the user is prompted with the following menu:

> SELECTION OF NEURAL NETWORKS
> 1 – Pre-trained NNS
> 2 – Un-trained Random NNS

If option #1 is selected the user would not have degrees of freedom in setting the neural parameters. In fact the code will continue on the same SFDIA and AFDIA NNs listed above.

If option #2 is selected, the user will be asked to enter in sequence:

- number of neurons in the hidden layer;
- learning rate (either constant or linearly decreasing);
- momentum rate (either constant or linearly decreasing).

The user will be shown recommended values or range of values for each of the above parameters. Note that single hidden layer architectures have been "imposed" for computational simplicity purposes. This input procedure has to be repeated for each of the SFDIA NNs and for each of the AFDIA NNs. Note that the individual inputs and outputs of the SFDIA and AFDIA NNs can only be changed in the software.

Following the selection relative to the neural training of the NNs the user should then proceed to provide information about the dynamic simulation in a totally similar format as in Task #1. The user should be aware that the simulation would be substantially slower due to the addition of the neural computations.

At the end of the simulation of the learning the software proceeds to display the neural outputs such as:

- p, p_DNN, p_MNN
- q, q_DNN, q_MNN
- r, r_DNN, r_MNN
- delta_e, delta_e_nn_long
- delta_a, delta_a_nn_lat

- delta_r, delta_r_nn_dir

in addition to the statistical parameters of the learning described in the previous section of this document (MQEE, OQEE, q_QEE, r_QEE, p_QEE). The longitudinal and lateral directional flight parameters are also displayed.

After all the time histories of the NNs learning errors and the simulation data are displayed, the user will be asked whether he/she wishes to save all the simulation (the flight data and the neural architectures) in the default files and 'nnet_simu_learn.mat' and 'simu_learn_all.mat' in the directory C:\S_AFDIA\TASK_2\SAVED_SIMULATIONS. If the user wishes to save different simulations he/she should change the name of the default file in the code (at the end of the file '*simu_learn.m*') to avoid overwriting.

4.3.3 - Task #3 – SFDIA simulation.

The code for this task is conceptually identical to the code of the previous task, with the only difference being the "injection" of simulated sensor failures.

The user is first prompted the following menu:

> SFDIA SCENARIO
> 1 - Pre-selected failures with pre-selected SFDIA NNs and thresholds
> 2 - User-defined failures with user-defined SFDIA NNs and thresholds;

If option #1 is selected, the code will guide the user through the set-up of a simulation with given failure scenarios and given NNs and thresholds, displayed on the screen prior to the simulation. The user has the option to set the simulation time, the failure time, and the levels of system and measurement noise throughout the simulation.

If option #2 is selected, the user will have more degrees of freedom in setting-up the simulation. In fact, the code will guide the user through a set of menus for the selection of the different failure parameters. The user can also select to start from new SFDIA NNs and thresholds if he/she wishes to do so – OR – start from pre-trained SFDIA NNs and/or previously selected SFDIA thresholds. As for option #1, the user can select the simulation time, the sensor failure time, as well as the level of system and measurement noise.

After the selection of all the different parameters, the simulation is conducted. The sensor failure detection time should be displayed on the screen. At the end of the simulation of the learning the software proceeds to display the neural outputs in addition to the statistical parameters of the learning described in the previous section of this document (MQEE, OQEE, q_QEE, r_QEE, p_QEE). The longitudinal and lateral directional flight parameters are also displayed showing the effects of the sensor failure and the subsequent SFDIA process.

After all the time histories of the NNs learning errors and the simulation data are displayed, the user will be asked whether he/she wishes to save all the simulation (both the flight data and the neural architectures) in the default files and 'sfdia_all.mat' and 'nnet_sfdia.mat' in the directory C:\S_AFDIA\TASK_3\SAVED_SIMULATIONS. If the user wishes to save different simulations he/she should change the name of the default file in the code (at the end of the file '*sfdia.m*') to avoid overwriting.

4.3.4 - Task #4 – AFDIA simulation.

Task #4 is conceptually identical to Task #3, except that it deals with the actuator failure and the AFDIA scheme in lieu of the sensor failure problem and the SFDIA scheme. It should be recalled that the two schemes share the same MNN for failure detection purposes.

From an I/O point of view the codes for Task #3 and Task #4 are very similar. In fact, the user is first prompted the following menu:

> AFDIA SCENARIO
> 1 - Pre-selected failures with pre-selected AFDIA NNs and thresholds
> 2 - User-defined failures with user-defined AFDIA NNs and thresholds;

If option #1 is selected, the code will guide the user through the set-up of a simulation with given failure scenarios and given NNs and thresholds, displayed on the screen prior to the simulation. The user has the option to set the simulation time, the actuator failure time, and the levels of system and measurement noise throughout the simulation.

If option #2 is selected, the user will have more degrees of freedom in setting-up the simulation. In fact, the code will guide the user through a set of menus for the selection of the different failure parameters. The user can also select to start from new AFDIA NNs and thresholds if he/she wishes to do so – OR – start from pre-trained AFDIA NNs and/or previously selected AFDIA thresholds. As for option #1, the user can select the simulation time, the actuator failure time, as well as the level of system and measurement noise.

It should be recalled that the code features an original aerodynamic modeling of the effects of actuator failures involving loss of control surfaces.

After an appropriate selection of all the different parameters, the simulation is conducted. The actuator failure time should be displayed on the screen. At the end of the simulation of the learning the software proceeds to display the neural outputs and the trend for the auto-correlation functions $R_{\Phi\Phi}$ and $R_{\Theta\Theta}$. The longitudinal and lateral directional flight parameters are also displayed showing the effects of the actuator failure and the subsequent AFDIA process with the re-gain of the equilibrium conditions.

After all the time histories of the NNs learning errors and the simulation data are displayed, the user will be asked whether he/she wishes to save the simulation data (both the flight data and the neural architectures) in the default files and 'afdia_all.mat' and 'nnet_afdia.mat' in the directory C:\S_AFDIA\TASK_4\SAVED_SIMULATIONS. If the user wishes to save different simulations he/she should change the name of the default file in the code (at the end of the file '*afdia.m*') to avoid overwriting.

4.3.5 - Task #5 – SFDIA/AFDIA simulation.

Task #5 represents the conceptual merge between Task #3 and Task #4. In fact the code in Task #5 requires all the input information as in the previous two tasks.

The SFDIA/AFDIA integration issue (Napolitano2000) has not been extensively considered in the technical literature. There is a realistic possibility that a sensor failure

could be interpreted as an actuator failure and vice-versa. There are two critical issues to understand the SFDIA/AFDIA integration proposed in this code:

- the dynamic signatures of actuator failures on the trend of the described auto and/or cross correlation functions is more clear and pronounced than the signatures by sensor failures;
- the dynamic signatures of sensor failures on the values of the quadratic performance indices of the neural learning is more clear and pronounced than the signatures by actuator failures.

The identification scheme between a sensor or an actuator failure will take advantage of these characteristics.

First, the user is first prompted the following menu:

> SFDIA and AFDIA SCENARIOS
> 1 - Pre-selected sensor and actuator failures with pre-selected SFDIA and AFDIA NNs and thresholds
> 2 - User-defined sensor and actuator failures with user-defined SFDIA and AFDIA NNs and thresholds;

As for the codes for Task #3 and #4, if option #1 is selected, the code will guide the user through the set-up of a simulation with given failure scenarios for both actuator and sensor and given NNs and thresholds, displayed on the screen prior to the simulation. The user has the option to set the simulation time, the failure time, and the levels of system and measurement noise throughout the simulation.

If option #2 is selected, the user will have more degrees of freedom in setting-up the simulation. In fact, the code will guide the user through a set of menus for the selection of the different failure parameters.

An important detail is that the user should set the sensor failure before the actuator failure. There is nothing unique about this set-up; the reason is that it might be difficult to visualize the effects of the SFDIA when conducted after the AFDIA since the only function of the AFDIA task is to bring back the aircraft to equilibrium conditions. As in previous tasks, the user can also select to start from new SFDIA and AFDIA NNs and thresholds if he/she wishes to do so – OR – start from pre-trained SFDIA and AFDIA NNs and/or previously selected SFDIA and AFDIA thresholds. As for option #1, the user can select the simulation time, the sensor/actuator failure time, as well as the level of system and measurement noise.

After an appropriate selection of all the different parameters, the simulation with both failures is conducted. The exact time for sensor and actuator failures should be displayed on the screen. It should be recalled that the code is fairly slow due to the very heavy computational effort on the top of the already "computationally-slow" Matlab environment. At the end of the simulation of the learning the software proceeds to display the outputs of the neural controllers, the statistical parameters of the SFDIA NNs and the trend for the auto-correlation functions $R_{\Phi\Phi}$, $R_{\Theta\Theta}$. The longitudinal and lateral directional flight parameters are also displayed showing the effects of the sensor/actuator failures and the subsequent SFDIA/AFDIA process with the aircraft re-gaining an equilibrium condition.

After all the time histories of the NNs learning errors and simulation data are displayed, the user will be asked whether he/she wishes to save all the simulation (both the flight data and the neural architectures) in the default files and 'safdia_all.mat' and 'nnet_safdia.mat' in the directory C:\S_AFDIA\TASK_5\SAVED_SIMULATIONS. If the user wishes to save different simulations he/she should change the name of the default file in the code (at the end of the file '*s_afdia.m*') to avoid overwriting.

References (Section #4)

[Atkeson1997a] - Atkeson, C. G., Moore, A. W., & Schaal, S. "Locally Weighted Learning."Artificial Intelligence Review, 11:11-73, 1997

[Atkeson1997b] - Atkeson, C. G., Moore, A. W., & Schaal, S. "Locally Weighted Learning for Control." Artificial Intelligence Review, 11:75-113, 1997

[Hoak1978] - Hoak, D.E., Finck, R.D., "USAF Stability and Control DATCOM", AFWAL-TR-83-3048, October 1960, Revised 1978

[Iliff1972] – Iliff, K.W., Taylor, L.W., "Determination of Stability Derivatives From Flight Data Using a Newton-Raphson Minimization Technique", NASA TN D-6579, 1972

[Iliff1976] - Iliff, K.W., Maine, R.E., "Practical Aspects of Using a Maximum Likelihood Estimation Method to Extract Stability and Control Derivatives From Flight Data", NASA TN D-8209, 1976

[Klein1978] - Klein, V. "Aircraft Parameter Estimation in Frequency Domain", Proceedings of the 1978 AIAA Atmospheric Flight Mechanics Conference, AIAA paper 78-1344, Palo Alto, Ca, August 1978

[Ljung1987] - Ljung, L. "System Identification : Theory for the User", Prentice Hall, Englewood Cliffs, NJ, 1987

[Maine1986] - Maine, R.E., Iliff, K.W., "Identification of Dynamic Systems: Theory and Formulation", NASA RF 1168, June 1986

[Mendel1973] - Mendel, J.M. "Discrete Techniques of Parameter Estimation", Marcel Dekker, New York, NY, 1973

[Morelli1997] - Morelli, E.A. "High Accuracy Evaluation of the Finite Fourier Transform Using Sampled Data", NASA TM 110340, June 1997

[Morelli1998] - Morelli, E.A. "In Flight System Identification", Proceedings of the 1998 AIAA Atmospheric Flight Mechanics Conference, AIAA paper 98-4261, Boston, Ma, August 1998

[Morelli1999] - Morelli, E.A. "Real-Time Parameter Estimation in the Frequency Domain", Proceedings of the 1999 AIAA Atmospheric Flight Mechanics Conference, AIAA paper 99-4043, Portland, Or, August 1999

[Napolitano1995] - Napolitano, M.R., Naylor, S., Neppach, C., Casdorph, V. "On-Line Learning Non-Linear Direct Neurocontrollers for Restructurable Control Systems", Journal of Guidance, Control and Dynamics, Vol. 18, No. 1, pp. 170-176, January-February 1995

[Napolitano2000] - Napolitano, M.R., Younghwan A., Seanor, B., "A Fault Tolerant Flight Control Systems for Sensor and Actuator Failures Using Neural Networks", Aircraft Design Journal, Pergamon Press, Volume 3 (2000), pp. 103-128.

[Neter1996] - Neter, J., Kutner, M. H., Nachtsheim, C. J., and Wasserman, W. "Applied Linear Regression Models", 3rd Edition, Richard D. Irwin, Inc., Burr Ridge, Illinois, 1996.

Section #5

Formal Specification of Requirements for Analytical Redundancy Based Fault Tolerant Flight Control Systems

Section #5 - Table of Contents

List of Symbols (Section #5)

Acronyms

AR	Anaiytical Redundancy
FBW	Fly-By-Wire
FCS	Flight Control System
FTC	Fault Tolerance Capability
FTFCS	Fault Tolerant Flight Control System
UAV	Unmanned Air Vehicles

5.1. – Definition of the Problem

5.1.1 - Fault Tolerance in Fly-By-Wire Flight Control Systems

The use of Fly-By-Wire (FBW) digital flight control systems is playing a more and more prominent role in commercial aviation. In FBW technology electronic devices coupled to a digital computer replace conventional mechanical controls. The net result is a more efficient, easier to control aircraft. However, this increased automation goes in parallel with an increased complexity of flight control systems with obvious consequences on reliability and safety. A FBW flight control system is made up of several subsystems including mechanical, electronic, and software components. Each of these subsystems may fail during flight; thus flight control systems must meet strict fault-tolerance requirements. The standard solution to achieving fault tolerance capability is the adoption of a multi-string architecture. This architecture is based on physical redundancy, that is redundant units working in parallel and a *voting scheme* that disengages a unit when identified to be faulty. Typically triple and quadruple string architectures are featured within flight control systems for both military and commercial aviation [Yeh96], [Favre94]. However, multi-string architectures further increase the complexity of the system, induce a reduction of overall reliability, bind to closer maintenance schedule, and require larger budgets. These factors have induced in recent years an increased interest toward alternative approaches to achieving fault tolerance in flight control systems.

Similar interest comes from Unmanned Aerial Vehicle (UAV) applications. Starting in the late '80s a variety of UAVs have been built for either military or scientific purposes. They vary significantly in size, mission profile, and payload weight carrying capability. With some of them having a payload weight below 20 lbs and dimensions below 15 feet it is clear how weight and room requirements are a major issue. Despite costs, complexity, and weight drawbacks physical redundancy is adopted to achieve fault tolerance.

5.1.2 - Fault Tolerance and Analytical Redundancy

Fault tolerance requires some form of redundancy within the system; redundancy provides alternative means to perform a specific task, thus making the system capable of continuing operation despite of localized malfunctions, i.e. of tolerating faults. Two different redundancy approaches are adopted in closed loop system: physical redundancy and analytical redundancy. Physical redundancy is based on a multi-channel architecture consisting of three or more intercommunicating systems that are able to work independently. A voting mechanism checks for consistency among the redundant components of each channel. Analytical redundancy identifies with the functional redundancy in the system dynamics. It does not require additional hardware; fault tolerance is achieved by means of software routines that process sensor outputs and actuator inputs to check for consistency with respect to the analytical model of the system. If an inconsistency is detected, the faulty component is isolated and the control law is reconfigured accordingly. Preserved observability allows estimating the measurement of an isolated (allegedly faulty) sensor, while preserved controllability

allows controlling the system with an isolated (allegedly faulty) actuator. Numerous survey papers and books [Patton89], [Chen99], [Patton00] discuss theoretical and practical aspects of adopting the analytical redundancy approach to achieve fault tolerance.

The conceptual structure of analytical redundancy based fault detection and identification systems consists of two stages: the *residual generation* stage and the *decision making* stage [Chow80]. The residuals provide a measure of the inconsistency between the actual behavior of the system and the system analytical model. Residual values close to zero imply a fault free system; on the other hand, residual values different from zero reveal a fault within the system, and the particular combination of residual values provides means for isolating the faulty component. Processing of the residuals to perform fault detection and isolation is the main task of the decision stage. Decision algorithms range from simple threshold testing on the instantaneous values or on the moving average of the residuals, to more sophisticated statistical testing based on the Generalized Likelihood Ratio test [Willsky76], or on the Sequential Probability Ratio test [Basseville88]. To achieve fault tolerance an additional recovery stage needs to be added. This stage consists of an adaptive or multi-model control law that processes information provided by the decision making stage to produce a suitable control law. All of the three stages play an equally important role toward the successful fault tolerant control system; however, most of the research focuses on the residual generation problem.

The first analytical redundancy scheme implemented within a flight control systems dates back to the 70's, when the same aircraft used to conduct research on fly-by-wire technology was also used as testbed for an analytical redundancy management algorithm [Szalai80]. The algorithm showed desirable performance during flight test; however, poor robustness to modeling errors and the degree of modeling necessary retrained further development. Since then, a number of results have been obtained in the area of robust fault diagnosis [Patton94].

5.1.3 - Certification of Analytical Redundancy based Fault Tolerant Flight Control Systems

While research on analytical redundancy has been obtaining desirable results, a design methodology involving requirements specification, feasibility analysis, and certification of analytical redundancy based fault tolerant flight control systems is still missing.

A major problem with the analytical redundancy approach is related to the certification process. Since analytical redundancy based systems exploit the functional redundancy of the plant, when applied in the field of flight control systems they need to be validated over the entire aircraft operational envelope. Instead, most of these solutions are evaluated using a simplified model of the aircraft dynamics, within a limited region of the flight envelope, and with a limited set of maneuvers and fault-modes. Furthermore, evaluation criteria are quite heuristic. A tentative list of criteria for assessing the performance of fault detection and identification systems can be found in [Patton89], and is summarized below:

- promptness of detection,
- sensitivity to incipient failure,

- false alarm rate,
- missed fault detection,
- incorrect fault identification.

A typical testing procedure for fault detection and identification systems involves injection of a set of failures within a simulation environment and computation of the above indexes. While obtained values can be effectively used to compare the performance of two different solutions, they have no absolute interpretation. The testing environment has a considerable impact on the evaluation of these indexes. Missed detection and false alarm rate do not provide any valuable information if they are not determined within the entire operational envelope of the system. These figures are highly dependent on the disturbances acting on the system, on the type of fault injected, and -- for non linear systems -- on the state of the system. Furthermore, the fault could be not detectable at all, thus leading to a missed rate of 100%. But this value is not an index of poor performance of the fault detection system; rather, it indicates a lack of functional redundancy within the system.

Exploring strengths, weaknesses, related degree of reduction of physical redundancy, and overall reliability is a fundamental step in the engineering of analytical redundancy based fault tolerant flight control systems. In order to provide an objective basis for the evaluation of such systems it is mandatory to develop the requirements specification. Validation of a system can be performed only against its specification.

5.1.4 - Research Objectives

The main objective of this part of the research was to investigate the possibility of capturing within a formal framework the constraints that the Analytical Redundancy approach imposes upon the system providing Fault Tolerance Capability (FTC). The focus was on the methodology underlying the development of formal specification for an Analytical Redundancy based FTC, as well as on the characteristics of the formal specification language.

5.2. – Definition of the Approach

5.2.1 - Formal specification of system requirements

The requirements specification involves a considerable amount of engineering analysis and judgement, it is the result of a long sequence of refinements, it is produced with the collaboration of personnel in different area of expertise, and typically results in a voluminous document with a complex set of dependencies. These factors make it difficult to produce a consistent and unambiguous requirements specification. Despite the considerable expressive power of plain-English, its use as specification language introduces an additional source of ambiguity and inconsistency. Considerable leverage can be obtained by adopting a *formal specification language*. A formal language is a language with a mathematically defined syntax and semantics. The mathematical definition of the language potentially eliminates the ambiguity problem, allows for checking the consistency of the specification, and leads to a specification amenable to automated analysis. Furthermore, the rigid structure of the formal specification serves as a guide in formulating the requirements resulting in a homogeneous document throughout the refinement iterations.

The specification framework adopted in this effort is based on *Relational Algebra.* While developing our specification framework we aimed at providing support for features that are important in the formulation of the AR-FTFCS requirements. These features are the capability of formulating requirements in both *explicit* and *implicit* form, the capability of formulating both *functional* and *operational* requirements, and the capability of specifying a refinement ordering among specifications. We have chosen to specify the requirements in terms of relations. This is not a choice of specification language as much as it is a choice of *reasoning framework*, and a choice of *semantic model*. The focus is on the methodology underlying the development of the formal specification rather than on the analysis of the specification and its use within the certification phase.

5.2.2 - Specification Approach

The following premises have been adopted as a basis for the development of the specification:

- the need to adopt a specification paradigm that is suitable for process-control systems
- the need to produce a formal specification *semantically* and *structurally* close to the natural language specification
- the need to specify both *functional* and *operational* requirements
- the need to organize the requirements in a hierarchical structure where introduction of design constraints results in the refinement of a higher level (approach independent) specification

To satisfy the first premise we adopt a history-based specification, where each requirement is specified by the set of acceptable histories of the relevant quantities. Furthermore, we adopt a *closed-system* approach [Feather-ACM87] where the entire environment is described in the specification; as opposed to the *open-system* approach where only the interface between system and environment is described. The history-based

paradigm combined with the closed-system approach is particularly suitable for the specification of the requirements of process control systems.

With the second premise in mind the objective is to bridge the gap between the original informal specification and the resulting formal specification. It is important that the *semantic distance* [Leveson94] between the two specifications is minimized in order to maximize the application expert's ability to review the specification. Since this is not always possible, unless the original specification is already well structured and formulated, we go one step further and produce a *reviewed* natural language specification. We take advantage of the feedback that naturally arises from the formalization process to enhance/refine the original requirements. Then, instead of just developing a formal specification that overcomes the flaws of the original requirement, we produce both a formal specification and a *reviewed* natural language (informal) specification. This informal specification is *semantically* closer to the formal counterpart, so it plays a valuable role in augmenting the specification communicability.

The third premise is related to an important feature of the original specification: the distinction between *FCS functional requirements* and *FCS operational state*. In the context of process-control systems the scope of functional requirements is wider than mere input/output behavior. It typically includes such aspects as timing and performance, because they are indistinguishable from the input/output behavior. The system operational state refers to the operating status of the components performing the system function. For example, a multiple channel failure does not necessarily compromise the FCS functions if the remaining channel/channels work correctly; however, it does modify the operational state of the FCS with obvious consequences on system safety and mission reliability. The classification of FCS operational states (Section 1.2.2 [MIL-F-9490D]) is adopted in the original specification to formulate requirements related to fault tolerance, aircraft safety, and mission reliability. We refer to these requirements as *operational requirements*.

The FCS functional requirements are formulated in terms of constraints over the time evolution of the environment state for a given system input. The system input/output behavior is implicitly specified by specifying the behavior of the closed loop system (system + environment), the description of the environment, and the interaction between the system and the environment. We use relations to formulate the system functional requirements. The relation signature specifies the domain and image variables in terms of which the requirement is specified in the relation predicate. To be able to formulate typical process control requirements the entire time-history of domain and image variables must be visible within the scope of the relational predicate. For this reason, we adopt variables that represent the time-history of the corresponding quantity, as opposed to the quantity's instantaneous value. This is a common approach in specifying requirements of process-control systems and has been adopted in the past within a formal framework [Parnas95]. The history-based paradigm combined with the closed-system approach is particularly suitable for the specification of the requirements of process control systems. The functional requirements for this type of reactive systems are typically formulated in terms of constraints over the time evolution of the environment state for a given system input. An example of such requirements is the maximum allowed RMS-error in a *tracking* problem. In such control problem the reactive system is required to perform an action (system output) upon the environment so that the environment state

(or one of its elements) tracks a reference signal (system input). The RMS-error is the square root of the integral of the square deviation between reference signal and environment state. This requirement is clearly expressed in terms of time-histories of system input and environment state. For illustrative purposes, in Appendix A we report the relation that specifies the Heading Hold (HH) control function requirement, along with related tables and some observations upon the interpretation of the original requirement.

As for the operational requirements we focus on those related to fault tolerance. These requirements are formulated in terms of the functions that the system must be able to provide under different operational states. We formulate the FCS functions by means of relations, as described above. We model the system operational state as a sequence of component's behaviors, each of them being formulated by means of a relation 'B'. For a particular operational state 'h', the actual behavior of each system component is represented by a relation 'O(h)' that refines the corresponding behavior 'B'. The fault tolerance requirements are formulated as a composition of functional requirements and actual system behavior under the relevant system operational states. For illustrative purposes, in Appendix B we report the relations that specify nominal operating mode and fault modes of the roll rate gyro.

Our last premise relates to the organization of the requirements in a hierarchical structure. We satisfy this premise by exploiting composition operators and refinement ordering of relational algebra. We first produce an approach independent specification of the fault tolerance requirements, then we introduce the design constraints resulting from adopting the AR approach. We believe that such an approach not only increases readability of the specification but also its reusability since the higher level specification can be used for different design approaches. It is worth noting that the lower level specification is targeted to control system engineers; only once control system engineers have produced the *laws* that they claim to meet the specification is it possible to go one level lower in the hierarchy and produce the specification of the software that will implement these laws.

5.2.3 - Scope of Fault Tolerance Requirements

In the USAF specification for Flight Control Systems (FCSs) [MIL-F-9490D] fault tolerance is addressed in different sections, each section focusing on one particular aspect of fault tolerance. Section 3.1.6 addresses the implications of FCS failure upon the mission and defines *mission accomplishment reliability* as "the probability of mission failure per flight due to relevant material failures in the flight control system". Section 3.1.7 addresses the implications of FCS failure on aircraft safety and defines *flight safety* as "the probability of aircraft loss per flight due to relevant material failures in the flight control system". Section 3.1.8 defines *survivability* as the capability to of the FCS to provide Operational States IV or V in extreme operating conditions. Section 3.1.9 defines *invulnerability* as the FCS capability to operate under "variations in natural environments, induced environments, failure of other systems, etc." We do not directly address the aspects listed above, though they are related to fault tolerance. Instead, we focus on the specification of *redundancy* (see section 3.1.3.1 of [MIL-F-9490D]) and on the different degrees of fault tolerance that should be achieved by exploiting redundancy,

namely *fail operational, fail passive,* and *fail safe* capabilities, as described in section 6.6 of [AFFDL-TR-74-116sup1]

5.2.4 - Specification Language: Relational Algebra

Relational algebra is the mathematical framework that allows operating and reasoning with requirements formulated in terms of predicate logic formulae. In the following a brief introduction to Relational Algebra is presented, the interested reader is referred to [Schmidt91].

Given two spaces (or sets) A and B, a relation R over A×B is a subset of A×B and is specified as follows:

$$R = \{(A,B) \mid F(A,B) \} \tag{5.1}$$

The above expression reads: relation R is the set of couples of elements $(A,B) \in A \times B$ such that F(A,B) evaluates true. The data-structure (A,B) is the *signature* of the relation. F is a predicate logic formula and is specified according to syntax and semantics of first order predicate logic [Alagar98]. If (A,B) is an element of relation R, then A is called an *antecedent* of R and B an *image* of R. The set of all antecedents of R is the *domain* of relation R and is denoted *dom*(R). The set of all images of R is the *range* of relation R and is denoted *rng*(R). The *universal* (or *total*) relation over the set A is defined by A×A and is denoted L_A. The *identity* relation over the set A is defined by $\{(A,A') \mid a = a'\}$ and is denoted by I_A.

Since relations are sets, they inherit all set operators and the inclusion ordering. Furthermore, new operators are introduced for relations such as the *inverse* of a relation, the *product* of two relations, the *pre-* and *post-restriction* upon relations, etc. The reader is referred to [Schmidt91]. In terms of requirements specification the expression

$$R = \{(A,B) \mid F(A,B) \} \tag{5.2}$$

is interpreted as follows: objects adopted in formulating the requirement are represented by domain and image variables; A and B are structures whose elements are the domain and image variables respectively; F(A,B) is a predicate logic formula that specifies the requirement in terms of domain and image variables, ad hoc introduced quantified variables, functions of such variables and constant values.

Relational algebra provides the formal framework for reasoning with predicate logic specifications. This framework is based on composition operators and a refinement ordering among requirement specifications. Composition operators allow *specification by parts*. This specification method consists in breaking the required behavior of the system into parts. Each part is specified by means of a relation. Then, these relations are composed together to form the whole requirements specification. The refinement ordering captures the idea of relative *strength* between two requirements. Also, it allows defining the *correctness* of a system implementation with respect to its requirements.
Relation R is said to *refine* (or be a *refinement* of) relation R' (denoted by RPR') if and only if

$$RL \subseteq R'L \ A \ R'L \cap R \subseteq R' \tag{5.3}$$

RPR' implies

- *dom*(R) $\supseteq$ *dom*(R')
  that is the requirement specified by R extends over a larger (or equal) number of input scenarios, and
- $\forall$ a : a $\in$ *dom*(R') (R(a) $\subseteq$ R'(a)),
  that is relation R is more specific in its assignment of outputs to inputs.

The defined ordering among specifications reflects the strength of the related requirements. Relation R refines relation R' if it defines a stronger (harder to satisfy) requirement. If RPR', then any system that satisfies R satisfies R' as well. The refinement ordering is also used to define correctness of an implementation. A system implementation P is said to be correct with respect to its specification R if and only if PPR.

The *product* of relation $R_1 \subseteq A \times K$ by relation $R_2 \subseteq K \times B$ is the relation over A×B denoted by $R_1 \Pi R_2$ (or R_1R_2) and defined by:

$$R_1 \Pi R_2 = \{ (A,B) \mid \exists \, k \, ((a,k) \in R_1 \, A \, (k,b) \in R_2) \} \tag{5.4}$$

The sum of the requirement information of two relations $R_1 \subseteq A \times B$ by relation $R_2 \subseteq A \times B$is called the *join* of R_1 and R_2 and is denoted by $R_1 \vartheta R_2$. The following implication holds:

$$R \ P \ (R_1 \vartheta R_2) \Leftrightarrow RPR_1 \ A \ RPR_2 \tag{5.5}$$

The join exists if and only if R_1 and R_2 do not contradict each other. A consistency condition is available to check whether the join of R_1 and R_2 exists. The common requirement information of two relations $R_1 \subseteq A \times B$ and $R_2 \subseteq A$ is called the *meet* of R_1 and R_2 and is denoted by R_1MR_2. The meet of two relations is defined as follows:

$$R_1MR_2 = R_1L \cap R_2L \cap (R_1 \cup R_2) \tag{5.6}$$

The following implication holds:

$$R = R_1MR_2 \Rightarrow R_1PR_2 \ A \ R_2PR \tag{5.7}$$

5.2.5 - Pilot Application

To develop the requirements specification of an Analytical Redundancy based Fault Tolerance Capability (FTC) system all relevant details of its environment must be specified. Hence, a pilot application is needed, an aircraft equipped with a FCS that can be adopted as environment for the FTC. The aircraft selected is the De Havilland DHC-2, also known as *Beaver*. This is a general aviation, single engine, high-wing aircraft with a wing span of about 15 meters, fuselage length of about 9 meters, and a maximum take-off

weight of about 2300 Kg. Its analytical model and its FCS are provided in the Flight Dynamics and Control (FDC) Toolbox for Matlab. Information in [Rauw93], [Rauw98], and [Tjee88] was adopted to provide a description of the components of the FTC environment. The cited documentation provides the analytical model of the DHC-2 aircraft, of the actuator-control-surface chain, the engine, and the continuous-time flight control laws. This information covers the description of the aircraft subsystem, the actuators block, and the FCL block. The description of the environment was completed by developing suitable analytical models for the remaining blocks.

Aerodynamic derivatives and moments of inertia from [Rauw98] were adopted for the analytical model of the aircraft; uncertainty bands about nominal values were introduced according to [Hoak68]. Actuator-control-surface models include elevators, ailerons, and rudder dynamics; the analytical model of the flaps is not included since the flaps are not used by the autopilot functions. The cited documentation does not contain any sensor model; hence, analytical models were developed from the technical specification of commercial sensors.

5.3. - Results

5.3.1 - Requirements Hierarchy

The block diagram in Figure 5.1 represents the hierarchy of the requirements for the Analytical Redundancy based Fault Tolerance Capability (AR-FTC).

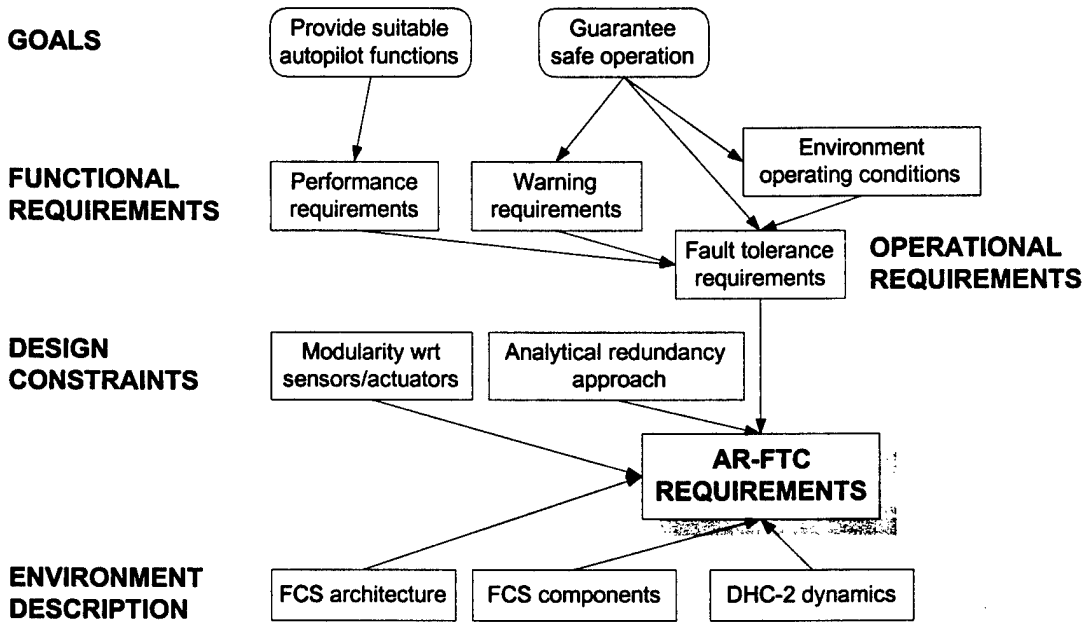


Figure 5.1 - Specification Structure

The topmost layer describes the goals of the system that are *suitable autopilot functions* and *safety of operation.* The functions and the related performance requirements under the relevant operating conditions are captured by the second layer of the requirements hierarchy: *Functional requirements* and *Operational requirements.* The *Design Constraints* layer introduces constraints requiring the adoption of the *Analytical Redundancy approach* and *modularity*, with respect to sensor and actuator subsystems. The very last layer describes the *environment* of the AR-FTC. The composition of the different requirements, constraints, and descriptions makes up the requirements for the AR-FTC system. In the following section each layer of the hierarchy is briefly described.

5.3.1.1 - Functional and Operational Requirements

The military specification MIL-F-9490D [MIL-F-9490D] is adopted as main source for fault tolerance, performance, and warning requirements for AFCS. MIL-F-9490D *"Flight Control Systems - Design, Installation and Test of Piloted Aircraft, General Specification for"* is the active specification for FCS for US Air Force manned piloted aircraft. It is supported by other military specifications, standards, handbooks, and non-military publications such as FAA Advisory Circulars, National Aircraft Standards,

Technical Reports, etc. The most relevant supporting documents with respect to this research are:

- the military specification MIL-F-8785C *"Flying Qualities of Piloted Airplanes"* [MIL-F-8785C],
- the supplement *"Appendix to Background Information and User Guide for MIL-F-9490D"* [AFFDL-TR-74-116sup1],
- the Technical Report *"Background Information and User guide for MIL-F-9490D"* [AFFDL-TR-74-116] and
- the Technical Report *"Background Information and User guide for MIL-F-8785C"* [AFWAL-TR-81-3109].

For the purpose of developing the requirements specification for the FTC system a narrow subset of requirements has been selected. The performance requirements for the following autopilot functions are considered: Pitch Attitude Hold (PAH), Altitude Hold, (ALH), Roll Attitude Hold (RAH), Heading Hold (HH), and Heading Select (HS). Coordination requirements for lateral-directional control functions, both in steady banked turns and in level flight are considered. Among the functional requirements the focus is on fault-tolerance requirement, limited to those relevant to fail-operational functions. Warning and Status Display requirements are also considered. Failure transient requirements are not considered since the focus is on fail-operational capability only.

5.3.1.2 - Design Constraints

Two design constraints are introduced in the specification: adoption of the analytical redundancy approach and modularity with respect to sensor and actuator subsystems. The redundancy approach constraint rules out the multi-string architecture for the FCS. Hence, this constraint affects the environment of the AR-FTC system, that is the AFCS within which the AR-FTC operates. The AFCS in described in the next section and is based on a single-string architecture. Use of analytical redundancy implies that detection of the fault and identification of the faulty component are performed by processing sensor outputs and actuator inputs. Fault recovery must be achieved by reconfiguration of the estimator filter and/or of the control law of the AFCS.

Analytical redundancy based fault tolerance is achieved at the software level; software routines process control law inputs and outputs to check their consistency against an analytical model of the controlled system (in this case the aircraft). Analytical redundancy, however, cannot be used to provide fault tolerance with respect to failure of all the FCS components. Any component of the FCS, either hardware or software, can fail. Analytical redundancy cannot help with software failures; nor it can help if in a single-channel FCS the FCC fails, since the FCC hosts the software that provides fault tolerance. Failure of either the control or the display panel cannot be accommodated at the software level; hence, analytical redundancy is -- under these conditions -- useless.

The remaining components of the FCS are the actuators and the sensors. Analytical redundancy based solutions presented in the literature typically separate the problems of actuator and sensor failure. The rationale behind this choice is simple: fault tolerance with respect to sensor failures is mostly an observation problem, while fault tolerance with respect to actuator failures is mostly a control problem; different expertise

and techniques are required for designing the two different FTC systems. For these reasons a modular constraint upon the AR-FTC is imposed in the specification.

5.3.1.3 - Environment Description

The environment of the FTC is the AFCS along with the whole DHC-2 aircraft dynamics. Figure 5.2 shows how the FTC system fits within its environment. Each component of the diagram is specified by means of a relation. A brief conceptual description of the picture is provided in the remaining part of this section. The detailed description of the relations represented in the block diagram is out of the scope of this report; the interested reader is referred to [DelGobbo00].

The arrows in the diagram represent data streams like forces and moments, electrical signals, software data, etc; the blocks represent processing units. Square-corner blocks represent hardware units, while round-corner blocks represent software units. Blocks are grouped by means of dash-lines to form subsystems or systems.

The DHC-2 aircraft system (the topmost subsystem in Figure 2) represents the aggregate of airframe, control surfaces, and engines. The control surfaces are typically included in the airframe; here they are separated since different fault hypotheses are introduced for the two units. The airframe block also includes the contribution of gravitational field and air turbulence to the aircraft dynamics. Each block represents a relation that formally specifies part of the description of the system. The aircraft dynamics is specified through force (R_{Feq}), moment (R_{Meq}), kinematic (R_{Keq}), and navigation (R_{Neq}) equations. The forcing terms for these equations are aerodynamics forces (R_{aef}) and moments (R_{aem}) exerted by the airframe, forces (R_{csf}) and moments (R_{csm}) exerted by the control surfaces, forces (R_{pf}) and moments (R_{pm}) exerted by the engine, and gravity (R_{grf}) and wind (R_{wf}) forces. This section also includes the relations that specify air-data quantities (R_{ad}) and kinematic acceleration at crew station (R_{ka}).

The group of blocks marked FTFCS represents the Fault Tolerant Flight Control System. It is composed of the flight control computer (FCC), the subsystem processing computer-output (Cout), and the subsystem generating computer-input (Cin). The FCC is composed of the Flight Control Software (FCSw) and of the DAC and ADC blocks. The last two blocks represent the transformation from electrical signals to software data and viceversa.

The FCSw is composed of three units: IN, OUT and FCL (Flight Control Law). IN and OUT serve as pre-processing and post-processing units to the flight control law, while FCL is the block that processes software representation of pilot inputs and sensor outputs to produce a software representation of the input to the actuators, the engines, and the display panel.

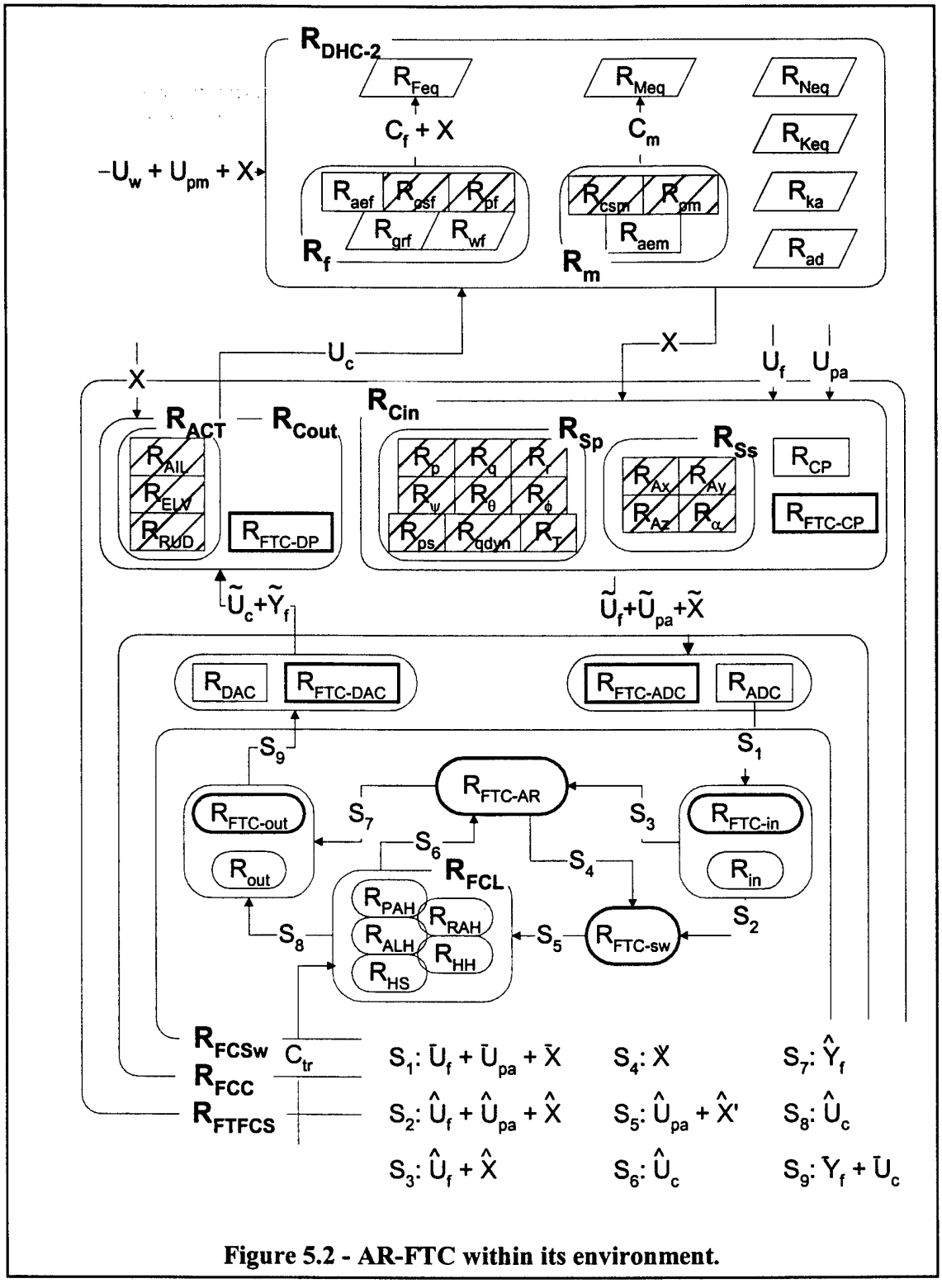


Figure 5.2 - AR-FTC within its environment.

The Cin and Cout subsystems contain blocks whose names are self-explanatory. ACT denotes the actuators, DP and CP denote the display panel and the control panel respectively, and Sp and Ss denote the *primary* and *secondary* sensors respectively. The primary sensors are those that produce measurements used within the FCL. Measurements from secondary sensors, instead, are used for other purposes, eventually from another control law not shown in the diagram.

The FTC system is composed of the blocks marked with a thicker outline. CP_{FTC} and DP_{FTC} represent the control and display panel of the FTC. They represent the interface with the pilot, and provide means to activate/deactivate the FTC and to signal the operating status (nominal/faulty) of the monitored sensors. ADC_{FTC} and DAC_{FTC} represent the interface between the electrical signals from the FTC control and display panels and the related software variables. FTC_{in} and OUT_{out} represent the software modules that serve as interface between the ADC_{FTC} and DAC_{FTC} blocks, and the FTC-AR block. FTC-AR is the core of the FTC; it represents the software routines that process sensor readings (from the IN block) and control inputs (from the OUT block) to check whether the correlation among them is consistent with the analytical model of the environment. This is the system that exploits analytical redundancy to provide fault tolerance.

5.3.2 - Analysis of the Fault Tolerance Requirements

In this section we illustrate the original formulation of the fault tolerance requirements in [MIL-F-9490D] our interpretation of the specification, and the resulting structure of the formal specification. The military specification defines three different degrees of fault tolerance that correspond to different degrees of *criticality* of the FCS function. These degrees are defined as follows:

- **Fail operational** "The capability of the FCS for continued operation without degradation following a single failure, and to fail passive in the event of a related subsequent failure."
- **Fail passive** "The capability of the FCS to automatically disconnect and to revert to a passive state following a failure."
- **Fail safe** "The capability of the FCS in a single channel mode of operation to revert to a safe state following an automatic disconnect in the event of a failure or pilot initiated disconnect."

We limit our study to autopilot control functions such as Pitch Attitude Hold, Altitude Hold, Heading Hold, etc. These are declared as *non-critical* functions since their loss does not affect flight safety, and, as such, they are required to be fail safe. For the purpose of this study we extend the autopilot fault tolerance requirements to the fail operational degree, and we neglect the details of the transition from automatic FCS to manual FCS following automatic disengagement. Another requirement related to the fault tolerance capability is the *FCS warning and status display* requirement (section 3.2.1.4.2.2 of [MIL-F-9490D]). The relevant part of this requirement reads:

"Failure warnings shall be displayed to allow the crew to assess the operable status of redundant or monitored flight control systems. Automatic disengagement of an AFCS mode shall be indicated by an appropriate warning display."

The definitions of fail operational and fail passive capability, the requirement that the autopilot control functions be fail operational, along with the above warning and status display requirement make up the original formulation of the fault tolerance requirements. The detailed analysis and specification of the FT-FCS requirements is beyond the scope of this study. We limit ourselves to point out three major flaws in the original definition of the fail operational capability, since the structure of the formal specification stems from this definition.

The first problem is one of coverage. From our interpretation the specification aims -- though it fails -- to distinguish between two operating conditions of the system: the condition with unrelated components being faulty, and the condition with two or more related components being faulty. The fact that the occurrence of the two faults might be sequential is incidental, the requirement of *continued operation without degradation* must be satisfied also in the case of simultaneous faults. Hence, the wording *in the event of a subsequent failure* is not appropriate. Furthermore, the specification fails to cover the case of unrelated faults.

The second problem is one of terminology. The specification uses the term *related failure* but it fails to define how to determine whether two failures are related. Nor is this concept defined anywhere else in the MIL-F-9490D document. We infer from the specification that the occurrence of related faults is more critical than the occurrence of unrelated faults. In fact, for related faults the FT-FCS is required to be fail passive (disengage), while for unrelated faults (according to the interpretation above) it is required to continue operation. We arbitrarily adopt a functional equivalence among the set of components, so that faults of components that belong to the same class are considered related. The rationale behind this interpretation is that failure of components serving similar functions (e.g. sensors, actuators, etc.) are potentially more critical than failures of components serving different functions. Hence, this interpretation is in agreement with the association between faults relatedness and faults criticality.

The third problem is one of scope. The specification fails to specify the set of components whose failures are considered. Is it the set of components of the FT-FCS? If this is the case, then failure of engines and control surfaces is not covered since they do not belong, by definition (section 1.1 of [MIL-F-9490D]), to this set. We include in the set of components whose failure is taken into account in the fault tolerance requirement, all fallible components that play a role in performing the autopilot control functions.

5.3.3 - Structure of the Formal Specification

As a result of the above analysis and of the requirements hierarchy five main parts can be distinguished in the structure of the specification:

- specification of the decomposition of the system to specify the scope of the requirements and the set of related components
- specification of the operational behavior of the system to support the specification of the fault tolerance requirements

- specification of performance and warning requirements that must be preserved/provided under faulty operating conditions
- specification of the operational requirements
- introduction of the design constraints into the operational requirements

5.3.3.1 - Specification of System Composition

Decomposition of the system and sets of related components are specified as follows:

- definition of relevant sets of components:
  - C: set of components of the system, C_c being the component identified through the subscript 'c'
  - F: set of components that are subject to failure
  - U: set of components used by the FT-FCS (i.e. playing a role in performing the autopilot control functions)
- partitioning of the set of fallible components F into subsets of functionally related components:
  - A subset of actuation components (power control units, control surfaces, engines)
  - M subset of measurement components (sensors)
  - P subset of processing components (software components)
  - I subset of interface components (elements of the control panel)
  - *related*(C_c, C_k) predicate that evaluates true if and only if C_c and C_k are related

The operational behavior of the system results from the operational behavior of its components. Relations are adopted to describe the nominal and faulty behaviors of the system components. Each relation describes the maximal set of histories over time that characterizes the component nominal operation or one of its fault modes. $B_{c,n}$ denotes the relation describing the n-th behavior of component C_c. For n=0, $B_{c,n}$ describes the component's nominal behavior, while for $0 < n \leq f_c$ it describes one of the component's fault modes; f_c being the number of fault modes. An ordered sequence of the above relations represents the system operational status. For example, the fault free operating condition is represented by the following sequence of relations:

$$h_{0,0} = [\, B_{p,0}, B_{q,0}, B_{r,0}, \ldots \,] \tag{5.8}$$

where p, q, r identify some of the system components, in the example roll, pitch, and yaw rate gyros. Such sequence is an element of the space Σ:

$$\Sigma = (X_p \rightarrow Y_p) \times (X_q \rightarrow Y_q) \times (X_r \rightarrow Y_r) \times \ldots \tag{5.9}$$

where (X_c, Y_c) denotes the signature of the relations $B_{c,n}$. Under the system operating condition $h \in \Sigma$ the operational behavior of component C_c is represented by a relation O_c (h) that refines the corresponding relation in the sequence. For example, the sequence $h_{0,0}$ described above implies that the roll rate gyro is operating as described by the relation $O_p(h_{0,0})$ P $B_{p,0}$.

Once all the individual components of the environment are specified the corresponding relations are composed to form the specification of subsystems and systems. To carry out the composition the join operator is used for simultaneous requirements and the product operator for sequential requirements. In order to satisfy the signature compatibility requirements of these operators projection and expansion relations are applied where needed. For the sake of readability these space-altering relations are not specified in the following example of composition. For compositions that do not follow in either of the two categories the direct composition of the relations is obtained by accessing the single elements of the relations' signatures.

The highest level of the composition consists of the composition of the DHC-2 aircraft specification and of the FTFCS specification. The relational formulation of such composition is illustrated below, while Figure (1) depicts a graphical representation of the composition. The figure highlights the signature of the operand relations as well as the signature of their composition. The parameter h represents the system operating condition.

$$O_{DHC2}(h) \otimes O_{FTFCS}(h) = \{((U_w, U_{pm}, U_{pa}, U_f), X) \mid \exists U_c(((U_{pa}, U_f, X), U_c) \in O_{FTFCS}(h) \wedge ((U_c, U_w, U_{pm}), X) \in O_{DHC2}(h))\} \quad (5.10)$$

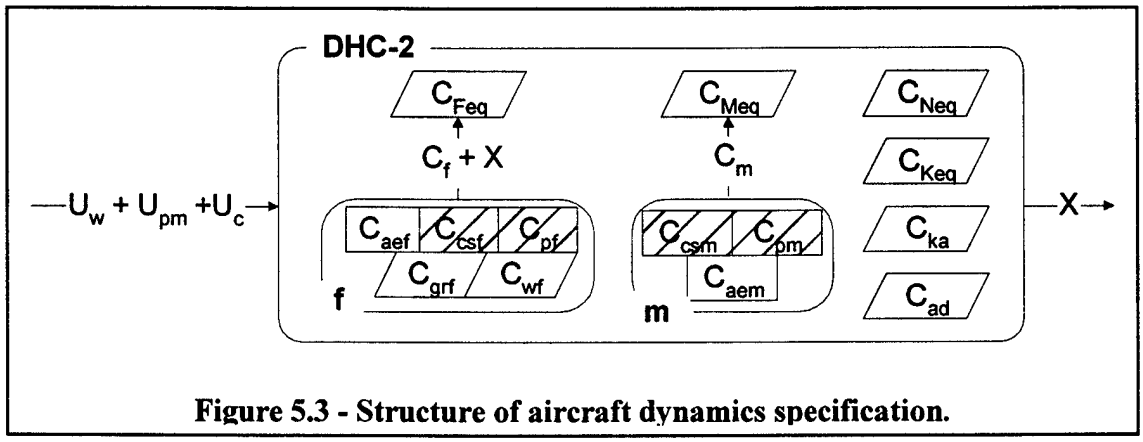


Figure 5.3 - Structure of aircraft dynamics specification.

The second level of the composition consists of the composition of the two separate specifications related to the DCH-2 aircraft and to the FTFCS respectively. For illustrative purposes the relational specification of the DHC-2 aircraft is reported below; it clearly maps the structure of the aircraft dynamics in Figure 2:

$$O_f(h) = O_{aef}(h) \vartheta O_{grf}(h) \vartheta O_{csf}(h) \vartheta O_{pf}(h) \vartheta O_{wf}(h) \quad (5.11)$$

$$O_m(h) = O_{aem}(h) \vartheta O_{csm}(h) \vartheta O_{pm}(h)$$

$$O_{DHC2}(h) = O_f(h) \Pi O_{Feq}(h) \vartheta O_m(h) \Pi O_{Meq}(h) \vartheta O_{Keq}(h) \vartheta O_{Neq}(h) \vartheta \vartheta O_{ad}(h) \vartheta O_{ka}(h)$$

5.3.3.2 - Specification of System Operating Conditions

The system operational conditions are grouped into three classes, on the basis of the different action required to the FTC system in the fault tolerance requirements. Each

class captures a *fault hypothesis* upon the system. The three hypotheses are named *fault-free* hypothesis H_0, the *single-fault* hypothesis H_1, and the *multiple-fault* hypothesis H_2. A major assumption underlies each fault hypothesis that is: *"All fallible components operate either according to their nominal behavior or according to one of their fault modes"*. The assumption states that manufacturer specification of nominal and faulty behaviors of each component form the basis of the fault tolerance requirements (and of the resulting certification process), regardless of the fact that those specifications actually cover all possible behaviors of the components. This assumption is formalized as follows:

$$H' = \{ h \in \Sigma \mid \forall c : C_c \in F \, (\exists m : 0 \leq m \leq f_c \, (O_c (h) \, P \, B_{c,m})) \} \qquad (5.12)$$

The Fault-free hypothesis reads: *"All of the components used by the FT-FCS are working according to their nominal behavior"*, and is formalized as follows:

$$H_0 = H' \cap \{ h \in \Sigma \mid \forall c : C_c \in U \, (O_c (h) \, P \, B_{c,0}) \} \qquad (5.13)$$

The Single-fault hypothesis reads: *"At least one component among those used by the FT-FCS is working according to one of its fault modes, and there are no related faults"*, and is formalized as follows:

$$H_1 = H' \cap \{ h \in \Sigma \mid \exists c : C_c \in U \, (\exists m : 0 \leq m \leq f_c \, (O_c (h) \, P \, B_{c,m}) \, A \quad \forall k : C_k \in U \, A \, k \neq c \, A \, \textit{related}(C_c, C_k) \, (O_c (h) \, P \, B_{c,0}) \,) \} \qquad (5.14)$$

The Multiple-fault hypothesis reads: *"At least two related components among those used by the FT-FCS are working according to one of the corresponding fault modes"*, and is formalized as follows:

$$H_2 = H' \cap \{ h \in \Sigma \mid \exists c \, \exists k \, \exists n \, \exists m : C_c \in U \, A C_k \in U \, A \, \textit{related}(C_c, C_k) \, A \quad 0 \leq m \leq f_c \, A \, 0 \leq n \leq f_k \, A \, (O_c (h) \, P \, B_{c,m} \, A \, O_k (h) \, P \, B_{c,n}) \} \qquad (5.15)$$

The underlying fault hypothesis assumption states that manufacturer specifications of nominal and faulty behaviors of each component form the basis of the fault tolerance requirements (and of the resulting certification process), regardless of the fact that those specifications actually cover all possible behaviors of the components. To describe the fault hypotheses we refer to Table 5.1. Each column of the table represents a fallible component of the system; grouped columns represent related components. For the sake of space the table contains only the section related to the components used by the FT-FCS. Each row of the table represents a system operating condition; grouped rows correspond to the same fault hypothesis. The entries of the table represent the behavior of a component under a specified operating condition of the system. A zero entry indicates a nominal behavior, while a non-zero entry indicates a faulty behavior. For example, the operational behaviors of the roll rate gyro C_p under the system operating conditions $h_{0,0}$, $h_{1,0}$, and $h_{2,0}$ are $O_p(h_{0,0})$ P $B_{p,0}$, $O_p(h_{1,0})$ P $B_{p,1}$, and $O_p(h_{2,0})$ P $B_{p,2}$ respectively (see Section 5.4.2).

By referring to this table, the fault-free hypothesis is described by a row with all zero entries. The single-fault hypothesis is described by any row that has at least one non-

zero entry and does not have multiple non-zero entries in any section of related components. The multiple-fault hypothesis is described by any row that has at least two non-zero entries in one section of related components. We will refer to this table again when we will discuss the refinement of the fault hypotheses resulting from the introduction of the design constraints.

		$F \cap U$												
		M				A				P			I		
		C_p	C_q	C_r	..	C_{ail}	C_{elv}	C_{rud}	..	C_{IN}	C_{OUT}	..	C_{Wp}	C_{Wq}	..
H_0	$h_{0,0}$	0	0	0	..	0	0	0	..	0	0	..	0	0	..
	$h_{1,0}$	1	0	0	..	0	0	0	..	0	0	..	0	0	..
H_1	$h_{1,1}$	2	0	0	..	0	1	0	..	0	0	..	3	0	..
	$h_{1,2}$	0	0	0	..	0	0	1	..	0	3	..	0	0	..
	...	..	..	..	..	..	..	..	..	..	..	..	..	..	..
	$h_{2,0}$	1	1	0	..	0	0	0	..	0	0	..	0	0	..
H_2	$h_{2,1}$	2	0	0	..	0	1	0	..	0	0	..	3	1	..
	$h_{2,2}$	0	0	0	..	3	2	1	..	1	3	..	0	0	..
	...	..	..	..	..	..	..	..	..	..	..	..	..	..	..

Table 5.1 - System Operating Conditions

5.3.3.3 - Specification of Functional Requirements

Performance and *warning and status display* requirements from [MIL-F-9490D] are also specified in terms of relations. Each requirement is formulated in terms of maximal set of admissible aircraft state histories corresponding to specified pilot inputs. Requirements for each control functions are captured by a different relation: the relation R_{PAH} specifies the requirements for the Pitch Attitude Hold control function, the relation R_{ALH} specifies the requirements for the Altitude Hold control function, etc. A set S_{ao} specifies the constraints upon the operation of the control functions as set of admissible histories of control inputs. The whole set of performance requirements is specified through the relation R_P. This relation is obtained by joining the single relations as described by the following equation:

$$R_P = S_{ao} \setminus (R_{PAH} \vartheta R_{ALH} \vartheta R_{RAH} \vartheta R_{HH} \vartheta R_{HS} \vartheta R_{SBT} \vartheta R_{LF})) \qquad (5.16)$$

For the details of the above requirement the reader is referred to [DelGobbo00] (for an example see Section 5.4.1).

As for the *warning and status display* requirements, they cannot be made explicit without specifying the redundancy approach. In fact, these requirements constitute a mapping from the space of system operating conditions Σ to the space of warning and disengagement signals. The redundancy approach makes this requirement operational in the sense that the elements of the space Σ are mapped into a relation over system

quantities. These quantities along with warning and disengagement signals make up the signature of the *warning and status display* requirements. For example, by adopting the physical redundancy approach the outputs of the redundant elements constitute the quantities processed by the FTC system. The operating status of the system is mapped into a consistency relation among the redundant outputs. This relation is named RID(h) and the generic quantities carrying redundant information Q. The warning and status display requirements are then formalized as:

$$\begin{aligned} R_W = \{ (Q)(W, Y_d) \mid & \\ & \exists\, h : h \in H_0 \; A \; Q \in RID(h) \; (W = OFF \; A \; Y_d = OFF) \; O \\ & \exists\, h : h \in H_1 \; A \; Q \in RID(h) \; (W = \textit{warning}(h) \; A \; Y_d = OFF) \; O \\ & \exists\, h : h \in H_2 \; A \; Q \in RID(h) \; (W = \textit{warning}(h) \; A \; Y_d = ON) \; \} \end{aligned} \tag{5.17}$$

the term RID(h) and the quantity Q will be made explicit in the AR-FTC specification.

5.3.3.4 - Operational Requirements

The fail operational requirement reads: *"The DHC-2 aircraft equipped with the FT-FCS shall meet the performance specification under all fault-free and single-fault operating conditions of the system"*, and is formalized as follows:

$$\forall\, h \in H_0 \cup H_1 : DHC(h) \otimes FTFCS(h) \; P \; R_P \tag{5.18}$$

The warning requirement reads: *"Under fault-free operating conditions of the DHC-2 equipped with the FT-FCS the FT-FCS shall stay engaged and all warnings shall be off; under single-fault operating conditions of the DHC-2 equipped with the FT-FCS, the FT-FCS shall stay engaged and only warnings corresponding to faulty components shall be on; under multiple-fault operating conditions of the DHC-2 equipped with the FT-FCS, the FT-FCS shall disengage and only warnings corresponding to faulty components shall be on"*, and is formalized as follows:

$$\forall\, h \in H_0 \cup H_1 \cup H_2 : FTFCS(h) \; P \; R_W \tag{5.19}$$

5.3.3.5 - Introduction of Design Constraints

The specification represented by Eqs (5.18) and (5.19) represent the high-level, redundancy approach independent requirements for the FTC. By introducing the design constraints a refined specification is obtained. The refinement process is driven by the introduction of two design constraints. The first constraint requires the AR-FTC to make provision for faults within the measurement and the actuation subsystems by exploiting analytical redundancy. The assumption is made that the AR-FTC system is introduced in a system that already provides fault tolerance with respect to faults within the Processing and Interface subsystems implying that faults on components of these two subsystems are transparent to the AR-FTC. This constraint leads to a refinement of the fault hypotheses in order to eliminate system operating conditions containing faults on processing and

interface components. To formalize such constraint the following sets of system operating conditions is introduced:

$$H_P = \{ h \in \Sigma \mid \forall c : C_c \in P \cap U\ (O_c\ (h)\ P\ B_{c,0}) \} \quad (5.20)$$

$$H_I = \{ h \in \Sigma \mid \forall c : C_c \in I \cap U\ (O_c\ (h)\ P\ B_{c,0}) \} \quad (5.21)$$

Table 5.2 represents the two classes of sequences H_P and H_I. The 'x' entries stand for 'any component behavior'. The intersection between the sequences in Table 5.2 and those in Table 5.1 represent the set of system operating conditions relevant to the specification of the AR-FTC requirements.

	$F \cap U$												
	M				A				P			I		
	C_p	C_q	C_r	..	C_{ail}	C_{elv}	C_{rud}	..	C_{IN}	C_{OUT}	..	C_{Wp}	C_{Wq}	..
H_P	x	x	x	..	x	x	x	..	0	0	..	x	x	..
H_I	x	x	x	..	x	x	x	..	x	x	..	0	0	..

Table 5.2 - Refinement of Fault Hypotheses

The constraint over the redundancy approach leads to a second refinement of the FT-FCS specification since it allows specifying the redundancy relation RID(h) and the quantity Q within the warning requirement R_W. The quantities carrying redundant information and providing redundant action upon the system state are the software variables representing sensor outputs X and actuator inputs U_c. These are the quantities processed by the AR-FTC to identify the system operating condition and to provide required fault tolerance. The couple (U_c , X) is checked against the consistency relations captured by $RID_{AR}(h)$:

$$RID_{AR}(h) = OUT(h) \otimes AS(h) \otimes DHC(h) \otimes MS(h) \otimes IN(h) \quad (5.22)$$

By substituting RID(h) with $RID_{AR}(h)$ and Q with (U_c, X) in Equation 17, and by projecting the output signals W and Y_d onto the AR-FTC output W and Y_d the warning requirement constrained by adopting the analytical redundancy approach is obtained:

$$\begin{aligned} R_{WAR} = \{ &((U_c, X), (W, Y_d)) \mid \quad (5.23)\\ &\exists h : h \in H_0\ A\ (U_c, X)\ \in RID_{AR}\ (h)\ (W = OFF\ A\ Y_d = OFF)\ O\\ &\exists h : h \in H_1\ A\ (U_c, X)\ \in RID_{AR}\ (h)\ (W = \textit{warning}(h)\ A\ Y_d = OFF)\ O\\ &\exists h : h \in H_2\ A\ (U_c, X)\ \in RID_{AR}\ (h)\ (W = \textit{warning}(h)\ A\ Y_d = ON) \} \end{aligned}$$

The second constraint requires the AR-FTC system to be modular with respect to Measurement and Actuation components. That is, two different FTC systems, denoted FTC_M and FTC_A, must provide fault tolerance each with respect to faults within one of the two subsystems. This constraint leads to a splitting of the performance requirements R_P into two separate requirements R_{OBS} and R_{ACT}. R_{OBS} specifies the *observation*

capability that the FT-FCS should provide, while R_{ACT} specifies the *actuation* capability it should provide. R_{OBS} and R_{ACT} result from the decomposition of the FCS in Figure 5.2:

$$R_{OBS} = MS \otimes IN \qquad (5.24)$$
$$R_{ACT} = FCL \otimes OUT \otimes AS \otimes DHC$$

From the correctness of the FCS design it results:

$$R_{OBS} \otimes R_{ACT}\ P\ R_P \qquad (5.25)$$

The relations σ_A and σ_M are used to project the requirements on the relevant set of warning signals for components of the Actuation subsystem and the Measurement subsystem respectively. The fail-operational requirements for the AR-FTC monitoring the measurement (sensor) subsystem reads: "*The FTC_M shall process software variables representing outputs from the Measurement subsystem and inputs to the Actuator subsystem to produce a set of validated software measurement variables that allows meeting the requirements for the observation subsystem R_{OBS}. The above function shall be performed under all fault-free and single-fault operating conditions of the DHC-2 equipped with the FT-FCS, and under the assumption of correct operation of Processing and Interface components*", and is formalized as follows:

$$\forall\ h \in (H_0 \cup H_1) \cap H_P \cap H_I: MS(h) \otimes IN(h) \otimes FTC_M(h)\ P\ R_{OBS} \qquad (5.26)$$

The warning requirements for the AR-FTC monitoring the measurement (sensor) subsystem reads: "*Under fault-free operating conditions of the DHC-2 aircraft equipped with the FT-FCS the FTC_M shall not cause disengagement of the FT-FCS and all warnings corresponding to faulty components of the measurement subsystem shall be off; under single-fault operating conditions of the DHC-2 aircraft equipped with the FT-FCS, the FTC_M shall not cause disengagement of the FT-FCS and warnings corresponding to faulty components of the measurement subsystem shall be on; under multiple-fault operating conditions of the DHC-2 aircraft equipped with the FT-FCS, the FTC_M shall cause disengagement of the FT-FCS and warnings corresponding to faulty components of the measurement subsystem shall be on*", and is formalized as follows:

$$\forall\ h \in (H_0 \cup H_1 \cup H_2) \cap H_P \cap H_I: FTC_M(h)\ P\ R_{War} \Pi \sigma_M \qquad (5.27)$$

5.4 – Case Studies

5.4.1 – Case Study #1: Heading Hold (HH) System

This section reports the formal specification of the Heading Hold (HH) control function requirement, along with related tables and some observations on the formalization process. The HH requirement is fairly simple, yet provides a number of meaningful points for discussion. The original formulation of the requirement from [MIL-F-9490D] is reported below:

3.1.2.2 Heading Hold

"In smooth air, heading shall be maintained within a static accuracy of +/- 0.5 degree with respect to the reference. In turbulence, RMS deviations shall not exceed 5 degrees in heading at the intensities specified in 3.1.3.7. When heading hold is engaged, the aircraft shall roll towards wings level. The reference heading shall be that heading that exists when the aircraft passes through a roll attitude that is wings level plus or minus a tolerance."

The HH requirement specifies accuracy requirements for the heading angle for operation in both smooth air and turbulence when the HH function is engaged. By analyzing the requirement we identify the quantities that are used within the specification; hence, we assign mathematical variable to each quantity. The mapping between the real quantities referred to in the natural language specification and the entities used in the formal one bridge the gap between the two specifications. This mapping is collected in tables where each variable and predicate used in the relations is precisely – though informally – defined to lead to a unique interpretation in the domain of flight control systems. Furthermore, these tables collect valuable information such as data-type, admissible value ranges, metric units. Tables 5.3 trough 5.6 list all mathematical entities used within the HH relational specification, and separate them into constants, domain, image, and quantified variables, and auxiliary functions. The signature and the predicate of the HH relational specification are given by the following two equations:

$$HH_{sign} = ((SW_{HH}(),u_{wt}(),v_{wt}(),w_{wt}()),(\psi(),\phi())) \tag{5.28}$$

$$\begin{aligned} HH_{pred} = {} & \forall t_1\ \forall t_3\ (\ 0 \le t_1 < t_3 < \infty \wedge \text{engaged}\ (SW_{HH}(), t_1, t_3) \Rightarrow \\ & \exists\ t_2\ \exists\ \psi_r (t_1 \le t_2 < t_3\ \wedge \forall\ t\ (\ t_2 \le t \le t_3 \Rightarrow |\phi(t)| < \phi_{acc}) \wedge \psi_r = \psi(t_2) \wedge \\ & \neg\ \text{turb}(t_1, t_3) \Rightarrow \forall t\ (t_2 \le t \le t_3 \Rightarrow | \psi(t) - \ \psi_r | < \psi_{acc}\ ) \wedge \\ & \text{turb}(t_1,t_3) \Rightarrow \text{RMS}(\psi() - \psi_r, t_2, t_3) < \ \psi_{RMS}\ )\ ) \end{aligned} \tag{5.29}$$

Hence, the relational specification for the HH control function requirement is:

$$R_{HH} = \{\ HH_{sign} \mid HH_{pred}\ \} \tag{5.30}$$

In this study we focused on the specification of the requirements within the framework of relational algebra, rather than on developing a set of complete and feasible requirements. Nevertheless, it is worth pointing out that we found a number of flaws within the original formulation of the requirements. Whenever possible we modified the requirements to correct them. An example from the HH control function requirement is in the paragraph: "When heading hold is engaged the aircraft shall roll towards wings level". This paragraph has been formalized by requiring the airplane to reach -- at a certain time instant t_2 -- a state with a bank angle *close* to zero. To quantify *how close*, we adopted the accuracy threshold ϕ_{acc} used in the Roll Attitude Hold specification (Section 3.1.2.1 of [MIL-F-9490D]). This solution is our best guess to make up for the lack of a threshold in the original requirement. Another problem that we found in the requirements of the autopilot functions is the lack of any indication of the transient time between autopilot engagement and the instant when the steady-state condition is reached. In the

case of the Heading Hold requirement the transient interval is $[t_1, t_2]$. The requirement applies only if this interval is larger than the interval Δt that it will take the aircraft to reach wing level attitude. Δt is function of the roll angle $\phi(t_1)$, the roll rate $d\phi/dt(t_1)$, and the roll rate capability of the aircraft in the current point of the flight envelope. Determination of Δt was out of the scope of our study, so we limited ourselves to specify the original requirement and report the problem.

ID and value	Type	Definition
$\psi_{acc} = 0.5$ degrees	Angle-T	Heading accuracy in smooth air
$\psi_{RMS} = 5$ degrees	Angle-T	RMS heading accuracy in turbulence
$\phi_{acc} = 1.0$ degrees	Angle-T	Roll accuracy in smooth air

Table 5. 3 - Constants

ID	Type	Definition
$SW_{HH}()$	Time-T → Switch-T	HH autopilot on/off switch
$u_{wt}(),v_{wt}(),w_{wt}()$	Time-T → Velocity-T	Turbulence components of wind velocity along body- axes
$u_{wg}(),v_{wg}(),w_{wg}()$	Time-T → Velocity-T	gust components of wind velocity along body-axes
$\psi()$	Time-T → Angle-T	Heading angle
$\phi()$	Time-T → Angle-T	Bank angle

Table 5. 4 - Domain and image variables

ID	Definition
engaged (SW(), t_a, t_b)	Predicate that holds true only if the switch SW() is engaged at t= t_a and stays engaged throughout the time interval $[t_a, t_b]$
RMS(f(), t_a, t_b)	Root Mean Square value of the function f() over the time interval $[t_a, t_b]$
turb(t_a,t_b)	Predicate that holds true if random and discrete turbulence wind components are not zero in the time interval $[t_a, t_b]$

Table 5.5 - Predicates and functions

ID	Type	Definition
t	Time-T	Generic time instant
t_1	Time-T	HH function engagement instant
t_2	Time-T	Time instant when the reference heading is determined
t_3	Time-T	Time instant delimiting the scope of the requirement
ψ_r	Angle-T	Reference heading

Table 5.6 - Quantified variables

5.4.2 – Case Study #2: Roll Rate Gyro Failure

In this section we report the specification of the roll rate gyro to illustrate some details on the specification of the system component's behaviors. We report the specification of the steady state behavior of the sensor along with three fault modes. The fault mode behaviors have been modeled by analyzing the sensor output after causing the failure. Three different fault modes have been considered: loss of power connection, loss of ground connection, loss of signal connection. Eq's. 5.31 through 5.34 report the corresponding relations. Each relation specifies the behavioral envelope of the sensor under the corresponding operating condition. The mapping between system operating conditions and the corresponding component behavioral envelopes is specified by Table 5.1. The table entry identified by the row marked $h_{0,0}$ and the column marked C_p is '0', implying that the behavioral envelope of the roll rate gyro (identified by C_p) under the system operating condition $h_{0,0}$ is $B_{p,0}$. Analogously, the behavioral envelope of the roll rate gyro for the system operating condition $h_{1,0}$ is $B_{p,1}$ since the corresponding table entry is '1'. The actual behavior of the sensor, while operating within the FCS, is represented by the relation $O_p(h)$. If the system operating condition is $h_{0,0}$ then the roll rate actual behavior is an instance of the corresponding behavioral envelope $B_{p,0}$; that is, $O_p(h_{0,0})$ P $B_{p,0}$.

Behavioral envelope 0 (nominal operation)

$$B_{p,0} = \{ (p, \tilde{p}) \mid \forall t \exists S_p \exists bias_p \exists v_p() (0 \le t < \infty \wedge S_g^- \le S_p \le S_g^+ \wedge BIAS_g^- \le bias_p \le BIAS_g^+ \wedge whiteNoise(v_p(), Npsd_g) \Rightarrow \tilde{p}(t) \approx [S_p[p(t)]_{IR_g^-}^{IR_g^+} + bias_p + v_p(t)]_{OR_g^-}^{OR_g^+} \} \quad (5.31)$$

Behavioral envelope 1 (fault mode corresponding to loss of power connection)

$$B_{p,1} = \{ (p, \tilde{p}) \mid \forall t \exists v_p() (0 \le t < \infty \wedge whiteNoise(v_p(), Npsd_g) \Rightarrow \tilde{p}(t) \approx [v_p(t)]_{OR_g^-}^{OR_g^+} \} \quad (5.32)$$

Behavioral envelope 2 (fault mode corresponding to loss of ground connection)

$$B_{p,2} = \{ (p, \tilde{p}) \mid \forall t \exists v_p() (0 \le t < \infty \wedge \text{whiteNoise}(v_p(), \text{Npsd}_g) \Rightarrow \quad (5.33)$$
$$\tilde{p}(t) \approx [V_{cc} + v_p(t)]_{OR_{\bar{g}}}^{OR_g^+} \}$$

Behavioral envelope 3 (fault mode corresponding to loss of signal connection)

$$B_{p,3} = \{ (p, \tilde{p}) \mid \forall t \exists v_p() (0 \le t < \infty \wedge \text{whiteNoise}(v_p(), 0.25) \Rightarrow \quad (5.34)$$
$$\tilde{p}\ (t) \approx [\, v_p(t)]_{OR_{\bar{g}}}^{OR_g^+} \}$$

5.5 - Conclusions

The formulas in the previous sections are an excerpt of the specification for the Analytical Redundancy Based Fault Tolerance Capability for sensor failures in an automatic flight control system. The development of the specification brought to light a number of issues, some of which are still left unsolved in the literature.

Prior to adopting relational algebra an attempt was made to adopt SCR (Software Cost Reduction) encouraged by the mature development environment and the rich set of automatic tools that support the SCR specification language; results of this effort are documented in [DelGobbo98], [DelGobbo99], [Cortellessa00], and [Alexander00]. A number of issues prevented the SCR specification from ever reaching a mature status. SCR transition-based paradigm and the intrinsic open-system approach of an SCR specification were not suitable for formulating the requirements we considered. The lack of structuring mechanisms made it difficult to build the system specification from the specification of its parts. The weak support of time modeling and the rigid specification model forced the elaboration of artificial solutions -- coding tricks – to formulate the requirements, thus showing poor expressiveness of the language for our application. Most of the above issues arise from the poor compatibility between SCR and this particular application, rather than from weaknesses in SCR itself.

As opposed to SCR, relational algebra does not come with automatic tools; on the other hand it provides:

- a suitable reasoning framework, within which it is possible to specify and reason about the various aspects of the system requirements,
- capability to specify requirements that are not formulated as explicit description of system inputs/outputs constraints,
- a flexible notation that allows representing the structure of the system,
- means to represent functional as well operational requirements.

References for Section #5

[AFFDL-TR-74-116] Background information and user guide for MIL-F-9490D. Technical report AFFDL-TR-74-116, USAF, 1975.

[MIL-F-9490D] Flight control systems - design, installation and test of piloted aircraft, general specification for. Military specification MIL-F-9490D, USAF, 1975.

[AFFDL-TR-74-116sup1] Appendix to background information and user guide for MIL-F-9490D. Technical report AFFDL-TR-74-116 sup1, USAF, 1980.

[MIL-F-8785C] Flying qualities for piloted airplanes. Technical Report MIL-F-8785C, USAF, 1980.

[AFWAL-TR-81-3109] Background information and user guide for MIL-F-8785C. Technical report AFWAL-TR-81-3109, USAF, 1982.

[Alagar98] Alagar V.S. and Periyasamy K., Specification of Software Systems, Springer Verlag, 1998.

[Alexander00] Alexander C., Cortellessa V., Del Gobbo D., Mili A., and Napolitano M., "Modeling the Fault Tolerant Capability of a Flight Control System: An Exercise in SCR" Fifth NASA Langley Formal Methods Workshop, Williamsburg, Virginia, June 13-15, 2000.

[Basseville88] M. Basseville. Detecting changes in signals and systems - a survey. Automatica, 24:309--326, 1988.

[Chen99] J.Chen and R.J. Patton. Robust Model-Based Fault Diagnosis for Dynamic Systems. Kluwer Academic Publishers, 1999.

[Chow80] E.Y. Chow and A.S. Willsky. Issues in the development of a general algorithm for reliable failure detection. Proc. of the 19th Conf. on Decision and Control, 1980.

[Cortellessa00] Cortellessa V., Cukic B., Mili A., Shereshevsky M., Sandhu H., Del Gobbo D., and Napolitano M., "Certifying Adaptive Flight Control Software", ISACC, 2000.

[DelGobbo98] Del Gobbo D., Napolitano M., Callahan J., and Cukic B., "Experience in Developing System Requirements Specification for a Sensor Failure Detection and Identification Scheme", Proc. of the Third IEEE High Assurance System Engineering Symposium, Washington D.C., Nov. 13-14, 1998.

[DelGobbo99] Del Gobbo D., Cukic B., Easterbrook S., and Napolitano M., "Fault Detectability Analysis for Requirements Validation of Fault Tolerant Systems", Proc. of the Fourth IEEE High Assurance System Engineering Symposium, Washington D.C., Nov. 17-19, 1999.

[DelGobbo00] D. Del Gobbo Formal Specification of Requirements for Analytical Redundancy based Fault Tolerant flight Control Systems. Ph.D. dissertation, West Virginia University, MAE Dept. Dec. 2000

[Feather87] M.S. Feather Language support for the specification and development of composite systems. ACM Transactions on Programming Languages and Systems. Vol. 9 No 2, April 1987, p.198-234

[Favre94] C.Favre, Fly-by-wire for commercial aircraft: The airbus experience. International Journal of Control, 59:139--157, 1994.

[Hoak68] D.E. Hoak, D.E. Ellison, et al. USAF Stability and Control DATCOM. Flight Control Division, Air Force Flight Dynamics Laboratory; Wright-Patterson Air Force Base, Ohio., 1968.

[Leveson94] N.G.~Leveson, M.~Heimdahl, H.~Hildreth, and J.D.~Reese. Requirements Specification for Process-Control Systems, IEEE Transactions on Software Engineering, Vol.20, No.9, Sept. 1994; p.684-707.

[Rauw-98] Rauw M. FDC 1.2 - A Simulink toolbox for flight dynamics and control analysis - user manual. http://www.mathworks.com/, 1998.

[Rauw-93] Rauw M.O. A Simulink environment for flight dynamics and control analysis - application to the DHC-2 'beaver'. Graduate's thesis. Delft Univ. of Thechnology, the Netherlands, 1993.

[Parnas95] D.L.~Parnas and J.~Madey. Functional Documentation for Computer Systems, Science of Computer Programming, Vol.25, No.1 October 1995, p.41-61.

[Patton94] R.J. Patton. Robust model-based fault diagnosis: The state of the art. IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes - SAFEPROCESS '94, 1:1--24, 1994.

[Patton00] R.J. Patton, P.M. Frank, and R.N.Clark (Eds.). Issues of Fault Diagnosis for Dynamic Systems. Springer Verlag, 2000.

[Patton89] Patton R.J., Frank P., and Clark R. Fault Diagnosis in Dynamic Systems, Theory and Applications. Prentice Hall, 1989.

[Tjee88] Tjee R.T.H. and Mulder J.A. Stability and control derivatives of the De Havilland DHC-2 'beaver' aircraft. Report LR-556, Delft Univ. of Technology, The Netherlands,1988.

[Schmidt91] G.Schmidt and T.Strohlein. Relations and Graphs. Springer Verlag, 1993.

[Szalai80] K.J. Szalai, R.R. Larson, and R.D. Glover. Flight experience with flight control redundancy management. In AGARD Lecture Series No.109. Fault Tolerance Design and Redundancy Management Techniques, AGARD Lecture Series, pages 8/1--27, AGARD, Neuilly-sur-Seine, France, October 1980. AGARD.

[Willsky76] A. S. Willsky. A survey of design methods for failure detection in dynamic systems. Automatica, 12:601--611, 1976.

[Yeh-AAC96] Y.C. Yeh. Triple-triple redundant 777 primary flight computer. IEEE Aerospace Applications Conference, February 1996.

Section #6

Flight testing results of the application of the fault tolerant schemes.

Section #6 – Table of Contents

List of Symbols (Section #6)

English

p	Aircraft angular velocity around the x body axis (roll rate), rad/sec
q	Aircraft angular velocity around the y body axis (pitch rate), rad/sec
r	Aircraft angular velocity around the z body axis (yaw rate), rad/sec

Greek

α	Angle of attack, rad or deg
β	Angle of sideslip, rad or deg
θ	Pitch Euler angle, rad or deg
δ	Control surface deflection, rad or deg
ϕ	Roll Euler angle, rad or deg
ψ	Yaw Euler angle, rad or deg

Subscripts

A	Aileron
E	Elevator
L	Left side
R	Right side
R	Rudder

Acronyms

AC	Automatic Control
AFA	Actuator Failure Accommodation
AFDI	Actuator Failure Detection and Identification
AFDIA	Actuator Failure Detection, Identification, and Accommodation
EBPA	Extended Back Propagation Algorithm
DNN	Decentralized Neural Network
DQEE	Decentralized Quadratic Estimation Error
FDI	Failure Detection and Identification
GCU	Ground Control Unit
MC	Manual Control
MLP	Multi-Layer Perceptron
MNN	Main Neural Network
MQEE	Main Quadratic Estimation Error
NNC	Neural Network Controller
OBC	On-Board Computer
OQEE	Output (of NN) Quadratic Estimation Error
PID	Parameter Identification
PTBU	(Neural) Parameters To Be Updated (at each computational step)
PWM	Pulse Width Modulation
RTW	Real Time Workshop
SFA	Sensor Failure Accommodation
SFDI	Sensor Failure Detection and Identification
SFDIA	Sensor Failure Detection, Identification, and Accommodation

6.1. – Flight Testing of the SFDIA Scheme

6.1.1. – Definition of the SFDIA Flight Testing

Sensor Failure Detection, Identification, and Accommodation (SFDIA) scheme
Sub-scheme 1 – SFDIA following pitch rate gyro failure (for different failures),
Sub-scheme 2 – SFDIA following yaw rate gyro failure (for different failures),
Sub-scheme 3– SFDIA following roll rate gyro failure (for different failures).
Requiring:

- MNN, "replicating" p,q,r;
- p_DNN ("replicating" p), q_DNN ("replicating" q), r_DNN ("replicating" r)

6.1.2. – Design of the SFDIA Maneuver

The description is provided for pitch rate gyro failures; identical flight profiles were designed for failures involving roll rate and yaw rate gyros. The aircraft is in manual control through the GCU. A tentative flight pattern is shown in Figure 6.1 while the entire SFDIA maneuver is summarized in Figure 6.2.

Step #0
Take off.

Step #1
Turn on the on-line learning for the MNN, the q_DNN, the p_DNN, and the r_DNN (on-line approximators) at nominal conditions as soon as the plane reaches altitude.

Step #2
With MNN and DNNs learning on, the pilot performs mild rolling, pitching, and yawing maneuvers with the goal of "fine tuning" on-line the learning of the MNN and DNNs.

Step #3
At point A in the flight path (see Figure 6.1), in the middle of a mild pitch, the pilot triggers the failure of a pitch rate gyro.

NOTE :
Due to the pitch rate gyro failure, q from the sensor and q from the MNN and q_DNN outputs should be substantially different. Thus the MNN parameter "MQEE" and the q_DNN parameter '$DQEE_q$' should have peaks exceeding the "a-priori" selected detection and identification thresholds. If the detection and the identification are both positive, according to the logic outlined in Section #4, the output of the q_DNN functionally replaces the measurement from the gyro. From that point on the q_DNN output should follow the nominal value (without the failure) of the pitch rate gyro.

Step #4
At point B restore nominal conditions and "remove" the sensor failure. The learning for the MNN and the DNNs is still turned on.

Step #5
At point C turn the MNN and DNNs learning off.

Step #6
Landing. Data downloading concludes the flight.

SFDIA - Pitch Gyro Failure - Flight Path

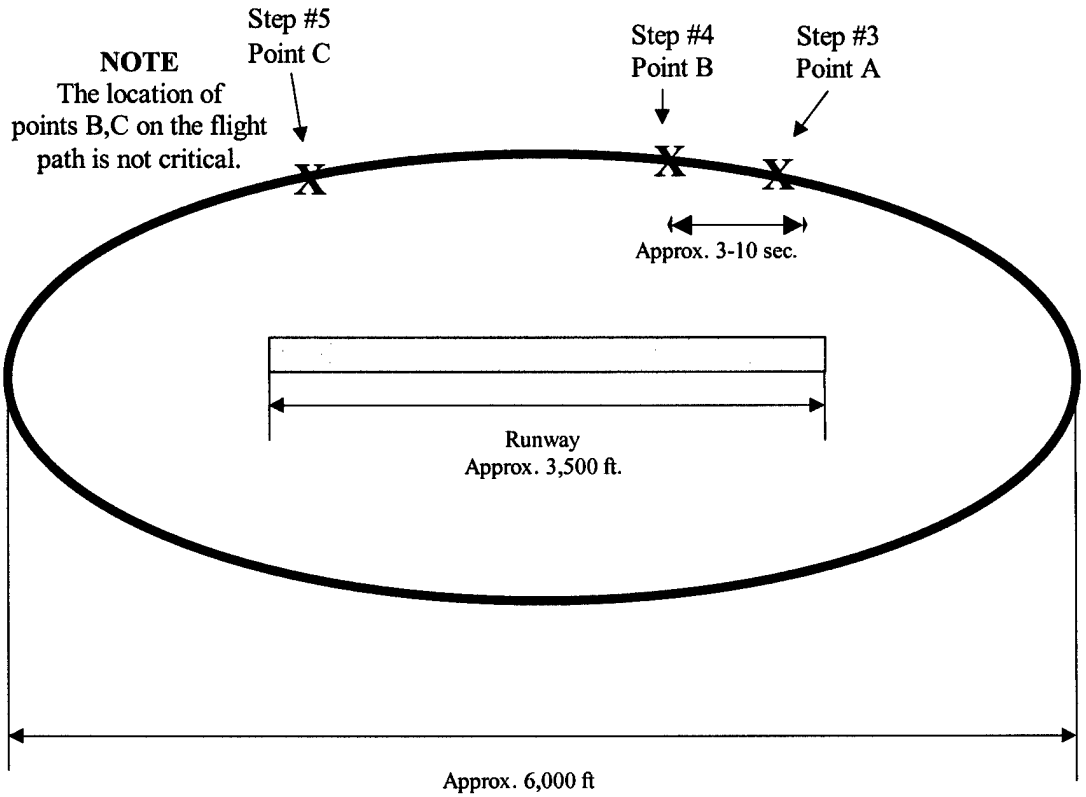


Figure 6.1 – Flight path of the SFDIA (pitch gyro failure)

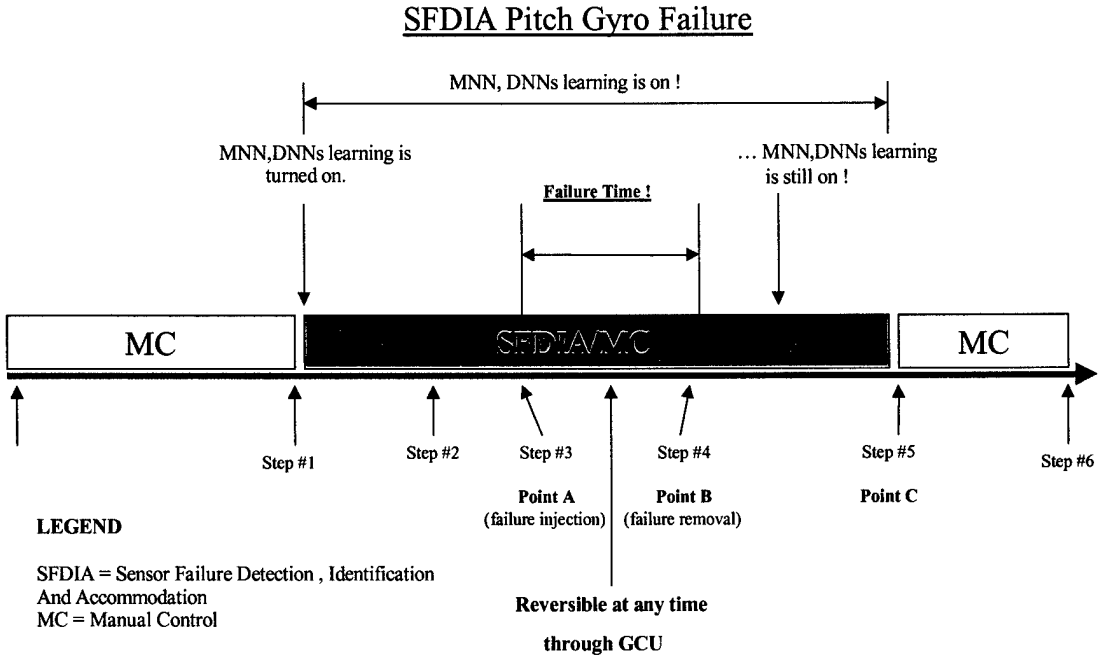


Figure 6.2 –Bar-coded task sequence for SFDIA (pitch gyro failure)

6.1.3. – Description of the SFDIA software

The SFDIA scheme has been described with details in Section #4. The scheme was developed and tested in a Matlab/Simulink environment prior to being interfaced on the OBC through the use of Real Time Workshop (RTW) utility in Matlab. The general Simulink SFDIA scheme is shown in Figure 6.3 below.

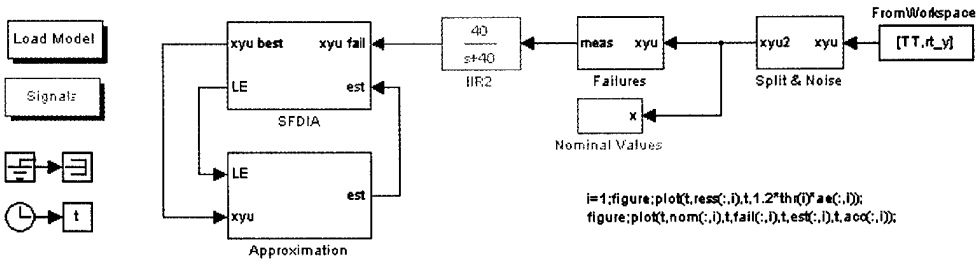


Figure 6.3 – General Simulink SFDIA scheme

The PID flight data collected within the 7 flights of Phase #5 were used for the development of the SFDIA scheme. In fact, the data allowed performing parametric studies toward the selection of "sub-optimal" architectures for the different neural estimators. Although only the p_DNN, the q_DNN, and the r_DNN were required for this SFDIA effort, additional DNNs estimating other dynamic parameters were also evaluated. The Simulink scheme with the different DNNs is shown in Figure 6.4 while the selected architectures for the different neural approximators are shown in Table 6.1.

	MNN	p_DNN	q_DNN	r_DNN
Input parameters (at previous instants)	p, q, r, δ_E, δ_A, δ_R,az,Φ,Θ	r,δ_A, δ_R, ay	δ_E,a_z,V	p, δ_A, δ_R, ay,Φ
Output (current instant)	p,q,r, estimates	p estimate	q estimate	r estimate
Input data pattern	3	3	3	3
Total number of inputs	27	12	9	15
Number of hidden layers	1	1	1	1
Number of neurons in The hidden layer(s)	10	5	3	6
Final neural configuration	27/10/3	12/5/1	9/3/1	15/6/1
Learning rate coefficient	0.05	0.05	0.05	0.05
Momentum coefficient	0	0	0	0
Total number of neural parameters updated at each step	339	83	42	87

Table 6.1 – Architectures of the neural approximators in the SFDIA scheme

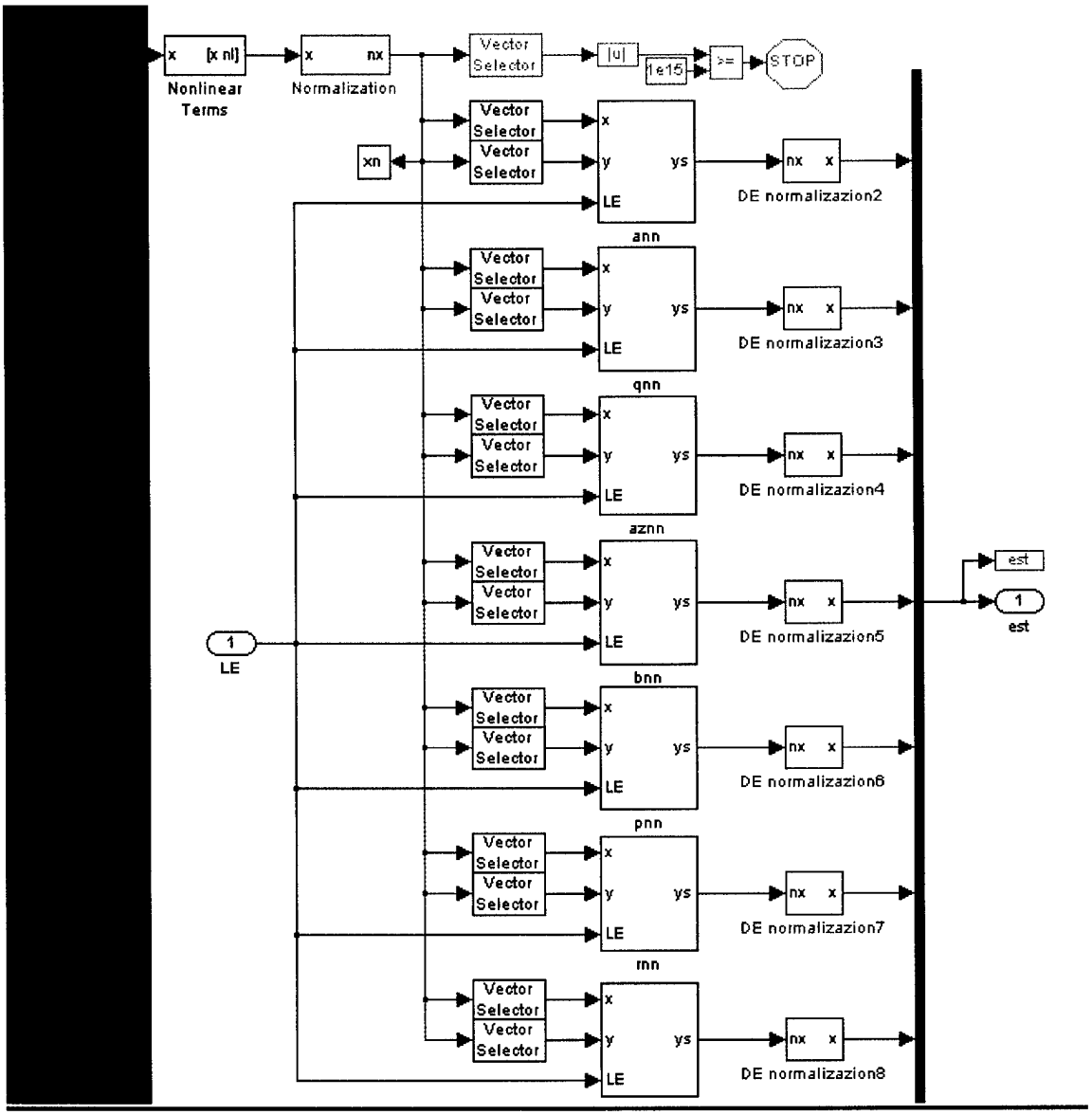


Figure 6.4 – Simulink scheme with the different DNNs

Following the selection of the architectures, the neural estimators described in Table 6.1 were simultaneously trained off-line through 10 iterations of the entire set of PID flights with a total of 76 different PID maneuvers.

6.1.4. – Results of the SFDIA (SFA) Flight Testing

A total of 7 flights were conducted with the YF-22 within Phase #5 with the purpose of collecting flight data for PID purposes and for performing the off-line training for the neural estimators (MNN, p_DNN, q_DNN, and r_DNN) within the SFDIA scheme. Due to time constraints no SFI and SFDI flight tests were conducted; the flight-testing focused directly on the general SFDIA scheme. A total 3 SFDIA flights were conducted within Phase #6 with each flight dedicated to the failure of one sensor (1st SFDIA flight for 'p' failure, 2nd SFDIA flight for 'q' failure, and 3rd SFDIA flight for 'r' failure). In particular, a soft and a hard failure were "injected" on each gyros at a certain

instant t_f by the pilot through the GCU; the failures were then "removed" through the GCU after 20-30 sec. To evaluate the SFDIA performance the following parameters have been introduced to characterize off-line the SFDIA (SFA) performance.

- T_{LE}: instant in which the NN learning is stopped;
- T_{AE}: instant in which the sensor fault is declared. In other words this instant represents the "estimation" of the instant t_f.
- *%LE*: it quantifies the percentage of time the NN learning is preventively stopped due to noise or mapping error before the real occurrence of the fault

$$\%LE_o = \frac{1}{k_f}\sum_{k=0}^{k_f}\left(1 - LE_0(k)\right) \tag{6.1}$$

- *%AE*: it quantifies the percentage of time the a sensor is incorrectly declared failed before the real occurrence of the fault:

$$\%AE_o = \frac{1}{k_f}\sum_{k=0}^{k_f}\left(1 - AE(k)\right) \tag{6.2}$$

- *N-False*: number of false alarm prior of the sensor failure declaration
- *Detection Ratio (DR)*: it is the ratio between the peak of the filtered residual SM(k) caused by the fault, and the peak of the same signal prior the occurrence of the fault:

$$Detection\ Ratio = \frac{\max\limits_{k_f \le k < k_f + k_D}\left(abs\left[S2(k)\right]\right)}{\max\limits_{0<k<k_f}\left(abs\left[S2(k)\right]\right)} \tag{6.3}$$

where $T \bullet k_D$ represents an extra delay time after t_f chosen to capture the entire spike of the residual caused by the fault.

To evaluate the effectiveness of the sensor failure accommodation (SFA) the following parameters are evaluated after the accommodation:

- *MEE, STDEE, and POWEE:* they represent respectively the mean, variance, and power of the estimation error sequence after enabling of the sensor accommodation.

$$\begin{aligned} MEE &= \frac{1}{(k_{end} - k_f)}\sum_{k=k_f}^{k_{end}} y(k) - \hat{y}(k) \\ STDEE &= \sqrt{\frac{1}{k_{end} - k_f}\sum_{k=k_f}^{k_{end}}\left[\left(y(k) - \hat{y}(k)\right) - MEE\right]^2} \\ POWEE &= MEE^2 + STDEE^2 \end{aligned} \tag{6.4}$$

Essentially these parameters measure the effectiveness of the NN approximator in reproducing the physical parameter at nominal conditions.

Examples of the flight-testing results are shown below. Figures 6.5-6.7 show time histories at nominal flight conditions to highlight the accuracy of the off-line training for each of the p_DNN, q_DNN, and r_DNN (with the data collected from Phase #5 for 76 different PID maneuvers). The SFDIA results are shown in Figures 6.8-6.13 for a soft failure (Type #3 described in Section #3), and in Figures 6.14-6.19 for an hard failure (Type #1 described in Section #3) for each of the roll, pitch, and yaw gyros. The soft failure (Type #3) is modeled as a 5 sec ramp to a +5 deg/sec bias while the hard failure (Type #1) is a quasi-instantaneous +5 deg/sec bias.

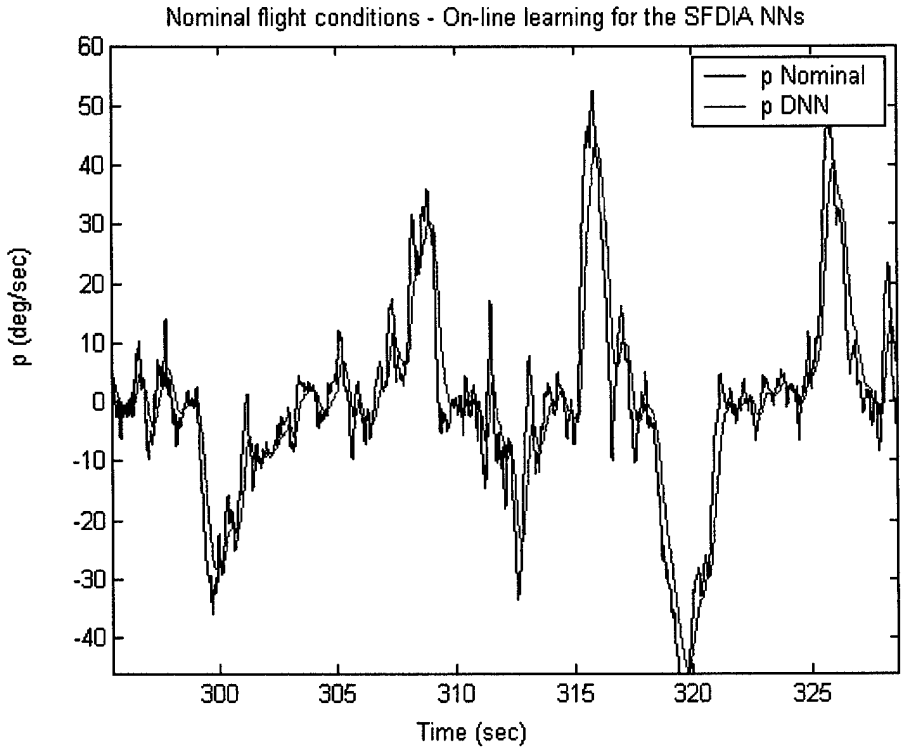


Figure 6.5 – Nominal flight conditions. Learning of the p_DNN

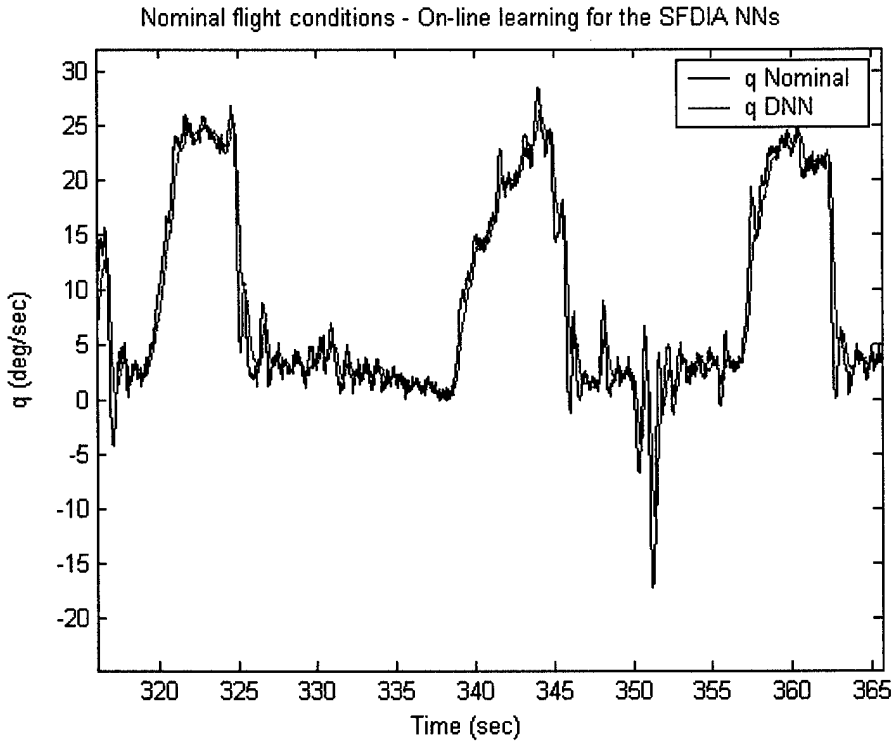


Figure 6.6 – Nominal flight conditions. Learning of the q_DNN

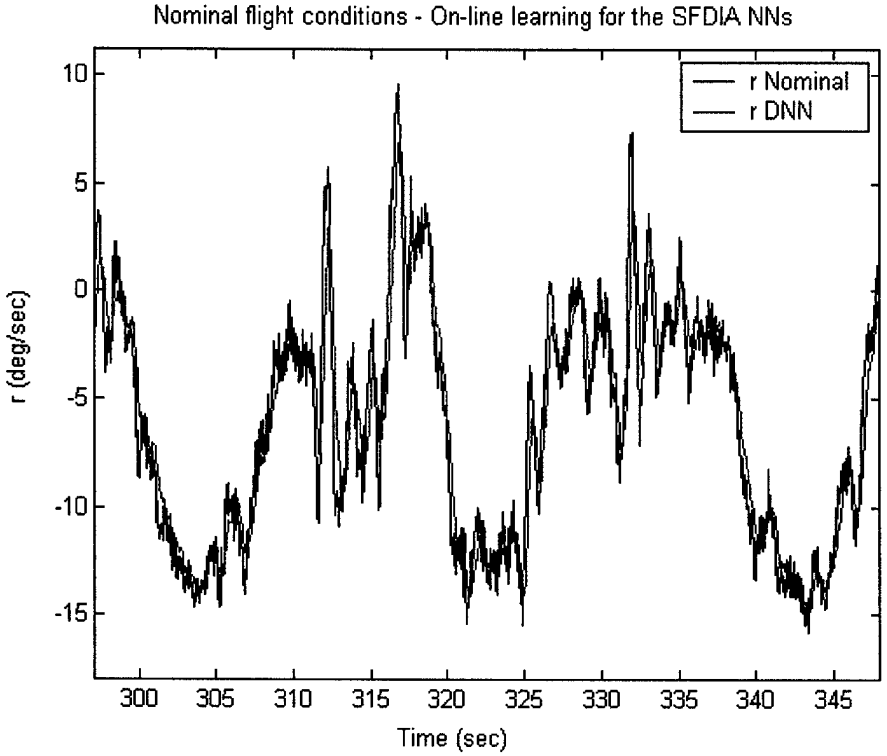


Figure 6.7 – Nominal flight conditions. Learning of the r_DNN

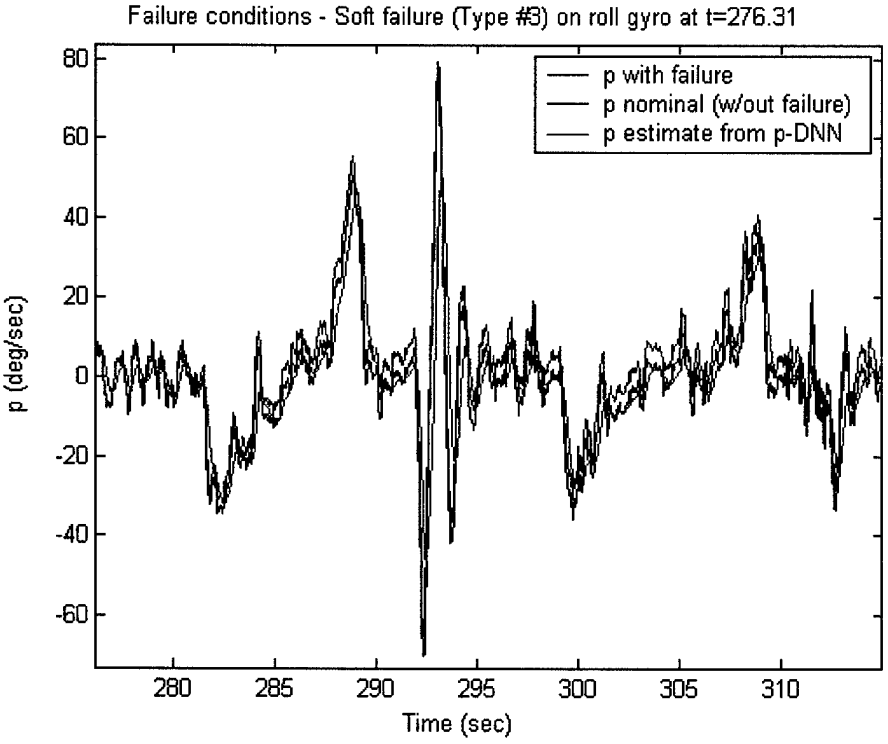


Figure 6.8 – 1st SFDIA flight. Soft failure (Type #3) for roll gyro.

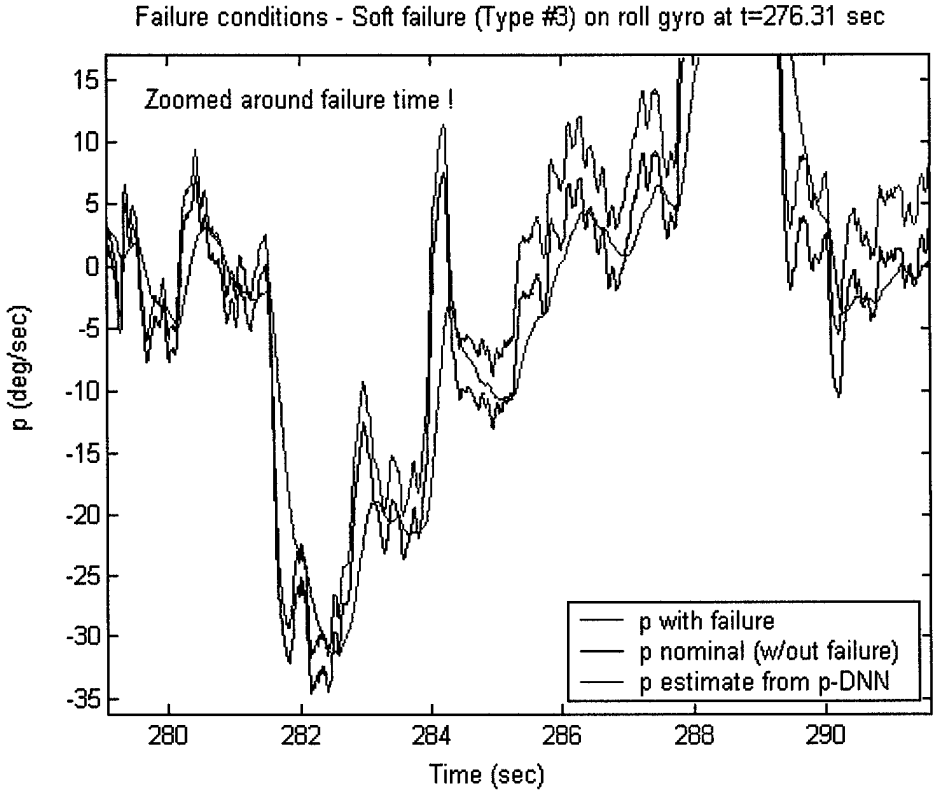


Figure 6.9 – 1st SFDIA flight. Soft failure (Type #3) for roll gyro (ZOOMED).

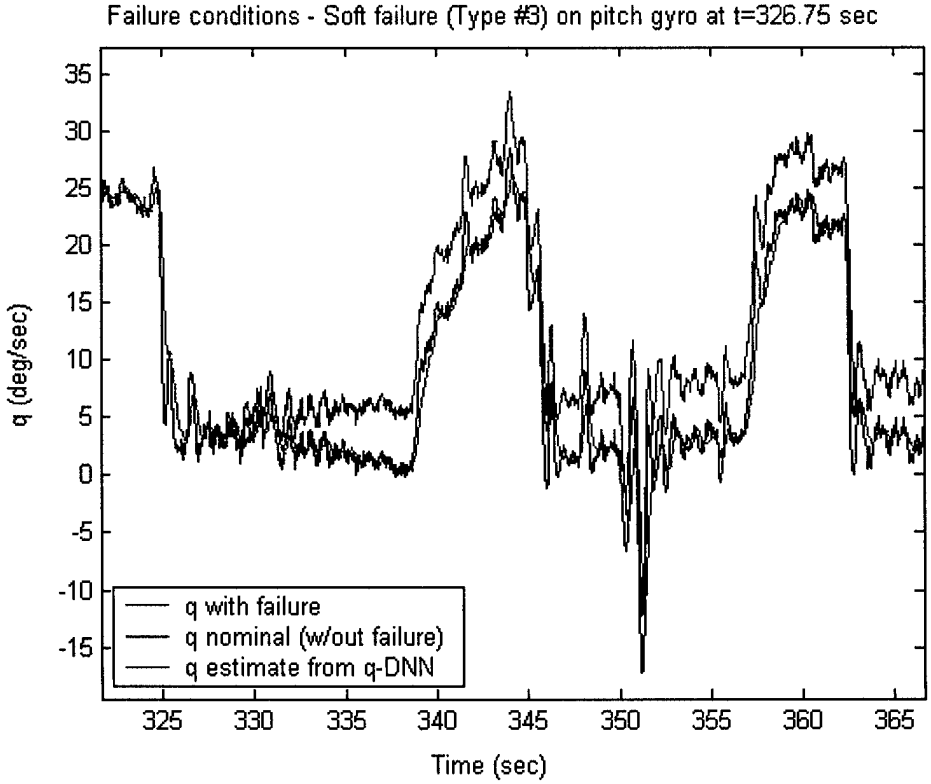


Figure 6.10 – 2nd SFDIA flight. Soft failure (Type #3) for pitch gyro.

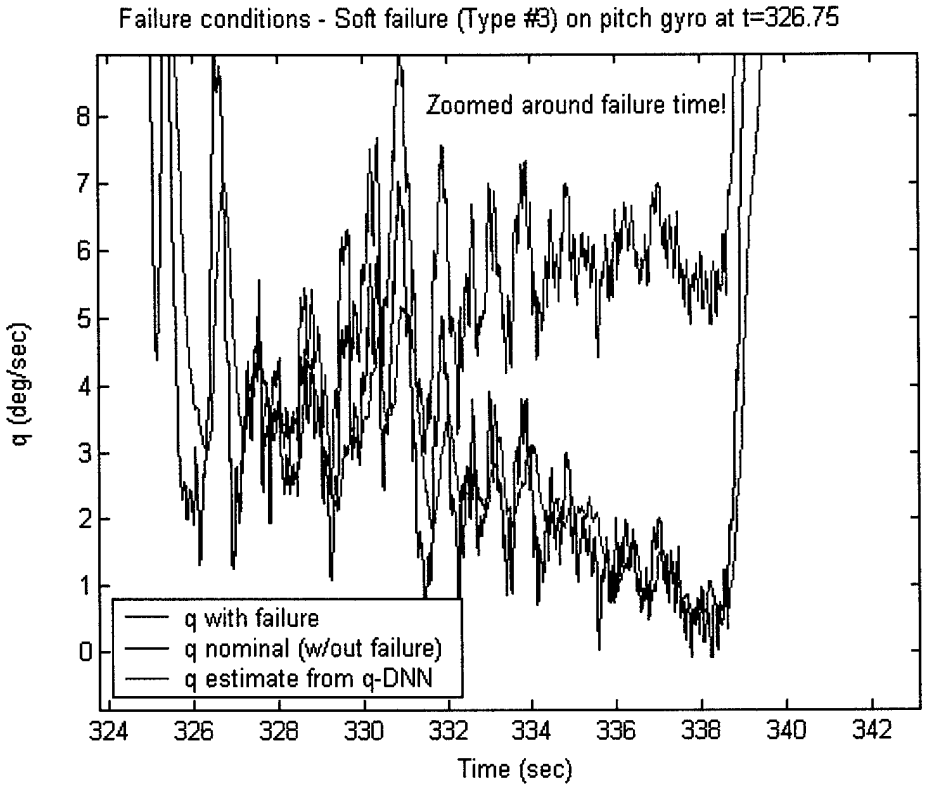


Figure 6.11 – 2nd SFDIA flight. Soft failure (Type #3) for pitch gyro (ZOOMED)

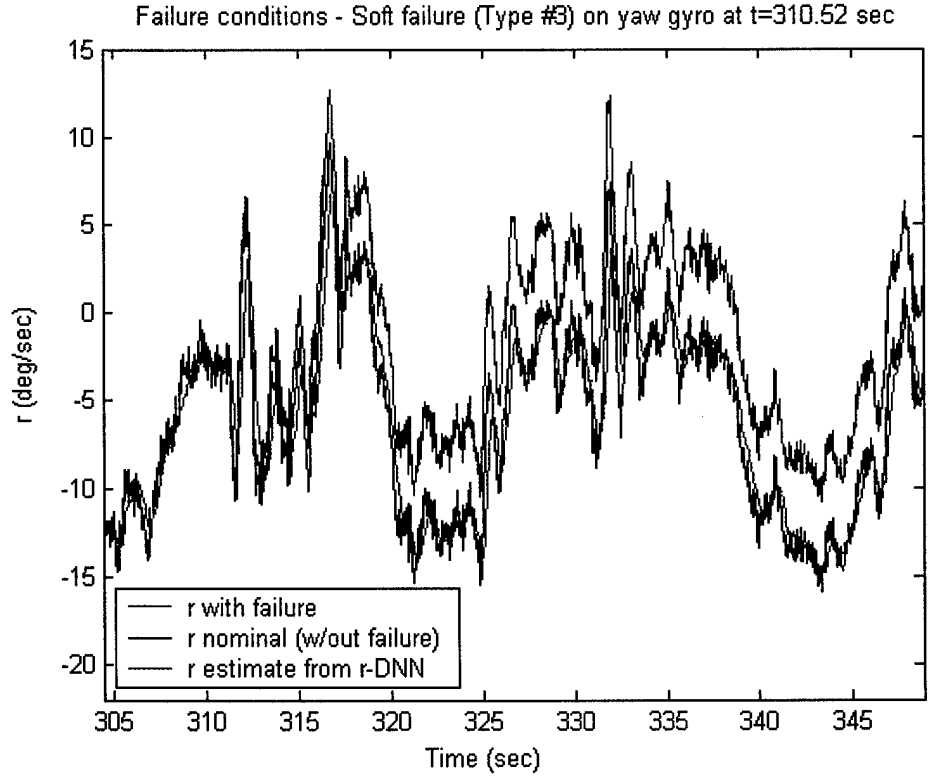


Figure 6.12 –3rd SFDIA flight. Soft failure (Type #3) for yaw gyro

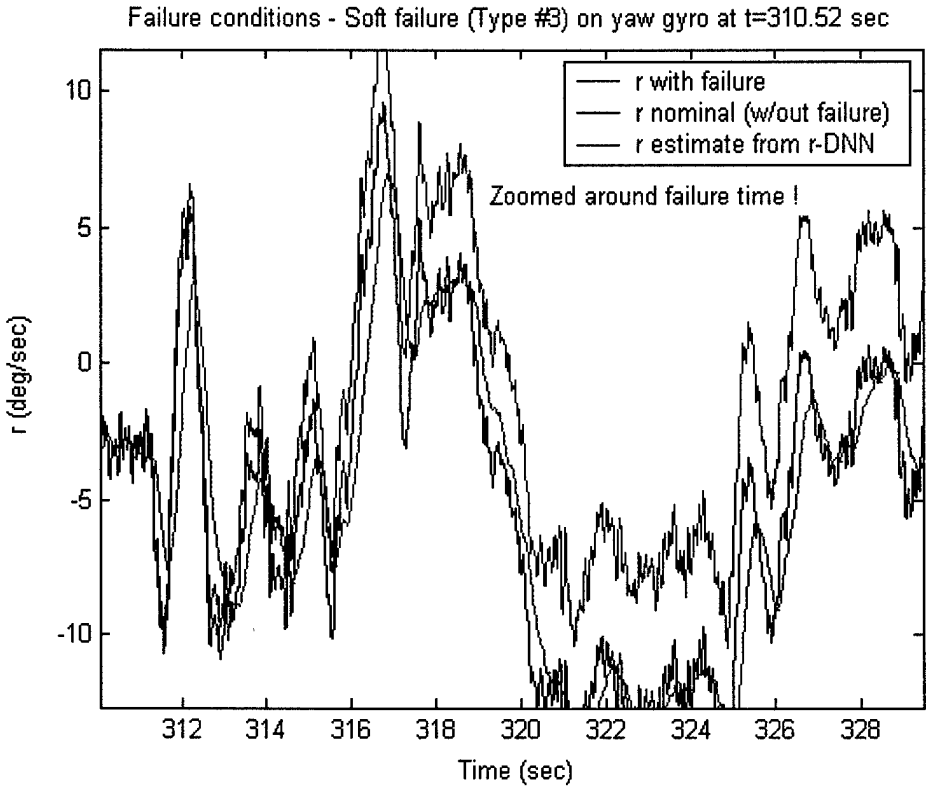


Figure 6.13 – 3rd SFDIA flight. Soft failure (Type #3) for yaw gyro (ZOOMED)

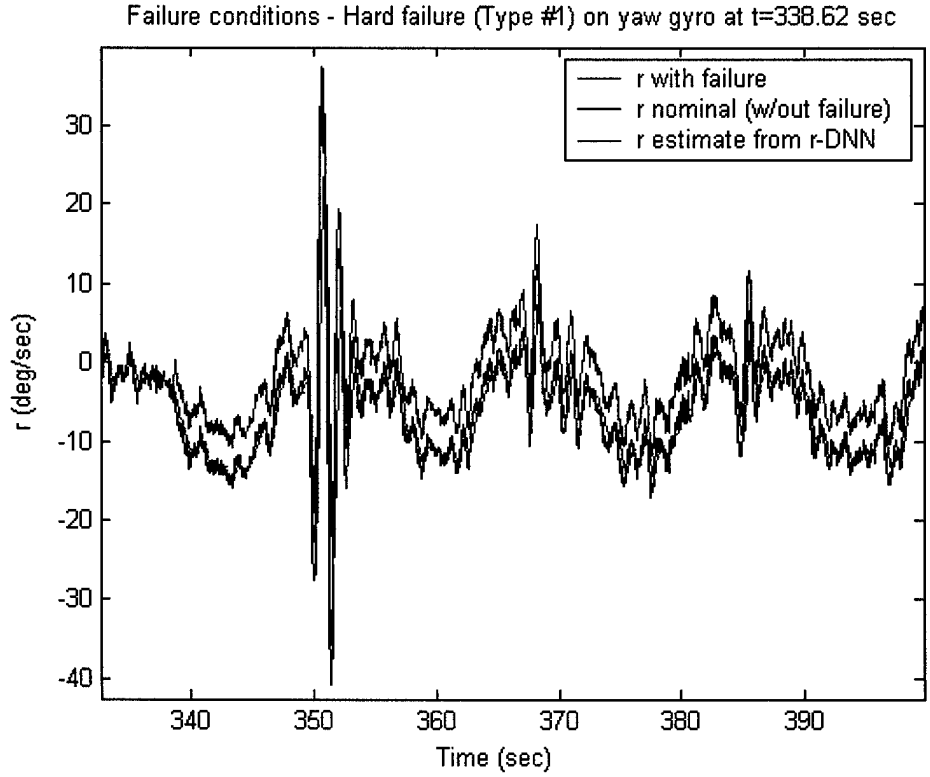


Figure 6.14 – 1st SFDIA flight. Hard failure (Type #1) for roll gyro

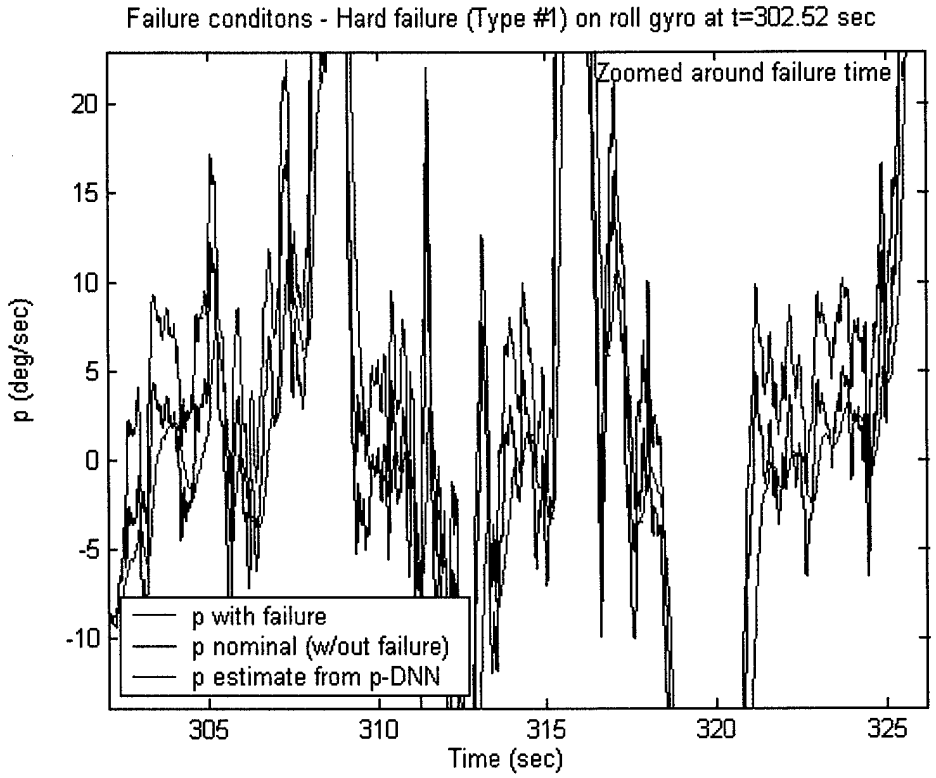


Figure 6.15 – 1st SFDIA flight. Hard failure (Type #1) for roll gyro (ZOOMED)

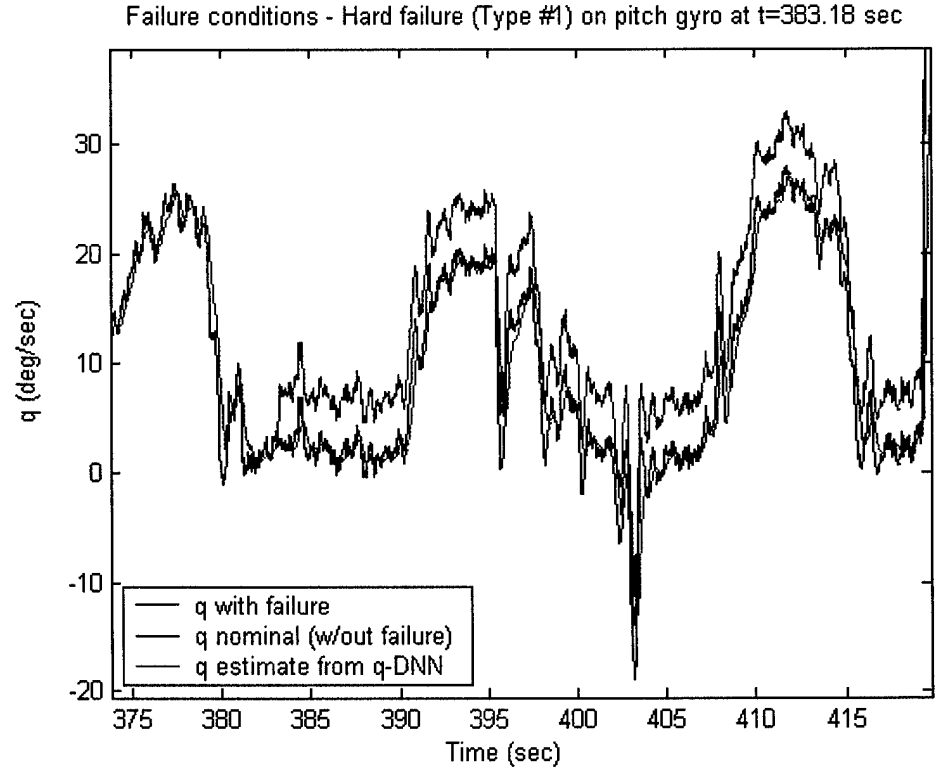


Figure 6.16 – 2nd SFDIA flight. Hard failure (Type #1) for pitch gyro

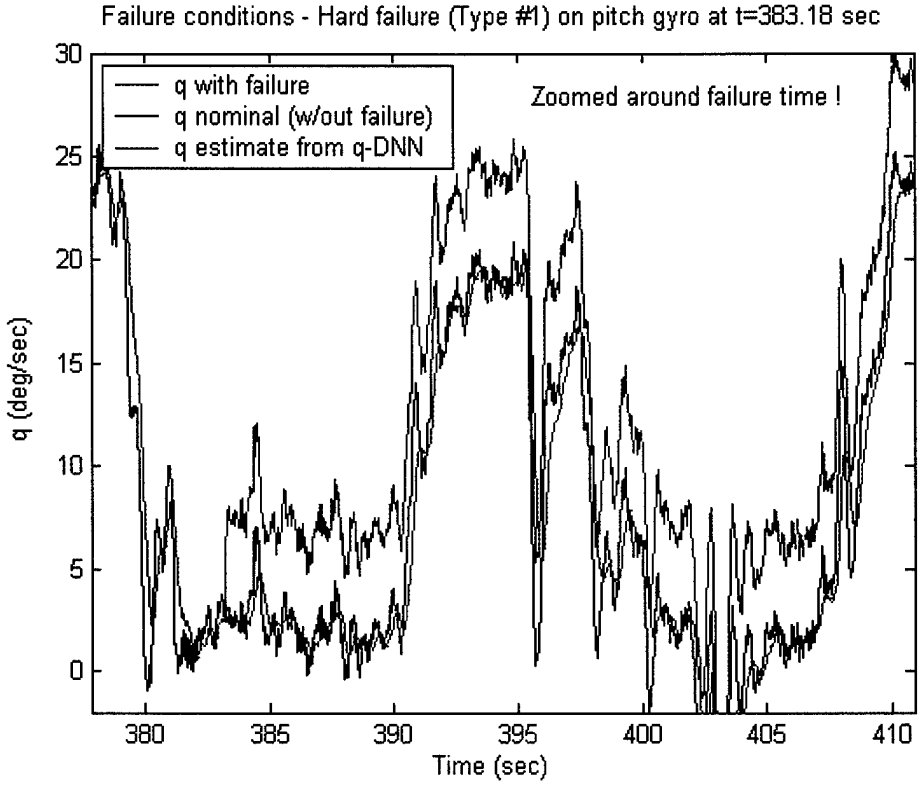


Figure 6.17 – 2nd SFDIA flight. Hard failure (Type #1) for pitch gyro (ZOOMED)

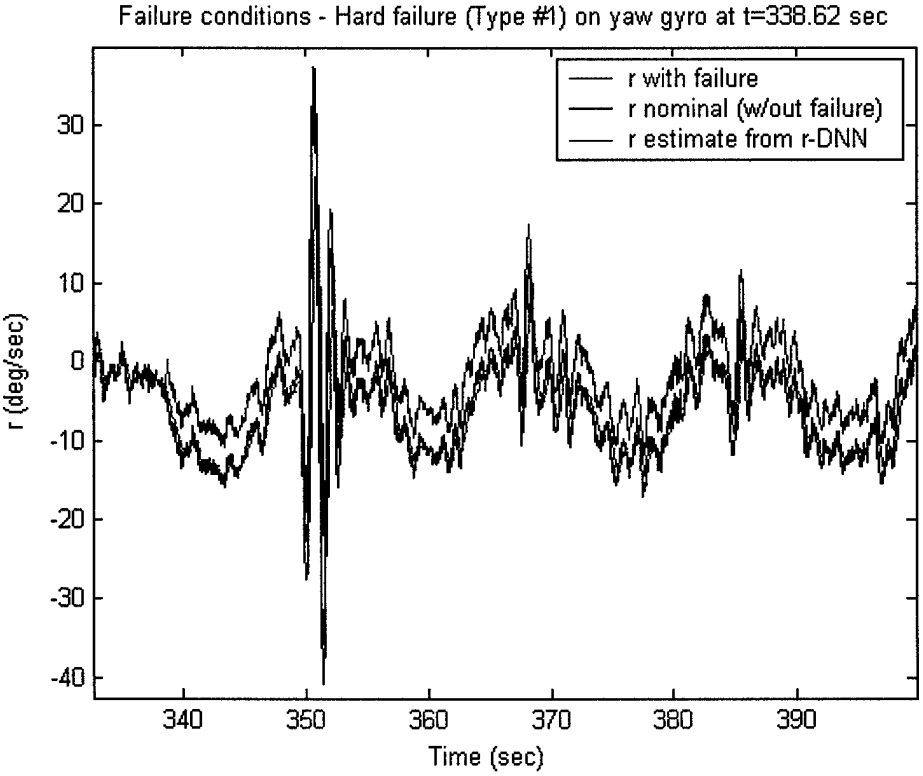


Figure 6.18 – 3rd SFDIA flight. Hard failure (Type #1) for yaw gyro

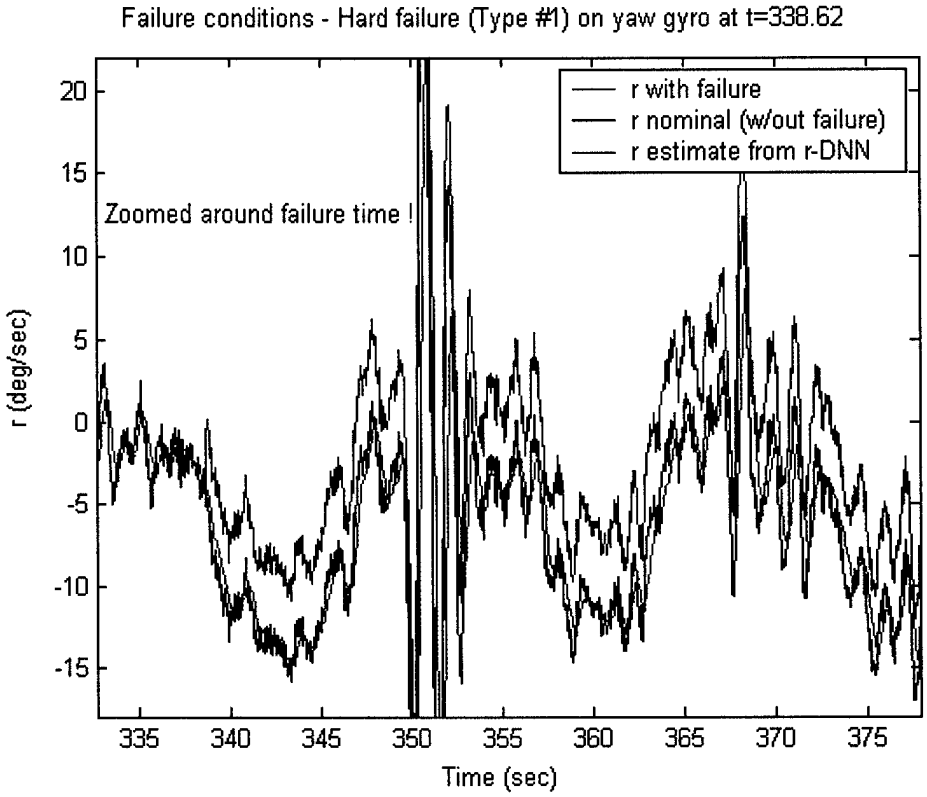


Figure 6.19 – 3rd SFDIA flight. Hard failure (Type #1) for yaw gyro (ZOOMED)

Since the emphasis was on the SFA portion of the SFDIA scheme, very low values of the SFD and SFI thresholds – described in Section #4 - were introduced. Therefore, the "false alarm" issue was not considered in this effort. The SFD and SFI thresholds are listed in Table 6.2.

Quadratic Estimation Error Parameters	Threshold values
MQEE	4 deg^2/sec^2
$DQEE_p$	2.25 deg^2/sec^2
$DQEE_q$	2.25 deg^2/sec^2
$DQEE_r$	2.25 deg^2/sec^2

Table 6.2. – Thresholds for the SFD and the SFI

The SFDIA results are summarized in Table 6.3 below.

	p-soft failure	p-hard failure	q-soft failure	q-hard failure	r-soft failure	r-hard failure
$t_{failure}$ *(sec)*	276.32	302.52	326.74	383.18	310.52	338.62
t_{LE}	281.88	304.11	336.08	383.96	316.02	341.52
t_{AE}	283.38	304.82	337.66	384.32	317.14	342.26
MEE	1.53489	1.74535	2.234225	1.543530	1.598763	1.84990
STDEE	1.94535	1.99702	2.01279	2.14389	2.07564	2.19778

Table 6.3 –Results of the SFDIA for each of the failures

The analysis of the SFDIA results confirms the capabilities of the MLP-based EBPA-trained neural approximators for providing reliable on-line estimates.

Each of the failures was detected; as expected the soft failures had longer detection delays, ranging from approx. 5 sec to 16 sec since the failures had slow transients. The detection performance were instead satisfactory for each of the hard failures with detection delays ranging from 0.5 sec to 3 sec. Again, since the selected SFI and SFD thresholds were fairly low, no false alarms were reported in the analyzed flight data. A more comprehensive flight-testing program should focus on the "failure detectability vs. false alarm" issue.

Regardless of the detection delays, the accommodation performance were from satisfactory to excellent for each of the failures due to the extensive learning accumulated by each of the DNNs from both the off-line training phase and the on-line learning.

6.2. – Flight Testing of the AFDIA Scheme

6.2.1. – Definition of the AFDIA Flight Testing

Actuator Failure Detection, Identification and Accommodation (AFDIA) scheme
Sub-scheme 1 – AFDIA following aileron failure
Sub-scheme 2 – AFDIA following elevator failure
Requiring:

- MNN, "replicating" p,q,r;
- Sum of the absolute values of 3 cross-correlation functions (R_pq, R_pr, R_qr) with N =10;
- Sum of the absolute values of the auto-correlation function (R_rr) with N=10;
- Longitudinal MLP neural controller (for pitch accommodation);
- Lateral MLP neural controller (for roll accommodation);
- Directional MLP neural controller (for yaw accommodation).

6.2.2. – Design of the AFDIA Maneuver

This is the most critical scheme since it includes the AFDI and the AFA scheme. All the different computations (for the neural networks and the statistical parameters) will be involved. The description is provided for elevator failure; however it is functionally identical to the case of aileron failure. The differences between the two failures are outlined below:

Aileron Failure
Surface locked at trim position (either $\delta_{A\text{-}Right}$ or $\delta_{A\text{-}Left} \approx 0$ deg.).
Compensating deflections :

- δ_R from DIR-NNC to offset the failure-induced yawing;
- symmetric δ_E from LONG-NNC to offset the failure-induced pitch;
- asymmetric δ_E from LAT-NNC to offset the failure-induced roll.

Elevator Failure
Surface locked at trim position (either $\delta_{E\text{-}Right}$ or $\delta_{E\text{-}Left} \approx 0$ deg.).
Compensating deflections:

- δ_R from DIR-NNC to offset the failure-induced yawing;
- remaining "healthy" δ_E (either right or left) from LONG-NNC to offset the failure-induced pitch;
- asymmetric (nominal) δ_A from LAT-NNC to offset the failure-induced roll.

The design of the AFDIA Scheme #4 (AFDIA) for elevator failure follows below. A tentative flight pattern is shown in Figure 6.20 while the entire SFDIA maneuver is summarized in Figure 6.21.

Step #0
Take off.

Step #1
Turn on the on-line learning for the MNN estimator at nominal conditions as soon as the plane reaches altitude. The MNN should "replicate" the values of p,q, and r.

Step #2
After the MNN learning has been turned on, the pilot performs mild rolling, pitching, and yawing maneuvers. This has the goal of "fine tuning" the learning of the MNN.

Step #3
Perform mild pitching maneuvers.

Step #4
At point A in the flight path, in the middle of a mild pitching, the pilot triggers an elevator failure (either on the right or the left elevator).

Step #5
Following the failure there should be a clear dynamic trend to be used by the DI (Detection and Identification) scheme. After setting thresholds (following analysis of the AFDI data) for the parameters:

$$MQEE, \sum |R_{pq}| , \sum |R_{pr}| , \sum |R_{qr}| , \sum |R_{rr}|$$

let the DI (detection and Identification) scheme on the OBC proceed to the next step (AFA) by switching the cost function to be minimized by the neural controllers.

Within Step #5 the following scenarios can actually take place:

Scenario #1 : If "D = negative" and "I = negative" following the "injection" of the failure, then "restore" nominal conditions. The AFDIA testing is then.

Scenario #2 : If "D = positive" and "I = negative" following the "injection" of the failure, then "restore" nominal conditions. The AFDIA testing is then unsuccessful.

Scenario #3 : If "D = positive" and "I = positive" following the "injection" of the failure, then the AFDIA logic in the OBC proceeds to Step #6.

Step #6
Following Scenario #3 from Step #5, at point B in the flight path switch the cost functions at the output layer of the 3 AFA MLP controllers (LAT_NNC, LONG_NNC, DIR_NNC) with the goal of restoring equilibrium conditions (as described in Section #4).

Step #7
At point C send the following commands at the same time :
- "restore" nominal (no-failure) conditions on failed elevator (left or right);
- "return" to manual control;
- "revert" the learning of the 3 NNCs to "mimic" nominal deflections.

Step #8
Turn off the learning of the DIR-NNC, LONG-NNC, and LAT-NNC at nominal conditions.

Step #9
Turn-off the learning for the MNN and stop the on-board computations of:
$\sum|R_{pq}|, \sum|R_{pr}|, \sum|R_{qr}|, \sum|R_{rr}|$

Step #10
Landing. Data downloading concludes the flight.

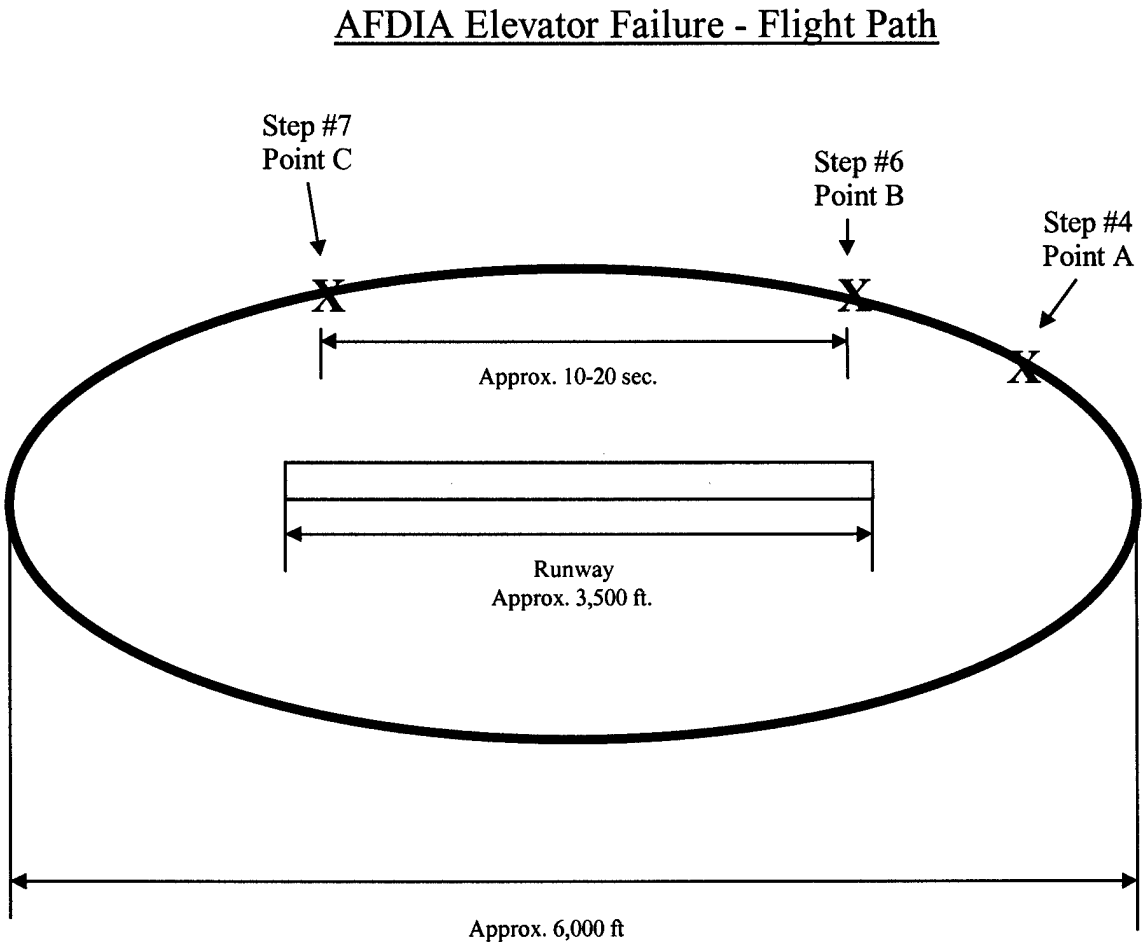
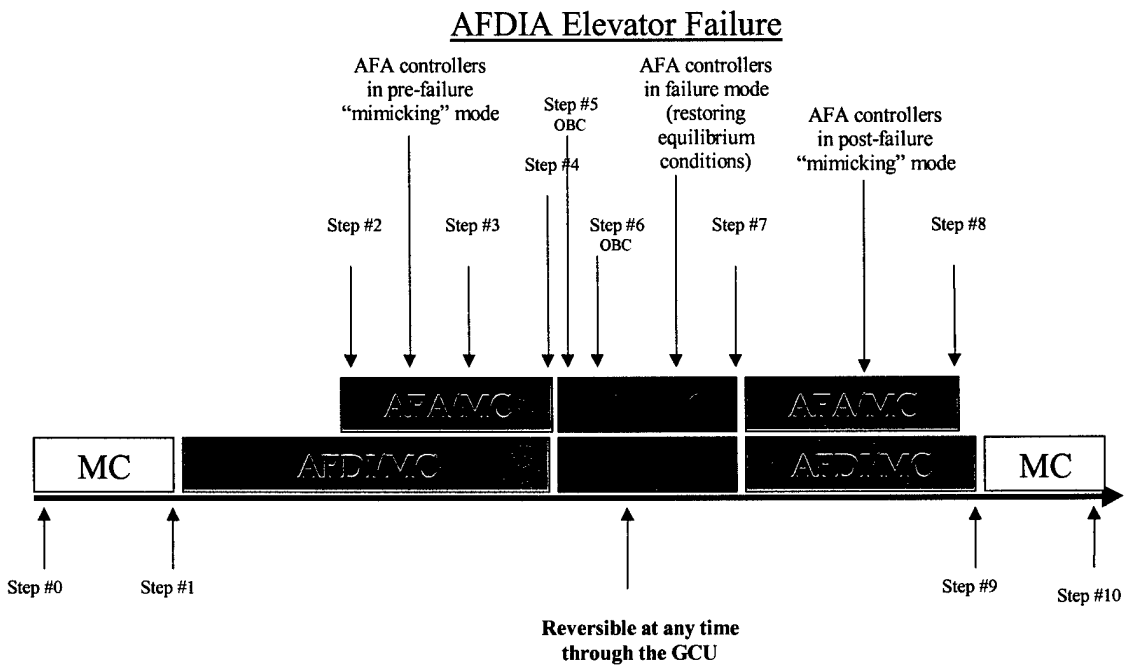


Figure 6.20 – Flight path of the AFDIA (elevator failure)



LEGEND
AC = Automatic Control
AFDIA = Actuator Failure Detection, Identification, Accommodation
MC = Manual Control

Figure 6.21 - Bar-coded task sequence for AFDIA (elevator failure)

6.2.3 – Description of the AFDIA software

The general AFDIA scheme has been described with details in Section #4. Two failure modes (aileron failure, and elevator failure) were planned. Some of the hardware components of the AFDIA scheme (control/switch board) were described in Section #2. This section provides a description of the on-board AFDIA software.

Again, 3 neural controllers (LAT_NNC, LONG_NNC, DIR_NNC) were designed for the AFDIA and AFA schemes. The on-board computer (OBC) sends out the control commands through the control/switch board. Each control surface on the aircraft can be either controlled by the pilot, OBC, or be locked at a certain angle (failure mode). The YF-22 electronic payload has the ability to be configured for different kinds of flight test with or without actuator failures, as described in Table 6.4 below.

Surface	Controlled by controller	Locked by controller (Failure mode)
Left Elevator	Yes	Yes
Right Elevator	Yes	Yes
Left Aileron	Yes	Yes
Right Aileron	Yes	Yes
Left Rudder	Yes	No
Right Rudder	Yes	No

Table 6.4 – Possible AFDIA surface configurations

During the flight, the pilot triggers the failure by flipping the control switch on the GCU. The receiver receives the control signal and sends it to the control/switch board. The control/switch board transfers the command and sends it to the OBC. The OBC selects the flight mode according to the "config" file stored in the flash disk - pre-programmed and loaded in the pre-flight software configuration prior to take-off - and sends out the controller commands – as calculated on-line by the 3 neural controllers LAT_NNC, LONG_NNC, DIR_NNC - to the control surfaces through the control board.

Each of the neural controllers (LAT_NNC, LONG_NNC, DIR_NNC) – described with details in Section #4 – is a 1-hidden layer Multi-Layer Perceptron (MLP) featuring on-line real-time adaptation of the neural architecture through the Extended Back Propagation Algorithm (EBPA). A N=10 instant window was selected for all the cross correlation functions (R_{pq}, R_{pr}, R_{qr}) and the auto correlation function (R_{rr}). The final selected input/output relationships and relative architecture of these neural controllers are outlined in Table 6.5 below.

	LONG_NNS	**LAT_NNC**	**DIR_NNC**
Input parameters	q, V, a_z, δ_E	p, r, a_y, δ_A, δ_R	p, r, a_y, δ_A, δ_R
Output	δ_E	δ_A	δ_R
Input data pattern	3	3	3
Total number of inputs	12	15	15
Number of hidden layers	1	1	1
Number of neurons in The hidden layer(s)	5	6	6
Final neural configuration	12/5/1	15/6/1	15/6/1
Final neural configuration	12/5/1	15/6/1	15/6/1
Learning rate coefficient	0.2	0.2	0.2
Momentum coefficient	0	0	0
Total number of neural parameters updated at each step	83	117	117

Table 6.5 – Selected architectures for the neural controllers

The scheme to trigger the failure mode is the single most critical link in the design of the WVU YF-22 payload. For the general safety of the operations, the pilot on the ground should be able to trigger the failure mode or regain the control of the plane at any time during the flight. According to the design, the pilot has a command switch on the GCU to set which "flight testing mode" the plane is in. When the on-board receiver receives the command signal from the GCU, it sends the PWM signal to the control box. The control box then decides the width of the signal. If it is wider than a selected threshold the control voltage is "high", otherwise it remains "low". When the OBC reads the "high" voltage, it reads the "config" file stored in the flash card to decide which surfaces to be locked/controlled. Next, the digital output on the data acquisition card sends out the selection signal to the control box. The serial port on the OBC sends out the controller command or the locking position to the command module in the control box. Next, the command module converts the command to PWM signals. According to

the control voltage and the digital selection signal from the OBC, the switchboard decides which PWM signal (from the receiver or from the command module) to use for each channel and send it to related servos. As an additional safety measure, if any of the components above is not working properly, the failure mode will not be triggered. The scheme was successfully tested in lab tests prior to be featured during the AFDIA flights in Phase #7. The information flow for the process described above is shown in Figure 6.22.

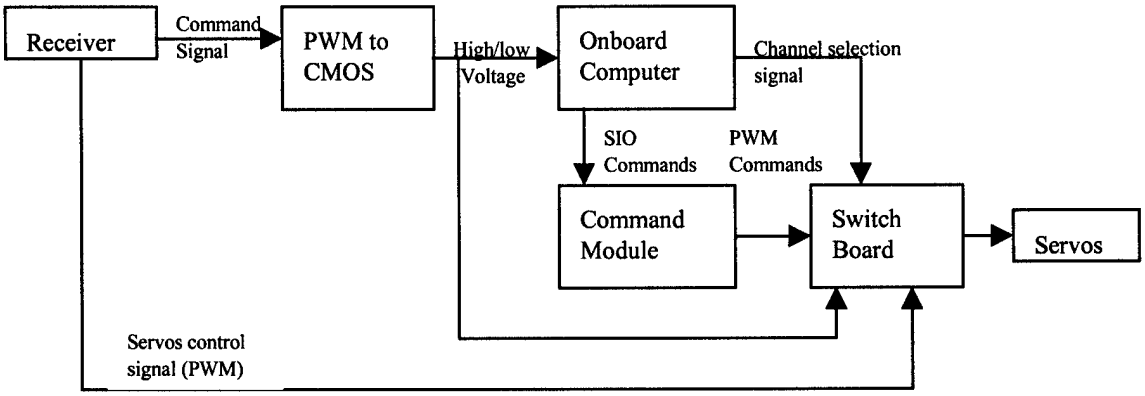


Figure 6.22 – Flow of the "failure" commands

The general block diagram of the AFDIA software is shown in Figure 6.23.

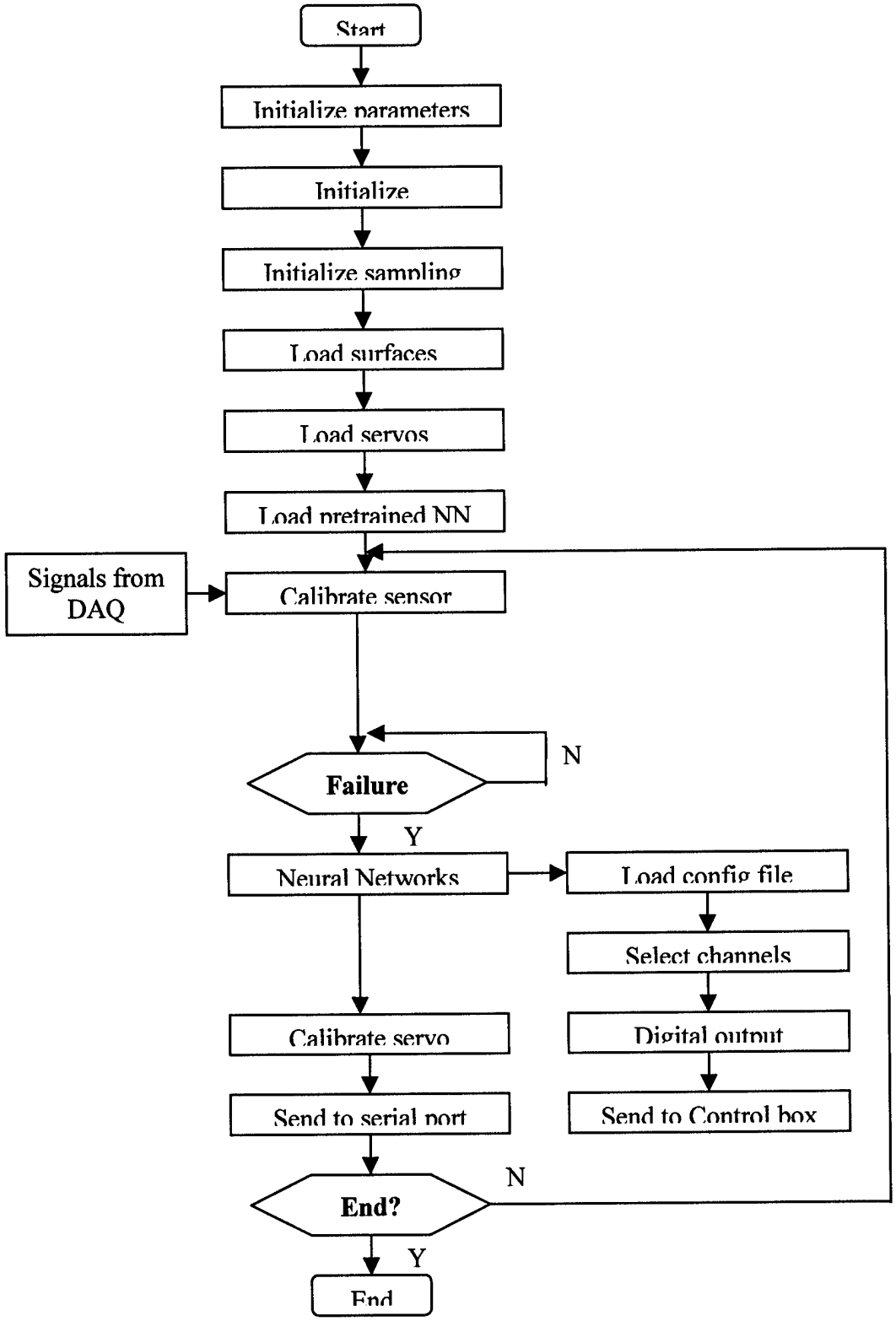


Figure 6.23 – Block diagram of the AFDIA software

6.2.4 – Results of the AFDIA Flight Testing

In addition to the pre-training of the neural estimators of the SFDIA scheme, the flight data collected for PID purposes within the flight testing activities of Phase #5 were also used for the pre-training of the neural controllers at nominal conditions (so that the neural controllers could "learn" the inverse dynamics at nominal conditions). Once again, the PID data showed to be very suitable for the pre-training of neural estimators and neural controllers. Due to time constraints, only the AFA scheme for the failure of the aileron actuator was tested in Phase #7 of the flight testing program. A total of 3 flights were performed for AFA/AFDIA purposes within Phase #7 during which 4 aileron failures were introduced. The failure consisted in locking the right aileron at trim (neutral) position during the flight. This position was then stored in the flash card of the OBC. The servo for the right aileron was also calibrated before the flight allowing the ability for the OBC to lock the right aileron at the trim position perfectly. The AFA flight-testing results relative to Failure #1 are shown in the figures below. Figure 6.24 shows the time history of the command switch voltage for the entire 2nd AFDIA flight highlighting that the aileron failure has been triggered and released twice during the flight.

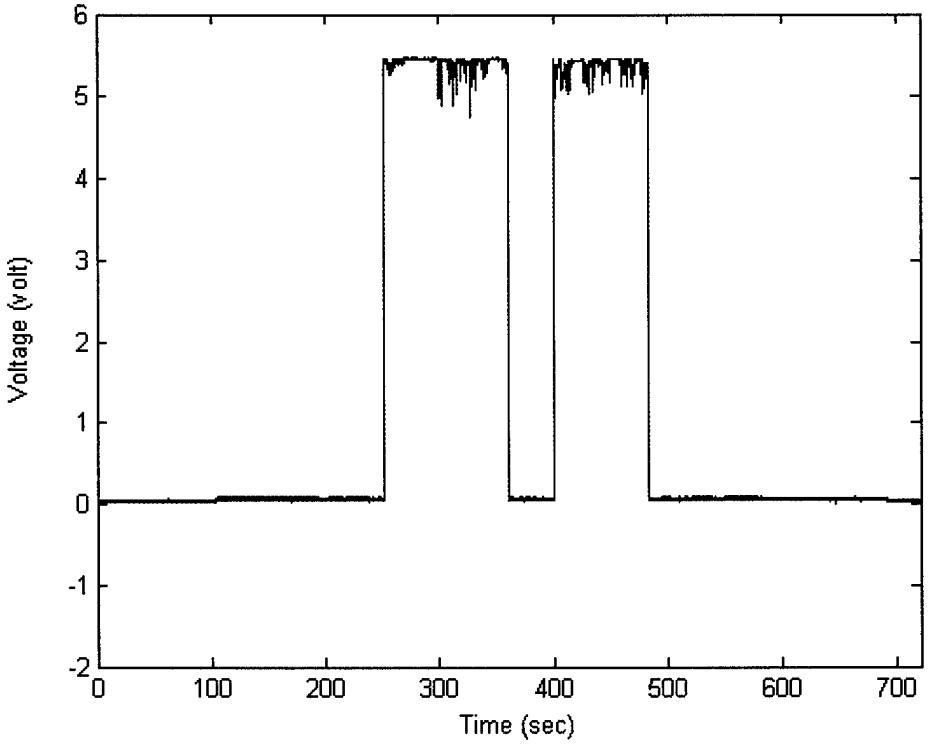


Figure 6.24 – 2nd AFDIA flight. Command switch voltage

Whenever the command signal is set at "high" voltage (> 4.5 volt), the aileron failure is triggered and released when the command signal goes back to a low voltage (< 4.5 volt). Thus, the first aileron failure was triggered at 252.04 sec (computer time) and "removed" at 360.4 sec; the second aileron failure was instead triggered at 400.04 sec and removed at 482.94 sec. Throughout the duration of these failures, the right aileron has been locked at the trim position, as shown in Figure 6.25.

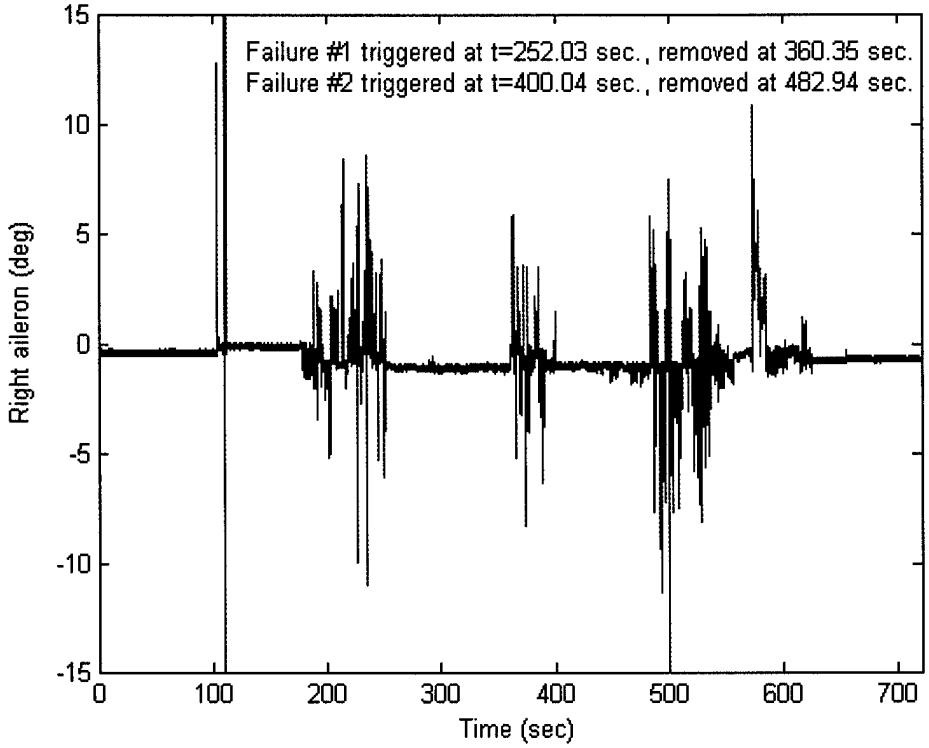


Figure 6.25 – 2nd AFDIA flight. Deflections of the right aileron

Throughout the duration of the failures on the right aileron the left aileron is commanded by both the LAT NNC (roll neural controller) and the pilot through the GCU during the turns (along the oval flight path).

The evaluation of the AFA performance of the roll neural controller can be performed through a detailed analysis of the time histories of the lateral parameters. The time history of the roll angle is shown in Figure 6.26. It should be pointed out that through the GCU the pilot keeps the YF-22 aircraft on a semi-oval path at an altitude between 1,000-1,200 ft through coordinated turns. The pilot first triggers the failure at the beginning of a straight path. Immediately after the trigger of the failure the roll NNC tends to regain leveled wings by calculating compensating deflections for the "healthy" left aileron. After a few seconds, at the end of the straight path (around t=262 sec), the pilot needs to proceed to turn. Since the pilot "knows" that the right aileron is locked, his maneuver is a bit slower and tentative. Finally, around t=276 sec, the roll NNC is turned on again and tends to re-level the wings until the next turn.

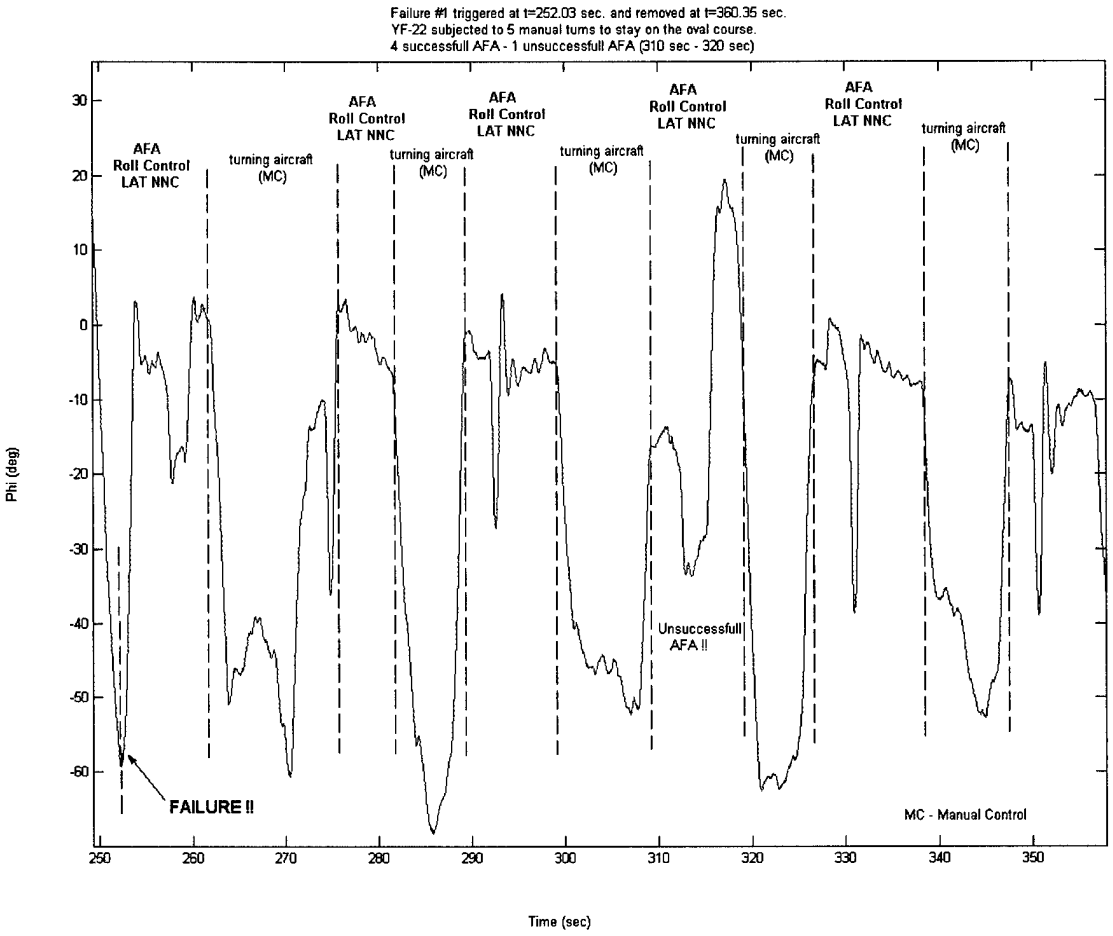


Figure 6.26 – 2nd SFDIA flight. Roll angle throughout "Right Aileron - Failure #1"

Next the pilot initiates another turn, still with the locked right aileron. This time his maneuver is sharper and somewhat shorter so that the aircraft is brought quicker at the beginning of another rectilinear path where the roll NNC is turned on again (around t=290 sec), leading, again, to leveled wings. This process continues until the failure is removed at approx. 360 sec. Throughout the duration of Failure #1 there was one instance when the roll NNC was unsuccessful (between 310 sec. and 320 sec.) and was turned off by the pilot through the GCU right before the pilot performed another turn. The outputs of the roll NNC, in terms of commanded deflections for the left aileron, and the left aileron deflections commanded manually by the pilot through the GCU are shown in Figure 6.27.

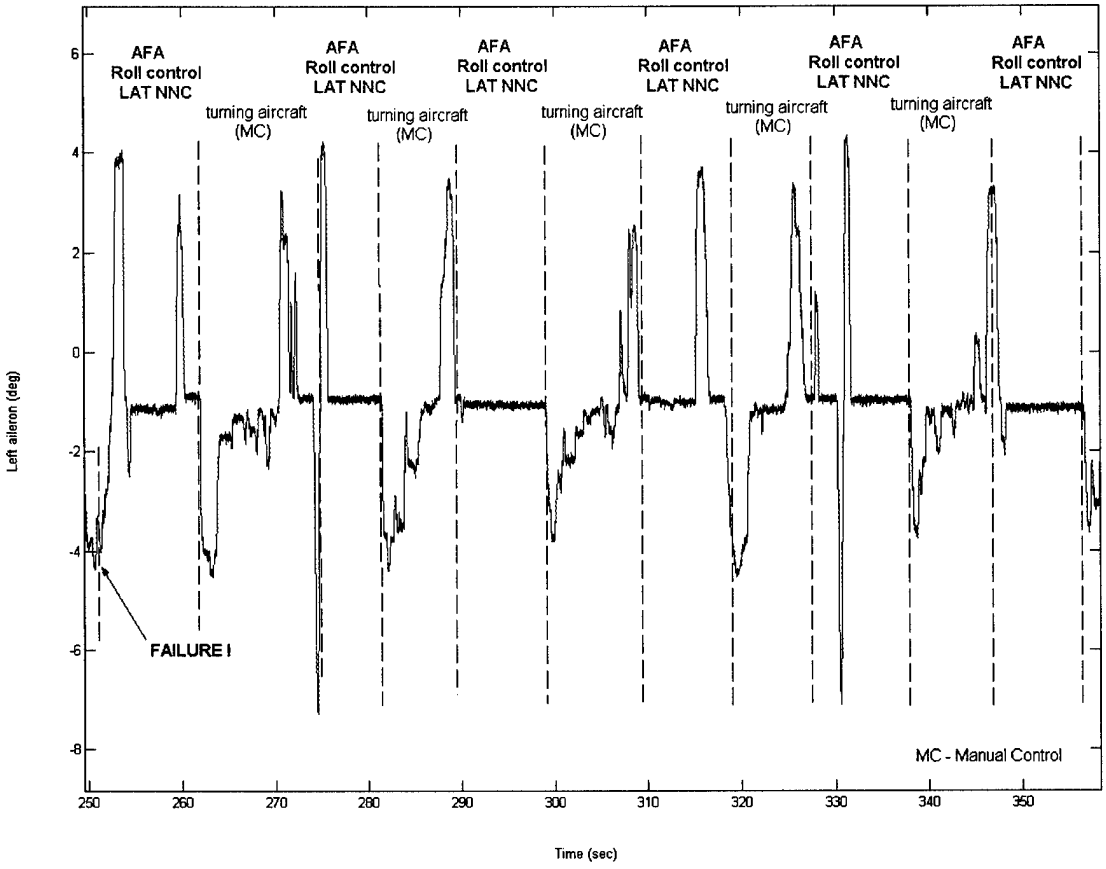


Figure 6.27 – 2nd AFDIA flight. Left aileron throughout "Right Aileron-Failure #1"

A total of 4 right aileron failures were "injected" within the 3 AFA (AFDIA) flight tests with each failure lasting an averaging of 70-80 seconds. Throughout these 4 failures the AFA LAT_NNC was turned on 19 times; for 16 instances the neural control allowed to regain the lateral control (wing level conditions), at times with some oscillations. In 3 instances the pilot felt that the LAT_NNC was not performing satisfactorily and returned to direct manual control, without disengaging the failure, prior to executing a turn. In none of these maneuvers the aircraft was flown into dangerous conditions. Due to time and resource constraints the AFDIA flight activities only involved failures on the right aileron.

Although the scope of the AFDIA investigation was limited, the experimental results confirmed the potential of neural controllers. The demonstrated capabilities of evaluating on-line a control sequence using some form of supervised teaching - through the application of EBPA-type or similar algorithms - make neural controllers a suitable alternative to 'gain scheduling-based' control laws for several flight control problems involving time varying and/or non-linearity issues.

Appendix A

Copies of publications from the project

Appendix A.1

Authors:

Fravolini, M.L., Campa, G., Napolitano, M.R.

Title:

"A Neural Network Based Tool for Aircraft SFDIA Modeling and Simulation",

Proceedings of the 2001 IASTED International Conference on Modeling and Simulation,
Pittsburgh, Pa, May 2001

A Neural Network Based Tool for Aircraft SFDIA Modeling and Simulation

Mario L. Fravolini (+), Giampiero Campa (*), Marcello Napolitano (**),
(+) Department of Electronic and Information Engineering
Perugia University, 06100 Perugia, Italy
(*, **) Department of Mechanical and Aerospace Engineering
West Virginia University, Morgantown, WV 26506/6106

ABSTRACT

This paper presents a Neural Network (NN) based tool for the modeling, simulation and analysis of aircraft Sensor Failure, Detection, Identification and Accommodation (SFDIA) problems. The SFDIA scheme exploits the analytical redundancy of the system to provide validation capability to measurement devices by employing Neural Networks as on-line non-linear approximators. The tool allows evaluating either the open loop or the closed loop performance of the SFDIA scheme. Several kinds of NN approximators and learning algorithms are employed, and a library comprising all these different adaptive neural networks is presented. In particular, Resource Allocating Networks featuring fully tuned Radial Basis Activation Functions are proposed as one of the most effective architectures. Finally, the results of a comparative study of different NN approximators applied to the SFDIA problem on a detailed nonlinear model of a De Havilland DHC-2 "Beaver" aircraft are reported.

KEY WORDS: Sensor Validation, Fault Detection, Neural Networks, Aircraft Modeling and Simulation.

1. INTRODUCTION

The traditional approach to provide fault tolerance following sensor failure is physical redundancy. Typically triple or quadruple redundancy is used in critical application, for example on military and civil aircrafts. This approach implies a penalty in term of weight, space, and, consequently, in the overall performance. An alternative approach can take advantage of the analytical redundancy [1] existing in the system. In fact, the information provided by a set of sensors along with a priori knowledge of the system allows detecting and identifying the faulty sensor, while estimating the related variable as a function of other measured variables.

Research on fault tolerance based on analytical redundancy has produced a quite mature framework especially for linear systems [2-3]; but unfortunately, the assumption of linearity is not often valid throughout the whole flight envelope of an aircraft, thus the performance of a fault tolerant scheme based on such assumption can become inadequate, for example providing a high false alarm rate in a wide portion of the flight envelope. Therefore, the control and the SFDIA of an aircraft should be addressed with non-linear estimation approaches. In this context, a very promising approach is to employ Neural Networks as the main nonlinear approximators of an SFDIA scheme. In recent years many NN-based SFDIA schemes have been proposed and developed [4-5]. Although the benefits of employing NNs for fault tolerance purposes within a non-linear flight control system are clear, the critical dependence of the performance on the choice of the NN architecture and learning algorithm it is also well known. Concerning the NN structure the most used function approximators are polynomials, rational and Spline functions, Wavelets, Multi Layer Perceptrons, as well as Radial Basis Functions (RBF NNs) [9]. Among the issues concerning the on-line training algorithm there is the time required to achieve an acceptable learning level, the difference between local and global learning, and the level of complexity of the NN architecture. Other important design issues concern the design of systematic procedures for the creation of non-linear estimators and the stability of the learning algorithm. Furthermore, the greatest part of the SFDIA scheme is often applied only at supervisory level because of the difficulty of clearly defining the interaction between the closed loop control law and the SFDIA scheme. The full comprehension of these issues requires dedicated simulation software in which it is possible to simulate the interaction between the closed loop dynamics of the Aircraft and the dynamics of the SDFIA system. In this work, the above-mentioned problem has been specifically addressed and a general SFDIA software has been designed in the Simulink environment. The tool allows evaluating either the open loop or the closed loop performance of the SFDIA scheme that employs different kinds of NN approximators and learning algorithms. A library comprising several different adaptive (i.e. online learning) NNs is presented.

(+) Assistant Professor.
(*) Research Assistant Professor.
(**) Associate Professor.

Finally, the results of a comparative study of different NN approximators applied to the SFDIA problem on a detailed nonlinear model of a De Havilland DHC-2 "Beaver" aircraft are reported.

2. NEURAL NETWORK-BASED SFDIA

In this section the SFDIA problem it is briefly reviewed. Reference is made to the general scheme previously described in ref. [5].
The scheme is summarized in figure 1.

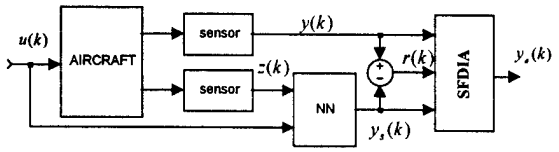


Fig. 1 – General SFDIA scheme

The rationale behind this scheme is based on the observability property of the system, which allows the use of analytical redundancy. Analytical redundancy implies that some system variables are functionally related [1]; namely, a variable $y(k)$ can be expressed as function of a suitable set of other variables $z(k)$ and inputs commands $u(k)$. On the basis of this assumption, the outputs of sensors, measuring different but functionally related variables, can be processed in order to detect a fault, identify the faulty component, and finally provide accommodation. The common way to perform SFDI is to online monitor a (filtered) residual signal $r(k)$, which is the difference between the sensor output $y(k)$ and its estimation $y_s(k)$ provided by a proper estimator (in this work the estimator is a NN). In other words, a supervisory block, which evaluates a quadratic function of the filtered residual, monitors the status of the sensor. When this quadratic function exceeds a predefined threshold, the state of the corresponding sensor is declared suspect and a suitable procedure is called to decide on the health status of this sensor (at the same time the learning of the NN is stopped in order to avoid the wrong measure from being learnt). If the state of the sensor is then declared faulty, a fault procedure is enabled, and an accommodated variable $y_a(k)$ is provided as output. In case of a multiple physical redundancy the accommodation usually consists on the simple replacement of the faulty sensor by a healthy one. On the other hand, if no physical redundancy is available, the accommodation procedure substitutes the faulty measure with the estimation given by the NN ($y_a(k)=y_s(k)$). Several options can be added to this basic scheme to increase robustness in presence of noisy measurements and/or intermittent sensor failures [5]. As for any other SFDIA approach, the following distinct capabilities are critical:

1) Failure *detectability* and false alarm rate (the sooner the fault is detected and the least the number of false alarm it is, the better is the SFDI system).
2) *Estimation error* (The least is the estimation error, the better is the quality of the accommodation).

3. THE SIMULATION ENVIRONMENT

The Neural Network based SFDIA modeling and simulation toolbox was built under the Simulink® environment (by The Mathworks Inc). In particular the freely available Flight Dynamics and Control (FDC) toolbox for Matlab [6] provides powerful tools for flight simulation, flight dynamic analysis, and flight control system design.

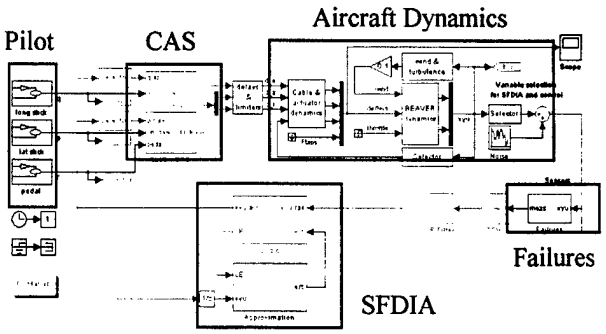


Fig. 2 - Modeling and simulation environment

In figure 2 the graphical interface of the proposed aircraft SFDIA tool is shown. The main blocks of the scheme are the following:

- **Aircraft Dynamics**: The tool has been built around a generic non-linear aircraft model, which has a modular design that provides maximum flexibility to the user. The aircraft model has been implemented as a Simulink block-diagram. The model is configured for the DeHavilland DHC-2 Beaver aircraft, but can be adapted for many different kinds of airplanes if required. The model provides a detailed characterization of aerodynamic forces, wind gusts, air turbulence, engine model, actuators (comprehensive of delays and limiters), and sensors (comprehensive of delays and measurement noise).
- **Control Augmentation System** (CAS): the block is implemented with a Stochastic Optimal Feedforward and Feedback controller as reported in [7].
- **Pilot commands**: the block allows the generation of arbitrary commands. It emulates typical pilot stick deflection commands.
- **Hard and soft sensor failures**: the block injects (adds) arbitrary soft and hard sensor failures to the desired measured signals
- **SFDIA Group**: It is the core of the tool and performs the main SFDIA procedures. It is constituted by two main sub-blocks:
  - **Approximators**: The block contains the Neural Network based function estimators. In figure 3 the scheme for the estimation of the three gyros rates $p(k),q(k),r(k)$ is shown. Different kinds of NN estimator blocks can be selected (the

available NNs and learning algorithms will be described in the next section).

- **SFDIA LOGIC**: The block performs the main threshold based sensor failure detection identification and accommodation operations (figure 4 and 5). Two filtered residuals (derived by filtering the absolute approximation error with both a "fast" low pass filters and a "slow" low pass filter) are contemporary evaluated for each sensor. When the fast filter output is bigger than a threshold, the corresponding NN learning is preventively stopped (LE=0), in order to prevent the possibly wrong signal from being learnt. When the slow filter output is bigger than a threshold, the corresponding sensor is declared failed (AE=0), so the accommodation logic is enabled, and the estimated signal is fed back through the controller instead of the faulty one. For more details about the SFDIA scheme see [8].

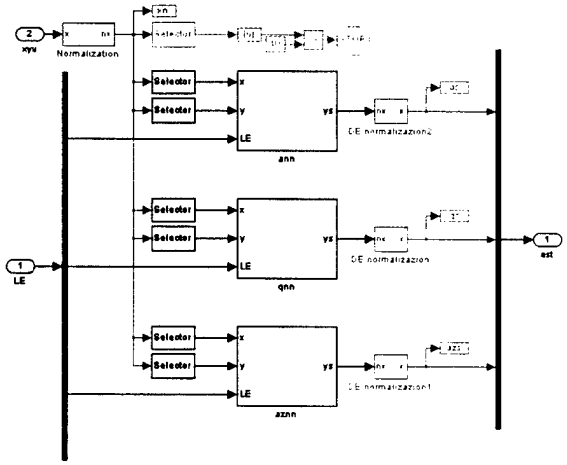


Fig. 3 - Estimators for *p(k) q(k) r(k)* signals

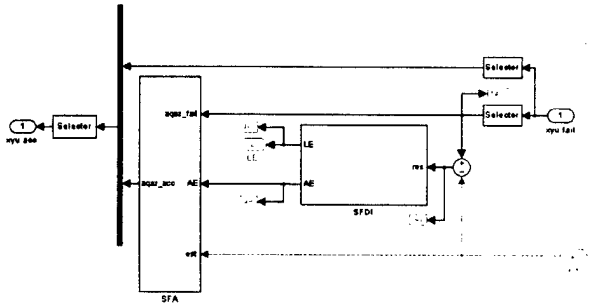


Fig. 4 – SFDIA Logic

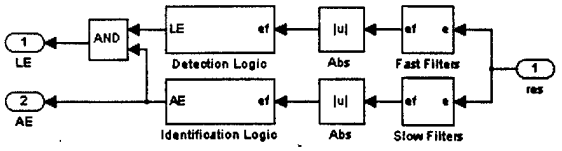


Fig. 5 – SFDI Logic

4. AN ADAPTIVE NEURAL NETWORKS LIBRARY FOR ON-LINE LEARNING.

In order to select a suitable optimal SFDIA scheme it is necessary to test and to compare the performance achieved by different kinds of online Neural Network estimators, to this purpose, a Simulink Library containing several NN architectures has been built (Fig. 6). The NNs in the library exploit linear, sigmoidal and radial basis as activation functions and employ different online learning algorithms:

ADaptive LInear Network: (ADALINE): It can be used in presence of almost linear aircraft dynamics. The ADLINE network approximates a vector signal $y \in \Re^n$ as a linear combination of M inputs signals $x \in \Re^m$, namely: $Y=Wx$ where $W \in R^{m \times n}$ is a matrix of real numbers that are updated online (states). The matrix W is updated by using a modified steepest descendent gradient in order to minimize the modified square error between measurements and estimates the resulting updating law is:

$$W(k+1) = W(k) - \gamma\, e(k)\, x(k) - \mathrm{mod}(e, W)$$

where γ is the learning rate, $e(k)=y(k)-y_s(k)$ is the current estimation error and $mod(e,W)$ is the "modification" that stabilizes the rule. Different kinds of such modifications to the basic steepest descent rule are available, e.g. sigma modification, e-modification, dead zone and projection [9-10]. By employing one of the above-mentioned modifications, it is possible to guarantee the ultimate boundedness of both estimation error and network weights, in presence of noise and uncertainty.

Multi-Layer Perceptron with Extended Back-Propagation. (MLP-EBP): It is a three layer NN, based on sigmoidal neurons with activation function:

$$f(net, U, L, T) = \frac{U - L}{1 + e^{-net/T}} + L$$

In the MLP-EBP the back-propagation algorithm it used not only to update the weights of the input and output matrices ($W(k)$, $V(k)$), but also to update the parameters U,L,T, that define the shape of each neuron. The application of MLP-EBP to SFDIA has been extensively treated in [5]. The study showed that EBP outperforms the standard Back-Propagation in terms of learning speed and approximation accuracy; these characteristics are particularly suitable for online learning problems.

Radial Basis Function (RBF) (Standard): In the standard RBF Network the estimations $y_s \in \Re^m$ are expressed as a linear combination of M Gaussian Basis functions $\phi_j(\bullet)$:

$$\hat{y}_i(x) = w_{i0} + \sum_{j=1}^{M} w_{ij}\phi_j\left(\left\|x-\mu_j\right\|\right) \quad (7)$$

$$\phi\left(\left\|x-\mu_j\right\|\right) = \exp\left(-\frac{1}{2\sigma_j^2}\left\|x-\mu_j\right\|^2\right) \quad (8)$$

where $x \in \Re^n$ is the input vector, the parameters μ_j and σ_j are the basis center and width respectively. In the basic implementation the hidden layer neurons are *a priori* statically allocated on a uniform grid that covers the whole input space and only the weight w_{ij} are updated. Although easy to implement, this approach requires an *exponentially increasing* number of basis functions versus the dimension of the input space. For this reason standard RBF can be applied only for *small dimensionality* problems. An inaccurate choice of centers and widths usually results in a poor performance especially in highly nonlinear systems. A way to partially reduce the online computational burden is to update just the weights in the neighborhood of the most activated gaussian. This approach has been called Extended-RBF.

Extended RBF (ERBF): A simple criterion to implement ERBF is to define an activation radius d^{max} and to update the weights of the gaussians that satisfy the relation:

$$\left\|\mathbf{x}-\mu_j\right\|^2 \le d^{\max}$$

Fully Tuned Minimal Resource Allocation Network RBF (MRAN-RBF): In order to avoid the dimensionality problems generated by standard RBF, Platt [11] proposed a sequential learning technique for RBFNs, where the emphasis was to learn quickly, generalize well and have a compact representation. The resulting architecture was called the Resource Allocating Network (RAN) and has proven to be suitable for online modeling of non-stationary processes. The RAN provides powerful method for on-line modeling with only an incremental growth in model complexity, which renders the algorithm especially suitable for on-line real-time applications. The RAN learning algorithm proceeds as follows: At each sampling instant based on three criteria, the network is either grown (i.e. some units are added) or the existing network parameters (the centers μ_j, the weights w_{ij} and the variances σ_j) are adjusted using a suitable online learning algorithm, which can be are formalized as follows:

- *Current estimation error criteria*, error must be bigger than a threshold:
$$e(k) = y(k) - \hat{y}(k) \ge E_1$$
- *Novelty criteria*, the nearest center distance must be bigger than a threshold:
$$\inf_{j=1}^{M}\left\|x(k)-\mu_j(k)\right\| \ge E_2$$
- *Windowed mean error criteria*, windowed mead error must be bigger than a threshold:
$$1/T\sum_{i=0}^{T}[y(k-T+i)-\hat{y}(k-T+i)] \ge E_3$$

To avoid an excessive increase of the Network size a *pruning strategy* can also be applied. When this happens the network is called Minimal RAN (MRAN) [12]. In all these RBF based networks, the learning algorithm is a modified steepest descent rule (as in the ADALINE). In [13] it has been shown that the employment of this algorithm guarantees the ultimate boundedness of both the estimation error and weights for a fully tuned MRAN, earlier results are available for generic RBF networks.

Fully Tuned Extended Minimal Resource Allocation Network RBF (EMRAN-RBF): This Neural Network is a powerful variation of the standard MRAN [12]. The growing and pruning mechanisms remains unchanged, while the parameters are updated following a "winner takes it all" strategy. In practice only the parameters of the most activated neuron are updated, while all the other are unchanged. This strategy implies a significant reduction of the number of parameters to be updated online, and for this reason it is particularly suitable for online applications. In [13] it was shown that the EMRAN algorithm implies just small performance degradation with respect to the MRAN.

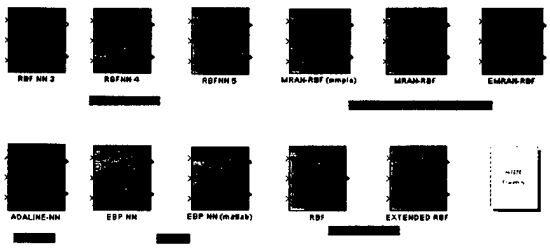


Fig. 6 – Adaptive Neural Networks Library

5. SIMULATION EXAMPLE

In this paragraph the comparative results of SFDIA capabilities provided by different NN estimators are shown. The system is a detailed 6DOF non-linear model of the De Havilland DHC-2 "Beaver" Aircraft. The SFDIA procedure is applied to the estimation of the three gyro rates *p(k),q(k),r(k)*. Due to space constraint issues, only the results for a failure of the pitch rate sensor *q(k)* are reported; similar results were also achieved for the roll and yaw rate sensors. The numerical simulation starts at nominal cruise conditions and an on-line learning process of 6000 sec of flight is run, allowing the NN to achieve a substantial amount of learning. A new set of maneuvers lasting 2000 sec is then flown with the failure of one the sensor occurring at exactly T_f=1098 s. Different soft and hard failure were injected in the system during the

simulation. Since we assumed no physical redundancy of the sensor, as soon as a fault has been detected, the faulty measure $q_f(k)$ is replaced by the estimation $q_s(k)$ in the feedback control loop. Three different NN architectures were tested. In particular we compared the performance achieved by ADALINE+MRAN, EMRAN and MLP-EBP (of comparable complexity) in presence of biasing signals of different amplitude and rise time added to the nominal values. The comparison of the performance provided by the three closed loop SFDIA schemes is quantified by evaluating classical SFDIA indicators.

Specifically, to evaluate the goodness of the **accommodation system**, the following parameters are considered:

$$MAEE(k) = \frac{1}{N}\sum_{k+1}^{N}\left|y(k) - \hat{y}(k)\right|$$

$$VAEE(k) = \frac{1}{N}\sum_{k+1}^{N}\left(\left|y(k) - \hat{y}(k)\right| - MAEE(k)\right)^2$$

where $MAEE(k)$ and $VAEE(k)$ represent respectively the mean and variance of the absolute estimation error. The integer 'N' represents the number of time steps from the instant when the failure of sensor is declared to the end of the flight (simulation).

To evaluate the goodness of the **detection/identification system**, the detection ratio S2/S1 between the main peak of the filtered residual signal during the failure transient ($1096 < t < 1120$) and the peek of the filtered residual before failure ($0 < t < 1090$) is considered. This ratio quantifies the detectability provided by the scheme. Furthermore, the time percentage in which LE=0 before the true-fault detection, is reported to quantify the false detection (false alarm) rate, and the time percentage in which AE=0 before the true-fault declaration, is reported to quantify the false accommodation rate.

All these results are reported in table I, where for each type of failure the following parameters are shown:

- The amplitude and the rise time of the biasing signal, the instant in which the fault is detected (Tdetect)
- The indexes *MAEE(1), VAEE(1), MAEE(2), VAEE(2)* evaluated prior (1) and after (2) the fault is detected.
- The Ratio S2/S1.
- The false detection rate and the false accommodation rates.
- The total simulation time. This is a parameter to evaluate the relative computational burden of the different NN architectures and implementations.

Subscript "1": before failure; subscript "2": after failure.

Amplitude	Rise Time	Tdetect	MAEE 1	VAEE 1	MAEE 2	VAEE 2	S1/S2	False Det	False Acc	Simtime
ADALINE+MRAN										
2.5	0.05	996.75	0.311711	0.083158	0.585721	0.11867	3.517762	1.086195	0	412.49
1	0.05	999.9	0.312441	0.083086	0.580709	0.117909	1.799109	1.165878	0	404.91
2.5	0.3	996.9	0.311711	0.083158	0.58541	0.118705	3.51502	1.086195	0	407.82
1	0.3	1000	0.312441	0.083086	0.580586	0.117818	1.801447	1.165878	0	407.38
2.5	4	996.8	0.312387	0.083078	0.582668	0.117952	3.525236	1.084072	0	406.12
1	4	1001.55	0.3122	0.082985	0.578411	0.116782	1.787364	1.1403	0	405.73
EMRAN										
2.5	0.05	998	0.363857	0.157222	0.362104	0.175498	6.441479	0.275745	0	405.41
1	0.05	1002.35	0.364221	0.15791	0.359562	0.173116	3.284032	0.391507	0	404.63
2.5	0.3	998.1	0.363857	0.157222	0.362086	0.175737	6.425145	0.273231	0	407.82
1	0.3	1002.45	0.364221	0.15791	0.359376	0.173011	3.270282	0.391507	0	405.99
2.5	4	1000.1	0.364804	0.158085	0.360973	0.175841	6.462528	0.280286	0	407.89
1	4	1003.7	0.362527	0.157198	0.357567	0.172815	2.99019	0.323527	0	408.65
MLP-EBPA										
2.5	0.05	996.25	0.397254	0.123787	0.420127	0.192858	3.286115	7.031628	1.571744	452.53
1	0.05	997.85	0.398244	0.123971	0.422461	0.191947	1.784715	7.146301	1.613026	450.77
2.5	0.3	996.4	0.397254	0.123787	0.420459	0.192936	3.284431	7.03449	1.571744	450.5
1	0.3	997.95	0.398244	0.123971	0.42241	0.192146	1.778442	7.146301	1.613026	450.99
2.5	4	998.05	0.398244	0.123971	0.420935	0.192985	3.344961	7.086003	1.571744	453.91
1	4	999.65	0.397947	0.124038	0.421829	0.192255	1.81221	7.146301	1.613026	452.85

Table 1: SFDIA Performance Parameters

As expected the sixth failure is the most difficult do detect by any NNs. The best detectability performance, despite the simplicity of the learning algorithm, was given by the EMRAN, which performs slightly better of the ADALIN+MRAN; on the contrary, the MLP-EBP after the accommodation causes the instability of the closed loop system, testifying, in this case, the non accurate online mapping achieved by this architecture. Moreover, as a general trend, it is clear that the quality of the approximation is crucial in allowing a good performance in term of fault detectability, low false detection and accommodation rates, and mostly important, stability (and performance) of the accommodated closed loop.

6. CONCLUSIONS

In this paper an aircraft SFDIA analysis tool was discussed. The tool allows the investigation of the interactions between the closed loop aircraft dynamics and the dynamics of the SFDIA system. A clear understanding of this interaction is of great importance since an incorrect choice of the internal SFDIA approximation architecture could result in the instability of the feedback control system. This aspect has been clearly pointed out by means of a simulation example in which instability occurs in the accommodation phase even if the sensor failure is correctly and promptly detected and isolated. In this respect, the main feature of the tool is the possibility of testing and comparing in closed loop the capabilities of SFDIA schemes exploiting different kinds of Neural Networks as nonlinear approximators. This capability is a consequence of the extensive modularity of the whole simulation tool, and allows an easy change of aircraft dynamics and feedback control law as well as NN estimators and SFDIA scheme.

ACKNOWLEDGEMENT

Partial support for the 1st author has been provided by grants from the Italian National Research Council (CNR). Partial support for the 2nd and 3rd author has been provided through the NASA Ames No. NAG 2-1158 and the DoD/EPSCoR Air Force grant No. F49620-98-1-0136 respectively.

REFERENCES

[1] R. J. Patton, P. M. Frank, R. N. Clark, *Fault diagnosis in dynamic systems, theory and applications* (Englewood Cliff, NJ, Prentice-Hall, 1989).

[2] H. Baruh, K. Choe, Sensor-Failure Detection Method for Flexible Structures, *AIAA Journal of Guidance, Control, and Dynamics, 1987, Vol. 10,* no 5, 474-482.

[3] Kerr, T.H., False Alarm and Correct Detection Probabilities over a Time Interval for Restricted Classes of Failure Detection Algorithms, *IEEE Transactions of Information Theory, 1982, IT-28,* No. 4, pp. 619-631.

[4] Ha, C.M., Wei, Y.P., Bessolo, J.A., Reconfigurable Aircraft Flight Control System Via Neural Networks, *Proceedings of the 1992 Aerospace Design Conference, AIAA Paper 92-1075,* Irvine, Ca, Feb. 1992.

[5] M. R. Napolitano, Y. An, B. Seanor, A fault tolerant flight control system for sensor and actuator failure using neural networks, *Aircraft Design, 2000, vol. 3,* pp. 103-128.

[6] Rauw, M.O.: *A Simulink Environment for Flight Dynamics and Control analysis - Application to the DHC-2 "Beaver"* (MSc-thesis, Delft University of Technology, Faculty of Aerospace Engineering, Delft, The Netherlands, 1993).

[7] N. Halyo, H. Direskeneli, B. Taylor, *A Stochastic Optimal Feedforward & Feedback Control Methodology for Superagility* (NASA CR 4471, November 1992).

[8] M. L. Fravolini, G. Campa, M. R. Napolitano, Minimal Resource Allocating Networks for Aircraft SFDIA, *IEEE Int. Conference on Advanced Intelligent Mechatronics 2001,* Como, Italy, July 2001.

[9] M. Polycarpou, On-Line Approximators for Nonlinear System Identification: A Unified Approach, *Control and Dynamic Systems Series, Volume 7, Neural Network Systems Techniques and Applications* (Academic Press, January 1998).

[10] D. S. Broomhead, D. Lowe, Multivariable Functional interpolation and Adaptive Networks, *Complex systems, Vol. 2* pp. 321-355, 1988.

[11] J. C. Platt, A Resource Allocation Network for Function Interpolation, *Neural Computation* 3(2), pp. 213--225, 1991.

[12] Y. Lu, N. Sundararajan, P. Saratchandran, Analysis of Minimal Radial Basis Function Network Algorithm for Real-time identification of nonlinear dynamic systems, *IEEE. Proc. Contr. Theory and application, Vol. 4,* no. 147, pp. 476, 2000.

[13] E. B. Kosmatopulos, M. M. Polycarpou, M. A. Christodoulou, P. A. Joannou, High-order neural Network structures for identification of dynamical systems, *IEEE Trans. Neural Networks, vol. 6,* no. 2, 1995.

[14] Li. Yan, N. Sundararajan, P. Saratchandran, Dynamically Structured Radial basis Function Neural Networks for robust aircraft flight control, *Proc. American Control Conference,* pp. 3501-3505, Chicago, 2000

Appendix A.2

Authors:

Napolitano, M.R., Younghwan A., Seanor, B.

Title:

"A Fault Tolerant Flight Control Systems for Sensor and Actuator Failures Using Neural Networks"

Aircraft Design Journal, Volume 3, 2000, pg. 103-128.

PERGAMON

Aircraft Design 3 (2000) 103–128

AIRCRAFT DESIGN

www.elsevier.com/locate/airdes

A fault tolerant flight control system for sensor and actuator failures using neural networks☆

Marcello R. Napolitano*, Younghwan An, Brad A. Seanor

Department of Mechanical and Aerospace Engineering, West Virginia University, P.O. Box 6106, Morgantown, WV 26506/6106, USA

Abstract

In recent years neural networks have been proposed for identification and control of linear and non-linear dynamic systems. This paper describes the performance of a neural network-based fault-tolerant system within a flight control system. This fault-tolerant flight control system integrates sensor and actuator failure detection, identification, and accommodation (SFDIA and AFDIA). The first task is achieved by incorporating a main neural network (MNN) and a set of n decentralized neural networks (DNNs) to create a system with n sensors which has the ability to detect a wide variety of sensor failures. The second scheme implements the same main neural network integrated with three neural network controllers. The contribution of this paper focuses on enhancements of the SFDIA scheme to allow the handling of soft failures as well as addressing the issue of integrating the SFDIA and the AFDIA schemes without degradation of performance in terms of false alarm rates and incorrect failure identification. The results of the simulation with different actuator and sensor failures with a non-linear aircraft model are presented and discussed.

1. Introduction

The relatively low procurement of high-performance military aircraft along with the cancellation of plans for building new aircraft has renewed interest in fault-tolerant flight control systems with capabilities for accommodating sensor and actuator failures. For military purposes it is clear that such a feature can increase the chances of survivability following a control surface failure in a combat scenario. For scientific applications, much emphasis has been placed on the design of

☆Aviation Operation and Safety Center.

* Corresponding author. Tel.: 1-304-293-4111 ext 2346; fax: 1-304-293-6689.

E-mail address: napolit@cemr.wvu.edu (M.R. Napolitano).

1369-8869/00/$ - see front matter
PII: S1369-8869(00)00009-4

Nomenclature

a	acceleration (ft/s^2)
k	discrete time index
m	pattern for the neural network input data
p	aircraft angular velocity around the x body axis (roll rate), rad/s
q	aircraft angular velocity around the y body axis (pitch rate), rad/s
r	aircraft angular velocity around the z body axis (yaw rate), rad/s
R	auto- or cross-correlation function
t	time, s
O	neural network output
Y	parameters to be estimated

Greek letters

α	angle of attack, rad or deg
β	angle of sideslip, rad or deg
θ	pitch Euler angle, rad or deg
δ	control surface deflection, rad or deg
ϕ	roll Euler angle, rad or deg
ψ	yaw Euler angle, rad or deg

Subscripts

A	aileron
E	elevator
L	left side
R	right side
R	rudder

Acronyms

AFA	actuator failure accommodation
AFDI	actuator failure detection and identification
AFDIA	actuator failure detection, identification, and accommodation
BPA	back propagation algorithm
DNN	decentralized neural network
DQEE	decentralized quadratic estimation error
EBPA	extended back propagation algorithm
FDI	failure detection and identification
MNN	main neural network
MQEE	main quadratic estimation error
NNC	neural network controller
OQEE	output (of NN) quadratic estimation error
PTBU	(neural) parameters to be updated (at each computational step)
SFA	sensor failure accommodation
SFDI	sensor failure detection and identification
SFDIA	sensor failure detection, identification, and accommodation

fault-tolerant flight control systems for low weight and unmanned aerial vehicles (UAVs) used for remote sensing purposes. In general, a full-fault-tolerant flight control system needs to perform:

- Sensor failure detection, identification, and accommodation (SFDIA) [1–4];
- Actuator failure detection, identification, and accommodation (AFDIA) [5–10].

Furthermore, the SFDIA task can be divided into:

- sensor failure detection, identification (SFDI), which monitors the degree of deterioration in the accuracy of the sensors.
- sensor failure accommodation (SFA), which replaces the faulty sensor with an appropriate estimation.

Similarly, the AFDIA task can be divided into:

- actuator failure detection and identification (AFDI), which detects significant abnormalities and searches for the cause or for a set of probable causes;
- actuator failure accommodation (AFA), which determines on-line what actions should be taken to recover the impaired aircraft.

Sensor failure detection and identification (SFDI) is an important issue, particularly when the measurements from a failed sensor are used in the feedback loop of a control law. Since the aircraft control laws need sensor feedback to set the current dynamic state of the airplane, even slight sensor inaccuracies, if left undetected and unaccommodated for, can lead to closed-loop instability, potentially leading to unrecoverable flight conditions. For SFA purposes, most of today's high-performance military aircraft as well as commercial jetliners implement a triple physical redundancy in their sensor capabilities. However, when reduced complexity, lower cost, and weight optimization are of concern, an analytical sensor redundancy approach is more appealing.

In terms of the AFDIA problem, an actuator failure may imply a locked surface, a missing part of the control surface, or a combination of both. In increasing order of severity, an actuator failure alters trim conditions and induces a dynamic coupling followed by deterioration of dynamic stability potentially leading to unrecoverable flight conditions. The final objective of the AFDIA is to achieve, in increasing order of importance, a lower failure-induced handling qualities degradation rate, a lower mission abort rate, and a lower aircraft loss rate.

Most of the early research work on the actuator failure detection problem was based on the application of Kalman filter techniques [11–16]. These FDI techniques, such as Generalized Likelihood Ratio and Multiple-Model Kalman Filtering, perform a continuous monitoring of the measurements from the sensors. At nominal conditions, these signals follow known patterns with a certain degree of uncertainty due to system and measurement noises. However, when sensor or actuator failures occur, the measurements deviate from the predictable trajectories computed on-line or off-line from state estimation schemes. The main problems associated with the application of these failure detection schemes are their suitability only for linear time invariant systems and their applicability only when the system model is identical to the filter or observer model and/or with a high signal-to-noise ratio. As an alternative approach, the implementation of Neural Networks (NNs) to the SFDIA and AFDIA problems has lately been proposed and developed in recent years [3,5–7,10]. A NN-based SFDIA has also been simulated off-line with actual aircraft flight data [4].

Within a more general overview, in recent years NNs have been proposed for identification and control of linear and non-linear dynamic systems [17–22]. An aircraft system in certain phases of its flight envelope can be considered a time-varying, non-linear system with measurement and system noises. Therefore, the control of such a system can be attempted with an adaptive scheme. For NN applicability to adaptive control systems, the following properties are important [3,20–23]:

- *Learning and adaptation*: NNs can be trained using past recorded or simulated data (off-line training) or current data (on-line learning).
- *Applicability to non-linear systems*: The applicability of NNs to non-linear systems originates from their demonstrated mapping capabilities [23].

- *Application to multivariable systems*: NNs are multi-input, multi-output (MIMO) entities and this, naturally, leads to their application to multivariable systems.
- *Parallel distributed processing and hardware implementation*: NNs have an inherent parallel architecture which, naturally, leads to high-speed parallel hardware implementations.

It is clear that these properties of NNs are very appealing for the purpose of providing fault-tolerance capabilities in a flight control system following sensor and/or actuator failures. A critical design choice in the use of NNs for both estimation and control purpose is on-line learning vs. off-line learning. Off-line learning implies that the NNs have a "frozen" numerical architecture and do not take advantage of the additional learning, which can be provided by on-line data. With on-line learning neural networks different issues are of concern. Among these issues there is the necessary amount of time required to achieve an acceptable learning level, the difference between localized and global learning, and the level of complexity of the NN architecture. Both these problems are related to the learning algorithm. Therefore, the performance and the acceptability of an on-line learning NNs depend on the performance of its training algorithm. In this paper we will consider Multi-Layer Perceptron NNs (MLP NNs or simply NNs).

MLP NNs have been used as estimators and/or controllers to model and/or control complex non-linear systems due to their approximation and adaptation capabilities. Within a larger picture MLP NNs can be considered as a class of approximators. Other types of approximators include polynomials, spline functions, rational functions, as well as a different class of NNs, known as radial basis function (RBF NNs). An important effort toward creating a unified approximation theory was recently undertaken [24,25]. The critical need for such a unified approach can be understood if one considers the virtually infinite number of degrees of freedom in the selection of the approximation structure. Furthermore, a systematic procedure for creating non-linear approximators as well as stable learning schemes using Lyapunov's theory was introduced.

To date, for MLP NNs the Back-Propagation algorithm (BPA), a gradient-based optimization method, has been widely used as a training algorithm for the NN architecture. However, limited learning speed and local minimum points are well-known drawbacks of the BPA [19]. Different researchers have proposed several algorithms and/or procedures for dealing with these problems. The approach used by the authors consists in using a heterogeneous network, meaning that each neuron in the hidden and output layers of the NN has the capability of updating the output range (upper and lower bounds U, L) and the slope of the sigmoid activation function (T) as given

$$f(x, U, L, T) = \frac{U - L}{1 + e^{-x/T}} + L \tag{1}$$

where "x" is the same argument as in the BPA sigmoid activation function. The relative learning algorithm has been named the Extended Back-Propagation algorithm (EBPA) and has demonstrated substantial performance improvements with respect to the BPA in terms of accuracy and learning speed [26].

This paper is very application oriented and focuses on outlining the improvements of a modified SFDIA scheme (with respect to the original scheme introduced by the first author) as well as addressing the problem of the integration between the SFDIA and the AFDIA schemes. The second is a very original topic in the fault-tolerance literature. It should also be underlined that

different neural paradigms and/or algorithms could be used within the SFDIA and the AFDIA schemes without any loss of generality.

It should be emphasized that this paper does not deal with a critical problem of NN-based fault-tolerant control laws, that is the validation and certification (V&C) of NN-based control laws. The V&C for these neural schemes must start from a clear definition of the SFDIA and AFDIA system requirements followed by the introduction of ad-hoc analytical tools for assessing the closed-loop stability of the system leading to the final system certification. These issues are not trivial and their resolution is a critical phase prior to the actual implementation of artificial intelligence fault-tolerant control schemes on on-board computers [27–30].

The paper is organized as follows: the next sections review the SFDIA and AFDIA schemes followed by a section describing the results of the simulations for sensor and actuator failures. The final section summarizes the paper.

2. Neural network-based sensor failure detection, identification, and accommodation

Using on-line learning NN estimators, the SFDIA problem can be approached by introducing multiple feed-forward NNs trained on-line with the EBPA. Particularly, the scheme consists of a main NN (MNN) and a set of n decentralized NNs (DNNs), where n is the number of the sensors in the flight control system without physical redundancy. The outputs of the MNN are the estimates of the same parameters measured by the n sensors at time "k", using measurements from time instant "$k-1$" to "$k-m$"; these estimates are compared with the actual measurements at time "k". For the ith of the n DNNs, the output is the estimate of the measurement of the ith sensor, that is, the prediction of the state at time "k", using measurements from "$k-1$" to "$k-m$" to be compared with the actual measurement at time "k". The inputs to the ith DNN are the measurements from any number to up "$n-1$" sensors, in other words, all the n sensors excluding the ith one.

The performance of this NN-based SFDIA scheme were compared in a previous study with the performance of an identical scheme with Kalman filters in lieu of the MNN and the DNNs [31]. This comparison showed the advantages of the on-line learning by the NN-SFDIA scheme vs. the robustness of the Kalman Filter SFDIA scheme for modeling discrepancies between the actual system and the filter model; additionally the study showed a similar level of computational effort for on-line applications.

For SFD purposes, when a quadratic estimation error parameter from the MNN exceeds some predefined threshold at a certain time instant, the scheme deduces that a sensor failure may be occurring or has already occurred. Following the positive sensor failure detection, the learning for each DNN is halted; then, a quadratic estimation error parameter from the DNNs exceeding, at the same time instant, another threshold provides the identification. For the accommodation phase, the ith DNN output is used to replace the measurement from the faulty sensor; ith DNN output is also used as input to the MNN for the purpose of allowing the MNN to provide detection capabilities until the end of the flight. This output is also passed to all other DNNs using the ith sensor as an input parameter. This "double trigger" approach using both MNN and DNNs has the purpose of reducing the rate of false alarms in the FDI process. Several options can be added to this scheme to add robustness for noisy measurements and/or intermittent sensor failures. For example, a lower and a higher threshold level can be introduced for the DNNs. If the estimation error for the

ith DNN exceeds the lower threshold once, the status of the corresponding ith sensor is declared "suspect" and the numerical architecture of the ith DNN is not updated. Should this status continue for a certain number of time instants and/or the estimation error in successive time instants exceeds the higher threshold, then the sensor is declared failed and is, therefore, replaced by the output from the ith DNN. Fig. 1 shows a block diagram of the SFDIA process.

For simplicity purposes, let us consider failures only for the pitch rate, the roll rate, and the yaw rate gyros. As for any other FDIA approach, the following distinct capabilities are critical:

- failure detectability and false alarm rate for SFDI purposes;
- estimation error for SFA purposes.

The following quadratic parameter is used for sensor failure detection (SFD) purposes:

$$\mathrm{MQEE}(k) = \frac{1}{2}\sum_{i=1}^{\text{Num. of DNNs}} (Y_i(k) - O_{i,\mathrm{MNN}}(k))^2$$

$$= \frac{1}{2}[(p(k) - \hat{p}_{\mathrm{MNN}}(k))^2 + (q(k) - \hat{q}_{\mathrm{MNN}}(k))^2 + (r(k) - \hat{r}_{\mathrm{MNN}}(k))^2]. \tag{2}$$

The sensor failure identification (SFI) can instead be achieved by monitoring the absolute value of the estimation error of each DNN, defined as

$$\mathrm{DQEE}_x(k) = \tfrac{1}{2}(x(k) - \hat{x}(k))^2 \quad \text{where } x = p, q, r. \tag{3}$$

For sensor failure accommodation (SFA) purposes, the following classic parameters for the estimation error are instead evaluated.

$$\mathrm{DAEE}_x = \frac{1}{N}\sum_{k=1}^{N} (x(k) - \hat{x}(k)), \tag{4}$$

$$\mathrm{DVEE}_x = \frac{1}{N}\sum_{k=1}^{N} [(x(k) - \hat{x}(k)) - \mathrm{DAEE}_x]^2 \quad \text{where } x = p, q, r, \tag{5}$$

where DAEE and DVEE represent the DNN estimation error mean and variance respectively. The "N" refers to the number of time steps from the instant when failure of sensor is declared to the end of the simulation.

In the case of a step-type sensor failure, the desirable detection capabilities are provided by the peak value of the MQEE parameter regardless of bias magnitude. However, in the case of soft sensor failures, particularly for ramp-type sensor failures, the MQEE-criterion has not shown reliable detection capabilities [4]. Therefore, a modification has been introduced to ensure desirable performance under conditions resulting from any type of failures. This modified SFDIA scheme makes use of a different NN quadratic estimation error defined by

$$\mathrm{OQEE}(k) = \frac{1}{2}\sum_{i=1}^{\text{Num. of DNNs}} (O_{i,\mathrm{MNN}}(k) - O_{i,\mathrm{DNN}}(k))^2$$

$$= \frac{1}{2}[(\hat{p}_{\mathrm{MNN}}(k) - \hat{p}_{\mathrm{DNN}}(k))^2 + (\hat{q}_{\mathrm{MNN}}(k) - \hat{q}_{\mathrm{DNN}}(k))^2 + (\hat{r}_{\mathrm{MNN}}(k) - \hat{r}_{\mathrm{DNN}}(k))^2], \tag{6}$$

where $\hat{p}_{\mathrm{DNN}}$, $\hat{q}_{\mathrm{DNN}}$, and $\hat{r}_{\mathrm{DNN}}$ are the estimates of p, q, and r from the respective DNNs.

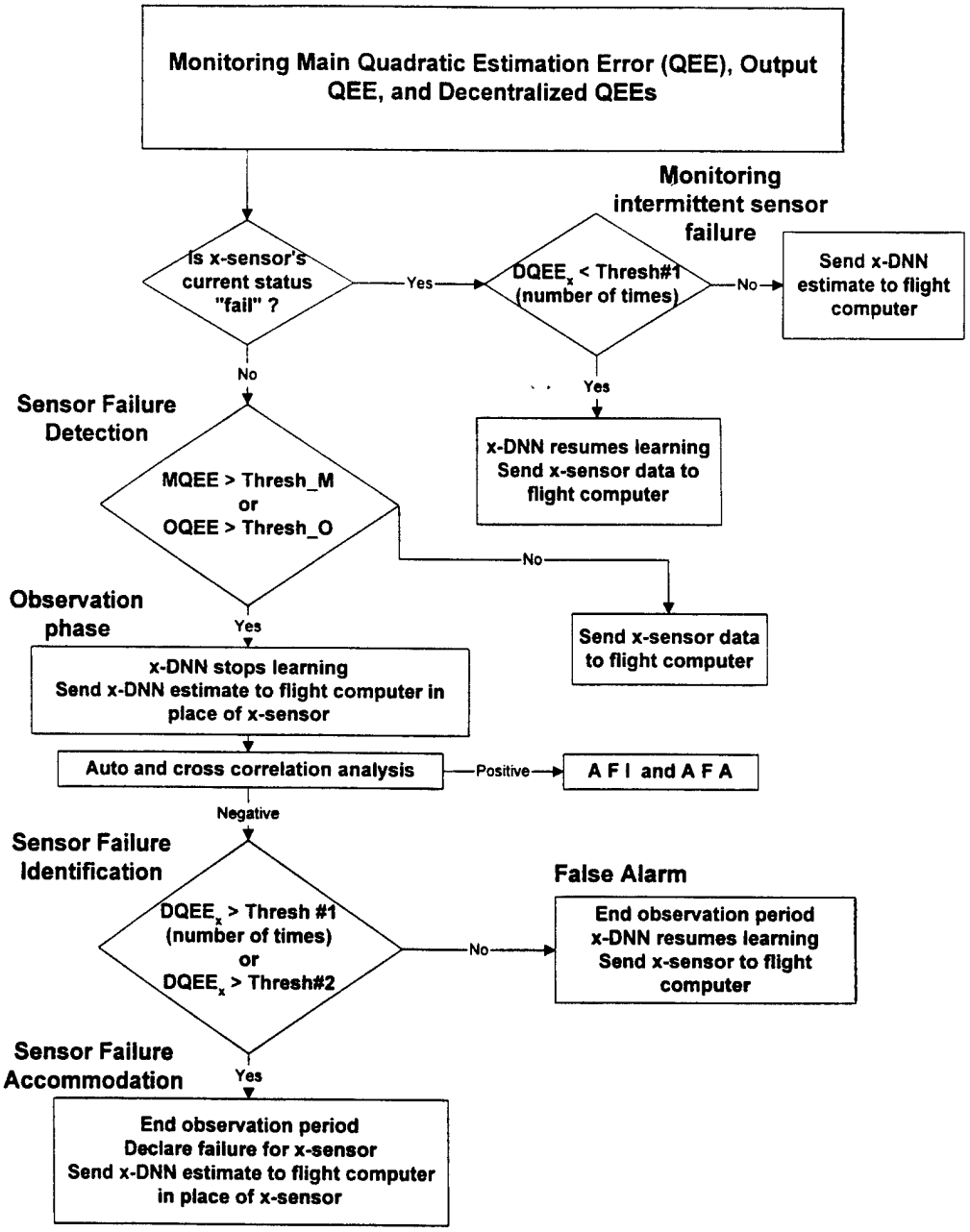


Fig. 1. Block diagram of the SFDIA scheme.

The need for the use of this new parameter for SFD purposes is better described by an analysis of a typical sensor failure. For example, when a pitch rate gyro fails, the three parameters $q(k)$, $\hat{q}_{\mathrm{MNN}}(k)$, and $\hat{q}_{\mathrm{DNN}}(k)$ within the "MQEE" and "OQEE" are considered. At nominal conditions $\hat{q}_{\mathrm{MNN}}(k)$ and $\hat{q}_{\mathrm{DNN}}(k)$ are estimated by the MNN and q-DNN, respectively, to emulate the actual pitch rate gyro. In the event of a ramp-type q-gyro failure, the $\hat{q}_{\mathrm{MNN}}$ tends to resemble the corrupted

Table 1
Architecture of the NNs for the SFDIA scheme

Parameter	MNN	q-DNN	p-DNN	r-DNN
Input data	$p, q, r, \phi, \theta, \delta_E, \delta_A, \delta_R$	$\alpha, a_n, a_x, \dot{w}$	$r, \dot{p}, \dot{r}, \phi, \delta_A, \delta_R$	$\beta, a_y, \dot{v}, \phi, \delta_A, \delta_R$
Input data pattern	5	5	5	5
Total no. of inputs	40	20	30	30
No. of hidden layer	1	1	1	1
No. of neurons in HL	25	20	30	18
No. of outputs	$3(\hat{p}, \hat{q}, \hat{r})$	$1(\hat{q})$	$1(\hat{p})$	$1(\hat{r})$
Learning rates[a]	0.1	0.01	0.01	0.01
	(0.1)	(0.0004)	(0.0004)	(0.001)
Momentum coefficients[a]	0.05	0.005	0.005	0.005
	(0.05)	(0.0002)	(0.0002)	(0.0005)
PTBU[b]	1187	504	1054	616

[a]Upper value is for pre-training, (lower) value is for on-line learning.
[b]Number of parameters to be updated at each iteration for the NN in the on-line learning mode.

signals from the gyro. In fact, since the pitch rate gyro is included as the input parameter in MNN (see Table 1), the MNN architecture is updated with the failed q-gyro values during the on-line learning. As a result, MQEE does not provide an accurate detection because the difference between $q(k)$ and $\hat{q}_{MNN}(k)$ is relatively small despite the gyro failure. However, the output of the q-DNN ($\hat{q}_{DNN}(k)$) in Eq. (6) follows the nominal "q" value (that is the value as it would be without a failure) relatively well despite the sensor failure. This is because the q-DNN does not receive, as input data, measurements from the faulty sensor, as shown in Table 1. The discrepancy between $\hat{q}_{MNN}$ and $\hat{q}_{DNN}$ in Eq. (6) causes, therefore, the peak of 'OQEE'. This problem does not occur for the step-type sensor failures. In fact, for these failures $\hat{q}_{MNN}(k)$ is not consistent with the measurement from the failed sensor, which induces therefore a peak for MQEE.

In general, it can be concluded that the MQEE provides better performance for step-type sensor failures whereas the OQEE performs better for ramp-type of sensor failures. The results of comparative studies for these detection schemes are shown in one of the following sections. It is concluded that a combined detection scheme using both "MQEE" and "OQEE" provides more reliable detection capability for the SFDIA for any type of failures.

3. Neural network-based actuator failure detection, identification, and accommodation

The occurrence of any actuator failure also implies that the parameter 'MQEE' defined above exceeds a selected threshold. Thus, the actuator failure detection (AFD) can be achieved by spotting substantial changes in the aircraft angular velocities following any type of actuator failure. Next, the actuator failure identification (AFI) can be performed by analyzing specific cross-correlation functions. In general, for two random processes $Y(k)$ and $X(k)$, a cross-correlation function is defined by

$$R_{YX}(n) = E[Y(k)X(k+n)]. \tag{7}$$

For AFI purposes the cross-correlation functions R_{pq}, R_{pr} and R_{qr} are used, taking advantage of the fact that any type of actuator failure on any aircraft control surface involves a loss of symmetry. This loss is then followed by a dynamic coupling between longitudinal and lateral-directional aircraft dynamics. Furthermore, the auto-correlation function, R_{rr}, has shown capabilities as a useful identification tool in the event of a rudder actuator failure.

Following a positive AFD and AFI, the immediate objective is to regain equilibrium and to compensate for the pitching, rolling, and yawing moments induced by the failure. Toward this goal, three separate NN controllers are introduced: a NN pitch controller, a NN roll controller, and a NN yaw controller.

The output of the NN pitch controller (NNC pitch) is the compensating deflection for the remaining healthy elevator (elevator failure case) or the symmetric elevators (aileron and/or rudder failure case). The on-line learning for the NN pitch controller is initiated as the simulation starts. Under nominal conditions, the controller is trained to emulate the actual control deflections for the symmetric elevators. Therefore, it minimizes the cost function

$$J_{\text{pitch}_{\text{nom}}} = (\delta_{H_{\text{L.R}}} - \hat{\delta}_{H_{\text{L.R}}}). \tag{8}$$

Following a positive AFD and AFI showing the need for a longitudinal AFA, the on-line learning NN pitch controller switches its target to minimize the cost function

$$J_{\text{pitch}_{\text{AFA}}} = k_1(q - q_{\text{ref}}) + k_2(\theta - \theta_{\text{ref}}) + k_3(a_{\text{n}} - a_{\text{n}_{\text{ref}}}), \tag{9}$$

where $q_{\text{ref}} = 0$, $\theta_{\text{ref}} = \theta_{\text{trim}}$, and $a_{\text{n}_{\text{ref}}} = 1$.

It should be noted that this cost function resembles a controller with a PID error formulation. It should also be mentioned that the on-line learning at nominal conditions (that is with the AFDIA scheme inactive) has no physical meaning; in fact, the NN pitch controller is just "emulating" control deflections at nominal conditions. However, this procedure has shown the benefit of improving the transient response by having the NN output within the same order of magnitude of the NN output necessary when the AFA process is turned on following a positive AFD and AFI.

Within the AFDIA scheme the NN roll and yaw controllers operate in a similar fashion. Under nominal conditions, these controllers learn to replicate the actual control deflections for the ailerons and the rudder by minimizing the cost functions

$$J_{\text{roll}_{\text{nom}}} = (\delta_{\text{A}_{\text{L.R}}} - \hat{\delta}_{\text{A}_{\text{L.R}}}), \tag{10}$$

$$J_{\text{yaw}_{\text{nom}}} = (\delta_{\text{R}} - \hat{\delta}_{\text{R}}). \tag{11}$$

Following positive AFD and AFI, the on-line learning NN roll and yaw controllers switch their targets to minimize the cost functions

$$J_{\text{roll}_{\text{AFA}}} = k_4(p - p_{\text{ref}}) + k_5(\phi - \phi_{\text{ref}}), \tag{12}$$

$$J_{\text{yaw}_{\text{AFA}}} = k_6(r - r_{\text{ref}}), \tag{13}$$

where the $p_{\text{ref}} = \phi_{\text{ref}} = r_{\text{ref}} = 0$ at trim conditions. Fig. 2 shows a block diagram of the AFDIA process while Fig. 3 shows a combined block diagram of the SFDIA and AFDIA schemes.

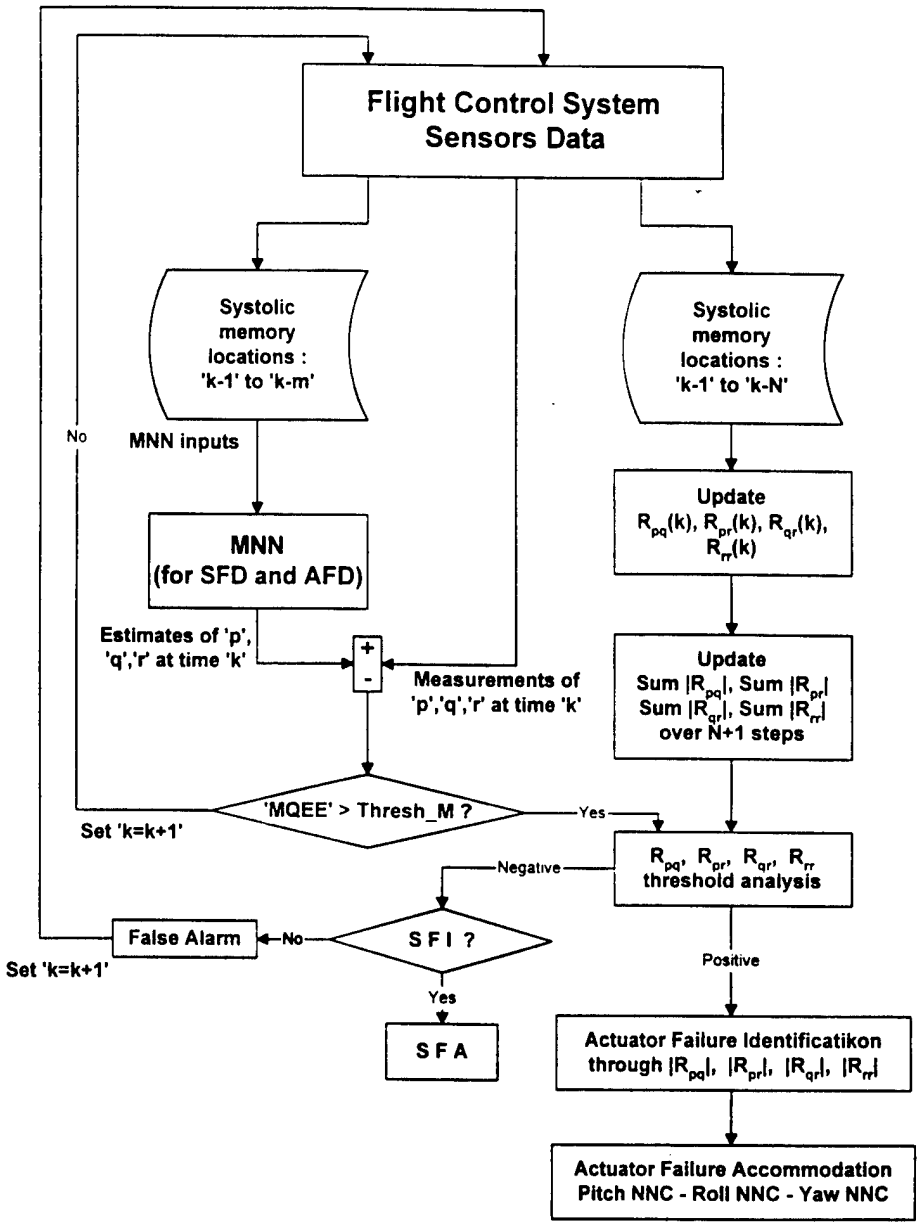


Fig. 2. Block diagram of the AFIDA scheme.

4. Simulation of the SFDIA and the AFDIA schemes

The mathematical model used for this study is the model of a B747-200 aircraft. Using aerodynamic and thrust data, a simulation code was developed [32]. This model features non-linear dynamics, linearized aerodynamics and includes system and measurement noise. The system noise is modeled as zero mean, white, Gaussian gust disturbance on the angle of attack and on the side-slip angle. The sensor noise is also assumed to be Gaussian and white. The standard deviations

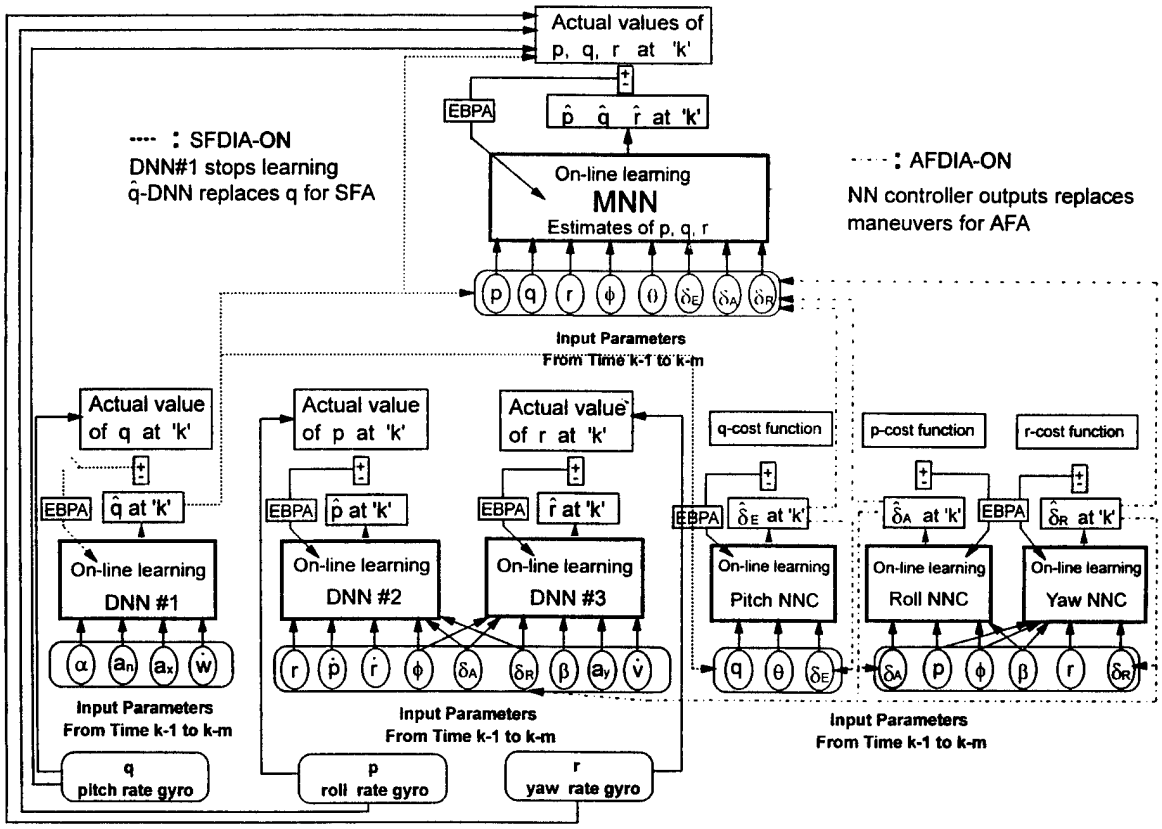


Fig. 3. Integration of SFDIA & AFDIA structure with failures of pitch rate gyro and right elevator failure.

Table 2
System noise standard deviations

Parameter	Standard deviation
α_{gust}	0.1°
β_{gust}	0.1°

for the system and sensor noise are given in Tables 2 and 3 [11]. The primary control surfaces consist of two differential elevators, two differential ailerons, and rudder.

As stated in the previous section, the SFDIA scheme is simulated only for failures of the pitch, roll, and yaw rate gyros. Therefore, the NN-based SFDIA scheme uses one main NN (MNN) with three decentralized NNs (DNNs). The architectures for the MNN and the DNNs are shown in Table 1. Note that input parameters of q-DNN do not include pitch rate measurements. Similarly, the p-DNN and the r-DNN do not use "p" and "r" as input parameters, respectively. The AFDIA scheme is simulated for failures of the actuators of elevators, ailerons, and rudder. The AFDIA scheme consists of three NN controllers and shares the MNN with the SFDIA as shown in Table 4.

Table 3
Sensor noise standard deviations

Parameter	Standard deviation
Airspeed indicator	11 ft/s
Roll rate gyro	0.14°/s
Pitch rate gyro	0.14°/s
Yaw rate gyro	0.14°/s
Longitudinal accelerometer	0.98 ft/s^2
Lateral accelerometer	0.98 ft/s^2
Directional accelerometer	0.98 ft/s^2
Attitude pitch gyro	0.57°
Attitude roll gyro	0.57°
Attitude yaw gyro	0.57°
Altitude rate indicator	0.25 ft/s
Altitude indicator	10 ft
Angle of attack	0.1°
Sideslip angle	0.1°

A generic sensor failure can be modeled as

$$x_{\text{failure},i} = x_{\text{nom},i} = \rho n_i, \tag{14}$$

where n_i is the direction vector for the ith faulty sensor, and ρ is the magnitude of the failure which can be positive or negative

- for step-type sensor failures, $n_i = 1$,
- for ramp-type sensor failures,

$$n_i = \begin{cases} \dfrac{t - t_{f1}}{t_{f2} - t_{f1}}, & t_{f1} \le t \le t_{f2}, \\ 1, & t \ge t_{f2}, \end{cases}$$

where t_{f1} and t_{f2} indicate the initial and final time instant of ramp-type sensor failure, respectively. The different types of sensor failures considered in this study based on the above formula are represented by:

Type #1: large sudden bias (2.5°/s);
Type #2: small sudden bias (1.0°/s);
Type #3: large drifting bias (2.5°/s) with fast transient period (0.3 s);
Type #4: small drifting bias (1.0°/s) with fast transient period (0.3 s);
Type #5: large drifting bias (2.5°/s) with slow transient period (4 s);
Type #6: small drifting bias (1.0°/s) with slow transient period (4 s).

The maximum control surface deflections in the simulation code are ± 15° for the elevators and ± 20° for the ailerons and the rudder. The code also models the actuation rate for each of the

Table 4
Architecture of the NNs for the AFDIA scheme

Parameter	MNN	Pitch NC	Roll NC	Yaw NC
Input data	$p, q, r, \phi, \theta, \delta_E, \delta_A, \delta_R$	q, θ, δ_E	p, ϕ, β, δ_A	$r, p, \beta, \phi, \delta_R$
Input data pattern	5	2	3	3
Total no. of inputs	40	6	12	15
No. of hidden layer	1	1	1	1
No. of neurons in HL	25	20	20	20
No. of outputs	$3(\hat{p}, \hat{q}, \hat{r})$	$1(\hat{\delta}_{E(L,R)})$	$1(\hat{\delta}_{A(L,R)})$	$1(\hat{\delta}_R)$
Learning rates	0.1	0.2	0.4	0.5
Momentum coefficients	0.05	0.1	0.1	0.3
PTBU	1187	224	344	404

control surfaces with a maximum deflection rate of $\pm 20°/s$. The actuator failures are assumed to occur randomly during high-speed cruise conditions. The considered actuator failures are given by:

Case #1: stuck R/L elevator at trimmed/untrimmed deflections without missing surface;
Case #2: stuck R/L elevator at trimmed/untrimmed deflections with missing surface;
Case #3: stuck R/L aileron at trimmed/untrimmed deflections without missing surface;
Case #4: stuck R/L aileron at trimmed/untrimmed deflections with missing surface;
Case #5: stuck rudder at trimmed/untrimmed deflection without missing surface;
Case #6: stuck rudder at trimmed/untrimmed deflection with missing surface.

It is important to note that the mathematical modeling of the actuator failures in the event of a missing part of the control surface is derived through a set of closed-form expressions for the non-dimensional stability and control derivatives. These expressions are functions of the normal force coefficient of the control surface whose actuator is assumed to have failed [7,10].

The numerical simulation starts at typical cruise conditions, defined by an altitude of 40,000 ft and airspeed of 871 ft/s. An on-line learning process is simulated following 30,000 s of off-line training. A maneuver lasting 100 s is then flown for each rate gyro failure case with a sensor failure occurring at exactly 50 s. The results of the comparative studies between two detection parameters for the q, p, and r rate gyro failures are shown in Tables 5 and 6.

From Table 5, it can be seen that the MQEE-criterion provides a quicker detection than the OQEE-criterion for the step-type sensor failure (Types #1 and #2), regardless of the magnitude of the step failure. Clearly, a larger bias induces a larger MNN error spike and the failure is declared instantaneously at 50 s for Type #1 and around 50.48 s for Type #2, depending on the size of the rate gyro-DNN error. For the ramp-type failures (Types #3–#6), regardless of the magnitude reached after the ramp, the MQEE-criterion does not trigger the failure detection. On the other side, the OQEE-criterion provides a detection for the SFDIA scheme due to the difference of output error between the MNN and the DNNs in Eq. (6). From the results presented in Table 5, the MQEE-criterion seems more suitable for step-type sensor failures while the OQEE-criterion performs better for ramp-type sensor failures. Thus, a "double detection trigger" (MQEE and/or OQEE) can provide desirable performance for any sensor failure types.

Table 5
Results of detection time between two detection parameters Rate gyro failure is occurred at $t = 50$ s

Sensor failure type	MQEE only			OQEE only			MQEE + OQEE		
	q	p	r	q	p	r	q	p	r
Type #1	50.00	50.00	50.00	50.02	50.02	50.04	50.00	50.00	50.00
Type #2	50.48	50.50	50.48	50.52	50.54	50.56	50.48	50.50	50.48
Type #3	[a]	[a]	[a]	50.26	50.24	50.44	50.26	50.24	50.44
Type #4	[a]	[a]	[a]	50.80	50.78	51.00	50.80	50.78	51.00
Type #5	[a]	[a]	[a]	52.10	51.66	51.56	52.10	51.66	51.56
Type #6	[a]	[a]	[a]	54.56	53.26	53.42	54.56	53.26	53.42

[a]No sensor failure declared.

Table 6
Results of comparative studies for estimation error

Sensor failure type		MQEE only			MQEE + OQEE		
		q	p	r	q	p	r
Type #3	Mean	−0.805	−1.919	−1.584	−0.096	−0.104	−0.023
	Vari.	0.146	0.428	0.416	0.006	0.028	0.017
Type #4	Mean	−0.363	−0.790	−0.644	−0.096	−0.108	−0.027
	Vari.	0.022	0.083	0.071	0.006	0.027	0.019
Type #5	Mean	−0.778	−1.879	−1.560	−0.103	−0.134	−0.036
	Vari.	0.140	0.447	0.421	0.006	0.027	0.023
Type #6	Mean	−0.356	−0.792	−0.646	−0.116	−0.176	−0.063
	Vari.	0.022	0.072	0.069	0.010	0.027	0.017

Table 6 shows the result of a comparative study in terms of estimation error mean and variance (Eqs. (4) and (5)) between the "MQEE" and "OQEE" detection criteria. These values were computed from the instant after the occurrence of the failure to the end of simulation. Since the first criterion does not trigger a detection of failure for the small/large bias with slow/fast ramp, it may not make sense to talk about estimation error mean and variation during the SFA for the case when "MQEE" is used alone. However, it is important to recognize how much the estimators deviate from the nominal sensor values when the first criterion does not provide a failure detection or even a timely failure detection. The failure-declared time of the second criterion is used in the comparative study. From the results presented in Table 6, the modified detection technique shows improved performance in terms of estimation mean and variance.

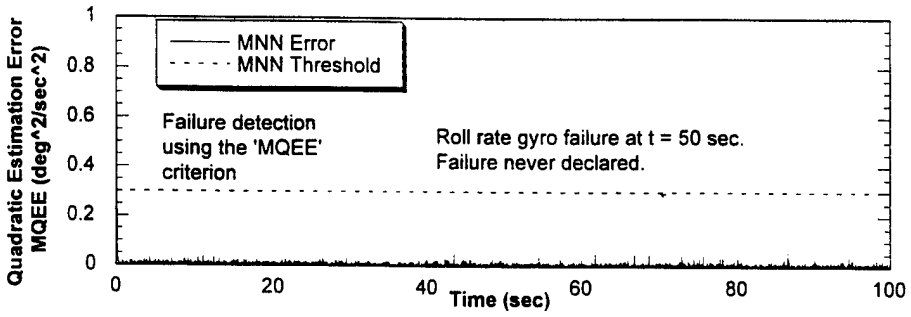


Fig. 4. Plot of MNN error vs. time for sensor failure type #6.

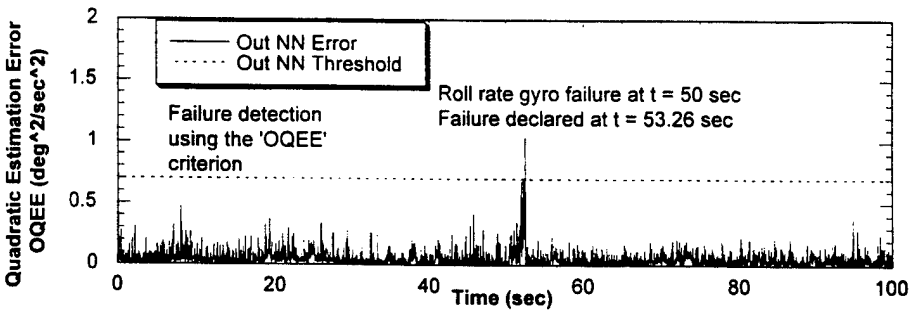


Fig. 5. Plot of Out NN error vs. time for sensor failure type #6.

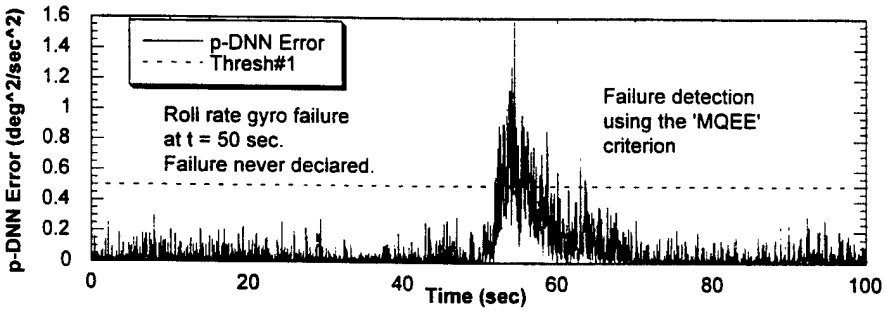


Fig. 6. Plot of *p*-DNN error vs. time for sensor failure type #6.

Graphical results of this comparative study relative to Type #6 failure are shown in Figs. 4–11. This type of failure is selected because it is believed to be the worst case scenario. In fact, the "soft" ramp-type failure may not degrade the system performance for the time after its occurrence, but if left uncompensated, can lead the system to critical and, eventually, catastrophic situations if the measurements from a faulty sensor are fed into the aircraft control laws. In Fig. 4, the MNN error

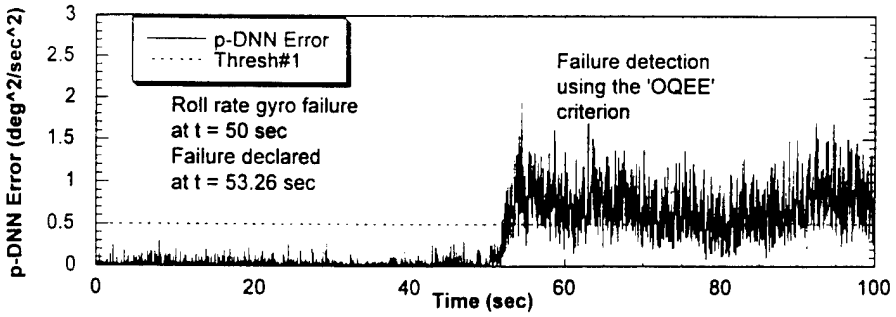


Fig. 7. Plot of p-DNN error vs. time for sensor failure type #6.

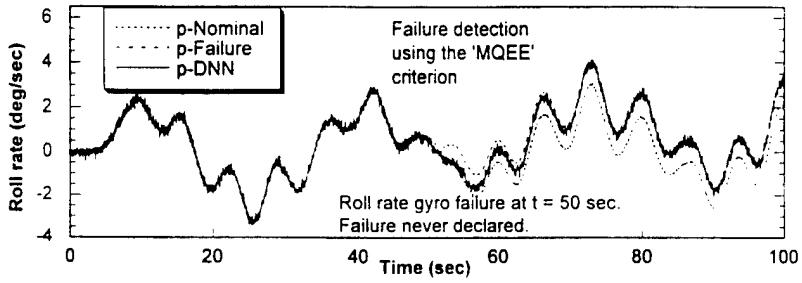


Fig. 8. Plot of p-Nominal, p-Fail, and p-DNN vs. time for sensor failure type #6.

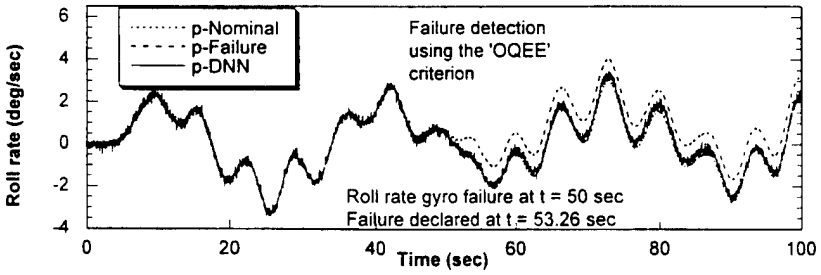


Fig. 9. Plot of p-Nominal, p-Fail, and p-DNN vs. time for sensor failure type #6.

does not exhibit a substantial detection spike although there were some DNN errors following the sensor failure, as shown in Fig. 6. Instead, Figs. 5 and 7 show that the SFDI is successful for the roll rate gyro following a sensor failure. In Fig. 5, the OQEE exceeded the predefined threshold during the observation period after sensor failure. Following a successful sensor failure identification (Fig. 7) and accommodation, the values of OQEE go back to low values as soon as the faulty roll rate gyro sensor is replaced by the estimate from the p-DNN. Fig. 8 shows that there is some decrease in the roll rate gyro-DNN's estimation accuracy, since faulty measurements for "p" are

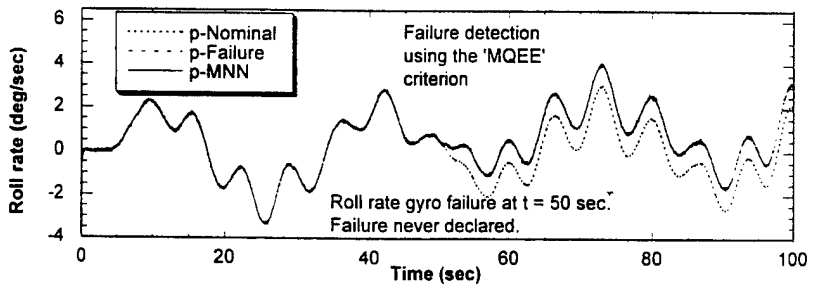


Fig. 10. Plot of p-Nominal, p-Fail, and p-MNN vs. time for sensor failure type #6.

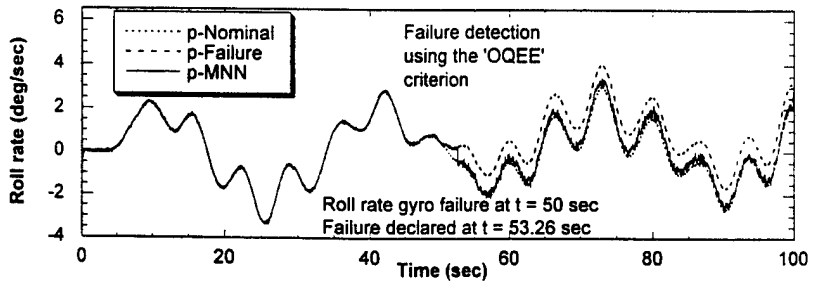


Fig. 11. Plot of p-Nominal, p-Fail, and p-MNN vs. time for sensor failure type #6.

continuously fed into the DNN's target value for on-line learning purposes. As a result, in Fig. 10, the estimate of the roll rate gyro from the MNN output still follows the corrupted p-gyro value after the sensor has failed. Note that the time histories of p-MNN and p-DNN are in agreement with the nominal time history for the roll rate gyro following successful SFD, as shown in Figs. 9 and 11.

An additional objective of this effort is to address the issues of the integration between the SFDIA and AFDIA schemes. Although both SFDIA and AFDIA problems have been extensively addressed and described in the technical literature in recent years, the authors have not been able to find a single reference describing the integration of the two schemes within a flight control systems.

As described in previous sections, the SFDIA and the AFDIA schemes share the same detection mechanism. Once the detection-alert is triggered, then the critical task is to decide among the occurrence of a sensor failure, an actuator failure, and a false alarm. For this purpose a key role is played by the trends shown by the on-line calculated and stored auto correlation (R_{rr}) and cross correlation functions (R_{pq}, R_{qr}, R_{pr}). Only when any of these functions exceeds a threshold an actuator failure declared, in lieu of a false alarm or a sensor failure. This is true even if at the same time any of the quadratic estimation errors from the DNNs is exceeding its thresholds.

As for any other failure detection scheme, the selection of the numerical values of the different thresholds is a trade-off between detection capabilities and low false alarm rate. In selecting threshold values few general rules from statistics can be used and a detailed knowledge of the system dynamics, measurement and system noise is very much required. For AFI purposes the

different levels of dynamic coupling shown by the cross-correlation functions play a major role. It suffices to say that a failure on any of the elevators actuators induces a much higher longitudinal–lateral dynamic coupling than ailerons actuators failures imply. On the other side, an actuator failure on a rudder induces a very distinctive behavior on the auto-correlation function R_{rr}. For the purpose of showing the complicated SFDIA-AFDIA integration scheme, a sequence of different failures is introduced within a 150 s simulation:

Failure #1: a pitch rate gyro failure involving large drifting bias (2.5°/s) with slow transient period (4 s) (Type #5). Failure occurrence: $t = 30$ s.

Failure #2: an actuator failure on the right elevator with a stuck surface at $+10°$ leading to a 25% reduction in aerodynamic effectiveness. Failure occurrence: $t = 60$ s.

Failure #3: a temporary roll rate gyro failure with large sudden bias (2.5°/s). Failure occurrence: $t = 90$ s.

Failure #4: an actuator failure on the rudder with a stuck surface at $-7°$ with a 25% reduction in aerodynamic effectiveness. Failure occurrence: $t = 120$ s.

Fig. 12 shows that the "OQEE" parameter exceeds the predefined threshold during the observation period. It triggers a "state-of-alert" for both actuators and sensors. However, since failure #1 is a sensor failure, this time a substantial longitudinal–lateral dynamic coupling, R_{pq}, has not occurred, as revealed in Fig. 13. Furthermore, the clear trend of the q-DNN error shown in Fig. 14

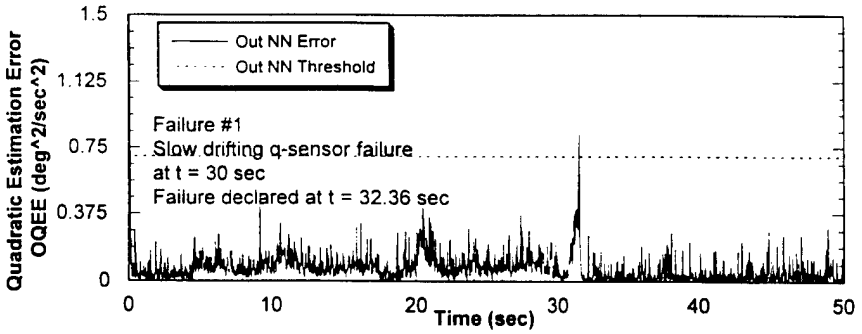


Fig. 12. "OQEE" vs. time for failure type #1 (pitch rate gyro failure at $t = 30$ s).

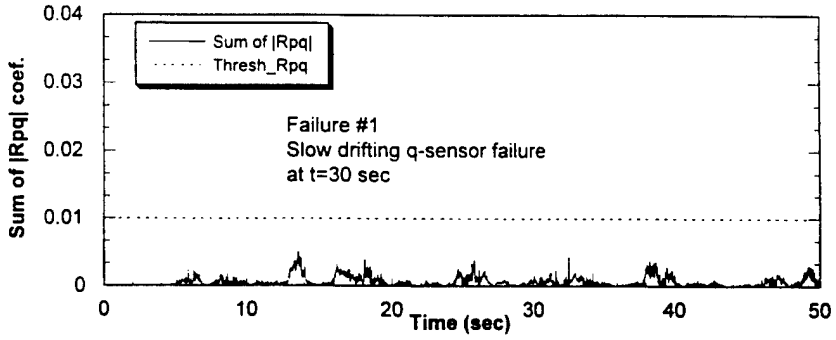


Fig. 13. Sum of the absolute values of the coefficients of R_{pq} vs. time for failure type #1 (pitch rate gyro failure at $t = 30$ s).

identifies the failure #1 as a "pitch rate gyro failure". It should be noted that from the instant q-DNN error exceeds threshold #1 the on-line learning for the q-DNN is halted; furthermore, from the instant q-DNN error exceeds threshold #2, the q-DNN error becomes meaningless since the output of the q-DNN replaces the reading from the pitch rate gyro. Fig. 15 shows a successful SFA following by a positive SFDI. Thus, the time history of q-DNN is in good agreement with the nominal value for pitch rate gyro.

Fig. 16 shows the trend of the "MQEE" parameter around the occurrence of failure #2. Although failure #2 is an actuator failure, it could be interpreted as a sensor failure. However, the suspected sensor failure is overruled by the trend of the cross-correlation functions. In other words, a comparison of the magnitudes of the sums of the absolute values of the cross-correlation R_{pq}, R_{pr} and R_{qr} allows the identification of the failure as an "elevator actuator failure", as shown in Fig. 17. It should be underlined that the decreasing trend in Fig. 17 is due to the successful AFA, following the positive AFDI, provided by the combined actions of the on-line learning pitch, roll, and yaw neural controllers. Fig. 18 shows the time history of the pitch rate, which confirms that the fault-tolerance schemes can regain equilibrium of the aircraft following the elevator actuator failure.

For failure #3, Fig. 19 shows that "MQEE" parameter exceeds its threshold triggering detection for both actuator and sensor failure. Once again, since failure #3 is a sensor failure, the sum of the absolute values of the coefficients of the cross-correlation function R_{pq} does not substantially

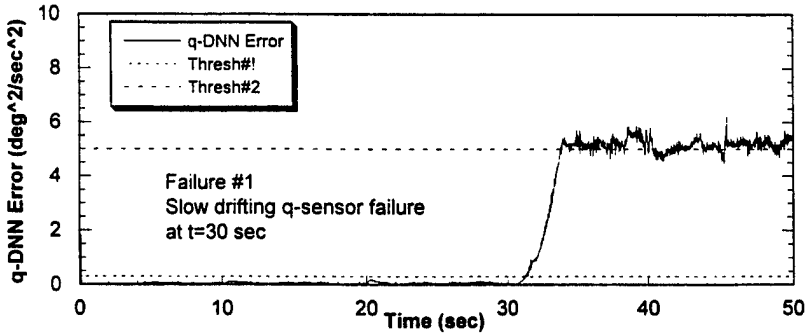


Fig. 14. "q-DNN Error" vs. time for failure type #1 (pitch rate gyro failure at $t = 30$ s).

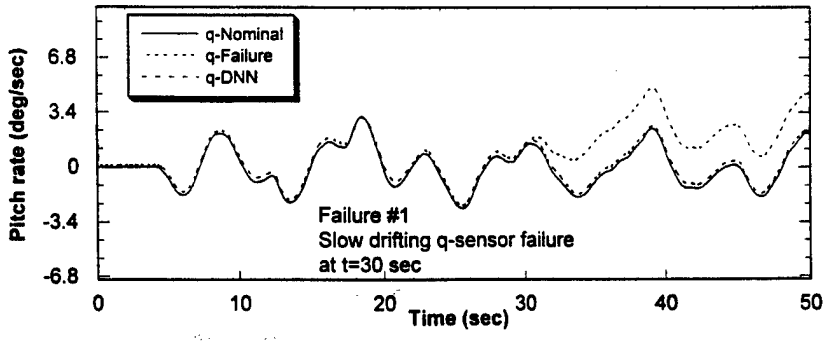


Fig. 15. "q-Failure", "q-Nominal", and "q-DNN" vs. time for failure type #1 (pitch rate gyro failure at $t = 30$ s).

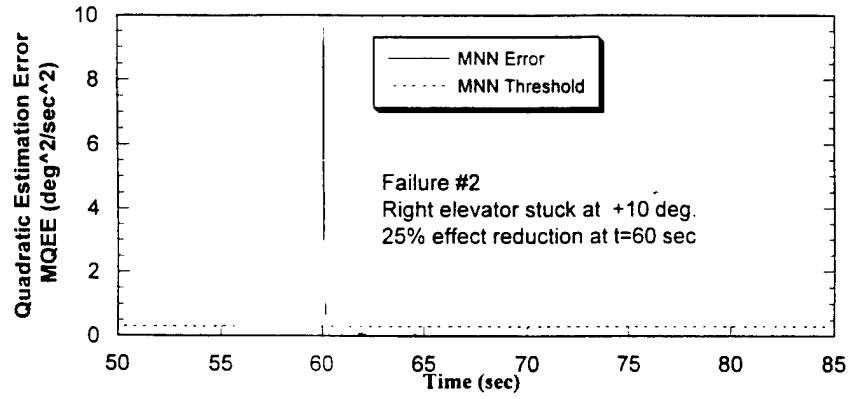


Fig. 16. "MQEE" vs. time for failure type #2 (right elevator failure at $t = 60$ s).

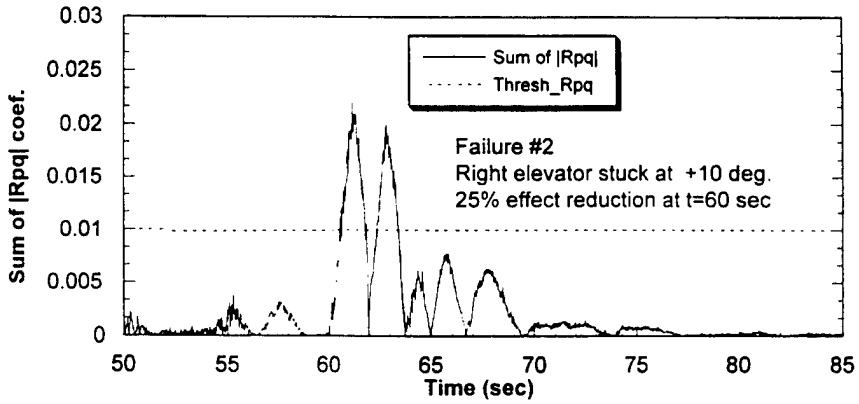


Fig. 17. Sum of the absolute values of the coefficients of R_{pq} vs. time for failure type #2 (right elevator failure at $t = 60$ s).

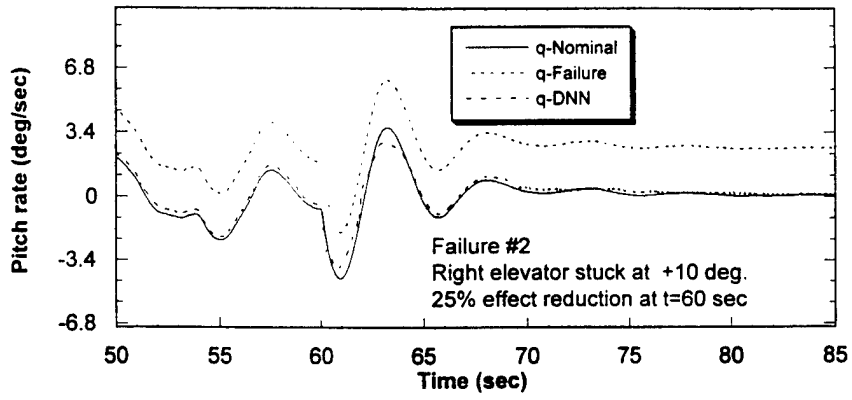


Fig. 18. "q-Failure", "q-Nominal", and "q-DNN" vs. time for failure type #2 (right elevator failure at $t = 60$ s).

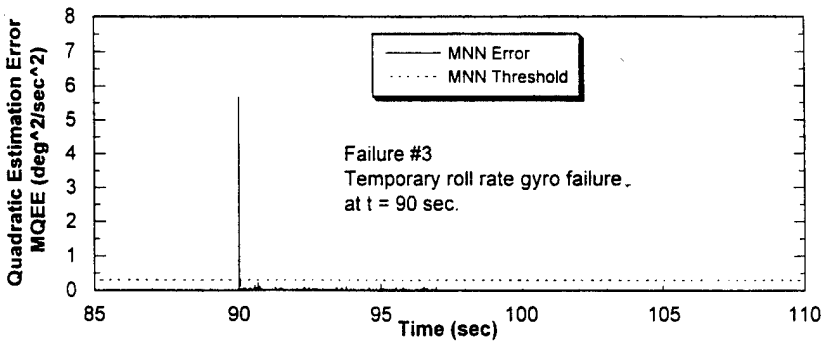


Fig. 19. "MQEE" vs. time for failure type #3 (roll rate gyro failure at $t = 90$ s).

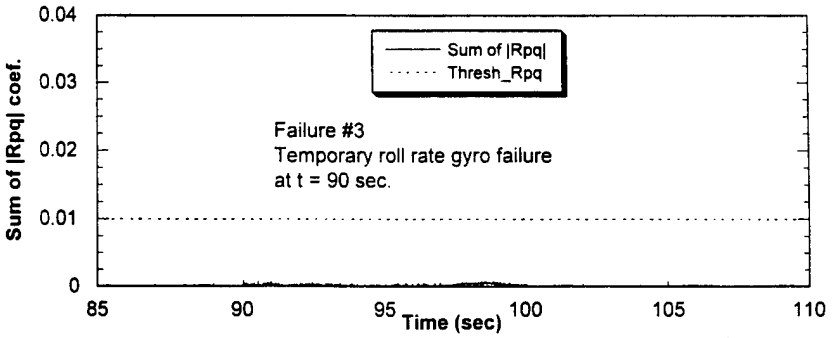


Fig. 20. Sum of the absolute values of the coefficients of R_{pq} vs. time for failure type #3 (roll rate gyro failure at $t = 90$ s).

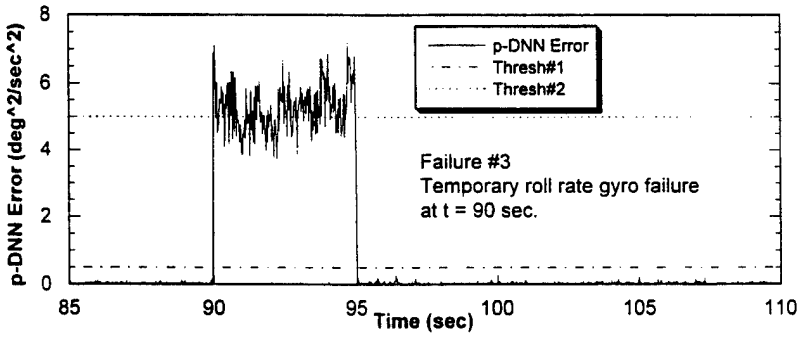


Fig. 21. "p-DNN Error" vs. time for failure type #3 (roll rate gyro failure at $t = 90$ s).

increase as shown in Fig. 20. Furthermore, since the p-DNN error exceeds its threshold #2 in Fig. 21, a "roll rate gyro failure" is declared. Since the sensor failure is of a temporary nature, this SFDIA scheme allows a "rehabilitation" of the failed sensor at some specified time after the failure occurrence by comparing the sensor reading p-failure with the output of p-DNN. If the difference between the two parameters is within a selected tolerance for a specified length of time, then the roll

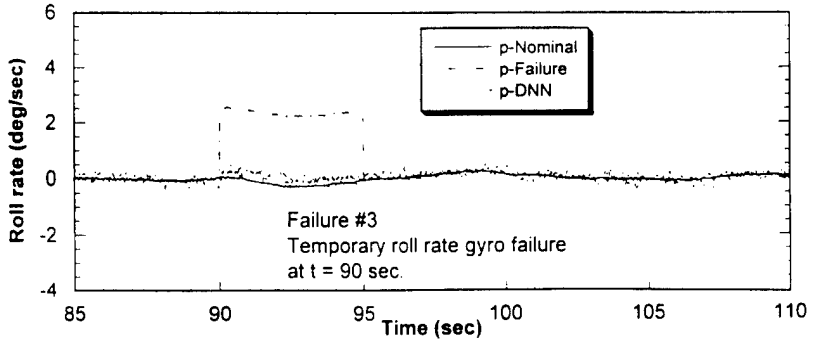


Fig. 22. "p-Failure", "p-Nominal", and "p-DNN" vs. time for failure type #3 (roll rate gyro failure at $t = 90$ s).

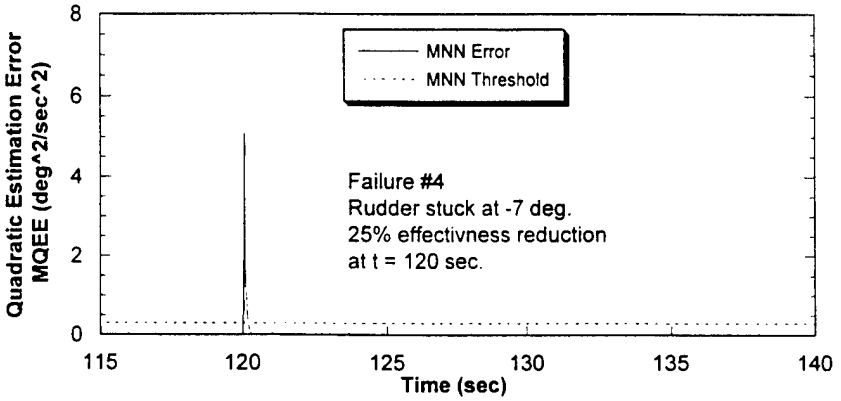


Fig. 23. "MQEE" vs. time for failure type #4 (rudder failure at $t = 120$ s).

rate gyro can be declared operational again, as shown in Fig. 22. Figs. 19 and 22 show instead the successful SFA achieved by replacing the value of the failed p-gyro with its estimate from the corresponding DNN.

Failure #4, the last failure in this simulation, represents a rudder actuator failure. Once again, the "MQEE" parameter exceeds its threshold triggering detection failure for both actuator and sensor in Fig. 23. Once again the r-DNN error exceeds the lower threshold, as shown in Fig. 24; this halts the on-line learning for the r-DNN. However, the suspected "sensor failure" is overruled by the trend of the sum of the absolute values of the coefficients of the auto-correlation function R_{rr} which exceeds its threshold, as shown in Fig. 25. Therefore a "rudder actuator failure" is declared. As in Fig. 17, a decreasing trend can be noticed in Fig. 25 due to the successful AFA achieved by the on-line learning roll and yaw controllers. It should be mentioned that this type of rudder actuator failure was the limit failure for which the aileron-induced yawing moment was able to accommodate the failure-induced yawing moment. It should also be emphasized that the actuator rudder failure is the worst case scenario for commercial aircraft.

The overall effectiveness of the AFA scheme for both elevator and rudder failures is more clear from the trends in Figs. 26 and 27, which show the deflections of the left and right elevators, left and

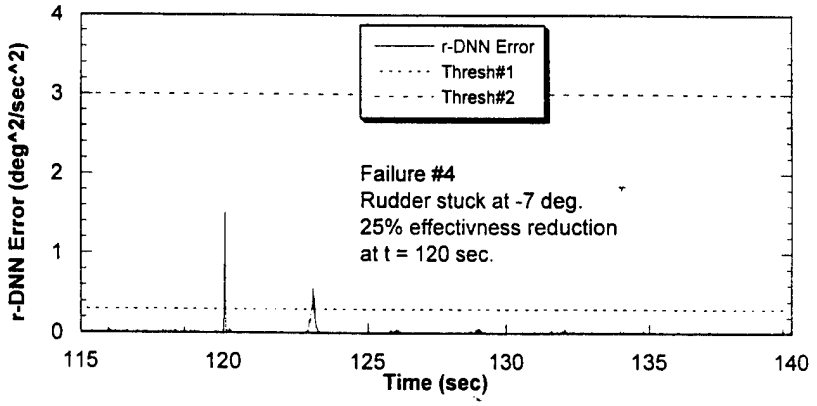


Fig. 24. "r-DNN Error" vs. time for failure type #4 (rudder failure at $t = 120$ s).

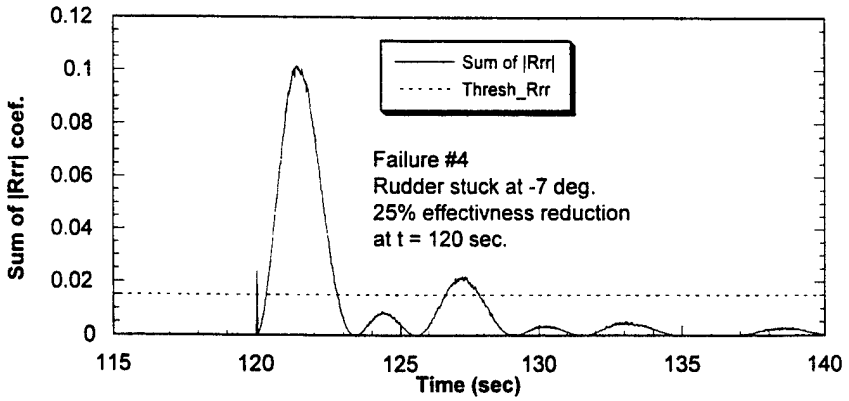


Fig. 25. Sum of the absolute values of the coefficients of R_{rr} vs. time for failure type #4 (rudder failure at $t = 120$ s).

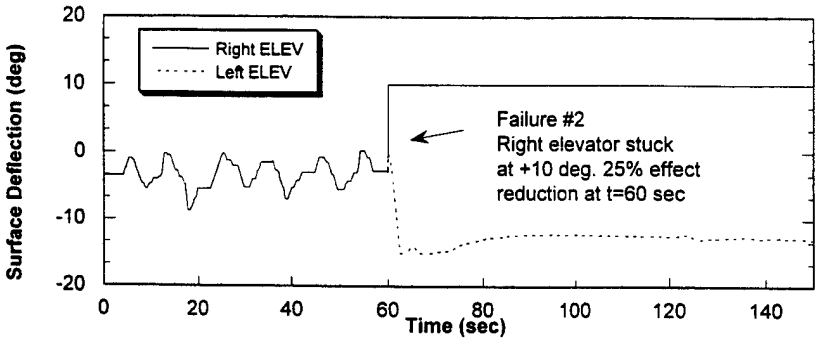


Fig. 26. Right and left elevator deflections vs. time for failure types #1–#4.

right ailerons, and rudder. As stated above, the primary goal following an actuator failure is to regain a trimmed equilibrium condition for the aircraft. For that purpose, following failure #2 the left elevator provides the necessary pitching deflection as calculated by the on-line learning pitch

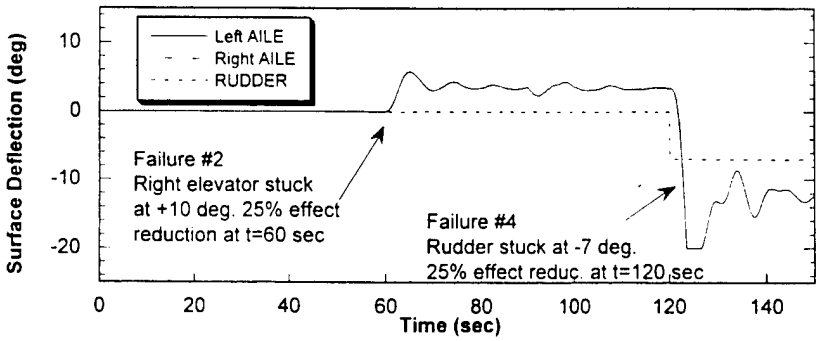


Fig. 27. Right and left aileron and rudder deflections vs. time for failure types #1–#4.

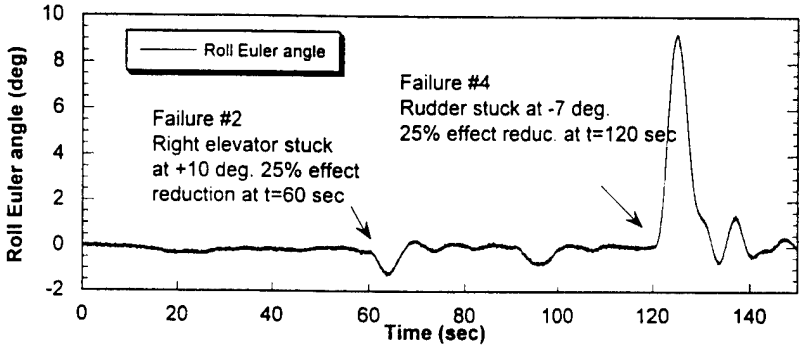


Fig. 28. Roll Euler angle vs. time for failure types #1–#4.

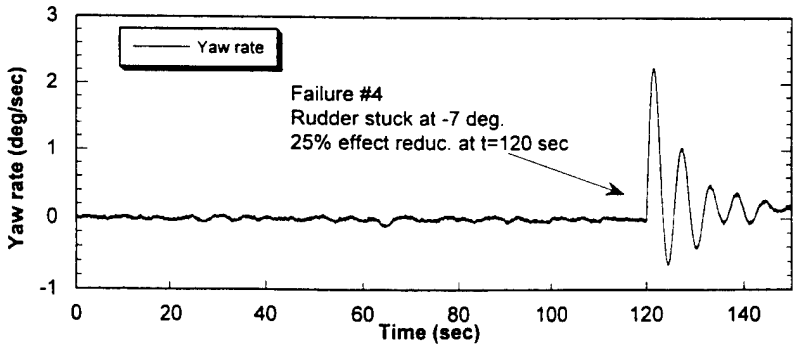


Fig. 29. Yaw rate vs. time for failure types #1–#4.

neural controller, as shown in Fig. 26. Fig. 27 shows instead the compensating aileron deflections (immediately after the positive FDI after $t = 60$ s) canceling the rolling moment induced as a cross-effect by the right elevator failure; also shown after $t = 120$ s is the aileron deflection canceling the yawing moment induced as a cross-effect by the rudder failure.

To complete the dynamic scenario of the simulation Figs. 28 and 29 show the time histories of key aircraft parameters, that is, the roll Euler angle, ϕ, and yaw rate, r, respectively.

5. Conclusions

This paper discusses the performance of a simulated neural network-based fault-tolerant flight control system with capabilities for detecting, identifying, and accommodating sensor and actuator failures. First, step-type and ramp-type transient failures of the rate gyros from a model of a commercial transport aircraft were simulated and discussed by using two different detection criteria. The simulation results indicate that the MQEE-criterion seems to be more suitable for step-type sensor failures while the OQEE-criterion performs better for ramp-type sensor failures. Therefore, a combined detection scheme provides more reliable detection capability for the SFDIA for any type of failures.

Next, a particular logic is introduced allowing the integration of the two schemes with the goal of minimizing the false alarm rate as well as incorrect failure identification. The results confirm the potential offered by on-line learning NNs for both state estimation and control purposes within fault-tolerant systems.

Acknowledgements

Partial support for the first author has been provided through the NASA Ames Grant No. NAG 2-1158. Partial support for the second and third author has been provided through the AFSOR Grant F49620-98-1-0136.

References

[1] Friedland B. Maximum likelihood failure detection of aircraft flight control sensors. AIAA Journal of Guidance, Control, and Dynamics 1982;5(5):498–503.

[2] Baruh H, Choe K. Sensor-failure detection method for flexible structures. AIAA Journal of Guidance, Control, and Dynamics 1987;10(5):474–82.

[3] Napolitano MR, Neppach CD, Casdorph V, Naylor S, Innocenti M, Silvestri G. A neural-network-based scheme for sensor failure detection, identification, and accommodation. AIAA Journal of Guidance, Control, and Dynamics 1995;18(6):1280–6.

[4] Napolitano MR, Younghwan A, Seanor B, Pispistos S, Martinelli D. Application of a neural sensor validation scheme to actual boeing B737 flight data. Proceedings of the '99 AIAA Guidance Navigation & Control Conference, August 1999.

[5] Ha CM, Wei YP, Bessolo JA. Reconfigurable aircraft flight control system via neural networks. Proceedings of the 1992 Aerospace Design Conference, AIAA Paper 92-1075, Irvine, CA, February 1992.

[6] Huang C, Tylock J, Engel S, Whitson J, Eilbert J. Failure-accommodating neural network flight control. Proceedings of the AIAA Guidance, Navigation and Control Conference, AIAA Paper 92-4394, Hilton Head, SC, August 1992.

[7] Napolitano MR, Chen CI, Naylor S. Aircraft failure detection and identification using neural networks. AIAA Journal of Guidance, Control, and Dynamics 1993;16(6):999–1009.

[8] Ho HS, Balakrishnan SN. Fuzzy logic in restructurable flight control systems. Proceedings of the AACC American Control Conference, Part 2, Seattle, Washington, June 1995. p. 1362–6.
[9] Ochi Y, Kanai K. Application of restructurable flight control system to large transport aircraft. AIAA Journal of Guidance, Control, and Dynamics 1995;18(2):365–70.
[10] Napolitano MR, Casdorph V, Neppach C, Naylor S. On-line learning neural architectures and cross-correlation analysis for actuator failure detection and identification. International Journal of Control 1996;63(3):433–55.
[11] Motyka P, Bonnice W, Hall S, Wagner E. The evaluation of failure detection and isolation algorithms for recontstructurable control. NASA Contractor Report 177983, 1985.
[12] Bonnice W, Motyka P, Wagner E, Hall S. Aircraft control surface failure detection and isolation using the OSGLR test. Proceedings of the AIAA Guidance, Navigation and Control Conference, AIAA Paper 86-2028, Williamsburg, VA, August 1986.
[13] Bonnice W, Wagner E, Hall S, Motyka P. The evaluation of the OSGLR algorithm for restructurable controls. NASA Contractor Report 178083, 1986.
[14] Bundick WT. Development of an adaptive failure-detection and identification system for detecting aircraft control-element failures. NASA Technical Paper 3051, 1991.
[15] Wilsky AS. Failure detection in dynamic systems. Agard LS-109, Neuilly sur Seine, France, October 1980. p. 2.1–14.
[16] Kerr TH. False alarm and correct detection probabilities over a time interval for restricted classes of failure detection algorithms. IEEE Transactions of Information Theory 1982;IT-28(4):619–31.
[17] Rumelhart D, McClelland J. Parallel distributed processing. Cambridge, MA: MIT Press, 1986.
[18] Nielsen RH. Neurocomputing. Reading, MA: Addison Wesley Publishing Company, 1990.
[19] Simpson PK. Artificial neural systems: foundations, paradigms, applications, and implementations. New York: Pergamon Press, 1990.
[20] Narendra KS, Partasarathy K. Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1990;1(1).
[21] Hunt KJ, Sbardato D, Zbikowski R, Gawthrop PJ. Neural networks for control systems – a survey. Automatica 1992;28(6):1083–97.
[22] Levin AU, Narendra KS. Control of non-linear dynamical systems using neural networks: controllability and stabilization. IEEE Transactions on Neural Networks 1993;4(2).
[23] Cybenko G. Approximation by superposition of sigmoidal functions. Mathematics of Control Signals and Systems 1989;2(4):303–9.
[24] Polycarpou M. On-line approximators for nonlinear system identification: a unified approach, control and dynamic systems series. Neural Network Systems Techniques and Applications, Vol. 7. New York: Academic Press, 1998.
[25] Polycarpou M, Vemuri AT. Learning methodology for failure detection and accommodation. IEEE Control Systems Magazine 1995;15(3):16–24.
[26] Chen CL, Nutter RS. An extended back-propagation learning by using heterogeneous processing units. Proceedings of International Joint Conference on Neural Networks, Baltimore, Maryland, June 1992.
[27] Alexander C, Cortellessa V, Del Gobbo D, Mili A, Napolitano M. Modeling the fault tolerant capability of a flight control system: an exercise in SCR specifications. to be presented at the LFM 2000, 5th NASA Langley Formal Methods Workshop, to be held on June 13–15, 2000.
[28] Del Gobbo D, Napolitano MR. Issues in fault detectability for dynamic systems. To be presented at the '2000 American Control Conference (ACC) Conference, Chicago, IL, June 2000.
[29] Del Gobbo D, Cukic B, Easterbrook S, Napolitano MR. Fault detectability analysis for requirements validation of fault tolerant systems. Proceedings of the 4th IEEE High-Assurance Systems Engineering (HASE) Symposium, Washington DC, November 1999.
[30] Del Gobbo D, Napolitano MR, Callahan J, Cukic B. Experience in developing system requirements specification for a sensor failure detection and identification scheme. Proceedings of the 3rd IEEE High-Assurance Systems Engineering Symposium, Washington DC, November 1998.
[31] Napolitano MR, et al. Kalman filter and neural network approaches for sensor validation in flight control systems. IEEE Control Systems Technology 1999;6(5):596.
[32] Roskam J. Airplane flight dynamics and automatic flight controls, Part I, design, analysis and research corporation. Kansas: Lawrence, 1995.

Appendix A.3

Authors:

Del Gobbo, D., Napolitano, M.R.

Title:

"Issues in Fault Detectability for Dynamic Systems"

Proceedings of the '2000 American Control Conference (ACC) Conference, Chicago, IL, June 2000

Paper ID:

ACC00-IEEE1261

Issues in Fault Detectability for Dynamic Systems

Diego Del Gobbo and Marcello R. Napolitano

Department of Mechanical and Aerospace Engineering, West Virginia University,

Morgantown, WV 26506-6106, USA

E-mail: *delgobbo@ds5500.cemr.wvu.edu, napolit@cemr.wvu.edu*

Abstract

Research on fault detection has witnessed an increasing interest in recent years. However, little attention has been indicated to certification of such systems. There is a significant need to develop means and methodologies that allow an effective verification of fault detection system capabilities. Certification processes involve specification of system requirements followed by testing of the system against them. Within this process we focus our attention on validation of detection requirements. An accurate analysis of detection feasibility is an important step toward specification of meaningful requirements. This paper presents a detectability condition in the frequency domain for faults in linear systems. The major advantage of this condition is that it is independent of the fault detection approach adopted. An example of detectability analysis on the inverted pendulum system is provided.

1 Introduction

Fault Detection (FD) systems are necessary to add fault tolerant capabilities in plants where safety is a major concern. In the past two decades the technical community has presented a large variety of techniques to address this need. However, little has been done to analyze how to specify this need and how to objectively assess if a given FD algorithm does provide reliable detection. Lack of a certification methodology is a major impediment to further development of FD systems.

Requirements specification plays a key role within this certification framework. It describes expected system functions, performance, and operative constraints, without involving any detail about their implementation. By testing the system against its requirements one can infer whether the system performs as required. In specifying the requirements for a FD system one must specify which plant components can fail, which class of faults are more likely to occur, disturbances and operative conditions of the plant, response time to fault occurrence. Before this information can be accepted as requirements specification, feasibility of required functions and performance must be validated [2]. Is it possible to detect any of the specified faults within the required timeframe under specified disturbance and operating conditions of the plant? To answer this question we need first to define what "detection feasibility" (or *detectability*) means, then we need to provide an analytical condition to assess if a fault is detectable under specified conditions.

Fault detectability is a relatively new concept in the FD field. Only a few papers in the technical literature focus on fault detectability; unfortunately, this concept assumes slightly different meanings in each of them. Definitions of detectability and "*strong fault detectability*" are presented in [1] and [10]. In the latter detectability conditions are

formulated in the context of parity equations while in [7] and [8] detectability is addressed in the context of unknown input Fault-Detection observers. In [9] the limits of fault detectability in systems with model uncertainty are derived in the time domain. In [5] the size of the smallest detectable fault in innovations-based FD schemes with optimum threshold is determined. Finally, in [4] a residual generator is synthesized to meet false alarm rate requirements and minimize the size of detectable faults.

In the above references, fault detectability is formulated with respect to a particular FD approach. Questions may arise, if there are constraints on the capability of a FD scheme that do not depend on the FD algorithm, but on the structure of the system and on the fault dynamics. Moreover, within a certification framework a definition is needed that does not depend on any FD technique.

To address this issue we provide a definition of fault detectability that does not depend on any FD approach. Detectability is analyzed as a system property and conditions for a fault to be detectable are derived in the frequency domain.

In the following section a definition of detectability as system property is formulated and compared to other definitions. Then, fault detectability conditions are derived in the frequency domain for linear systems with structured disturbances. In the last section the results of fault detectability analysis on the sensors of an inverted pendulum are reported.

2 Definition of fault detectability

Assessment of detection feasibility is part of the validation process of the requirements for a FD system. Since requirements specify system functions and

performance without referring to any particular solution, detection feasibility too must be addressed independently from any FD technique. Our only assumption is that the FD system adopts an analytical redundancy approach, that is detection is accomplished by processing available measurements of system inputs and outputs.

Detection feasibility is of course strictly related to the fault dynamics. Slow varying faults, low size faults can be more difficult to detect. Disturbances also play a major role. Because of them, the correlation between system inputs and outputs suffer an inherent uncertainty. Analytical redundancy based FD algorithms depend on this correlation to accomplish their task. Hence, disturbances are a key parameter in fault detectability assessment. The dynamics of the system plays an important role too. Depending on the system dynamics, system outputs can mask or enhance information related to the fault. After these considerations on detectability issues, the following definition can be stated:

> *"A fault on a component of the system is said to be detectable if knowledge of system inputs and outputs over a finite time interval following the occurrence of the fault allows the detection in spite of disturbances".*

With the above definition, detectability of a component fault is a system property that can be analyzed regardless of the FD approach. Detectability only depends on the fault dynamics, on the system dynamics, and on the disturbance distribution.

Definitions of detectability presented in the literature capture some of the characteristics of the above interpretation. In [7] Frank at al. define a system to be "*unknown-input fault detectable*" if for "*almost all faults, an arbitrary small time interval allows a unique decision for the fault only on considering the known input and the available output data in this time interval*". According to this definition fault detectability

is a system property that holds for "almost all faults" while in our definition detectability is related to each single fault. In a system there can be both detectable and undetectable faults at the same time. An example is provided in section 4. Hence, Frank's definition leads to more restrictive detectability conditions.

In [9] Horak states that a fault is detectable if its effects on system outputs overtake the effects of model uncertainties. Using an optimization procedure based on the Maximum Principle, the maximum possible deviation between nominal output values and actual output values ("*reachable measurements interval*") is derived at each time step. A fault is declared detectable if it causes the system output to assume values outside the reachable measurement interval. This definition of fault detectability highlights the concept of detection feasibility in spite of disturbances. However, it differentiates detectable from undetectable faults in terms of fault effects in time domain only. Faults that add to system output a low-amplitude, high-frequency component could be detected by analyzing the frequency content of the measurements. Nevertheless, they are undetectable, according to Horak's definition, if fault effects fall within the reachable measurement interval.

Despite some differences, our definition and those provided in [7] and [9] represent an effort to define fault detectability in terms of system capability to provide fault-related information for a successful detection. In other definitions presented in the literature fault detectability is either intended as detection capability of a particular FD system ([1] and [6]), or as the best detection capability achievable by adopting a certain FD approach ([8] and [10]). None of the above classes of definitions reflect our need of a solution independent definition of fault detectability. However, in the latter, fault detectability can be regarded as a system property given the generality of the FD approach. In [8] the

authors focus on unknown input fault-detection observers, while in [10] the authors focus on parity space based FD schemes.

In [10] a fault is "*detectable if there exists a residual generator such that the transfer function from the fault to the residual is non-zero*". A fault is "*strongly detectable if there exists a residual generator such that the steady state value of the transfer function from the fault to the residual is non-zero*". Both conditions are then expressed within the parity space approach. Simple detectability and strong detectability were introduced in [1] in order to distinguish between residuals where the effects of the fault can disappear even if the fault still exists and residuals where the effects of the fault persist if the fault is present. In [1] the definitions are referred to a particular residual generator, while in [10] detectability and strong detectability are seen as system properties.

3 Fault Detectability condition

Consider the following state-space description of a linear, time invariant system, with structured disturbances and fault inputs:

$$\dot{x}(t) = Ax(t) + Bu(t) + E_x d(t) + F_x f(t) \tag{1}$$

$$y(t) = Cx(t) + E_y d(t) + F_y f(t) \tag{2}$$

$x(t)$, $y(t)$, and $u(t)$ are the state, output, and input vectors respectively; $d(t)$ is the disturbance vector acting on the dynamics and on the measurements; $f(t)$ is the fault vector whose elements represent malfunctions in system components, (either actuators, system elements, or sensors). A, B and C are the state, input, and output matrices respectively; E_x and E_y are the matrices that characterize the disturbance action on system dynamics and measurements respectively; F_x and F_y are the matrices that characterize the

fault action on system dynamics and measurements respectively. The system can either be open or closed loop; in the latter case the input vector is function of the state. To preserve the linearity of the system, only additive faults have been considered. Disturbances are considered to be independent from the state of the system and can represent uncontrollable inputs as well as modeling errors.

In the system described by (1) and (2), detectability of faults does not depend on the state trajectory since neither disturbances nor faults are function of the state. Hence, fault detectability conditions can be formulated considering only the effects of disturbances and faults on the output of the system. A fault $\hat{f}(t)$ on the ith component is detectable if:

$$\exists \left\{ j \in [0, m],\ \Delta\omega = (\omega_1, \omega_2) \right\} / \left\| Y_{ij}(j\omega) \right\|_{\Delta\omega} > \left\| Y_{dj}(j\omega) \right\|_{\Delta\omega} \qquad (3)$$

where m is the number of outputs, $Y_{ij}(j\omega)$ is the Fourier transform of the output on the jth sensor generated by the fault $\hat{f}(t)$ on the ith system component, $Y_{dj}(j\omega)$ is the Fourier transform of the output on the jth sensor generated by the disturbance $d(t)$, $\| \cdot \|_{\Delta\omega}$ denotes the power of the signal in the frequency band $\Delta\omega$:

$$\left\| a(j\omega) \right\|_{\Delta\omega} = \frac{1}{2\pi} \int_{\omega_1}^{\omega_2} \left| a(j\omega) \right|^2 d\omega \qquad (4)$$

Condition (3) can also be stated as follows: "a fault $\hat{f}(t)$ on the ith component is detectable if a frequency band $\Delta\omega$ exists, such that the power in that band of the jth output due to the fault signal is larger than the power due to disturbances".

In order to verify if the above condition is satisfied the following quantities are needed:

- $\frac{Y(j\omega)}{F_i(j\omega)}$ transfer function matrix between the i^{th} element of the fault vector and system output;
- $\frac{Y(j\omega)}{D(j\omega)}$ transfer function matrix between the disturbance vector and system output;
- $\hat{F}(j\omega)$ Fourier transform of the fault signals;
- $|S_d(j\omega)|^2$ power spectral density of disturbances;

Hence, the power spectral density of the quantities in (3) can be computed as:

$$|Y_{ij}(j\omega)|^2 = \left|\frac{Y_j(j\omega)}{F_i(j\omega)}\right|^2 \cdot |\hat{F}(j\omega)|^2 \tag{5}$$

$$|Y_{dj}(j\omega)|^2 = \left|\frac{Y_j(j\omega)}{D(j\omega)}\right|^2 \cdot |S_d(j\omega)|^2 \tag{6}$$

By comparing spectral densities in (5) and (6) over all frequencies and all outputs it is possible to state if a frequency band $\Delta\omega$ and an output $y_j(t)$ exist such that (3) is satisfied.

4 Fault detectability analysis for the inverted pendulum sensors

In this section fault detectability analysis for the sensors of an inverted pendulum is performed. Some results of FD schemes applied to this system can be found in [1], [10], and [3]. The block diagram in Fig. 1 represents a continuous model of the inverted pendulum controlled in closed loop. For simplicity purposes, only disturbances on the system dynamics are considered. The full state is assumed to be available for control purposes, but only cart position and rod angle measurements are considered for fault detectability analysis. Fault dynamics is assumed to be exponential and is described by:

$$f_a(t) = A_a\left(1 - e^{-t/\tau}\right) \tag{7}$$

$$f_p(t) = A_p\left(1 - e^{-t/\tau}\right) \tag{8}$$

$f_p(t)$ and $f_a(t)$ denote faults on position and angle sensors respectively. The following values have been used for the parameters:

$$A_p = 0.1 \text{ (m)}, \quad A_a = 1 \text{ (degree)}, \quad \tau \in [0, 2] \text{ (sec)} \tag{9}$$

Disturbances have been modeled as white noise with power spectral density $|S_d|^2 = 0.001$, filtered by $H_d(s)$. The filter has been selected to allow both low and high frequency disturbances to enter into the system. The question we wish to answer is whether it is possible to detect faults described by (7), (8), and (9) on position and angle sensors under given disturbance conditions.

In order to condcut the detectability analysis the transfer function matrices between fault and output and between disturbance and output need to be computed. The transfer function matrix between fault input and system output is given by:

$$\begin{bmatrix} G_{pp}(s) & G_{ap}(s) \\ G_{pa}(s) & G_{aa}(s) \end{bmatrix} = \left(I + G_o(s)BK_x\right)^{-1} \tag{10}$$

The elements $G_{pp}(s)$, $G_{pa}(s)$, $G_{aa}(s)$, and $G_{ap}(s)$ are the SISO transfer functions between fault inputs and system outputs. The first subscript locates the fault, while the second subscript indicates the output (p, a standing for position and angle sensor respectively).

The transfer function matrix between disturbances and system output is given by:

$$\begin{bmatrix} G_{dp}(s) \\ G_{da}(s) \end{bmatrix} = \left(I + G_o(s)BK_x\right)^{-1} G_o(s)H_d(s) \tag{11}$$

The elements $G_{dp}(s)$ and $G_{da}(s)$ are the SISO transfer functions between disturbance input and system outputs respectively.

By analyzing the system response to faults with different value of τ it is possible to evaluate the capability of the system to provide fault-related information. Figure 2 shows sensor outputs related to a fault on the angle sensor for two different values of the time constant τ. In both cases the steady state values of the effect of the fault are different from zero showing a marked correlation between presence of the fault and faults effect on system outputs. In Figure 3, the outputs related to a fault on the position sensor are shown for the same values of τ. This time the steady state values are zero. Thus, even if the fault is still present, the system does not provide any fault-related information after a sufficient time is elapsed from the occurrence of the fault. Because of its own dynamics the system is not able to provide durable fault-related information for fault on the position sensor. No matter which FD technique is adopted, detection of faults on the position sensor is not possible for some of the faults described by (8) and (9).

In Figure 4 the magnitude of the transfer functions between fault input and system outputs are shown. Both $|G_{pp}(s)|$ and $|G_{pa}(s)|$ decrease substantially at low frequencies. This feature is related to the lack of strong detectability for fault on the position sensor; fault effects are visible only during the transient. At high frequency $|G_{pp}(s)|$ is quite large; hence, faults with a large content in high frequency are likely to be detectable. $|G_{ap}(s)|$ and $|G_{aa}(s)|$ are both non-zero at steady state conditions; hence, faults should be easily detected if their low frequency content is high.

Figure 5 shows the power spectral density of the sensors output for a fault on the angle sensor, along with the power spectral density of the output component due to disturbances. Referring to the detectability condition stated in the previous section, the

fault is detectable if a frequency interval exists where either condition (12) or (13) is satisfied:

$$\left| Y_{ap}(j\omega) \right|^2 > \left| Y_{dp}(j\omega) \right|^2 \qquad (12)$$

$$\left| Y_{aa}(j\omega) \right|^2 > \left| Y_{da}(j\omega) \right|^2 \qquad (13)$$

From the power spectral density diagrams, the class of fault considered reveals to be detectable for any value $\tau \in [0, 2]$. In fact, at frequencies below 0.02 rad/sec both of the above conditions are satisfied.

In Figure 6 similar quantities are shown for faults on the position sensor. For a fault with time constant $\tau = 0$ the detectability condition is satisfied since:

$$\left| Y_{pp}(j\omega) \right|^2 > \left| Y_{dp}(j\omega) \right|^2 \qquad \forall\omega > 7\ rad\ /\ \sec \qquad (14)$$

On the other hand, a fault with time constant $\tau = 2$ reveals to be undetectable, since:

$$\left| Y_{pp}(j\omega) \right|^2 < \left| Y_{dp}(j\omega) \right|^2 \qquad \forall\omega \qquad (15)$$

$$\left| Y_{pa}(j\omega) \right|^2 < \left| Y_{da}(j\omega) \right|^2 \qquad \forall\omega \qquad (16)$$

Results of the detectability analysis can be summarized as follows:

- faults on the angle sensor with dynamics described by (7) are detectable, regardless of the value of the time constant τ. Fault effects overtake disturbances at low frequencies on both system outputs.
- detectability of faults on the position sensor whit dynamics described by (8) depends on the value of the time constant τ. For low values (abrupt faults) effects of the fault overtake disturbances on the position sensor output at frequencies larger than 7rad/sec. For large values of τ (slow varying faults) the fault is not detectable.

These results state that detection requirements of faults specified by (8) and (9) are not feasible. If such faults are to be detected, then an approach other than analytical redundancy must be adopted.

Conclusions

Objective certification tools are critical for the future development of fault detection systems. Within this framework the paper illustrates the importance of detectability analysis to allow specifying meaningful requirements for such systems. Detectability conditions for additive faults on linear systems have been formulated in the frequency domain. These conditions are independent of the fault detection technique adopted, allowing a solution-independent specification of detection requirements.

The importance of detectability analysis has been shown by analyzing detectability properties of faults on two sensors of an inverted pendulum system. The analysis revealed insufficient capability of the system to provide fault-related information for faults on the position sensor. Furthermore, it highlighted frequency ranges where system outputs provide fault-related information, thus providing valuable information for an enhanced design of the fault detection system.

Acknowledgement

Partial support for the first author has been provided by DEPSCOR/AFOSR grant F49620-98-1-0136. Partial support for the second author has been provided by the Institute for Software Research (ISR) under grant NAG4-163.

References

[1] Jie Chen and R.J. Patton. A re-examination of fault detectability and isolability in linear dynamic systems. Fault Detection, Supervision and Safety for Technical Processes, pp. 567-73, Espoo, Finland,1994. IFAC.

[2] D. Del Gobbo, B. Cukic, S. Easterbrook, and M. Napolitano, Fault Detectability Analysis for Requirements Validation of Fault Tolerant Systems. To be published on Proceedings of 4th IEEE International High Assurance Systems Engineering Symposium, 1999.

[3] D. Del Gobbo, Sensor Failure Detection and Identification using Extended Kalman Filtering. Master thesis; 1998.

[4] X. Ding, P.M. Frank and L. Guo. An approach to residual generator and evaluator design and synthesis. IFAC 12th Triennial World Congress, pp.383-86, 1993.

[5] A. Emami Naeini, M.M. Akhter and S.M. Rock. Effect of model uncertainty on failure detection: the threshold selector. IEEE Transaction on Automatic Control, vol. 33, no. 12; pp.1106-15, 1988.

[6] Y.E. Faitakis, S. Thapliyal and J.C. Kantor. An LMI approach to the evaluation of alarm thresholds. International Journal on Robust Nonlinear Control, vol. 8; pp. 659-67, 1998.

[7] P.M. Frank and B. Koppen. Review of optimal solutions to the robustness problem in observer-based fault detection. Journal of Systems and Control Engineering, vol. 207, no. 12; pp.105-12, 1993.

[8] P.M. Frank and J. Wunnenberg. Robust fault diagnosis using unknown input observer schemes. In (11), pp.47-98.

[9] D.T. Horak. Failure detection in dynamic systems with modeling errors. Journal of Guidance, Control, and Dynamics, vol. 11, no. 6; pp. 508-16, 1988.

[10] M. Nyberg and L. Nielsen. Parity functions as universal residual generators and tool for fault detectability analysis. 36th IEEE Conference on Decision and Control, vol. 5; pp.4483-9.

[11] R.J. Patton, P.M. Frank and R.N. Clark. Fault diagnosis in dynamic systems: theory and applications, 1989 (Prentice Hall, Englewood Cliffs, NJ).

FIGURES

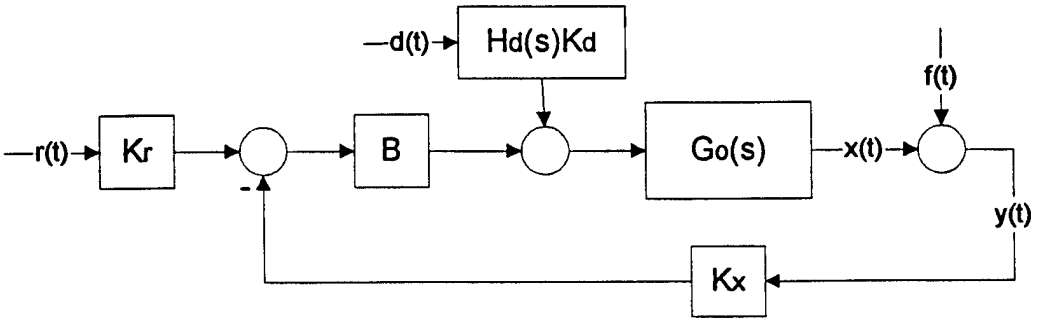


Figure 1. Block diagram of the inverted pendulum.

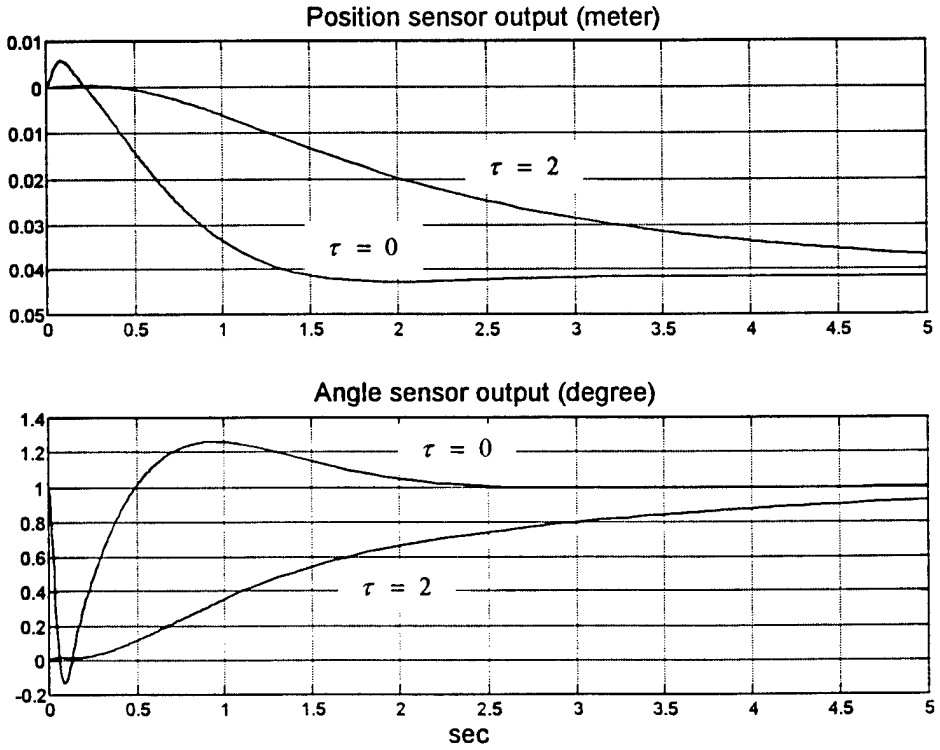


Figure 2. Sensor outputs after a fault on the angle sensor

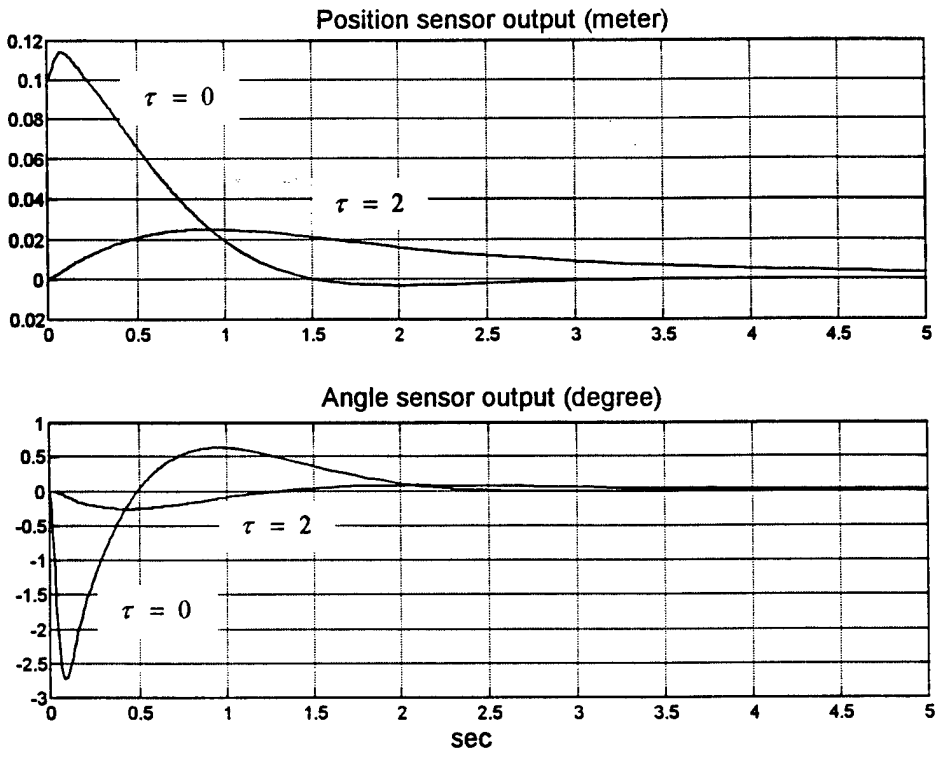


Figure 3. Sensor outputs after a fault on the position sensor

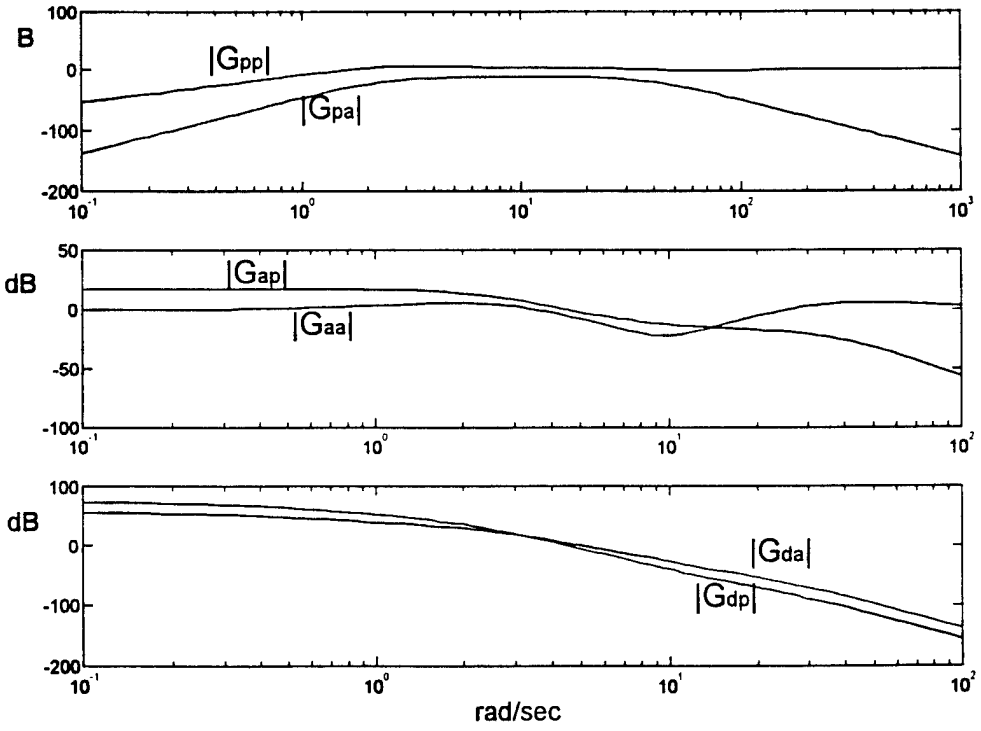


Figure 4. Magnitude of transfer functions

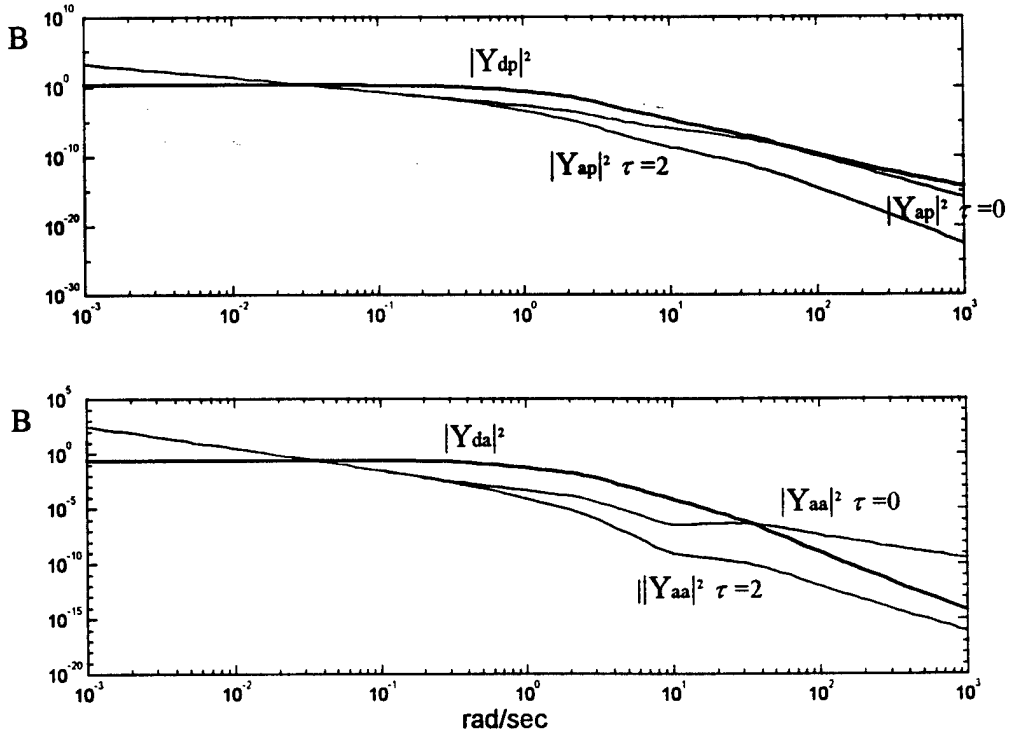


Figure 5. Power spectral density of sensor outputs for a fault on angle sensor

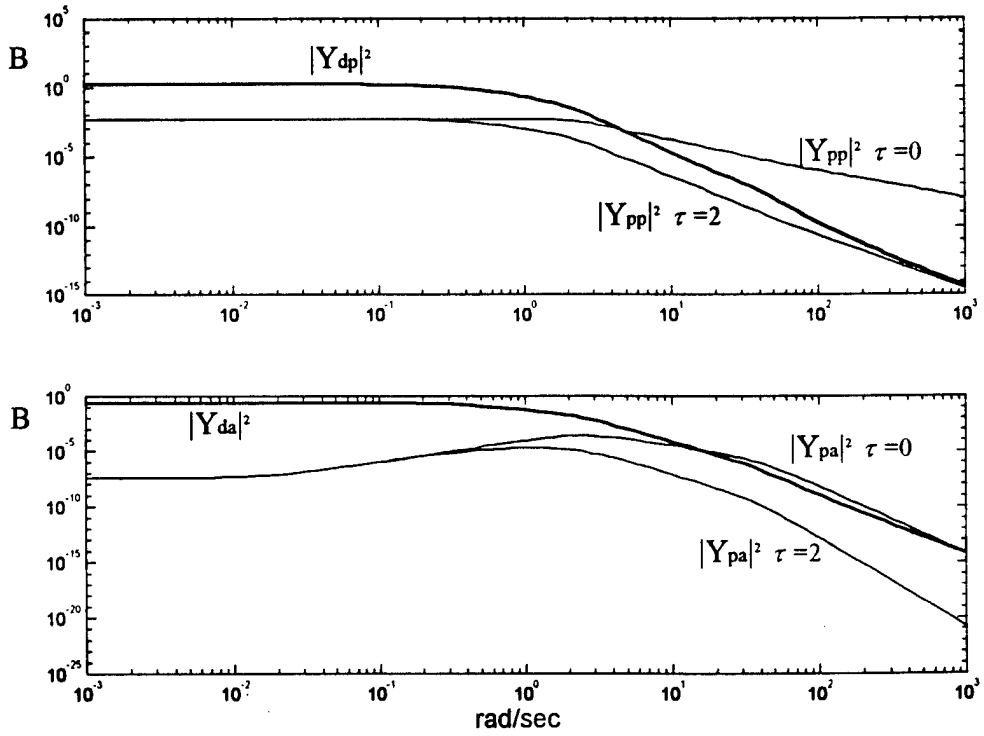


Figure 6. Power spectral density of sensor outputs for a fault on position sensor

Appendix A.4

Authors:

Del Gobbo, D., Cukic, B., Easterbrook, S., Napolitano, M.R.

Title:

"Fault Detectability Analysis for Requirements Validation of Fault Tolerant Systems"

Proceedings of the 4th IEEE High-Assurance Systems Engineering (HASE) Symposium, Washington DC, November 1999

Fault Detectability Analysis for Requirements Validation of Fault Tolerant Systems

Diego Del Gobbo*, Bojan Cukic†, Marcello R. Napolitano*

S. Easterbrook

*Department of Mechanical and Aerospace Engineering
†Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV 26506-6106
delgobbo@ds5500.cemr.wvu.edu, cukic@csee.wvu.edu,
napolit@cemr.wvu.edu

Department of Computer Science
University of Toronto
6, King's College Rd.,
Toronto, Ontario M5S 3H5
sme@cs.toronto.edu

Abstract

When high assurance applications are concerned, life cycle process control has witnessed steady improvement over the past two decades. As a consequence, the number of software defects introduced in the later phases of the life cycle, such as detailed design and coding, is decreasing. The majority of the remaining defects originate in the early phases of the life cycle. This is understandable, since the early phases deal with the translation from informal requirements into a formalism that will be used by developers. Since the step from informal to formal notation is inevitable, verification and validation of the requirements continue to be the research focus. Discovering potential problems as early as possible provides the potential for significant reduction in development time and cost.

In this paper, the focus is on a specific aspect of requirements validation for dynamic fault tolerant control systems: the feasibility assessment of the fault detection task. An analytical formulation of the fault detectability condition is presented. This formulation is applicable to any system whose dynamics can be approximated by a linear model. The fault detectability condition can be used for objective validation of fault detection requirements. In a case study, we analyze an inverted pendulum system and demonstrate that "reasonable" requirements for a fault detection system can be infeasible when validated against the fault detectability condition.

1. Introduction

Fault identification has been recognized as an important mechanism for improving system safety. For example, Failure Modes and Effects Analysis (FMEA) [17] considers the failure of any component within a system and tracks its effects to determine the ultimate consequences. Similarly, Event Tree Analysis (ETA) takes as its starting points the events that can affect the system and tracks them forward to determine their potential consequences. Fault Tree Analysis (FTA) differs from ETA in that it tackles the problem in the reverse direction. It starts with the identification of system hazards and works backwards to determine their possible causes.

FMEA, ETA, and FTA provide good frameworks for fault identification. Based on their results, system designers employ fault tolerant mechanisms to minimize the probability that, during system operation, identified potential faults propagate and cause the system to fail. However, fault identification techniques do not attempt to study *fault detectability*. In other words, these techniques may point to a potential problem and its consequences, but do not provide any assurance in the ability of the system to detect the problem if it appears during system operation.

In this paper, we focus on the specific requirements validation technique applicable in the analysis of dynamic fault tolerant control systems, such as nuclear and chemical plants, aircraft, and spacecraft. Fault tolerance is required in these systems for obvious safety, mission criticality and cost issues and it is often achieved by physical redundancy of the critical components. A Fault Detection (FD) scheme monitors these critical components and, as a fault is detected, recovers the system operation by switching from the faulty unit to one of the (non-faulty) backup units. Physical redundancy is a straightforward approach to fault tolerance. However, it presents some drawbacks, such as additional cost, power consumption, weight (important design parameters in aircraft and spacecraft) as well as the introduction of additional (but not essential) complexity.

To overcome these limitations analytical redundancy ([5], [6], and [14]) has been considered as an alternative approach to physical redundancy in the technical literature. This approach is based on the fact that system inputs and outputs are functionally related variables that, when properly processed, allow the detection of faults in the system. High performance, reduced costs and reduced weight for these schemes make them desirable for practical applications. A variety of solutions has been considered in the technical literature and many issues related to their application have been addressed. Nevertheless, the practical implementation of analytical redundancy based fault tolerance schemes is still very limited. Fault tolerance algorithms are typically based on Kalman filtering, neural networks, and fuzzy logic techniques. The analytical complexity of these schemes can be problematic, especially when compared to the straightforward simplicity of physical redundancy. The skepticism with which analytical redundancy is sometimes received cannot be fully removed until a trustful verification procedure to certify a given fault detection scheme's performance becomes available.

Testing procedures for analytical redundancy schemes are often informal and inadequate. A typical testing environment involves the simulation of the system dynamics, along with the FD scheme. Faults are then injected into the simulated system and the FD scheme performance is assessed based on its promptness in detecting faults, its sensitivity to incipient faults (small size, slow varying faults), its completeness, as well as its false alarm rates. These analyses provide an overall measure of the scheme's capabilities, but this is still far from a formal verification of its functionality and performance. A formal procedure needs to involve system requirements specification, as well as verification and validation techniques spanning throughout the development life cycle. Requirements specification plays a fundamental role in the verification process since verification is meaningless without requirements.

In this paper we focus on the specific validation technique for the requirements of fault tolerant dynamic systems, with particular attention on the validation of feasibility of the requirements. Formal methods contribute significantly in validating consistency and non-ambiguity of requirements; however, there is no guarantee that a set of well-formed consistent requirements fully captures the desired behavior of the system and its interaction with the environment. The evaluation of the matching between "actual" functions needed within the environment and the specified requirements is often left to a subjective judgement. In this paper, we demonstrate that the validation of the requirements feasibility can be conducted in a more objective and formal framework.

For fault tolerant systems, the validation of the requirements needs to include the assessment of feasibility of the fault detection task. The degree of "visibility" and, consequently, the detectability of a fault in the system depends on the fault magnitude, on the signal-to-noise ratio related to the fault, as well as on the system dynamics.

The National Transportation Safety Board (NTSB) database lists several accidents where the fault detectability problem appears to be the critical issue. For example, [9] is a report related to an accident that occurred in 1983, involving a McDonnell DC-10-30 airplane. The aircraft experienced the separation of 50 inches of the right flap; however, "the incident was not noticed until a local resident called to report the fallen article". Detection of such a fault would be desirable to prevent a progressive deterioration of the surface with potentially catastrophic consequences. However, requiring to detect such a fault may be unrealistic if the detection task is unfeasible for circumstances related to the operational environment.

The purpose of the paper is to illustrate the scope of the fault detectability problem and to present a condition to assess fault detectability for analytical redundancy based FD schemes. The proposed condition is formulated in the frequency domain and is applicable to any system whose dynamics is well approximated by a linear model. Because of its analytical nature, it is a formal and objective tool for feasibility validation of system requirements for fault tolerant dynamic systems.

In Section 2, a brief overview of the validation process is provided to better define the feasibility validation task within the requirements specification process. Section 3 illustrates the fault detectability problem and provides the detectability condition. Then, in Section 4, an application of the detectability condition to the requirements of a simple system is presented to show how "reasonable" requirements can turn out to be unfeasible. We conclude the paper with a summary, in Section 5.

2. Validation of requirements

Specification of requirements involves three main steps: formulation of a high-level model of the system, definition of the requirements, and validation of the requirements. The first phase aims to capture the most relevant functions of the system to build and its interactions with the environment. Defining a conceptual

model of the system requires a thorough understanding of the environment in which the system will operate and of the functions that the system will provide [7]. Definition of the requirements involves the description of the services that the system should provide, the constraints under which it must operate and the desired performance attributes. Validation aims to assess if the requirements are complete, unambiguous, consistent, and realistic.

For high assurance systems, requirements validation is a particular concern [16]. The terms "Verification" and "Validation" are commonly used in software engineering to mean two different types of analysis. Validation is concerned with checking that the system will meet the customer's actual needs, while verification is concerned with whether the system is well-engineered. The distinction is clearer if one considers the role of a specification. Validation is the process of checking whether the specification captures the customer's needs, while verification is the process of checking that the software meets the specification.

Validation of requirements is problematic because it usually involves a subjective judgment of how well the system under consideration addresses a real-world need. In contrast, verification should be a relatively objective process, in that if the various products and documents are expressed precisely enough, no subjective judgments should be needed [4]. The use of formal methods for software specification is based on this observation. However, formal methods provide little help with requirements validation, because there is no guarantee that a set of well-formed, consistent requirements actually captures the desired behavior of the system and its interaction with the environment.

The current state of practice in requirements validation relies on a wide variety of validation techniques, ranging from informal and formal inspection processes, to detailed prototyping of user interfaces, and the use of simulations [8]. Ideally, the requirements described in the specification should be realistic, complete, consistent, unambiguous, and testable. Sommerville [15] lists four tasks in the requirements validation process:

1. Check if required functions satisfy user's needs,
2. Check if requirements include all functions and related constraints,
3. Check if requirements are unambiguous and consistent (non-conflicting),
4. Check if functions and constraints specified by the requirements are realistic.

It should be clear from this list of tasks that requirements validation is a systems engineering activity. A system level understanding of the purpose of the system, its environment, and the available technology is crucial. Prototyping and simulations can be used to assist with the first two tasks, while formal analysis can assist with the third. However, the last task, that of determining whether the requirements are feasible, has been given limited attention in the requirements engineering community.

An analysis of requirements feasibility is especially important for requirements that relate to safety and reliability. If it cannot be determined ahead of time whether the safety requirements for a system can feasibly be met, a number of problems may result. Firstly, time and effort can be wasted trying to meet the requirements. Worse, a significant change to safety requirements late in the lifecycle can be disastrous. Such a change will generally mean a different approach to safety and assurance, impacting a large number of design decisions that have already been made.

3. Validation of FD systems requirements: Fault detectability analysis

In this section an approach to feasibility validation of requirements for FD schemes is presented. The FD schemes considered are based on analytical redundancy. In order to formulate a detectability condition, we need to understand the analytical redundancy approach to fault tolerance and the dynamics of how the detectability problem arises.

Let us consider an actuator fault on an aircraft. Modern aircraft have several control surfaces, which are controlled either by the pilot or by the flight control computer. The "actuating chain" consists of an aerodynamic surface, an actuating unit, and linkages along which a control command is sent from the cockpit or the on-board computer. All of the components of the actuating chain can be duplicated except the aerodynamic surface. Therefore, if one of the actuating units fails, it can be replaced with its backup. But if a problem occurs with the aerodynamic surface, neither detection of the fault, nor its accommodation is possible. Aircraft accidents involving separation of aerodynamic surfaces during flight are not unusual. [10], [11], and [12] are the reports of three different aircraft that experienced partial separation of one of the trailing edge flaps. In all three cases the flight conditions were nominal until the descent for approach and landing. After deflection of the flaps the aircraft experienced rolling; the induced rolling moment had to be compensated through ailerons deflections by the pilot for the aircraft to land safely.

These examples not only illustrate the limitation of physical redundancy, but also show a possible different solution to detection and accommodation problems.

The separation of the flap caused a rolling moment which was not consistent with the aircraft inputs at that particular instant of time. A clear understanding of the aircraft dynamics and availability of roll angle and aileron deflection is sufficient to detect the fault. This is the basis of the analytical redundancy approach to the FD problem. Furthermore, the aircraft control was regained through aileron input by the pilot, implying that physical redundancy is not needed when the fault does not compromise the controllability property of the system.

To understand how the detectability problem may arise let us consider two interesting accident reports by the NTSB.

[12] states that on March 1997, 18 feet of the right outboard flap separated from a Boeing 767-232 during the airplane's approach to the Dallas/Fort Worth International Airport (DFW), Texas. The captain reported that "take-off and departures were routine, as were all aspects of the enroute phase of flight". According to the Digital Flight Data Recorder, after 1 minute and 54 seconds since deflection of the flaps the aircraft started rolling to the right. By using a significant amount of left aileron the first officer regained control and safely landed.

[9] reports that on September 1983, a McDonnell DC-10-30 experienced the separation of 50 inches of the right flap. The report follows saying that "the incident was not noticed until a local resident called to report the fallen article".

In the first case, the magnitude of the fault was such that the pilot clearly felt the separation of the surface. In the second accident the effects of the fault were negligible with respect to normal disturbances, and the problem passed undetected. Detectability of the fault clearly depends on the relationship between fault and disturbances effect on the output of the system. More specifically it depends on their ratio, so that what is actually relevant in the detectability problem is the signal-to-noise (S/N) ratio of the fault. Another key element that is brought to light by the first accident is the relationship between fault detectability and the state of the system (in this case the maneuver of the aircraft). It is unlikely that 18 feet of the flap surface suddenly detach from the wing without any warning. It is plausible that the detachment of the flap, or more specifically, the fracture of the bolts which fastened the surface to the carriage support beam (as reported in [12]) slowly propagated during flight and erupted during approach when the flaps are fully loaded. Unfortunately, the effects of an on-going fault on a flap are not visible in straight-and-level flight, while they become visible, and potentially dangerous, during approach, when the flaps are deflected and the missed surface induces a rolling moment on the aircraft.

So far we have considered the actual effects of the fault on the system. However, analytical redundancy based FD schemes use commanded inputs and measured outputs, which may not reflect the actual response of the system. If a fault occurs on one of the sensors, the actual output of the system will be different from the sensor output. This implies that the effects of the fault could be masked in the measured outputs even when they are evident in the actual output of the system. The key elements in this "game" of actual vs. measured outputs are the system and fault dynamics.

Following the previous discussion, the concepts summarized below are identified as the critical concepts in the fault detectability problem.

Detectability is strictly related to the fault dynamics: slow varying faults and low magnitude faults can be more difficult to detect. Disturbances also play a major role. Because of them, the functional relationships between system inputs and outputs suffer an inherent uncertainty. Analytical redundancy based FD algorithms depend on these functional relationships to accomplish their task; hence, disturbances are a key parameter in fault detectability assessment. The dynamics of the system plays an important role as well. Depending on the system dynamics, system outputs can mask or enhance information related to the fault. Hence, the elements involved in detectability assessment are the fault dynamics, the system dynamics, and the disturbance power spectrum.

The above examples and following argumentation provide an intuitive understanding of the detectability problem. The next step is to provide an objective definition of the problem and an analysis tool that can be used in the requirements validation process.

To specify meaningful requirements, detection feasibility of the fault set of interest needs to be assessed. In other words, the feasibility of the functions and performance constraints required from the FD scheme needs to be validated. Only a few papers in the technical literature have focused on fault detectability and a definition unanimously accepted by the technical community is still missing. In [3] and [13] different definitions of detectability are used, each referring to a specific FD approach. However, the definition of detectability is useful in the process of requirements

validation only if it is general, i.e., not specific to the particular FD technique employed. The FD scheme requirements, in fact, specify the desired functions of the scheme without describing the details about their implementation. The definition of detectability condition as a system property that does not depend on the adopted FD scheme has been recently proposed in [2]. This condition can be summarized as follows:

"A fault on the i^{th} component is detectable if a frequency band $\Delta\omega$ exists, such that the power in that band of the j^{th} output, due to the fault signal, is larger than the power due to disturbances".

In analytical terms, a fault on the i^{th} component is detectable if:

$$\exists\{j \in [0,m], \Delta\omega = (\omega_1, \omega_2)\}/ \quad \left\|Y_{ij}(j\omega)\right\|_{\Delta\omega} > \left\|Y_{dj}(j\omega)\right\|_{\Delta\omega} \tag{1}$$

where m is the number of outputs, $Y_{ij}(j\omega)$ is the Fourier transform of the output on the j^{th} sensor generated by a fault on the i^{th} system component, $Y_{dj}(j\omega)$ is the Fourier transform of the output on the j^{th} sensor generated by the disturbance, and $\|\cdot\|_{\Delta\omega}$ denotes the power of the signal in the frequency band $\Delta\omega$.

Eq. (1) is the condition needed to validate detection capabilities required from an FD system. Its application to any linear (or linearized) system is straightforward since the power spectral densities related to faults and disturbances can be easily computed separately and then compared. By comparing spectral densities over all frequencies and all outputs, it is possible to state whether a frequency band $\Delta\omega$ and an output $y_j(t)$ exist such that condition (1) is satisfied. Hence, it is possible to assess the detectability characteristics of the fault.

The system linearity constraint imposed to the application of Eq(1) is not as constrictive as it may appear. The dynamics of a system like an aircraft is described by a set of non-linear differential equations resulting form the application of basic physic laws. However, under certain flight conditions the aircraft dynamics can be approximated by a linear system. In fact, most of the design of flight control laws is based on this linear model. Use of linearized models is common practice in system engineering and it allows application of condition (1).

In the following section a practical example is provided where the linearized model of a non-linear system is used to validate the detection requirements for an analytical redundant FD scheme.

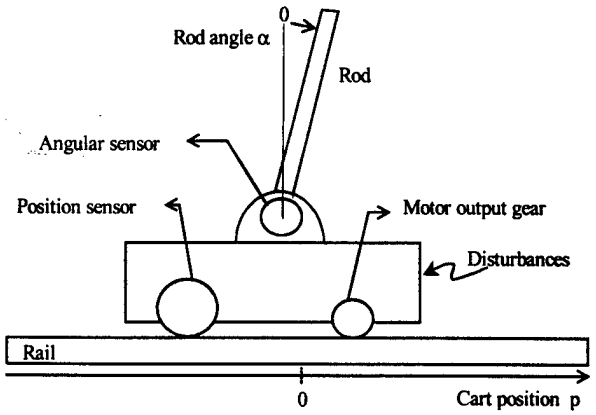


Figure 1. Inverted pendulum system.

4. Case study

Consider the inverted pendulum system in Figure 1. This system consists of a cart and a rod. The cart slides on a 1-meter length rail driven by a motor, while the rod is free to rotate around an axis perpendicular to the direction of motion of the cart. The system is equipped with sensors that measure cart position and rod angle. These variables are used within a feedback loop to stabilize the system around the unstable equilibrium condition with the rod balanced in the vertical position. The system is affected by disturbances whose statistical properties and power spectra are known.

A simplified block diagram of the system is shown in Figure 2 [1, 2]. $G_o(s)$ represents the non-linear dynamics of the inverted pendulum, K_r and K_x implement the control law, $H_d(s)$ describes the effects of disturbances within the system, $r(t)$ is the reference signal for the position of the cart, $d(t)$ is the disturbance input, $f(t)$ is the fault input used to represent the occurrence of a fault on the system outputs, $x(t)$ is the system output (actual output), $y(t)$ is the sensor's output (measured output). Since $y(t)$ represents the measurement of the system output, $y(t)$ and $x(t)$ will assume the same values (within the limit of the sensor accuracy) unless a fault occurs on one of the sensors. The reference input to the system is a square wave, so that the cart slides periodically from one end of the rail to the other, keeping the rod balanced within the capability of the control law and in spite of the disturbance action.

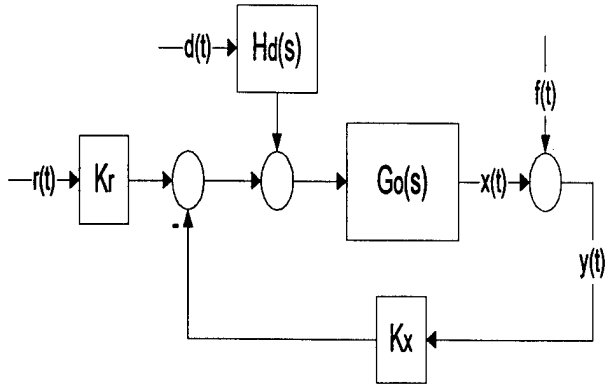


Figure 2. Block diagram of the inverted pendulum system.

A fault on the cart position sensor may cause the system to go beyond the rail ends, with possible damages to the system. Thus, an FD scheme is needed. Assume that the dynamics of the fault is exponential with a given size and time constant within a known interval as described by:

$$f(t) = A\left(1 - e^{-t/\tau}\right)$$
$$A = 0.1\ (m),\ \tau \in [0,2]\ (sec) \qquad (2)$$

The following constraint could be the core of the requirements specification for the FD system:

" the FD system shall detect any fault belonging to the class described by Eq. (2) "

From an engineering point of view the above constraint seems reasonable. The fault class described by Eq. (2) is realistic and a fault-size of *0.1 m* is considerable when compared to the rail length of *1 m*, so that the effects of the fault are visible on the system. The development of the requirements for such a system is beyond the scope of this paper; nevertheless, it is reasonable to assume that it is possible to specify a complete set of requirements including the above constraint. By using a formal language such requirements could be expressed without ambiguity and inconsistencies. What is left to be addressed is the feasibility validation of the requirements.

In Figure 3, the reference input, the actual position of the cart, and the output of the position sensor, are displayed against time for two different faults. The faults differ for the value of the time constant. In Figures 3a and 3b, the faults with time constants $\tau = 0$ and $\tau = 2$ are

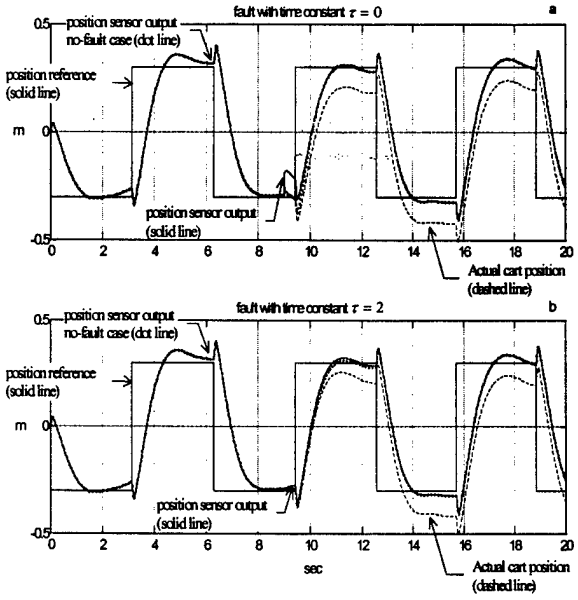


Figure 3. Fault effects on position sensor output.

considered, respectively. Both faults occur at $t = 9$ sec. The output of the position sensor in absence of fault (no-fault case) is also displayed on both graphs. Recall that, in the absence of a fault, actual cart position and position sensor output coincide.

By comparing the actual cart position with the output of the position sensor in the no-fault case, it is clear that the fault effects are visible for both faults. After the occurrence of the fault, the cart shifts downward, touching the end of the rail at $t = 15.7$ sec. Unfortunately, only measurements of system outputs are available for the FD scheme, while the actual position of the cart is not. In Figure 3a, the effects of the fault are visible on the position sensor output. However, after a short impulse at $t = 9$ sec, the effects of the fault rapidly disappear and, for $t > 10$ sec., the position sensor output in presence or absence of the fault almost coincide. Nevertheless, the fault is still present as the actual position of the cart demonstrates. Things deteriorate in Figure 3b, where the output of the position sensor does not show any effect of the fault.

From the above analysis it can be concluded that the system has a remarkable weakness in providing fault-related information for slow varying faults on the position sensor. To quantify this weakness, fault detectability analysis will be performed.

Figure 4 shows the spectral density of the sensors output for a fault on the position sensor, along with the power spectral density of the output component due to disturbances, for two different values of the time constant. All spectral densities have been computed using a linearized model of the non-linear dynamics $G_o(s)$. According to the detectability condition stated in Eq. (1), the fault is detectable if a frequency interval exists, where either the condition (3) or (4) is satisfied:

$$\left|Y_{pp}(j\omega)\right|^2 > \left|Y_{dp}(j\omega)\right|^2 \tag{3}$$

$$\left|Y_{pa}(j\omega)\right|^2 > \left|Y_{da}(j\omega)\right|^2 \tag{4}$$

From the power spectral density diagrams, it is clear that, for a value $\tau = 0$ of the time constant, the detectability condition is satisfied since:

$$\left|Y_{pp}(j\omega)\right|^2 > \left|Y_{dp}(j\omega)\right|^2 \quad \forall\omega > 7\,\mathrm{rad/sec} \tag{5}$$

On the other hand, a fault with time constant $\tau = 2$ reveals to be undetectable, since:

$$\left|Y_{pp}(j\omega)\right|^2 < \left|Y_{dp}(j\omega)\right|^2 \forall\omega \tag{6}$$

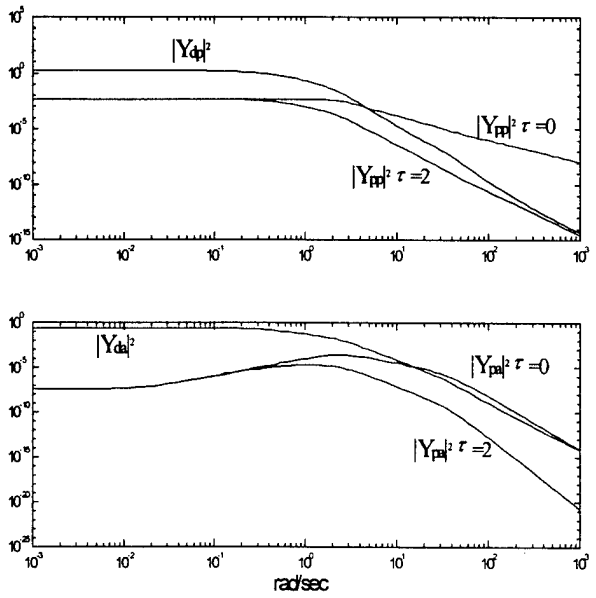


Figure 4. Power spectral density of sensor outputs for a fault on position sensor.

$$\left|Y_{pa}(j\omega)\right|^2 < \left|Y_{da}(j\omega)\right|^2 \forall\omega \tag{7}$$

Any fault with time constant $\tau \geq 2$ is masked by the effects of disturbances. Therefore, the detectability analysis has proved that the constraint stated at the beginning of this section is not realistic, and any set of requirements including that constraint is meaningless, regardless of its completeness and consistency properties.

5. Conclusion

System requirements specification and their validation play a key role in the development and the analysis of high insurance applications. Unfortunately, the extent to which system requirements meet the correctness, completeness and feasibility criteria is often left to the subjective judgement of the system analyst.

The assessment of requirements feasibility is a problem whose solution largely depends on the application field. We have focused on this problem within the field of analytical redundancy based FD schemes for dynamic systems. This study provides an analytical condition that allows assessing the feasibility of the fault detection task in systems whose dynamics can be modeled by a set of differential equations. The detectability condition captures the three key elements in assessment of fault detection feasibility: the dynamics of the fault, the impact of system disturbances on the functional relationship between system inputs and outputs, and the dynamics of the system.

The detectability condition does not directly provide a basis for the implementation of a FD system. However, it highlights the strengths and weaknesses of the system in propagating the effects of the fault throughout its output as a function of frequency. Hence, it provides information on which frequency regions are more suitable for the detection of a fault. Moreover, the detectability condition clearly states when a system is incapable of detecting fault propagation effects so that an approach other than analytical redundancy can be adopted for fault detection purposes.

Fault detectability analysis is an important step in requirements validation of FD systems. The suggested fault detectability condition provides an objective assessment tool within this framework. This has been demonstrated by applying the detectability condition in a case study involving the requirements for the FD scheme for an inverted pendulum system.

Acknowledgement

Partial support for research reported in the paper has been provided by DEPSCOR/AFOSR grant F49620-98-1-0136, NASA Ames grant NAG 2-1158, and NASA Dryden grant NAG 4-163 (administered by the Institute for Software Research).

References

[1] D. Del Gobbo, "Sensor Failure Detection and Identification using Extended Kalman Filtering", Master thesis, Department of Mechanical and Aerospace Engineering, West Virginia University, 1998.

[2] D. Del Gobbo and M. R. Napolitano, "Issues in Fault Detectability for Dynamic Systems", submitted for presentation.

[3] X. Ding, P.M. Frank and L. Guo, "An approach to residual generator and evaluator design and synthesis", IFAC 12th Triennial World Congress, pp. 383-86,1993.

[4] S. M. Easterbrook, "The Role of Independent V&V in Upstream Software Development Processes," Journal of Integrated Design and Process Science, vol. 2, pp. 37-46, 1998.

[5] P.M. Frank, "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy-A Survey and Some New Results", Automatica, Vol. 26, No. 3, pp. 459-474, 1990.

[6] R. Isermann, "Process Fault Detection Based on Modeling and Estimation Methods-A Survey", Automatica, Vol. 20, No. 4, pp. 387-404, 1984.

[7] M. Jackson, "The Meaning of Requirements," Annals of Software Engineering, vol. 3, pp. 5-21, 1997.

[8] M. Lubars, C. Potts, and C. Richter, "A Review of the State of the Practice in Requirements Modeling," IEEE International Symposium on Requirements Engineering, San Diego, CA, IEEE Computer Society Press, pp. 2-14.

[9] NTSB final report, Accident no. MIA83IA218, September 1983.

[10] NTSB final report, Accident no. CHI93IA354, September 1993.

[11] NTSB final report, Accident no. NYC96IA169, August 1996.

[12] NTSB final report, Accident no. FTW97IA144, March 1997.

[13] M. Nyberg and L. Nielsen, "Parity functions as universal residual generators and tool for fault detectability analysis", 36th IEEE Conference on Decision and Control, vol. 5; pp.4483-9.

[14] R. Patton, P. Frank and R. Clark, "Fault Diagnosis in Dynamic Systems, Theory and Applications", Prentice Hall, 1989.

[15] Sommerville, "Software Engineering", 3rd Ed., Addison Wesley, 1989.

[16] D. R. Wallace and R. U. Fujii, "Software Verification and Validation: Its Role in Computer Assurance and Its Relationship with Software Project Management Standards", NIST Computer Systems Laboratory, Gaithersburg, MD, NIST Special Publication 500-165, 1989.

[17] N. G. Leveson, "Safeware: System Safety and Computers", Addison-Wesley, 1995

Appendix A.5

Authors:

Napolitano, M.R., Younghwan, A., Seanor, B., Pispistos, S., Martinelli, D.

Title:

"Application of a Neural Sensor Validation Scheme to Actual Boeing B737 Flight data"

Proceedings of the '99 AIAA Guidance Navigation & Control Conference, Paper 99-4236, Portland, OR, August 1999

Application of a Neural Sensor Validation Scheme to Actual Boeing 737 Flight Data

Marcello Napolitano*†, Younghwan An*‡, Brad Seanor*‡, Stelios Pispitsos*‡
*Aviation Operation and Safety Center, Department of Mechanical and Aerospace Engineering
David Martinelli§†
§Staggers Transportation Center, Department of Civil and Environmental Engineering
West Virginia University, Morgantown, WV 26506/6106

Abstract

Detection, identification, and accommodation of sensor failures can be a challenging task for complex dynamic systems such as aircraft. Classic state estimation tools based on observers and/or Kalman filters as well as alternative methods using on-line learning approximators have been proposed in the technical literature to address this problem.

This paper presents the results of a neural network based scheme to provide fault tolerance following sensor failures. Results from the application of the methodology to actual flight data of the B737 aircraft are presented. Particularly, the paper compares the performance of two different detection schemes within the SFDIA architecture applied to failures of different aircraft sensors. The non-nominal conditions are obtained by artificially imposing different failures on nominal flight data. The scheme features 'n+1' on-line learning neural approximators where 'n' is the number of sensors for which a fault tolerance scheme is desired.

The overall results confirm the interesting capabilities of neural approximators for this application and show the improvements achievable with the refined scheme in dealing with the most conservative case of soft failure.

Symbols

a = acceleration, g's
A = aileron
CAS = Calibrated Air Speed, ft/sec
DNN = Decentralized Neural Network
E = elevator
h = altitude, ft
k = discrete time index
LEF = Leading Edge Flaps
LES = Leading Edge Slats
MNN = Main Neural Network
N_1 = percentage of maximum fan RPM
nom = nominal conditions
QEE = Quadratic Estimation Error
R = rudder
TEF = Trailing Edge Flaps
δ = deflection of control surfaces, deg
μ = mean
σ^2 = variance
θ = Euler pitch angle, deg
ϕ = Euler bank angle
ψ = Euler heading angle

Introduction

Recent aviation catastrophic events have renewed an interest toward the design of fault tolerant flight control systems. While not a single aircraft with built in fault tolerance capabilities has been yet designed and tested, several efforts are undergoing to demonstrate the capabilities of different fault tolerant schemes for both military and commercial aviation. On a different side of the industry, low weight, unmanned aerial vehicles and spacecraft have been recently introduced for remote sensing for both scientific and military applications. Clearly a fault tolerant control system is very desirable for these unmanned aircraft and spacecraft.

A full fault tolerant flight control system is, in general, required to accomplish failure detection, identification, and accommodation for sensor and actuator failures (SFDIA and AFDIA). This paper focuses on the sensor failure problem and describes the results of the application of a neural network-based SFDIA scheme.

Although basic concepts of Neural Network (NN) theory were introduced in the 40s and 50s, it is only in the last decade that applications of NNs to all engineering fields has populated the technical literature. During an initial phase NNs have been applied to pattern recognition and classification. In more recent years, NNs have been used as approximators and/or controllers to model and/or control complex non-linear systems due to their approximation and adaptation capabilities[1].

Within a larger picture multi-layer neural networks can be considered as one class of approximators. In this context the words 'neural approximator' and 'neural estimator' are used interchangeably. Other types of approximators include polynomials, spline functions, rational functions, as well as a different class of NNs, the radial basis function NNs. An important effort toward creating a unified approximation theory including all these different classes was recently undertaken[2,3]. The need for such a unified approach is clear given the virtually infinite number of degrees of freedom in the selection of the approximator

architecture. For example, after having selected a multi-layer neural network among all the classes of approximators, consider the dilemma of choosing the number of inputs, the number of layers, the number of neurons per layer(s) as well as selecting a learning rate and/or momentum coefficient. Given the lack of systematic procedures for an optimal selection for the class and the topology of an approximator, a unified theoretical approach is needed. Furthermore, within this unified approach, a systematic procedure for creating non-linear approximators as well as stable learning schemes using Lyapunov's theory has been introduced[2,3] .

A partial list of interesting neural properties includes :

- applicability to MIMO non-linear systems with robustness to noise;
- parallel distributed processing and hardware implementation;
- learning (off-line, with recorded or simulated data, or on-line).

It is this last property which is particularly appealing for SFDIA purposes. In fact, classic SFDIA methods, and, at large, general FDI methods, have traditionally been based on linear modeling which clearly restricts the type of failures and failure scenarios to a limited subset. Other points of concern for the application of linear FDI and SFDIA tools to a real-life problem are their robustness to modeling discrepancies between the actual system and the filter model as well as their robustness to non white and/or biased residuals.

The paper presents the results of the application of a SFDIA scheme based on neural estimators applied to real data of a B737/300 aircraft. The data were relative to a flight test at nominal conditions. The failures are simulated through imposition of failure models on the real data; since the available data are at nominal conditions one can compare the prediction of the neural estimators with the real data to assess the overall effectiveness of the SFDIA scheme. The paper is structured as it follows. The next section reviews critical SFDIA issues while the following section briefly outlines the SFDIA scheme. A following section describes the available data and it summarizes the selected topologies of the neural estimators of the scheme. Another section describes the simulated failures. The results of the statistics of the on-line learning as well as the results of the failures simulation are described in another section. A final section concludes the paper.

Issues in sensor failure tolerance

There are two conceptually different approach to the SFDIA problem : *physical* and *analytical* redundancy. Traditional flight control systems deploy triple or quadruple physical redundancy in their network of sensors to achieve the level of reliability necessary for the aircraft certification. Typical physical redundancy SFDIA techniques are based on voting and mid-value selection schemes. It is clear that there are weight, power, size and economic penalties associated with a physical redundancy approach to the SFDIA problem.

Most of the current research activities on SFDIA focus on the use of analytical redundancy techniques. These techniques feature continuous monitoring of the measurements from the sensors. At nominal conditions these signals follow some known patterns with a certain degree of uncertainty due to the presence of system and measurement noises. However, when sensor failures occur, the observable outputs deviate from the predicted values calculated on-line or off-line from estimation schemes generating a residual. A sensor failure can be declared when the associated residual exceeds, for a single or for multiple time instants, a certain numerical threshold.

A partial list of analytical SFDIA techniques includes Generalized Likelihood Ratio (GLR), Multiple Model Kalman Filtering (MMKF), Extended Kalman Filtering and Iterative Extended Kalman Filtering (EKF and IEKF), Sequential Probability Likelihood Ratio Test (SPLRT), and Generalized Likelihood Test/Maximum Likelihood Detector (GLT/MLD)[4-7].

Although the capabilities of the techniques above are very appealing, there are some important issues related to the application of these techniques in terms of robustness to non-linearities, low signal-to-noise ratios (S/N), and modeling discrepancies between the actual system and the filter model[5-7]. In addition, there is an even more critical issue: the FDI techniques listed above are based on the use of Kalman filters which are assumed to be of the same order as the actual system. The detection task relies on residuals being, at nominal conditions, a white and unbiased sequence. However, in real-life conditions, the filter residuals may be non-white and/or biased due to :

- occurrence of a sensor failure;
- bad measurement and/or intermittent failure (statistical outliers, data gaps, temporary loss of signals);
- use of a reduced-order filter, because of constraints on available computational power.

As a result, any SFDIA scheme which does not acknowledge the last two events will be subject to a high level of false alarm rates and will, therefore, be unacceptable for real life applications.

The issues outlined above can be addressed if one can take advantage of a SFDIA scheme with the

capabilities of learning from on-line data at non-linear conditions[8-9] and, furthermore, capable of dealing with unanticipated sensor failures of non-trivial nature, such as the large class of soft-failures. Although the need for non-linear modeling methods to be used within a SFDIA scheme was clear for many years[10] several problems in the formulation of suitable non-linear approximators and the on-line implementation within the available computational resources have not allowed their use for SFDIA purposes.

Only recently, starting from the proofed capability of a single hidden layer feed-forward NN with supervised learning to map any non-linear systems - once enough training is allowed[11-12] - promising results toward assessing the stability of the neural learning have been obtained using Lyapunov's theory [2,13].

The first author believes that the problems outlined above are only a subset of a wider problems, that is the lack of formal system and software requirements for the development of SFDIA algorithms, or any other algorithm for fault tolerance. Additionally, the use of on-line learning for the neural approximators of the SFDIA scheme implies that the behavior of the system with the SFDIA might vary during its operational usage. Therefore, an integral approach to verification and validation of neural SFDIA schemes has to address the following concerns :

- Specification Assurance to System and Software Requirements (that is, *build the right thing*)
- Design and Implementation Assurance (that is, *build the thing right*)
- Reliability Assessment and Reliability Monitoring Methods (that is, *quantify assurance*).

Some may argue that this is a minor issue and/or a pure computer science or software issue; nevertheless, the first author believes that it is a Flight Control/Computer Science community problem and, furthermore, only when the requirements issue will be fully addressed, the validation and verification of neural SFDIA schemes will be possible leading the way to actual implementation of these schemes to flight control systems. Ref.[14] shows preliminary results by the first author research group in addressing the system requirements problem for the SFDIA problem.

Reviews of the NN-based SFDIA scheme

As stated in the section above, the advantage of a NN-based SFDIA scheme is that it capitalizes on the capabilities of using on-line learning. The NN-based SFDIA scheme used in this effort is shown with details in Ref.[15-17]. The advantages of on-line learning within a SFDIA scheme vs. the general robustness of a SFDIA scheme based on Kalman filters is described with details in Ref.[15].

The Extended Back Propagation (EBP) algorithm has been used by the authors for the on-line learning[9]; this algorithm has been selected for its performance in terms of learning speed, convergence time, and stability when compared to the conventional BP algorithm.

The SFDIA scheme used for this study is based on the observability of an aircraft system. It consists of a main NN (MNN) and a set of 'n' decentralized NNs (DNNs), where 'n' is the number of the sensors in the flight control system for which a SFDIA is desired. The outputs of the MNN replicate, through on-line prediction, the actual measurements from the 'n' sensors with one time instant delay, that is prediction of the state at time 'k' using measurements from 'k-1' to 'k-p' to be compared with the actual measurement at time 'k'. For the i-th of the 'n' DNNs the output is the on-line prediction of the measurement of the i-th sensor, that is, again, the prediction of the state at time 'k' using measurements from 'k-1' to 'k-p' to be compared with the actual measurement at time 'k'. The inputs to the i-th DNN are the measurements from any number to up 'n-1' sensors, that is all the 'n' sensors excluding the i-th . An estimation error exceeding at a certain time instant a certain threshold at the MMN's output provides a *detection* while an estimation error exceeding at the same time instant another threshold at the i-th DNN provides the *identification*. Then, the *accommodation* is achieved by replacing the i-th sensor with the output of the i-th DNN itself which, at the same time, is no longer in a learning mode and has a frozen numerical architecture. To allow the MNN to still be able to provide detection until the end of the flight, the output of the i-th DNN also replaces the measurement of the i-th sensor as input to the MNN.

This 'double trigger' approach, on the same hierarchical level, has been introduced for to reduce the rate of false alarms in the FDI process. Several options can be added to this scheme to add robustness to bad measurements and/or intermittent failures. For example, a lower and a higher threshold level can be selected for the MNN and the DNNs. If a lower threshold for the i-th DNN is exceeded once, the status of the corresponding i-th sensor is declared suspect and the numerical architecture of the i-th DNN is not updated. Should this status persist for a certain number of time instants and/or should the estimation error in successive time instants exceed the higher threshold, then the sensor is declared failed and is, therefore, replaced by the i-th DNN.

If the on-line computation burden is an issue, a solution would be to update the numerical

architecture of the DNNs at a lower frequency than the sensor measurement rate. This clearly increases the learning time; however, the learning can take place over a number of flights. Additional features of this scheme include the capability of handling multiple sensor failures, as long as they do not occur simultaneously, and intermittent failures. Figure 1 shows a block diagram for the SFDIA process for a system with three sensors with a simulated failure for sensor #1. A functional block diagram is instead shown in Figure 2.

SFDIA for simulated failures using B-737 data

Based on their size, sensor failures can be classified as :

- hard-over failures, catastrophic but easy to detect;
-soft failures, difficult to detect and, if uncompensated, potentially catastrophic.

In modern flight control systems Built-In-Testing (BIT) on each sensor, in lieu of FDI schemes, is typically used to detect and identify hard-over sensor failures. Therefore, this type of failure is less critical from a detection point of view. Soft failures are clearly more subtle and, potentially, more dangerous since they are not very detectable. These failures may not degrade the system performance for some time, however, if left uncompensated, can lead the system to critical and, eventually, catastrophic conditions.

In dealing with soft failures the following points need to be evaluated :

- type of soft failures (small bias, slow-drifting, combination of both);
- observability of the effect of the failures from the available measurements;
- amount of time required to accumulate enough measurements to detect the occurrence of the failure in the presence of noise;
- uniqueness of the failure and degree of distinguishability from other types of failures for unambiguous detection and isolation.

The SFDIA scheme reviewed in the section above has been tested on actual B737 data. Only simulated failures for the accelerometers and the Euler angles sensors were considered for this paper; thus the SFDIA scheme consists of 1 MNN and 6 DNNs. A complete set of dynamic data for a flight testing flight of a B737 was used; the data was provided by the NTSB for a different type of research effort[18] but proofed to be excellent data for the purpose of the this study. The time histories received from the NTSB from an extended FDR equipped B737-300 consisted of the following parameters:

θ, ϕ, ψ, a_n, a_x, a_y, CAS, Altitude, N_1, δ_E, δ_A, δ_R, δ_{TEF}, δ_{LEF}, δ_{LES}

Figure 3 shows an altitude and airspeed profile of the data. The Boeing 737-300 features on each wing 2 trailing edge flaps, 2 leading edge flaps, and 2 leading edge slats. The deflections of the leading edge surfaces are scheduled with the deflection of the trailing edge flaps and the trailing edge flaps are scheduled as a function of airspeed. The aircraft also features flight spoilers and ground spoilers which were not activated during these flight tests.

The only problem with the given data is that the sampling frequencies varied greatly among the different parameters from 1 samples/sec for N_1 to 8 samples/sec for the linear accelerations and the Euler angles. Therefore, prior to any neural processing, the data underwent a cubic interpolation in order to have the same sampling frequency of 8 samples/sec for each time history. A 5th order digital Butterworth filter was also applied to the data to reduce the noise level. The cut-off frequency was selected to avoid removing significant portions of the aircraft rigid body dynamics; a cut-off frequency of 5 rad/sec was used. A sample of the raw and filtered data is shown in Figure 4.

The next issue was the selection of the architecture for the MNN and DNNs. As it was mentioned above, there are several degrees of freedom in the selection of the neural architecture. Overall this was considered a secondary problem; the conclusions from Refs.[15,16] provided general guidelines for selecting the MNN and DNNs topologies. Therefore, architectures with a single hidden layer, with a number of hidden neurons slightly higher than the number of input data, with low learning rates, were selected. They are shown in Table 1.

The set of data covered 14,000 sec. of flight. Excluding the take-off, climbing, descent, and landing phases, there were approximately 10,500 sec. of maneuvered flight. The training for the MNN and the DNNs was conducted using 8,000 sec. of data with the remaining 2,500 sec. being used for the testing of the SFDIA scheme. The training consisted of submitting the 8,000 sec. training data to the MNN and DNNs for a total of 300 and 500 iterations, respectively. The training was monitored by "freezing" the numerical architecture of each of the NNs every 10 iterations and by letting the "frozen" NNs go through the first 500 sec. of testing data. The following classical statistical parameters for the estimation error were introduced for the MNN and each of the DNNs:

$$\mu_{Est.errX} = \frac{\sum_{i=1}^{N}(X_{actual} - X_{NN})}{N} \qquad (1)$$

$$\sigma^2_{Est.errX} = \frac{\sum_{i=1}^{N}[(X_{actual} - X_{NN}) - \mu_{Est.errX}]^2}{N} \quad (2)$$

where X indicates any of the parameters for which a SFDIA is desired (that is linear accelerations and Euler angles); the subscript NN indicated any of the DNNs or MNN outputs. N is instead the total number of data points for the testing phase (N=500 x 8 samples/sec = 4,000). Figures 5 and 6 show the statistical trend of the DNNs estimation error for 300 or 500 iterations in terms of mean and variance.

Following the simulated on-line learning of the MNN and the DNNs, the next task was to evaluate the overall SFDIA performance. The following capabilities are critical :

- failure detectability and false alarm rate for SFDI purpose;
- estimation error for SFA purposes.

In general a sensor failure can be modeled as :

$$X_{failure,i} = X_{nom,i} \pm Mn_i \quad (3)$$

where n_i is the direction vector for the i-th faulty sensor, and M is the magnitude of the failure. Based on the time-dependency failures can be classified within step-type and ramp-type failures. Therefore we would have :

- for step-type sensor failures :

$$n_i = 1 \qquad \text{for } t > t_{f1}$$

- for ramp-type sensor failure :

$$n_i = \frac{(t - t_{f1})}{(t_{f2} - t_{f1})} \quad \text{for } t_{f1} < t < t_{f2}$$

$$n_i = 1 \qquad \text{for } t > t_{f2}$$

where t_{f1} and t_{f2} indicate the initial and final time instant of ramp-type sensor failure, respectively. Both types of failures were considered in this study. Particularly, based on their magnitude, the following sensor failures were considered based on the above formula :

Failure #1: Large bias (0.5 g for acceleration - 5 deg for angles);
Failure #2: Small bias (0.2 g for acceleration - 2 deg for angles);
Failure #3: Large bias (0.5 g for acceleration - 5 deg for angles) with fast ramp transient (0.5 sec);
Failure #4: Small bas (0.2 g for acceleration - 2 deg for angles) with fast ramp transient (0.5 sec);
Failure #5: Large bias (0.5 g for acceleration - 5 deg for angles) with slow ramp transient (5 sec);
Failure #6: Small bias (0.2 g for acceleration - 2 deg for angles) with slow ramp transient (5 sec).

Failures #1,#2 represent step-type failures while Failures #3 - #6 represent soft-type failures. In the original SFDIA scheme[15-17] the following quadratic parameter for the MNN was monitored for SFD purposes :

$$QEE_{1-MNN}(k) = \frac{1}{2}\sum_{j=1}^{l}(X_j(k) - \hat{X}_{j-MNN}(k))^2 \quad (4)$$

where 'l' is the number of DNNs (in this case l=6) and the symbol '^' denotes the estimate of the j-th parameter X (either an acceleration or a Euler angle) from the MNN. However, the use of this parameter did not provide acceptable performance when dealing with soft type sensor failures. After a detailed analysis it was found that better SFD performances in dealing with soft-failures were obtained by monitoring the following quadratic parameter :

$$QEE_{2-MNN}(k) = \frac{1}{2}\sum_{j=1}^{l}(\hat{X}_{j-DNN}(k) - \hat{X}_{j-MNN}(k))^2 \quad (5)$$

where the first and the second term in the parenthesis indicate the estimate of the j-th parameter X from the the j-th DNN and from the MNN respectively. An explanation for the problems encountered using the QEE_{1-MNN} parameter for SFD purposes as well as a reason for the improvements experienced with the QEE_{2-MNN} in dealing with soft failures was found to be the following. Consider, for example, the failure of the sensor measuring the Euler angle θ and let us suppose to monitor the actual measurement of θ as well as its estimate from the MNN and the θ-DNN. At nominal conditions, at time step k the θ estimate from the MNN and the θ-DNN are supposed to emulate the actual output from the θ sensor. In the event of a ramp-type θ sensor failure the estimate of θ from the MNN tends to resemble the corrupted signal from the sensor. In fact, since the θ measurement is included as an input parameter in the MNN (see Table 1), the MNN is updated for at least one instant with the failed θ sensor value during the on-line learning. As a result, the parameter QEE_{1-MNN} does not provide an accurate detection because the difference between the measurement from the θ failed sensor and the estimate of θ from the MNN is relatively small despite the sensor failure. Instead, the estimate of θ from the θ-DNN follows the nominal value of θ relatively well. This is because the θ-DNN does not receive, as input data, the measurements from the failed sensor, as also shown in Table 1. The inconsistency between the estimates of θ from the MNN and the θ-DNN generates, therefore, the peak of the parameter QEE_{2-MNN} which is critical for SFD purposes. This problem does not occur for step-type sensor failures, for which higher peaks for QEE_{1-MNN} are induced instead.

In general, it can be said that the use of the QEE_{1-MNN} detection criterion provides better performance for step-type sensor failures whereas the QEE_{2-MNN}

detection criterion performs better for ramp-type sensor failures. The SFDIA scheme also uses the following sensor failure detection (SFD) parameter for monitoring the on-line learning of the DNNs :

$$QEE_{j-DNN} = \frac{1}{2}(X_j(k) - \hat{X}_{j-DNN}(k))^2 \quad (6)$$

For sensor failure accommodation(SFA) purposes the following parameters for the estimation error are evaluated:

$$\mu_{AEE_{j-DNN}} = \frac{1}{N_{Res}} \sum_{k=1}^{N_{res}} (X_{j-nom}(k) - \hat{X}_{j-DNN}(k)) \quad (7)$$

$$\sigma^2{}_{AEE_{j-DNN}} = \frac{1}{N_{Res}} \sum_{k=1}^{N_{res}} [(X_{j-nom}(k) - \hat{X}_{j-DNN}(k)) - \mu_{AEE_{j-DNN}}]^2 \quad (8)$$

where the subscript 'AEE' indicated 'Accommodation Estimation Error' and N_{Res} indicates the residual number of time steps from the instant when the failure of the sensor is declared to the end of the simulation.

Results of the SFDIA applied to the B-737 data

For simplicity purposes only the results relative to the simulated sensor failures for the pitch angle sensor and for the normal accelerometer are presented. The occurrence of the different sensor failures defined in the previous section was simulated. The results in terms of sensor failure detection (SFD) and sensor failure identification (SFI) are summarized in Table 2. It is clear that the $QEE_{1\text{-}MNN}$ detection parameter does not perform well in the event of soft failures and/or ramp type failures. Vice versa, the $QEE_{2\text{-}MNN}$ criteria is less effective for hard failures and/or large bias but it provides a reliable SFD in presence of soft failures and/or ramp type failures. This behavior suggests that a combination of the $QEE_{1\text{-}MNN}$ and the $QEE_{2\text{-}MNN}$ criteria provides desirable levels of SFD and SFI performance.

Clearly, as for any other SFDIA scheme, the SFDIA performance are also function of other important factors, that is the threshold(s) used for the SFD task in the MNN and the thresholds used for the SFI task in the DNNs. The selection of these thresholds also affects the false alarm/detection ratio, a critical performance of the SFDIA scheme[4-7].

A visualization of the effects of a missed detection for failure type #6 (the most conservative case – a small bias with a slow ramp transient) for the normal accelerometer is shown in Figures 7-9. In fact, using the $QEE_{1\text{-}MNN}$ detection criteria, the failure is never detected since the estimation error never exceeds the SFD threshold, as shown in Figure 7. Furthermore, a close look at Figure 8 reveals the occurrence of an undesirable event when using the $QEE_{1\text{-}MNN}$ criteria. Since the SFD using the $QEE_{1\text{-}MNN}$ criteria is not successful (Figure 7) the a_n-DNN learning is never halted and, therefore, the a_n-DNN tends to emulate the "new" dynamics through the measurements from the failed normal accelerometer (Figure 8). A different behavior is exhibited in Figures 7,8 when the combination of the $QEE_{1\text{-}MNN}$ and the $QEE_{2\text{-}MNN}$ criteria is used instead. Since the S/N ratio is lower, higher SFD thresholds are selected. When this threshold is exceeded, immediately the a_n-DNN learning is halted; therefore, the a_n-DNN will not try to learn the new dynamics explaining why the a_n-DNN quadratic error using the $QEE_{1\text{-}MNN}$ and the $QEE_{2\text{-}MNN}$ detection criteria is higher.

The results of the accommodation for the accelerometer failure are shown in Figure 9; it can be seen that a_n-DNN estimate after using the $QEE_{1\text{-}MNN}$ and the $QEE_{2\text{-}MNN}$ detection criteria is closer to the nominal a_n values than the a_n-DNN estimate after using the $QEE_{1\text{-}MNN}$ detection criteria.

The SFDIA trends described above are even more clear in the event of a θ sensor failure, documented in Figures 10-12, once again relative to the worst case scenario of a soft failure with a slow ramp. As for the a_n failure case, the use of the $QEE_{1\text{-}MNN}$ for detection does not allow the SFD (Figure 10) and, therefore, the θ-DNN tends to learn the "new" dynamics, causing the θ-DNN quadratic error to decrease after some learning. On the other side, using combination of $QEE_{1\text{-}MNN}$ and $QEE_{2\text{-}MNN}$ for detection, the θ sensor failure is detected (Figure 10) and the θ-DNN learning is halted (Figure 11). The difference in behavior between the use of $QEE_{1\text{-}MNN}$ only and the use of $QEE_{1\text{-}MNN}$ and $QEE_{2\text{-}MNN}$ is then extended to the accommodation results in Figure 12, which clearly shows the effects of the missed detection using $QEE_{1\text{-}MNN}$. In fact, the θ-DNN estimates with $QEE_{1\text{-}MNN}$ tend, with time, to replicate the outputs of the failed θ sensor; on the contrary, the θ-DNN with $QEE_{1\text{-}MNN}$ and $QEE_{2\text{-}MNN}$ closely follw the nominal θ values, that is the actual recorded θ values.

Conclusions

The paper has presented the results of the application of a neural sensor failure detection, identification, and accommodation (SFDIA) scheme to actual flight data of a B737-300 aircraft. The aircraft flight control system is assumed to be without physical redundancy in the sensors. The study confirms the capabilities of neural networks as on-line approximators. Different sensor failures were "injected" on the top of nominal flight data with

emphasis on the analysis of soft failures, whose detection and identification is more challenging. As for any other FDI schemes, the effectiveness and the performance of this neural scheme are functions of the selection of the detection and identification thresholds. In particular, the paper has emphasized the performance improvements, in the presence of soft failures, achievable with a modified detection scheme featuring an error signal originating from a main neural network (MNN) – which provides a centralized sensor failure detection – and from a set of decentralized neural networks (DNNs) – which provide a confirmation of the detection, a positive identification, followed by the accommodation.

Acknowledgements

Partial supports for the authors were provided by a NASA Ames Grant and a DoD/EPSCOR Grant.

References

1 - K.S. Narendra, and K. Partasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Transactions on Neural Networks, Vol 1, No.1, 1990

2 - M. Polycarpou, "On-Line Approximators for Nonlinear System Identification : A Unified Approach", Control and Dynamic System Series

3 - M. Polycarpou, A.T. Vemuri, "Learning Methodology for Failure Detection and Accommodation", IEEE Control Systems Magazine, Vol. 15, No. 3, June 1995, pp. 16-24

4 - A.S. Willsky, "A Survey of Several Failure Detection Methods", Automatica, Vol 12, No 6, pp. 601-611, 1976

5 - P.M. Frank, "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-Based Redundancy: A Survey and Some New Results", Automatica, vol 26, pp.459-474, 1990

6 - T. Kerr, "False Alarm and Correct Detection Probabilities Over a Time Interval for Restricted Classes of Failure Detection Algorithms", IEEE Transactions on Information Theory, vol. 20, no 4, pp.619, 1982

7 - R.J. Patton, P.M. Frank, and R.N. Clark, "Fault Diagnosis in Dynamic Systems : Theory and Applications", Prentice Hall, 1989

8 - M. Polycarpou, P.A. Ioannou, "Stable Non-Linear System Identification Using Neural Networks Models", G. Beckey and K. Goldberg Editions, Neural Networks in Robotics, pp. 147-164, Kluwer Academic Publishers, 1993

9 - M. Napolitano, C. I. Chen, S. Naylor, "Aircraft failure Detection, and Identification Using Neural Networks", AIAA Journal of Guidance, Control and Dynamics, Vol. 16, No. 6, 1003, pp.999-1009

10 - J. Gertler, "Survey of Model-Based Failure Detection and Isolation in Complex Plants", IEEE Control Systems Magazine, Vol. 8, No. 3, 1988, pp.3-11

11 - G. Cybenko, "Approximation by Superposition of Sigmoidal Functions", Math. Contr. Signals and Syst., Vol. 2, No. 4, 1989, pp. 303-309

12 - K. Hornik, M. Stinchcombe, H. White "Multilayer Feedforward Networks Are Universal Approximators", Neural Networks, Vol.2, 1989, pp. 359-366

13 - K.S. Narendra, A.N. Annaswamy, "Stable Adaptive Systems", Prentice Hall, 1989

14 - D. Del Gobbo, M.R. Napolitano, M.R.., J. Callahan, B. Cukic, "Experience in Developing System Requirements Specification for a Sensor Failure Detection and Identification Scheme", Proceedings of the 3rd IEEE High-Assurance Systems Engineering Symposium, Washington DC, November 1998

15 - M.R. Napolitano, D.A. Windon, J.L. Casanova, G. Silvestri, M. Innocenti, "Kalman Filter and Neural Network Approaches for Sensor Validation in Flight Control Systems", IEEE Control Systems Technology, Volume 6, Number 5, pp. 596, September 1998

16 - M.R. Napolitano, C. Neppach, V. Casdorph, S. Naylor, M. Innocenti, G. Silvestri, "Sensor Failure Detection , Identification and Accommodation in a System Without Sensors Redundancy", AIAA Journal of Control and Dynamics, Vol. 18, Number 6 , Nov-Dec 95, pp. 1280-1286

17 - M.R. Napolitano, G. Silvestri, D.A. Windon, J.L. Casanova, M. Innocenti, "Sensor Validation Using Hardware-Based On-Line Learning Neural Networks", IEEE Transactions on Aerospace and Electronic Systems, Volume 34, Number 2, April 1998, pg. 456

18 - M.R. Napolitano, R.D. Martinelli, D.A. Windon, J.L. Casanova "Extending Current Flight Data Recorders Capabilities : The Virtual Flight Data Recorder", AIAA-Paper 97-3538. Proceedings of the AIAA '97 Guidance Navigation and Control Conference, New Orleans, August 1997.

Table 1 Selected architectures for the MNN and the DNNs of the SFDIA

	MNN	θ-DNN	a_n-DNN
Input parameters:	I_{MNN}	$I_{\theta-DNN}$	I_{an-DNN}
Data pattern:	3	3	3
Total number of inputs:	15	12	12
Number of hidden layers:	1	1	1
Number of hidden layer neurons:	20	15	15
Number of outputs:	6	1	1
	$(\hat{\theta}, \hat{\phi}, \hat{\psi}, \hat{a}_n, \hat{a}_x, \hat{a}_y)$	$(\hat{\theta})$	$(\hat{a}_n)$
Initial learning rates:	0.01	0.01	0.001
Initial momentum coefficients:	0.1	0.1	0.1

$I_{MNN} = \theta, \phi, \psi, a_n, a_x, a_y, \delta_E, \delta_A, \delta_R, \delta_F, N_l$

$I_{\theta-DNN} = a_n, a_x, \delta_E, \delta_F$

$I_{an-DNN} = \theta, a_x, \delta_E, \delta_F$

Table 2 Comparison of detection and identification time between two detection cirteria

SF Types		$QEE_{1\text{-}MNN}$ only		$QEE_{2\text{-}MNN}$ only		$QEE_{1\text{-}MNN} + QEE_{2\text{-}MNN}$	
		θ	a_n	θ	a_n	θ	a_n
SF#1	SFDT	430.000	430.000	430.125	430.125	430.000	430.000
	SFIT	430.000	430.000	430.125	430.125	430.000	430.000
SF#2	SFDT	430.000	430.000	430.125	430.125	430.000	430.000
	SFIT	430.375	430.375	430.500	430.500	430.375	430.375
SF#3	SFDT	430.125	430.125	430.375	430.250	430.125	430.125
	SFIT	430.500	430.500	430.500	430.500	430.500	430.500
SF#4	SFDT	430.250	-	431.500	430.375	430.250	430.375
	SFIT	430.750	-	431.875	430.750	430.750	430.750
SF#5	SFDT	-	-	432.625	432.125	432.625	432.125
	SFIT	-	-	433.000	432.625	433.000	432.625
SF#6	SFDT	-	-	434.375	434.500	434.375	434.500
	SFIT	-	-	434.750	435.000	434.750	435.000

Sensor failure occurred at t = 430 sec

'-' indicates that no sensor failure is declared.

SFDT = Sensor failure detection time

SFIT = Sensor failure identification time

Table 3 Comparison of Estimation Error for Sensor Failure Accommodation purposes between the $QEE_{1\text{-}MNN}$ and $QEE_{2\text{-}MNN}$ Criteria

SF Types		$QEE_{1\text{-}MNN}$ only		$QEE_{1\text{-}MNN} + QEE_{2\text{-}MNN}$	
		θ	a_n	θ	a_n
SF#1	Mean	-0.013389	-0.034811	-0.022118	-0.035364
	Variance	0.001186	0.000867	0.001184	0.000866
SF#2	Mean	-0.008511	-0.034688	-0.012440	-0.034949
	Variance	0.001193	0.000868	0.001193	0.000869
SF#3	Mean	-0.010810	-0.034567	-0.017130	-0.034804
	Variance	0.001194	0.000869	0.001192	0.000869
SF#4	Mean	-0.010746	-0.084446	-0.052581	-0.034753
	Variance	0.001198	0.000981	0.001202	0.000872
SF#5	Mean	-1.902674	-0.138826	-0.058402	-0.034288
	Variance	0.867365	0.002974	0.001209	0.000853
SF#6	Mean	-0.976100	-0.141558	-0.076360	-0.034342
	Variance	0.202598	0.004558	0.001209	0.000846

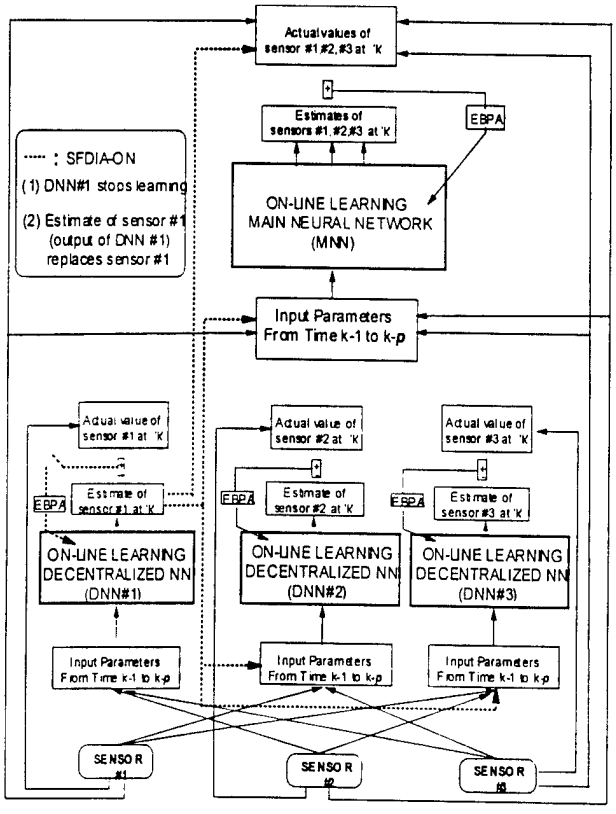


Figure 1 Block diagram of the SFDIA Scheme with sensor #1 failure

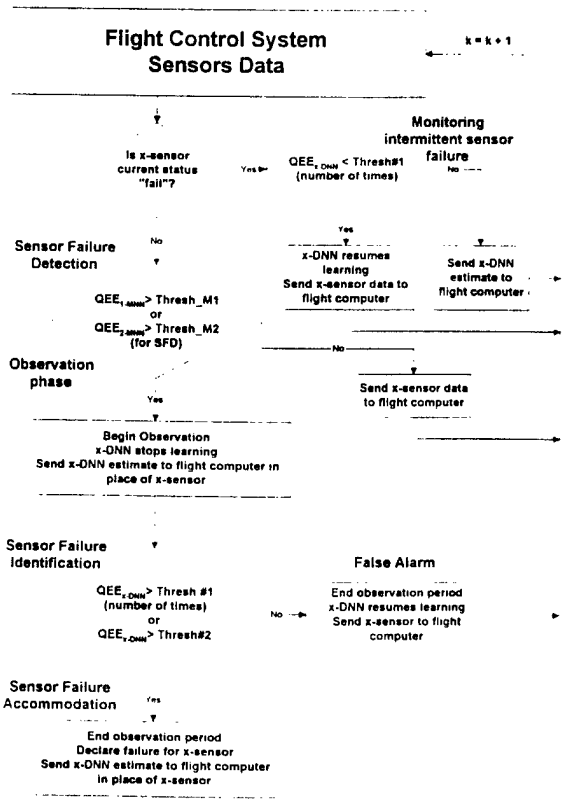


Figure 2 Functional Block Diagram of the SFDIA Scheme

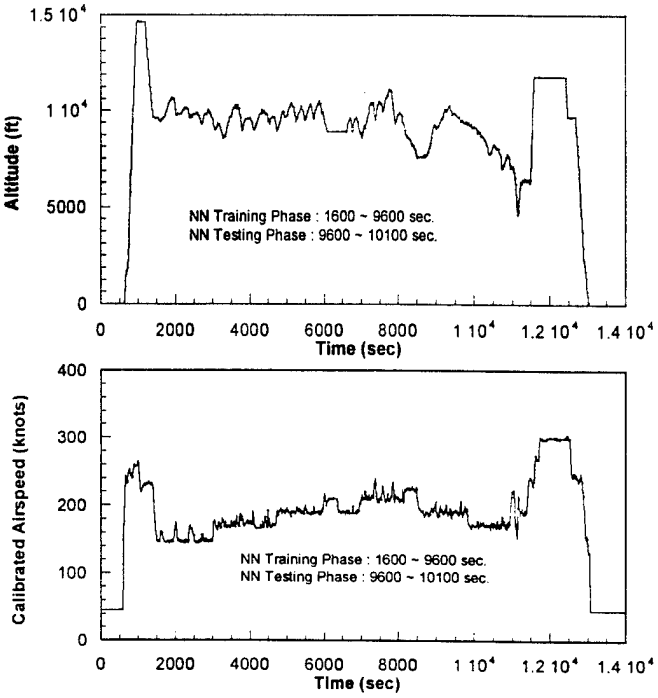


Figure 3 Altitude and airspeed profile of the B737-300 flight data

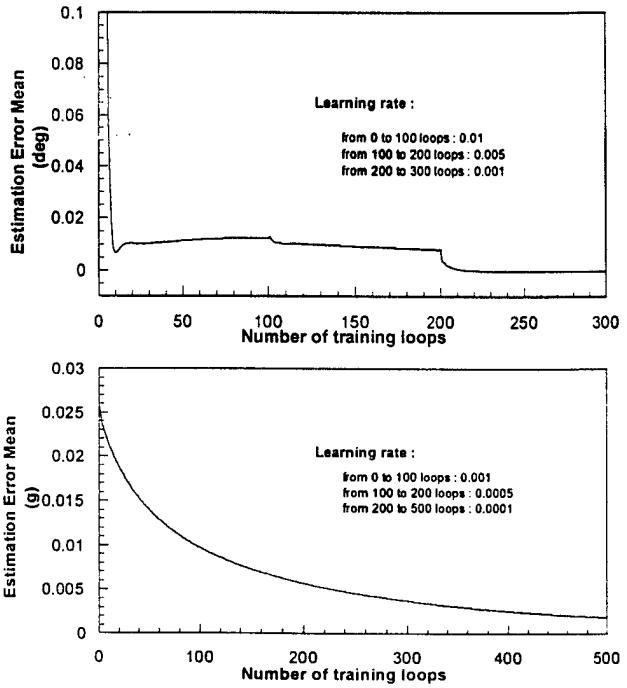


Figure 5 Estimation error mean vs. number of training loops

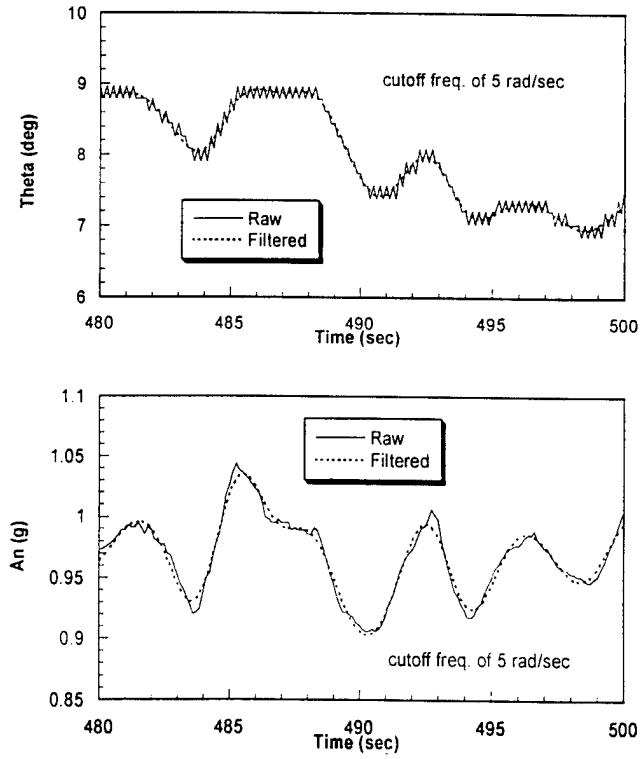


Figure 4 Sample of raw and filtered B737-300 flight data

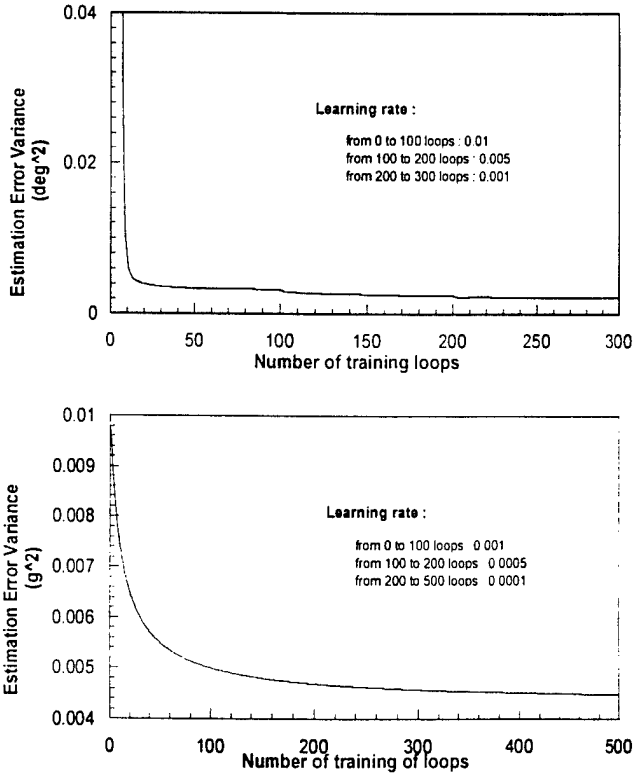


Figure 6 Estimation error variance vs. number of training loops

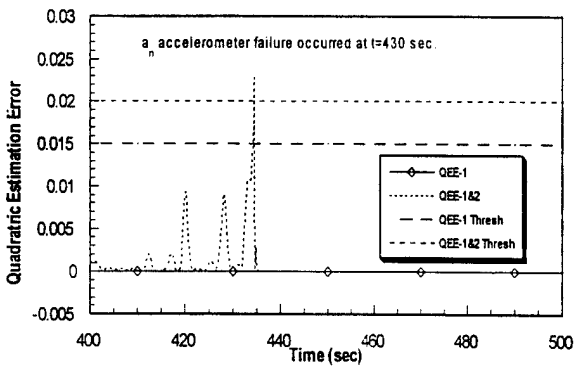


Figure 7 Plot of $QEE_{1\text{-}MNN}$ and $QEE_{2\text{-}MNN}$ vs. Time for a_n Sensor Failure Type #6

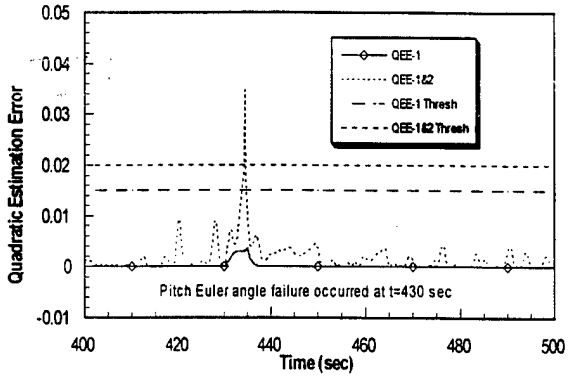


Figure 10 Plot of $QEE_{1\text{-}MNN}$ and $QEE_{2\text{-}MNN}$ vs. Time for θ Sensor Failure Type #6

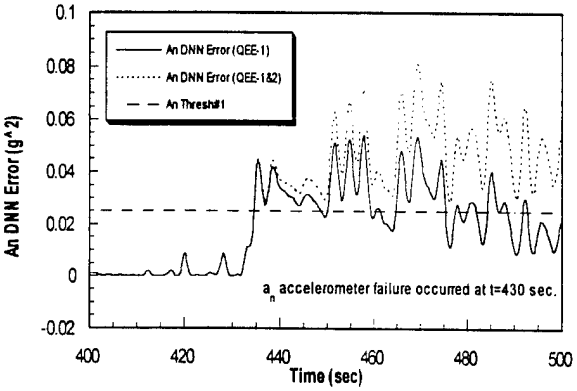


Figure 8 Plot of $QEE_{an\text{-}DNN}$ vs. Time for a_n Sensor Failure Type #6

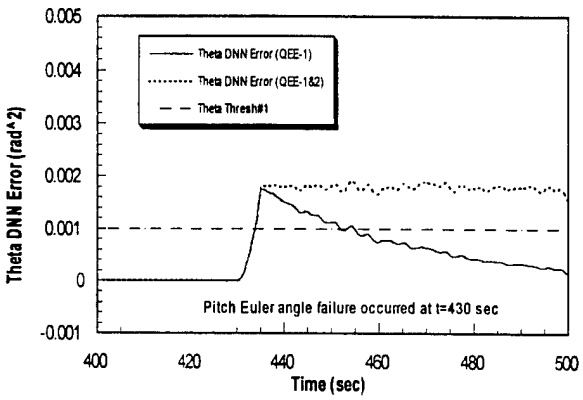


Figure 11 Plot of $QEE_{\theta\text{-}DNN}$ vs. Time for θ Sensor Failure Type #6

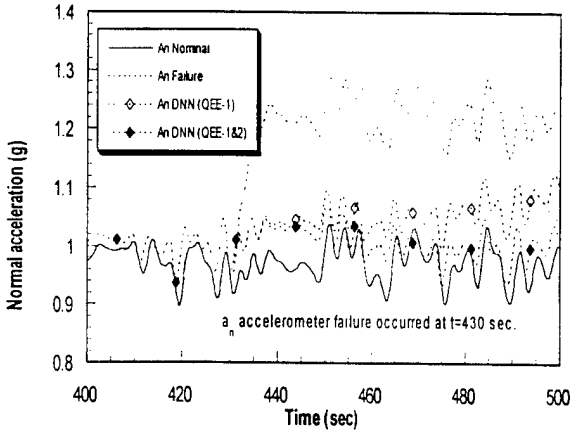


Figure 9 Plot of the Sensor Failure Accommodation for a_n Sensor Failure Type #6

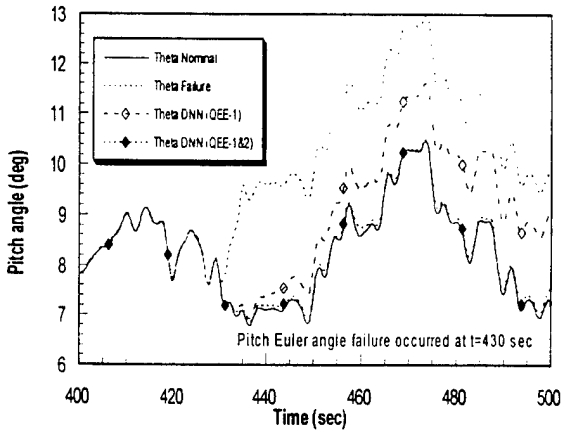


Figure 12 Plot of the Sensor Failure Accommodation for θ Sensor Failure Type #6

Appendix A.6

Authors:

Del Gobbo, D., Napolitano, M., Callahan, J, Cukic, B.

Title:

"Experience in Developing System Requirements Specification for a Sensor Failure Detection and Identification Scheme"

Proceedings of the 3rd IEEE High-Assurance Systems Engineering Symposium, Alexandria, Va, November 1998

Experience in Developing System Requirements Specification for a Sensor Failure Detection and Identification Scheme

Diego Del Gobbo, Marcello Napolitano
Mechanical and Aerospace Engineering
West Virginia University
Morgantown, WV 26505-6106
Email: {delgobbo,napolit}@cemr.wvu.edu

John Callahan, Bojan Cukic
Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV 26505-6109
Email: {callahan,cukic}@csee.wvu.edu

Abstract

This paper presents insights gained while developing System Requirements Specification of a flight control system within a formal framework. SCR methodology has been used for the description of the requirements of Sensor Failure Detection and Identification Scheme. The emphasis is on the practical aspects and experience gained through the application of a formal method in developing the system level requirements for the given application.

1. Introduction

In this paper the focus is on the application of formal methods in system requirements specification for Failure Detection and Identification (FDI) schemes. The main task of the FDI scheme is proper detection and identification of faults, in order to allow on-line or off-line accommodation by the system. The schemes under analysis are those dedicated to dynamic systems and based on analytical redundancy. Analytical redundancy is the alternative to the traditional physical redundancy approach. The latter can be summarized as a 'voting' scheme that compares redundant outputs from similar hardware elements to check for consistency [10]. One element is declared faulty if its output deviates substantially from the average value of the others. This technique is widely used and its application to sensor FDI is straightforward. On the other hand, its application to monitoring of a generic system component is not always feasible, due to the additional weight and cost.

The analytical redundancy approach is based upon the idea that the output of sensors measuring different but functionally related variables can be processed in order to detect a failure and identify the faulty component [8]. Thus, the FDI scheme considered consists of an algorithm that processes data related to the system dynamics. Many different schemes have been proposed in the technical literature. These schemes have been tested via simulations, but their validation results are not fully acceptable. Regulating agencies, such as the FAA, require a more disciplined approach to developing and validating FDI schemes. However, to date, an integrated approach to the development and validation of FDI schemes has not been addressed. The first step in this direction can be the definition of System Requirements Specification (SyRS). A well structured format for the requirements document is recommended in order to facilitate the requirements validation by domain experts[2, 6]. However, this may not be enough. Formal methodologies for SyRS, as well as automated validation tools are becoming increasingly important. By developing a formal model of the requirements, a greater level of assurance can be obtained in their validation.

In this study, an attempt is made to specify the requirements for an FDI scheme within a formal framework. The system considered is an aircraft; the objective of the FDI scheme is to detect possible failures on the roll, pitch and yaw rate gyros and to report the faulty sensors to the Recovery System. The formal technique adopted is SCR [4, 5]. The employment of a formal method for the requirements specification of the FDI scheme facilitated a better understanding of the required system behavior within its environment. While developing the system level requirements, we also gained important insights in formal description of the environment.

In the following section, the SyRS process is briefly reviewed along with formal methods. In section three, the SCR description of the system requirements for a sensor FDI scheme for an aircraft is provided. Section four illustrates the results of this study, which are summarized in section five.

2. Formalizing System Requirements

At the system level, requirements specification represent an abstraction of the system behavior. It contains all the required features of the system, without involving any details about their implementation. The construction of this high-level model of the system within a formal framework allows effective analysis and validation. A large number of formal methods and tools have been suggested in the technical literature. Most of them originated with software developers.

Despite the large number of methods, most of them have not been successfully applied in SyRS [9]. The following features turn out to be essential for a formal method to be successful in practical applications:

- A clear, immediate understanding of the system to be developed (graphical representations are usually preferred);
- Tools to analyze the specifications from a static (syntax, consistency, redundancy, etc.) and a dynamic point of view (the model of the system should be executable);
- The evolution of the requirements during the system development process requires a periodic analysis of the specifications which is unmanageable without an automated tool.

The formal method adopted in this study is SCR [4, 5]. In SCR, system behavior is described as a *mode class* (finite state machine) defined on the *monitored variables* (or input variables). The system action in the environment is modeled by means of the *controlled variables* (or output variables). In addition to inputs, outputs and modes, three more entities are used within the description: *terms*, *conditions*, and *events*. A term is an auxiliary function that helps make the specification concise; a condition is a predicate defined on the system entities; an event occurs when any system entity changes value. The description of the system is given in a tabular format. A *mode transition table* describes the finite state machine, while *condition tables* and *event tables* are used to describe output variables and terms.

3. FDI scheme: requirements specification

The FDI system is installed on an aircraft, referred to as the Aircraft System (AS). The AS is instrumented with three rate gyros that measure roll, pitch and yaw angular velocity of the aircraft. The set of the three gyros will be referred to as the Monitored System (MS). The outputs of the gyros are used within the control system loop of the AS. The AS is also equipped with a Recovery System (RS). It is assumed that faults affect only the components of the MS. If a failure occurs and the faulty gyro is identified within a certain time period from the occurrence of the failure, then the RS is able to reconfigure the control system to accommodate the failure. The accommodation of the failure will guarantee the safety of the AS. However, it will induce some degradation in the controllability of the AS. The main task of the FDI System is to identify the faulty sensor after the occurrence of a failure. The FDI System will be operational only in cruise flight condition. Thus, it should be possible to make it non-operative at take-off and landing.

The previous description has been used as a guideline for a formal specification of the FDI System requirements. Since requirements are "condition over phenomena of the environment" [7], a formal description of the environment is necessary. Hence, the SCR description illustrated here contains an abstraction of the FDI System as well as an abstraction of the environment. This description does not claim to be a complete. Rather, it represents an attempt towards the development of the requirements specification within a formal framework, with the objective of testing the suitability of the SCR method for this application.

In Figure 1, the FDI System, the RS, and AS are shown, along with their inputs and outputs. The description of the inputs and outputs for each System is provided within the SCR model.

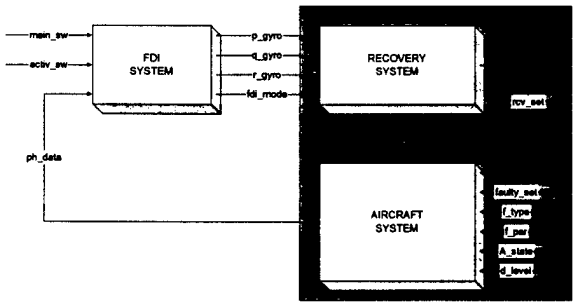


Figure 1: FDI system within its environment

The SCR model is made up of three mode classes, one for each system in Figure 1. The mode class related to the AS is called *A_ctrl* and contains modes in the set *{good, acceptable, deteriorating, bad}*. These modes represent the controllability state of the aircraft. The monitored variables for this mode class are *faulty_set, f_type, f_par, A_state, d_level, rcv_set*. By means of these variables, different fault scenarios can be described. *Faulty_set* is the set of failures that may occur in the MS. *F_type* and *f_par* are the variables that describe the type of failure occurred and the related parameters. For example the failure types of interest could be "gyro output stuck on a value", or "gyro output biased by an offset". The related failure parameters could be "stuck value" and "offset value", respectively. *A_state* represents the state of the aircraft, for example the "current maneuver". *D_level* represents a generic disturbance level, for example "turbulence".

Rcv_set is the set of recoveries performed by the RS. The *A_ctrl* mode class has only one controlled variable, *ph_data*, and one term, *nr_failure*. *Ph_data* represents the failure observable through the physical data from the aircraft instrumentation. *Nr_failure* represents the failure not yet recovered by the RS. In order to simplify the specification of the tables describing the *A_ctrl* mode class and the related entities, two functions have been introduced: *Td(nr_failure, A_state, f_type, f_par)* and *observable(nr_failure, f_type, f_par, d_level,*

A_state). The first function returns the maximum time interval the aircraft can handle the failure specified in the input parameters, before the loss of controllability. The second function returns "yes" if the failure specified in the input parameters is observable when the aircraft is in the state *A_state* under the influence of a disturbance *d_level*. A graphical representation of the AS behavior is shown in Figure 2.

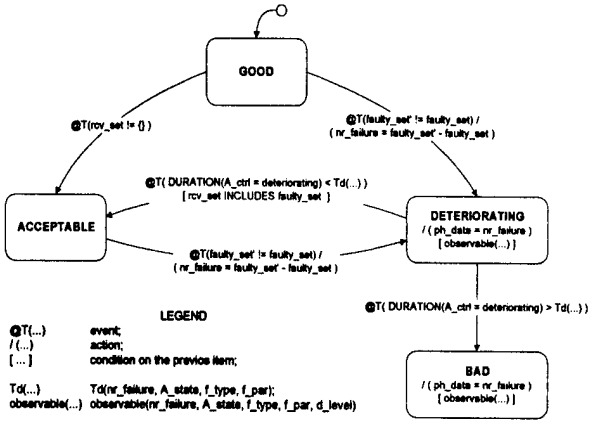


Figure 2: Aircraft System Behavior

The AS starts in *good* mode. If a failure occurs, it enters the *deteriorating* mode. If the failure is accommodated by the RS before the deadline returned by *Td(nr_failure, A_state, f_type, f_par)*, then the AS enters the *acceptable* mode, otherwise it enters the *bad* mode and the system is no longer recoverable. After the AS enters the *acceptable* mode, every failure brings the system in the *deteriorating* mode, from which it exits, as described above. It is worth noting that the AS can enter the *acceptable* mode even if no failures have occurred. In fact, in case of false detection, the RS reconfigures the AS control system inducing a degradation of the AS.

The mode class related to the FDI System is called *fdi* and contains modes in the set *{off, standby, detection}*. The monitored variables are *main_sw, activsw,* and *ph_data. Main_sw* is the switch that turns the FDI System *on* and *off*. *Activ_sw* switches the FDI System from "operative" to "non-operative". When it is "non-operative" the outputs of the system are frozen to the old values. *Ph_data* is the physical data from the AS. The controlled variables are *p_gyro, q_gyro, r_gyro, fdi_mode.* The first three variables represent the state of the roll, pitch, and yaw gyros, respectively. *Fdi_mode* represents the mode of the FDI System. A graphical representation of the system behavior is shown in Figure 3. In a typical scenario, the system starts in *off* mode where all outputs related to the gyro states are "ok".

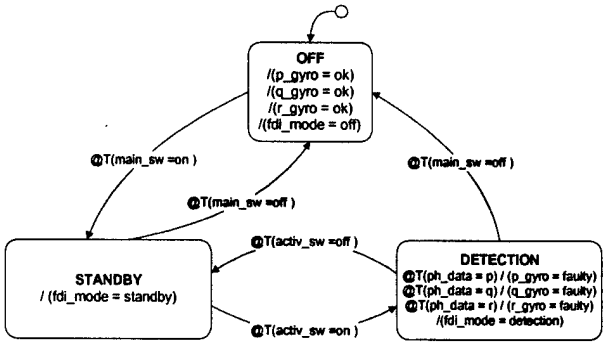


Figure 3: FDI System Behavior

After take-off the *main_sw* is turned on and the system enters the *standby* mode. In this mode the system in not yet operative; i.e., no attempt is made to detect a possible failure. When the aircraft is in cruise flight condition the *activ_sw* is turned on and the system enters the *detection* mode where it is fully operative. If a fault is detected on one of the gyros, the corresponding output is set to "faulty" allowing the RS to accommodate the AS. The aircraft flies toward the nearest airport; before initiating the landing maneuver the *activ_sw* is turned off, so that the FDI System is not operative with the output frozen to the previous values. In this way the RS keeps the reconfiguration scheme related to the detected failure, allowing the safe landing of the aircraft. Then the *main_sw* can be turned off and the FDI System outputs are all reset to "ok".

The mode class related to the RS is called *rcv* and contains the modes *{off, on}*. The monitored variables correspond to the controlled variable of the *fdi* mode class: *p_gyro, q_gyro, r_gyro, fdi_mode.* The controlled variable is *rcv_set* and represents the set of recoveries performed by the RS. A graphical representation of the RS system behavior is shown in Figure 4. The RS starts in *off* mode. When the *fdi_mode* monitored variable assumes the value "standby" the system enters the *on* mode. When the *fdi_mode* monitored variable assumes the value "off", the system gets back to the *off* mode.

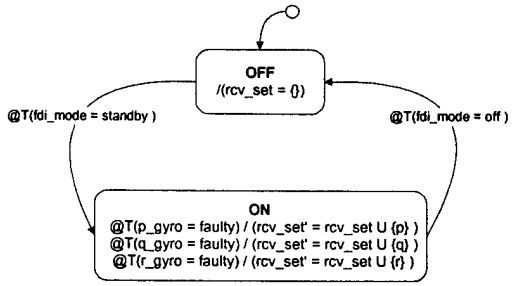


Figure 4: Recovery System Behavior

It is worth noting that graphical representations used to describe the FDI System and its environment are not part of the SCR model. The state-charts in Figures 2, 3, and 4 provide clear and concise representation of the overall system in comparison with that provided by the corresponding tables. Since one of the objective of a formal requirements specification is a communicative description of the system, we find the lack of a graphical representation within the SCR methodology a disadvantage.

4. Experiences

The major result obtained by applying a formal technique for the specification of the FDI scheme behavior within its environment has been a better understanding of the whole system. The introduction of the AS, as well as the RS within this model, played an important role in the description of the FDI scheme behavior. The major strength of this all-comprehensive abstraction lies in the interconnection between the FDI System and the AS. While the failures are injected on the sensor as an input to the AS *(faulty_set)*, the FDI System learns about these failures through the physical data *(ph_data)* recording the dynamics of the AS. If the state of the environment, as captured by *faulty_set, f_type, f_par, A_state,* and *d_level,* is such that the failure is not observable on the physical data set, then the FDI System cannot detect the failure. However, it cannot be held responsible for this. This recalls the problems related to the evaluation of the performance of the FDI System. The performance evaluation aspect has not been addressed in this study and it will be part of the future work in this area. But, a significant step toward a proper evaluation of the FDI scheme performance has been made by describing both FDI System and its environment, the performance of the FDI System can be related to the observable failure rather than to the injected failures.

5. Conclusions

In this study, the problem of formal specification of the System Requirements for an FDI scheme has been addressed. An SCR description of the FDI scheme, whose objective is to monitor the state of the roll, pitch, and yaw rate gyros of an aircraft, has been built. The FDI System environment has been modeled along with the FDI System. The need to introduce the environment abstraction in the SCR model originated from the nature of the requirements as "conditions over phenomena of the environment". The result of such approach is a better understanding of the FDI System within its environment and a system-level testing framework for the FDI System as well as for the environment. The abstract model obtained clearly expose the problems related to the assessment of the FDI scheme performance, representing a first step toward a more disciplined validation of the system.

In the near future, the SCR description of the system will be used for consistency checking. Meanwhile, State charts and Model checking techniques will be employed to improve the visual representation of the model and to express the FDI scheme requirements in terms of Temporal Logic Formulae to allow their validation.

Acknowledgments

The authors would like to thank Dr. Wu Wen and Dr. Steve Easterbrook at NASA/WVU Software Research Laboratory for helpful discussions. Partial support for the first and second author has been provided by AFOSR/DEPSCOR and NASA Ames grants.

References

[1] Brooks, F. No Silver Bullet: Essence and Accidents of Software Engineering, *Computer*, Apr. 1987.

[2] Dorfman, M. *Requirements Engineering from Software Requirements Engineering*, 2nd ed., IEEE CS Press, Los Alamitos, Calif., 1997.

[3] France, R.B., Bruel, J., and Raghavan, G. Taming The Octopus: Using Formal Models to Integrate the Octopus Object Oriented Analysis Models, Proc. of HASE"97, Washington, D.C., August,1997.

[4] Heitmeyer, C.L., Kirby, J., and Labaw, B.G. Tools for Formal Specification, Verification, and Validation of Requirements, Proc., COMPASS '97.

[5] Heitmeyer, C.L., Jeffords, R.D., and Labaw, B.G. Automated Consistency Checking of Requirements Specifications, *ACM Trans. SE and Methodology,* V5, No.3, July 1996.

[6] IEEE P1233/D3. Guide For Developing System Requirements Specifications, December 1995.

[7] Jackson M. The meaning of requirements, *Annals of Software Engineering,* V3, 1997.

[8] Patton R., Frank P. and Clark R., *Fault Diagnosis in Dynamic Systems, Theory and Applications*, Prentice Hall, 1989.

[9] Tsai, W., Mojdehbakhsh, R., and Rayadurgam, S. Experience in Capturing Requirements for Safety-Critical Medical Devices in an Industrial Environment, *Proc. HASE'97*, Washington, D.C., August 1997.

[10] Willsky A.S., A Survey of Design Methods for Failure Detection in Dynamic Systems, *Automatica*, Vol. 12, pp. 601-611, 1976.