2009

Modeling of jet engine abnormal conditions and detection using the artificial immune system paradigm

Jaclyn Marie Porter
West Virginia University

Modeling of Jet Engine Abnormal Conditions and Detection Using the Artificial Immune System Paradigm

Jaclyn Marie Porter

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in Aerospace Engineering

Committee:
Dr. Larry Banta
Dr. Marcello Napolitano
Dr. Mario Perhinschi, Chair

Department of Mechanical and Aerospace Engineering

Morgantown, WV
2009

Keywords: Artificial Immune System, Aircraft Engine Modeling, Failure Modeling,
Fault Detection

Abstract

Modeling of Jet Engine Abnormal Conditions and Detection Using the Artificial Immune System Paradigm

Jaclyn Marie Porter

Previous research at WVU has yielded promising results in the detection of aircraft sub-systems malfunctions using the artificial immune system (AIS) paradigm. However, one aircraft component that requires improvement is the aircraft propulsion system. In this research effort, MAPSS, a non-real time low-bypass turbofan engine model distributed by NASA, has been linearized and interfaced with the WVU F-15 model and the WVU 6 degrees-of-freedom flight simulator to provide a more complex engine model and create more options for engine failure modeling and engine failure detection. A variety of engine actuator and sensor failures were modeled and implemented into the simulation environment. A detection scheme based on the AIS approach was developed for specific classes of failures including throttle, burner fuel flow valve, variable nozzle area actuator, variable mixer area actuator, low-pressure spool speed sensor, low-pressure turbine exit static pressure sensor, and mixer pressure ratio sensor.

A 5-dimensional feature hyper-space is determined to build the “self” within the AIS paradigm for abnormal condition detection purposes. The WVU AIS interactive design environment based on evolutionary algorithms was used for data processing, detector generation, and limited optimization. Flight simulation data for system development and testing was acquired through experiments in the WVU 6 degrees-of-freedom flight simulator over extended areas of the flight envelope. The AIS-based detection scheme was tested using both nominal and engine failure conditions and its performance evaluated in terms of detection rates and false alarms. As compared to the previous failure detection results, significant improvement has been demonstrated as well as excellent potential for detection of the newly modeled engine failures.

Acknowledgements

First, I would like to thank my research advisor and committee chairman, Dr. Mario Perhinschi, who gave me this opportunity as well as the guidance and support I needed to complete this research and thesis. I would also like to extend thanks to Hever Moncayo and Jennifer Davis who dedicated their time to helping with my research while very busy with their own. This research could not have been completed without the pilots, David Loud, Alejandro Posada, and especially Steve Mullins, who spent countless hours in the simulator performing flight tests. In addition, thank you to my committee members, Dr. Larry Banta and Dr. Marcello Napolitano, for agreeing to take this journey with me as well as helping to provide me with the skills and knowledge needed to reach this point in my academic career.

This thesis is dedicated to my wonderful family. My parents, Jack and Brenda Porter, have provided constant love, support, and influence throughout my life and have made me who I am. They are there when I need it most, always pushing me towards something more. My crazy brother Kyle, who always provides a hug or laugh when it’s needed, you have had a huge impact on my life. I’m extremely proud of you and look forward to pushing you toward great things (not that you really need it). I would like to thank Josh for being my rock, my shoulder, and my encouragement. I definitely could not have done this without your undying love, thank you for sticking by my side. Finally, I would like to extend thanks to my friends and extended family, whether it be shopping, sporting events, nights out, or movies in, you know how to help relieve my stress or just be there for me. I love you all. Let’s go mountaineers!

Table of Contents

List of Tables

List of Figures

Nomenclature

Symbol	**Description**
English	
A	Area
K	Sensor factor
M	Mach number
N	Newton
RPM	Revolutions per minute
deg	Degrees
f	Thrust
ft	Feet
h	Altitude
in	Inch
lb	Pound
m	Slope
max	Maximum
min	Minimum
psi	Pounds per inches^2
psia	Pounds per inches^2 absolute
sec	Second
t	Time
Greek	
Δ	Difference
κ	Throttle Position
σ	Sensor Output
Subscript	
a	Actual
b	Bias
d	Drift
ef	Efficiency
f	Force or failure
m	Mass or measured
max	Maximum
min	Minimum
n	Net or noise
s	Scaling

Acronyms

AIS	Artificial Immune System
AVDS	Aviator Visual Design Simulator
BP	Bypass
CAD	Controller and Actuator Dynamics
CLM	Component Level Model
DOF	Degrees-of-freedom
FDC	Flight Dynamics and Control
FDI	Failure Detection and Isolation
FDIEA	Failure Detection, Identification, Evaluation, and Accommodation
GUI	Graphical User Interface
HP	High-Pressure
HPC	High-Pressure Compressor
HPT	High-Pressure Turbine
LP	Low-Pressure
LPT	Low-Pressure Turbine
MAPSS	Modular Aero-Propulsion System Simulation
NASA	National Aeronautics and Space Administration
PLA	Power Lever Angle
SDF	Symbolic Dynamic Filtering
VABI	Variable Area Bypass Injector
WVU	West Virginia University

Chapter 1: Introduction

1.1 Problem Definition

Typically, aircraft jet engines are modeled in a very simple way for most flight simulation and control applications using a look-up table for Mach, altitude, and throttle. Recently, the potential of using engines to control the aircraft when standard control actuators fail has been given a lot of attention. Many efforts are made to integrate engine control with aerodynamic control for improved performance and fault tolerance. In general, aircraft upset conditions (prevention, detection, and accommodation) are currently of high interest due to the increasing concern and awareness regarding aircraft safety. There is a need for "integrated and comprehensive" systems that solve this problem. This means that the system must be able to handle autonomously (without human direct intervention, without switching operational modes) abnormal conditions of "all" aircraft sub-systems, of "all" types, and over all regions of the flight envelope. An "intelligent" control system with high fault tolerance capabilities is needed.

Efforts at West Virginia University (WVU) in Fault tolerant control laws have led to the development of the WVU F-15 aircraft model within the Matlab®/Simulink® environment. The WVU aircraft simulation environment uses a graphical user interface (GUI) to allow the user to choose the aircraft flight conditions, pilot inputs, failure scenarios, etc. for the flight. The F-15 aircraft model includes modeling of nominal flight conditions, control surface failure, sensor failure, engine failure (which uses only a look-up table), and structural damage. This thesis focuses on the need for more accurate engine models to be included into the existing simulation environment and integrated into the general failure detection, identification, evaluation, and accommodation (FDIEA) process.

Artificial Immune System (AIS) techniques are being used in an attempt to solve the FDIEA problem. The WVU AIS design environment was developed to create, optimize, and test immunity based failure detectors. The AIS design environment and the WVU Simulation Environment of the F-15 aircraft model are used together to test this possible solution to the

FDIEA problem. Previous research has yielded promising results in the failure detection and identification phases for many of the modeled failures.

Since the current engine model consists of a thrust look-up table, no internal engine parameters are available to use as detectors. The existing failure detection schemes must use aircraft dynamics to try to identify engine failure. In this paper, development of a more complex engine model will create more options for engine failure modeling and engine failure detection. The new linear engine model is based on a low-bypass turbofan engine, which contains many actuators and sensors. The actuators and sensors which have the most effect on the dynamics of the engine model will be modeled as failures. It is predicted that this will not only enhance the accuracy of the engine model but also the failure detection. After the engine model development, the new available parameters are used to create immunity based failure detectors. This thesis will show improvement of the detection rates and false alarms compared to the previous failure detection results and good potential for detection of the newly developed engine failures.

1.2 Research Objectives

The research effort presented in this thesis was aimed at reaching the following research objectives.

- Develop and test a more accurate linear engine model to integrate with the WVU F15 Matlab/Simulink® model replacing the previous engine model based only on thrust look-up tables.
- Develop and test various engine specific actuator and sensor failures.
- Investigate/design detection schemes based on the Artificial Immune System. The latest is based on a detailed analysis of interactions between failed components and the rest of the engine, and the dynamic response of the aircraft. In other words, analysis of how much individual engine parts affect everything else. This is useful to eventually define the set of identifiers to be used in the AIS-based detection.
- Compare the results of the AIS detection schemes with previous results that were based only on the aircraft response.

1.3 Thesis Overview

The following chapter structure describes the organization of this thesis.

- Chapter 2 contains the literature review, which presents descriptions of engine and engine failure model development and various research on the subjects of AIS and FDIE.
- Chapter 3 introduces/discusses the WVU Simulation Environment which includes the Simulink® model, the WVU 6-DOF Flight Simulator, and the Graphical User Interface.
- Chapter 4 describes the engine model development, including the creation of the linear model and the description of the controller, sensor, and actuator dynamics subsystems. This chapter also contains an example and explanation of a healthy engine simulation.
- Chapter 5 consists of the engine failure modeling and provides an example of an unhealthy engine simulation.
- Chapter 6 presents the design of the failure detection scheme including descriptions of AIS, the self, data processing, identifiers, and detectors used for the engine model.
- Chapter 7 consists of the experimental procedures, which explains in detail, the flight plans and testing, the development of the self, and the detection of failures.
- Chapter 8 displays and discusses validation of the system as well as the results of the AIS on the flight tests of several engine failures. New failures are examined for the first time while others are compared to the results of previous engine failure detection.

Chapter 2: Literature Review

2.1 Engine Models

In order to create the architecture needed for a more complex engine model, investigation into existing engine models was done. A number of models were found which were being used for different purposes. The type of engine model needed for application in the WVU F-15 aircraft model should represent a generic military type jet engine.

While nonlinear models may be accurate, they are typically non-real time and cannot be used in real-time simulation and are difficult (even impossible) to be used as such for control system design. Simpler and faster linear models must be developed from nonlinear differential and algebraic equations for use in these applications [1]. A nonlinear engine model is estimated, usually from test data, around an operating point to obtain a linear model. These linear time invariant (LTI) models are used for engine control despite their limited flight envelope.

For transient engine operations, the linear model would ideally include multiple operating points [1]. This is done through a piecewise-linear model, which interpolates several linear models at different operating points in the flight envelope. While the piecewise-linear models provide greater accuracy for fast simulations, they are not convenient for controller design since they are technically still nonlinear. “Jet Engine Model for Control and Real-Time Simulations” [1] discussed a simple/fast quasilinear engine model based on nonlinear functions used for controller design. Simulation results comparing nonlinear and fast models were found to reasonably agree.

Many of the turbofan engine models and simulators used in industry are based on numerous lookup tables and empirical data derived from real experiments [2, 3]. For the design of low complexity gain-scheduled control laws, linear models are necessary at relevant points in the flight envelope. Henrion et al. [4] have presented a methodology for the derivation of such linearized models of aircraft turbofan engine dynamics from standard engine simulators used in industry. Several versions of the Matlab linearization function *linmod* were used for this purpose and some issues related to the performance of these algorithms were identified and discussed. The authors also emphasized the sensitivity of model accuracy with the selection of flight condition and excitation input.

Despite the effectiveness of engine models based on look-up tables for control design, NASA researchers observed interest in intelligent engine component technology particularly in the field of aircraft health management [5]. As a result, the Modular Aero-Propulsion System Simulation, or MAPSS, was developed. MAPSS uses a nonlinear simulation of a generic turbofan engine in Simulink® along with a GUI to create a computer aided control design and simulation package with graphical representation of the engine systems. Validation was done using a FORTRAN engine simulation. MAPSS provides easy access to health, engine, and control parameters along with the option to create linear models based on the nonlinear steady-state response to certain operating conditions for Mach, altitude, and power lever angle (PLA) [5, 6].

2.2 Engine Failure Models

For most applications, engine failure models consist of simple failures/malfunctions based on available input parameters, such as throttle failures or reduced thrust output, etc. Currently within the WVU F15 aircraft model are three options for engine failures [7]. The "stuck throttle" engine failure corresponds to a functioning engine with no response to throttle actuation. The "thrust runaway" failure represents a malfunction of the fuel control system which reaches maximum fuel flow resulting in increased thrust. Since modeling of the fuel control system is unavailable, this failure is modeled by increasing the throttle to maximum over a period of time. Finally, reducing the throttle input by a constant factor represents the "power/thrust reduced control efficiency". It is apparent that these engine failures are modeled based on throttle input since it is the only available parameter [7].

Clearly, once a more complex engine model is developed, more complex engine failure models can be used. Sarkar et al. [8] discuss some aircraft engine failures on a simulation test bed. The engine model used is similar to the NASA MAPSS model. Here, the engine failures are based on the efficiency health parameter and the flow health parameter. Each of the six main engine components (defined in Chapter 4) has these health parameters which affect the efficiency and flow of that particular component. Once the health parameter values at healthy conditions are obtained, the failure is created by setting the nominal condition of each parameter at 1.0 below the healthy condition. Reducing the health parameter value below the nominal condition represents an engine component failure [8].

With engine models such as MAPSS [6] that contain actuator dynamics and sensor outputs, the same modeling techniques used in other engine actuator and sensor failures can be applied. Aircraft actuator failures within the WVU F-15 aircraft model include modeling of a stuck aerodynamic control surface where the control surface remains fixed in the current position/deflection or moves to a user defined position at the time of failure [7]. Also within this aircraft model are aircraft sensor failures which consist of an output bias. The sensor output transitions to this bias either instantaneous, known as a step bias, or over a period of time, called a drifting bias. The size of the bias (large or small) and the speed of the drift (fast or slow) are user defined [7].

2.3 Failure Detection and Accommodation

The main purpose for building a more complex engine model is to improve the detection of engine failures. Many failure detection schemes are being developed in an attempt to improve safety and reliability of aircraft operations. Sarkar et al. [8, 9] present concepts for fault detection and isolation (FDI) with application on an aircraft gas turbine model. The focus here is detection and isolation of failures close to the time of occurrence. The FDI algorithms used by the authors are based on symbolic dynamic filtering (SDF), which is built upon the principles of symbolic dynamics, statistical pattern recognition, and information theory [9]. SDF was found to be effective not only with catastrophic failures but also with small magnitude failures. Since the engine failure model is based on health parameters, the failure can be made to start out small and evolve over a period time. The purpose of detection of these growing engine failures is to monitor the deterioration of different engine parts over the life of the engine.

Another promising technique, which is in development at WVU [7], is based on the human immune system. This concept is called the artificial immune system (AIS). AIS-based failure detection is designed to distinguish parts of the system that do not belong to the “self”. The basic idea is that an abnormal condition, or failure, is declared when the current system does not correspond to normal conditions. Moncayo et al. [7] have used this concept with the WVU F-15 aircraft model in an attempt to solve the fault detection, identification, and evaluation (FDIE) problem in a variety of aircraft sensor, actuator, propulsion (engine), and structural failures/damages over an extended flight envelope. The system’s effectiveness is measured in terms of rates of detection and false alarms. “Detection” refers to positive detection during the time period after the failure occurs while “false alarms” are positive detections when no failure is present. The authors show promising potential of the AIS in the field of aircraft sub-system failure detection and identification.

Chapter 3: WVU Simulation Tools

3.1 General Description of the WVU Simulation Environment

An advanced simulation environment has been developed at WVU [10] to support the design, evaluation, and validation of aircraft fault-tolerant control laws. Matlab® and Simulink® are used to ensure maximum portability and flexibility. Model blocks from the *Flight Dynamics and Control* (FDC) toolbox [11] are included for solving the equations of motion, and modeling wind and atmospheric turbulence effects. The dynamic model is interfaced with the *Aviator Visual Design Simulator* (AVDS) simulation package [12] to provide visual cues when used directly on a desktop computer. The main components of the WVU simulation environment are presented in Figure 3.1. These five modules are:

- Aircraft Model Module
- Control System Module
- Aircraft Sub-System Failure Models
- Failure Detection and Identification Schemes
- User Interface

The high level Simulink model for the direct desktop configuration is shown in Figure 3.2.

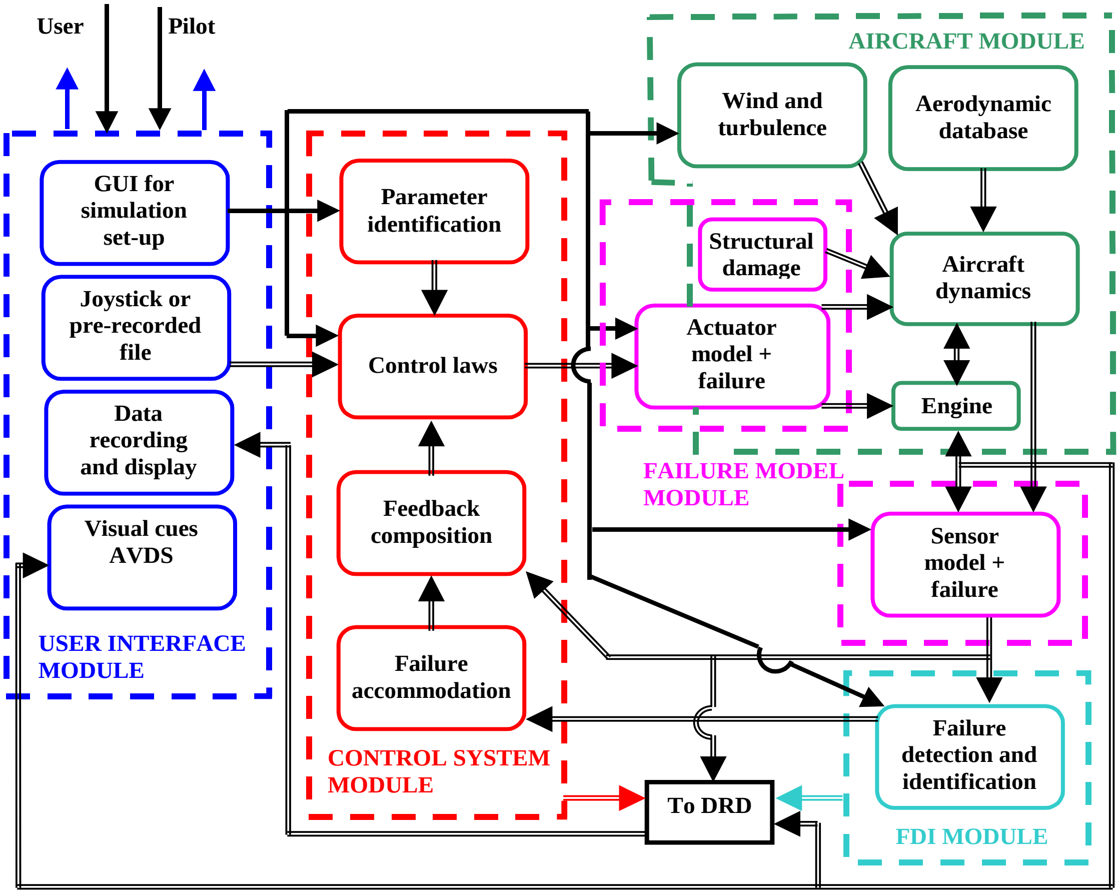


Figure 3.1: General Architecture of the WVU Simulation Environment

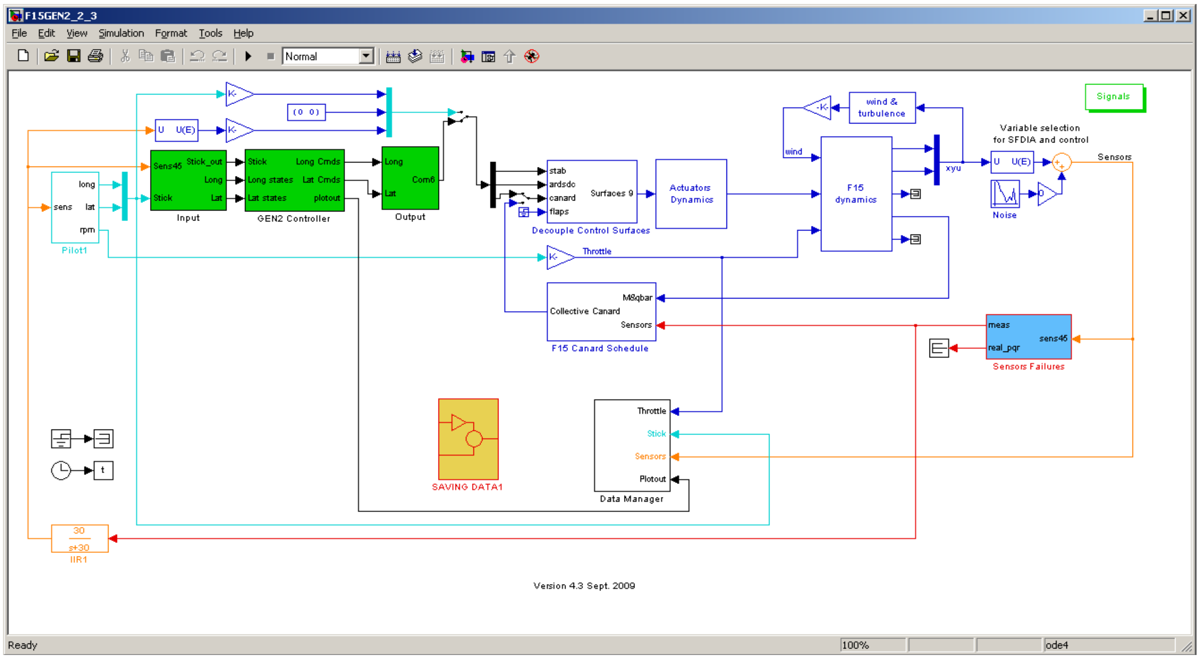


Figure 3.2: Simulink Model of the WVU Simulation Environment – Direct Desktop Configuration

The simulation environment can also be interfaced with the WVU 6 degrees-of-freedom flight simulator interacting with the simulator supporting software, X-Plane [13]. This configuration, illustrated in Figure 3.3 [14], is used to perform tests in the flight simulator. The Simulink model interfaced with the WVU 6-DOF flight simulator and X-Plane is shown in Figure 3.4.

The aircraft dynamic model can be flown using a joystick or a set of pre-recorded command time histories. User-friendly GUI menus are used to set the conditions for the simulation scenarios, including a variety of options related to the architecture of the control laws, failure type and magnitude, and input/output content. An example of the user interface provided by the AVDS visualization and the monitoring of relevant engine parameters using Simulink® scopes is presented in Figure 3.5.

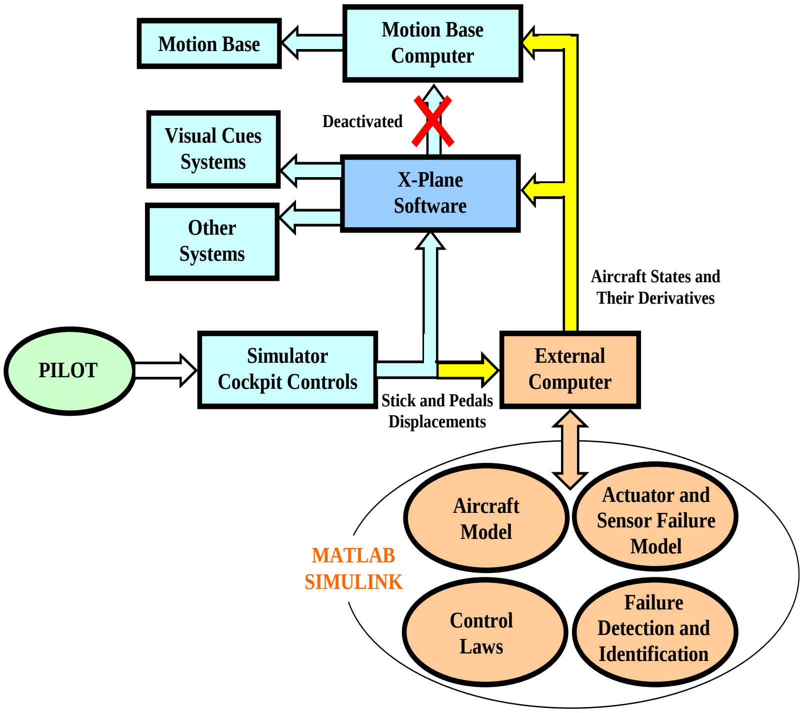

Motion Base
Motion Base Computer
Deactivated
Visual Cues Systems
X-Plane Software
Other Systems
Aircraft States and Their Derivatives
PILOT
Simulator Cockpit Controls
Stick and Pedals Displacements
External Computer
MATLAB SIMULINK
Aircraft Model
Actuator and Sensor Failure Model
Control Laws
Failure Detection and Identification

Figure 3.3: Integration of the WVU Flight Simulator with External Models [14]

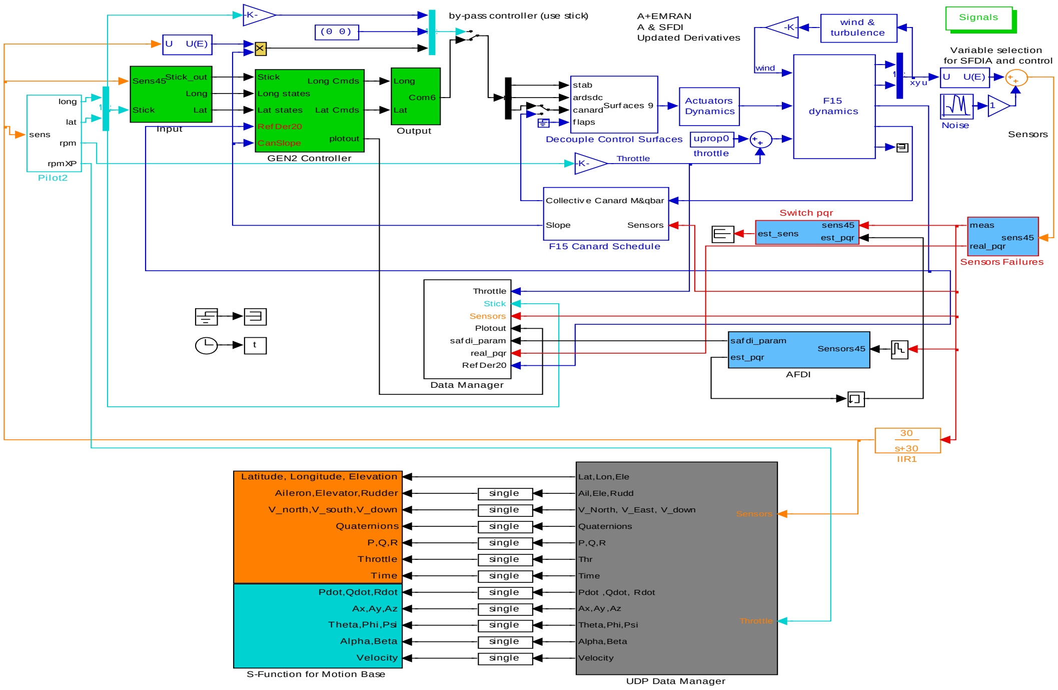
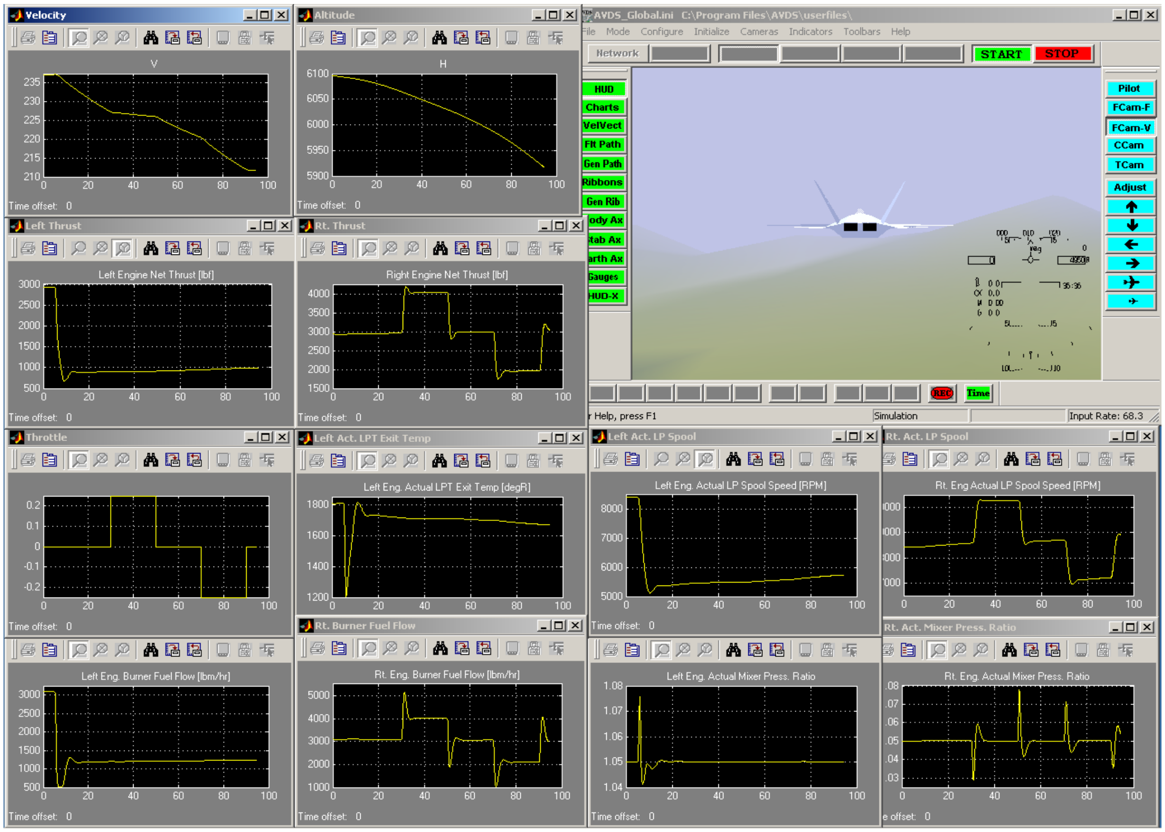


Figure 3.4: Simulink Model of the WVU Simulation Environment – Integration with 6-DOF Flight Simulator



Velocity
V
Time offset: 0
Altitude
H
Time offset: 0
AVDS_Global.ini C:\Program Files\AVDS\userfiles\
File Mode Configure Initialize Cameras Indicators Toolbars Help
Network
START
STOP
HUD
Charts
VelVect
Flt Path
Gen Path
Ribbons
Gen Rib
ody Ax
tab Ax
arth Ax
Gauges
HUD-X
Pilot
FCam-F
FCam-V
CCam
TCam
Adjust
REC
Time
r Help, press F1
Simulation
Input Rate: 68.3
Left Thrust
Left Engine Net Thrust [lbf]
Time offset: 0
Rt. Thrust
Right Engine Net Thrust [lbf]
Time offset: 0
Throttle
Time offset: 0
Left Act. LPT Exit Temp
Left Eng. Actual LPT Exit Temp [degR]
Left Act. LP Spool
Left Eng. Actual LP Spool Speed [RPM]
Time offset: 0
Rt. Act. LP Spool
Rt. Eng Actual LP Spool Speed [RPM]
Rt. Burner Fuel Flow
Left Eng. Burner Fuel Flow [lbm/hr]
Time offset: 0
Rt. Eng. Burner Fuel Flow [lbm/hr]
Time offset: 0
Left Eng. Actual Mixer Press. Ratio
Time offset: 0
Rt. Act. Mixer Press. Ratio
Rt. Eng. Actual Mixer Press. Ratio
e offset: 0

Figure 3.5: Interface of the WVU Simulation Environment

3.2 Aircraft Model

The Aircraft Model Module can be considered to include four major components: the wind and turbulence model, the aircraft dynamic equations of motion, the aerodynamic database, and the engine model. The development of a more accurate and flexible engine model was a major objective of this thesis. The initial engine model within the WVU simulation environment consisted of a simple 3-dimensional look-up table for thrust as a function of throttle, Mach, and altitude. Chapter 4 provides details regarding the development of the new engine model including all major internal actuators and sensors as well as the internal dedicated engine controller.

The Dryden model implemented within the FDC toolbox was used to simulate turbulence effects and constant wind of pre-determined direction and magnitude. The FDC toolbox 'Equations of Motion Solver' was used, which implements general rigid body dynamics assuming constant mass and inertias. The computation of aerodynamic forces and moments is distributed for each control surface, wing, and engine. This specific feature is necessary for modeling the failures with adequate level of generality.

The aerodynamic model was derived from a non-linear model of a high performance military aircraft distributed by NASA to academic institutions in 1990 within a student design competition [15]. The aerodynamic database is structured with a number of look-up tables, which are functions of one or more dynamic variables, such as Mach number and/or angle of attack.

3.3 Control System Model

Two general strategies for adaptive control laws are implemented within the WVU simulation environment: indirect and direct adaptive flight control laws [10]. Indirect adaptive flight control laws consist of optimal control design approach and frequency domain-based on-line parameter estimation. The direct adaptive flight control laws design is based on non-linear dynamic inversion at a reference nominal flight condition plus artificial neural networks augmentation to compensate for inversion errors and abnormal flight conditions. For the purpose of this research effort, the direct adaptive control laws were used. The engine has its own internal controller, which will be described in Chapter 4.

As part of the general control system, the WVU simulation environment includes failure detection identification and accommodation (FDIA) schemes for aerodynamic actuators and sensors relying on monitoring relevant dynamic measurements, information processing based on neural estimators, and comparison against pre-determined thresholds. In this thesis, a novel approach is developed and analyzed for engine abnormal conditions detection based on an artificial immune system. The detailed design and performance evaluation of this detection method is presented in Chapters 6 and 8.

3.4 Failure Models

In this section, a brief overview of sub-system failure models previously implemented within the WVU simulation environment is presented. The engine failure models associated to the more accurate engine model developed for the purposes of this thesis are discussed in detail in Chapter 5.

Two types of aerodynamic control surface failure are implemented within the WVU simulation environment. The first failure type corresponds to an actuator mechanism failure. The control surface remains fixed in the current or in a user prescribed position at post-failure conditions. The second failure type corresponds to a physical destruction and/or deformation of the control surface. It consists of a deterioration of the aerodynamic “efficiency” of the control surface starting at the occurrence of the failure [16]. The user can select different failure parameters such as type, occurrence time, position, and magnitude affecting any of the individual eight control surfaces of the baseline aircraft, which is left or right stabilators, ailerons, canards, or rudders.

All the sensors that are typically used in the control laws feedback (e.g. angular rate sensors) have been first represented as first order systems [10]:

$$\sigma_a(s) = \frac{1}{\tau_s s + 1} x(s) \qquad \textbf{Eq 3-1}$$

where σ_a is a noiseless sensor output and x is the actual measured variable as it results from the mathematical model of the aircraft. The sensor output can be expressed in general as:

$$\sigma_m(t) = \max\left[\min\left(\overline{K}_s(t)\cdot\sigma_a(t) + \overline{\sigma}_b(t) + \overline{K}_d(t)\cdot(t - t_f), \sigma_{max}\right), \sigma_{min}\right] + \overline{K}_n(t)\cdot\sigma_n(t) \quad \textbf{Eq 3-2}$$

where $\overline{K}_s$, $\overline{K}_d$, $\overline{K}_n$, and $\overline{\sigma}_b$ are, respectively, a sensor output scaling factor, a drift factor, a noise amplifier, and a bias. The current time is denoted by t and the moment of failure occurrence by t_f. By properly selecting values for the 8 modeling parameters (K_s, K_d, K_n, σ_b, σ_{max}, and σ_{min},) the following types of sensor failures can be simulated:

- biased sensor output with variable rate
- drifting output
- constant or saturated output
- increased output noise

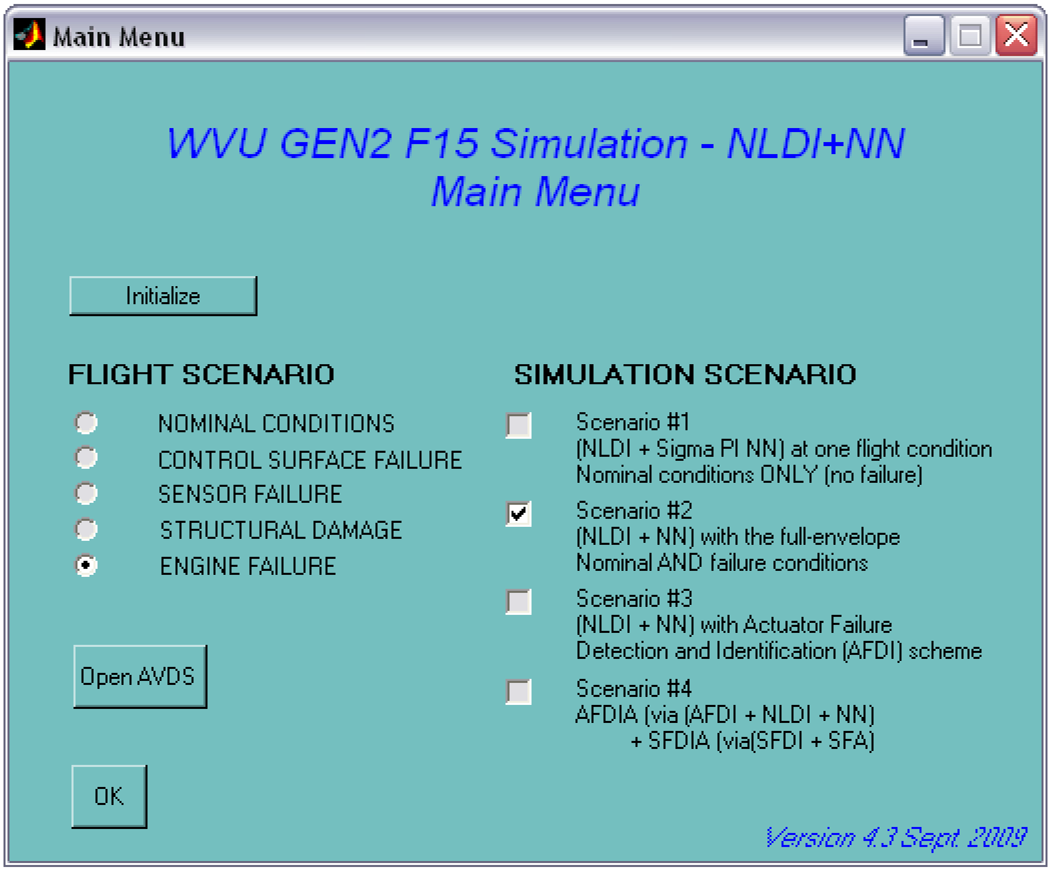
A simple model of wing damage was implemented considering both aerodynamic and gravimetric effects. It is assumed that the structural damage to the wing will affect the aerodynamic forces and moments through the reduction of the aerodynamically active area and through the alteration of the aerodynamic characteristics in terms of stability derivatives. These effects are modeled using two parameters, a wing area damage factor and an aerodynamic damage factor. To model the effects of the wing damage on the gravimetric characteristics of the aircraft, the wing area damage factor is used to proportionally reduce the total mass of the aircraft and to alter the moments of inertia. Note that the equations of motion are still based on the assumptions of constant mass, inertias, and center of mass location. The effects of the non-symmetric location of the center of mass after the occurrence of the failure are modeled as equivalent alterations of the moment coefficients.

The following engine failure/malfunction models have initially been included in the WVU simulation environment: stuck throttle, thrust runaway, and power/thrust reduced control efficiency. The further development of these models is presented in Chapter 5.

3.5 Graphical User Interface (GUI)

The main portal menu, presented in Figure 3.6, allows the user to select simulation under nominal ("healthy") flight conditions or under abnormal operation of one of the main sub-systems: control surfaces, airframe control system sensors, structure, or propulsion. Different failure detection schemes can also be selected from this menu.

The menu presented in Figure 3.7, allows the user to select the source of the input to the simulation



FLIGHT SCENARIO

○ NOMINAL CONDITIONS
○ CONTROL SURFACE FAILURE
○ SENSOR FAILURE
○ STRUCTURAL DAMAGE
● ENGINE FAILURE

SIMULATION SCENARIO

□ Scenario #1
(NLDI + Sigma PI NN) at one flight condition
Nominal conditions ONLY (no failure)

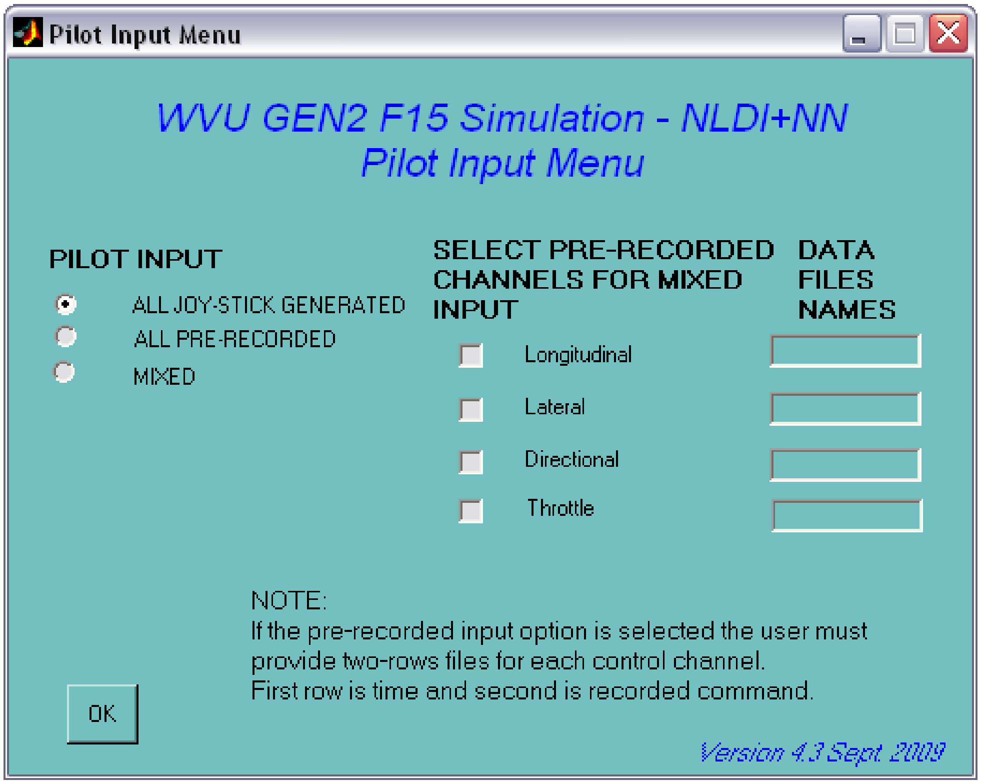
☑ Scenario #2
(NLDI + NN) with the full-envelope
Nominal AND failure conditions

□ Scenario #3
(NLDI + NN) with Actuator Failure
Detection and Identification (AFDI) scheme

□ Scenario #4
AFDIA (via (AFDI + NLDI + NN)
+ SFDIA (via(SFDI + SFA)

Figure 3.6: WVU Simulation Environment – Main Portal Menu

Should a simulation scenario at abnormal conditions be selected from the menu in Figure 3.6 for any of the main aircraft four sub-systems, corresponding input windows are opened to allow the user to configure the type, magnitude, and moment of occurrence of the failure. In Figure 3.8, the input menu for the engine related failure is presented.



Pilot Input Menu

WVU GEN2 F15 Simulation - NLDI+NN
Pilot Input Menu

PILOT INPUT

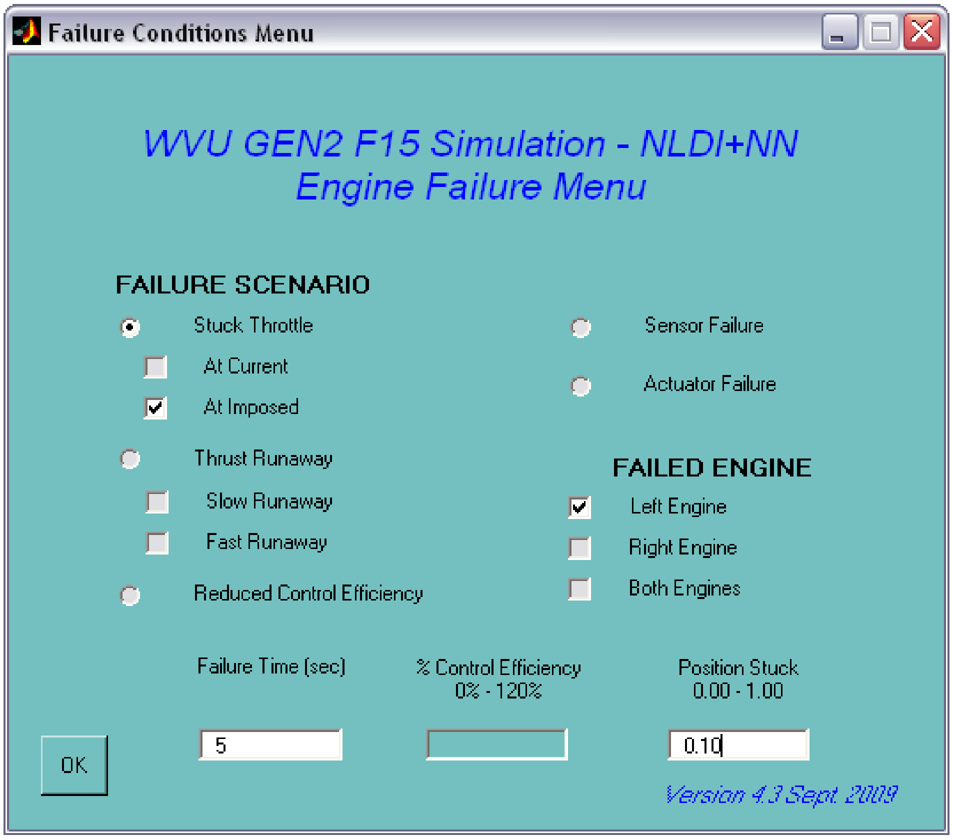
● ALL JOY-STICK GENERATED
○ ALL PRE-RECORDED
○ MIXED

SELECT PRE-RECORDED CHANNELS FOR MIXED INPUT | DATA FILES NAMES

□ Longitudinal
□ Lateral
□ Directional
□ Throttle

NOTE:
If the pre-recorded input option is selected the user must provide two-rows files for each control channel.
First row is time and second is recorded command.

Figure 3.7: Selection of Simulation Input Source



Failure Conditions Menu

WVU GEN2 F15 Simulation - NLDI+NN
Engine Failure Menu

FAILURE SCENARIO

● Stuck Throttle
- □ At Current
- ☑ At Imposed

○ Thrust Runaway
- □ Slow Runaway
- □ Fast Runaway

○ Reduced Control Efficiency

○ Sensor Failure

○ Actuator Failure

FAILED ENGINE

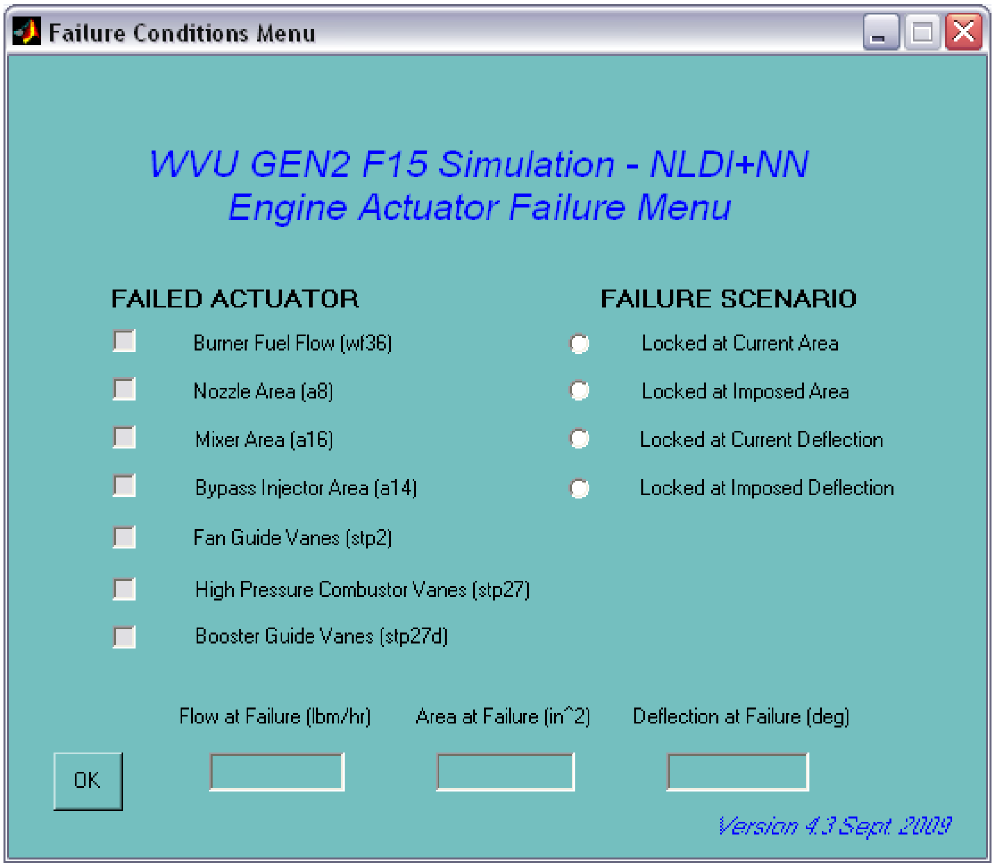
☑ Left Engine
□ Right Engine
□ Both Engines

Failure Time (sec): 5
% Control Efficiency 0% - 120%:
Position Stuck 0.00 - 1.00: 0.10

Figure 3.8: Selection of Engine Abnormal Operation Condition

Two large categories of failures are implemented: failures related to the throttle actuator and the command chain between the cockpit and the engine and failures affecting the internal actuators or sensors of the engine. Three types of failure directly related to the throttle actuator can be selected by checking the input boxes on the left side of the input window. The options include stuck throttle at current position or at a user specified position, slow or fast thrust runaway, and reduced control efficiency. The magnitude of such a failure can be established on this menu as is the moment of occurrence. The specific failed element for an internal engine failure is specified using a next window as well as the failure magnitude. Only the time of occurrence is input here. Any of the two engines or both can be affected by the failures.

If an internal engine actuator failure is desired, the menu presented in Figure 3.9 is opened. It allows the user to specify which engine actuator out of seven will fail, if the failure condition involves lockage at current situation or migration to an imposed one, and what the magnitude of the failure is.



Failure Conditions Menu

WVU GEN2 F15 Simulation - NLDI+NN
Engine Actuator Failure Menu

FAILED ACTUATOR

□ Burner Fuel Flow (wf36)
□ Nozzle Area (a8)
□ Mixer Area (a16)
□ Bypass Injector Area (a14)
□ Fan Guide Vanes (stp2)
□ High Pressure Combustor Vanes (stp27)
□ Booster Guide Vanes (stp27d)

FAILURE SCENARIO

○ Locked at Current Area
○ Locked at Imposed Area
○ Locked at Current Deflection
○ Locked at Imposed Deflection

Flow at Failure (lbm/hr)

Area at Failure (in^2)

Deflection at Failure (deg)

OK

Version 4.3 Sept. 2009

Figure 3.9: Set-up of Engine Actuator Failure

If an internal engine sensor failure is desired, the menu presented in Figure 3.10 is opened. It allows the user to specify which engine sensor out of eight will fail, the characteristic or type of failure and the magnitude of the failure. The types of failure implemented include: sensor output step bias (small or large), sensor output drifting bias (all combinations of large, small, fast, and slow), and constant sensor output, either null output or saturated output at the minimum or maximum possible value.

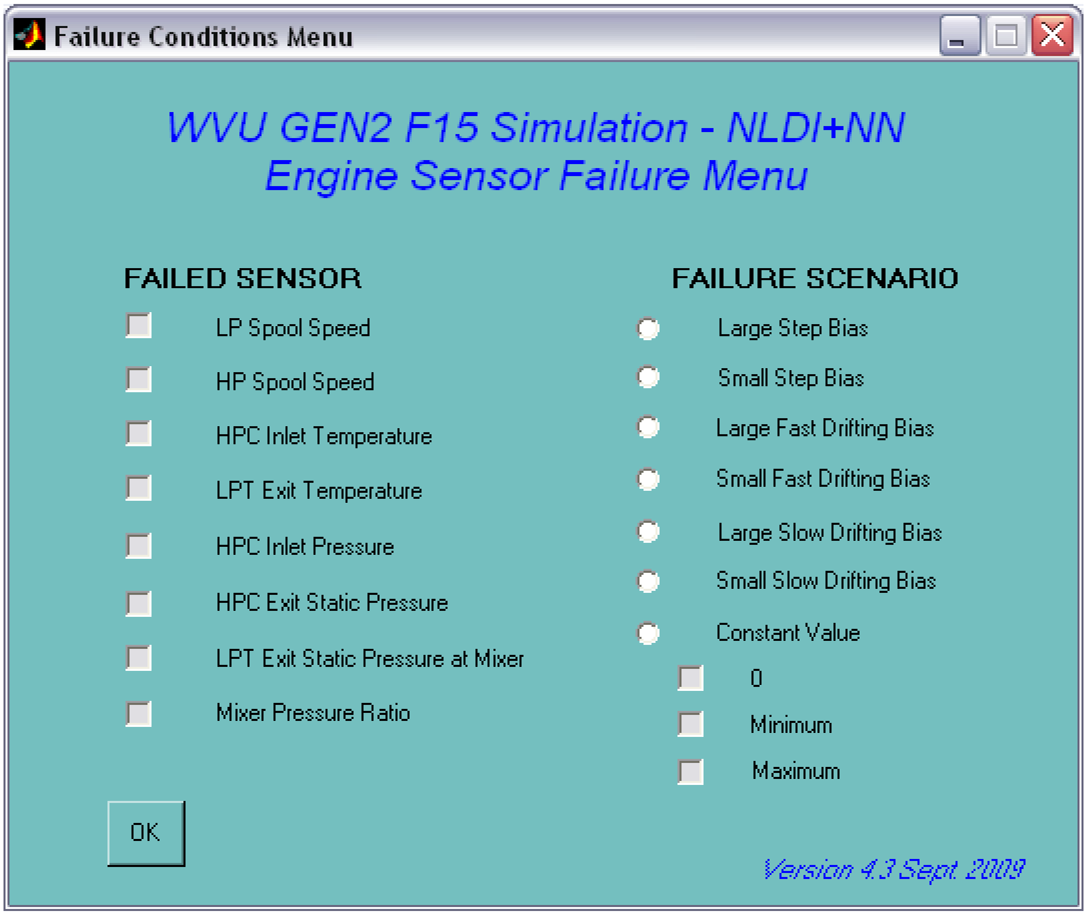
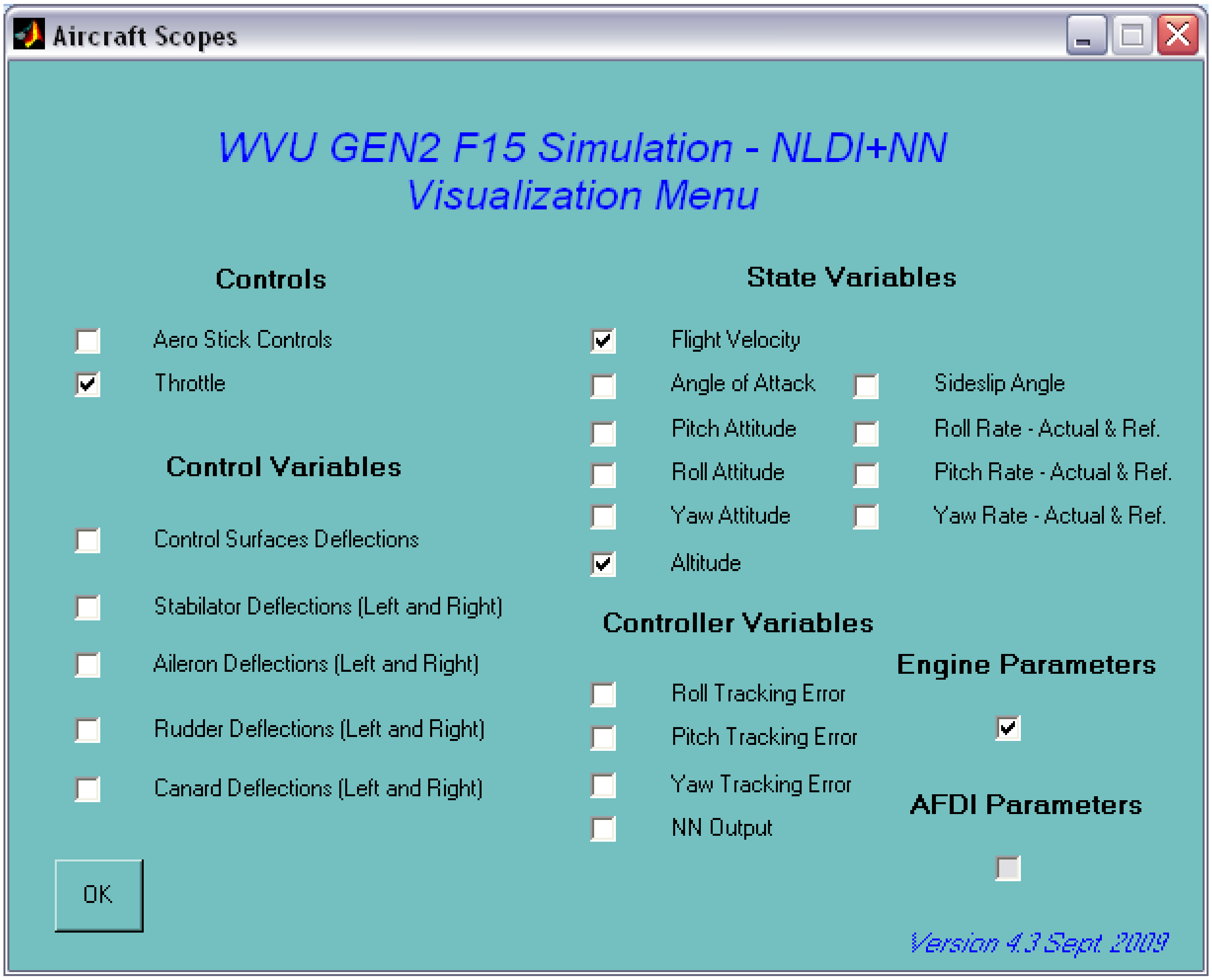


Figure 3.10: Set-up of Engine Sensor Failure

Simulink scopes can be used – as shown in Figure 3.5 – to monitor relevant parameters during simulation and/or to investigate their time histories after the simulation. The menu presented in Figure 3.11 allows the selection of the aircraft states, controls, and other variables to be monitored. The menu presented in Figure 3.12 allows the selection of engine related actuators and sensors, for both engines, to be monitored.



Aircraft Scopes

WVU GEN2 F15 Simulation - NLDI+NN
Visualization Menu

Controls

- □ Aero Stick Controls
- ☑ Throttle

Control Variables

- □ Control Surfaces Deflections
- □ Stabilator Deflections (Left and Right)
- □ Aileron Deflections (Left and Right)
- □ Rudder Deflections (Left and Right)
- □ Canard Deflections (Left and Right)

State Variables

- ☑ Flight Velocity
- □ Angle of Attack
- □ Sideslip Angle
- □ Pitch Attitude
- □ Roll Rate - Actual & Ref.
- □ Roll Attitude
- □ Pitch Rate - Actual & Ref.
- □ Yaw Attitude
- □ Yaw Rate - Actual & Ref.
- ☑ Altitude

Controller Variables

- □ Roll Tracking Error
- □ Pitch Tracking Error
- □ Yaw Tracking Error
- □ NN Output

Engine Parameters

☑

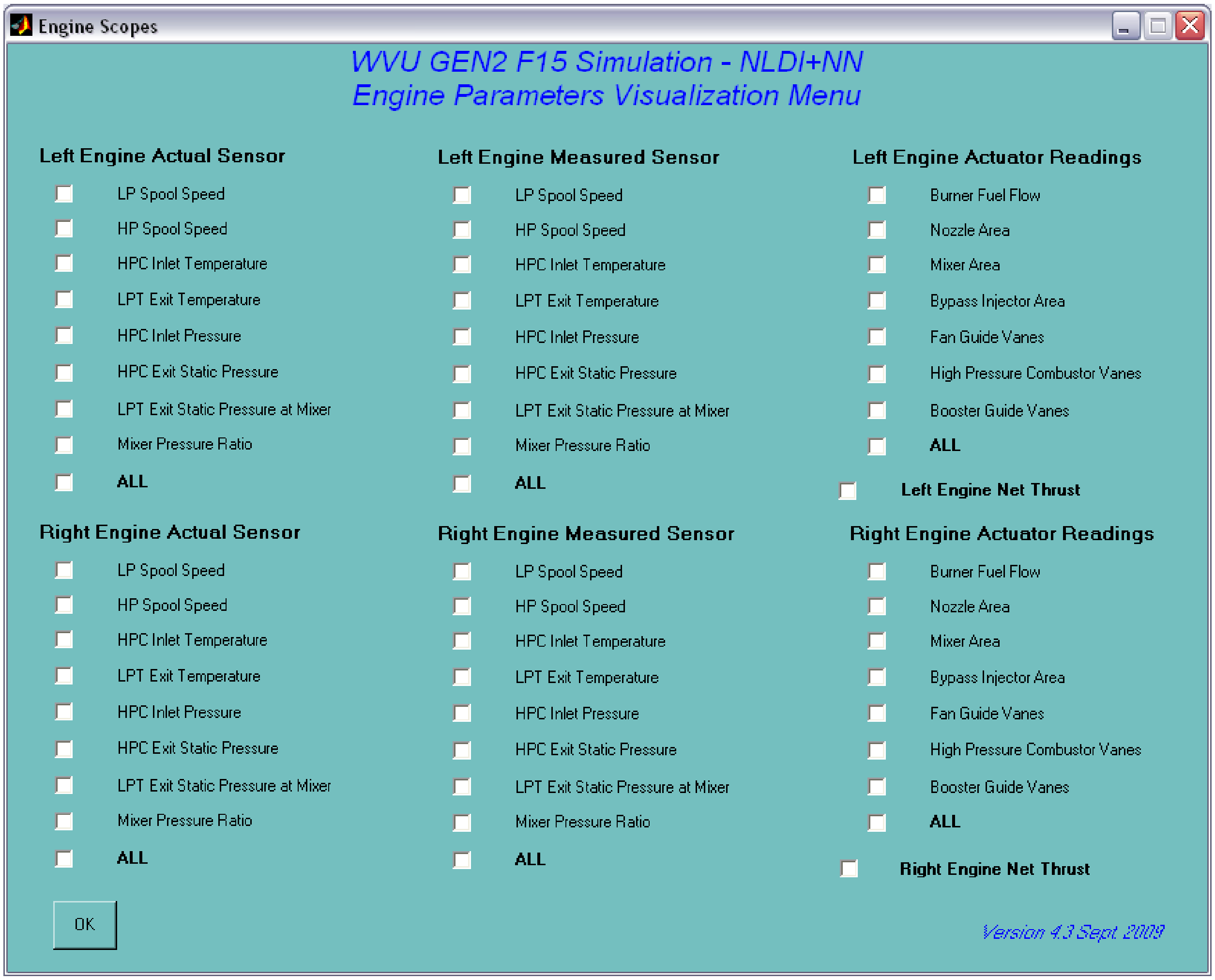
AFDI Parameters

□

OK

Version 4.3 Sept. 2009

Figure 3.11: Visualization Menu – Monitoring General Aircraft Parameters



Engine Scopes

WVU GEN2 F15 Simulation - NLDI+NN
Engine Parameters Visualization Menu

Left Engine Actual Sensor

- □ LP Spool Speed
- □ HP Spool Speed
- □ HPC Inlet Temperature
- □ LPT Exit Temperature
- □ HPC Inlet Pressure
- □ HPC Exit Static Pressure
- □ LPT Exit Static Pressure at Mixer
- □ Mixer Pressure Ratio
- □ **ALL**

Left Engine Measured Sensor

- □ LP Spool Speed
- □ HP Spool Speed
- □ HPC Inlet Temperature
- □ LPT Exit Temperature
- □ HPC Inlet Pressure
- □ HPC Exit Static Pressure
- □ LPT Exit Static Pressure at Mixer
- □ Mixer Pressure Ratio
- □ **ALL**

Left Engine Actuator Readings

- □ Burner Fuel Flow
- □ Nozzle Area
- □ Mixer Area
- □ Bypass Injector Area
- □ Fan Guide Vanes
- □ High Pressure Combustor Vanes
- □ Booster Guide Vanes
- □ **ALL**

□ **Left Engine Net Thrust**

Right Engine Actual Sensor

- □ LP Spool Speed
- □ HP Spool Speed
- □ HPC Inlet Temperature
- □ LPT Exit Temperature
- □ HPC Inlet Pressure
- □ HPC Exit Static Pressure
- □ LPT Exit Static Pressure at Mixer
- □ Mixer Pressure Ratio
- □ **ALL**

Right Engine Measured Sensor

- □ LP Spool Speed
- □ HP Spool Speed
- □ HPC Inlet Temperature
- □ LPT Exit Temperature
- □ HPC Inlet Pressure
- □ HPC Exit Static Pressure
- □ LPT Exit Static Pressure at Mixer
- □ Mixer Pressure Ratio
- □ **ALL**

Right Engine Actuator Readings

- □ Burner Fuel Flow
- □ Nozzle Area
- □ Mixer Area
- □ Bypass Injector Area
- □ Fan Guide Vanes
- □ High Pressure Combustor Vanes
- □ Booster Guide Vanes
- □ **ALL**

□ **Right Engine Net Thrust**

OK

Version 4.3 Sept 2009

Figure 3.12: Visualization Menu – Monitoring Engine Actuators and Sensors

Chapter 4: Engine Model

4.1 Jet Engine Model

The engine model developed is designed to integrate with the WVU F-15 Aircraft model; therefore, a generic military type jet engine was chosen. Most jet fighter engines are low/medium bypass turbofans with a mixed exhaust, afterburner, and variable area final nozzle. Since low bypass turbofans are more effective around Mach of 0.75 and the airspeed of the flight envelope is centered about this Mach, a low-bypass turbofan is the right selection. The engine modeled is a gas powered, high-pressure ratio, dual spool, low by-pass, turbofan aircraft engine with a digital controller. The engine components consist of a fan, high-pressure compressor (HPC), burner, booster, high-pressure turbine (HPT), low-pressure turbine (LPT), bypass duct, mixer, afterburner, and nozzle. These components are shown on the jet engine diagram in Figure 4.1.

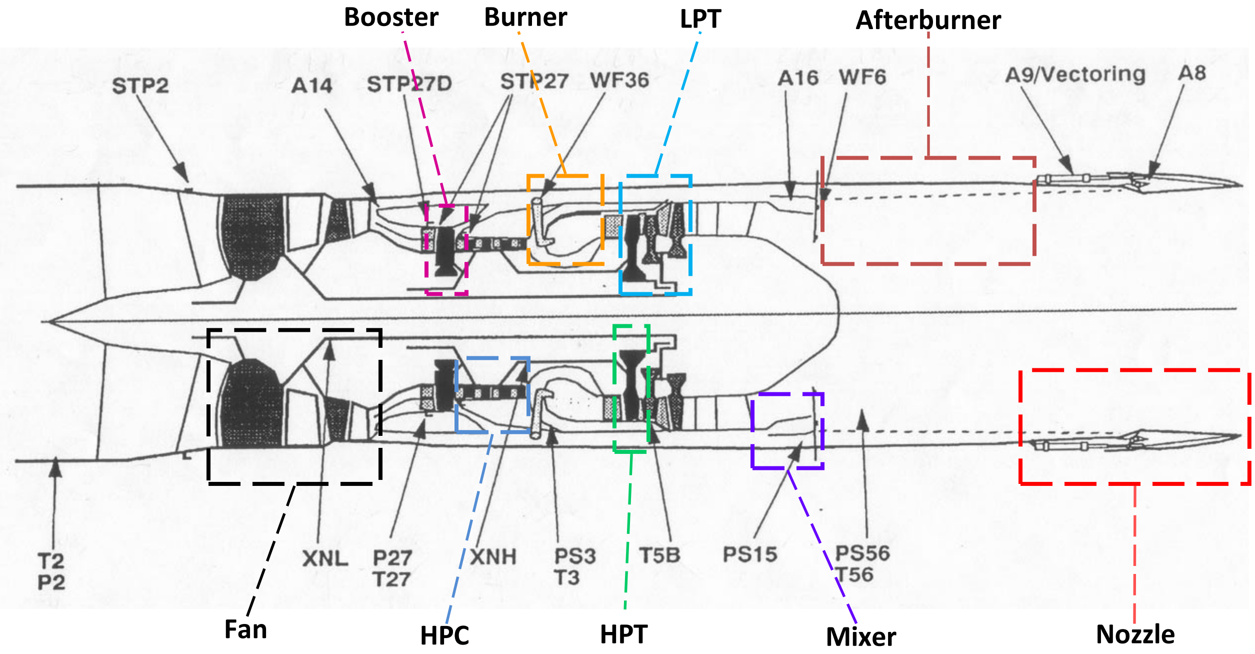


Figure 4.1: Diagram of Aircraft Jet Engine [5]

Also included in Figure 4.1 are the locations where measurements are taken (i.e. areas, temperatures, etc.) The parts of the diagram that are of main focus include 7 actuators and 8 sensors listed in Table 4.1 and Table 4.2, respectively.

Table 4.1: Actuator Name and Location for Aircraft Jet Engine

Actuator	Location on Figure 2.1
Main Burner Fuel Flow	WF36
Variable Nozzle Area	A8
Variable Mixer Area	A16
BP Injector Area	A14
Fan Guide Vanes	STP2
HPC Guide Vanes	STP27
Booster Guide Vanes	STP27D

Table 4.2: Sensor Name and Location for Aircraft Jet Engine

Sensor	Location on Figure 2.1
LP Spool Speed	XNL
HP Spool Speed	XNH
HPC Inlet Temperature	T27
HPC Inlet Pressure	P27
HPC Exit Static Pressure	PS3
LPT Exit Temperature	T56
LPT Exit Static Pressure	PS56
Mixer Pressure Ratio	N/A (P16/P56)

Another way to look at the system components is shown in Figure 4.2. (Flow chart from FDI Part II) This figure also includes the actuator and sensor locations.

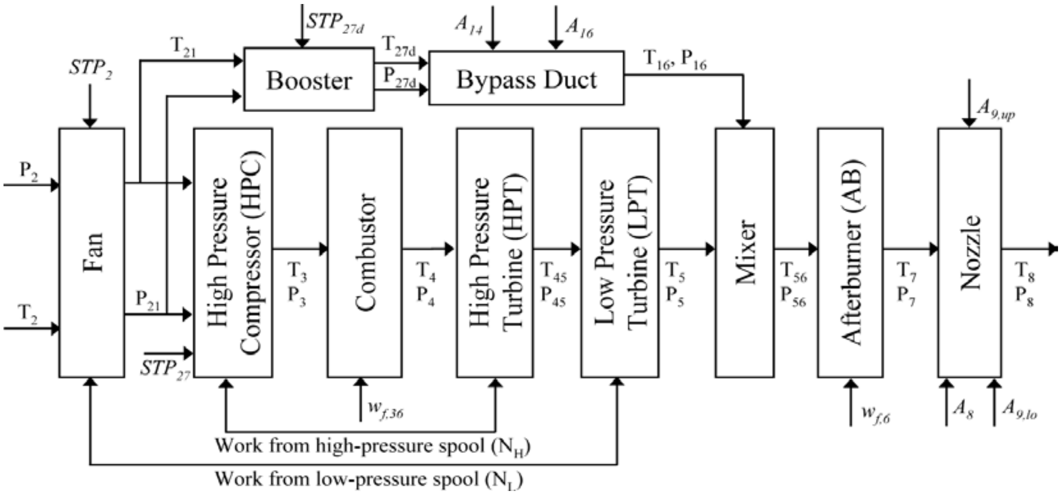


Figure 4.2: Flow Chart of the Different Component Interactions of the Aircraft Jet Engine [5]

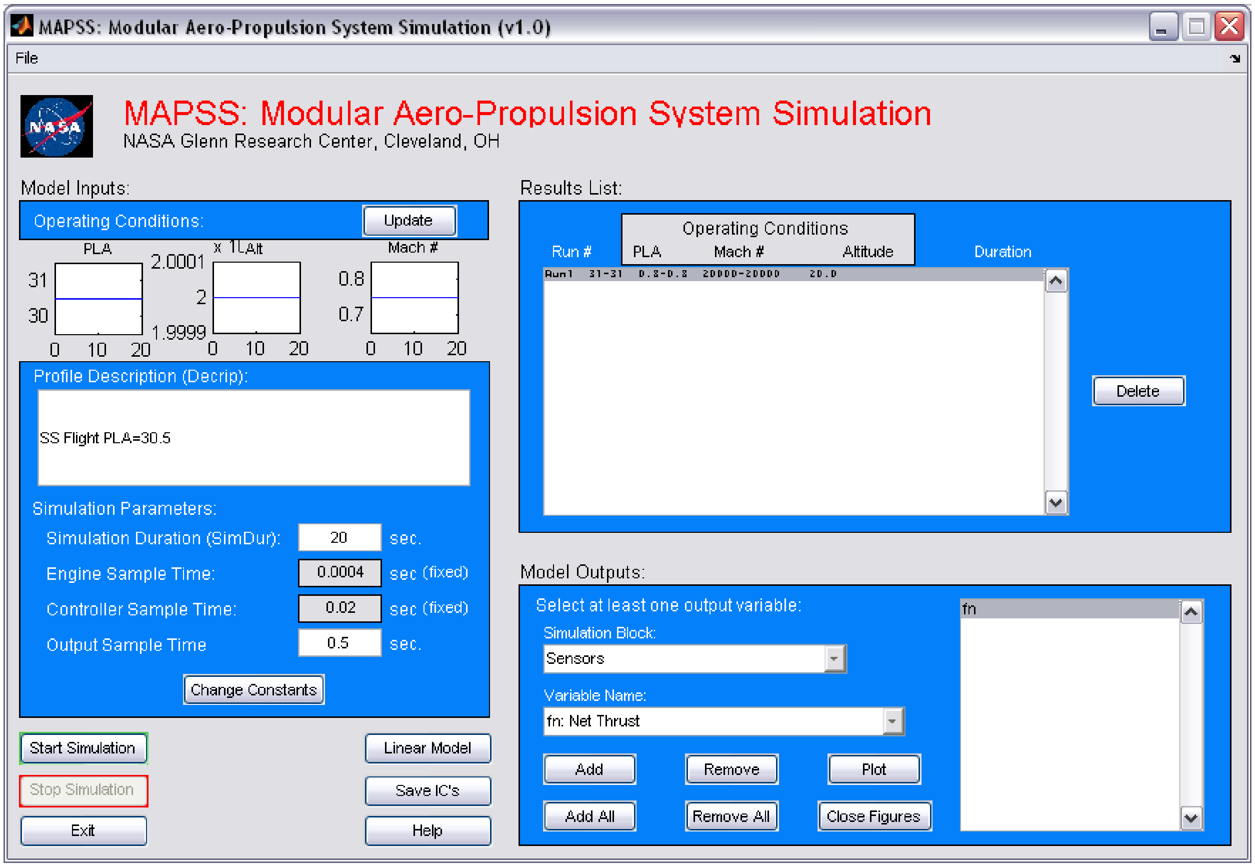
Figure 4.2 illustrates that the first engine component is the fan. The fan and booster are powered by the low-pressure turbine (LPT). One shaft called the low-pressure (LP) spool connects these components and the LP spool speed sensor measures its speed in revolutions per minute, or RPM's. The fan supplies air to the engine core as well as the bypass (BP) duct. The air in the bypass duct mixes with the low pressure turbine exhaust before flowing through the mixed flow nozzle. The high-pressure compressor (HPC) is driven by the high-pressure turbine (HPT). The speed of the connecting high-pressure (HP) shaft is measured as the HP spool speed in RPM's.

The low bypass ratio turbofan has a multi-stage fan which develops at relatively high pressure ratio and yields a high exhaust velocity. This low bypass ratio military type engine has variable inlet guide vanes, which directs air onto the first rotor stage and improves the fan surge margin in the mid-flow range [17].

An important part of the engine is the afterburner, which is a combustor located between the turbine blades and the nozzle. The afterburner has its own specific fuel injectors. The temperature of the exhaust gases increases significantly when the afterburner is lit creating higher exhaust velocity and thrust. To accommodate this extra flow, the variable nozzle area must increase. Since the afterburner uses a lot of fuel to create the thrust increase, it cannot be used at all times [17].

4.2 Linear Engine Model

Using a nonlinear model for this project to include the main internal engine sub-systems was not feasible. MAPSS is a non-real time model and as such cannot be interfaced with the motion-based simulator. Besides, most of the internal structure of the model is not accessible to a user for failure/malfunction modeling. Instead, a linear model was obtained using the MAPSS program and was interfaced with the WVU F-15 model. MAPSS first makes profiles of the operating conditions including the inputs (PLA, Mach, and altitude) versus time. The user decides the profiles to use. Here, the input values are constant over time. However, in order to get to the input values for the linear model, the values had to increase over the profiles. Figure 4.3 shows the GUI for MAPSS including the input profiles.



MAPSS: Modular Aero-Propulsion System Simulation

NASA Glenn Research Center, Cleveland, OH

Profile Description (Decrip): SS Flight PLA=30.5

Simulation Duration (SimDur): 20 sec.

Engine Sample Time: 0.0004 sec (fixed)

Controller Sample Time: 0.02 sec (fixed)

Output Sample Time 0.5 sec.

Run #	Operating Conditions PLA	Mach #	Altitude	Duration
Run1	31-31	0.8-0.8	20000-20000	20.0

Simulation Block: Sensors

Variable Name: fn: Net Thrust

fn

Figure 4.3: Main Menu for the Modular Aero-Propulsion System Simulation [6]

To find the input values some trial and error was needed. The original aircraft engine model was used to find the thrust value at Mach of 0.75 and altitude of 20000ft. This value was 26200N for both engines combined, which is equivalent to 13100N or 2945lb$_f$. Now, the same Mach and altitude are run in the MAPSS model with varying PLA values. The PLA is updated until the thrust output of MAPSS matches the thrust output of the aircraft model. This was also verified using a linear equation to find PLA. MAPSS was run with two PLA values and constant Mach and altitude to find the net thrust, f_n, at each point. For a PLA input of 26.0deg, Mach of 0.75, and altitude of 20000ft, the net thrust is 1745lb$_f$. For a PLA input of 30.0deg, Mach of 0.75 and altitude of 20000ft, the net thrust is 2800lb$_f$. Now the slope can be found from the equation

$$m = \frac{PLA_2 - PLA_1}{f_{n2} - f_{n1}}$$

Eq 4-1

$$m = \frac{30.0 - 26.0}{2800 - 1745} = \frac{4}{1055}$$

which can be used in the linear equation

$$PLA_1 = mf_{n_1} + b \qquad \textbf{Eq 4-2}$$

Now, b is found to be

$$b = 26.0 - \frac{4}{1055}1745 \qquad \textbf{Eq 4-3}$$
$$b = 19.38$$

Finally the linear equation to calculate PLA is

$$PLA = \frac{4}{1055}f_n + 19.38 \qquad \textbf{Eq 4-4}$$

With the desired thrust of 2945lb$_f$ used, the PLA input should be 30.55deg. However, MAPSS only recognizes up to the tenths; therefore, 30.5deg is used as the linearization point. The thrust output from MAPSS is shown in Figure 4.4. The PLA value of 30.5deg reaches a steady-state thrust of 2931.5lb$_f$. Now the engine model is linearized around Mach of 0.75, altitude of 20000ft, and PLA of 30.5deg.

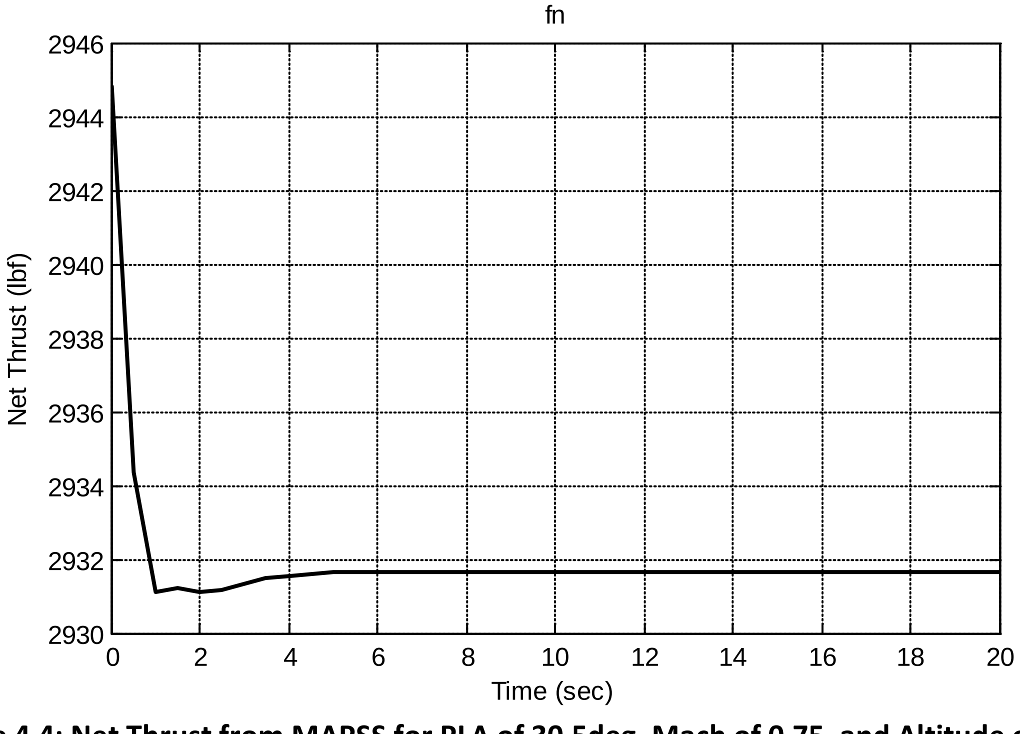


Figure 4.4: Net Thrust from MAPSS for PLA of 30.5deg, Mach of 0.75, and Altitude of 20000ft

Once the linearization is complete, MAPSS provides matrices A, B, C, and D needed for a state-space model as well as the input, output, and state trim values. The input values, U, are the burner fuel flow, nozzle exit area, and bypass exit area. The states, X, are the low-pressure rotor speed, high-pressure rotor speed, and the average hot section metal temperature, which are determined by the PI control action in the controller. Finally, the output values, Y, are the calculated and sensed outputs from the CLM.

Now, the new engine model is created from the linear model matrices and the main components of the MAPSS model. The Simulink® model of the system in shown in Figure 4.5. The 'Controller and Sensor' block contains the sensor dynamics, atmospheric conditions, and the digital controller. The 'Actuator Dynamics' block calculates the movements of the actuators. The inputs of this system are the PLA, Mach, and altitude from the pilot controller joystick and the main output is the thrust, which continues through the aircraft model. However, the actuator and sensor outputs are also used later.

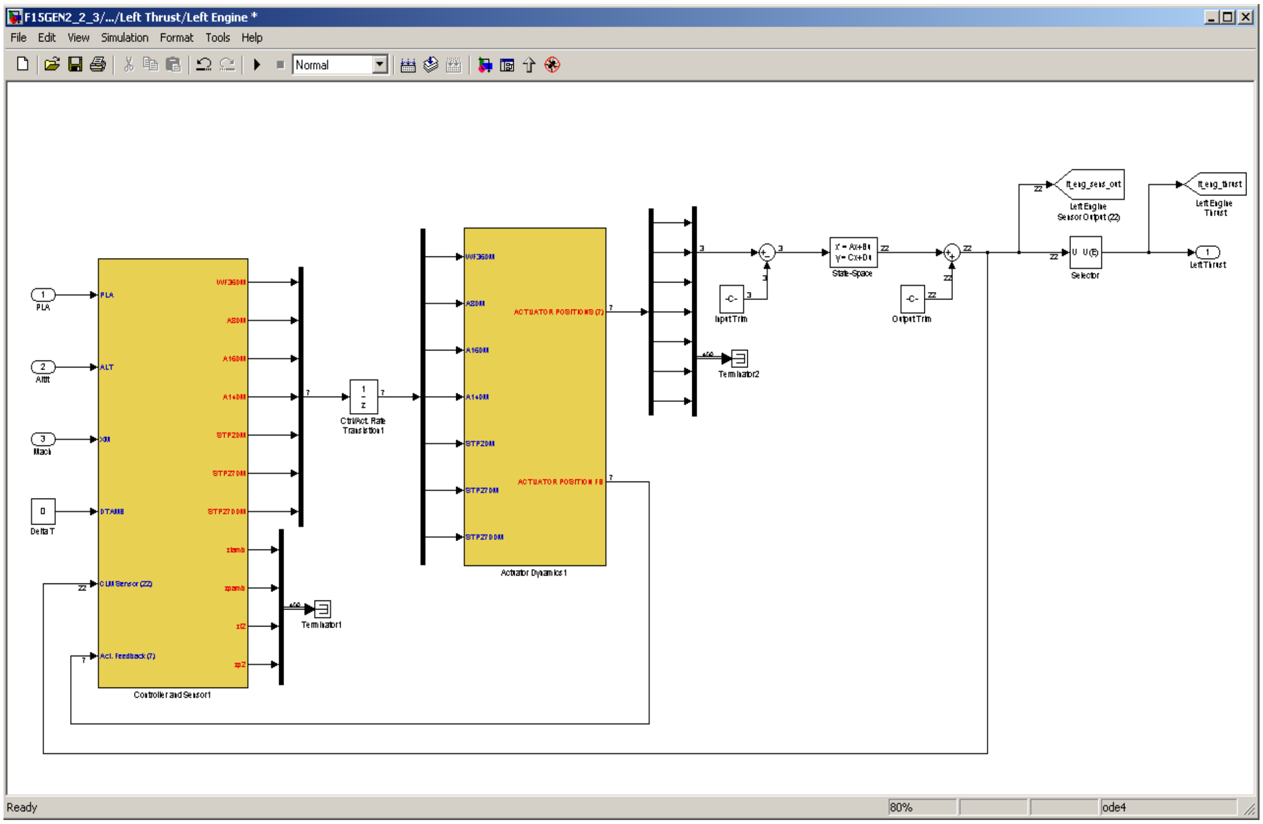


Figure 4.5: Linearized Engine Model in Simulink

Finally, this linear engine model is tested using the same constant input parameters and the MAPSS simulation. Figure 4.6 shows the net thrust output from the linear engine model compared to the thrust output from MAPSS. The steady-state thrust response from the linear model matches the response from MAPSS, which confirms that the linear model works.

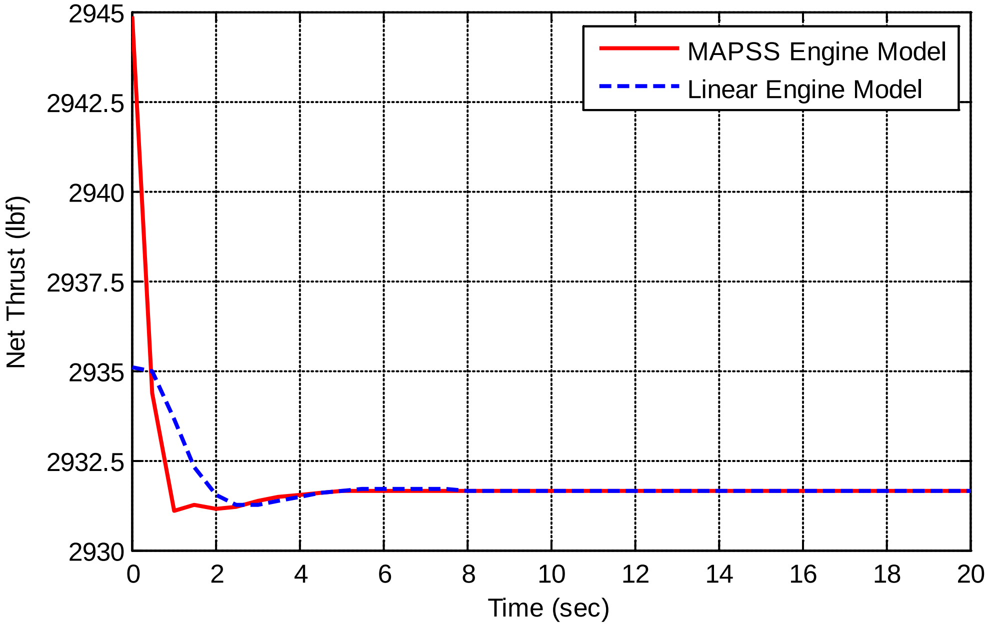


Figure 4.6: Plot of Net Thrust versus Time from MAPSS and Linear Engine Models

Since this linear engine model was created about three specific input points, the model is not valid for all flight conditions. Standard atmospheric conditions apply to altitudes up to 30000ft, where the values for temperature and pressure will begin to change. For these tests, the flight envelope is limited to approximately 30000ft. Also, the PLA levels for the MAPSS system are low, medium, and high [5]. Since the operating point is 30.5, the joystick throttle input must be converted to a linear PLA value. Since 37.5 is the upper limit for the medium power level and testing shows that 21.0 is the minimum option, the PLA input must stay within this range. The PLA value of 30.5 must correspond to zero throttle input; therefore the minimum and maximum PLA input is between 23.5 and 37.5deg, respectively. This means that there is never a true maximum thrust for the system and it doesn't quite react the way the actual engine would.

4.3 Example of Healthy System Simulation

To gain perspective on the dynamics of the engine model, testing was performed to show the response to standard throttle inputs. The output plots of this test show how each parameter reacts to the given input. This shows which parameters are more likely to be affected by the failure inputs later and which may not be affected at all. It is important to note that many of the parameter responses resemble second order responses as opposed to expected first order responses. There was a question as to whether this was due to inaccuracy of the linearized model or the engine modeling of the actuator and sensor dynamics. Additional tests were performed which confirm that the responses are valid for the PLA inputs used.

The first of the plots for nominal conditions is in Figure 4.7, which includes the single engine thrust, velocity, and altitude. As shown in the figure, the thrust output follows closely along with the throttle input. The velocity remains close to constant while the throttle is at 50% and increases or decreases while the throttle is increased or decreased. The altitude also shows a response even though the other pilot controls were held steady. However, the change is within about plus or minus 5 ft so it is only minor.

The next plot, Figure 4.8, contains the response of four of the seven actuators, which consist of the burner fuel flow, the nozzle throat area, the aft VABI (mixer) area, and the bypass injector area. As the plot shows, the burner fuel flow follows the path of the throttle deflection with some overshoot. This is because the throttle controls the fuel to the burner in order to increase or decrease the thrust. As explained in Chapter 2, an increase in thrust corresponds to an increase in the volume of flow through the engine, and an increase in the nozzle area is required to compensate. Therefore, the nozzle area also follows the path of the throttle input. Next, the mixer area has peaks that correspond to the changes in the throttle but remains semi-constant while the throttle is constant, despite the magnitude. Also shown is that the area decreases or increases as the throttle change is positive or negative, respectively. The bypass injector area shown as the last subplot in Figure 4.8 remains constant throughout the test. This is due to the fact that the bypass area is an either all or nothing deal. Once the PLA input triggers the area to open the area is 150.8. This area will remain until the PLA decreases enough to trigger the area to close [5]. Due to initial conditions, the area will always begin at the open position. Now, say the threshold to trigger opening is 30deg and the threshold to trigger closing is 25deg. Once the area is open, the PLA will have a decrease drastically to reach the value to close. At that point the area will remain closed until the open threshold is met again.

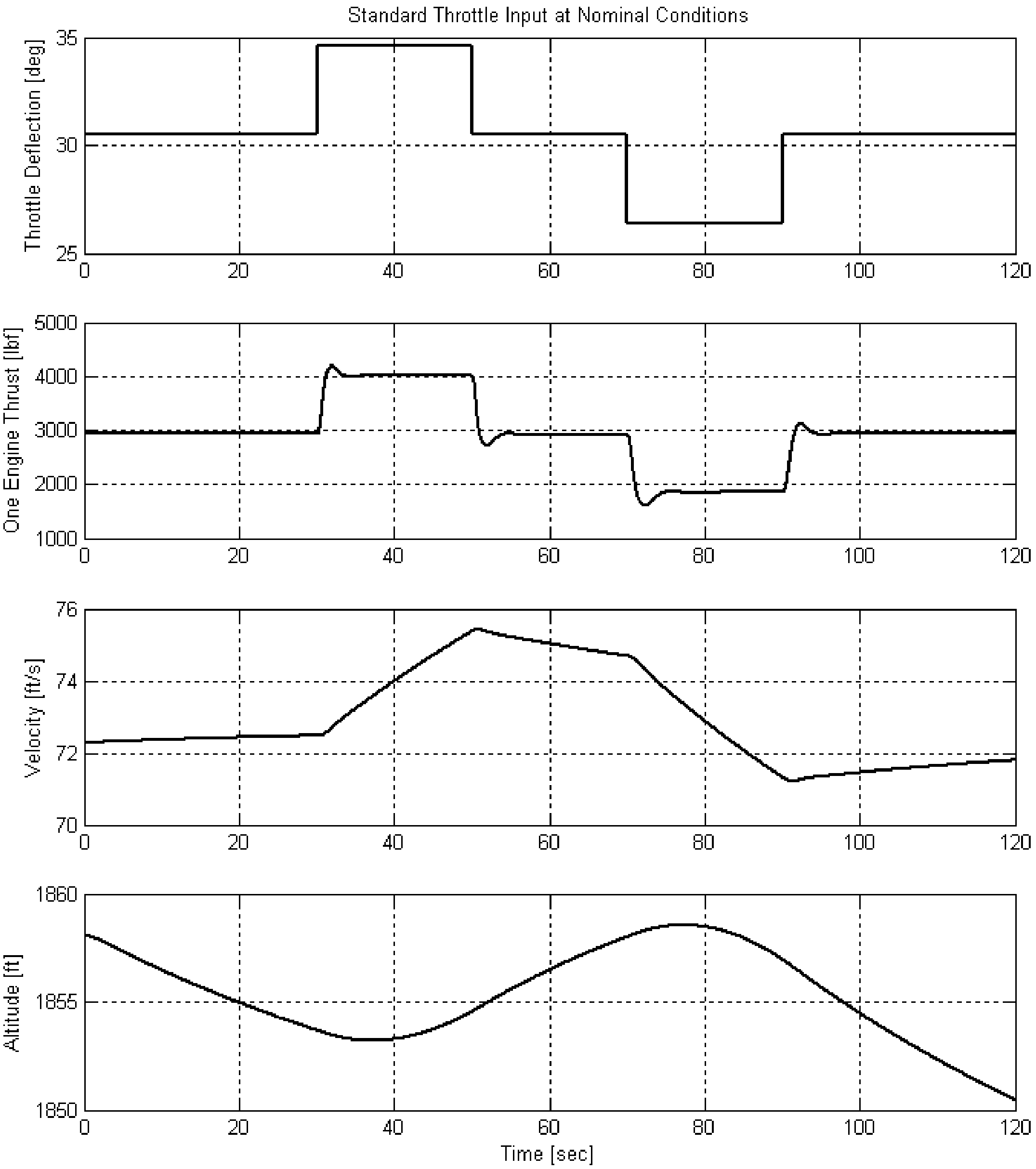


Figure 4.7 Standard Throttle Input Response at Nominal Conditions (1)

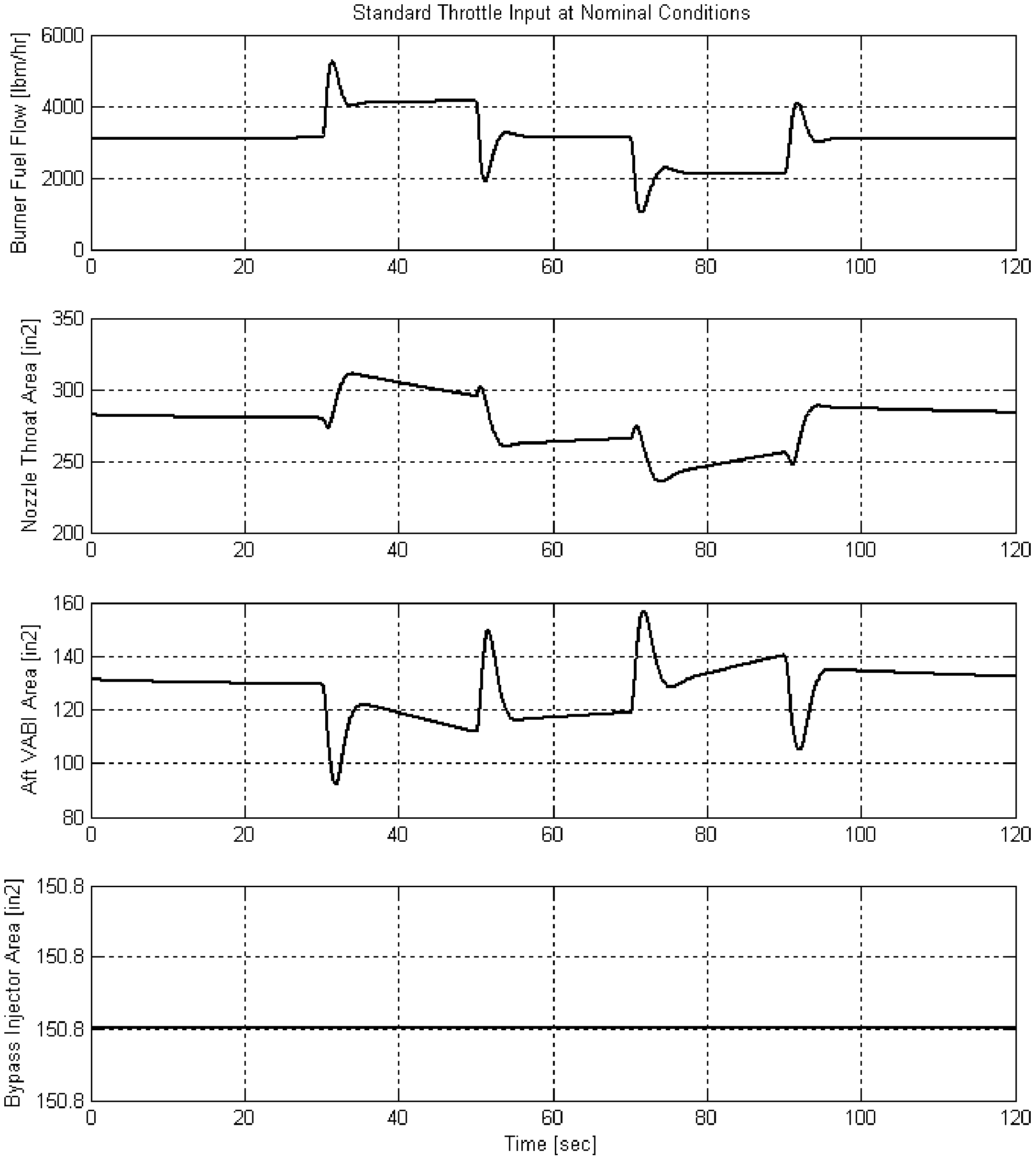


Figure 4.8: Standard Throttle Input Response at Nominal Conditions (2)

The next figure contains the response for the remaining actuators, including the fan stator vanes, the HPC stator vanes, and the booster stator vanes. All of these guide vanes follow the opposite profile from the throttle input, whereas the angle of each actuator decreases as the throttle increases, remains mostly constant as the throttle does, and increases as the throttle decrease.

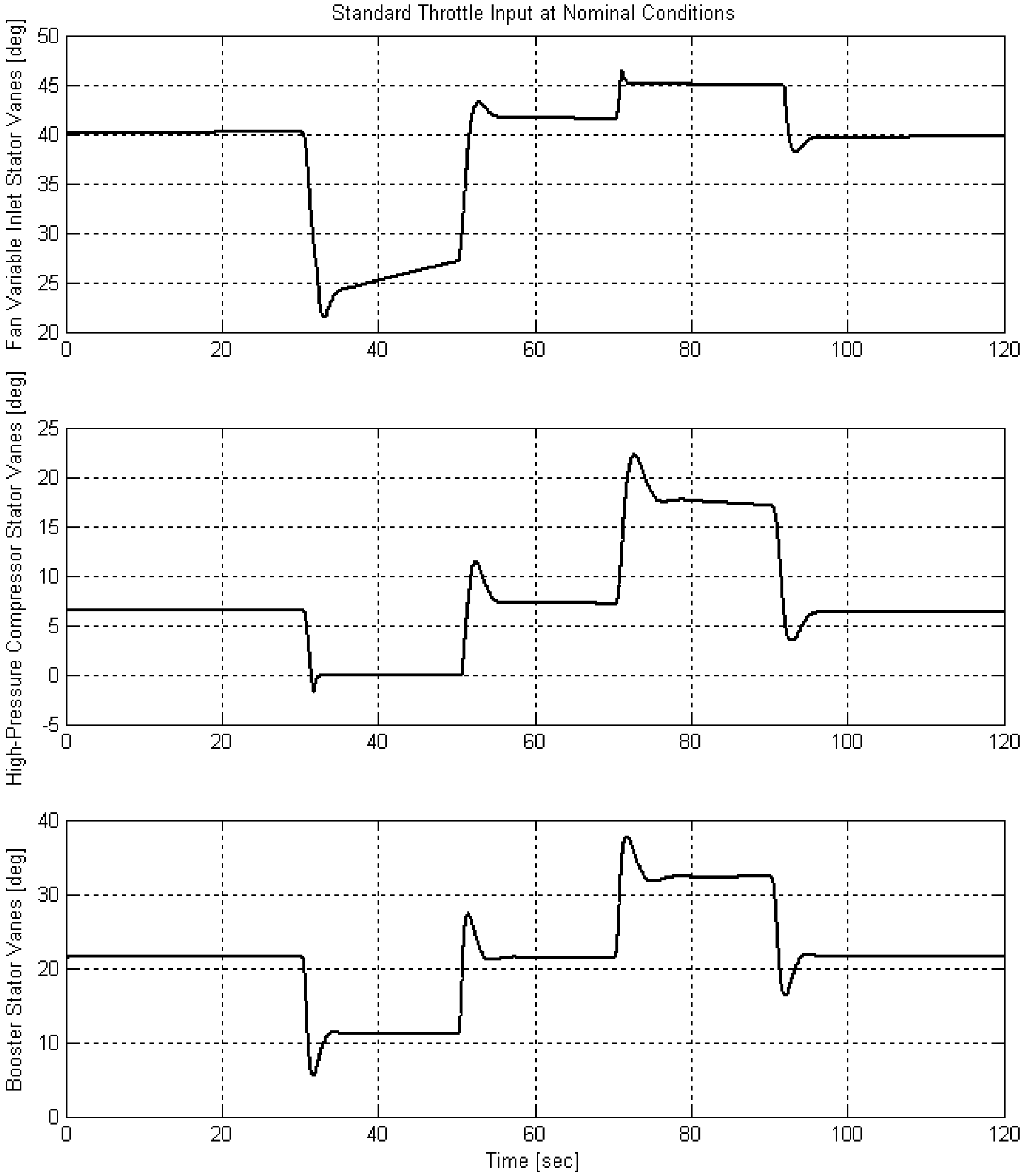


Figure 4.9: Standard Throttle Input Response at Nominal Conditions (3)

Figure 4.10 displays the response of the LP spool speed, HP spool speed, HPC inlet temperature, and LPT exit temperature sensors. The profiles of the LP spool speed, HP spool speed and HPC inlet temperature all have the same profile as that throttle input. The spool speeds logically follow the path of the throttle since they drive the compressor and turbines. Increased spool speed will increase air flow to increase thrust and vice versa. The LPT exit temperature, however, remains fairly constant with spikes corresponding to the throttle changes.

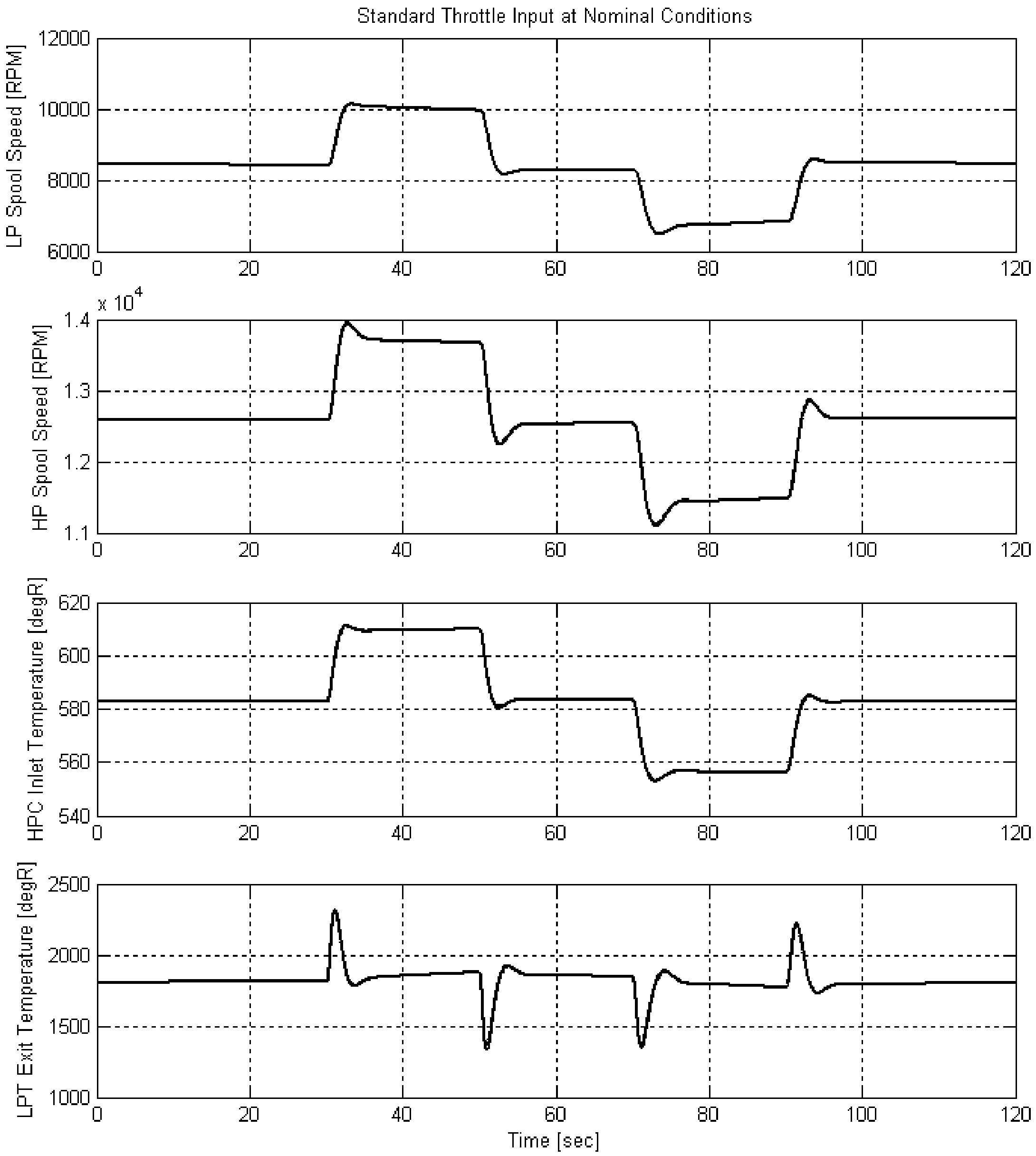


Figure 4.10: Standard Throttle Input Response at Nominal Conditions (4)

The final plot for healthy conditions is Figure 4.11 which contains the responses of the sensors pertaining to pressure. Each of the pressure sensors also follows the profile of the throttle input. The pressure in the engine increases and decreases as the throttle does. The bottom subplot is for the mixer pressure ratio response. This sensor remains at approximately 1.05 for all constant throttle inputs and has spikes when the throttle changes. These spikes will initially decrease as the throttle increases and vice versa.

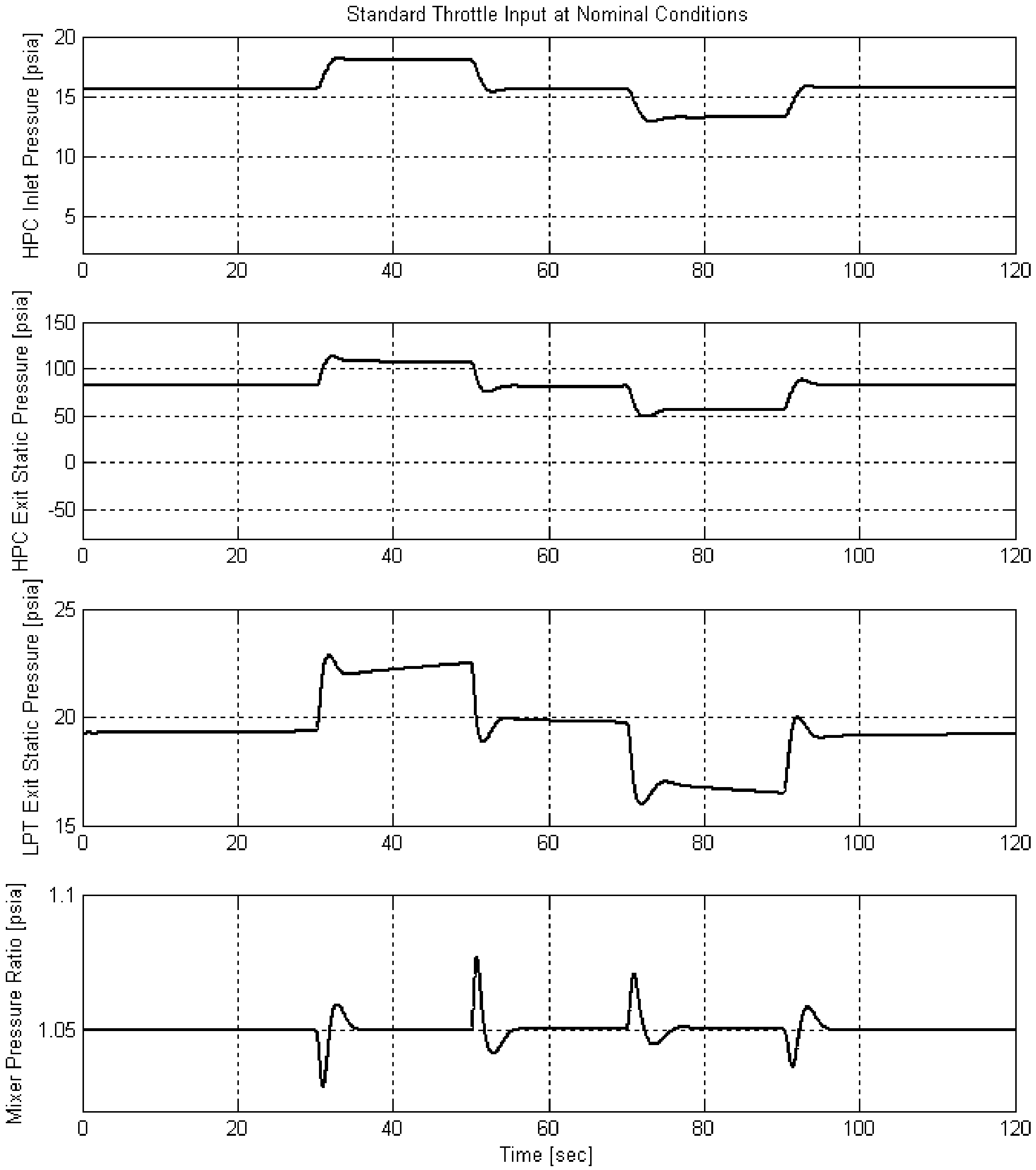


Figure 4.11: Standard Throttle Input Response at Nominal Conditions (5)

Chapter 5: Engine Failure Model

5.1 Types of Failures

The failures on the engines include stuck throttle, thrust runaway, reduced efficiency, stuck engine actuator, and biased and stuck engine sensor. The failures can be applied one at a time on either the left engine, right engine, or both engines. The time of the failure is inputted into the GUI as well as the parameters needed for the particular failure. Descriptions of each failure follow.

The stuck throttle failure implies normal operation of the engine but no response to power lever actuation. In the case of stuck throttle at current position the throttle $\kappa(\mathrm{t})$ remains constant at the value reached at the moment of failure occurrence $\mathrm{t_f}$:

$$\kappa(t) = \begin{cases} pilot\ input & for \quad t < t_f \\ \kappa(t_f) & for \quad t \geq t_f \end{cases} \qquad \textbf{Eq 5-1}$$

For the stuck at imposed position option, the user provides a throttle value between 0.0 and 1.0 at which the throttle remains constant for $t \geq t_f$.

The thrust runaway failure models a malfunction of the fuel control system, which causes the increase of the fuel flow to maximum and the increase of the thrust as a result. This is modeled by increasing the throttle to maximum with first order dynamics and time constant set-up by the user.

$$\kappa(s) = \begin{cases} pilot\ input, & for \quad t < t_f \\ \dfrac{\kappa_{\max} - \kappa(t_f)}{\tau_{ef} s + 1} + \kappa(t_f), & for \quad t \geq t_f \end{cases} \qquad \textbf{Eq 5-2}$$

The power/thrust reduced control efficiency is modeled by scaling down the throttle input by a constant factor selected by the user.

$$\kappa(t) = \begin{cases} pilot\ input & for \quad t < t_f \\ K_{ef} \cdot (pilot\ input), \quad K_{ef} \leq 1 & for \quad t \geq t_f \end{cases} \qquad \textbf{Eq 5-3}$$

For $\mathrm{K_{ef}} = 0$, total loss of power is simulated.

The first three failures mentioned directly affect the thrust of the engine without involving the other components of the engine model. The engine actuators and sensors failures on the other hand, are somewhat more complex affecting the other actuator and sensors in the engine and along with the thrust.

The actuators used in the failure model include the burner fuel flow, nozzle area, mixer area, BP injector area, fan guide vanes, HPC guide vanes, and booster guide vanes. The options for the failure are an inputted value for the fuel flow (lb$_m$/hr), locked at current or imposed values for the areas (in^2), and locked at current or imposed deflections for the guide vanes (deg). Each actuator block within the engine model contains a failure block. This block continues with normal operation of the engine but with no response to the power lever actuation. For option involving imposed values, the actuator goes to and remains at that constant value for $t \geq t_f$. Similarly, the actuator remains constant at the value reached at the time of failure for the current position option. For example, the area, *A(t),* is

$$A(t) = \begin{cases} pilot\ input & for \quad t < t_f \\ A(t_f) & for \quad t \geq t_f \end{cases} \qquad \textbf{Eq 5-4}$$

The sensors failures were applied to eight of the 22 sensors in the engine model. These particular sensors were chosen because they are involved in the control of the engine model. The sensors included are the LP spool speed, HP spool speed, HPC inlet temperature, LPT exit temperature, HPC inlet pressure, HPC exit static pressure, LPT exit static pressure at mixer and mixer pressure ratio. The options to apply to the sensors are a large or small step bias; large or small fast drifting bias; large or small slow drifting bias; or a constant value of 0, minimum, or maximum.

The minimum and maximum are based on the individual sensor. For the constant value failures, the measured sensor output, $\sigma_m(t)$, goes to and remains at the selected constant value (0, $\sigma_{\max}$, or $\sigma_{\min}$) for $t \geq t_f$. The bias of the sensor output, $\overline{\sigma}_{\mathrm{b}}$ is defined as

$$\overline{\sigma}_b(t) = \begin{cases} 0 & for \quad t < t_f \\ \overline{K}_d(t - t_f) & for \quad t_f \leq t < t_f + \Delta t \\ \sigma_b & for \quad t \geq t_f + \Delta t \end{cases} \qquad \textbf{Eq 5-5}$$

In Eq 5-5, $\overline{K}_d$ is the drifting factor, σ_b is the magnitude of the bias and $\Delta \mathrm{t}$ is the time interval to reach to the bias. The drifting factor is represented by

$$\overline{K}_d(t) = \begin{cases} 0 & for \quad t < t_f \\ K_d & for \quad t \geq t_f \end{cases} \qquad \textbf{Eq 5-6}$$

where K_d is

$$K_d = \frac{\sigma_b}{\Delta t} \qquad \textbf{Eq 5-7}$$

For the case of a step bias sensor failure, $\overline{K}_d$ and $\Delta \mathrm{t}$ are 0, and $\overline{\sigma}_{\mathrm{b}}$ is simply 0 or σ_b. The general formula to represent the measured sensor output is

$$\sigma_m(t) = \max\left(\min\left(\sigma_a(t) + \overline{\sigma}_b(t), \sigma_{\max}\right), \sigma_{\min}\right) \qquad \textbf{Eq 5-8}$$

where σ_a is the actual sensor value, and $\sigma_{\min}$ and $\sigma_{\max}$ are the thresholds of the sensor.

5.2 Example of Unhealthy System Simulation

With the engine failure models in place, it is important to see how each one affects the nominal flight conditions. Using the same standard throttle input as for the healthy data, the response of the healthy engine versus the unhealthy engine is plotted. Since there are so many parameters to plot, only one example is discussed in this section. The failure responses of the other failures can be found in Appendix A. In this section, failure of the burner fuel flow valve is displayed and discussed. For the following plots, a left engine burner fuel flow failure at 0lb$_m$/hr is injected into the simulation at 5sec. This is a strong failure; therefore, the effects on the parameters are large and easy to see. Since this is such a strong failure, however, the engine model cannot properly model all of the parameters, so the values will go to the thresholds of the model.

In Figure 5.1 through Figure 5.5, the failed left engine is represented by a solid line while the healthy right engine is shown as a dotted line. Figure 5.1 has the results for the burner fuel flow, throttle input, and thrust output. While the right engine follows the throttle input as it did before, the left engine quickly reaches 0lb$_m$/hr for the burner fuel flow and approximately -700lb$_f$ for the thrust. The thrust in this case is negative due to the engine creating only drag after the failure, since a fuel flow failure at 0lb$_m$/hr essentially turns off the engine. The rest of the parameters in Figure 5.2 through Figure 5.5 react in a similar way, as expected. The right, healthy engine carries on following basically the same profiles as before while the actuators and sensors of the left, unhealthy, engine reach a constant values shortly after the failure and remain there throughout the test.

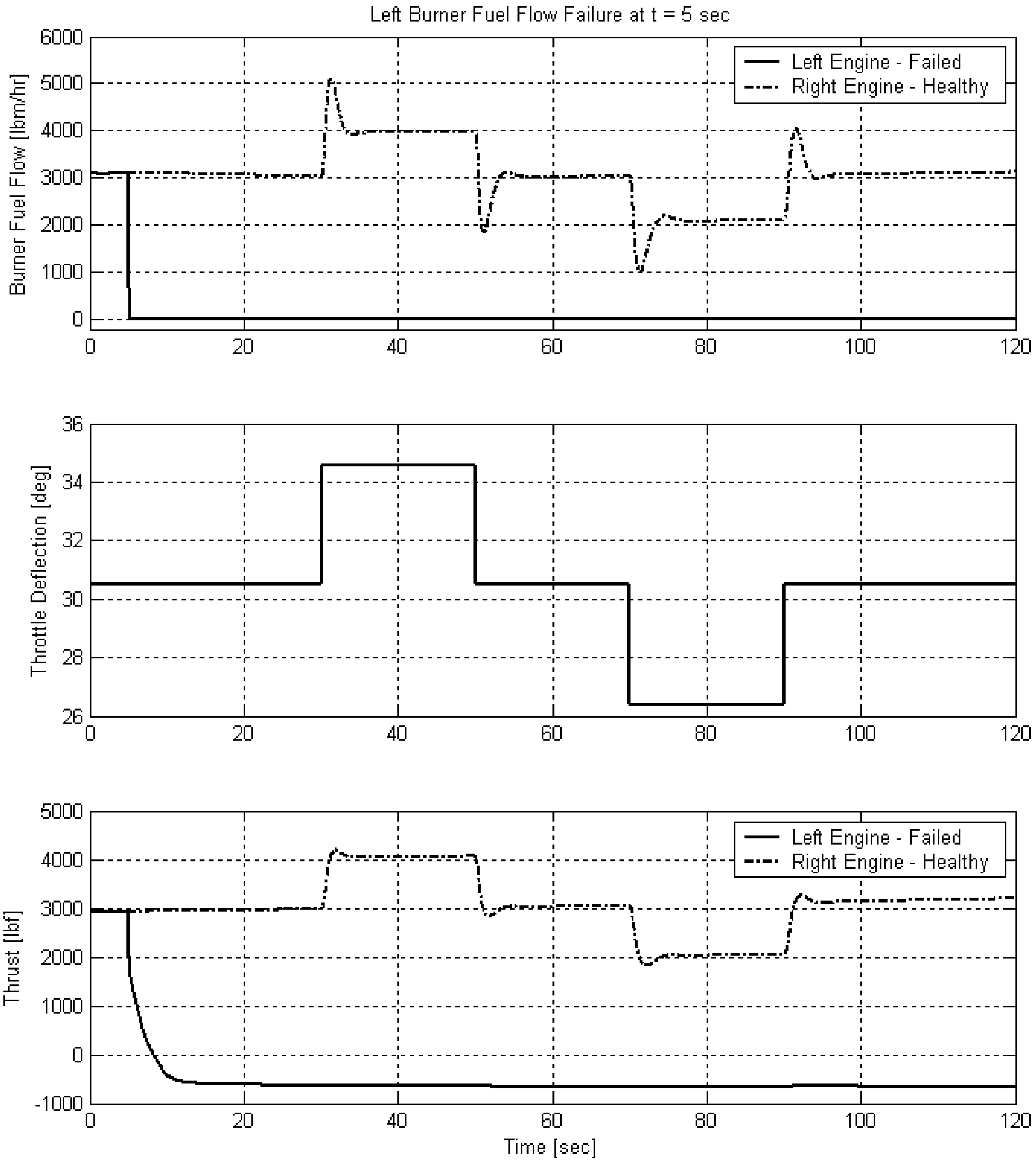


Figure 5.1: Standard Throttle Input Response with a Burner Fuel Flow Failure (1)

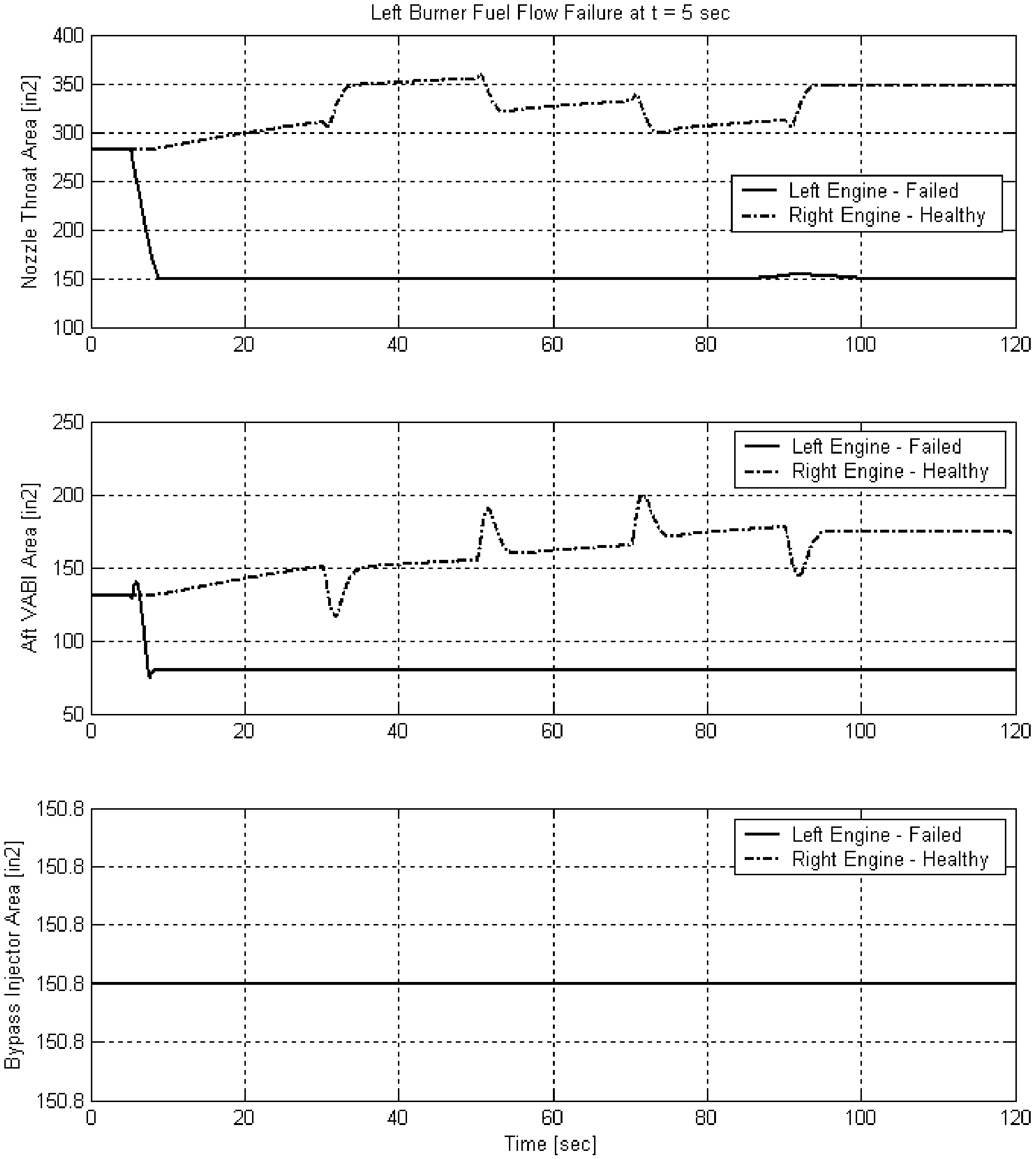


Figure 5.2: Standard Throttle Input Response with a Burner Fuel Flow Failure (2)

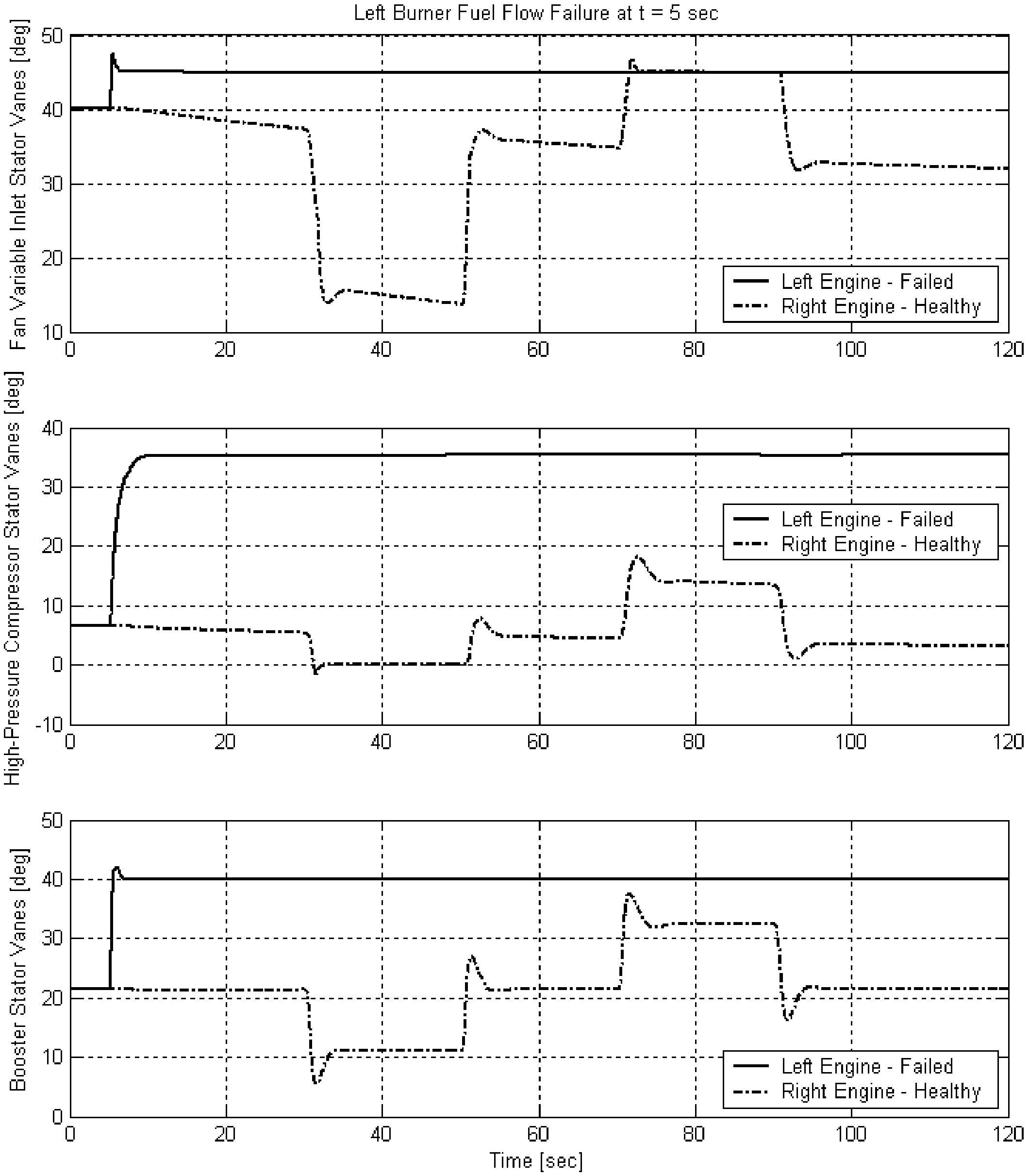


Figure 5.3: Standard Throttle Input Response with a Burner Fuel Flow Failure (3)

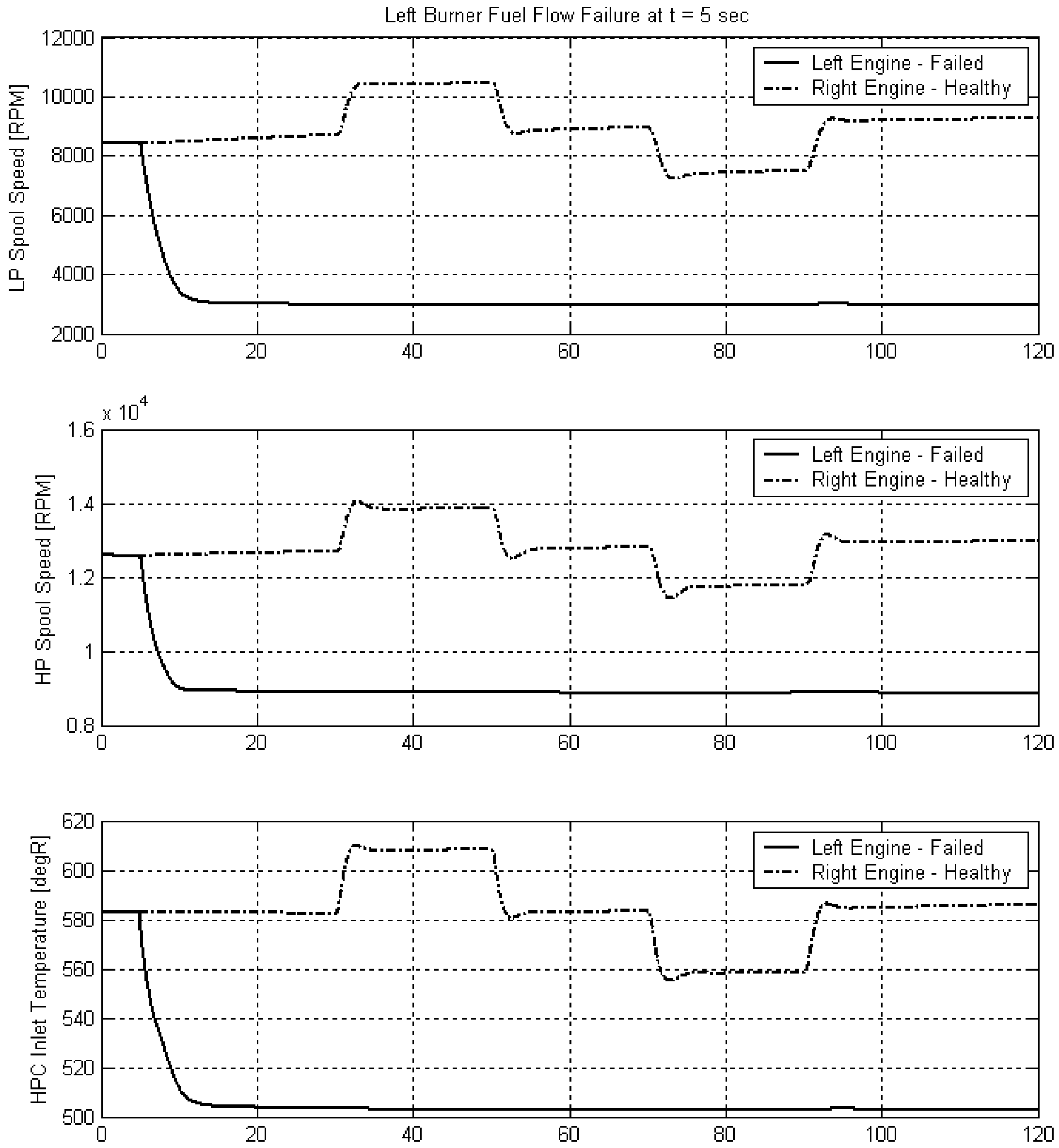


Figure 5.4: Standard Throttle Input Response with a Burner Fuel Flow Failure (4)

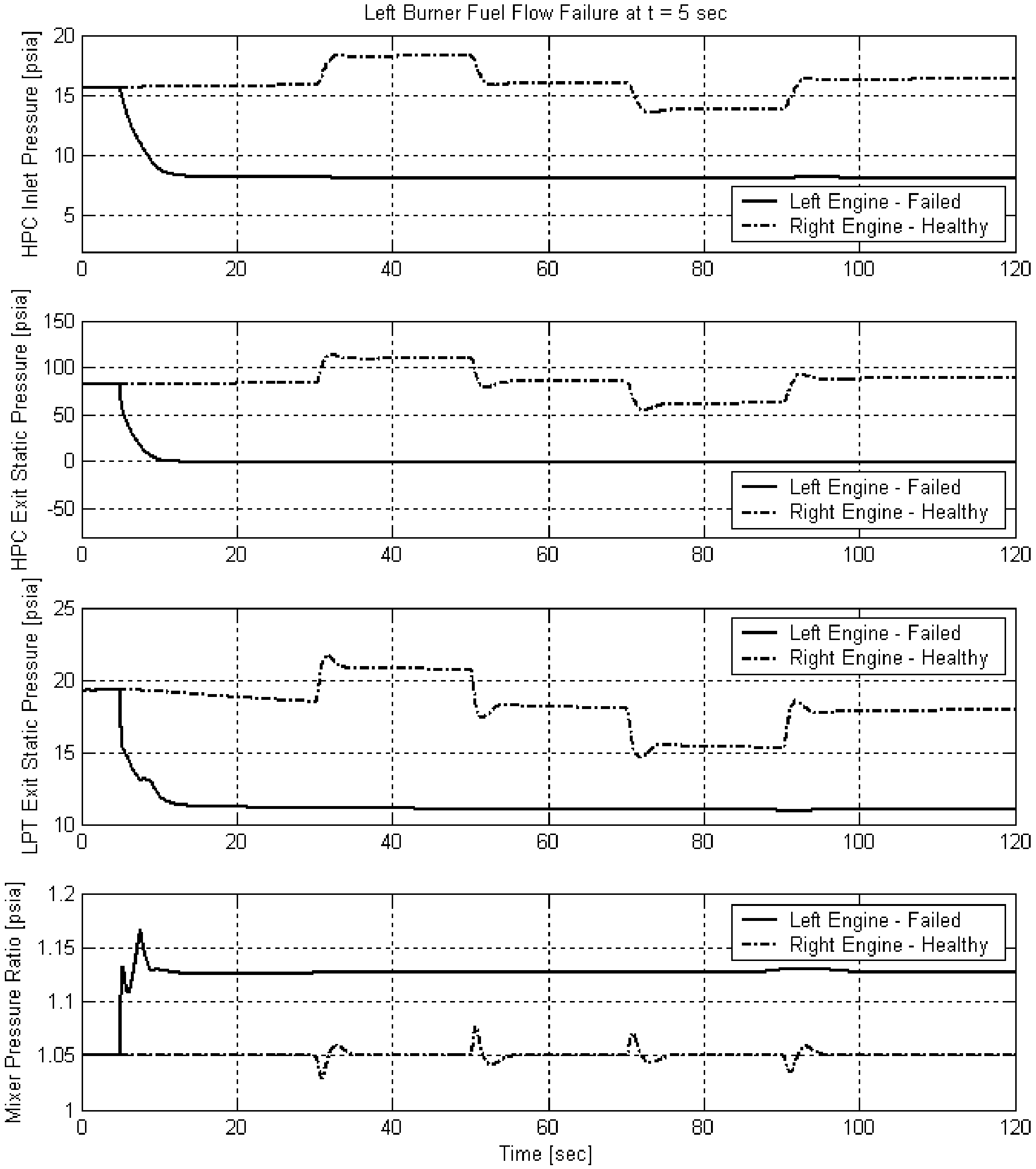


Figure 5.5: Standard Throttle Input Response with a Burner Fuel Flow Failure (5)

Chapter 6: Failure Detection

6.1 Artificial Immune System

Living organisms are endowed with mechanisms that detect intruding pathogens and destroy them to protect the host from diseases. This important function is performed by the biological immune system. In particular, so-called T cells are produced in the thymus [18] that exhibit features that are unlike any cell in the organism. These features are strings of organic chemical compounds, which are produced during a quasi-random cell generation process. Those T-cells who end up having such strings that are similar to the ones of the organism, are destroyed. The ones that are different are allowed to mature and then released into the blood stream. This process is referred to as *negative selection*. If these T-cells encounter another cell that matches their own chemical string, then they have encountered a cell that does not belong to the organism and they mark it for destruction.

The operation of the biological immune system generated the idea to use similar mechanisms for fault detection for engineering systems. If a current configuration of "features" (current measured data that play the role of the intruding cell) matches any configuration (T-cell), referred to as a detector, known NOT to be a normal condition, then an abnormal condition or failure may be declared. These "features" can include various sensor outputs, states estimates, statistical parameters, or any other information expected to be relevant to the behavior of the system referred to as the "self". All regions of the hyperspace defined by the "features" that do not belong to the "self" are referred to as "non-self". Extensive experimental data are necessary to determine the "self" or the region of the hyperspace corresponding to normal conditions. Adequate numerical representations of the self/non-self must be used and the data processed such that they are manageable. The detectors must then be generated and optimized and finally, detection logic must be designed for real time operation with high detection rate and low number of false alarms.

The Artificial Immune System (AIS) - as a new computational paradigm in artificial intelligence – has been applied in recent years to solve a large variety of problems, such as anomaly detection [19, 20], pattern recognition [21, 22], data mining [23], computer security [24, 25], adaptive control [26, 27], and fault detection [29]. The immunity-based fault detection for aircraft has also been investigated [28, 29].

An integrated framework for the detection, identification, and evaluation of a wide variety of sensor, actuator, propulsion, and structural failures/damages using the bio-immune system metaphor has been formulated and implemented at WVU [30]. Interactive computational tools based on evolutionary algorithms have been developed for the generation and optimization of artificial immune system-based failure detectors using the negative selection strategy [31]. The utility performs the design of an AIS in three different phases. A preliminary phase consists of processing data from flight simulation tests for “self” definition through normalization, duplicate removal, and clustering. A first phase of the evolutionary algorithm produces a sub-optimal set of detectors. An iterative algorithm generates detectors that do not overlap with the “self” and achieve a prescribed level of coverage of the “non-self”. A second phase consists of a classic genetic algorithm that optimizes the set of detectors based on three criteria. These criteria are: minimum number of detectors, minimum overlapping between detectors, and maximum coverage of the “non-self”.

The main steps in generating and using AIS-based detectors for engine abnormal condition detection are (see also Figure 6.1):

1. Definition of “features” or “identifiers” that define the operation of the system and are capable of capturing the signature of the failures that must be detected.

2. Acquisition of flight simulation data at nominal conditions over a desired region of the flight envelope for AIS training and validation (determination of false alarms). Acquisition of flight simulation data at abnormal conditions (with failures) for detection scheme testing and evaluation (determination of detection rates).

3. Flight simulation data processing for self definition.

4. Generation of detectors through negative selection.

5. Detection scheme formulation and testing using data at failure conditions.

Note that within this thesis, only failure detection has been addressed.

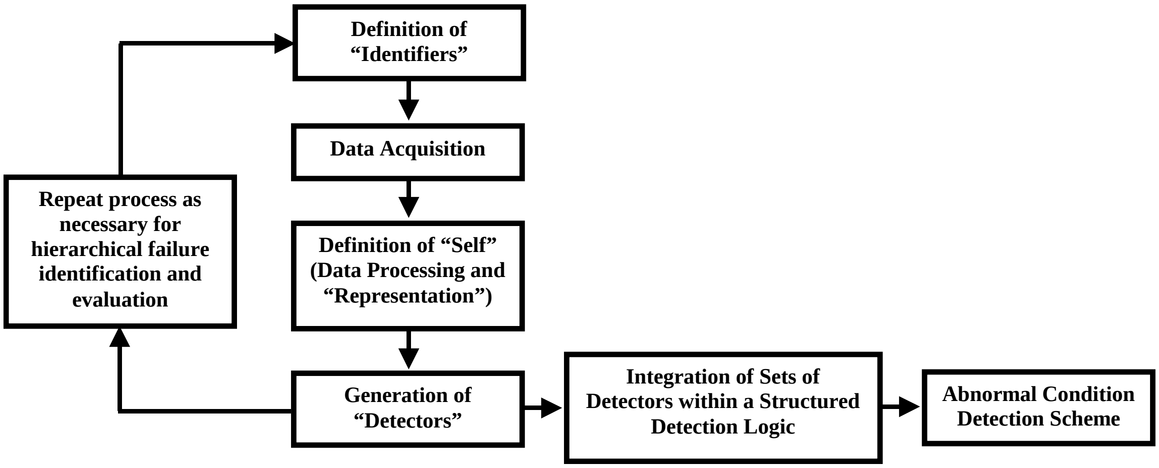

Definition of "Identifiers"
Data Acquisition
Definition of "Self" (Data Processing and "Representation")
Generation of "Detectors"
Integration of Sets of Detectors within a Structured Detection Logic
Abnormal Condition Detection Scheme
Repeat process as necessary for hierarchical failure identification and evaluation

Figure 6.1: Artificial Immune System-Based Abnormal Condition Detection [30]

6.2 Selection of Features for the Definition of Self

The selection of the features for the definition of the self is a critical process for the success of the detection scheme, which will depend on the capability of these selected parameters to capture the dynamic signature of all failures targeted.

A list of candidate features has first been formulated. It included throttle input, thrust, the seven engine actuators and the eight engine sensor outputs available within the engine model. Simulation of failures affecting the throttle, three engine actuators, and three engine sensors has been performed and the effects on candidate features analyzed. These effects were evaluated on a four-level scale, major (**+++**), moderate (++), minor (+), insignificant (0) as shown in Table 6.1. A list of 5 successful candidates was then selected such that the signatures of all failures considered are likely to be captured. The selected features are:

- Throttle Command
- Mixer Area Actuator
- Fan Guide Vane Deflection
- HPC Exit Temperature Sensor
- Mixer Pressure Ratio Sensor

Table 6.1: Effect of Actuator and Sensor Failure on Engine Parameter Measurements

Candidate Feature	Stuck Throttle	Stuck Burner Fuel Flow Valve	Stuck Nozzle Area	Stuck Mixer Area	Faulty LPT Exit Pressure Sensor	Faulty Mixer Pressure Ratio Sensor	Faulty LP Spool Speed Sensor
Thrust	+++	+++	+	+	+	0	+
Throttle	+++	+++	+	+	+	0	+
Burner Fuel Flow	+++	+++	+	+	++	+	+
Nozzle Area	++	+++	++	+	+++	+++	+
Mixer Area	+++	+++	+++	+++	+++	+++	0
Fan Vanes	++	++	++	+	+++	+	+++
HPC vanes	+++	+++	++	+	++	+	+
Booster Vanes	+++	+++	++	+	+++	+	+
LP Spool RPM	+++	+++	+	+	+++	+	+
HP Spool RPM	+++	+++	+	+	++	+	+
HPC Inlet Temperature	+++	+++	+	+	0	+	+
LPT Exit Temperature	++	++	+	0	+++	0	++
HPC Inlet Pressure	+++	+++	+	0	+	+	0
HPC Exit Pressure	+++	+++	+	+	+	+	0
LPT Pressure	+++	+++	++	+	+++	+	+
Mixer Pressure Ratio	++	+++	+++	+++	+++	+++	+++

6.3 Data Processing and Detector Generation

The WVU AIS interactive design environment [31] developed in Matlab was used for the purpose of this research effort. The main portal to the WVU AIS design environment is presented in Figure 6.2.

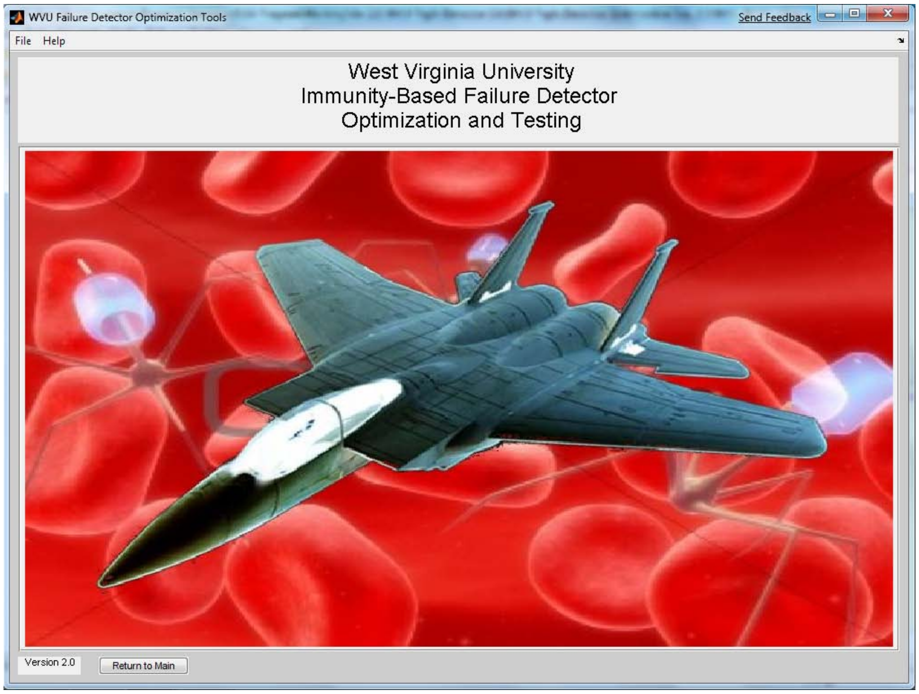
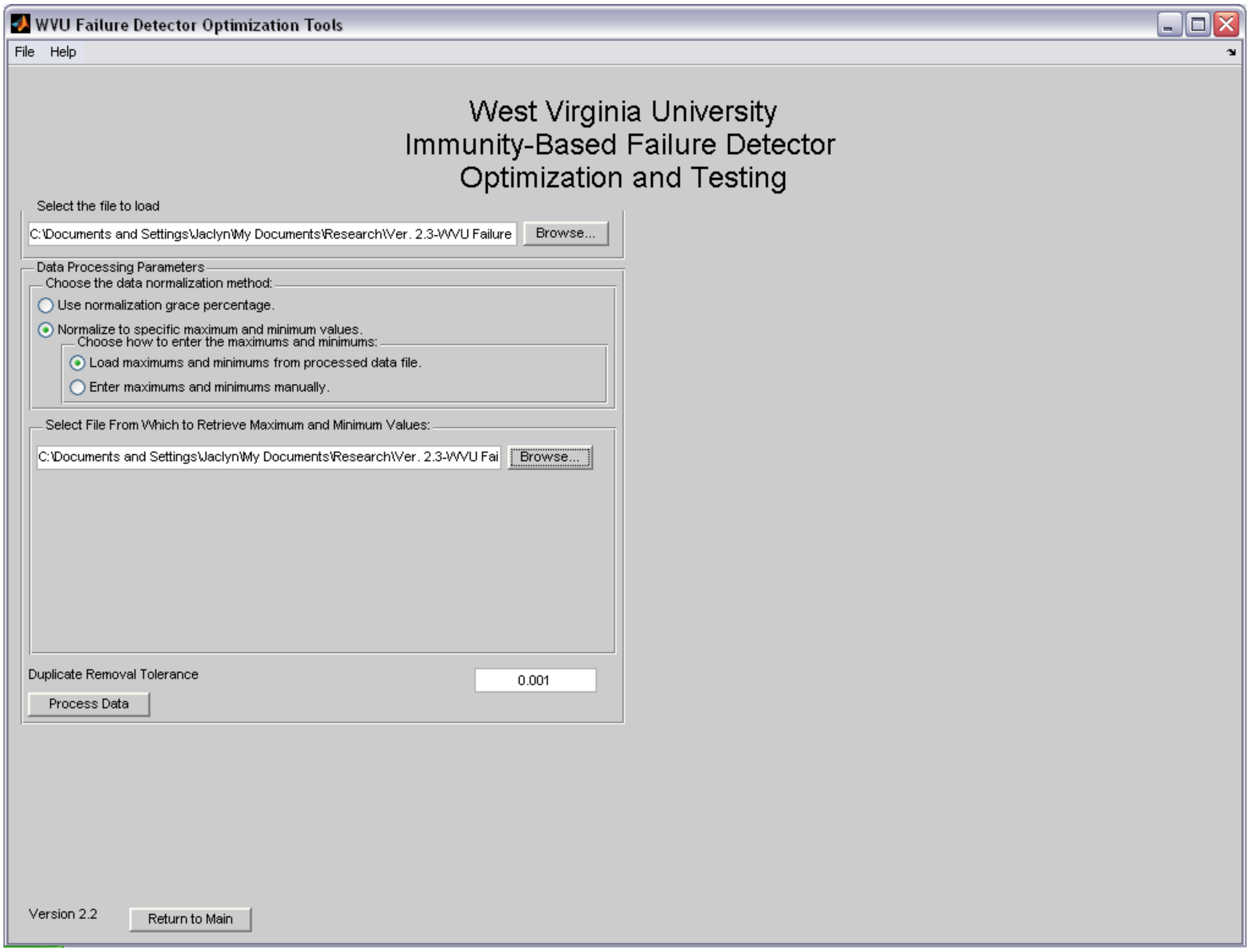


Figure 6.2: Main Menu of the WVU AIS Design Environment [31]

The WVU AIS Design Environment goes through a step-by-step process leading to the detectors, or antibodies, of the system. The first step is to load and process the raw data from the nominal flight tests. In the case of building the self, "processing" the data translates to normalizing and removing the duplicate data points. The duplicate elimination is performed because multiple data at the same point are undesirable and unneeded. Having duplicate data points will unnecessarily increase the computational effort. The menu for this procedure is shown in Figure 6.3. Since several tests at nominal conditions were included in the self, some of the data were merged before processing. However, the processing was not simply performed on the set of all data because the resulting file was too large. In order for the detection generation process to work correctly, all of the different data sets had to be normalized to the same factors. The maximums and minimums were found by merging all of the data together and performing only the normalization. Then the multiple merged data sets were processed individually using these same values so that all of the sets were consistent.



West Virginia University
Immunity-Based Failure Detector
Optimization and Testing

Select the file to load

C:\Documents and Settings\Jaclyn\My Documents\Research\Ver. 2.3-WVU Failure

Data Processing Parameters

Choose the data normalization method:

○ Use normalization grace percentage.

● Normalize to specific maximum and minimum values.

Choose how to enter the maximums and minimums:

● Load maximums and minimums from processed data file.

○ Enter maximums and minimums manually.

Select File From Which to Retrieve Maximum and Minimum Values:

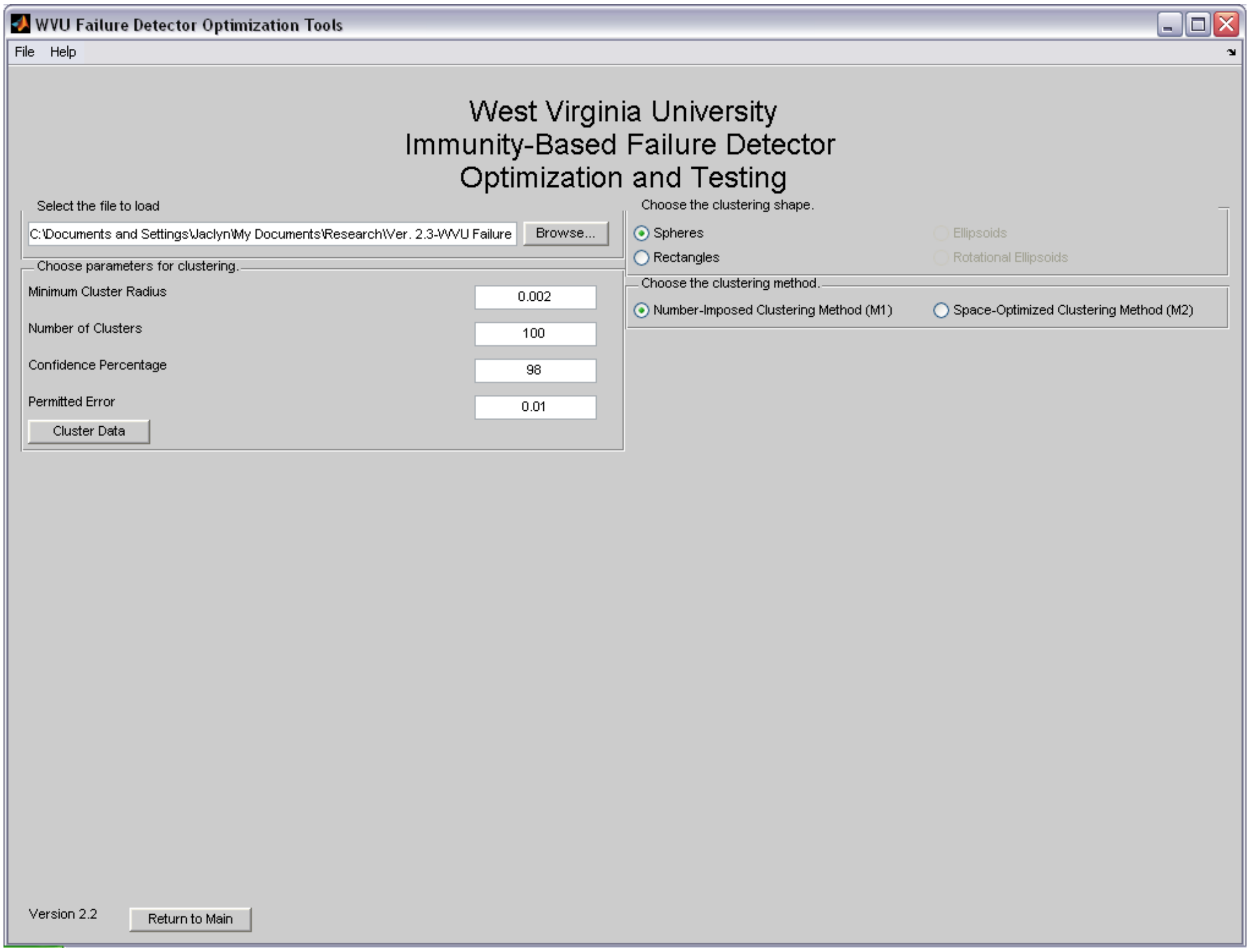
C:\Documents and Settings\Jaclyn\My Documents\Research\Ver. 2.3-WVU Fai

Duplicate Removal Tolerance 0.001

Version 2.2

Figure 6.3: Data Processing Menu

The next step toward creating the self is to cluster the processed data. The menu for this is shown in Figure 6.4. The processed data are loaded and then the user chooses the parameters for clustering, for which the values chosen are based on previous research. Once clusters are generated for all sets of training data, the clusters are merged together. This can also be performed in the AIS design environment.



West Virginia University
Immunity-Based Failure Detector
Optimization and Testing

Select the file to load: C:\Documents and Settings\Jaclyn\My Documents\Research\Ver. 2.3-WVU Failure

Choose parameters for clustering.

Minimum Cluster Radius: 0.002
Number of Clusters: 100
Confidence Percentage: 98
Permitted Error: 0.01

Choose the clustering shape.

● Spheres
○ Rectangles
○ Ellipsoids
○ Rotational Ellipsoids

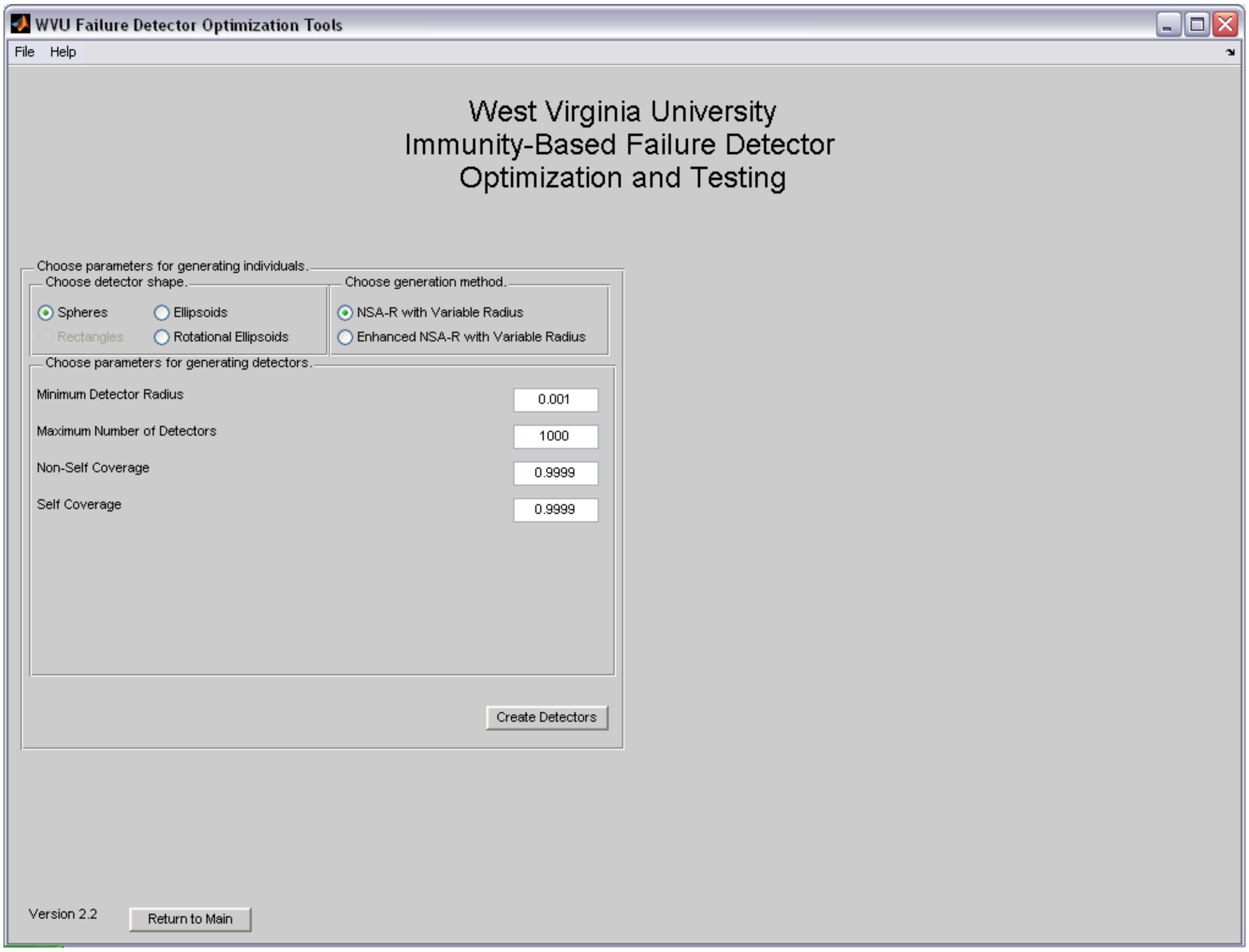
Choose the clustering method.

● Number-Imposed Clustering Method (M1)
○ Space-Optimized Clustering Method (M2)

Version 2.2

Figure 6.4: Clustering Menu

With the final set of clusters created, the total number being about 13600, the detector generation can begin. First, the merged cluster file is loaded into the program. Then, Negative Selection and Create Detectors (Phase I only) are chosen from the Detector Optimization menu. The menu for Phase I is shown in Figure 6.5. Using only Phase I means that detectors are only generated and no optimization is performed. For the antibodies, the shapes were spheres with variable radii. The maximum number of detectors was set to 1000, but the final number of detectors was 991. These detectors can now be validated and applied to the failure tests. This is done using the detector testing menu with the options shown in Figure 6.6.



WVU Failure Detector Optimization Tools

File Help

West Virginia University
Immunity-Based Failure Detector
Optimization and Testing

Choose parameters for generating individuals.

Choose detector shape.

● Spheres ○ Ellipsoids
○ Rectangles ○ Rotational Ellipsoids

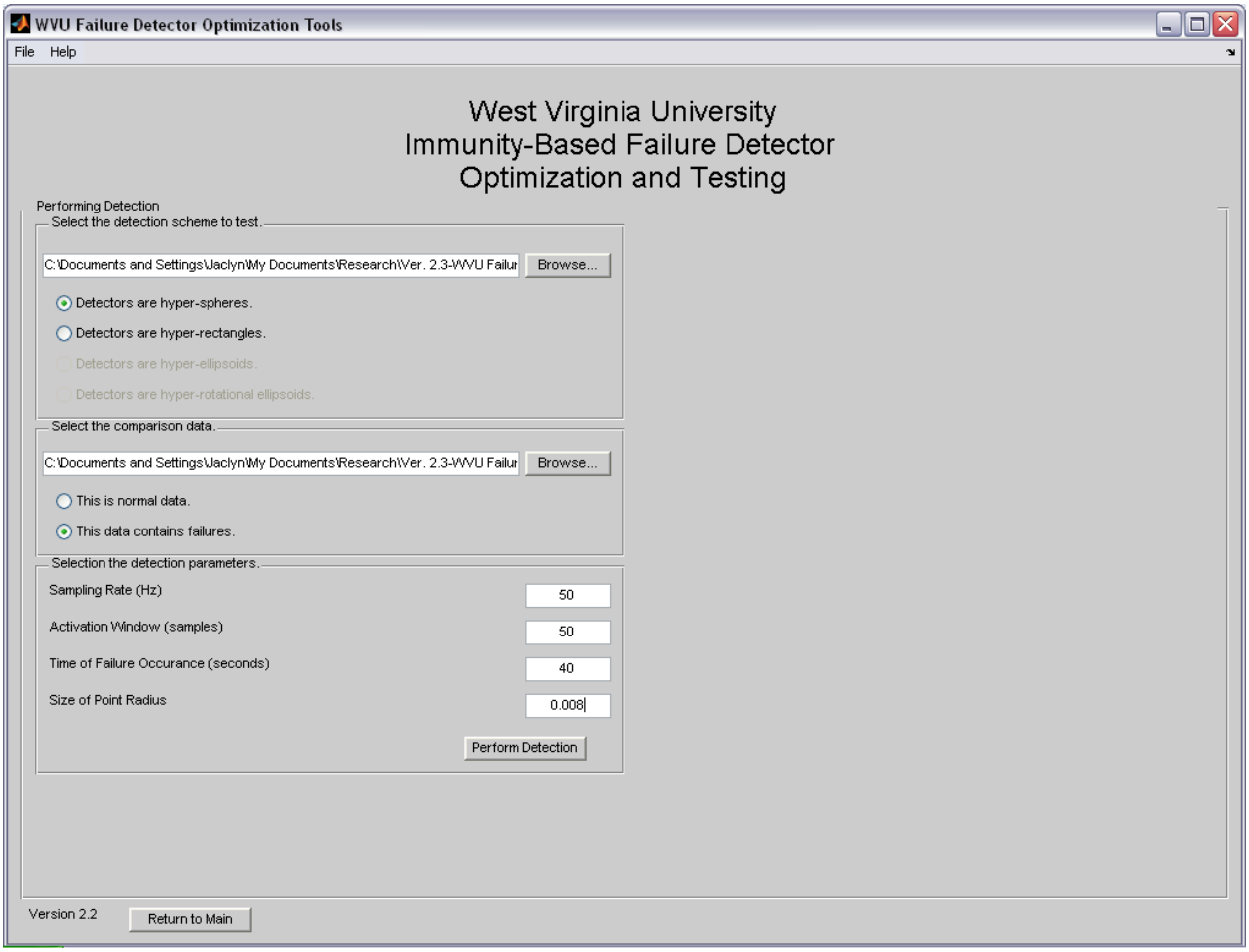
Choose generation method.

● NSA-R with Variable Radius
○ Enhanced NSA-R with Variable Radius

Choose parameters for generating detectors.

Minimum Detector Radius	0.001
Maximum Number of Detectors	1000
Non-Self Coverage	0.9999
Self Coverage	0.9999

Figure 6.5: Detector Generation Menu for Phase I



West Virginia University
Immunity-Based Failure Detector
Optimization and Testing

Performing Detection

Select the detection scheme to test.

C:\Documents and Settings\Jaclyn\My Documents\Research\Ver. 2.3-WVU Failur

● Detectors are hyper-spheres.

○ Detectors are hyper-rectangles.

○ Detectors are hyper-ellipsoids.

○ Detectors are hyper-rotational ellipsoids.

Select the comparison data.

C:\Documents and Settings\Jaclyn\My Documents\Research\Ver. 2.3-WVU Failur

○ This is normal data.

● This data contains failures.

Selection the detection parameters.

Sampling Rate (Hz)	50
Activation Window (samples)	50
Time of Failure Occurance (seconds)	40
Size of Point Radius	0.008

Version 2.2

Figure 6.6: Detection Testing Menu

Chapter 7: Data Acquisition

7.1 Flight Plan

In previous research using the artificial immune system paradigm at WVU [32], the flight data used to train the self included all paths of the flight envelope shown in Figure 7.1. To validate the system, flight tests were performed at points A, B, C, and D and then run through the detection scheme. The results showed that training the AIS for the surrounding flight paths (numbered points) was sufficient to cover the area in between the lines (lettered points) [32].

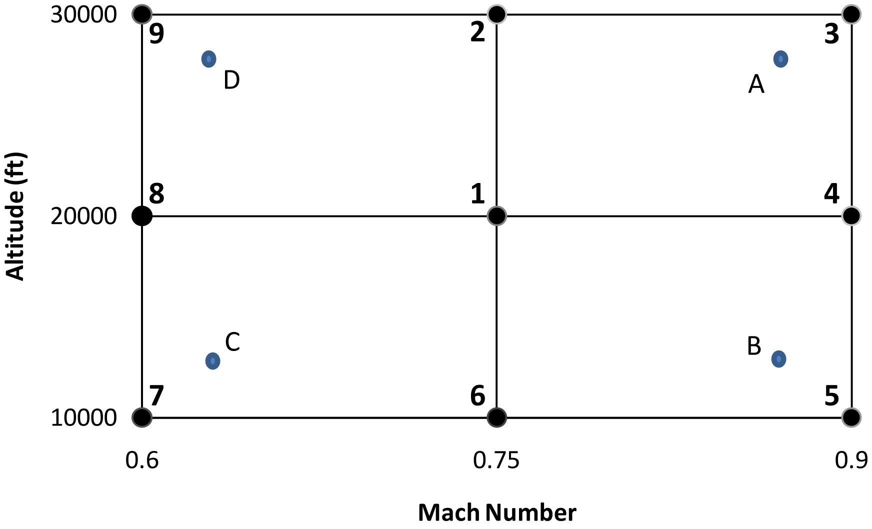


Figure 7.1: F15 Flight Envelope

For this experiment, the research was pushed farther to see if using only the center lines including points one, two, four, six and eight to train the system would be sufficient to create an accurate self. In other words, the pilot would start at point 1, accelerate to point 4, decelerate to point 8, return to point 1, ascend to point 2, descend to point 6, and return to point 1. Typical maneuvers were performed throughout these tests. Actually, the test was broken in two to keep the data files smaller and from having to repeat long tests in case of error.

Preliminary testing showed that training the system on paths one to two to six and one to four to eight did not contain all data needed to sufficiently cover flight from point one to three. Therefore, flight paths one to two to three and one to four to three were included in the training data and resulted in low false alarms. The validation results are farther discussed in Chapter 8.

In every test, the pilot performed three thrust doublets and three yaw doublets at different magnitudes/speeds. Since the engine performance was in question, the most important maneuvers came from the throttle and yaw inputs. The throttle controls the function of the engine while the difference in thrust from the healthy and unhealthy engines affects yaw; therefore, normal function of these maneuvers was needed.

The same process was used to test several engine failures. Each failure was performed along the paths one to two to six and one to two to six including the doublets at each point. All of the failures were performed on the left engine with some failures also performed on the right engine. This showed, as expected, that there was no difference between the two.

7.2 Flight Testing

All flight testing was performed by student and faculty volunteer pilots in the WVU 6 degrees-of-freedom flight simulator. The WVU Motus 600 Flight Simulator, shown in Figure 7.2, was manufactured by Fidelity Flight Simulation, Inc., Pittsburgh, PA. The components of the simulator include

- 6 DOF motion platform driven by electrical induction motors
- Laminar Research X-Plane flight simulation software
- LCD mosaic wall four-monitor external visual display
- Instructor operating station
- Computer and control cabinet

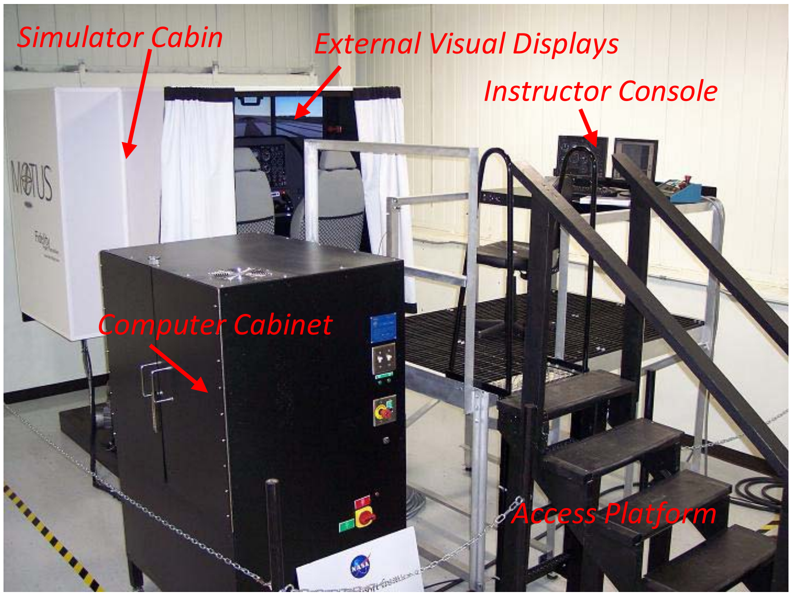


Figure 7.2: WVU 6-DOF Flight Simulator

The motion based platform shown in Figure 7.3 provides six-degrees-of-freedom translational and rotational motion cues. Electrical motors are used to drive the motion base, which represents a very versatile and inexpensive solution to this type of application. Motion drive algorithms convert the motion of the aircraft as resulting from the dynamic model into motion of the platform such that the perception of the pilot is optimized within the physical limitations of the ground based simulator.

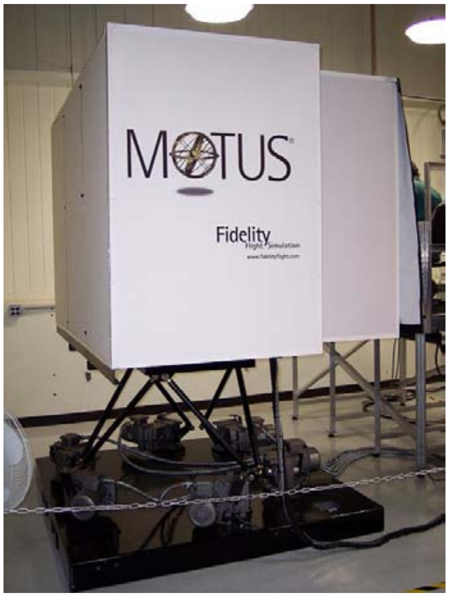


Figure 7.3: WVU 6-DOF Flight Simulator Flight Cabin System

The WVU Flight Simulator works with a commercial aircraft simulation package called X-Plane [13]. X-Plane features high capabilities and flexibility in selecting the simulation scenario. The 2-seat cockpit, shown in Figure 7.4 houses dual controls and instrument clusters. 6 LCD displays visually present information in the cockpit. Two of these visual displays are dedicated to the instrument clusters while the other four provide external (environmental) visual cues.

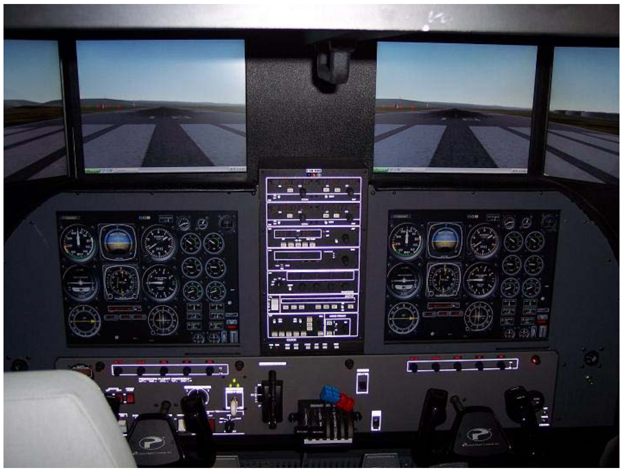


Figure 7.4: Visual Displays, Controls, and Instrument Cluster

The instructor console previously illustrated in Figure 7.2 has two visual displays to monitor the simulation and to perform simulation scenario set-ups/changes. A better view of the instructor operating station located on the access platform is shown in Figure 7.5.

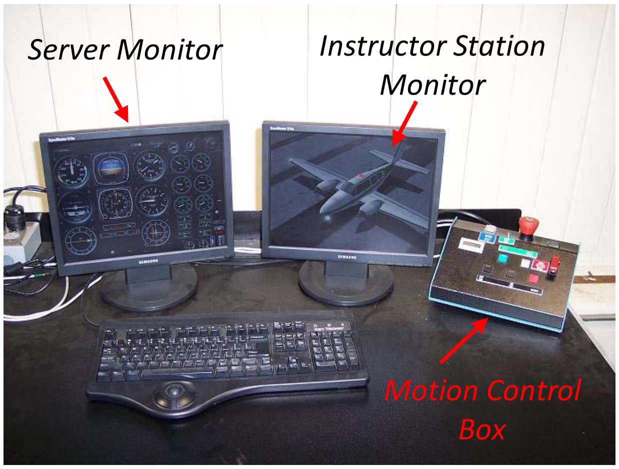


Figure 7.5: Instructor Console

The large black aluminum cabinet next to the cabin in Figure 7.2 houses all electrical and computing hardware. Five computers are used to operate the WVU Flight Simulator. Computer #1 drives the left 45° visual display. Computer #2 drives the left and right forward visual displays. Computer #3 drives the right 45° visual display. Computer #4 is the “Server” computer and runs the core flight simulation software and the pilots’ instruments. All simulation data to be used for analysis is stored on this computer. Computer #5 is the instructor’s operating station. All 5 computers can be controlled using the keyboard on the instructor’s desk. All functions of the motion base can be controlled through a separate Motion Control Box, also shown in Figure 7.5.

A total of 34 flight simulator tests with three different pilots have been performed for the purpose of the project, adding up to approximately 20 hours of testing including integration of engine model with flight simulator, pilot preliminary training, simulator preparation, and data retrieval. Pilot #1 performed approximately 13 hours of the testing while Pilot #2 and Pilot #3 performed about 6 hours and 1 hour, respectively.

Chapter 8: Simulation Results and Discussion

8.1 Validation at Nominal Conditions

In all the figures showing testing results within this chapter, the top subplot shows the normalized data for all the identifiers used to build the self. These parameters include the throttle command, mixer area, fan guide vanes, HPC temperature, and mixer pressure ratio. The plots typically show more variation of the parameters during the doublets and smoother sections while the pilot accelerated, etc., to a new point in the flight envelope. It is important to note that the throttle command is represented by the joystick signal with inverted sign as well as normalized between 0 and 1; therefore, when the value is low or decreasing, the pilot was accelerating. The bottom subplot shows the number of antibodies (or detectors) that detected failure over a previous time window of 1 second. When the plot shows a zero value then no abnormal condition is detected, while a non-zero value means exactly the opposite. The highest value of activated antibodies is always 50 since the sampling rate used in the detection scheme was 50Hz.

The flight test for validation of the AIS at nominal conditions consisted of doublets performed at point 1, acceleration and ascension to point 3, and doublets performed at point 3. The data from this test was normalized and used to test the antibodies. An attempt was first made to determine the level of robustness of the data acquisition process in capturing self characteristics over extended regions of the flight envelope. Preliminary research has shown that data acquired over the contour of the box in Figure 7.1 plus the internal cross-like segments were enough to represent the self at all points inside the box. Therefore, a first self was obtained based on tests along the cross-like segments only. This preliminary self created from the training data resulted in the generation of 1000 antibodies. The results of the validation flight test using the first set of antibodies are shown in Figure 8.1. These results show the number of false alarms to be very high, approximately 62%. Also shown in Figure 8.1 is the fact that the false alarms do not occur during flight at point 1, where training data was available, but rather in the vicinity of point 3. The rate increased shortly after acceleration and ascension to point 3 began (around 250sec). This preliminary test shows that the generation of the self must include additional information at flight conditions close to point 3.

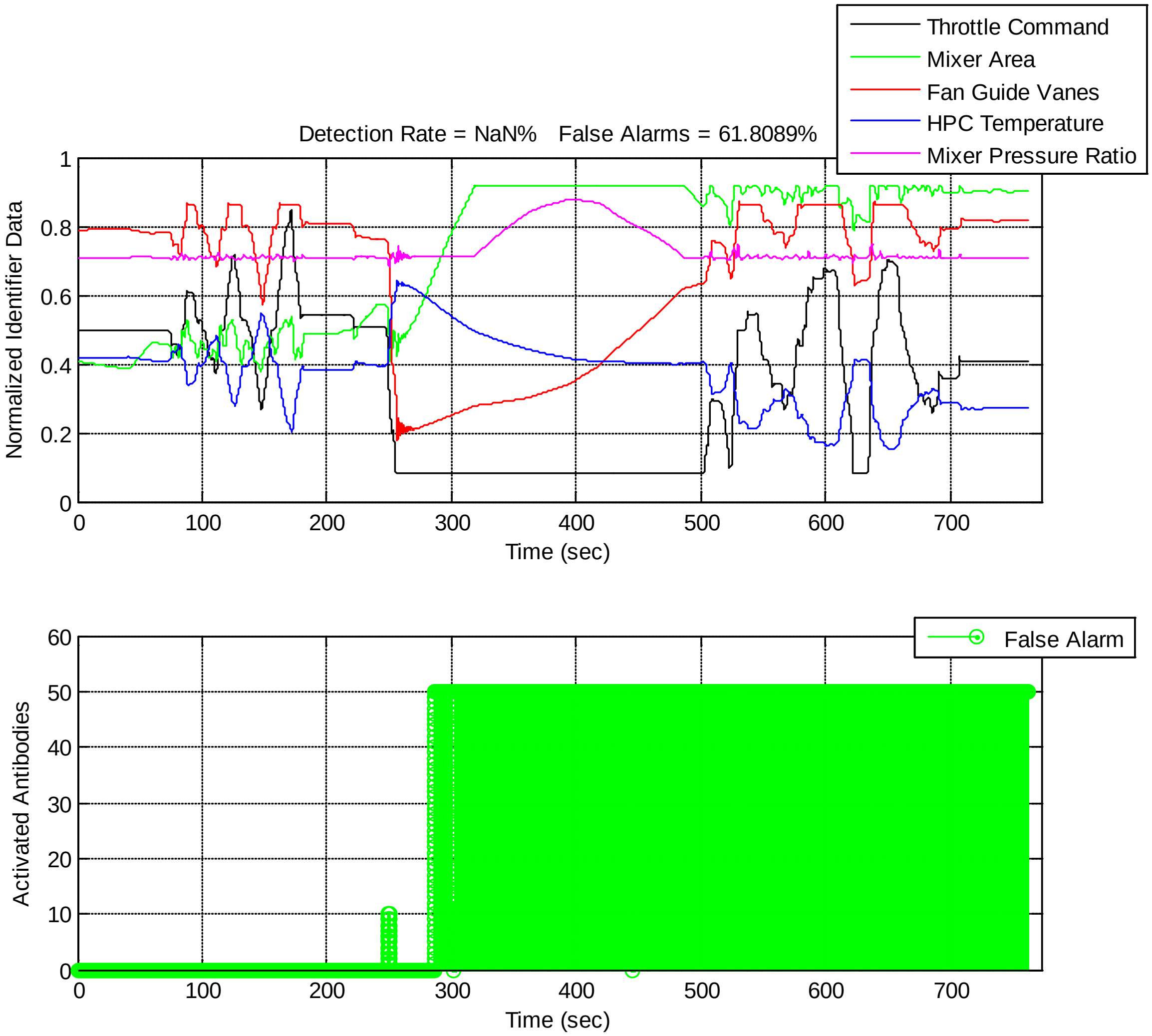


Figure 8.1: Preliminary False Alarms for Left Engine Validation Data for Flight Path 1 to 3

To more clearly understand this conclusion, the 2-D plots in Figure 8.2 and Figure 8.3 are used to illustrate the difference in the points included in the self and points from the validation data for the left engine. Figure 8.2 shows that about half of the red validation data points are within the green cluster data, which represents the nominal flight at point 1. The other section of data corresponding to the vicinity of point 3 is outside of the self, which the antibodies viewed as unhealthy flight data causing the false alarms. Figure 8.3 uses two different identifiers to show the same results with about half of the validation data outside of the self cluster data.

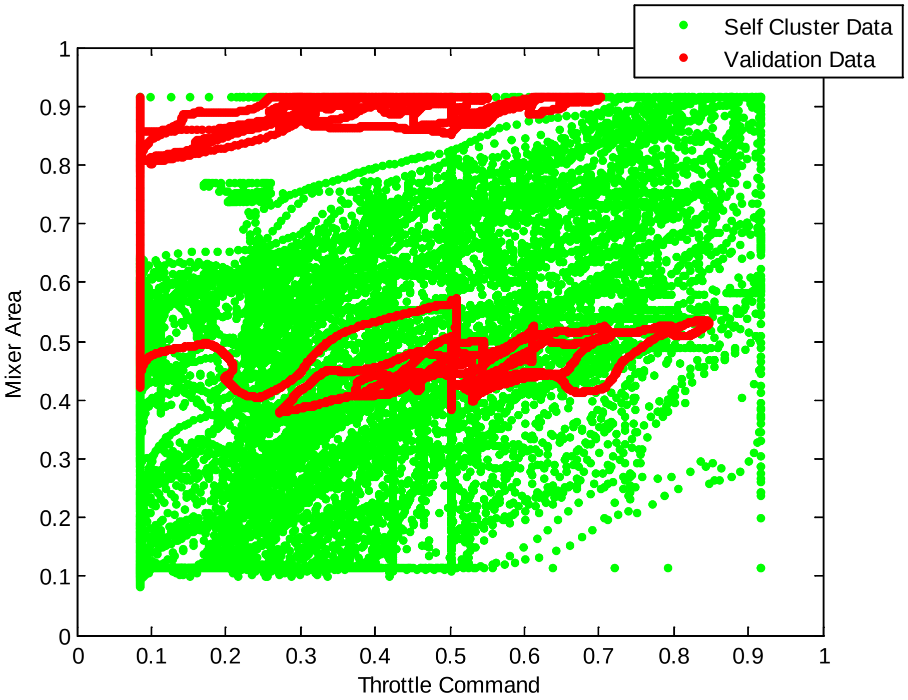


Figure 8.2: Normalized Validation Data compared to Self Cluster Data for Throttle Command versus Mixer Area

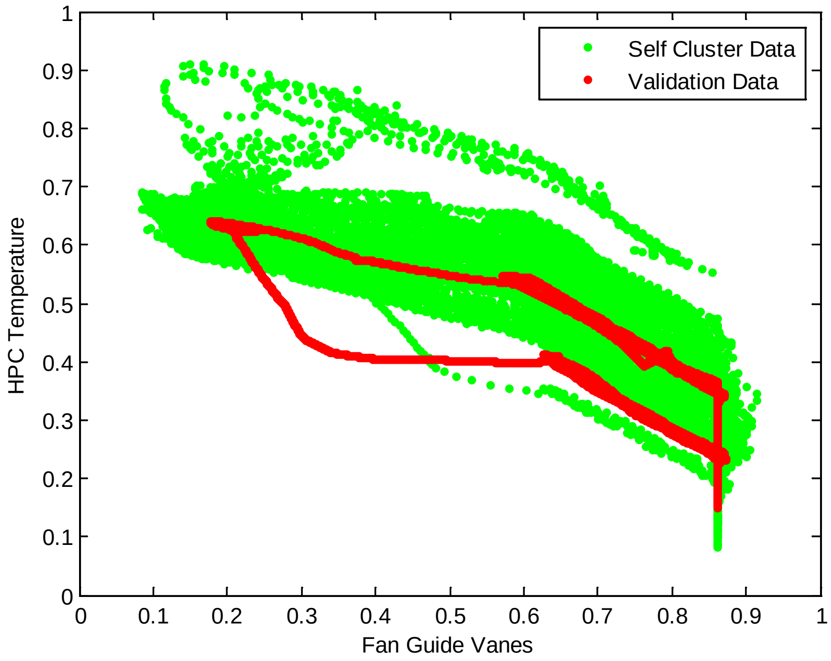


Figure 8.3: Normalized Validation Data compared to Self Cluster Data for Fan Guide Vanes versus HPC Temperature

Since these preliminary results clearly show that the training data was not adequate to create an accurate self, more training was added around point 3 of the flight envelope. This data included two flight tests. One flying from point one to 2 to 3 with the doublets performed at point 3 and another from 1 to 4 to 3 with doublets at point 3. The new antibodies created from this data yielded the results shown in Figure 8.4. Note that the validation data used in this test was the exact data used in Figure 8.1. With false alarms of less than 0.6%, the validation results are excellent.

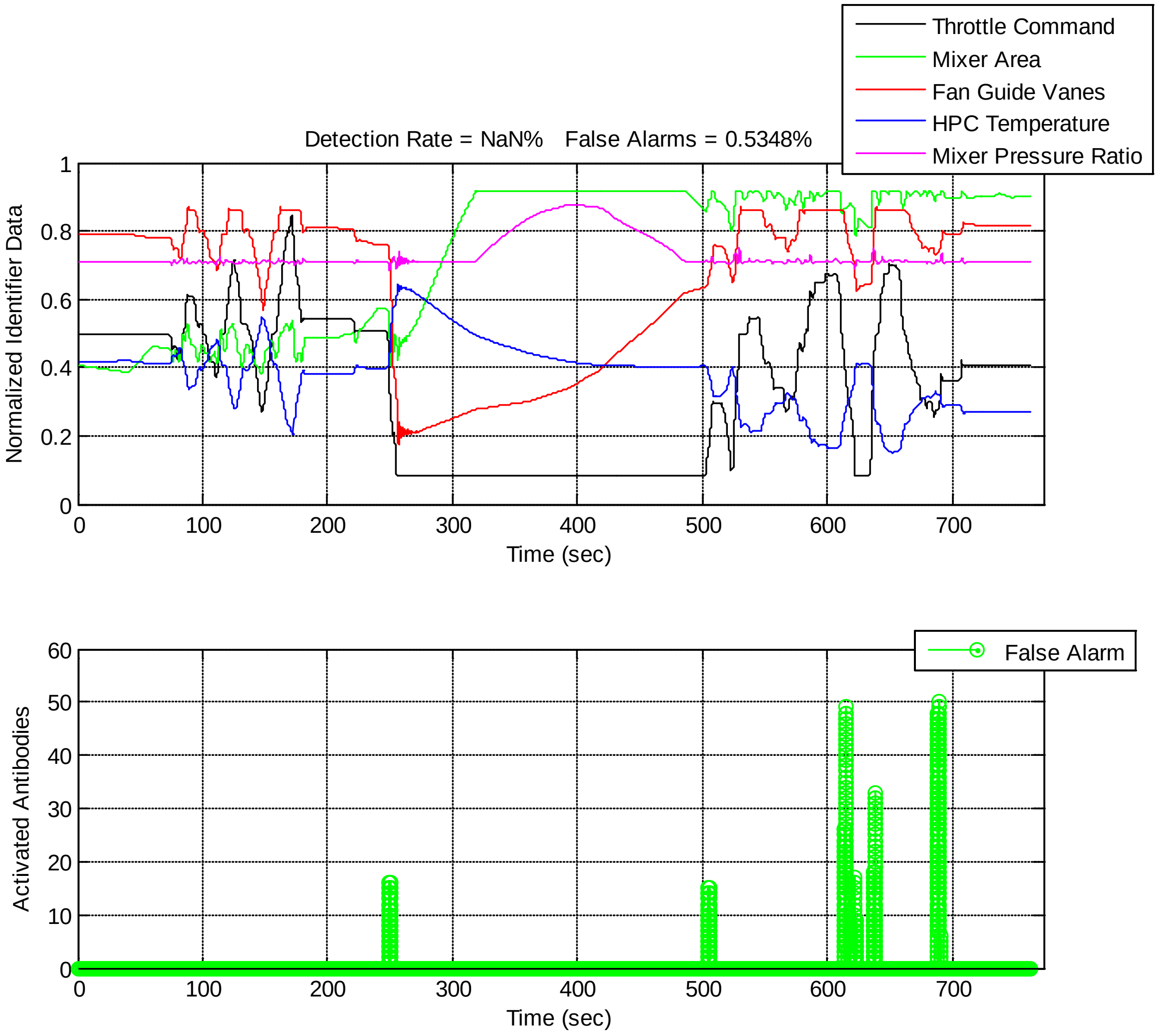


Figure 8.4: False Alarms for Left Engine Validation Data for Flight Path 1 to 3

The new 2-D plots are also included in Figure 8.5 and Figure 8.6 to show the difference from the previous self. Although not all of the validation data is included in the self, enough is added to significantly decrease the number of false alarms and increase the validity of the self.

This also shows that only training the system for the middle part of the flight envelope is not adequate to represent the flight at the other points. It does, however, show that continuing to train around the outside of a section can be enough to represent flight in the middle. This result confirms the level of "robustness" previously determined in the acquisition of data for self definition.

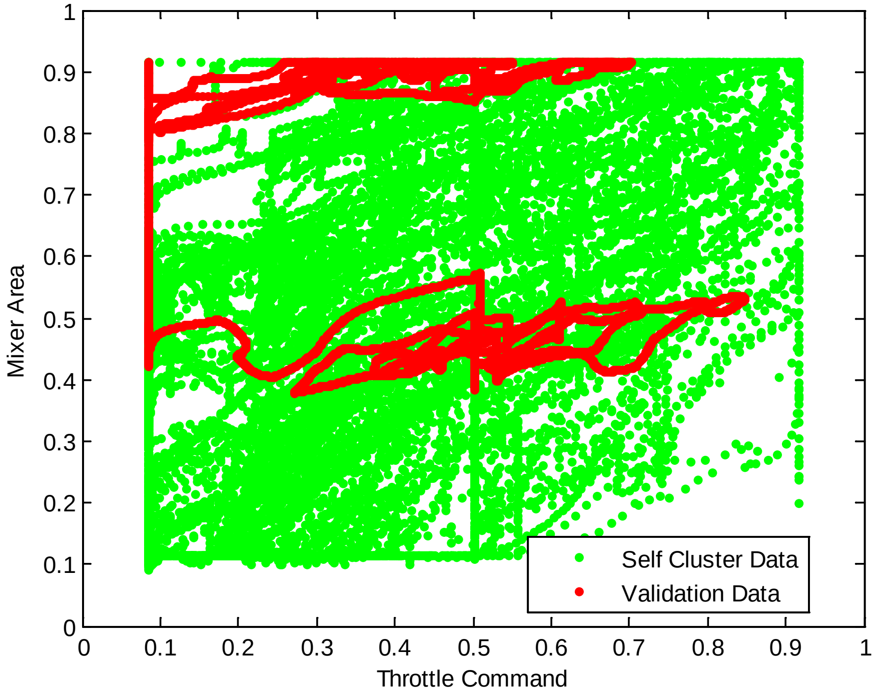


Figure 8.5: Normalized Validation Data compared to New Self Cluster Data for Throttle Command versus Mixer Area

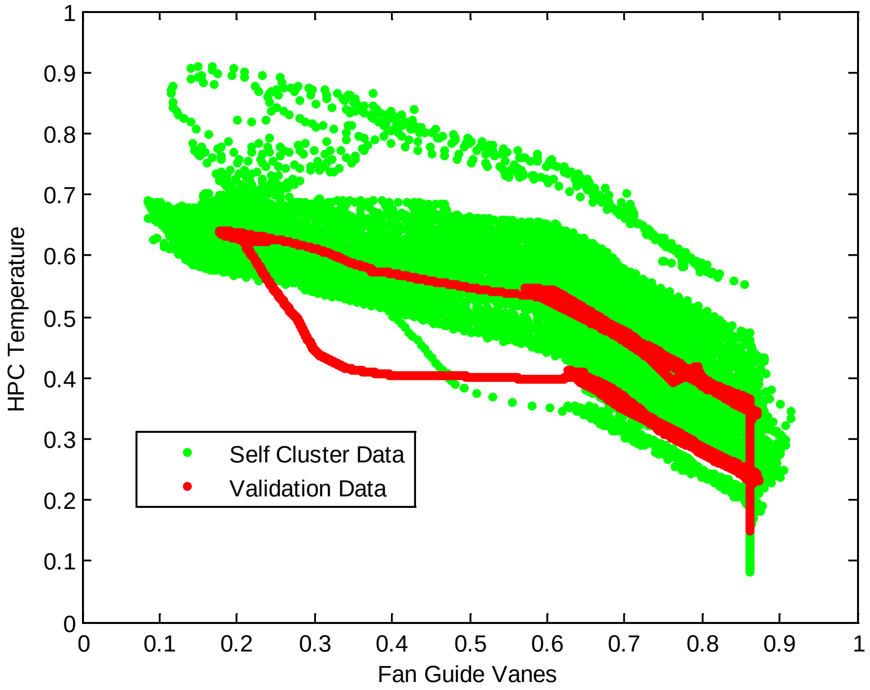


Figure 8.6: Normalized Validation Data compared to New Self Cluster Data for Fan Guide Vanes versus HPC Temperature

8.2 Throttle Failure

One of the strongest engine failures is the throttle stuck at zero. This means close to zero thrust will be produced by the unhealthy engine. This failure should be easily detected due to its impact on the engine functionality. Another reason to analyze this failure was to compare it with the detection rate using the previous engine model. The stuck throttle failure was performed on both the left and right engines. In this case, the failure was too strong to accelerate or ascend to other points of the flight envelope; therefore, the flight tests were broken into sections. These figures are set up like the previous figures for validation. The only difference is now the detection rate is included with the false alarms. The green part of the bottom subplot on each figure represents the false alarms that occur during nominal flight before the failure is injected. The number of false alarms must remain low for all failure tests to justify the AIS. The red section of the subplot represents detection of the failure. Basically, the more red area shown in the plot, the better the detection rate. Figure 8.7 and Figure 8.8 show the stuck throttle detection at point one of the flight envelope. Figure 8.7 for the left engine shows a good detection rate at just under 95% with no false alarms. The right engine also had good results with no false alarms and over 90% detection rate. The only gaps in these two plots seem to come when the throttle is decreasing, which coincides with engine burnout. This may be corrected by including a temperature sensor as an identifier to detect the decrease in temperature during burnout.

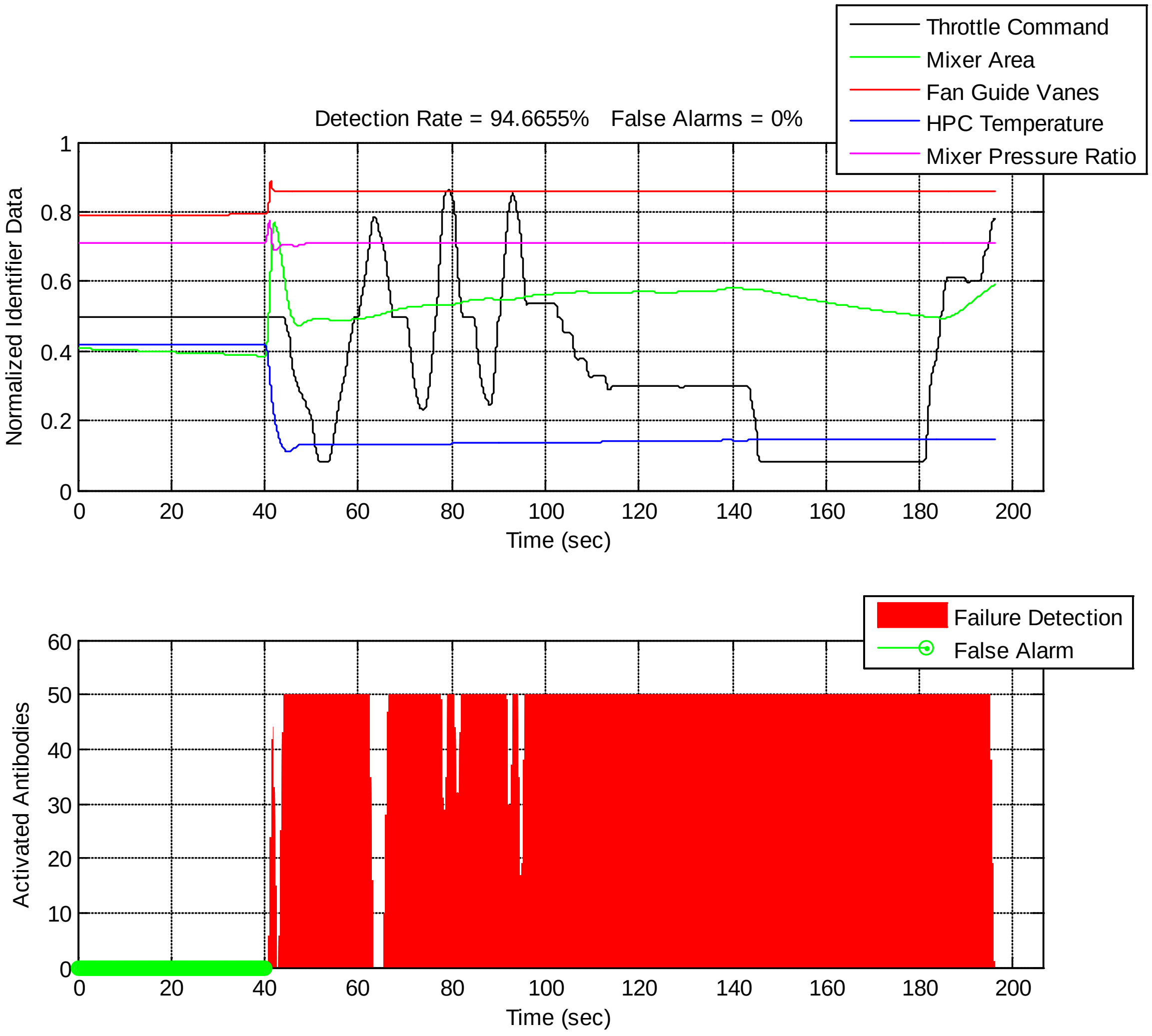


Figure 8.7: Left Engine Stuck Throttle Failure at 40sec for Flight at Point 1

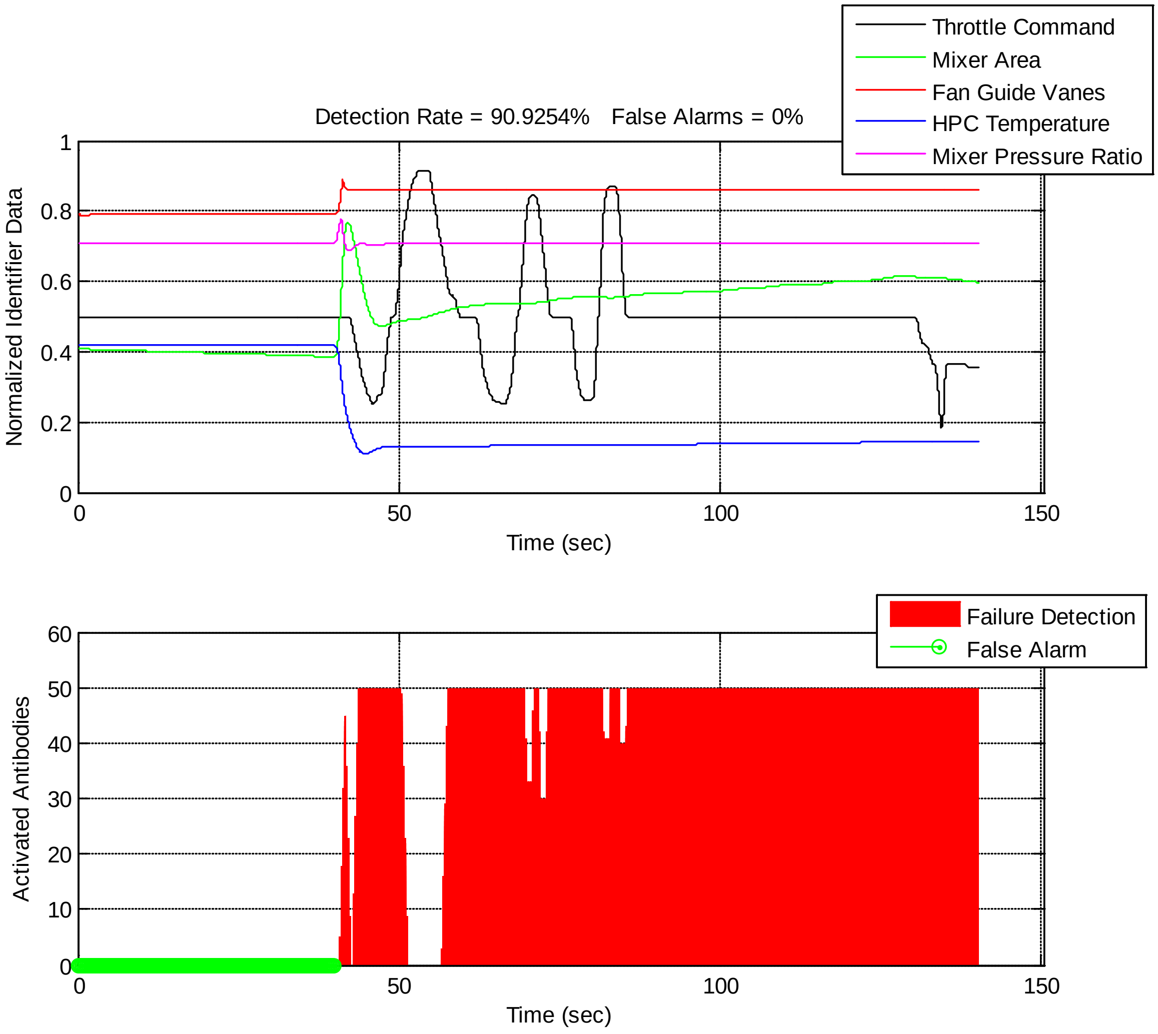


Figure 8.8: Right Engine Stuck Throttle Failure at 40sec for Flight at Point 1

The next part of the flight envelope to be tested includes points 4 and 8, corresponding to changing Mach and constant altitude. The failure occurs at 60sec in these tests in order to give the pilot time to accelerate to point 4 first since the speed decreases rapidly after the failure. Figure 8.9 shows the results of the left engine failure. The detection rate here is still high at about 86% with very low false alarms of 3%. The small number of false alarms is most likely due to the sudden pilot acceleration while the gap in the detection was during steady deceleration to point 8, which may not have had enough throttle excitation to show the failure.

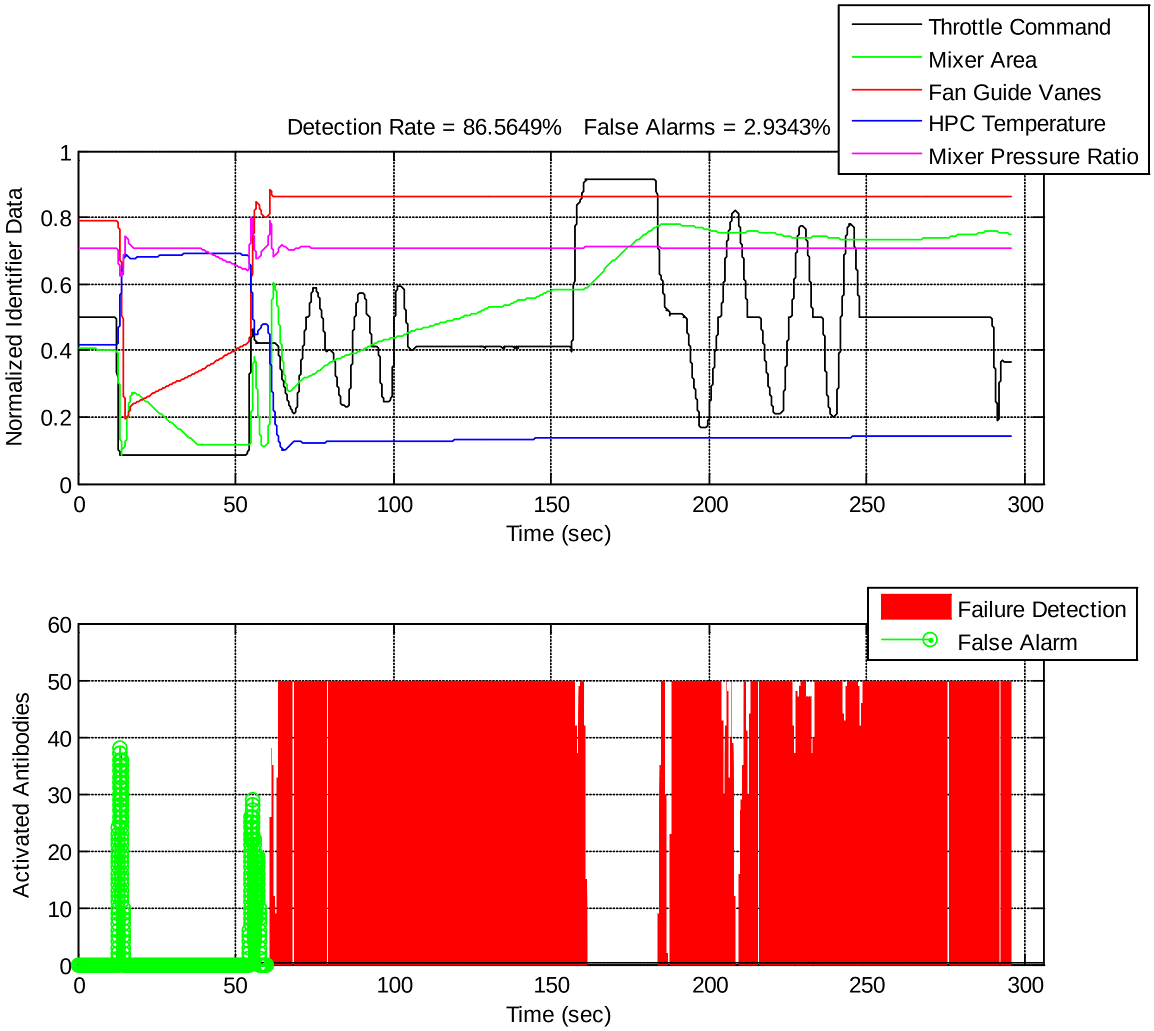


Figure 8.9: Left Engine Stuck Throttle Failure at 60sec for Flight Path 4 to 8

Figure 8.10 shows similar results for the right engine with a detection rate of about 80% but with zero false alarms. The breaks in detection again correspond to deceleration to point 8 and the decreasing part of the throttle doublets.

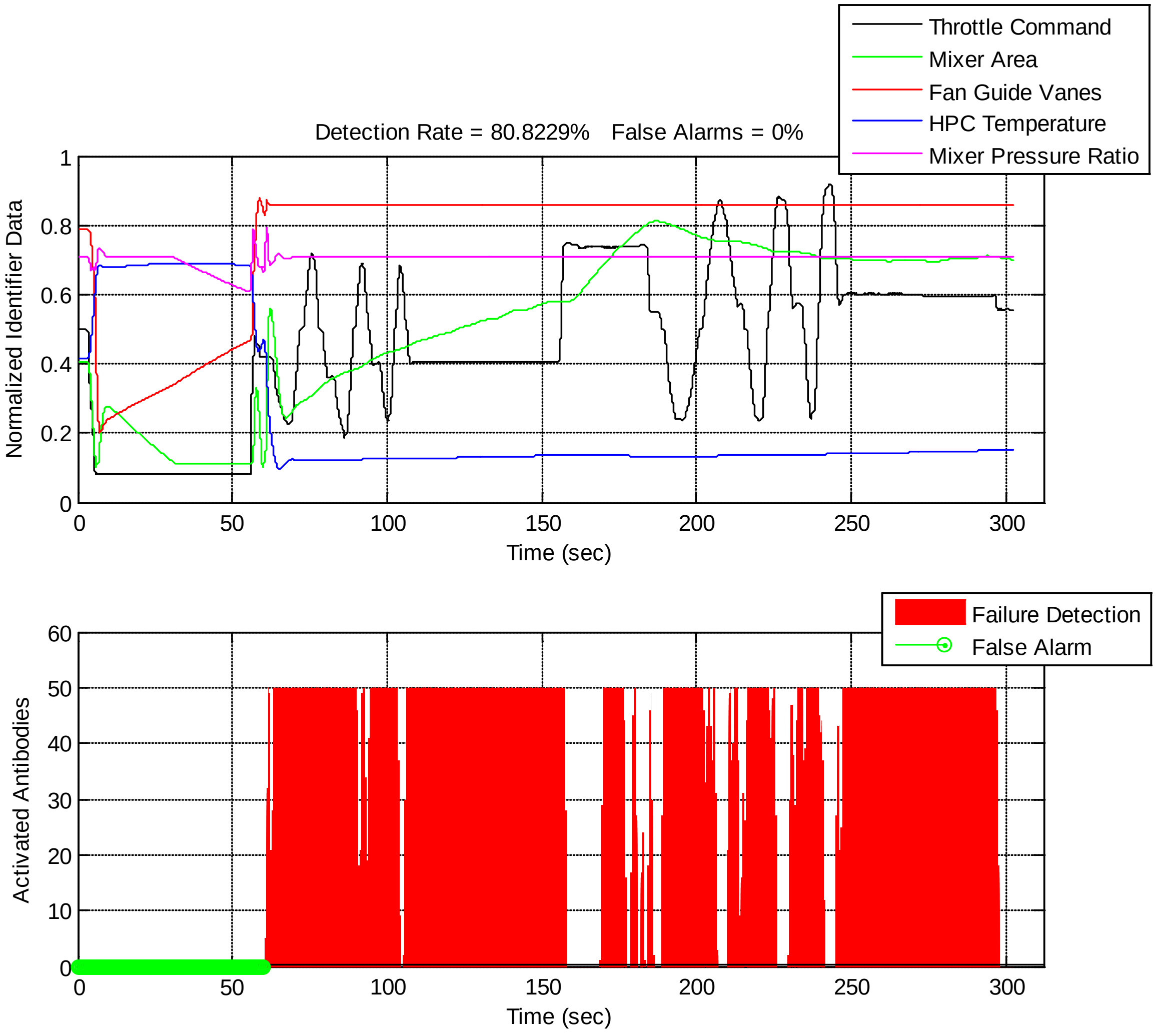


Figure 8.10: Right Engine Stuck Throttle Failure at 60sec for Flight Path 4 to 8

The final section of the flight envelope to test includes points 2 and 6, which holds Mach steady and varies the altitude. The failure occurs at 240sec for this test since the pilot had to climb before the failure. The detection rate this time was smaller than before, but still a solid value at 67% with very low false alarms. These false alarms correspond to the initial pilot maneuvers while the decrease in the detection rate matches the time when the pilot slowly descends to point 6. Here, there is basically no pilot input while the aircraft goes to the new altitude.

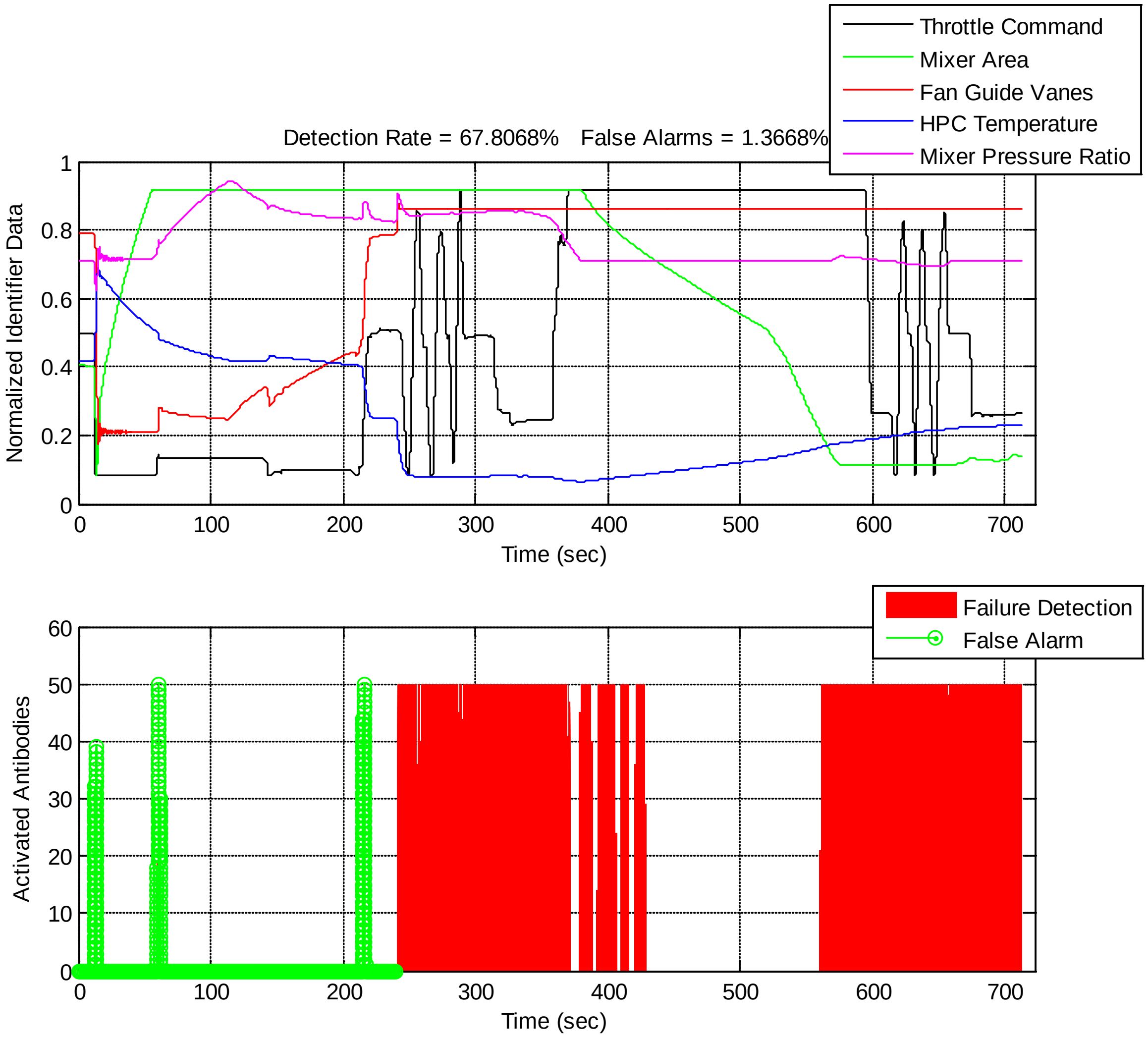


Figure 8.11: Left Engine Stuck Throttle Failure at 240sec for Flight Path 2 to 6

All of these results can be compared to the previous engine failure detection rates and false alarms. The most comparable scenario to evaluate the improvement of the AIS failure detection is the 98% reduced efficiency engine failure for the reduced flight envelope [32]. The thrust data from the throttle failure tests show that the failed engine was functioning at approximately 13% of the healthy engine and therefore corresponds to an 87% reduced efficiency engine failure. Comparison of the detection schemes is included later in this chapter.

8.3 Engine Actuator Failures

Several actuator failures were tested in the flight simulator to find the detection rate and number of false alarms. The actuator failure included the burner fuel flow stuck at $0lb_m$/hr, the nozzle area locked at the area at the time of failure, and the mixer area locked at the area at the time of failure.

8.3.1 Burner Fuel Flow Valve Failure

The most beneficial test here was the burner fuel flow. This failure was much like the stuck throttle failure and can be easily compared to previous research results. Setting the burner fuel flow to 0, however, was a stronger failure in the fact that the thrust from the unhealthy engine was negative, since the engine was producing drag. The burner failure was tested for both the left and right engines. The following figures show the results.

Figure 8.12 and Figure 8.13 both test a left engine failure at 60sec for flight at point one while Figure 8.14 contains the results of the right engine failure. Both of the left engine tests and the right engine test have very high, over 99.9%, detection rates with no false alarms. The detection performance for this type of failure may be evaluated as excellent.

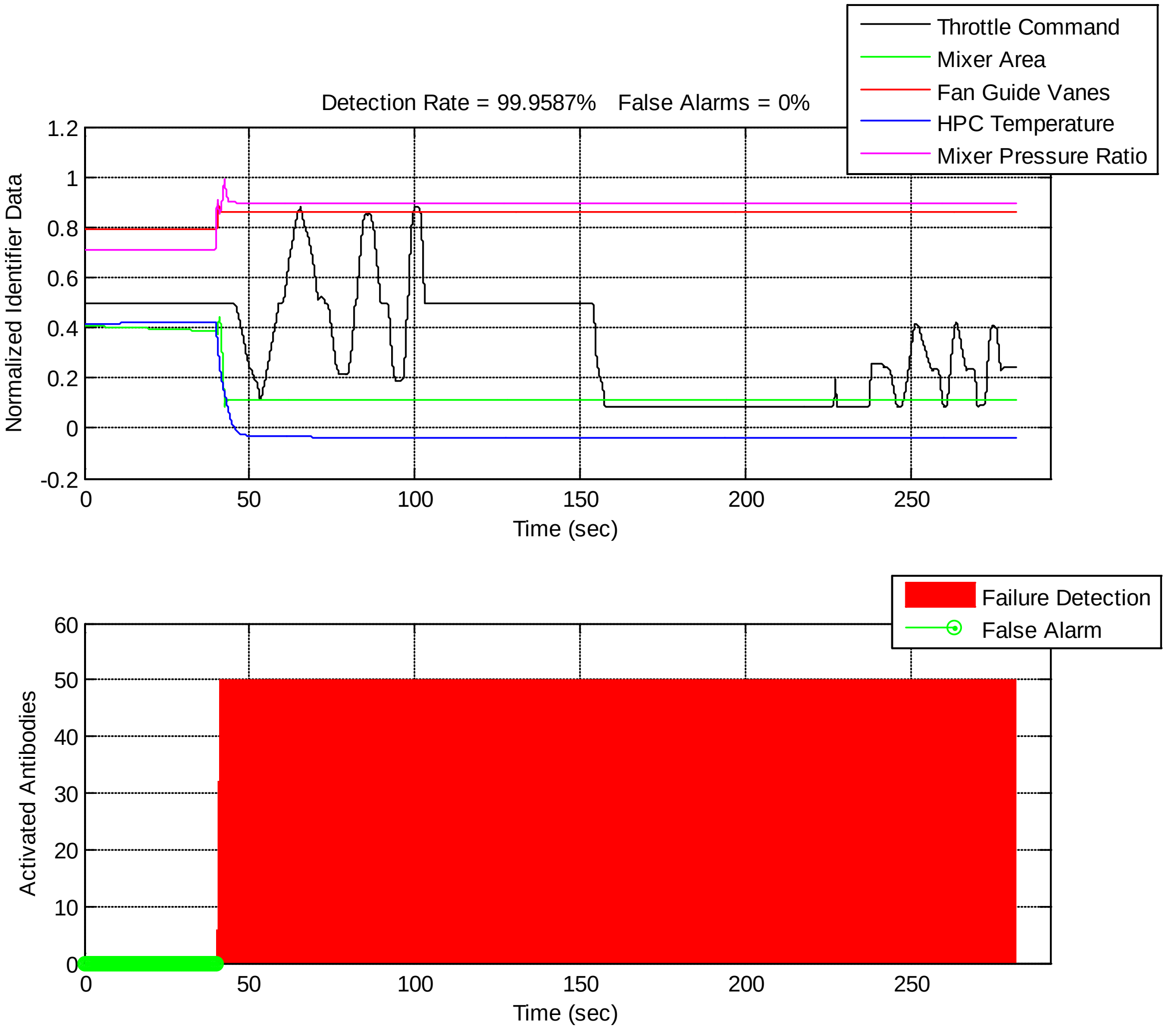


Figure 8.12: Left Engine Burner Fuel Flow Failure at 40sec for Flight at Point 1

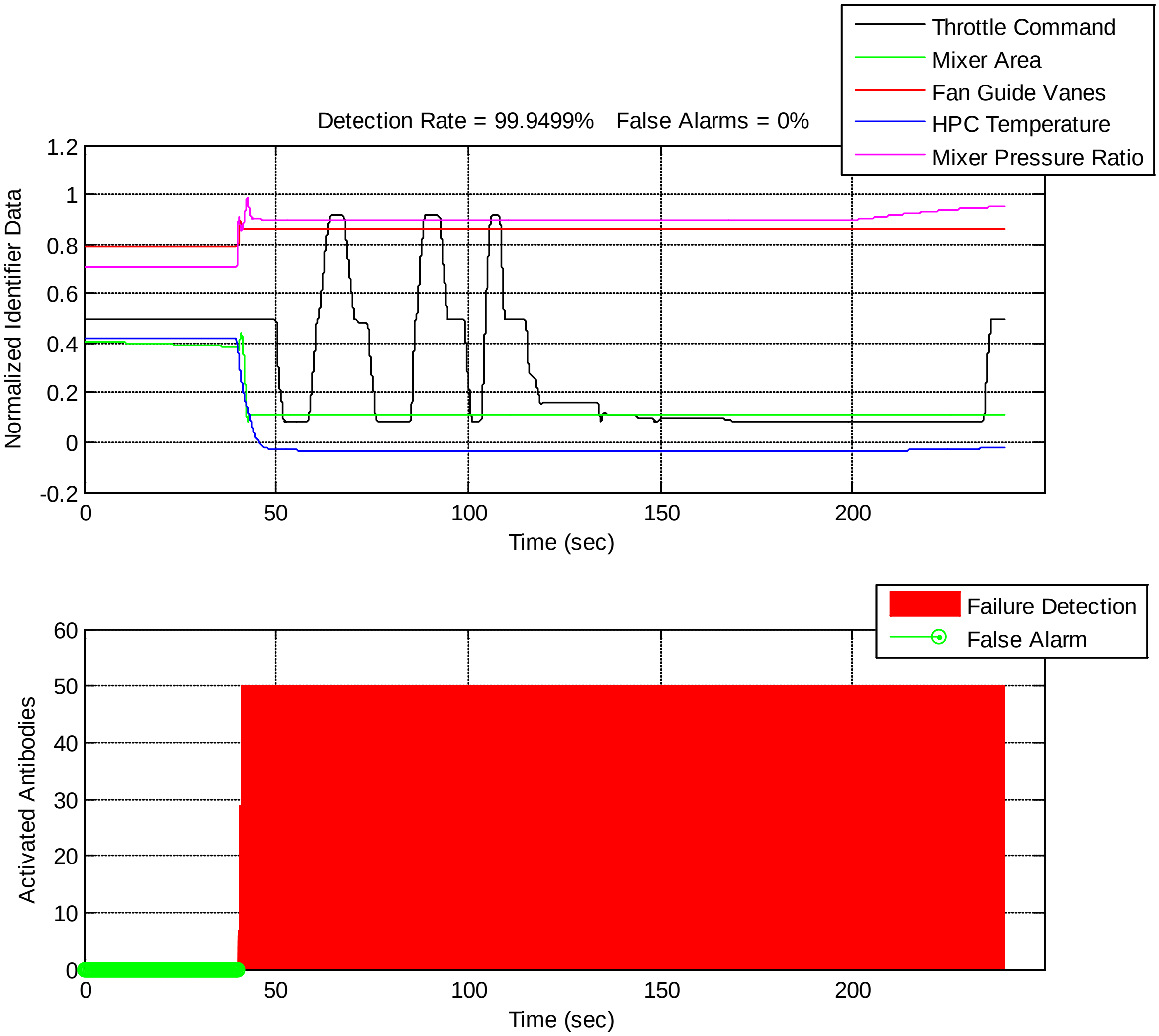


Figure 8.13: Left Engine Burner Fuel Flow Failure at 40sec for Flight at Point 1

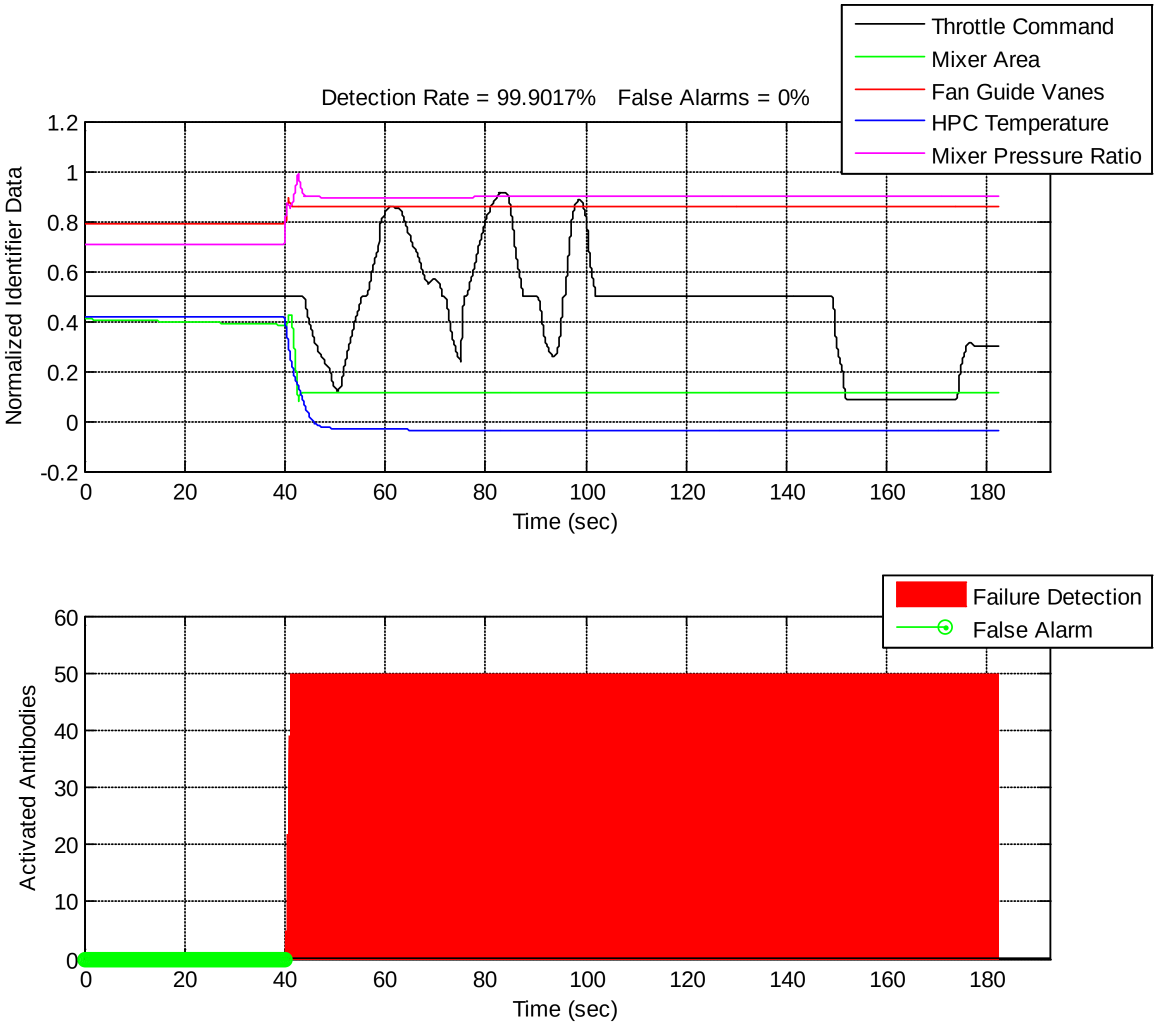


Figure 8.14: Right Engine Burner Fuel Flow Failure at 40sec for Flight at Point 1

The flight tests to cover points 4 and 8 of the flight envelope have the same results as point 1. The failure in Figure 8.15 and Figure 8.16 for the left and right engine, respectively, occurs at 60sec to allow the pilot time to accelerate before the failure. With detection rates over 99.9% and zero false alarms, the results of these flight tests are outstanding.

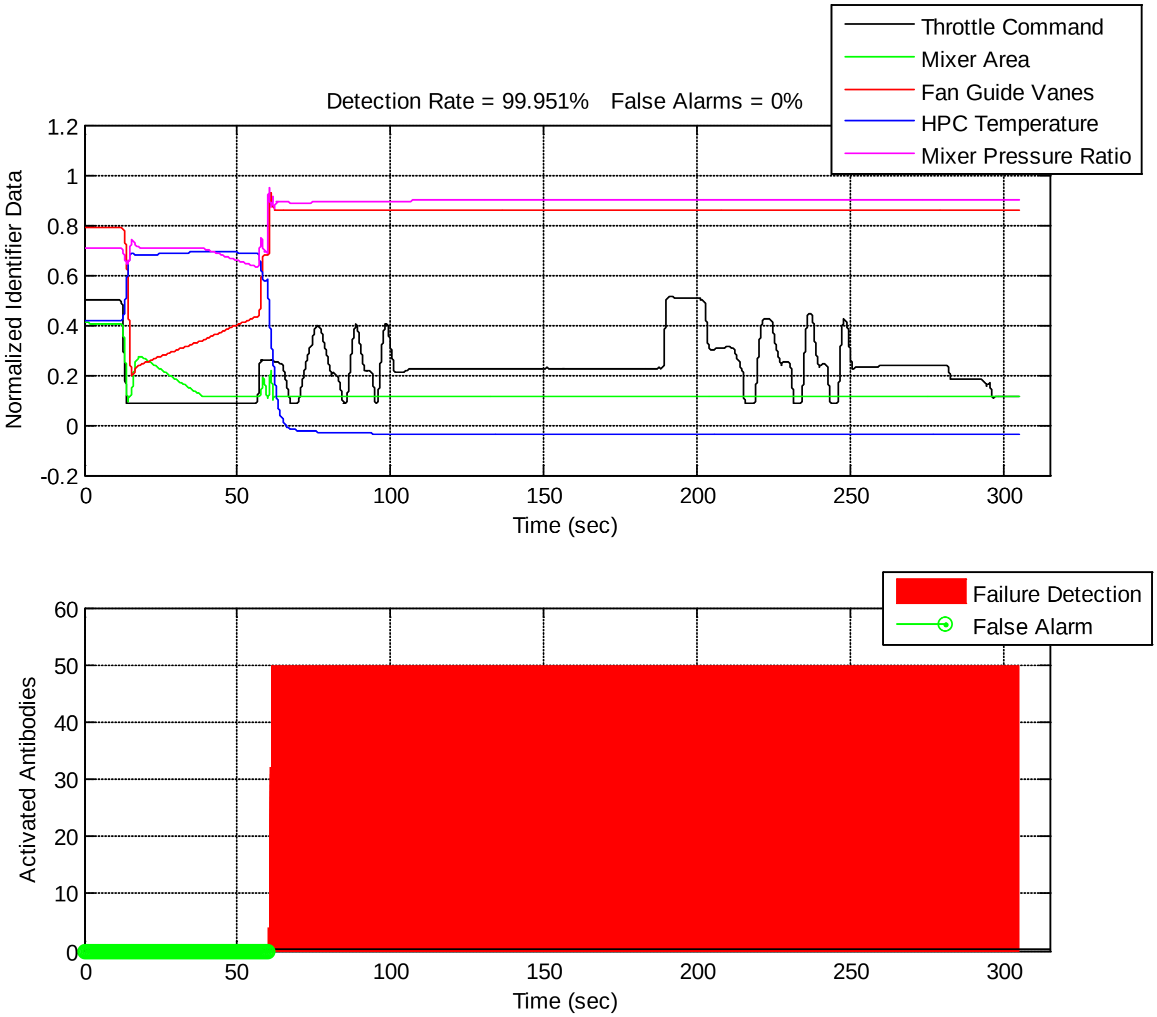


Figure 8.15: Left Engine Burner Fuel Flow Failure at 60sec for Flight Path 4 to 8

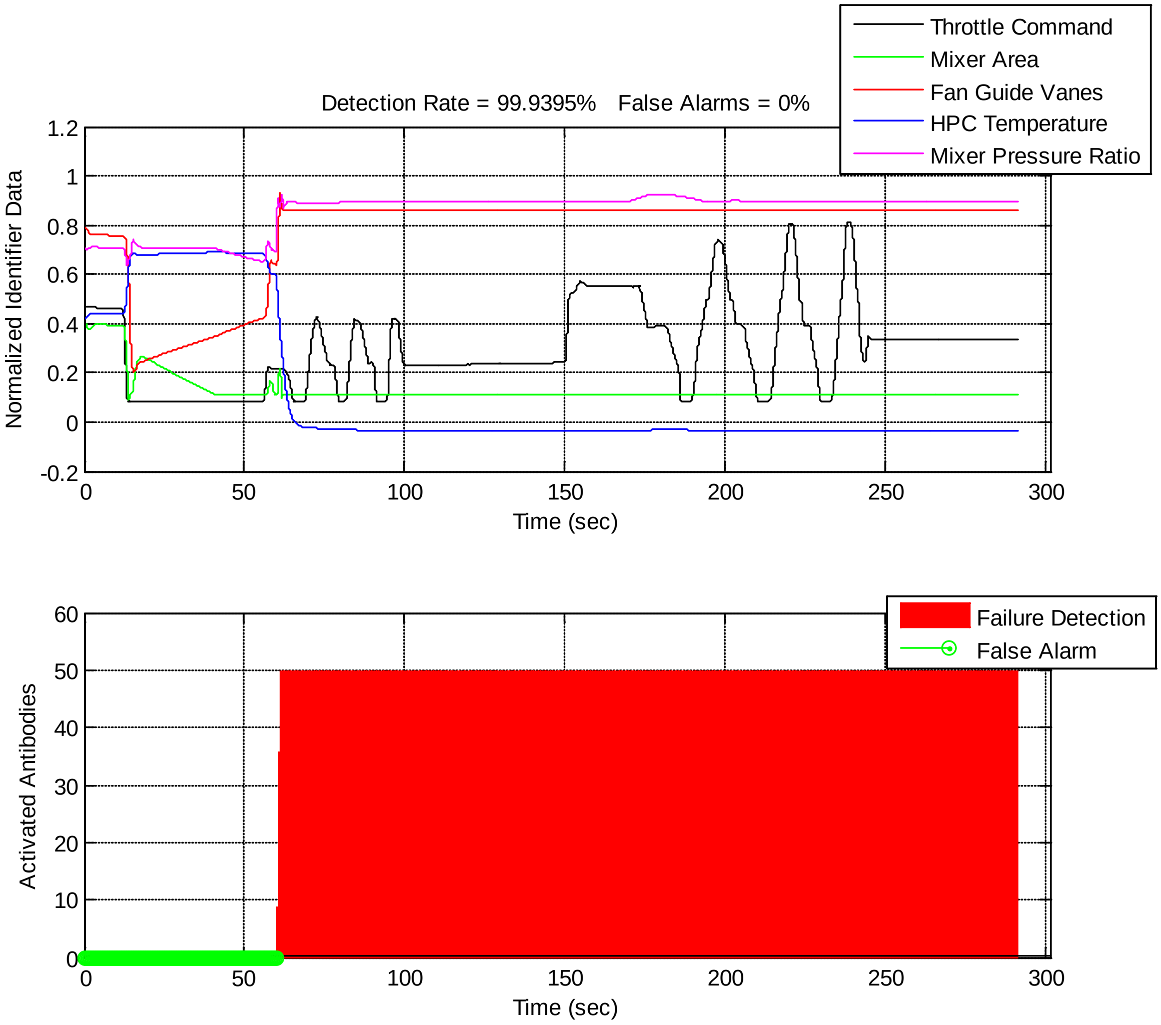


Figure 8.16: Right Engine Burner Fuel Flow Failure at 60sec for Flight Path 4 to 8

The final test for the burner fuel flow area involves flight at points 2 and 6. In this case, the failure in implemented at 480sec. This allows the pilot to ascend to point 2 more slowly and hopefully decrease the false alarms compared to Figure 8.11. This is successful with false alarms of less than 1% and detection again over 99.9%. These results will later be compared to the previous research. Even though this failure is somewhat stronger, it is still comparable to the failed throttle results. The equivalent reduced engine efficiency failure would be at approximately 113%.

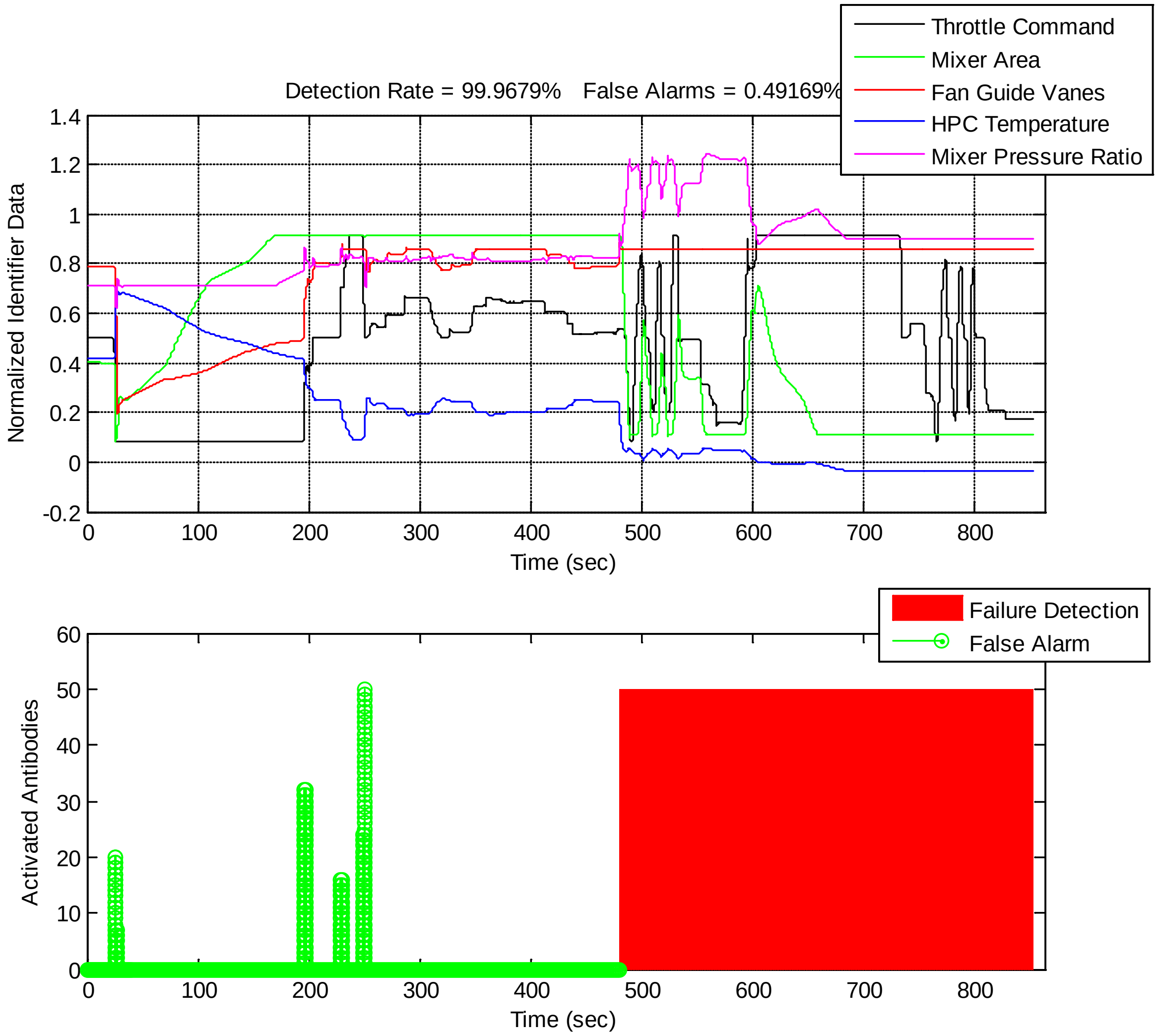


Figure 8.17: Left Engine Burner Fuel Flow Failure at 480sec for Flight Path 2 to 6

8.3.2 Nozzle Area Actuator Failure

While failure of the nozzle area does not have as big of an impact on the flight as the previous failures, the actuator controlling the nozzle area is an important part of the engine; therefore, detection of the actuator malfunction is desired. The results from the flight tests are shown in Figure 8.18 for a left engine failure along the path from 1 to 4 to 8. The detection rate appears to be very low for this type of failure; however, the same failure is detected at a very good rate of 55% with no false alarms if it occurs at different flight conditions as shown in Figure 8.19. This result suggests that although the set of features selected to define the self is adequate, the detector generation process failed to produce detectors in regions of the hyper space corresponding to the flight path 1-4-8. This may be due to a too “generous” clustering process with the inclusion of large “empty spaces” and/or to low coverage of the non-self during the detector generation. In both situations, areas of the non-self are considered normal and therefore lead to poor detection. The complete optimization of the detector set using both phase I and phase II of the evolutionary algorithm [ref] is expected to considerably improve the detection in this area of the flight envelope.

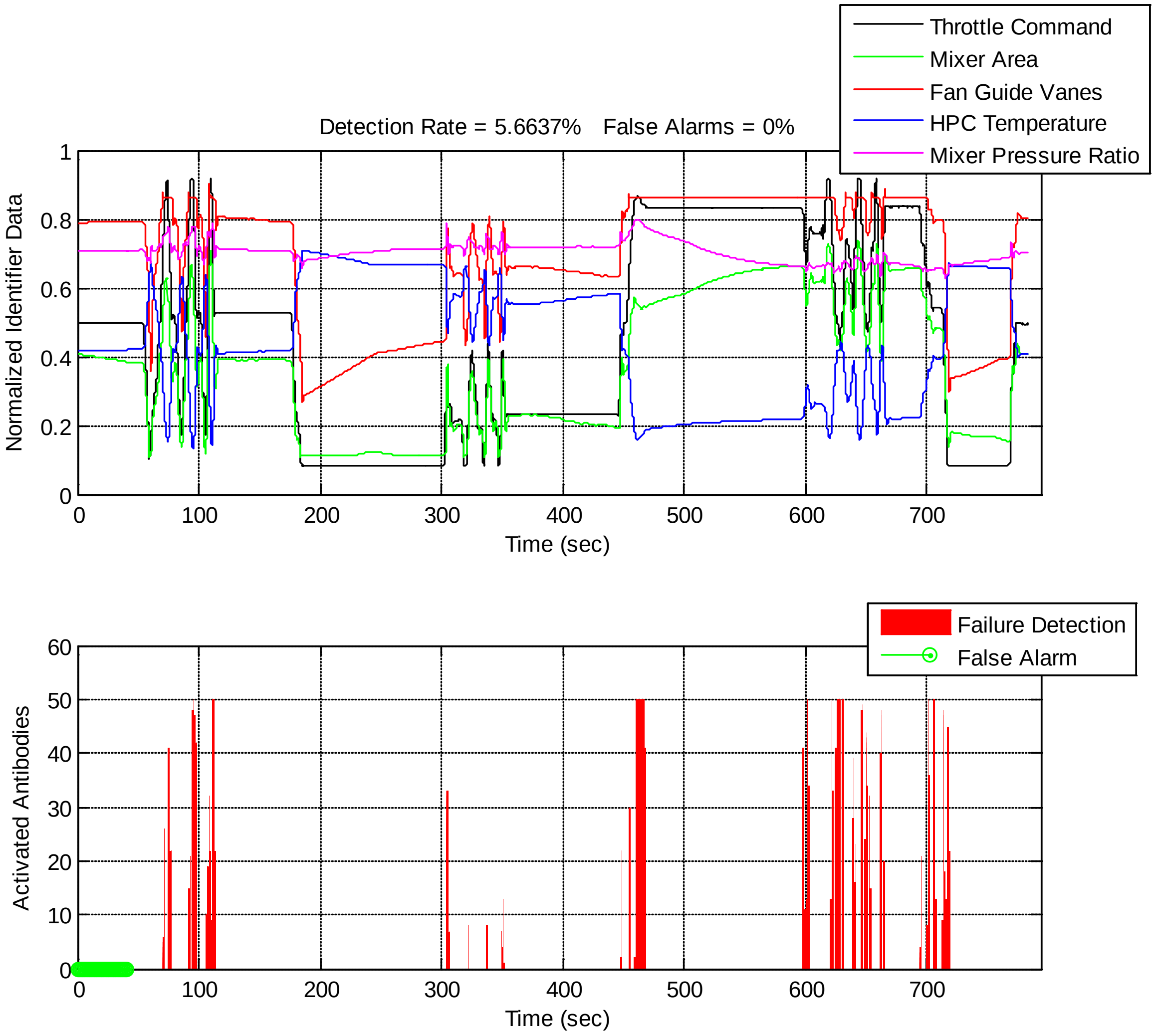


Figure 8.18: Left Engine Nozzle Area Actuator Failure at 40sec for Flight Path 1 to 4 to 8

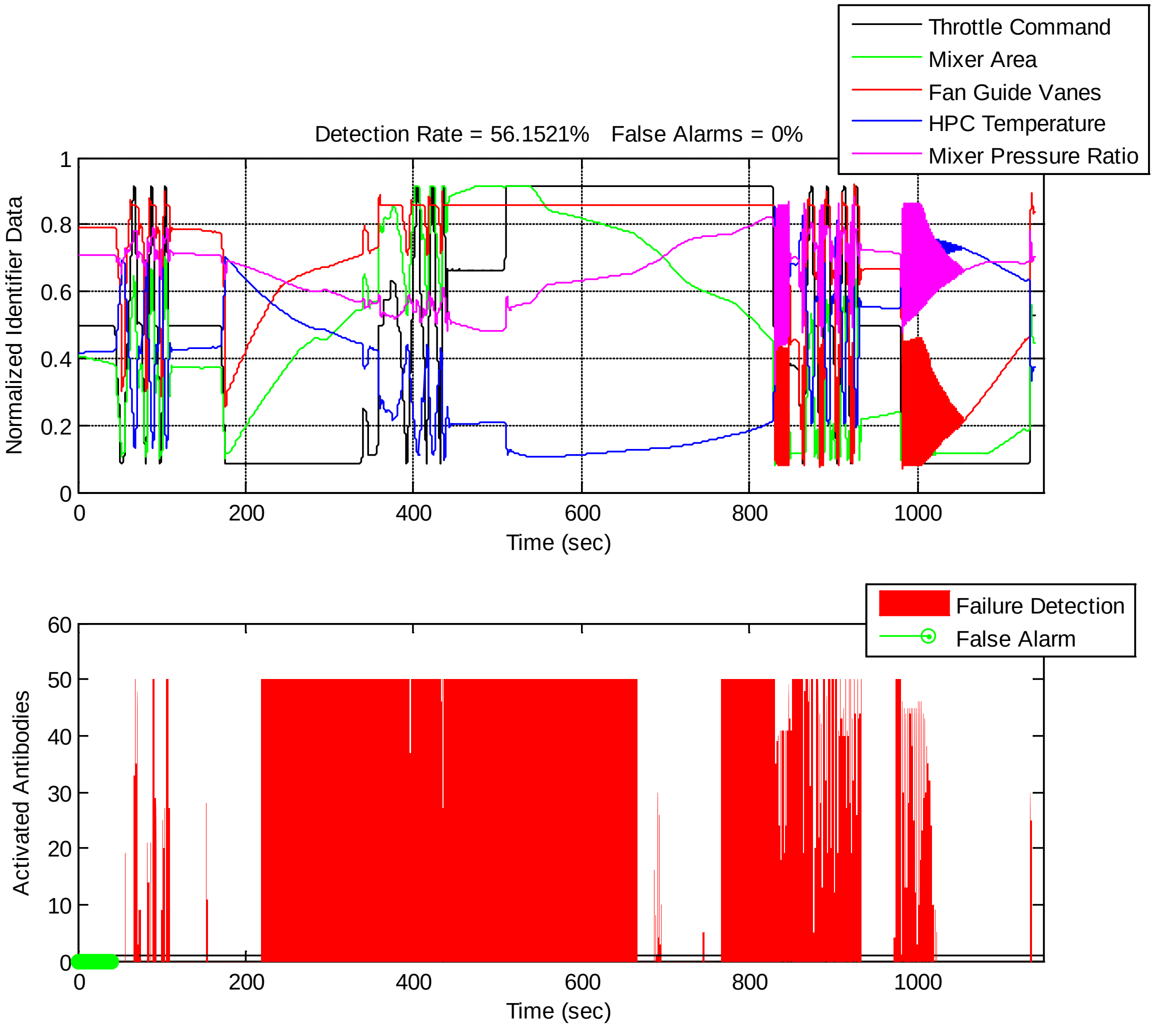


Figure 8.19: Left Engine Nozzle Area Actuator Failure at 40sec for Flight Path 1 to 2 to 6

8.3.3 Mixer Area Actuator Failure

Next, the actuator controlling the mixer area was locked at the current position. This failure had good results for all tests, which are shown in Figure 8.20, Figure 8.21, and Figure 8.22. The flight path from point 1 to 4 to 8 and back to 1 was performed during failure on the left and then the right engine. Both tests yield a detection rate of about 67% and no false alarms. In Figure 8.20 and Figure 8.21, the detection is not very good while the aircraft is flying at point 1 and accelerating to point 4. However, once at point 4, the detection is much better.

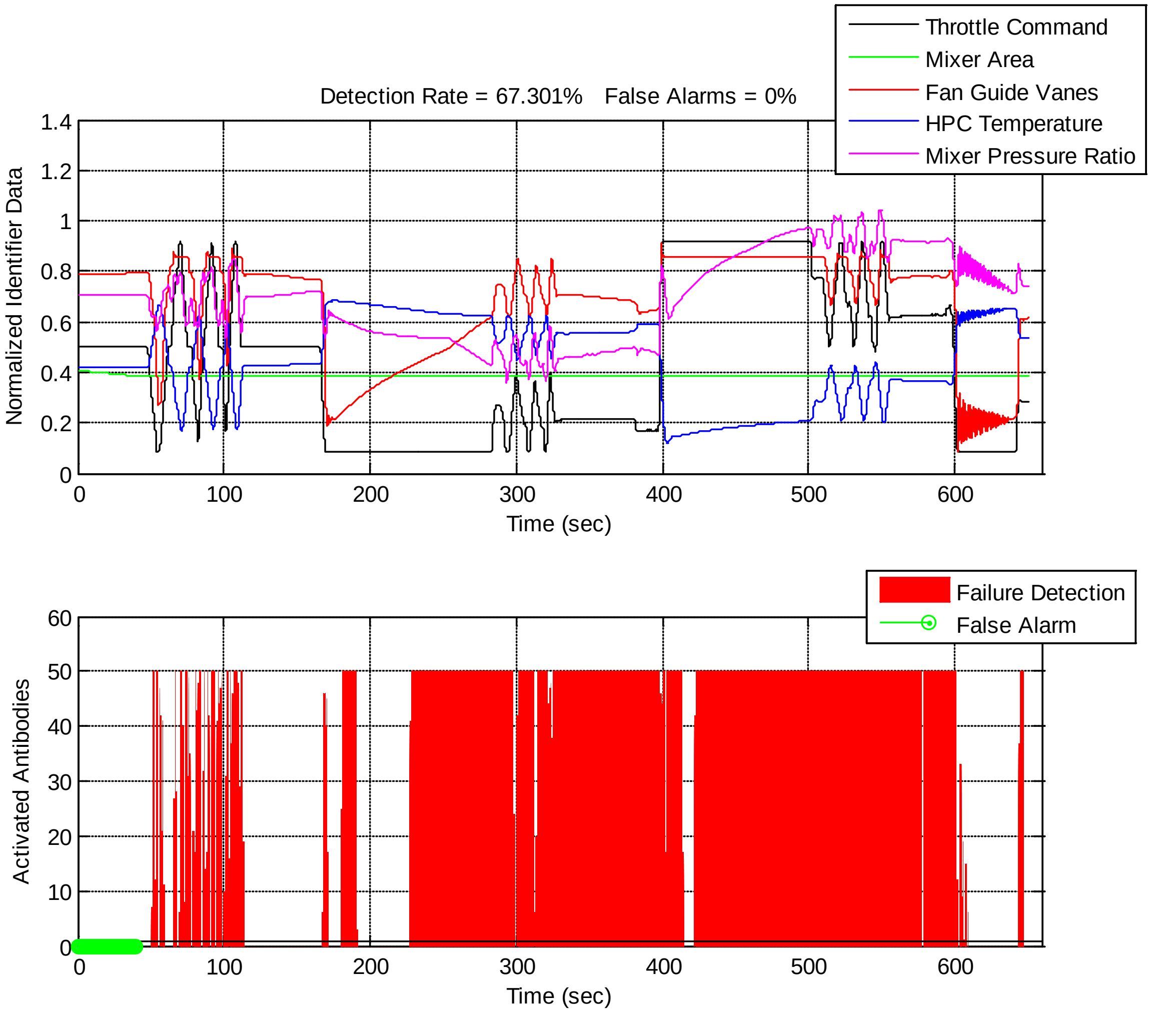


Figure 8.20: Left Engine Mixer Area Actuator Failure at 40sec for Flight Path 1 to 4 to 8

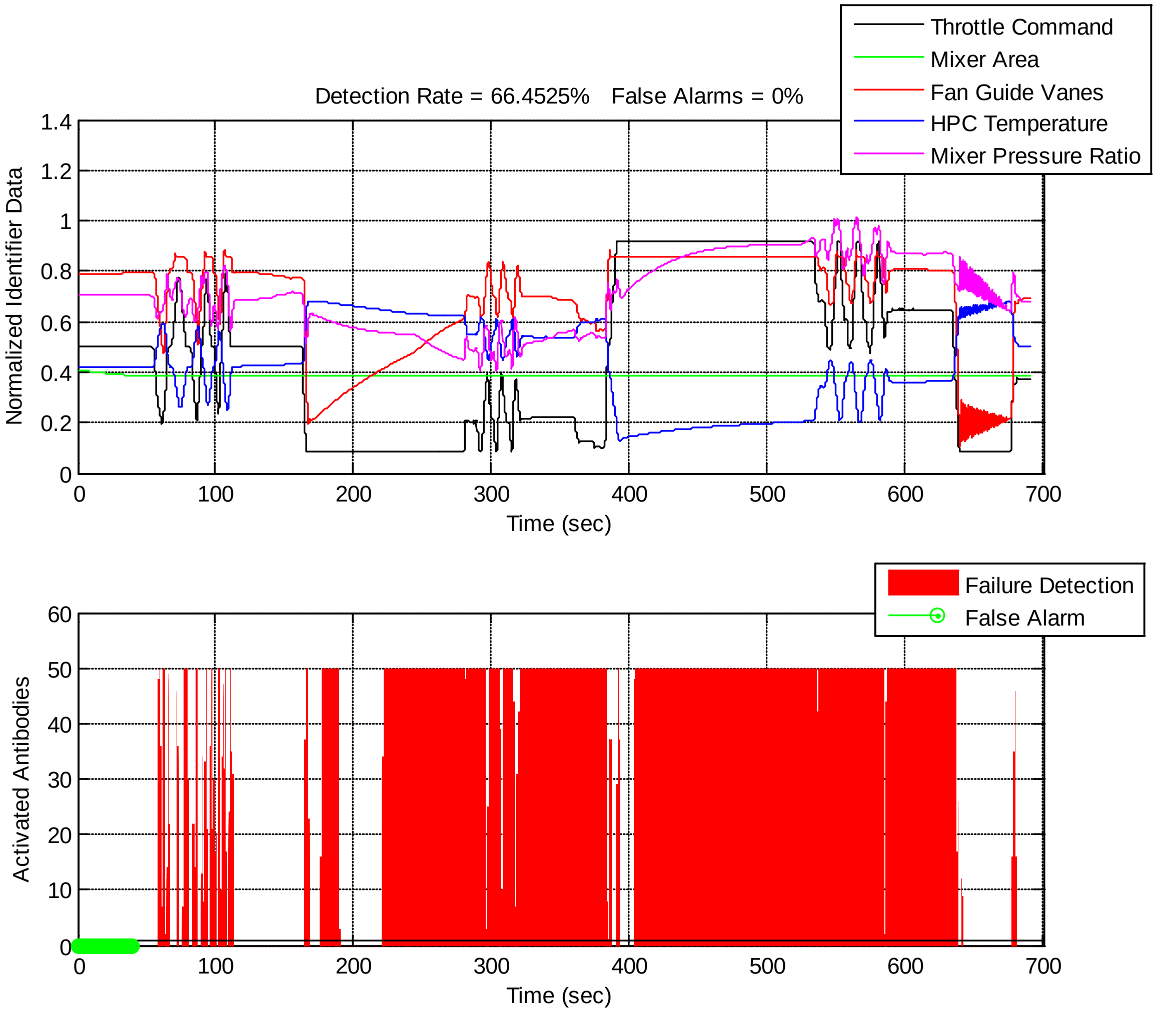


Figure 8.21: Right Engine Mixer Area Actuator Failure at 40sec for Flight Path 1 to 4 to 8

The mixer area failure applied to the flight path 1 to 2 to 6 and back to 1 has an increased detection rate at 89% and still no false alarms. Similar to the previous plots, Figure 8.22 has a decreased detection rate at point 1 until ascension to point 2 begins. This could mean that the mixer area does not change significantly from throttle doublets around point 1 and must be excited more by changing the Mach or altitude. Although it is desirable to detect the failure at the time of the occurrence, at least the system does detect the failure when it starts to affect the engine parameters.

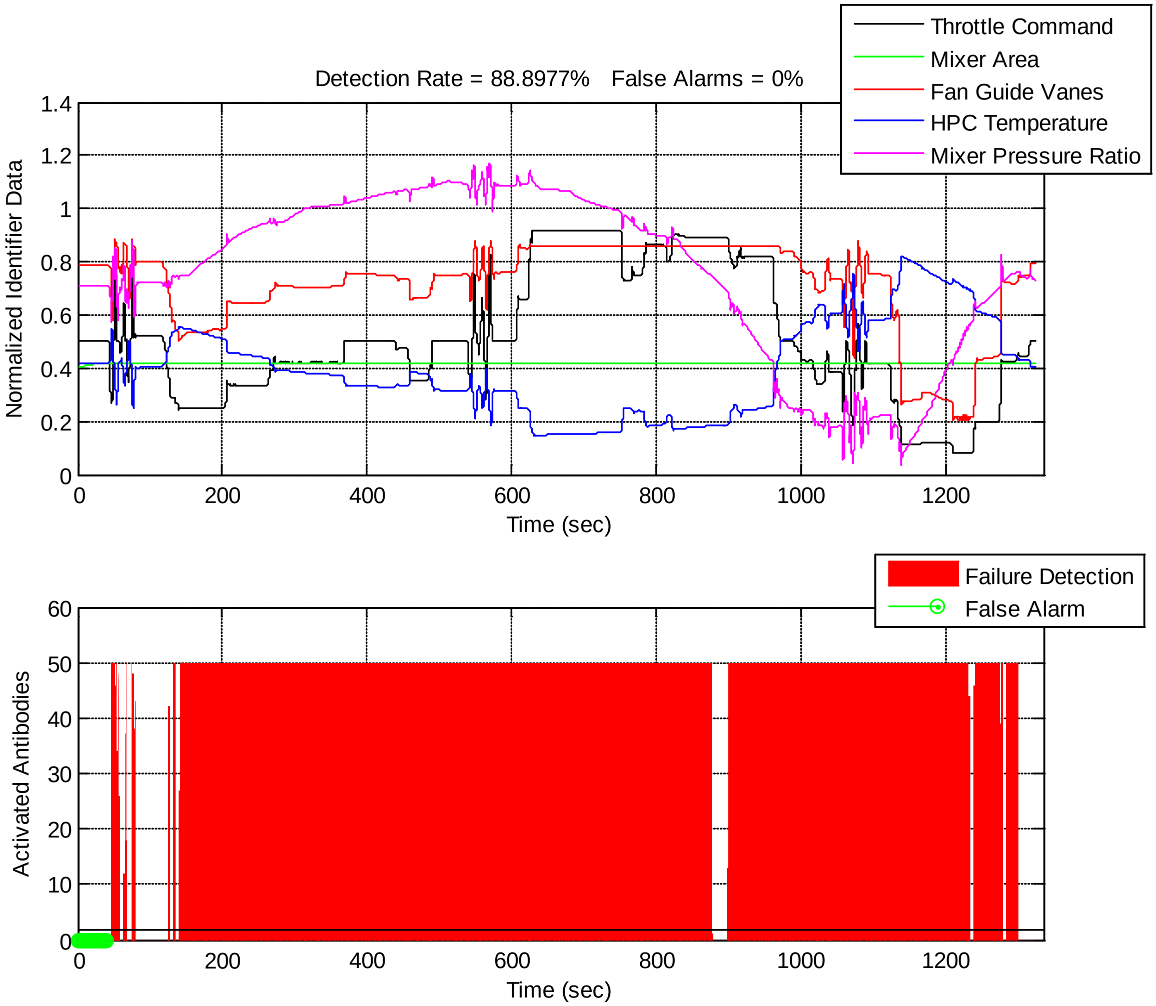


Figure 8.22: Left Engine Mixer Area Actuator Failure at 40sec for Flight Path 1 to 2 to 6

8.4 Engine Sensor Failures

The sensor failures chosen for flight testing were the LP spool speed locked at the minimum value; the LPT exit static pressure locked at zero and the minimum value; and the mixer pressure ratio locked at the maximum value. These failures were chosen because they had a visible effect of the engine parameters as shown in the Appendix.

8.4.1 LP Spool Speed Sensor Failure

The LP spool speed failure was performed on the left engine over two flight tests. The results are shown in Figure 8.23 and Figure 8.24. The detection rate for flight at points 1, 4, and 8 was 47% with zero false alarms. With this failure, the flight at point 1 did not seem to excite the system enough for good detection. Also, the section with poor detection from approximately 375sec to 700sec contains lower throttle values to fly to and at point 8.

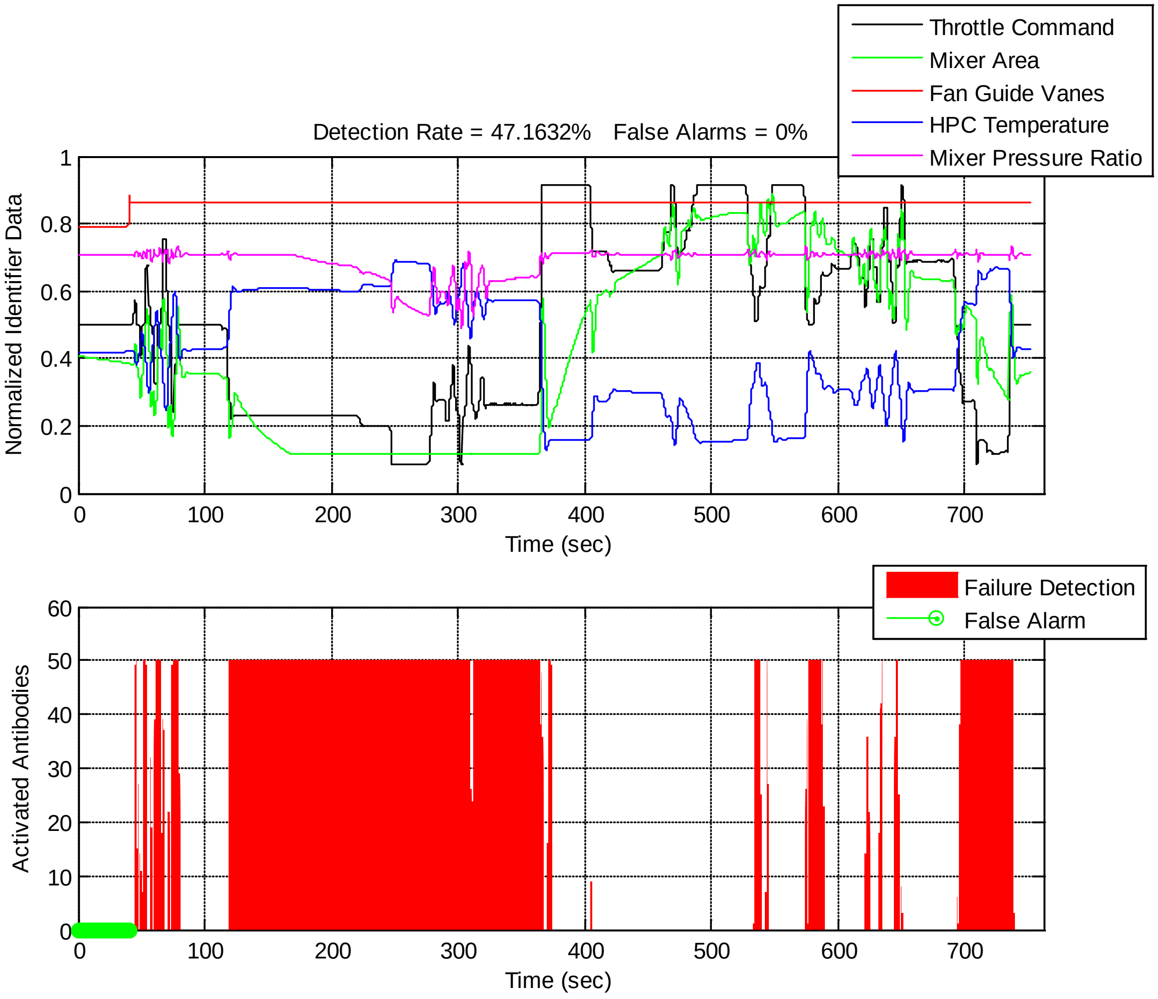


Figure 8.23: Left Engine LP Spool Speed Sensor Failure at 40 sec for Flight Path 1 to 4 to 8

The detection rate shown in Figure 8.24, which includes flight at points 2 and 6, is better at 62% with no false alarms. However, the same issues occur at point 1 and while the pilot descends using low throttle input.

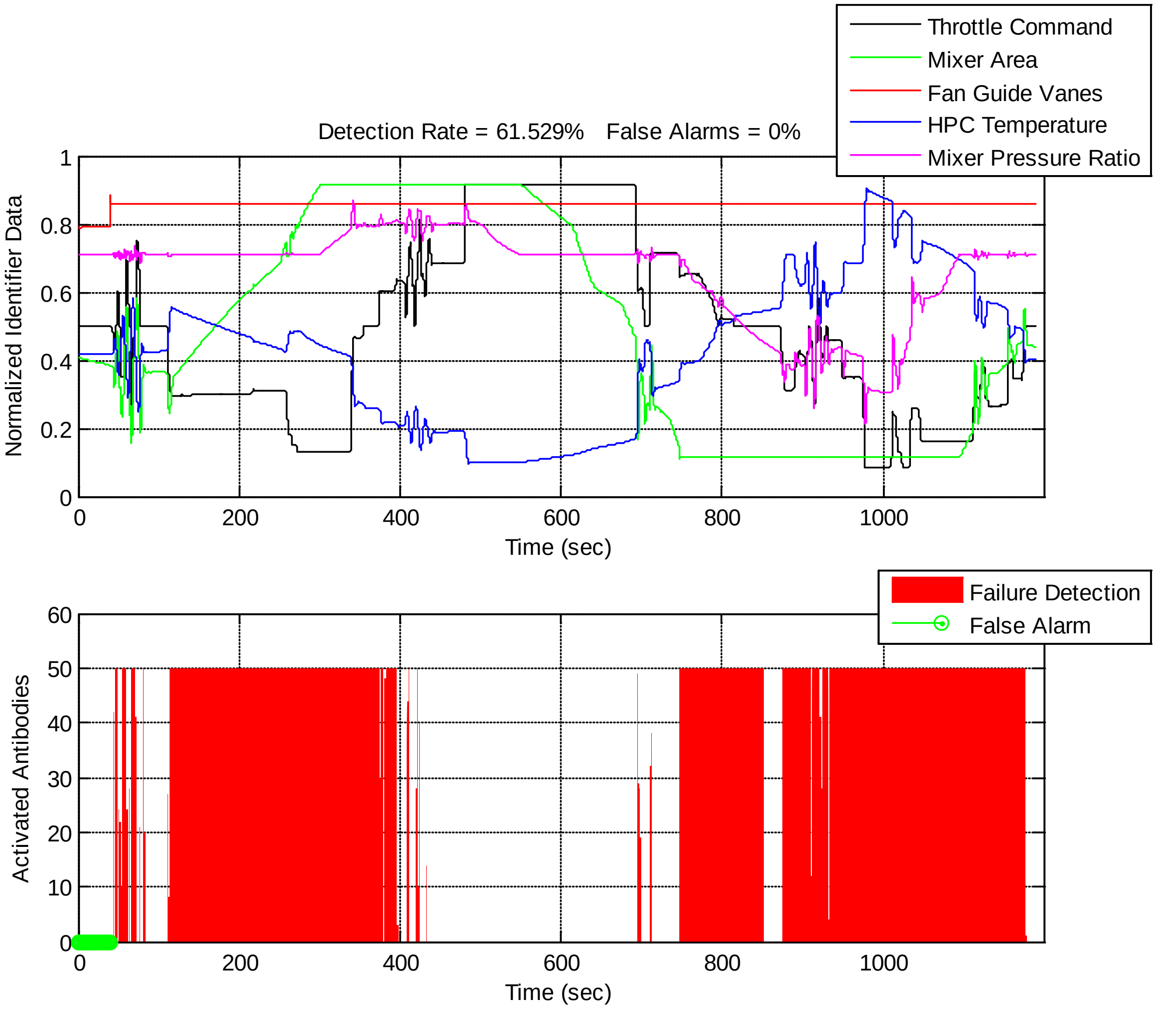


Figure 8.24: Left Engine LP Spool Speed Sensor Failure at 40sec for Flight Path 1 to 2 to 6

8.4.2 LPT Exit Static Pressure Sensor Failure

The next sensor to test was the LPT exit static pressure sensor. This failure was performed for both a zero constant value and the minimum constant value. Figure 8.25 and Figure 8.26 show the results of the zero constant failure. Figure 8.25 has a detection rate of 96% and false alarms of 0% for flight at points 1, 4, and 8. The flight test at points 1, 2, and 6 shown in Figure 8.26 has a smaller, but still very good, detection rate of 77% with no false alarms.

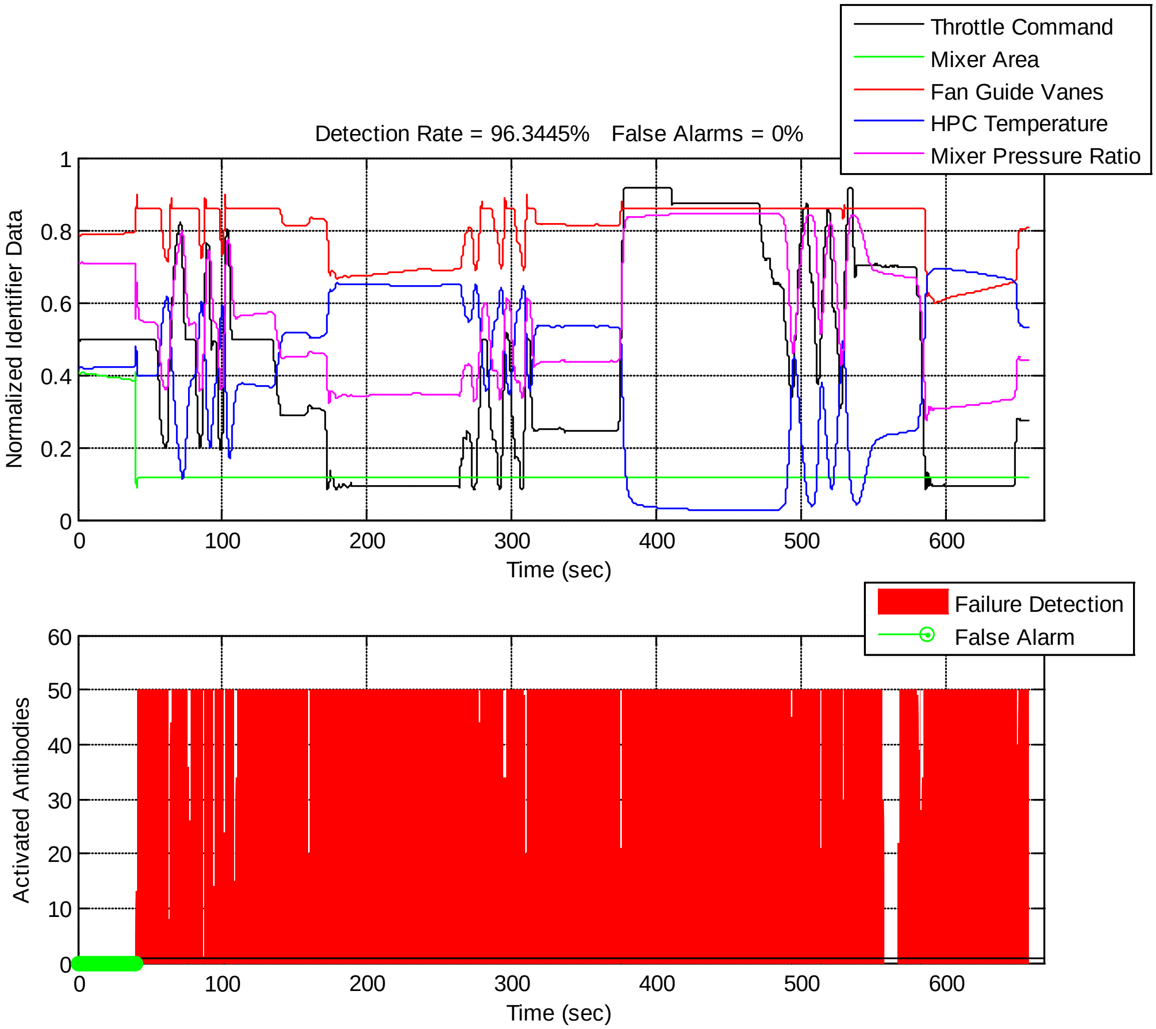


Figure 8.25: Left Engine LPT Exit Static Pressure Sensor Failure at Zero at 40sec for Flight Path 1 to 4 to 8

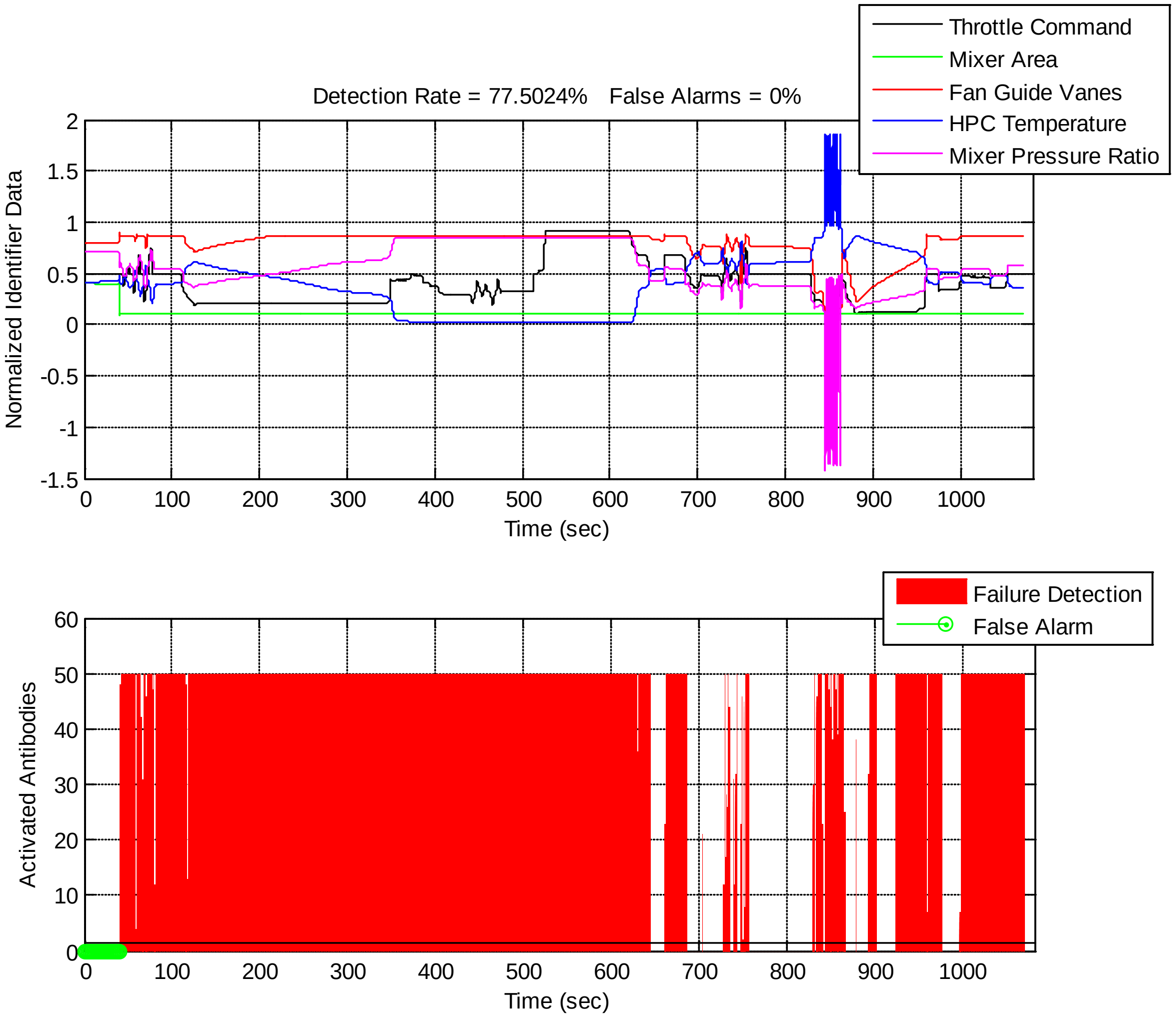


Figure 8.26: Left Engine LPT Exit Static Pressure Sensor Failure at Zero at 40sec for Flight Path 1 to 2 to 6

Figure 8.27 and Figure 8.28 contain the results of the LPT exit static pressure sensor at the minimum value. The first flight test has a detection rate of 90% and zero false alarms, while the second has a detection rate of 76% and no false alarms. These results are very similar to the zero sensor failure. For the lower rates in Figure 8.26 and Figure 8.28, the poor detection corresponds to flight at point 6.

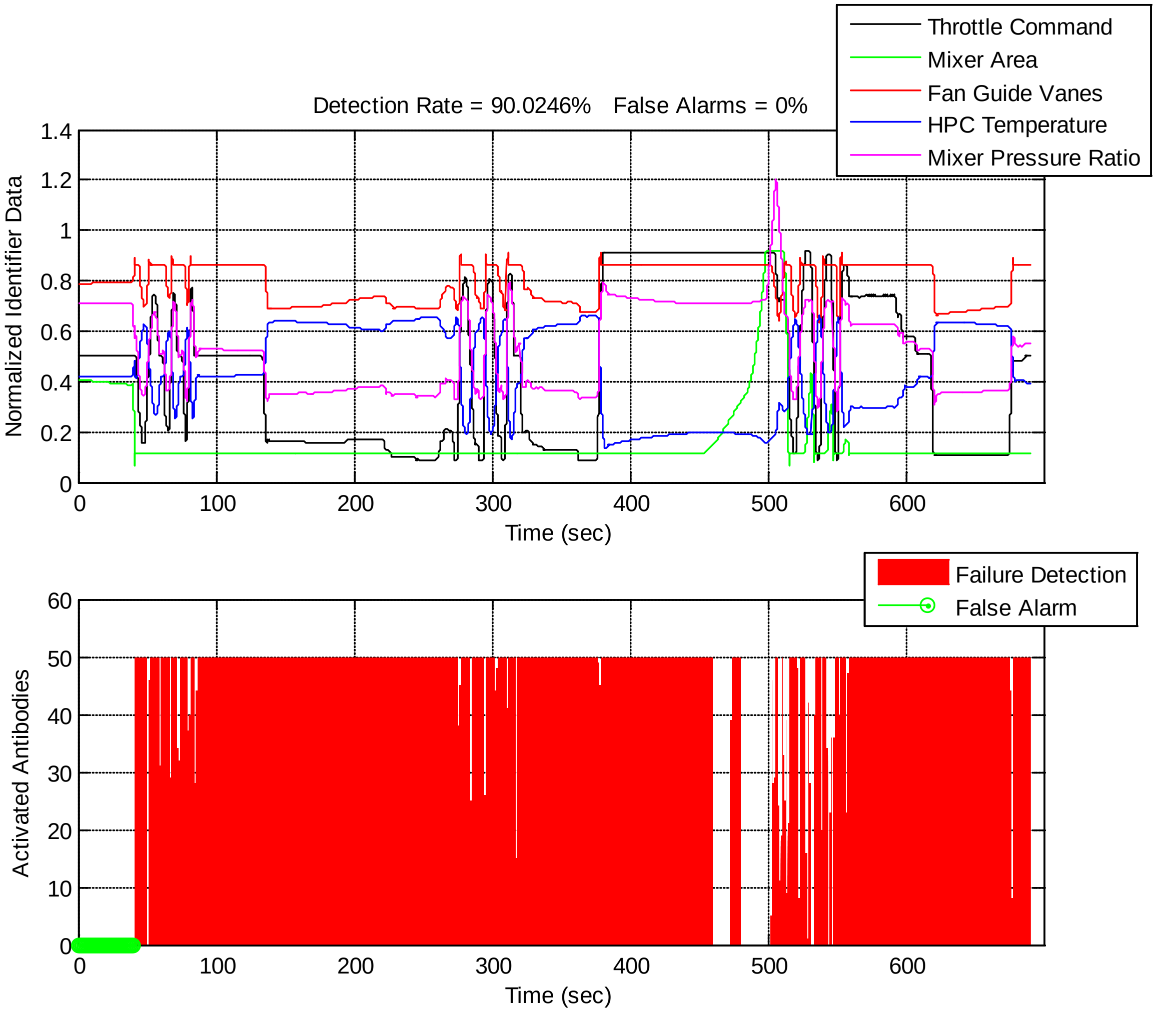


Figure 8.27: Left Engine LPT Exit Static Pressure Sensor Failure at Minimum at 40sec for Flight Path 1 to 4 to 8

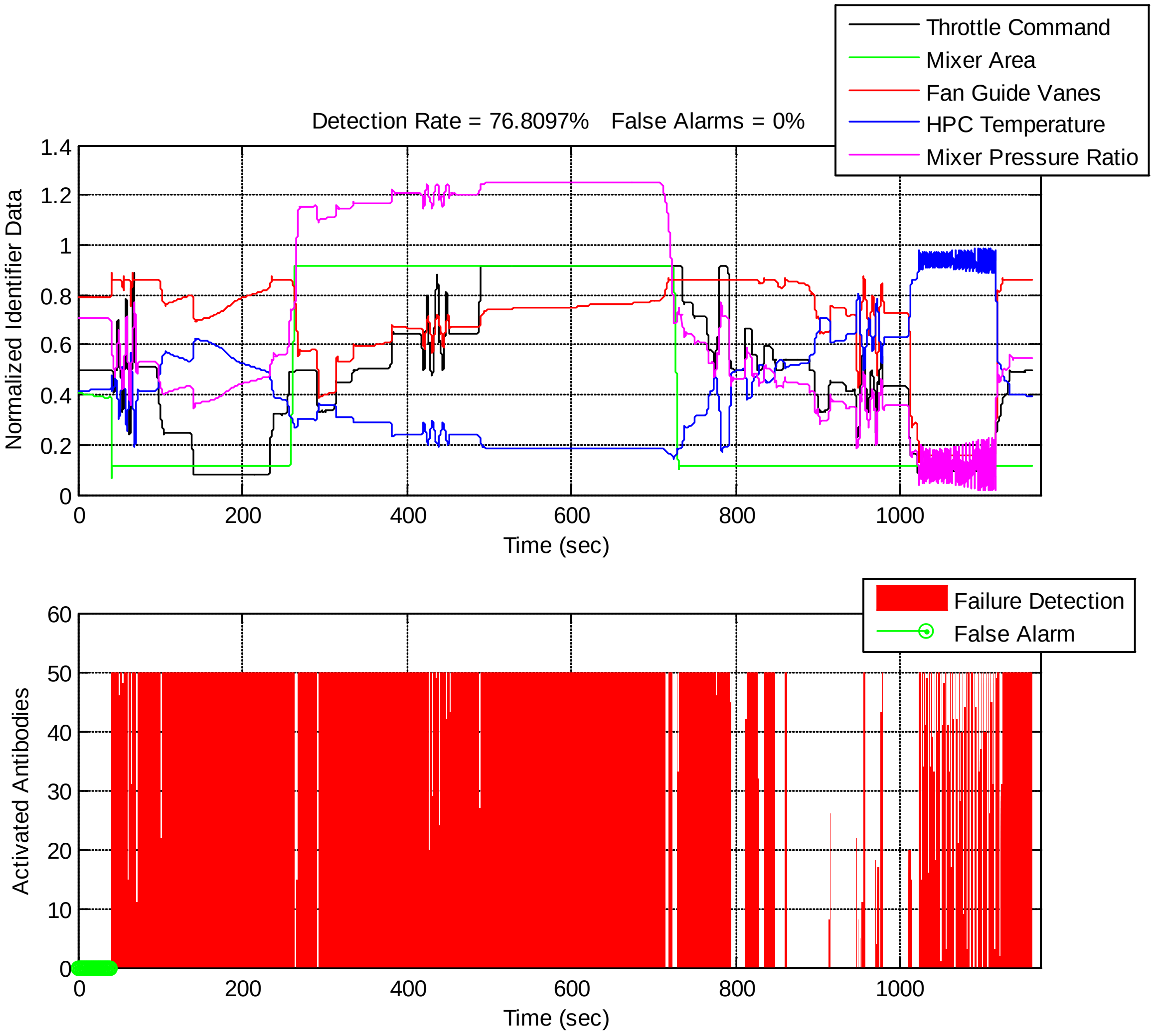


Figure 8.28: Left Engine LPT Exit Static Pressure Sensor Failure at Minimum at 40sec for Flight Path 1 to 2 to 6

8.4.3 Mixer Pressure Ratio Sensor Failure

The final tested failure is the mixer pressure ratio sensor at a constant maximum value found for operational point 1. The simulation of this failure suggested that it would have a significant effect on general operation of the engine, mainly because it is connected to so many of the other parameters – actuators and sensors. This was expected to allow for high detection rates. Figure 8.29 and Figure 8.30 illustrate the detection results. Figure 8.29 shows the flight test at points 1, 4, and 8 with a detection rate of 55% and 0 false alarms. The section showing poor detection is during deceleration to point 8 with a low throttle command. Figure 8.30 has no false alarms and a detection rate of 47%, where the poor detection occurs at decreasing or low throttle.

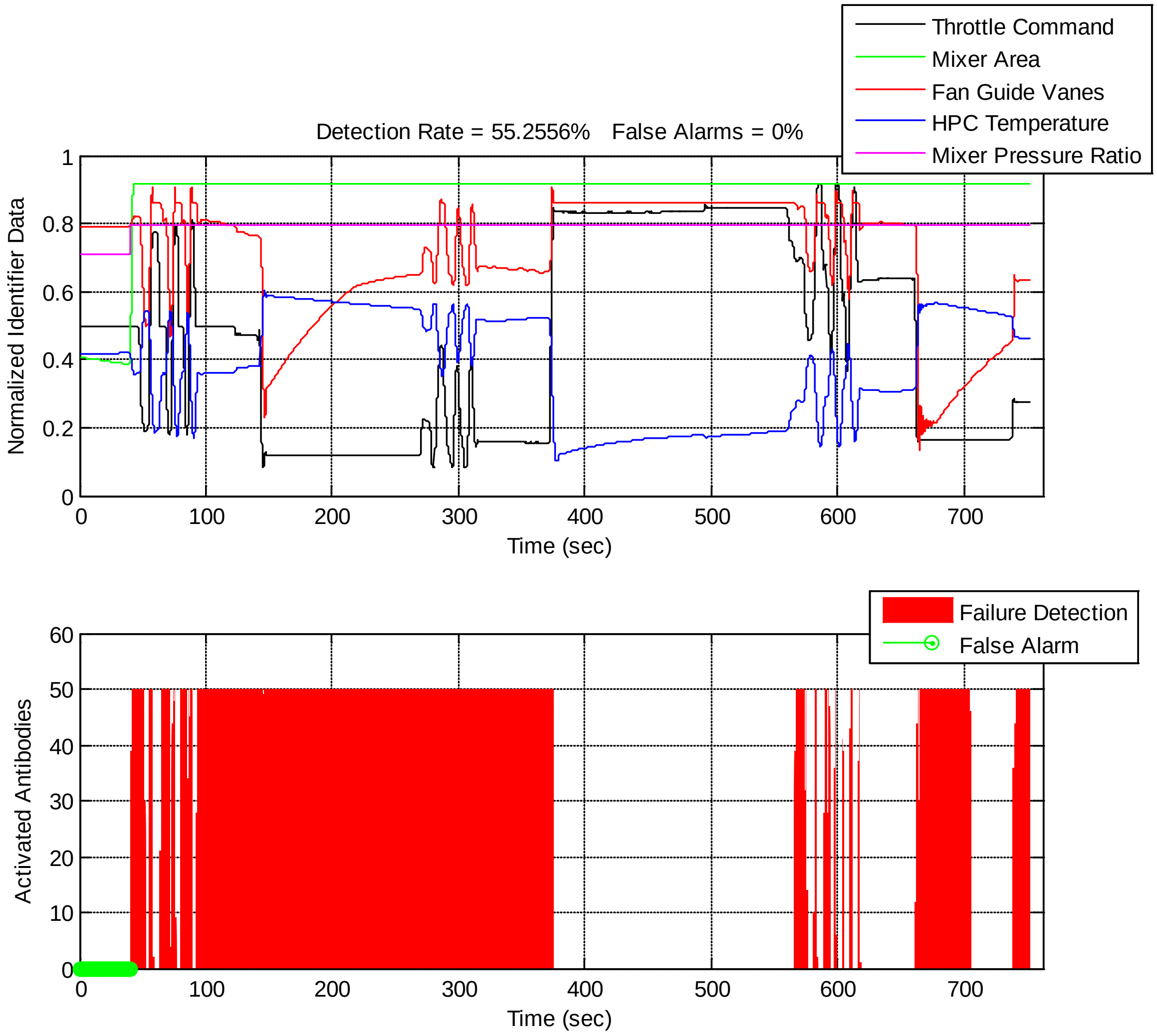


Figure 8.29: Left Engine Mixer Pressure Ratio Sensor Failure at 40sec for Flight Path 1 to 4 to 8

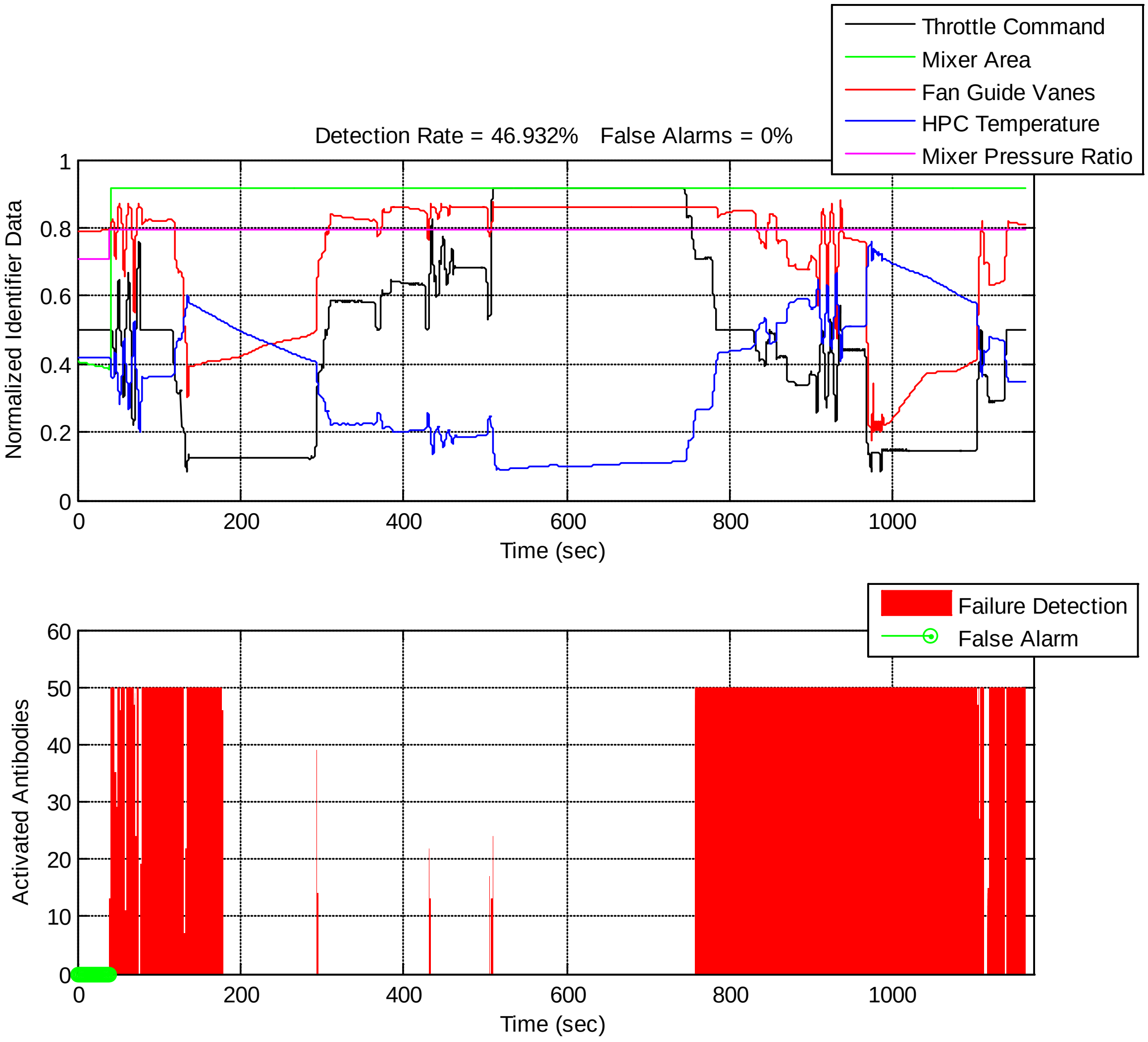


Figure 8.30: Left Engine Mixer Pressure Ratio Sensor Failure at 40sec for Flight Path 1 to 2 to 6

The explanation of the lower than expected detection rates for this failure is the fact that the mixer pressure ratio is used as a detector. The mixer pressure ratio is influenced by the most parameters in the system, which is why it was chosen as a detector. However, when the mixer pressure ratio sensor fails and its output is constant at this high value, its effectiveness is voided and the system has to rely on the other parameters to detect the failure. This is due to the fact that large portions of the normal operation take place in the vicinity of this maximum output value as can be seen in Figure 8.5 and Figure 8.6. Additionally, a second identifier, the mixer area saturates at a value that is close to the normal operational point. This places the regions of this failure very close to the self and exposed to erroneous inclusion into the self. It is expected that complete optimization of the detector set may succeed in better separating these regions from the self. Even without the optimization, the detection rates shown are still pretty good and detecting the failure immediately after its occurrence, as shown in Figure 8.30, is very important.

8.5 Overall Performance

To better compare results of the failures tested, all of the detection rates are tabulated in Table 8.1. As shown in the table, all of the results, with the exception of the nozzle area, have very good detection rates and low false alarms. Even the nozzle area has an acceptable rate over the tested flight envelope. The overall average of the false alarms including the validation data was 0.246% which is extremely low.

Table 8.1: Results of Average Rates for Each Left Engine Failure Across the Flight Envelope

	Nominal	Stuck Throttle at 0	Burner Fuel Flow Valve at 0	Stuck Nozzle Area	Stuck Mixer Area	LP Spool Speed Sensor at Minimum	LPT Exit Pressure Sensor at 0	LPT Exit Pressure Sensor at Minimum	Mixer Pressure Ratio Sensor at Maximum
Average Detection Rate	N/A	83.012%	99.959%	30.908%	78.099%	54.346%	86.924%	83.417%	51.094%
Average FalseAlarm Rate	0.535%	1.434%	0.246%	0%	0%	0%	0%	0%	0%

Some of the failure detection results can now be compared to the results from previous research [32]. Table 8.2 shows the detection rates and false alarms for 6 different selves tested as well as for the self developed for this thesis. The results for Self#1 through Self#6 are for a 98% engine failure for flight from 1 to 2 to 3. These selves are based on aircraft dynamic measurements, artificial neural network state estimates, tracking errors, and artificial neural network compensation produced by the fault tolerant adaptive control laws because no internal engine parameters were yet modeled. The results for the new self include engine throttle failure and engine burner fuel flow valve failure averaged over their flights for 1 to 2 to 6 and 1 to 4 to 8.

Table 8.2: Comparison of Previous and Current Engine Model Failure Detection Rates [32]

Engine Failure Type	Self Configuration	# of Self Dimensions	Number of Detectors	Engine Failure Test Data Detection Rates (%)	Nominal Test Data False Alarms (%)
98% Reduced Efficiency	Self#1	3	516	80.62	19.59
	Self#2	6	504	68.12	0.06
	Self#3	6	507	44.13	0.56
	Self#4	8	504	59.8	2.5
	Self#5	3	507	0.84	0.78
	Self#6	4	515	2.73	5.83
Throttle Failure	New Engine Self	5	991	83.01	1.43
Fuel Flow Failure	New Engine Self	5	991	99.96	0.25

It is important to mention the differences and subsequent effects of the previous and current research results. The number of detectors for the new AIS is approximately double that of the former selves; however, an analysis in previous research showed that no significant improvement was obtained by increasing the number of detectors in the 6 selves. The detector set generated for the new engine model is only partially optimized; meaning further efforts using the WVU AIS interactive design environment can improve the failure detection results.

The best two self sets from the 98% reduced efficiency engine failure were self #1 and self #2. Self #1 had a high detection rate but also high false alarms while self #2 had low false alarms and an acceptable detection rate. The strength of the engine failures is important when comparing the detection results. Since the throttle failure is approximately equal to an 87% reduced efficiency engine failure, it is a softer failure than 98% reduced efficiency, i.e. should be harder to detect. The throttle failure average detection rate was 83% only slightly higher than the 81% of self #1, but the false alarms for the throttle failure were much smaller. Compared to self #2, the throttle failure has a much higher detection rate and close false alarms. The engine burner fuel flow failure is stronger at about 113% reduced efficiency, but the detection rate is much higher at 100% with very good false alarms. By comparing the throttle failure to the engine burner fuel flow failure, it is apparent that reducing the strength of the failure does not reduce the detection rate drastically. Overall, the self generated from the new engine model shows better detection rates and very good false alarms, and even though the engines models are slightly different, the general conclusions are the same.

Chapter 9: Conclusions

MAPPS was used to develop a linear engine model that allows for flexibility in modeling and detecting a variety of engine malfunctions. Integrated with the WVU F-15 model, a user is capable of implementing new advanced engine failures in addition to previously modeled engine failures. The failures include 7 engine actuators and 8 engine sensors each with several options for the failure scenario.

Real-time flight data was obtained from the WVU F-15 model and the WVU 6-DOF flight simulator where pilots performed typical maneuvers within the defined flight envelope. While the linear engine model was valid for these experiments, some effort to create a more advance linear model effective over a larger flight envelope and/or with full power capabilities could be beneficial.

The WVU AIS design environment was used to develop a detection scheme which tested the experimental flight data for engine failures. The detection scheme developed from internal engine parameters demonstrated improvement compared to previous results. Some failures produced excellent detection rates and false alarms while still others had very good results. Also, the same level of robustness of the data acquisition process has been confirmed for propulsion system malfunction as previously determined for other aircraft sub-systems. Results could be further improved by complete optimization of detector sets, increase of the amount of training data, and/or additional features.

References

1. Lichtsinder, M., Yeshayahou, L., “Jet Engine Model for Control and Real-Time Simulations”, *ASME Journal of Engineering for Gas Turbines,* Vol. 128, pp 745-53 Oct. 2006

2. Bruzelius, F., Breitnolz, C., Pettersson, S., (2002). “LPV-based Gain Scheduling Technique Applied to a Turbofan Engine Model”, Proc. IEEE Int. Conf. Control Applications, Glasgow, Scotland, UK.

3. Frederick, D. K., Garg, S., Adibhatla, S., (2000). “Turbofan Engine Control Design Using Robust Multivariable Control Techniques”, *IEEE Trans. Control Syst. Tech.,* 8:961-970.

4. Henrion, Reberga, Bernussou, Vary F, “Linearization and Identification of Aircraft Turbofan Engine Models”, IFAC, 2004

5. Parker, Khary I., Guo, Ten-Heui, “Development of a Turbofan Engine Simulation in a Graphical Simulation Environment”, NASA Technical Memorandum 2003-212543

6. Parker, Khary, I., Melcher, Kevin J., “The Modular Aero-Propulsion System Simulation (MAPSS) Users’ Guide”, NASA Technical Memorandum 2004-212968

7. Moncayo, H., Perhinschi, M., Davis, J., “Immunity-Based Aircraft Failure Detection and Identification Using an Integrated Hierarchical Multi-Self Strategy”, Proc. of AIAA Guidance, Navigation, and Control Conference, Chicago, IL., Aug. 2009

8. Sarkar, S., Yasar, M., Gupta, S., Ray, A., Mukherjee, K., “Fault detection and isolation in aircraft gas turbine engines. Part 2: validation on a simulation test bed”, Proc. IMechE Vol. 222 Part G: J. Aerospace Engineering, 2008

9. Gupta, S., Ray, A., Sarkar, S., Yasar, M., “Fault detection and isolation in aircraft gas turbine engines: Part 1: underlying concept”, Proc. IMechE Vol. 222 Part G: J. Aerospace Engineering, 2008

10. Perhinschi M. G., Napolitano M.R., Campa G., “A Simulation Environment for Design and Testing of Aircraft Adaptive Fault-Tolerant Control Systems”, *Aircraft Engineering and Aerospace Technology: An International Journal*, Vol. 80, Iss. 6 pp 620-632, Dec. 2008

11. Rauw, M.O. (1998), “FDC 1.2 – A SIMULINK Toolbox for Flight Dynamics and Control Analysis”, Delft University of Technology, Delft, The Netherlands

12. Rassmussen, S. (2000), "*Aviator Visual Design Simulator (AVDS) – User Manual*", Rassmussen Simulation Technologies, Ltd

13. Meyer A., Van Kampen H., "X-Plane On-Line Instruction Manual", Laminar Research Inc., Columbia SC, 8th edition, 2002 (www.X-Plane.com)

14. Perhinschi M. G., Napolitano M. R., "Teaching Aircraft Health Management - A Simulation-Based Approach", scheduled for publication in *Computers in Education Journal*, Oct.-Dec., 2009

15. Antoniewicz, R.F., Duke, E.L. and Patterson, B.P. (1988), "User's Manual for Interactive LINEAR, a Fortran Program to Derive Linear Aircraft Models", NASA Technical Paper 2835

16. Perhinschi M. G., Campa G., Napolitano M.R., Lando M., Massotti L., Fravolini M. L., "Modeling and Simulation of a Fault Tolerant Control System", *International Journal of Modelling and Simulation*, vol. 26, no. 1, pp. 1-10, Jan. 2006

17. Benson, Ted (editor), "Turbofan Engine", July 11, 2008, NASA Glenn Research Center, http://www.grc.nasa.gov/WWW/K-12/airplane/aturbf.html, Retrieved Nov. 2, 2009.

18. Benjamini, E., "Immunology, A Short Course", Wiley-Liss Publications, New York, NY, 1992.

19. Zhi-tang, L., Yao, L., Li, W. (2005), "A Novel Fuzzy Anomaly Detection Algorithm Based on Artificial Immune System", Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05).

20. Dasgupta, D., and Majumdar, N., "Anomaly Detection in Multidimensional Data Using Negative Selection Algorithm", *Proceedings of the Congress on Evolutionary Computation CEC '02*, Vol. 02, 2002, pp 1039-1044.

21. De Castro, L., Timmis, J. (2002), "Artificial Immune Systems: A Novel Paradigm to Pattern Recognition", In Artificial Neural Networks in Pattern Recognition , J. M. Corchado, L. Alonso, and C. Fyfe (eds.), SOCO-2002, University of Paisley, UK, pp. 67-84, 2002.

22. Dasgupta, D., Nino, F. (2000), "Comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection", In the Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), Nashville.

23. Dasgupta, D., Forrest, S. (1996), "Novelty Detection in Time Series Data Using Ideas from Immunology", Proceedings of the International Conference on Intelligent Systems, Reno, Nevada

24. Forrest S, Perelson AS, Allen L, Cherukuri R. (1994), "Self-nonself discrimination in a computer", Proc. of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, pp. 202–212.

25. Gonzalez, F., Dasgupta, D., Kizma, R.(2002), "An Immunogenetic Technique to Detect Anomalies in Network Traffic", GECCO 2002: 1081-1088

26. Karr, C., Nishita, K., Kenneth S. (2005), "Adaptive Aircraft Flight Control Simulation Based on an Artificial Immune System", Applied Intelligence 23, 295–308.

27. Ko, A., Lau, H., Lau, T. (2004), "An Immuno Control Framework for Decentralized Mechatronic Control", ICARIS 2004, LNCS 3239, pp. 91–105.

28. KrishnaKumar, K., "Artificial Immune System Approaches for Aerospace Applications", *Proceedings of the 41st Aerospace Sciences Meeting & Exhibit*, AIAA-2003-0457, Reno, Nevada, 2003.

29. Dasgupta, D., KrishnaKumar, K., Wong, D., and Berry, M., "Negative Selection Algorithm for Aircraft Fault Detection", G. Nicosia et al. (Eds.): *Proceedings from the 3rd International Conference on Artificial Immune Systems 2004*, Catania, Sicily, Italy, September 13-16, 2004, pp. 1–13.

30. Perhinschi M. G., Moncayo H., Davis J., "Integrated Framework for Aircraft Sub-System Failure Detection, Identification, and Evaluation Based on the Artificial Immune System Paradigm", Proc. of the *AIAA Guidance, Navigation, and Control Conference*, Chicago, IL, August 2009

31. Davis J., Perhinschi M. G., Moncayo H., "Evolutionary Algorithm for Artificial Immune System-Based Failure Detector Generation and Optimization", accepted for publication in *AIAA Journal of Guidance, Control, and Dynamics*, Oct. 2009

32. Moncayo, Hever, "Immunity-Based Detection, Identification, and Evaluation of Aircraft Sub-System Failures", Ph.D. Dissertation, Dept. of Mechanical and Aerospace Engineering, West Virginia University, 2009

Appendix A

This appendix contains several more examples for engine failure responses. Each failure has the same standard throttle input with the left failure injected at 5sec. The figures for each failure follow.

Left Engine Power Lever Stuck at 0:

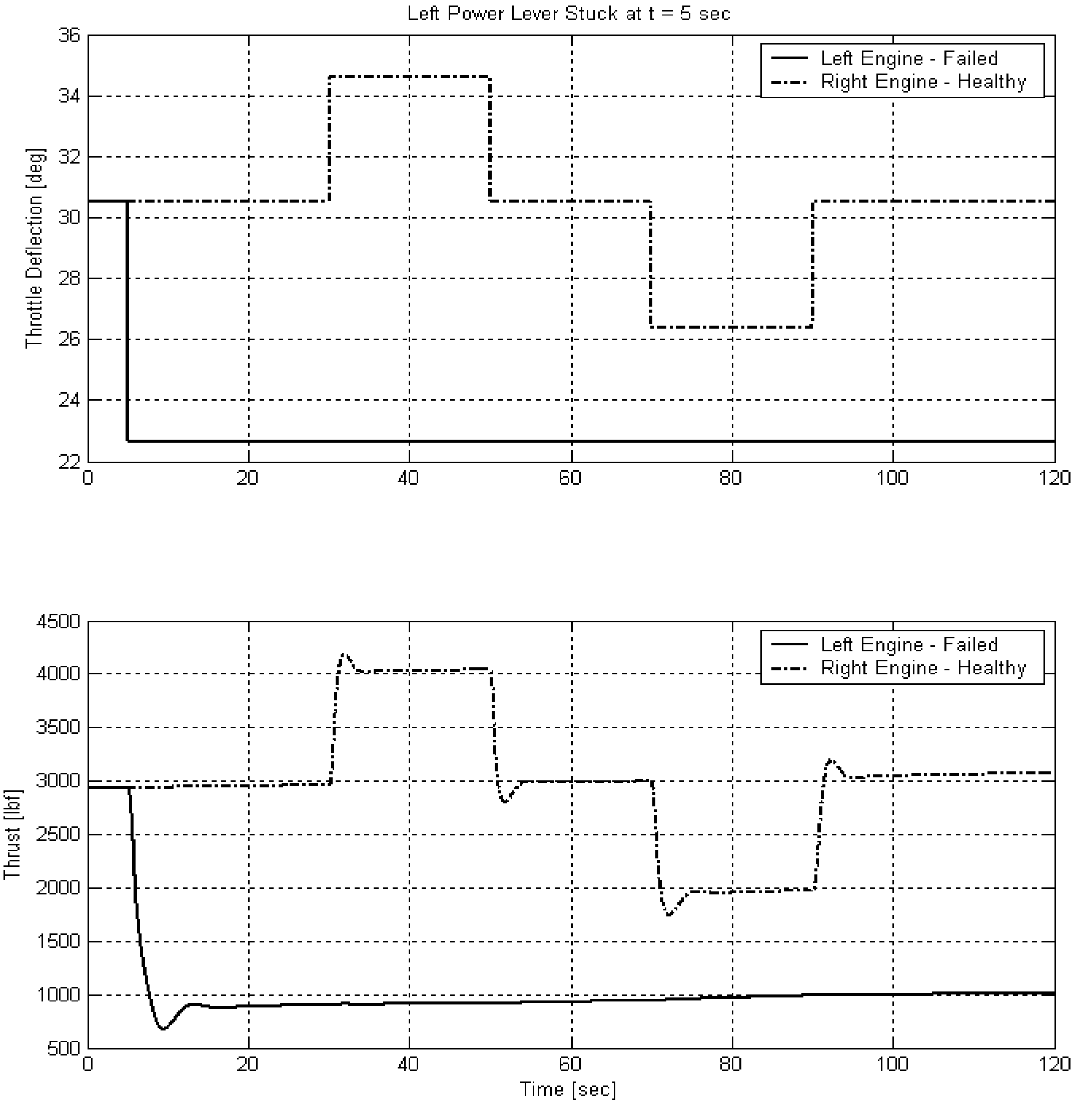


Figure A.1: Standard Throttle Input Response with a Stuck Throttle Failure (1)

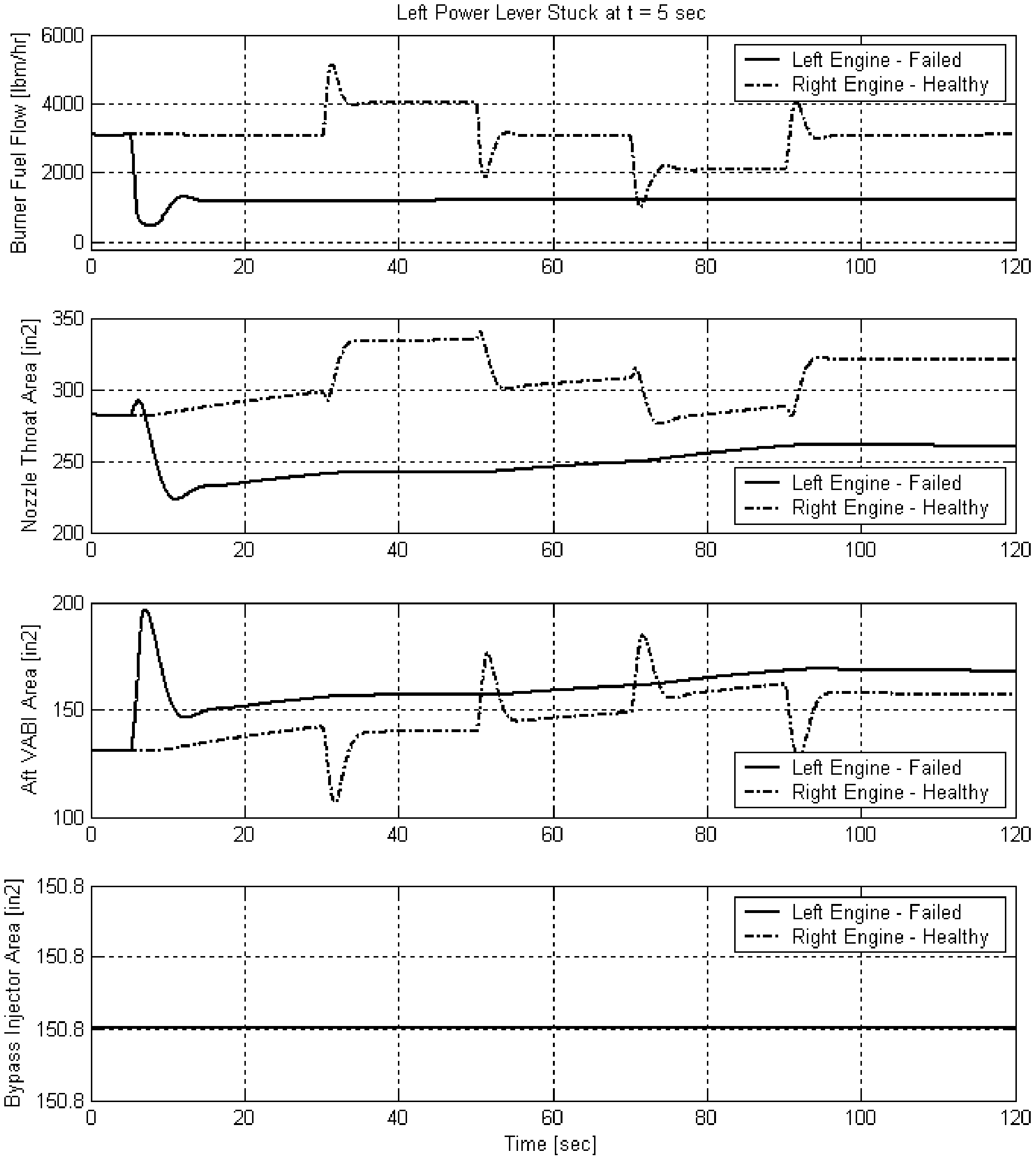


Figure A.2: Standard Throttle Input Response with a Stuck Throttle Failure (2)

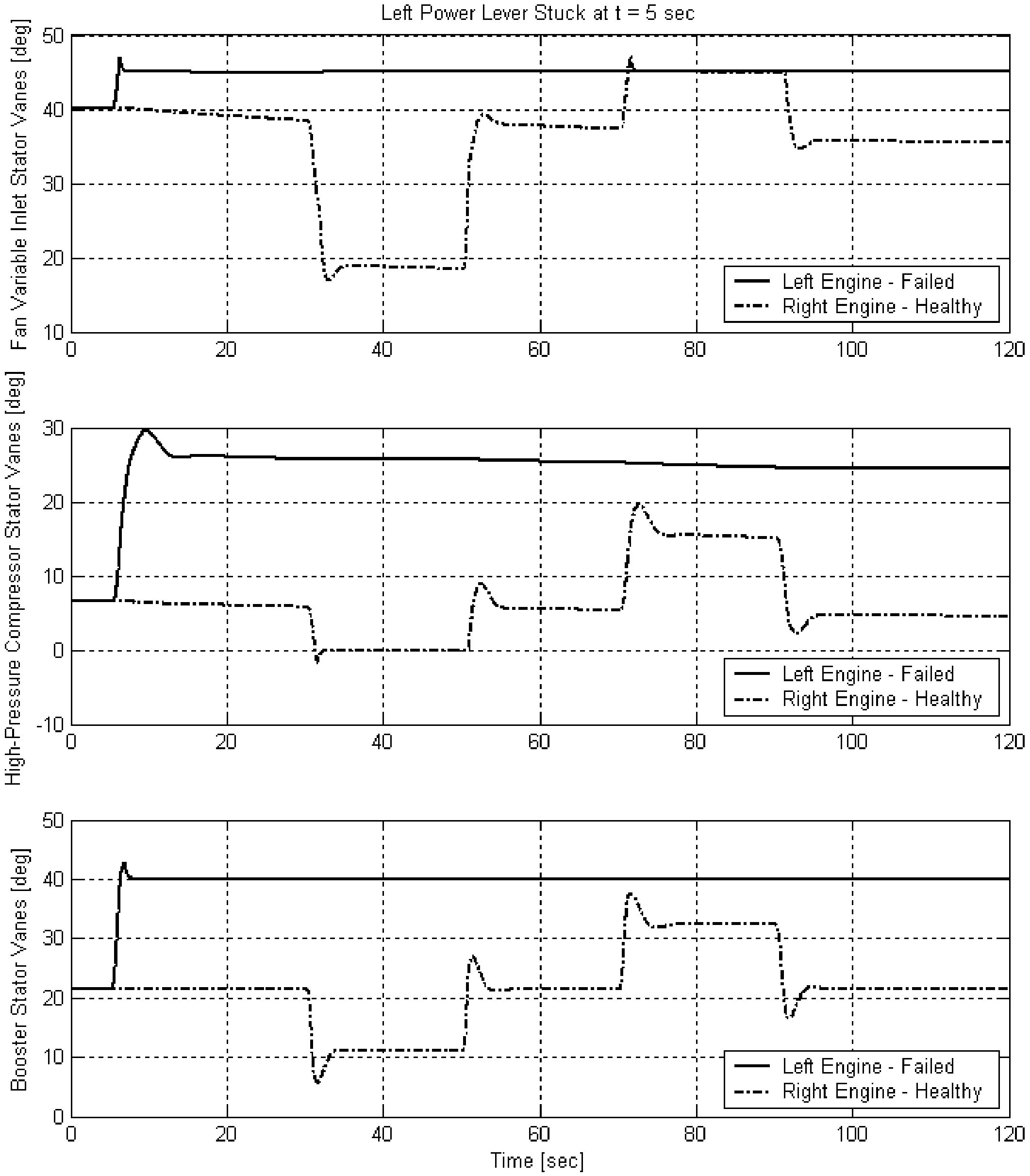


Figure A.3: Standard Throttle Input Response with a Stuck Throttle Failure (3)

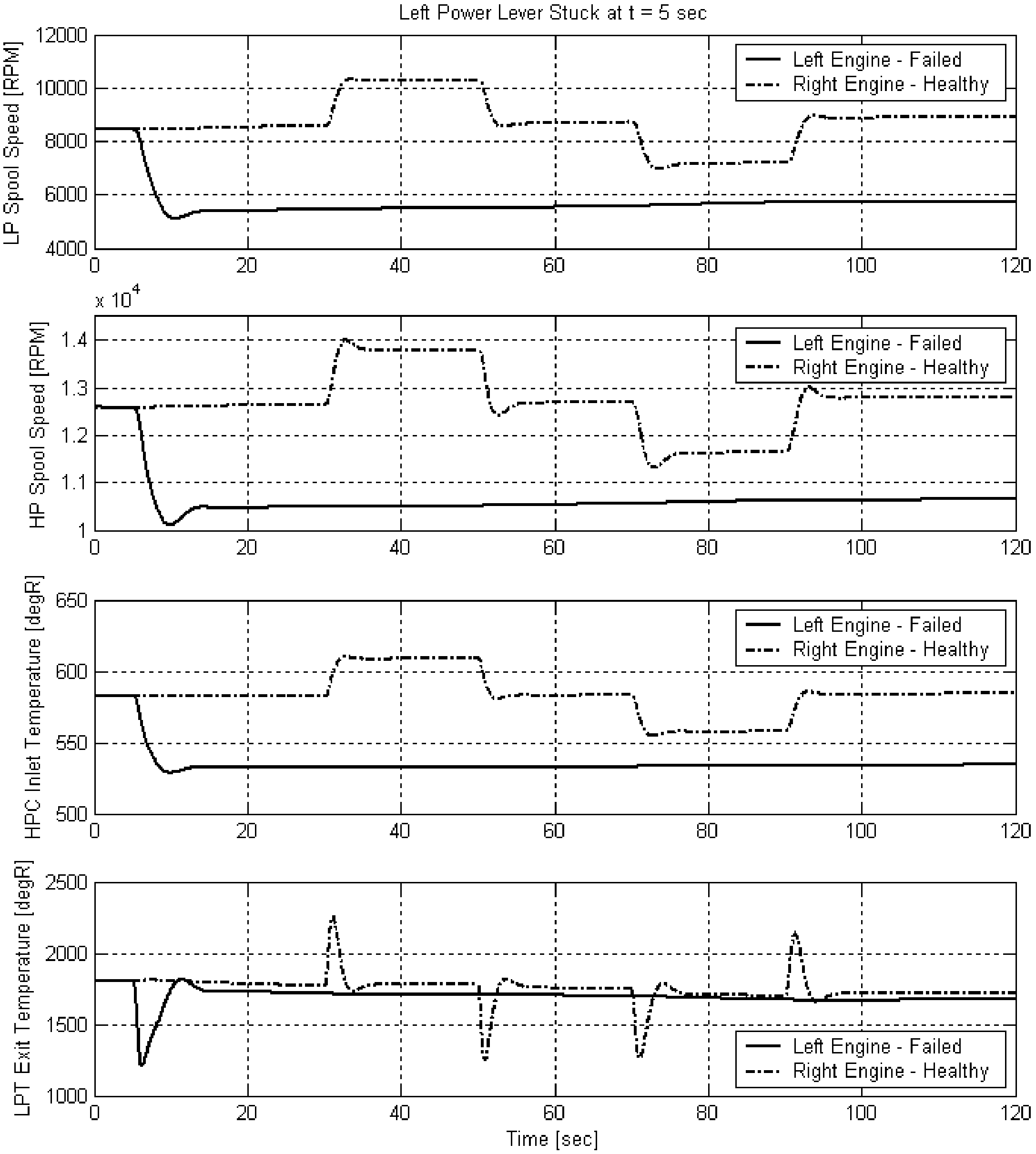


Figure A.4: Standard Throttle Input Response with a Stuck Throttle Failure (4)

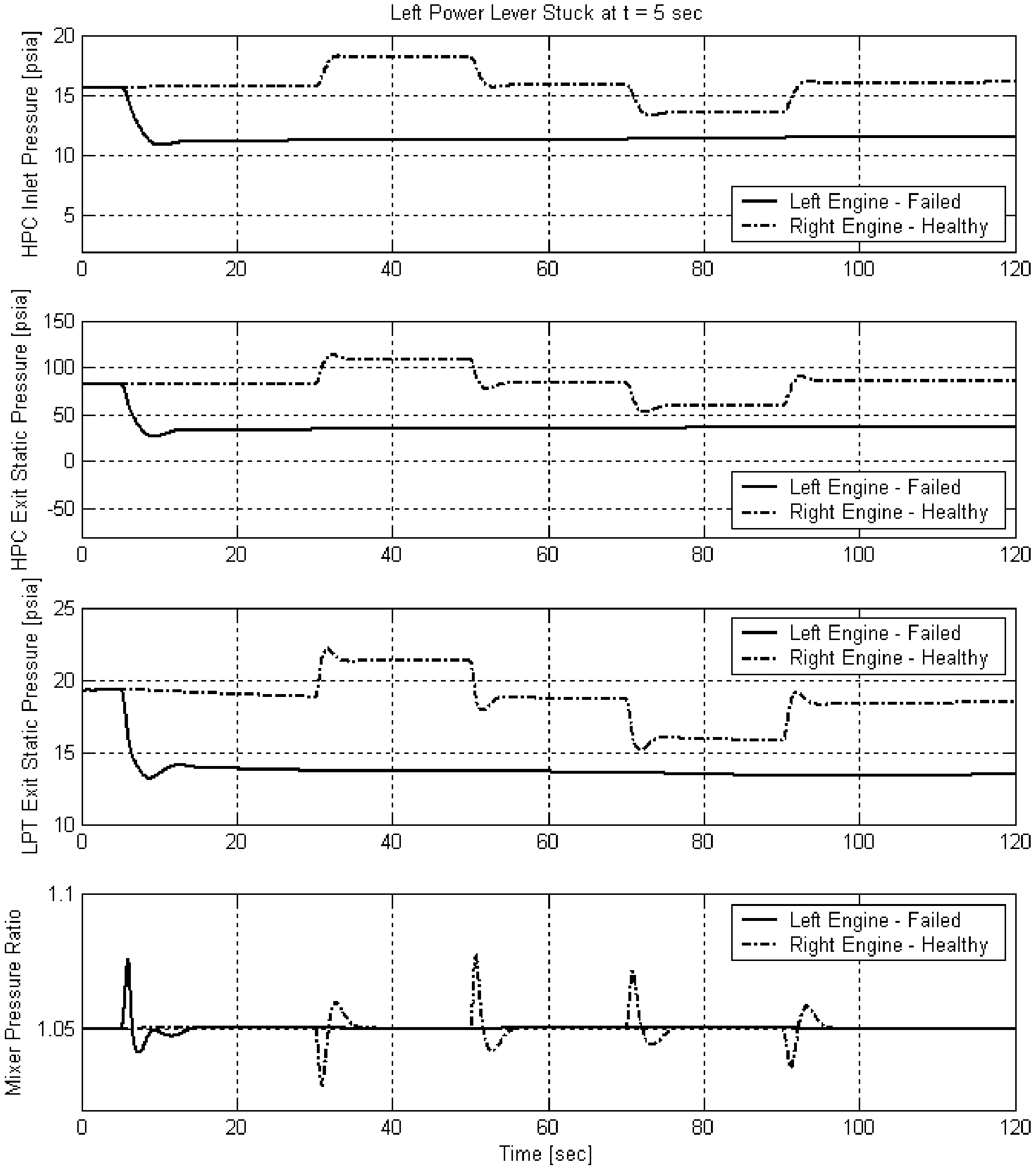


Figure A.5: Standard Throttle Input Response with a Stuck Throttle Failure (5)

Left Engine Burner Fuel Flow at 0lb$_m$/hr:

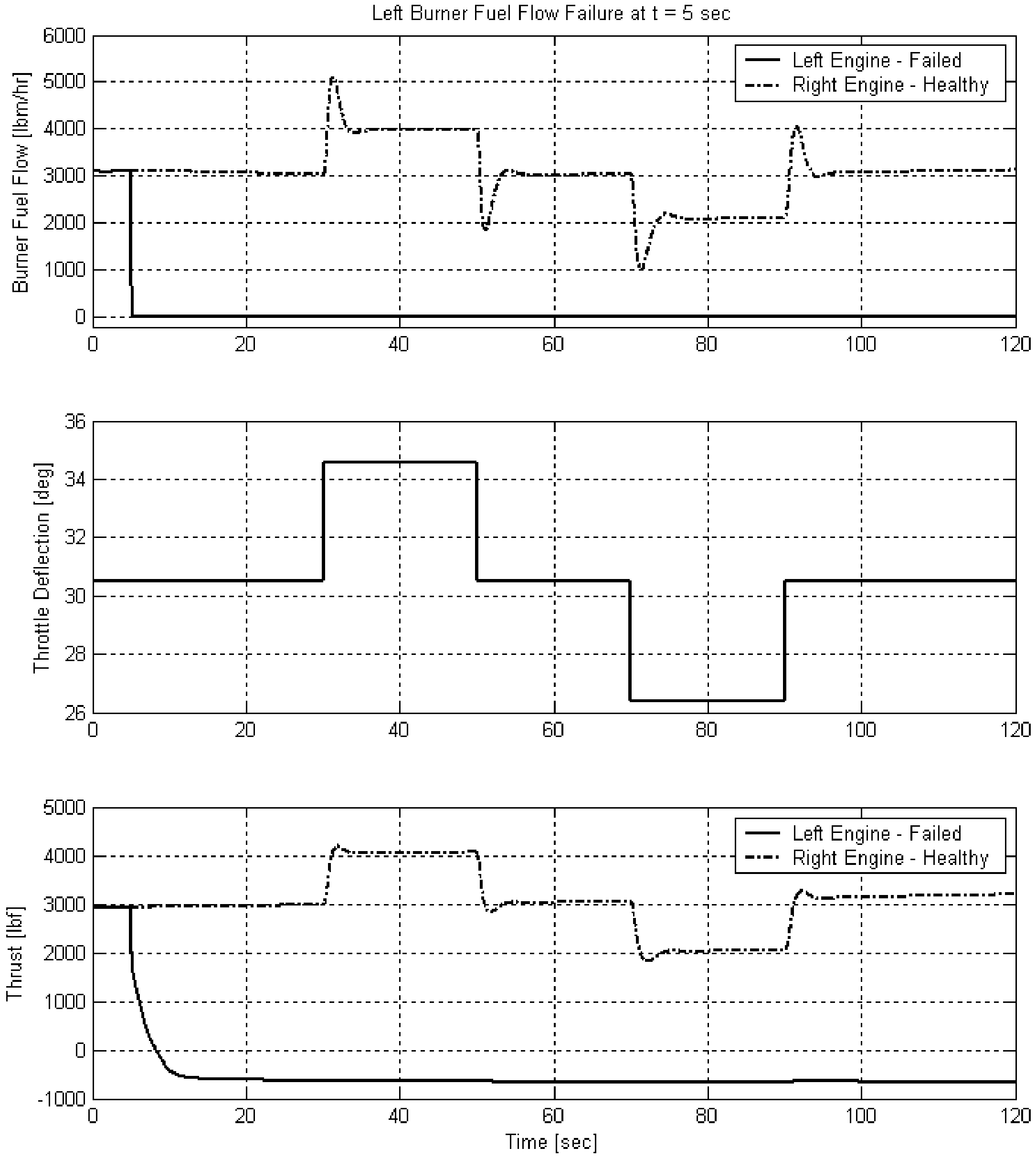


Figure A.6: Standard Throttle Input Response with a Burner Fuel Flow Failure (1)

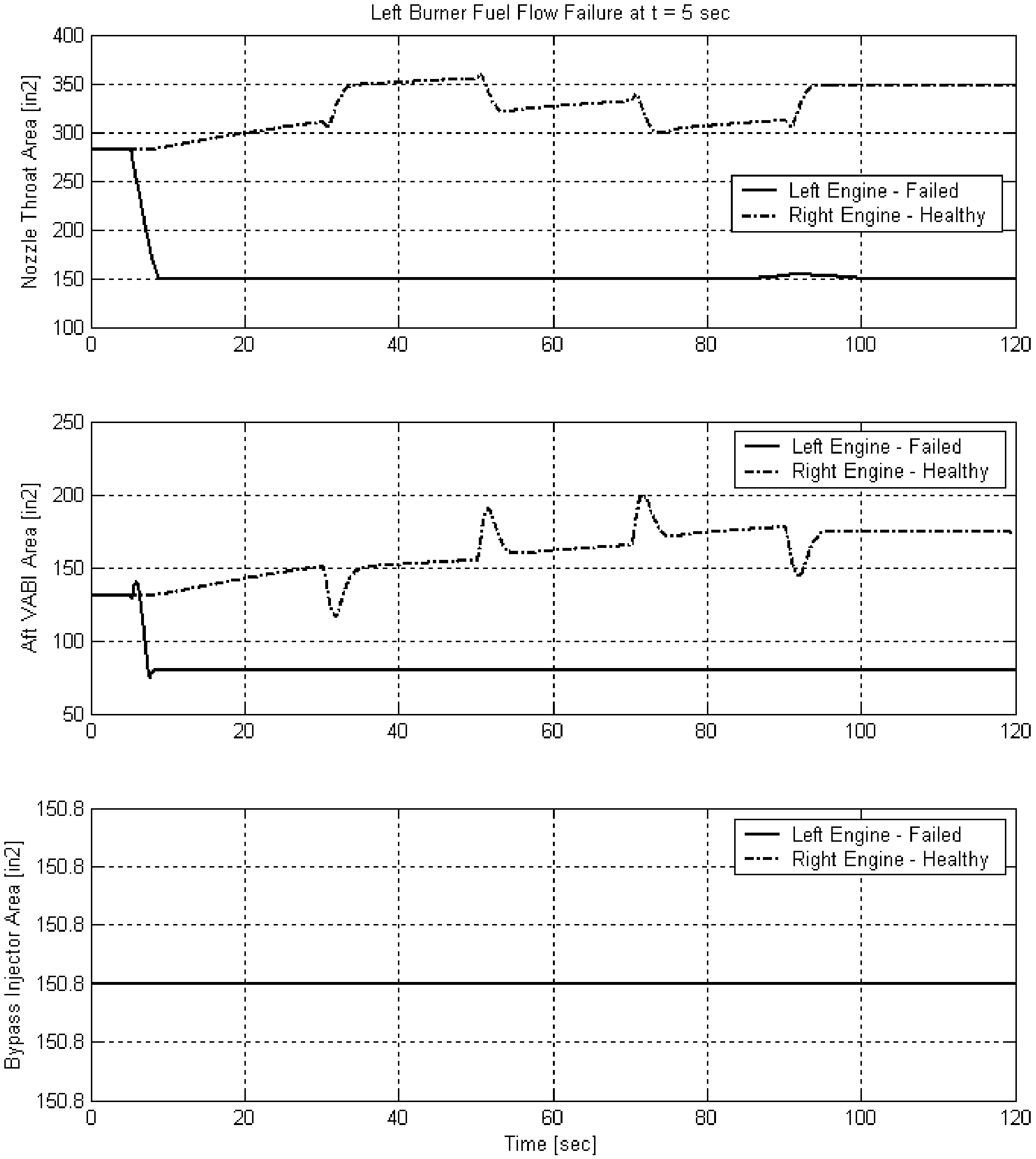


Figure A.7: Standard Throttle Input Response with a Burner Fuel Flow Failure (2)

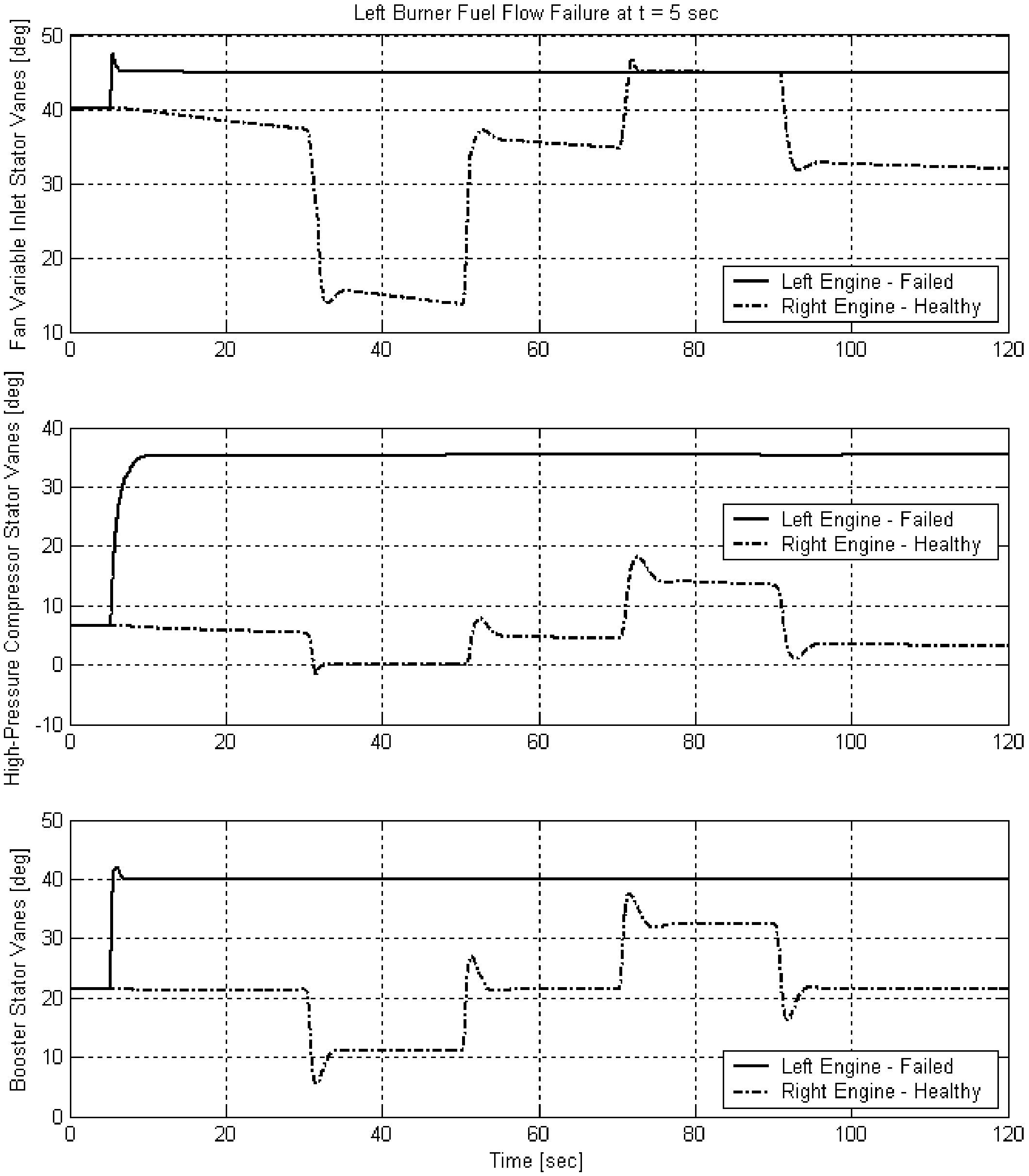


Figure A.8: Standard Throttle Input Response with a Burner Fuel Flow Failure (3)

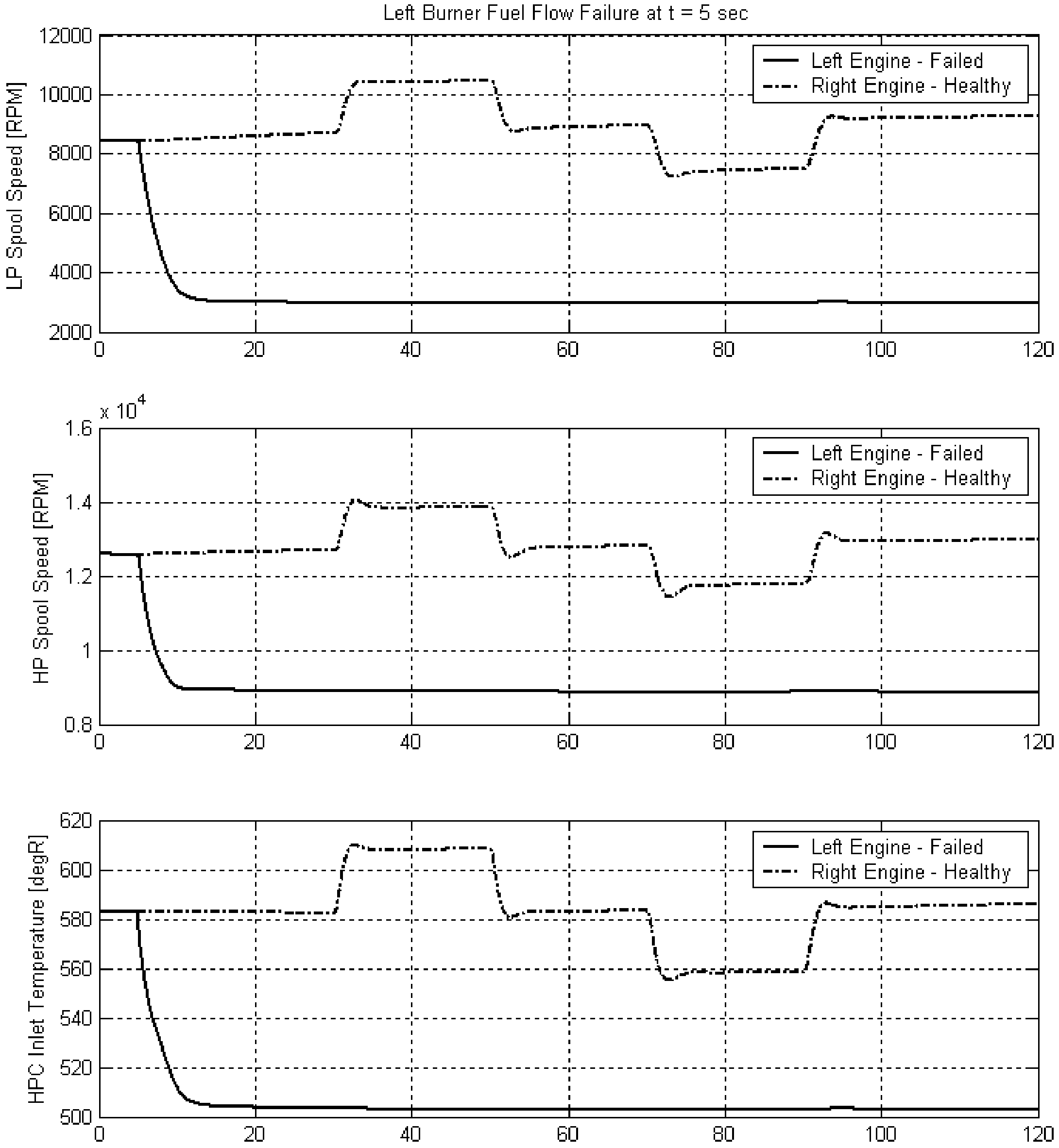


Figure A.9: Standard Throttle Input Response with a Burner Fuel Flow Failure (4)

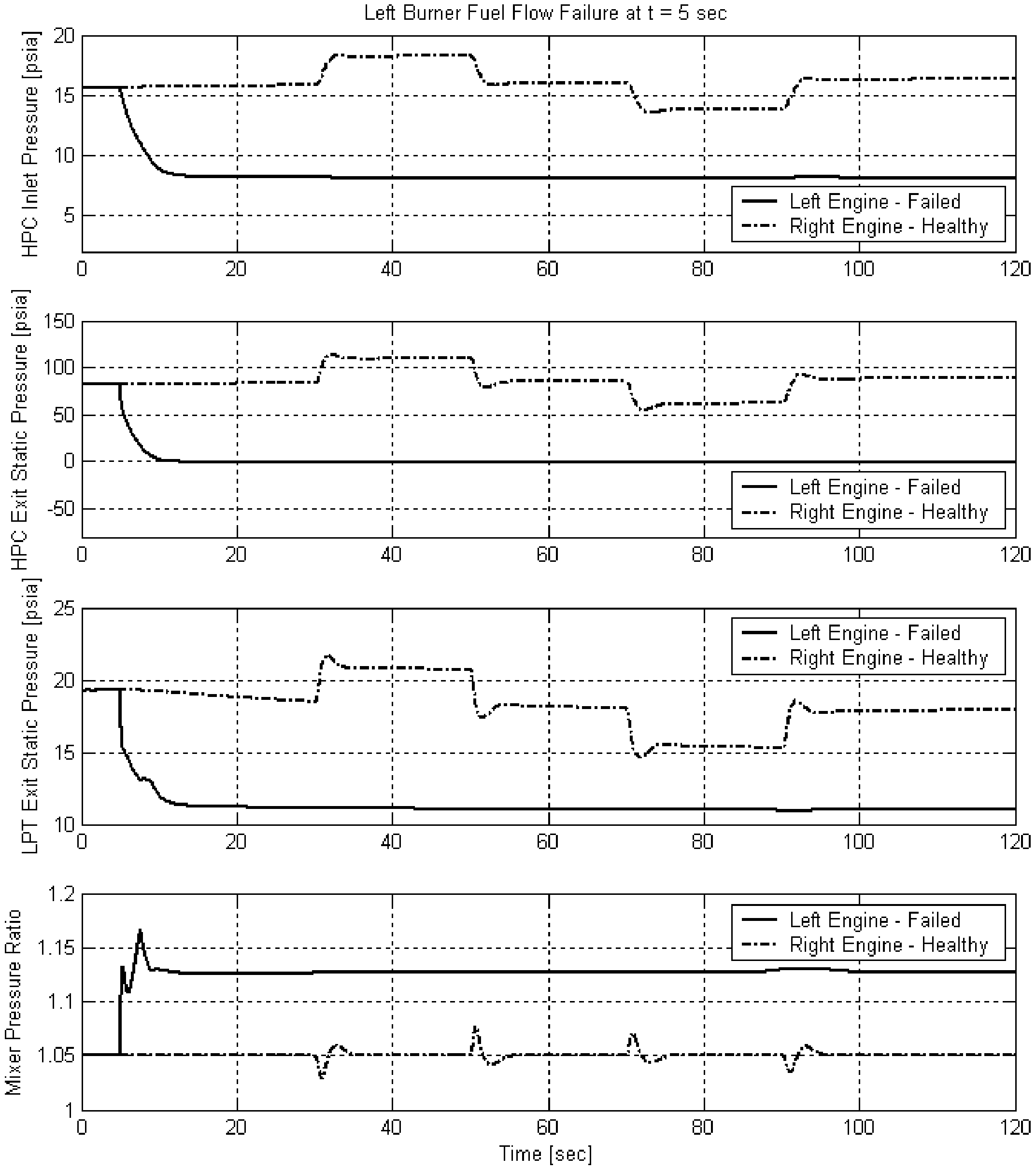


Figure A.10: Standard Throttle Input Response with a Burner Fuel Flow Failure (5)

Left Engine Nozzle Area Locked at 200in^2:

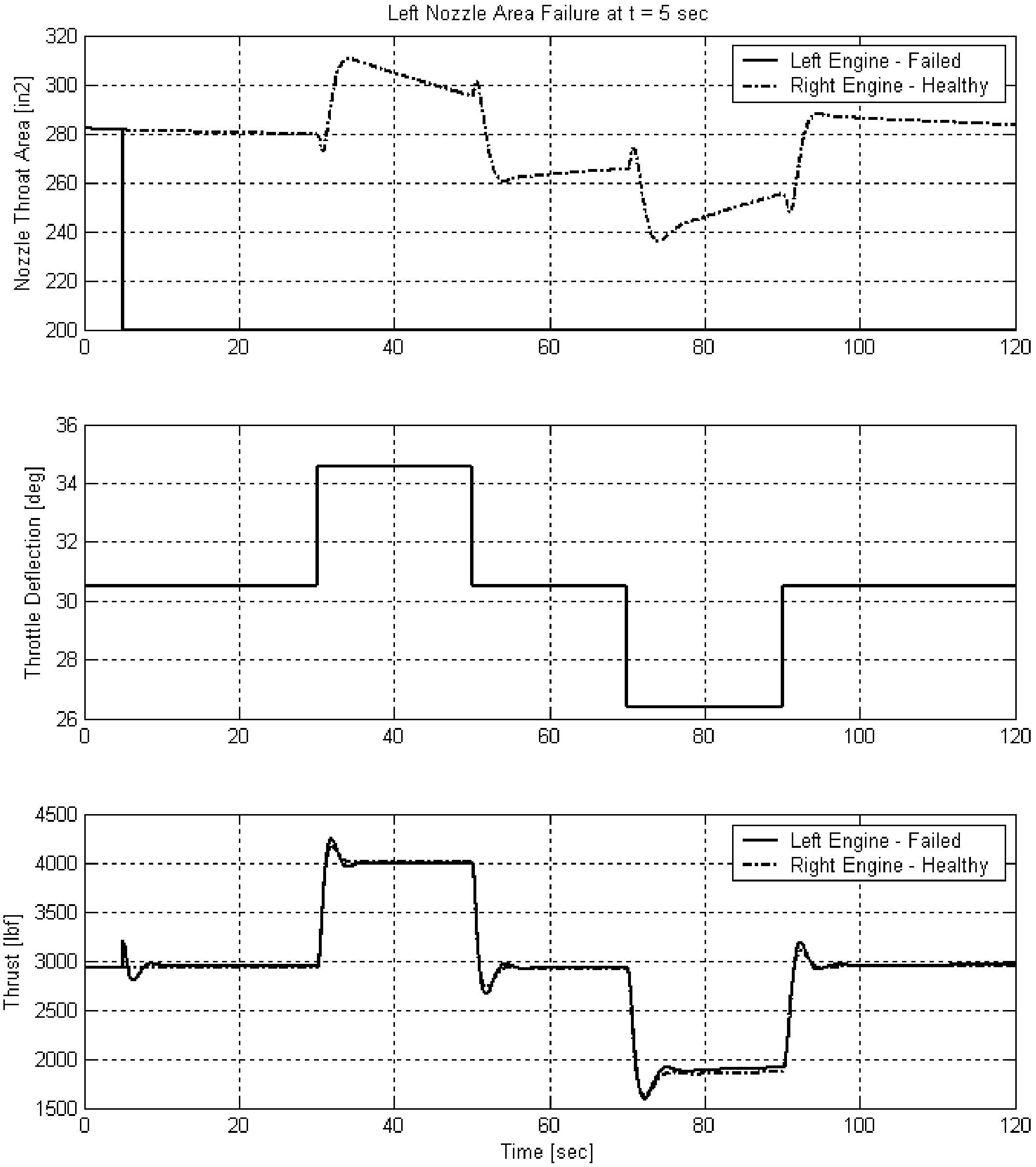


Figure A.11: Standard Throttle Input Response with a Nozzle Area Actuator Failure (1)

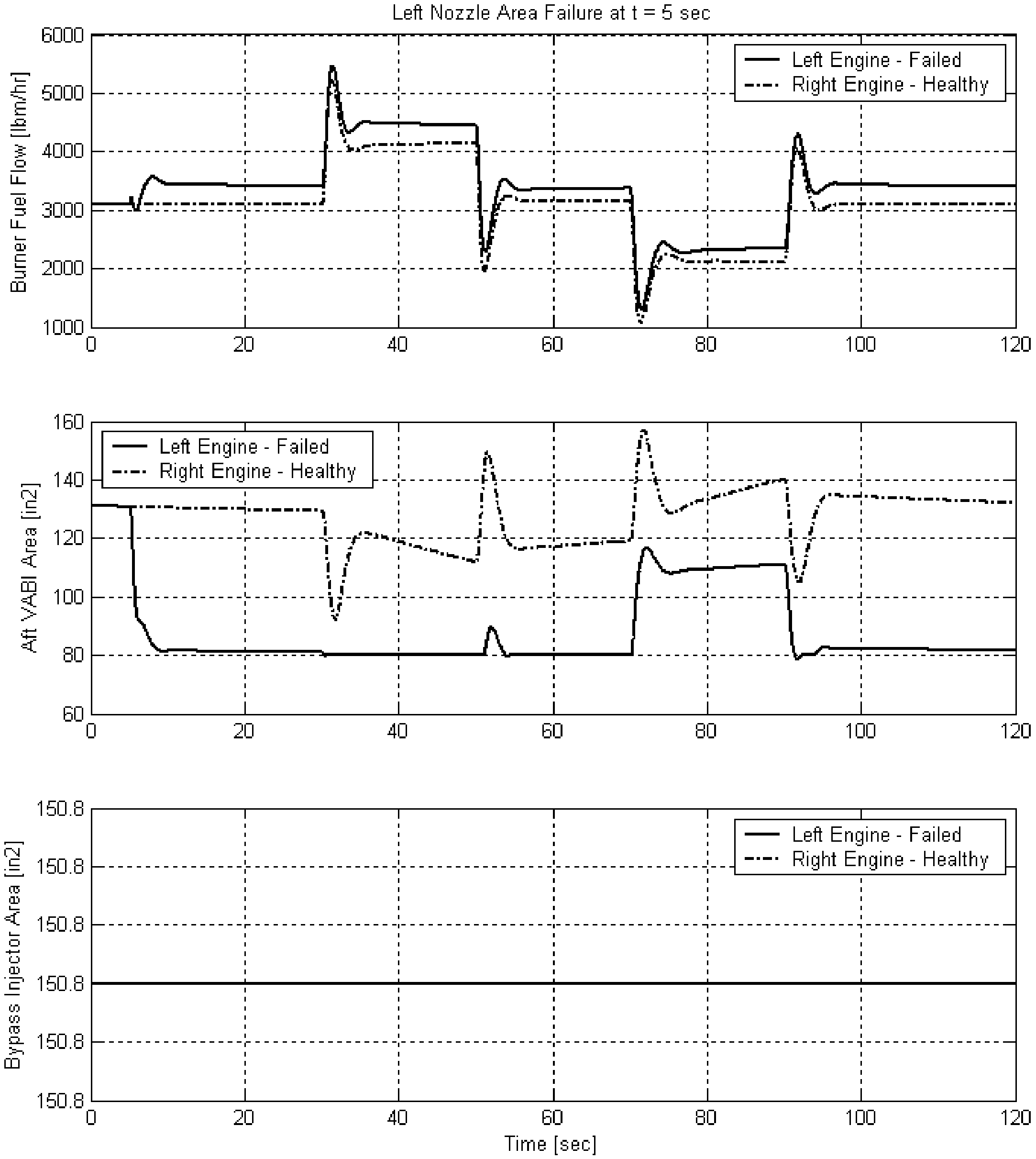


Figure A.12: Standard Throttle Input Response with a Nozzle Area Actuator Failure (2)

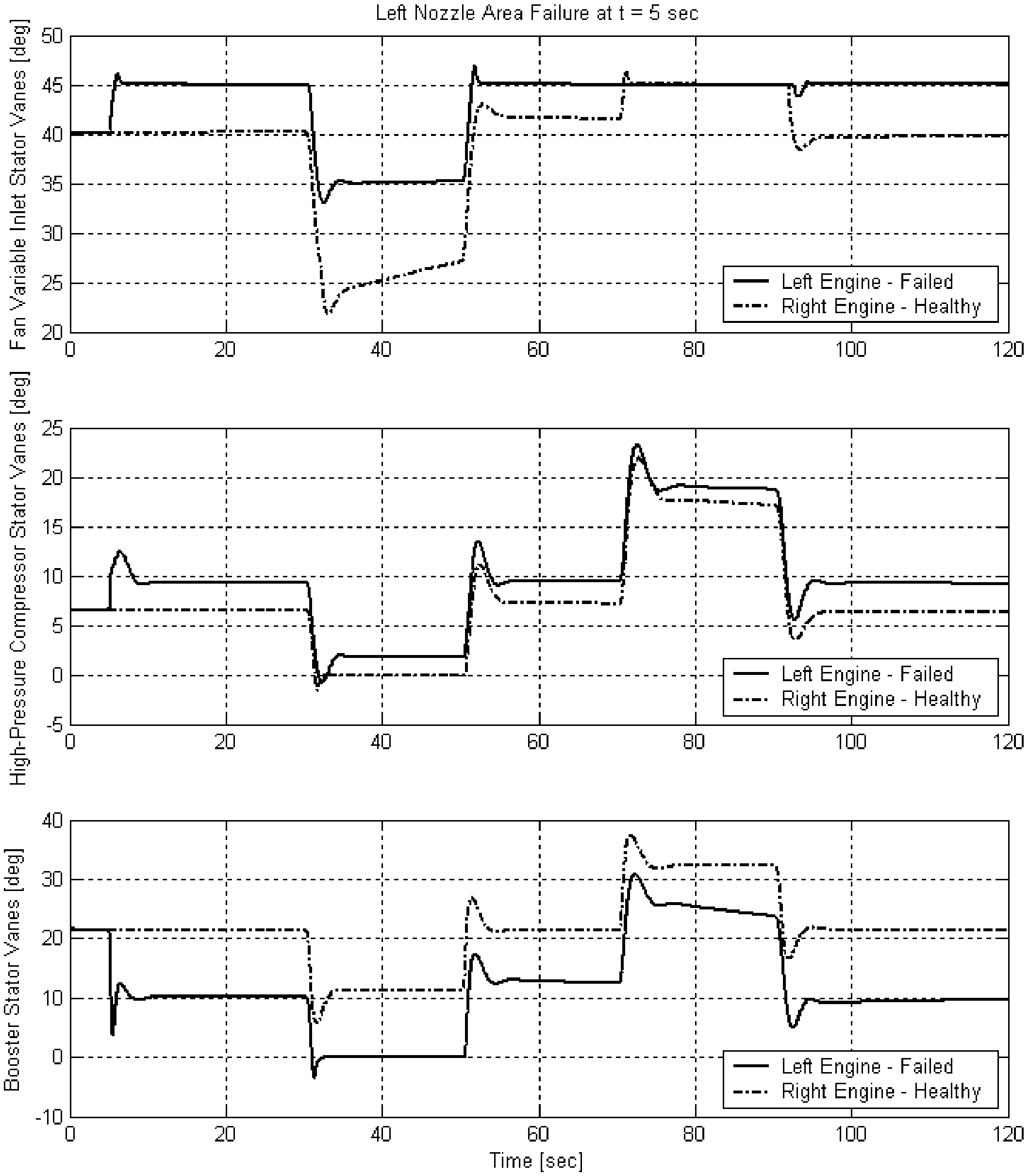


Figure A.13: Standard Throttle Input Response with a Nozzle Area Actuator Failure (3)

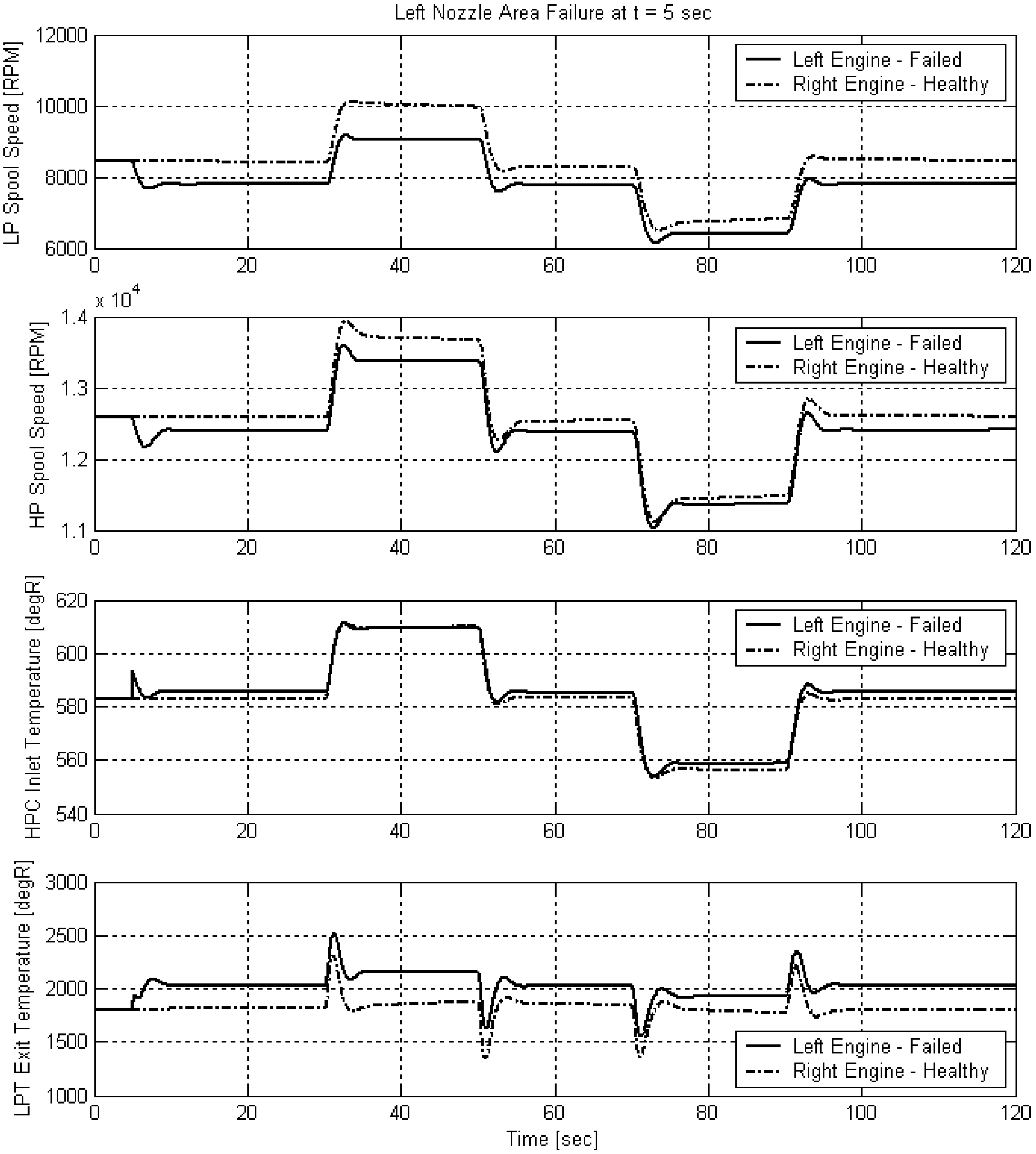


Figure A.14: Standard Throttle Input Response with a Nozzle Area Actuator Failure (4)

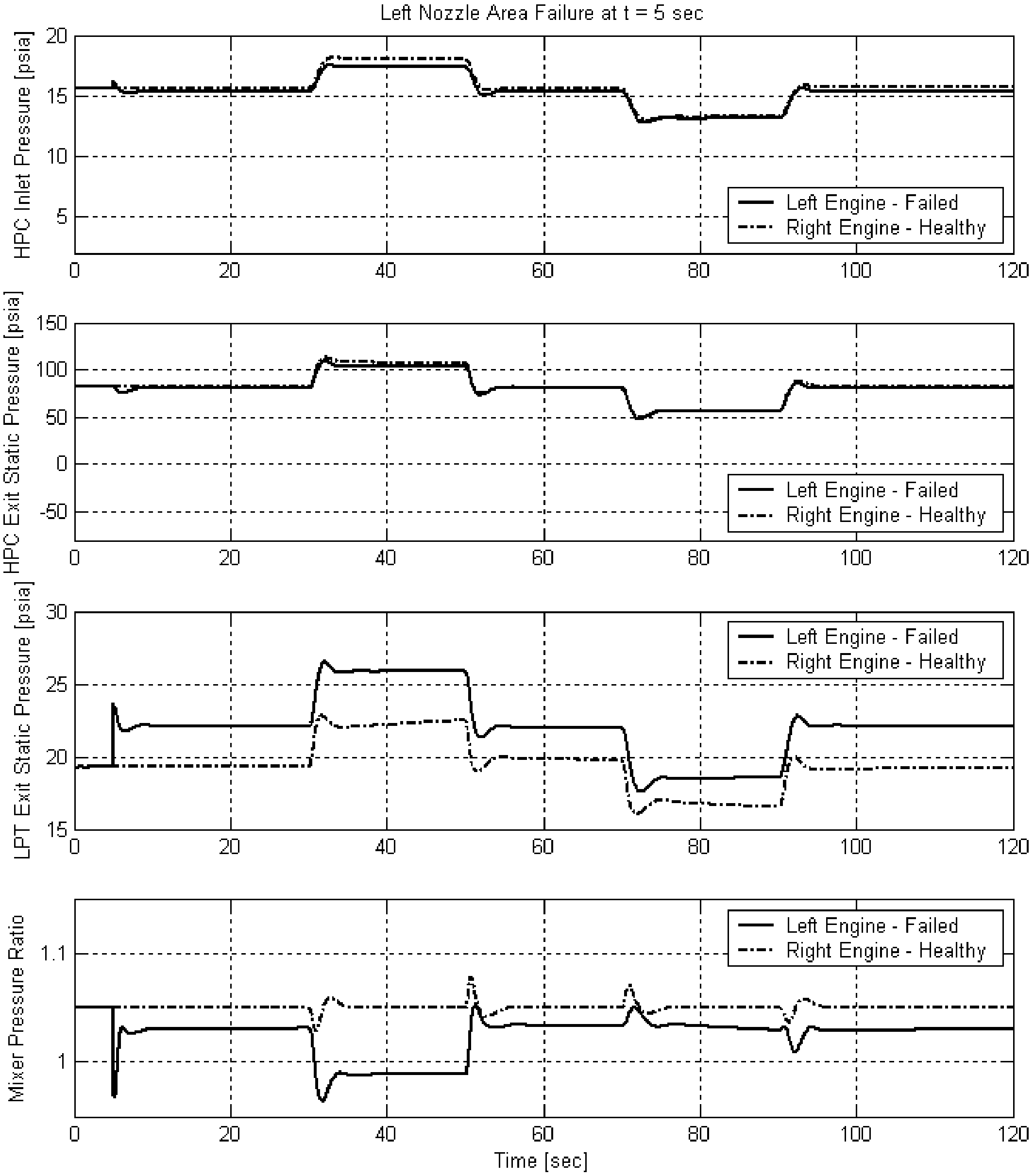


Figure A.15: Standard Throttle Input Response with a Nozzle Area Actuator Failure (5)

Left Engine Mixer Area Locked at 50in^2:

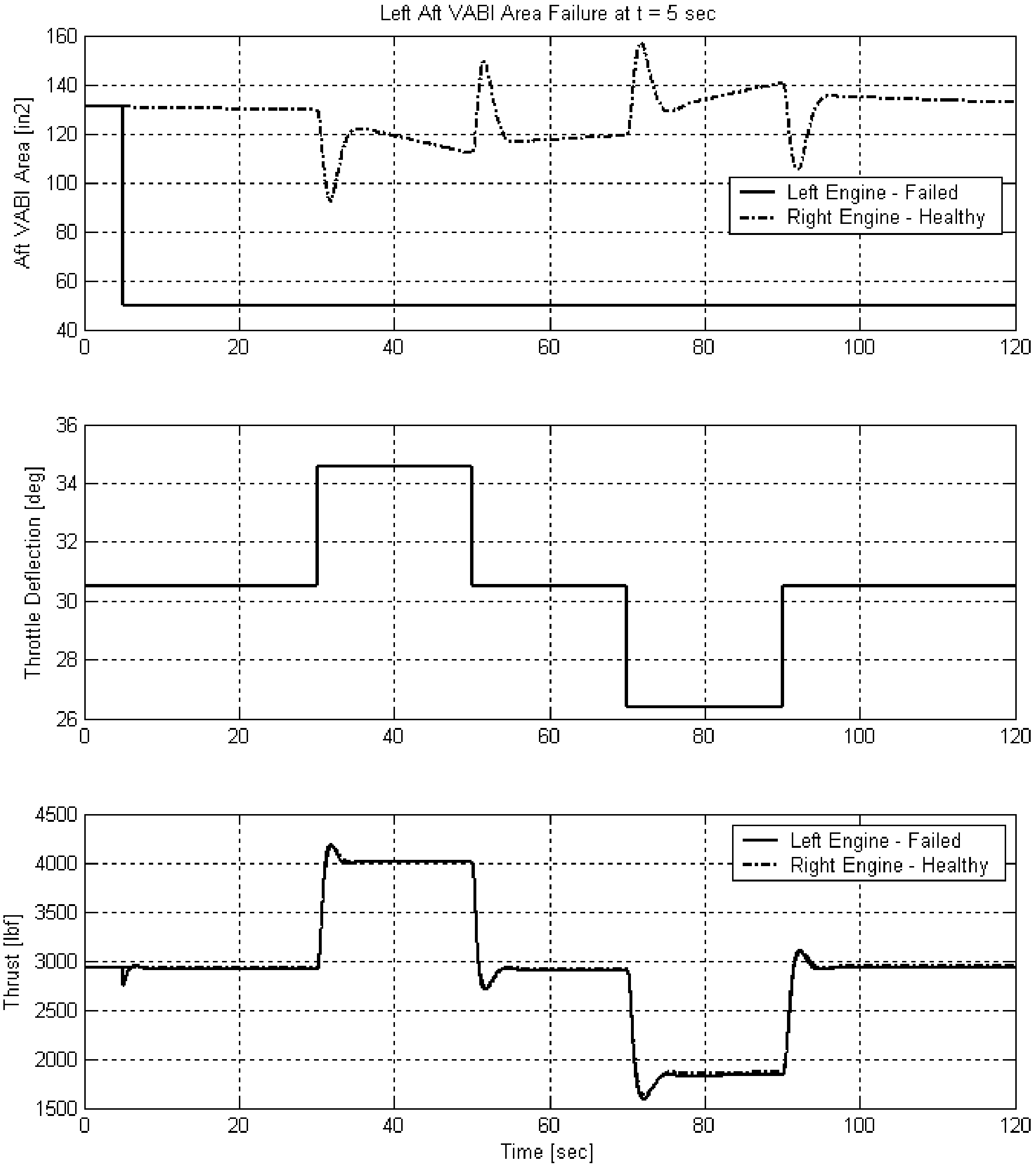


Figure A.16: Standard Throttle Input Response with a Mixer Area Actuator Failure (1)

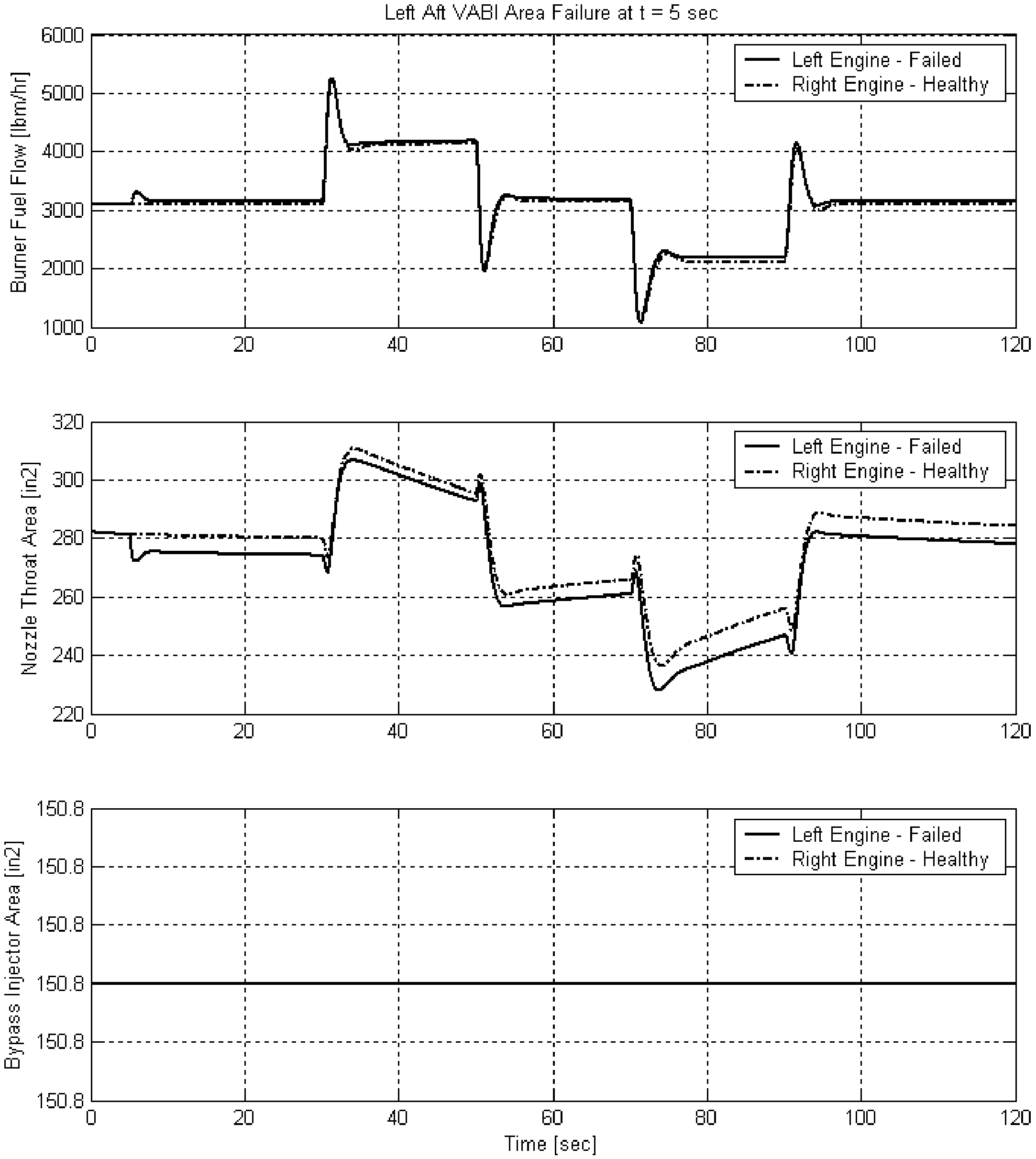


Figure A.17: Standard Throttle Input Response with a Mixer Area Actuator Failure (2)

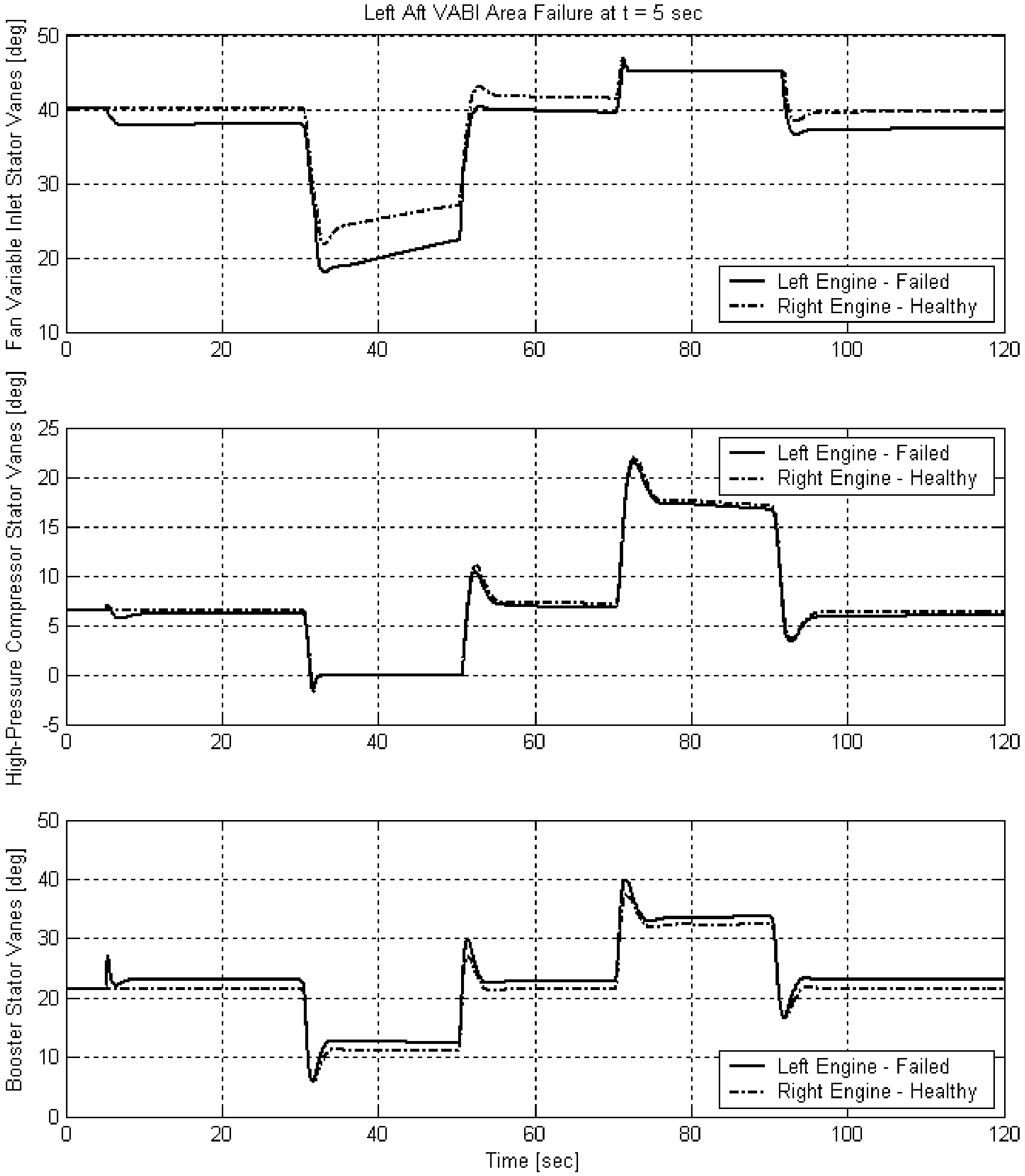


Figure A.18: Standard Throttle Input Response with a Mixer Area Actuator Failure (3)

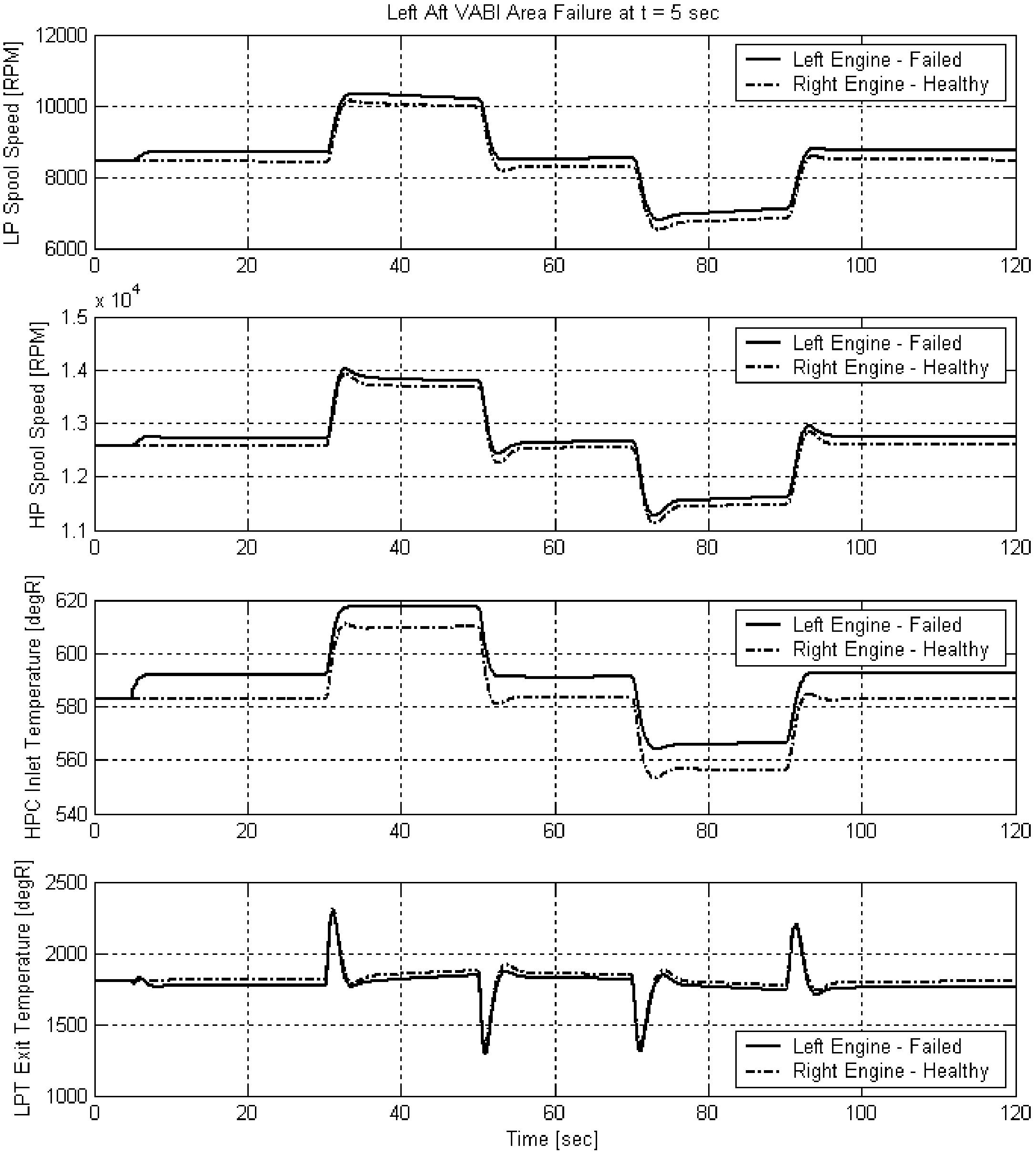


Figure A.19: Standard Throttle Input Response with a Mixer Area Actuator Failure (4)

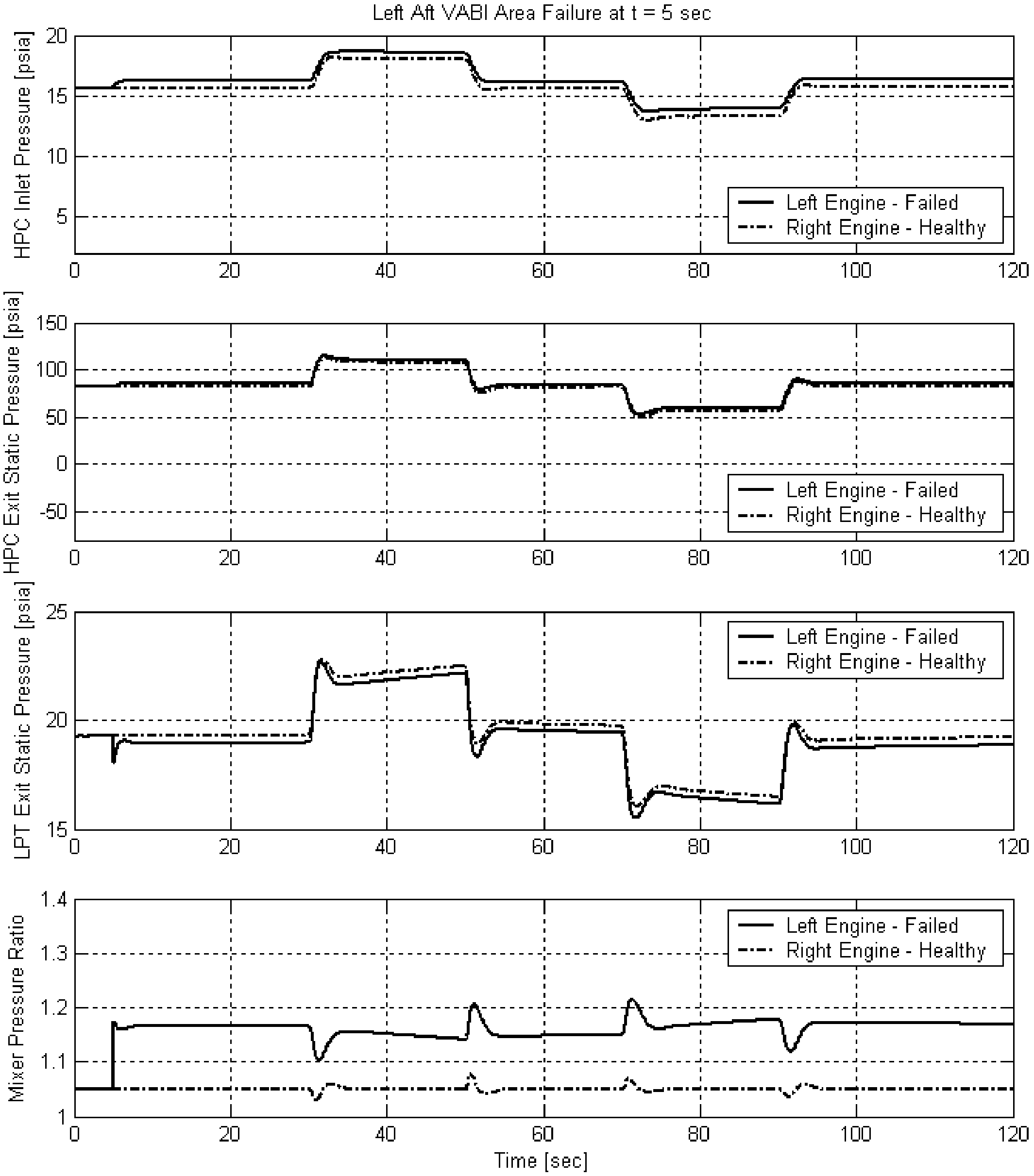


Figure A.20: Standard Throttle Input Response with a Mixer Area Actuator Failure (5)

Left Engine Low-Pressure Spool Speed Sensor at Minimum of 5500RPM:

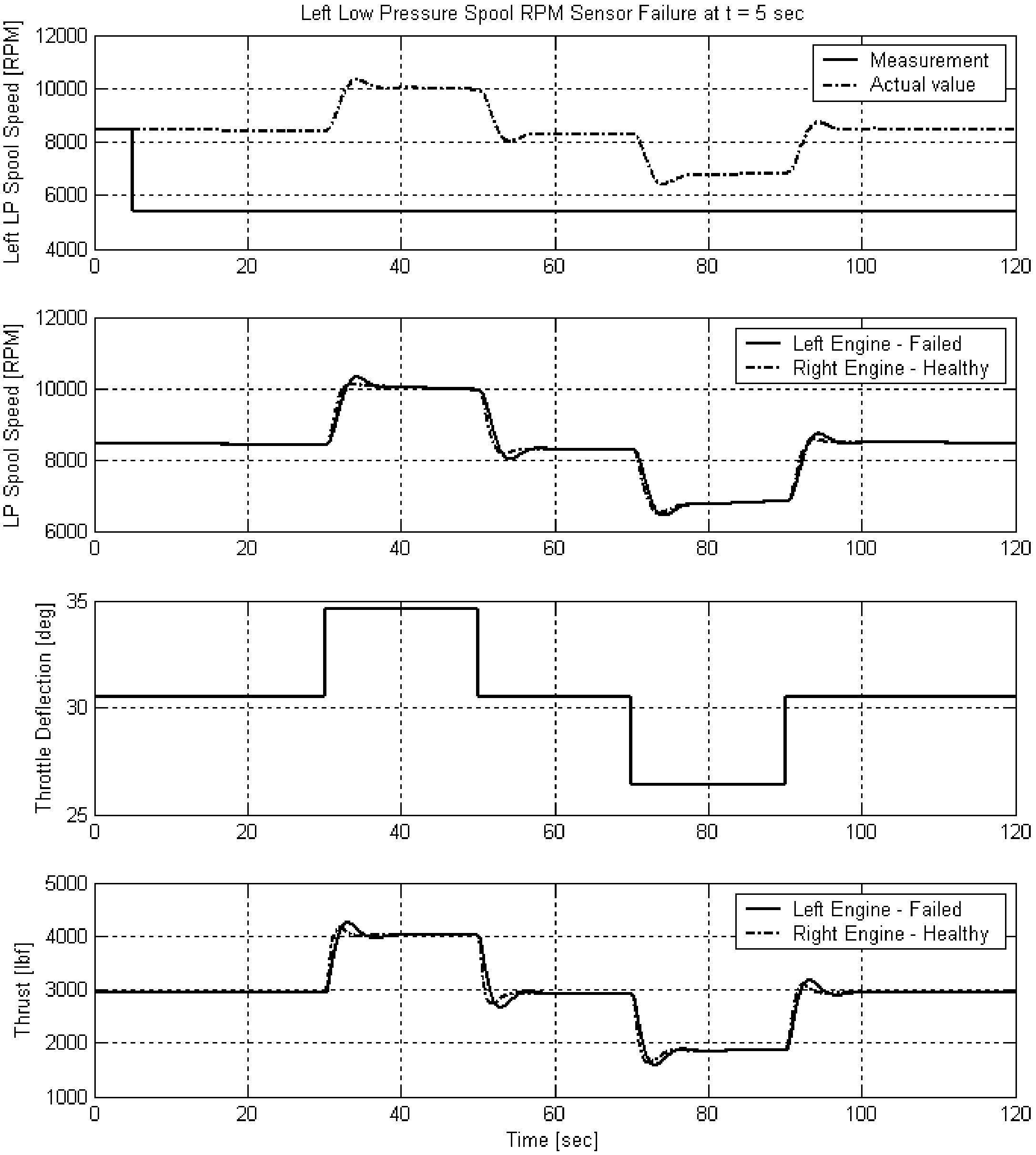


Figure A.21: Standard Throttle Input Response with a LP Spool Speed Min. Failure (1)

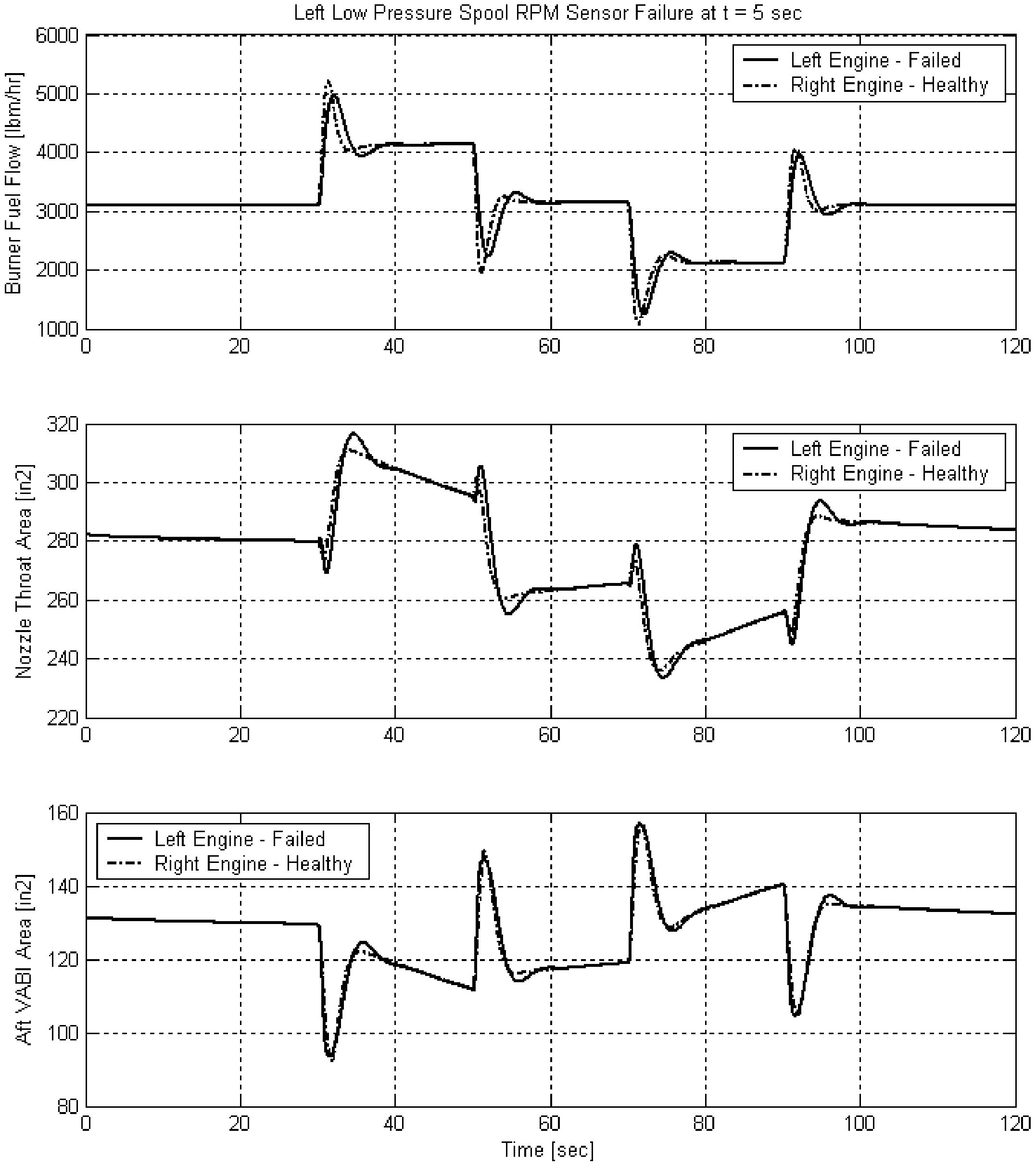


Figure A.22: Standard Throttle Input Response with a LP Spool Speed Min. Failure (2)

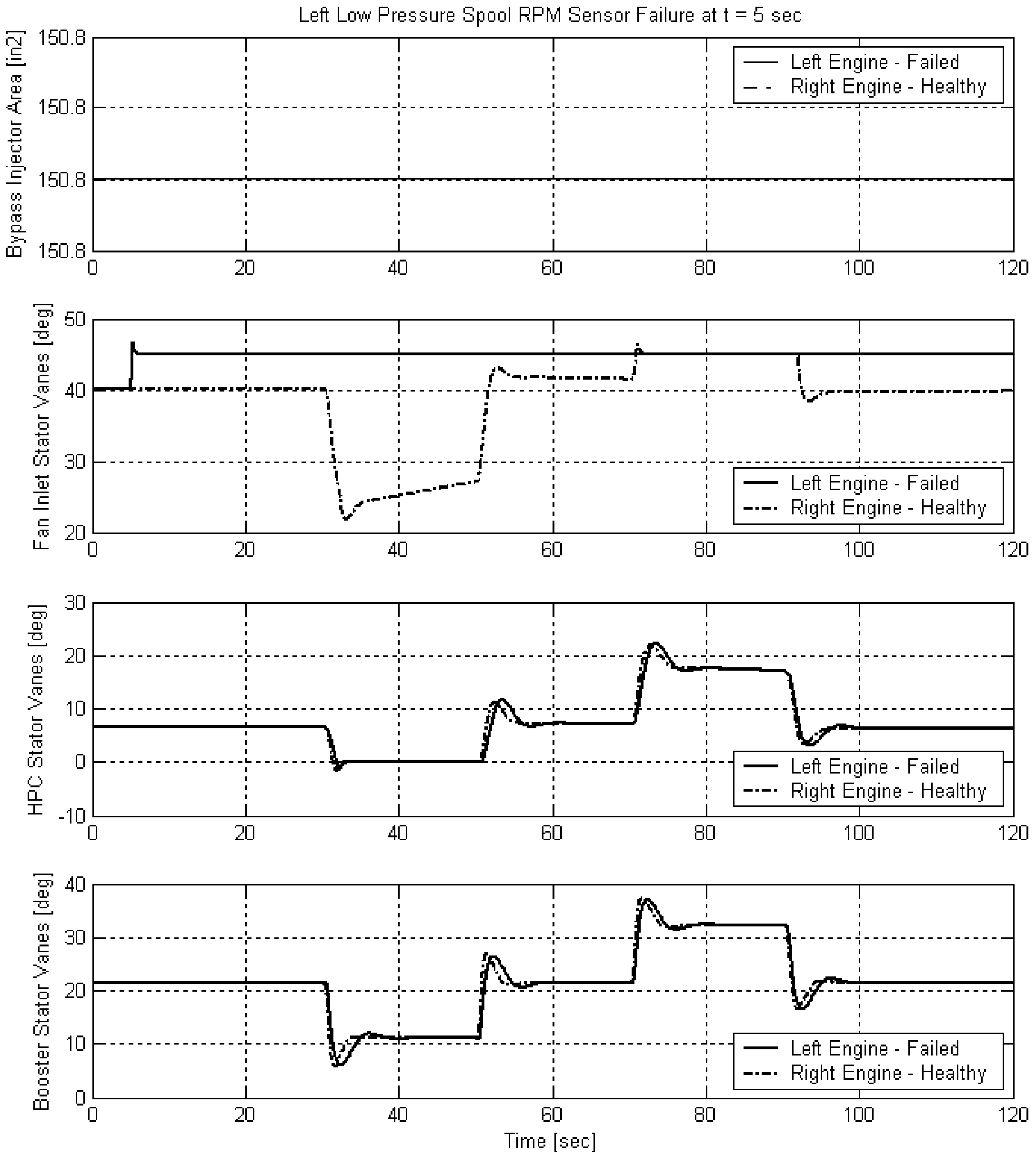


Figure A.23: Standard Throttle Input Response with a LP Spool Speed Min. Failure (3)

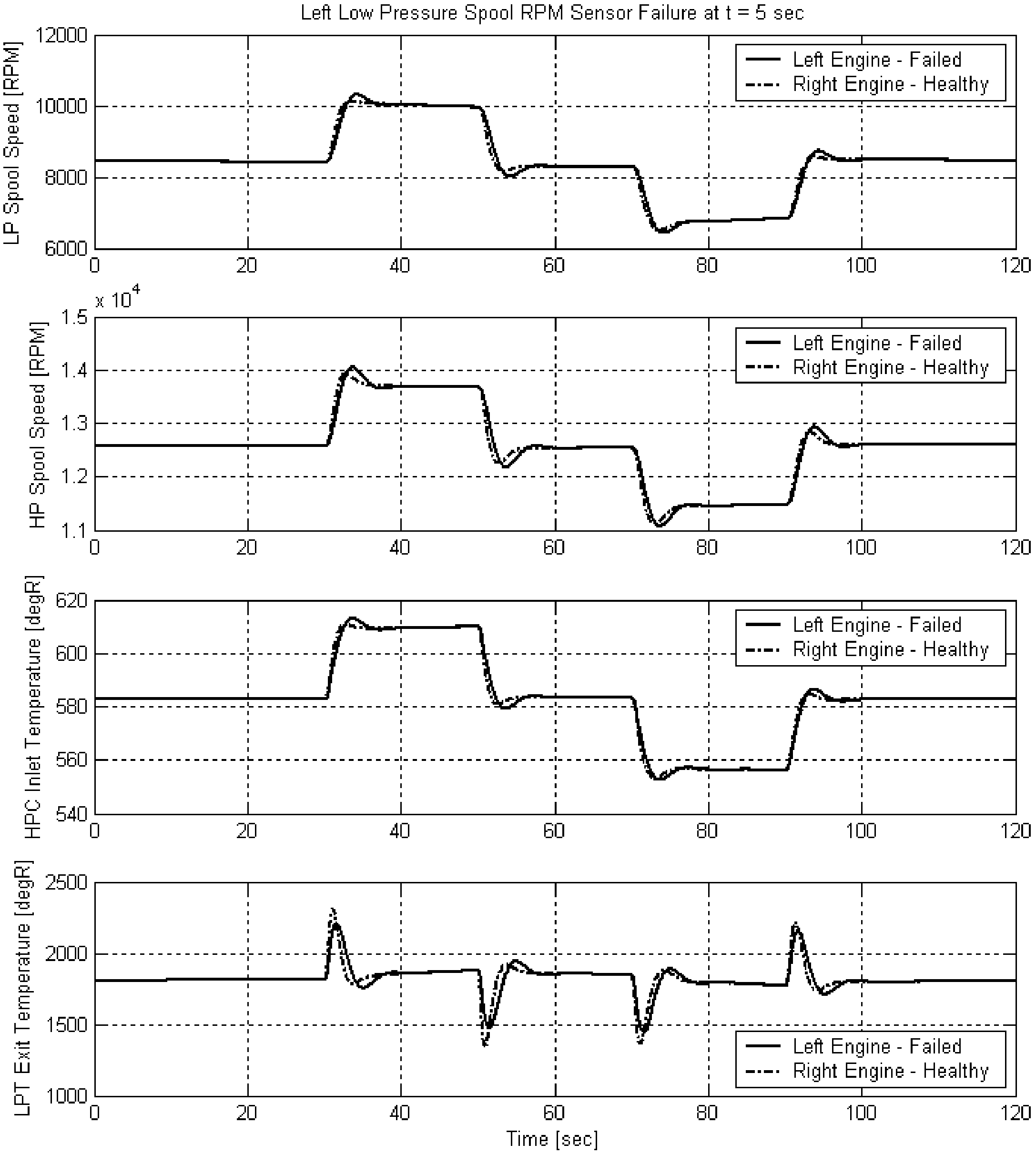


Figure A.24: Standard Throttle Input Response with a LP Spool Speed Min. Failure (4)

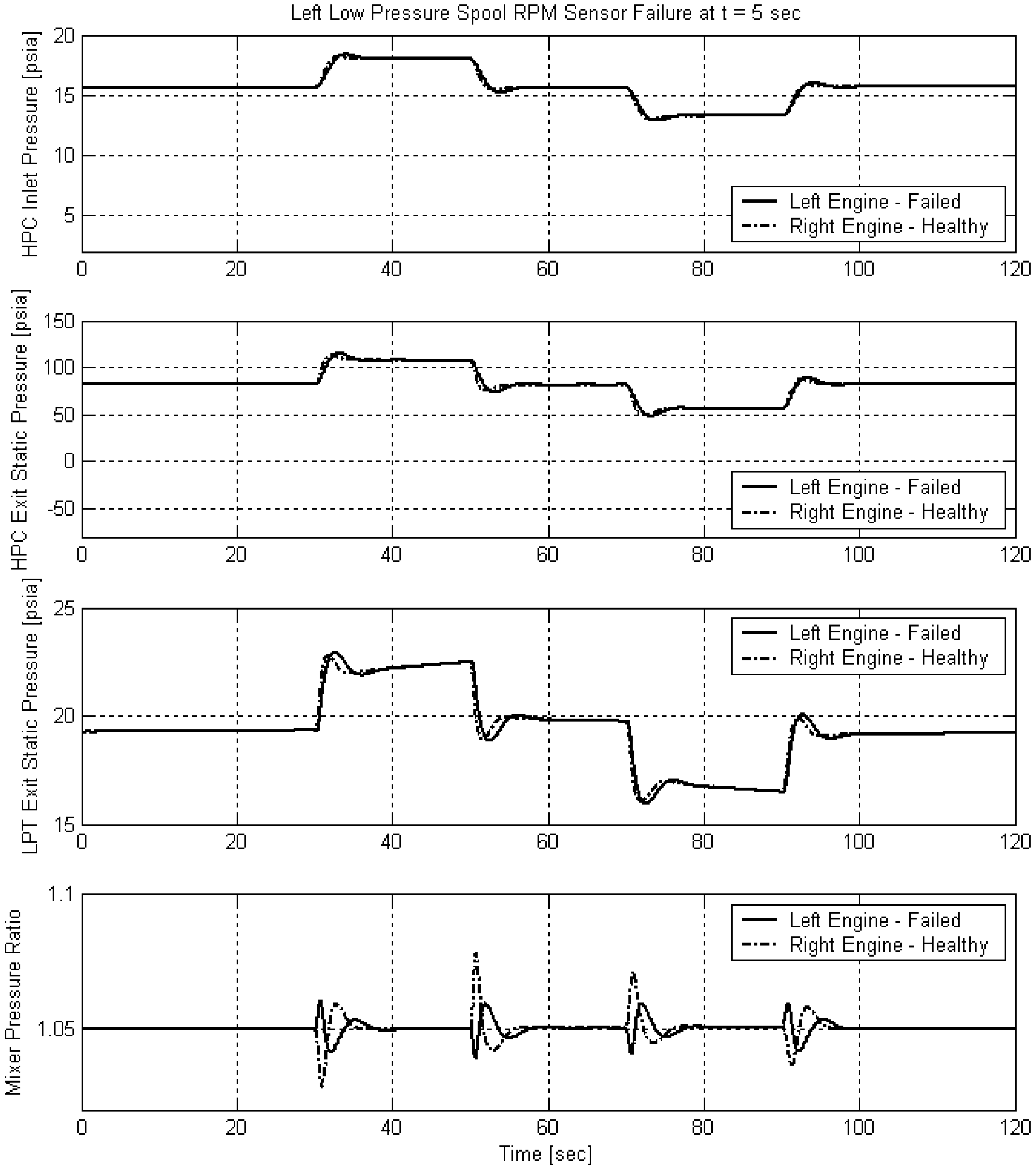


Figure A.25: Standard Throttle Input Response with a LP Spool Speed Min. Failure (5)

Left Engine High-Pressure Compressor Exit Pressure Sensor at Maximum of 135psia:

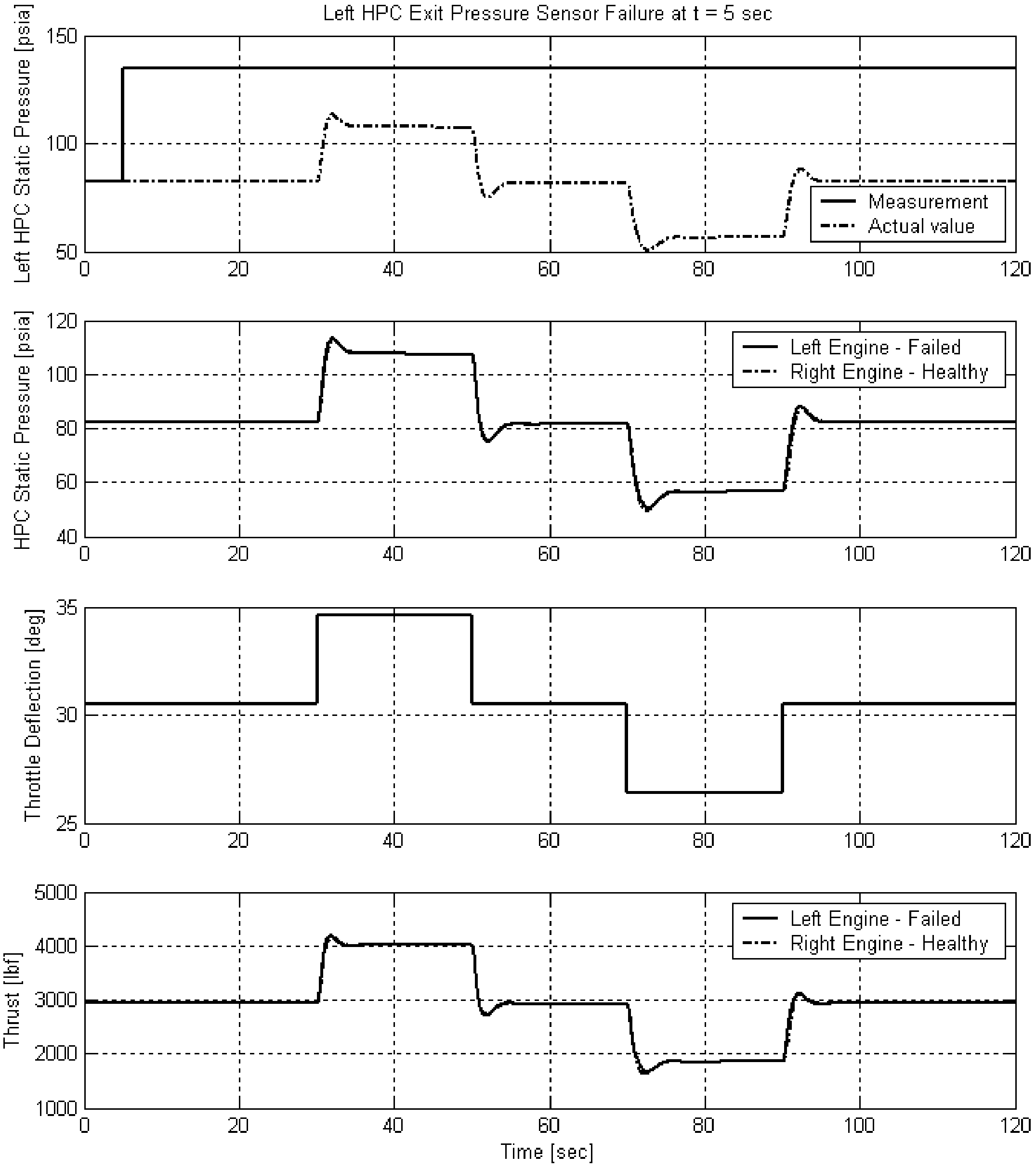


Figure A.26: Standard Throttle Input Response with a HPC Exit Pressure Max. Failure (1)

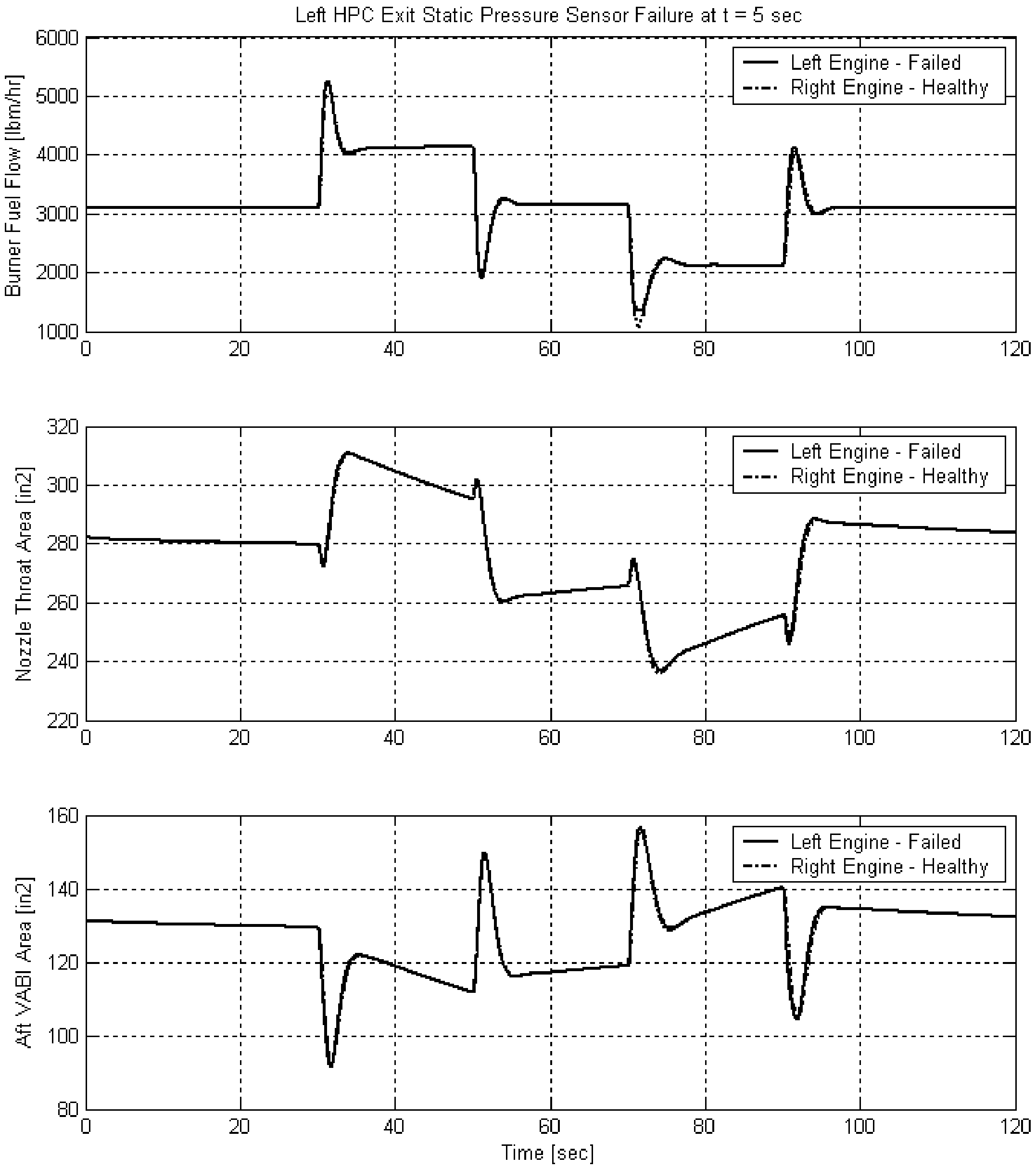


Figure A.27: Standard Throttle Input Response with a HPC Exit Pressure Max. Failure (2)

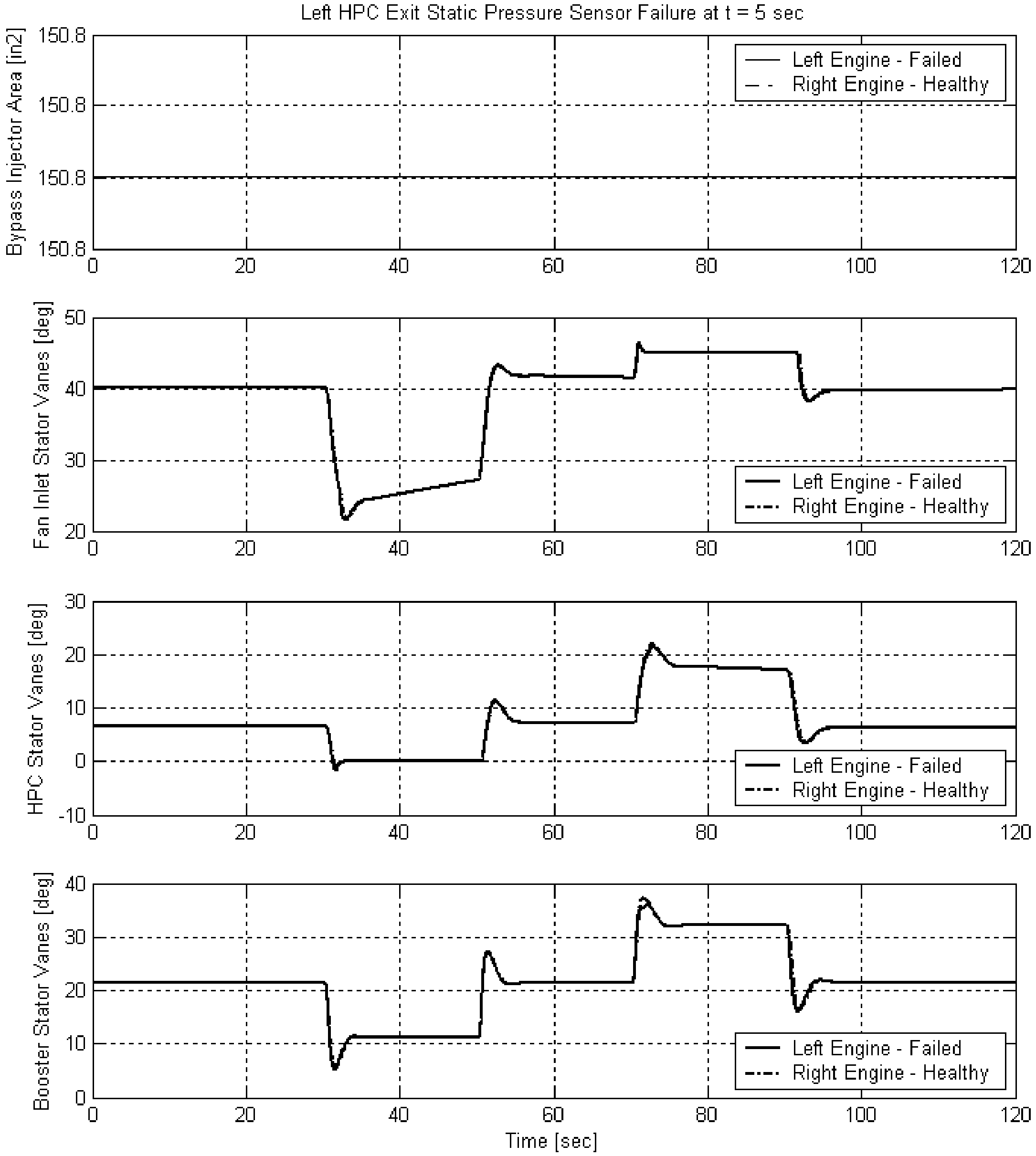


Figure A.28: Standard Throttle Input Response with a HPC Exit Pressure Max. Failure (3)

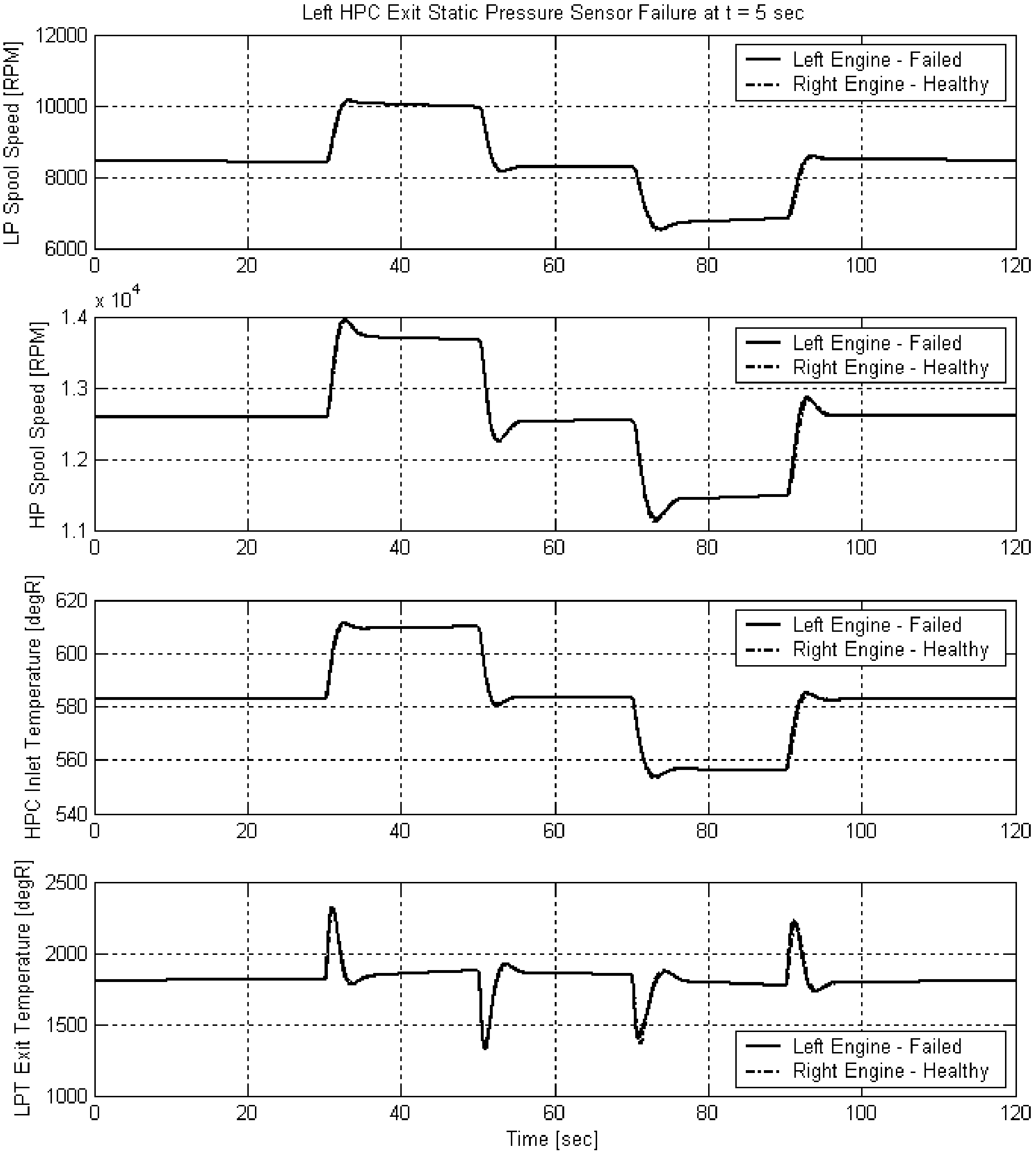


Figure A.29: Standard Throttle Input Response with a HPC Exit Pressure Max. Failure (4)

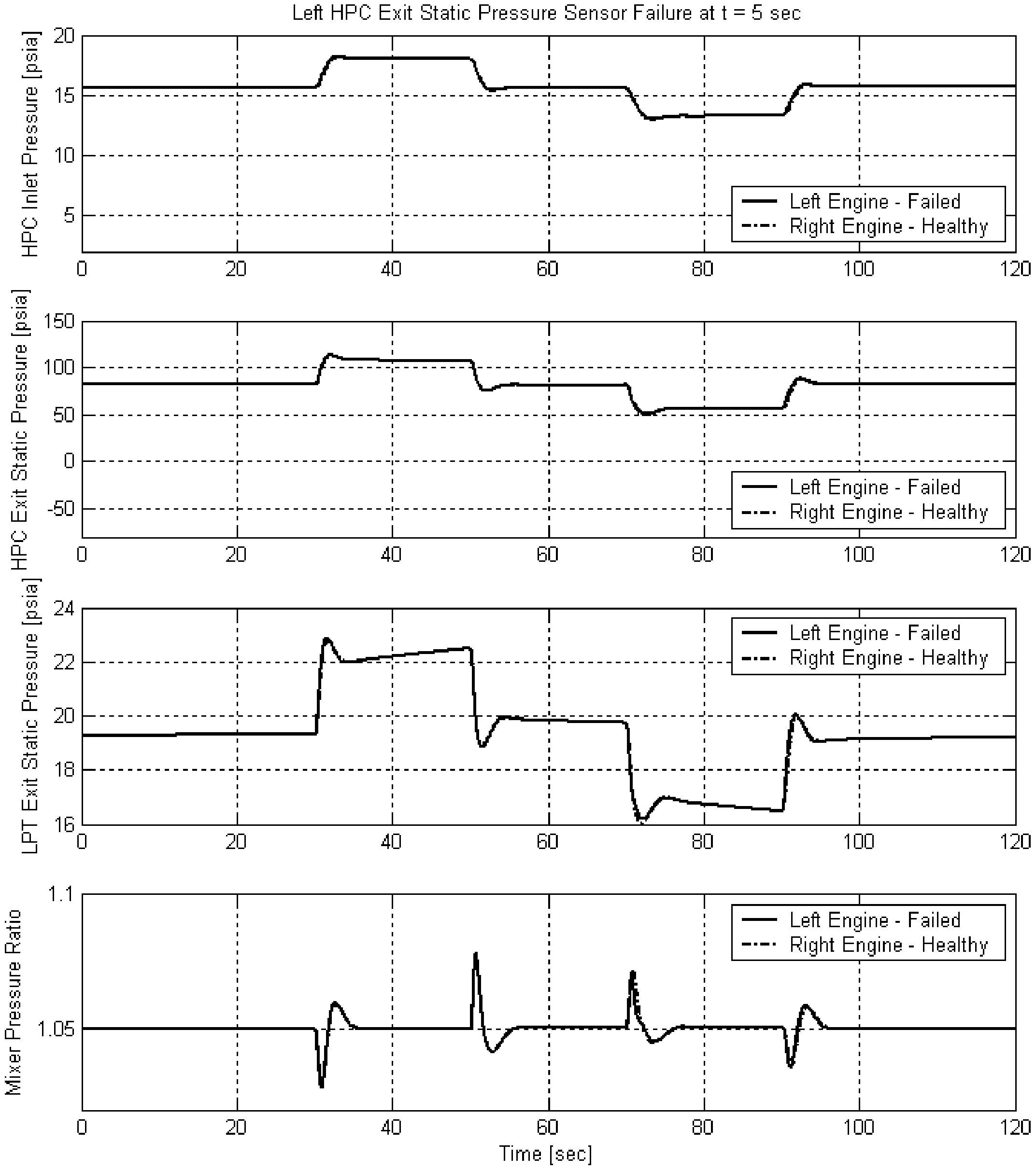


Figure A.30: Standard Throttle Input Response with a HPC Exit Pressure Max. Failure (5)

Left Engine High-Pressure Compressor Exit Pressure Sensor at Minimum of 30psia:

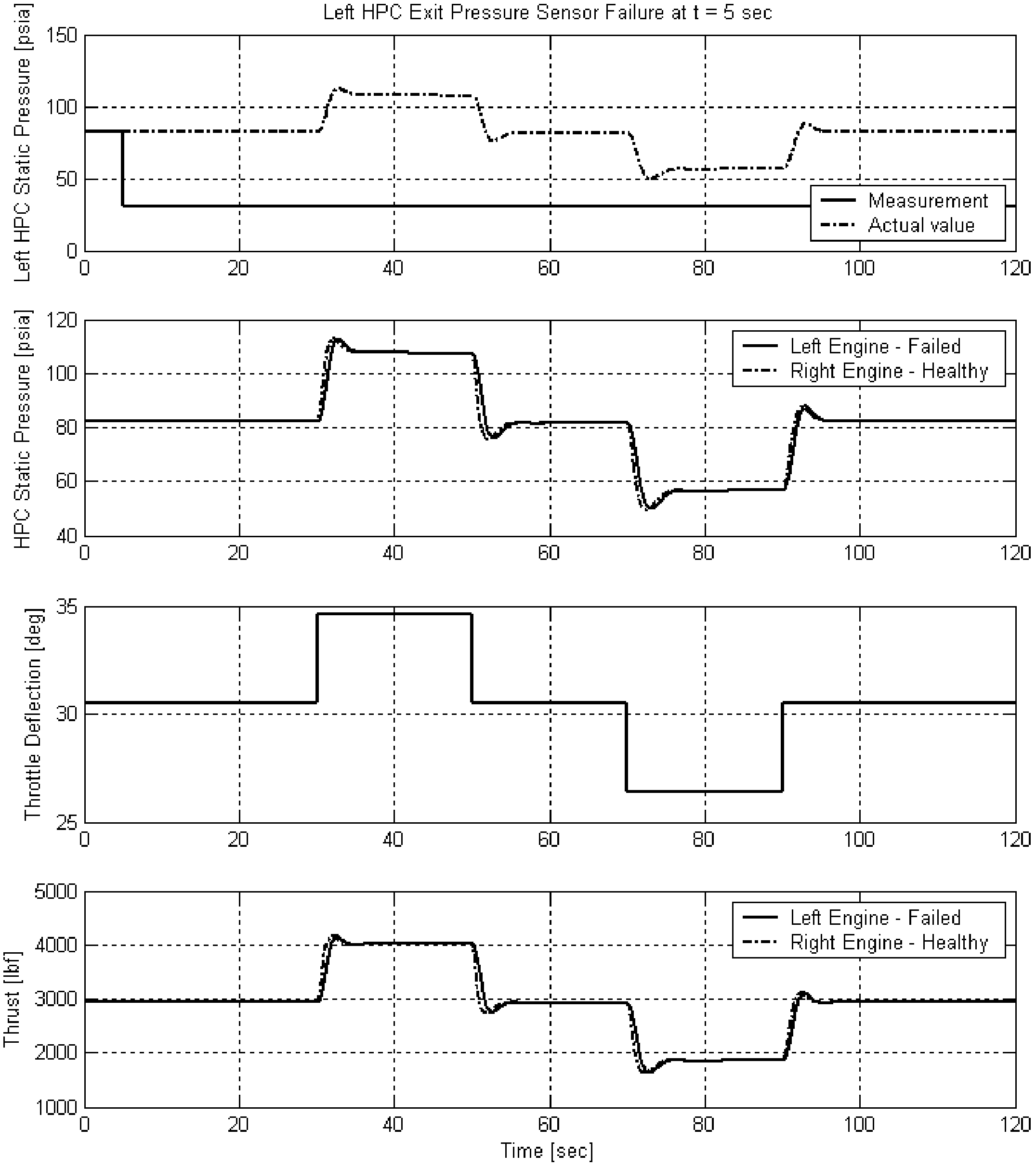


Figure A.31: Standard Throttle Input Response with a HPC Exit Pressure Min. Failure (1)

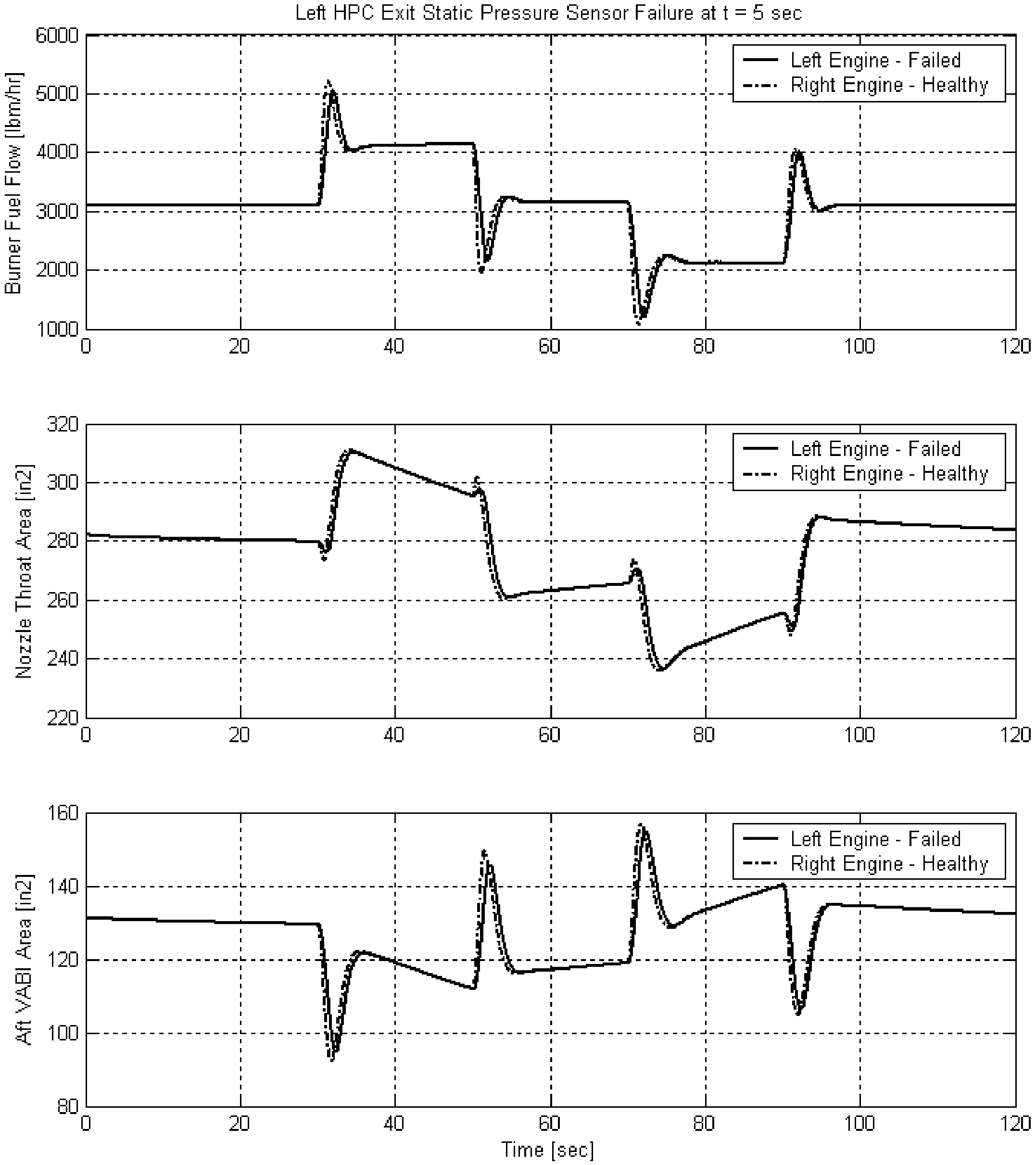


Figure A.32: Standard Throttle Input Response with a HPC Exit Pressure Min. Failure (2)

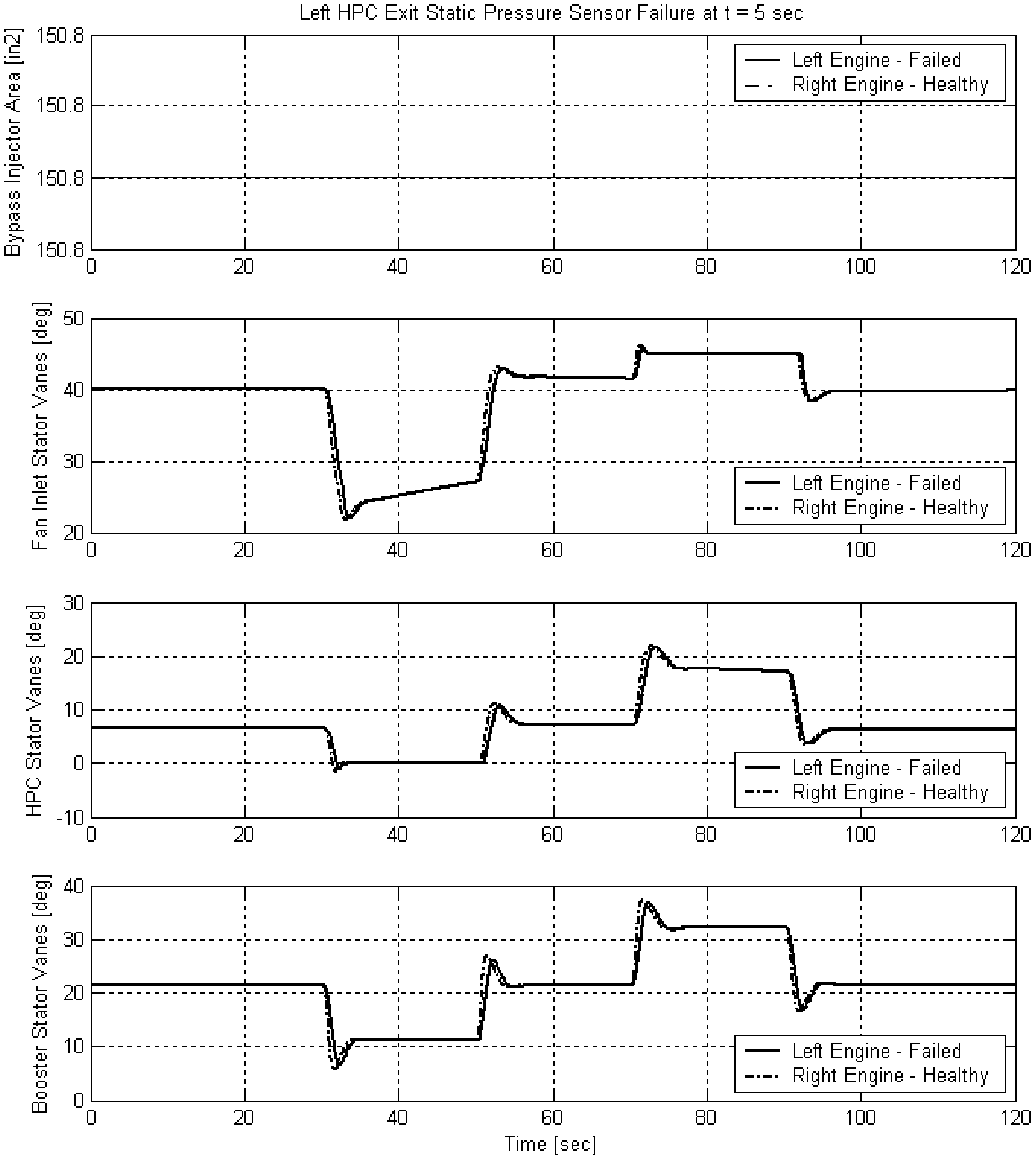


Figure A.33: Standard Throttle Input Response with a HPC Exit Pressure Min. Failure (3)

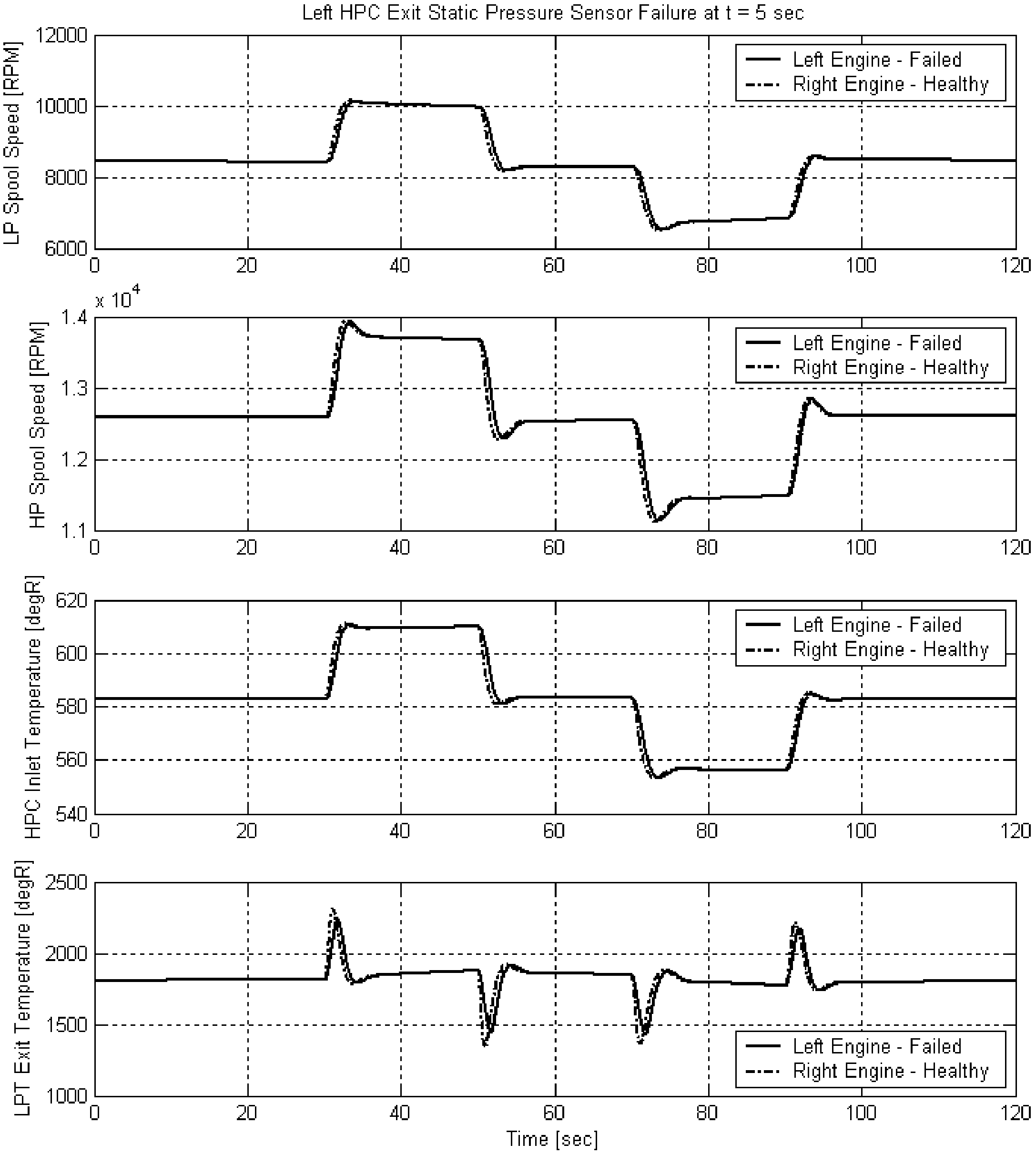


Figure A.34: Standard Throttle Input Response with a HPC Exit Pressure Min. Failure (4)

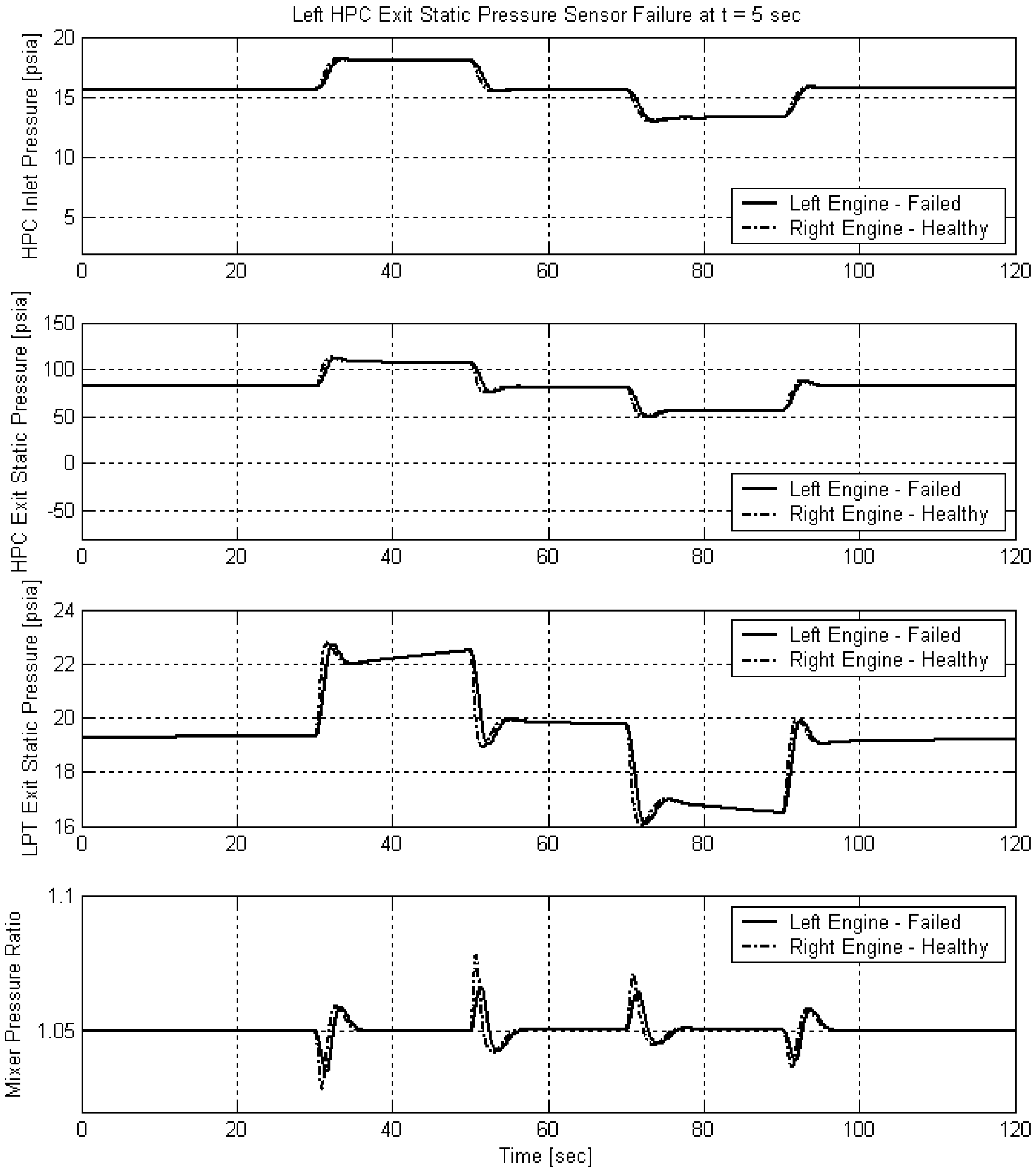


Figure A.35: Standard Throttle Input Response with a HPC Exit Pressure Min. Failure (5)

Left Engine Low-Pressure Turbine Exit Pressure Sensor at Minimum of 13psia:

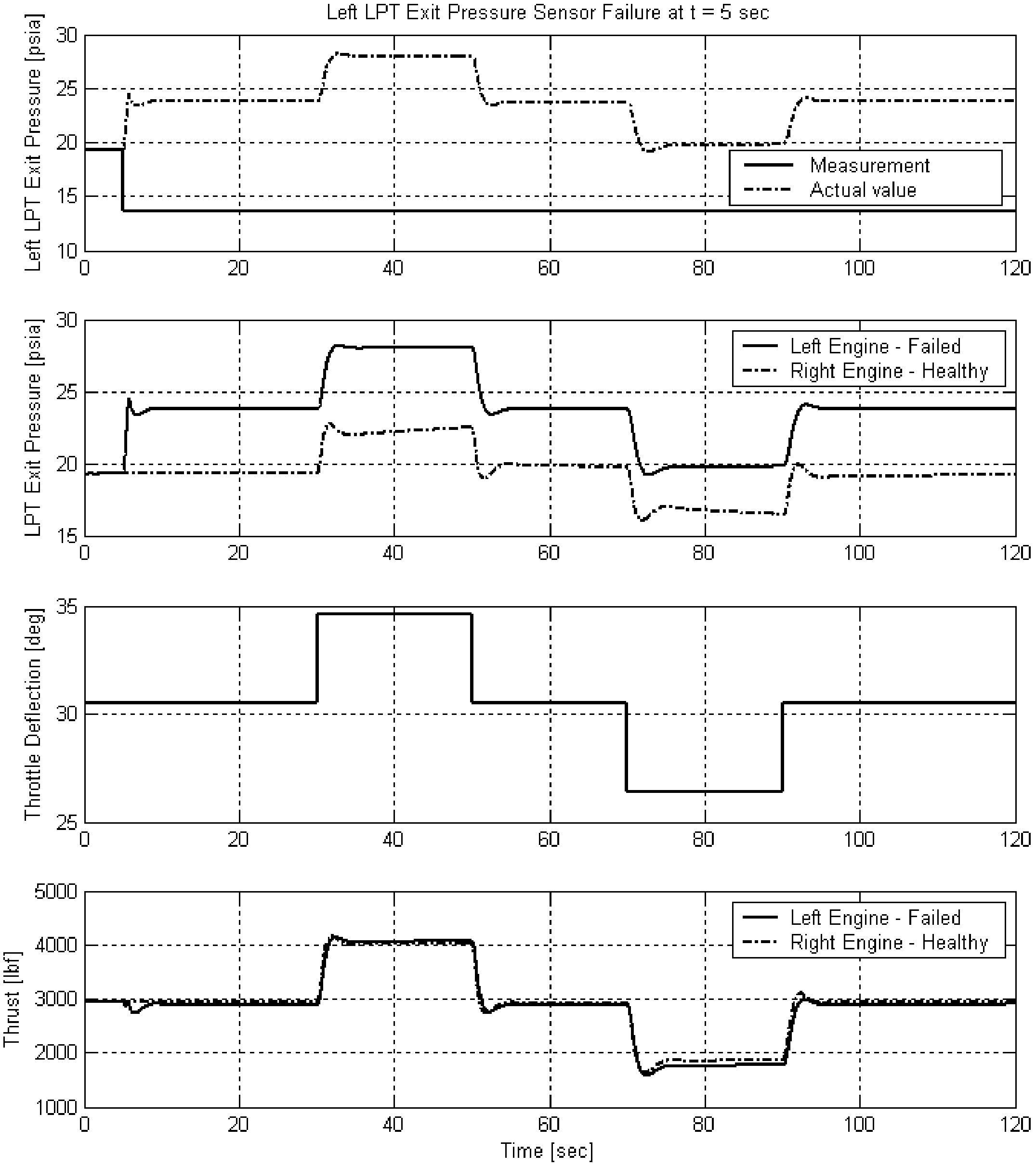


Figure A.36: Standard Throttle Input Response with a LPT Exit Pressure Min. Failure (1)

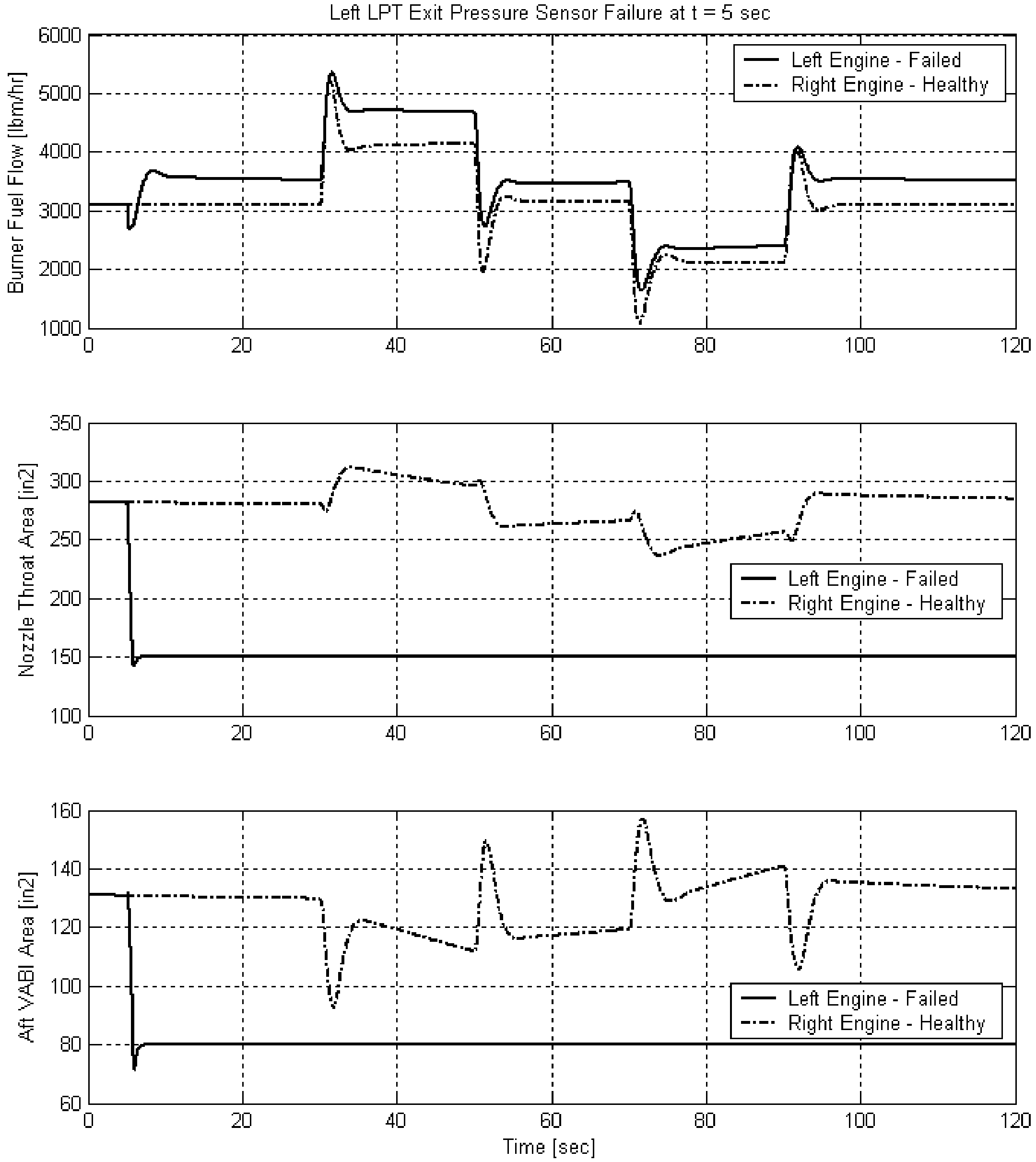


Figure A.37: Standard Throttle Input Response with a LPT Exit Pressure Min. Failure (2)

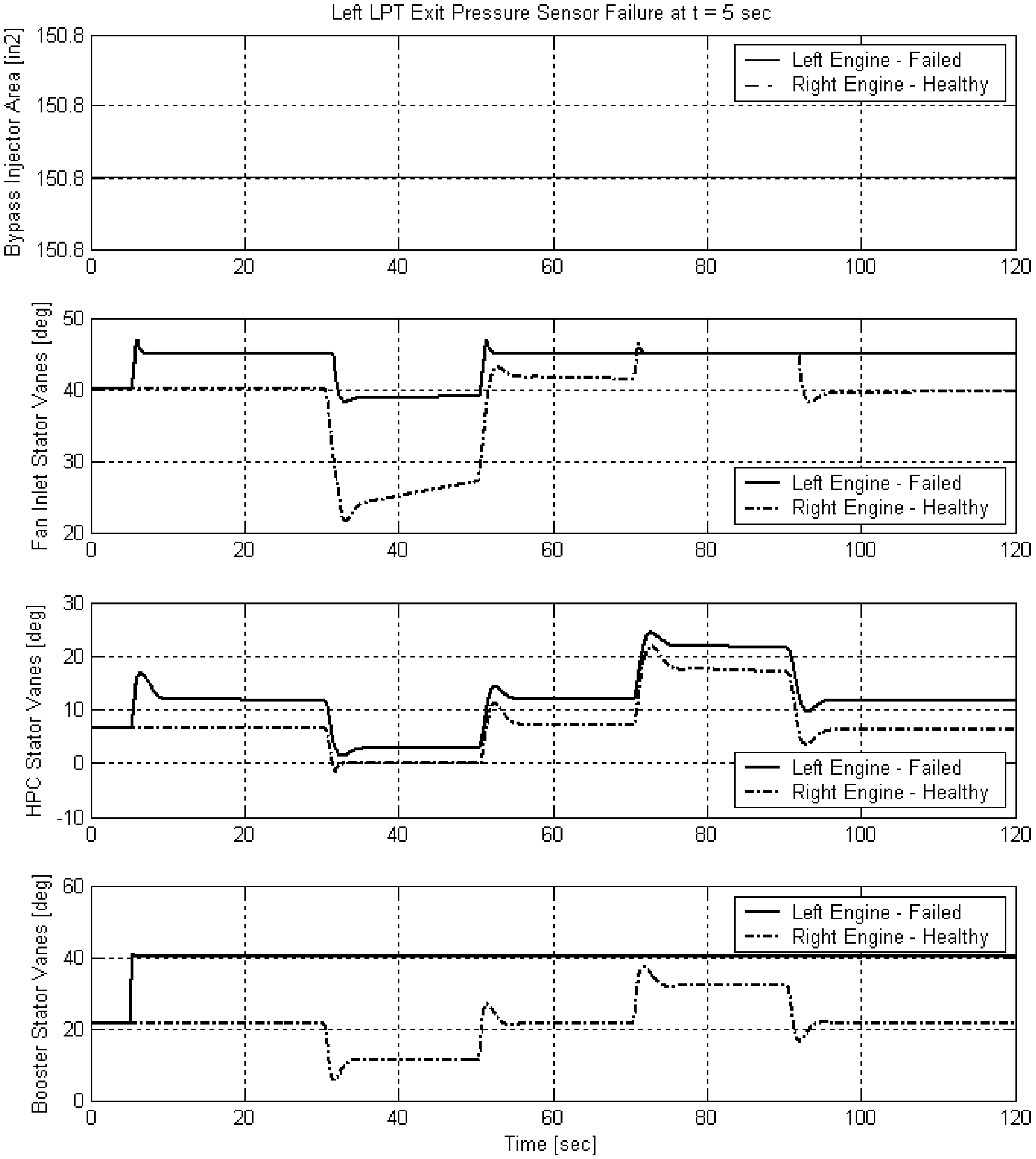


Figure A.38: Standard Throttle Input Response with a LPT Exit Pressure Min. Failure (3)

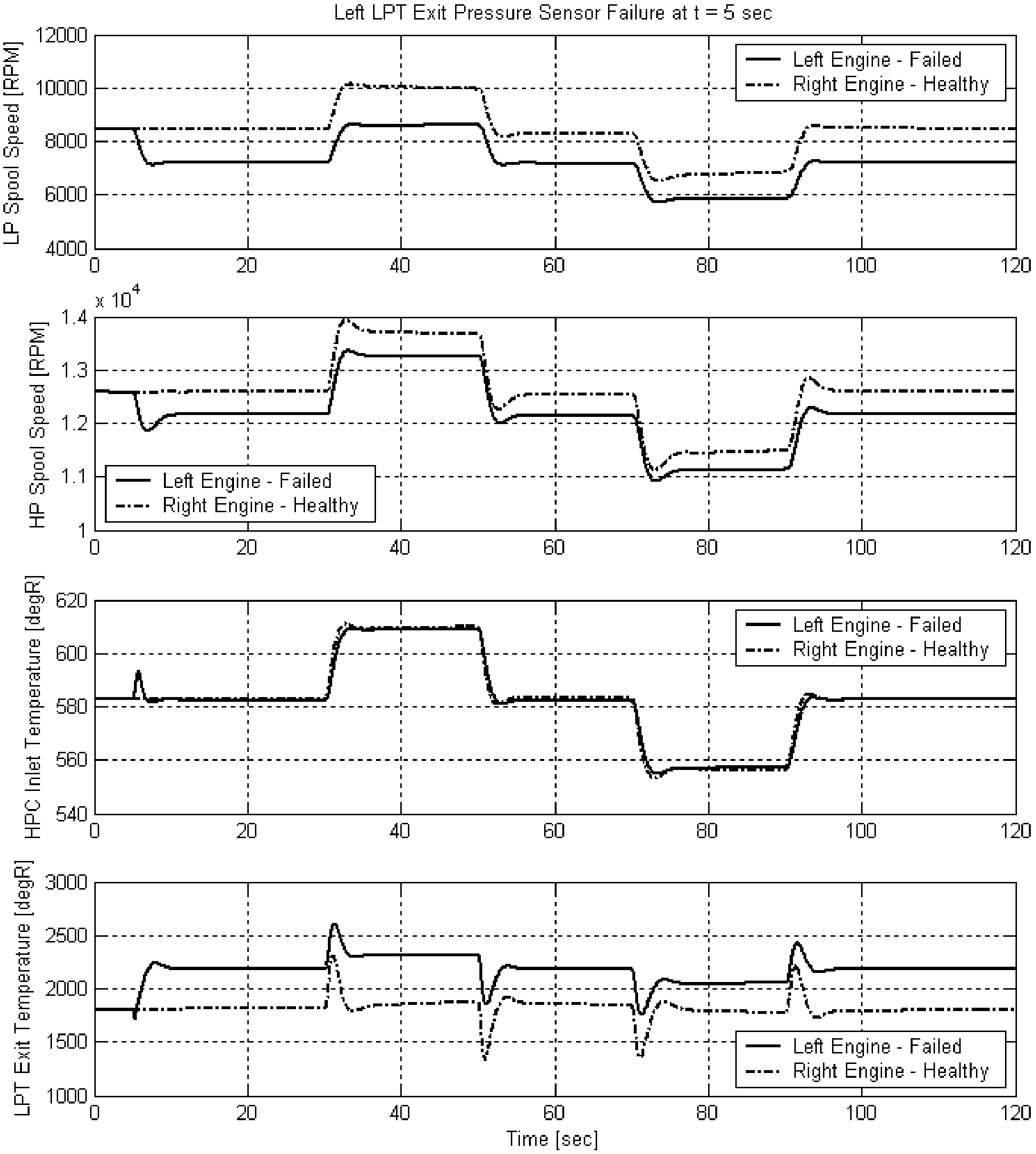


Figure A.39: Standard Throttle Input Response with a LPT Exit Pressure Min. Failure (4)

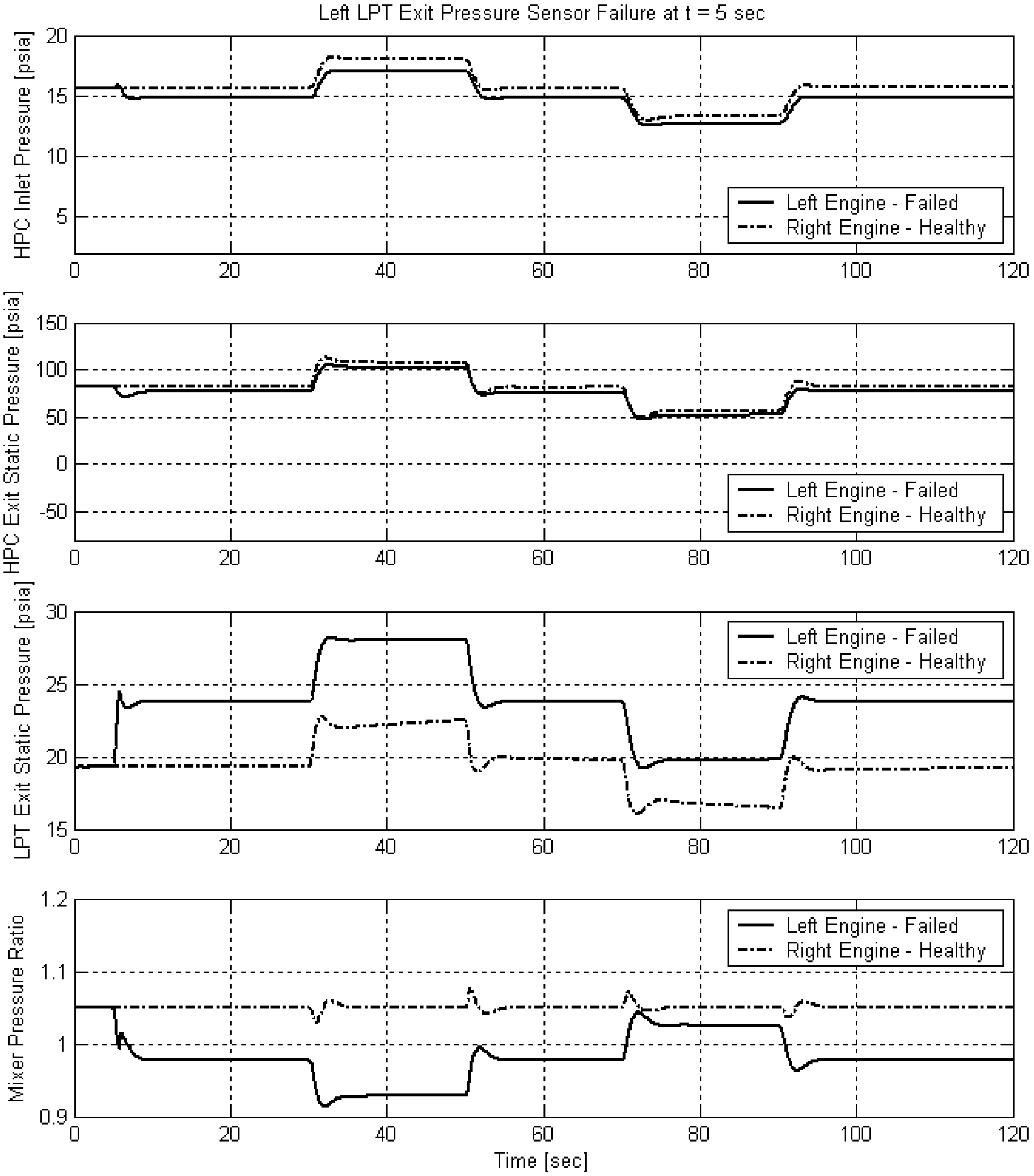


Figure A.40: Standard Throttle Input Response with a LPT Exit Pressure Min. Failure (5)

Left Engine Mixer Pressure Ratio Sensor at Maximum of 1.09:

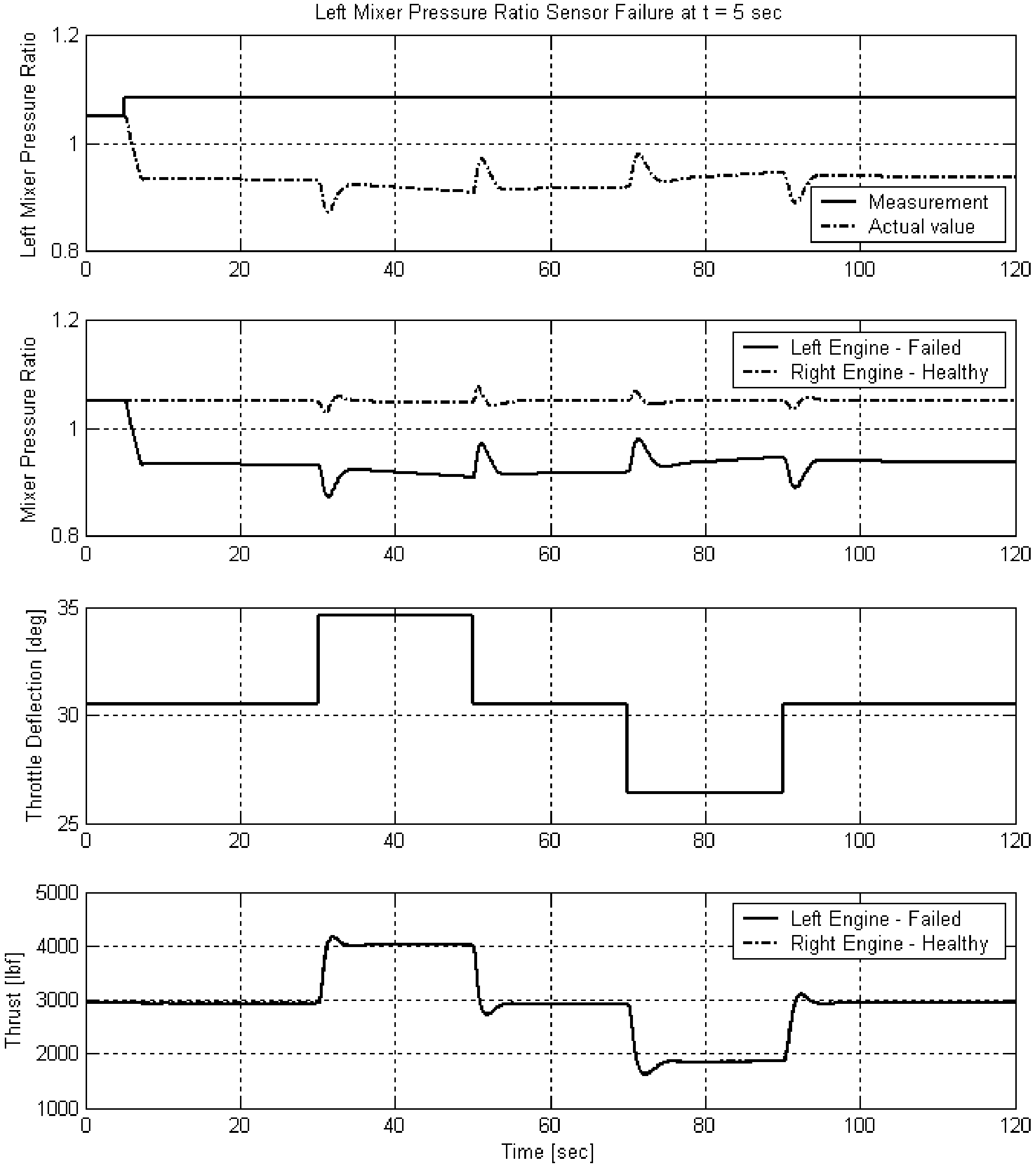


Figure A.41: Standard Throttle Input Response with a Mixer Pressure Ratio Max. Failure (1)

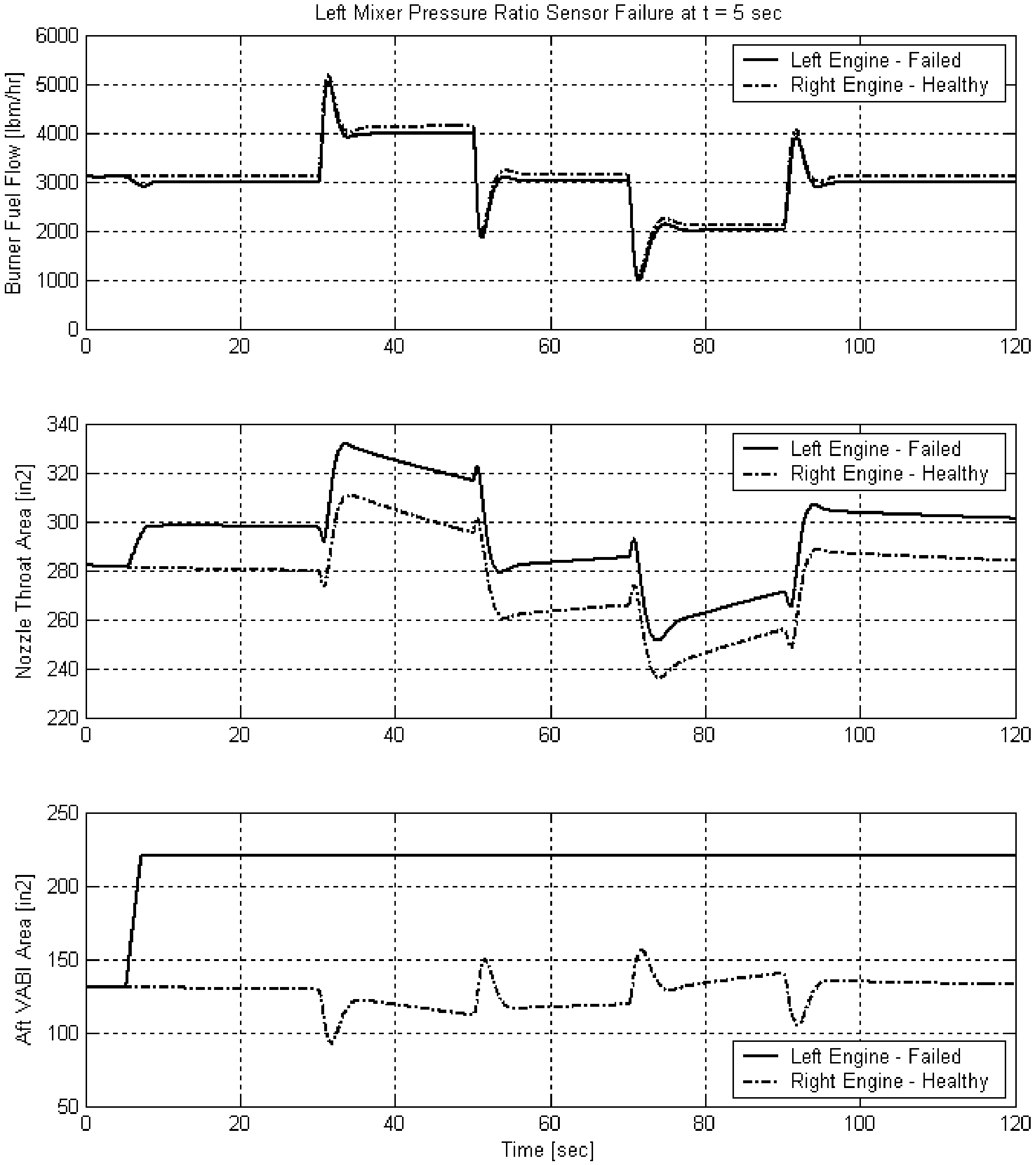


Figure A.42: Standard Throttle Input Response with a Mixer Pressure Ratio Max. Failure (2)

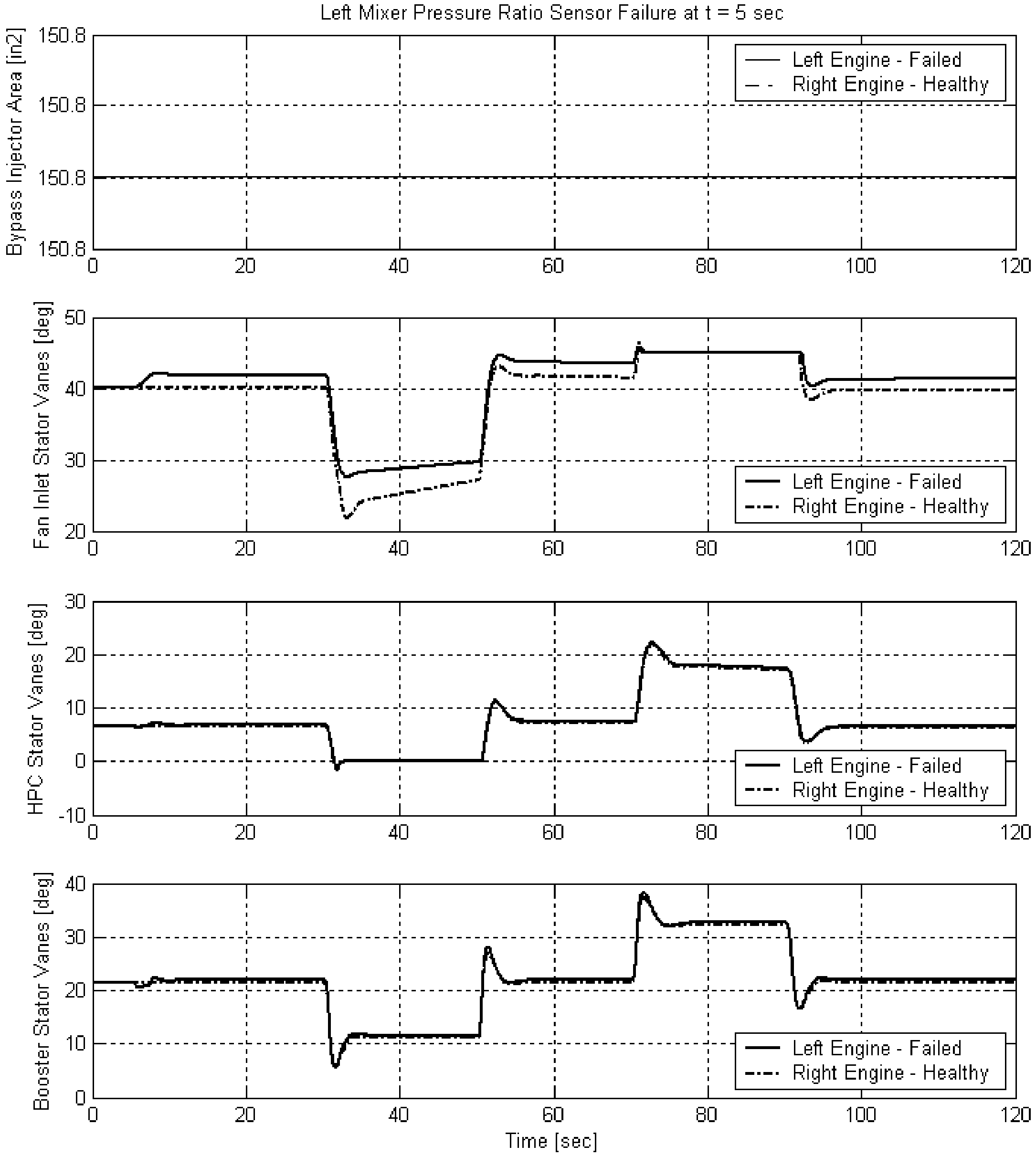


Figure A.43: Standard Throttle Input Response with a Mixer Pressure Ratio Max. Failure (3)

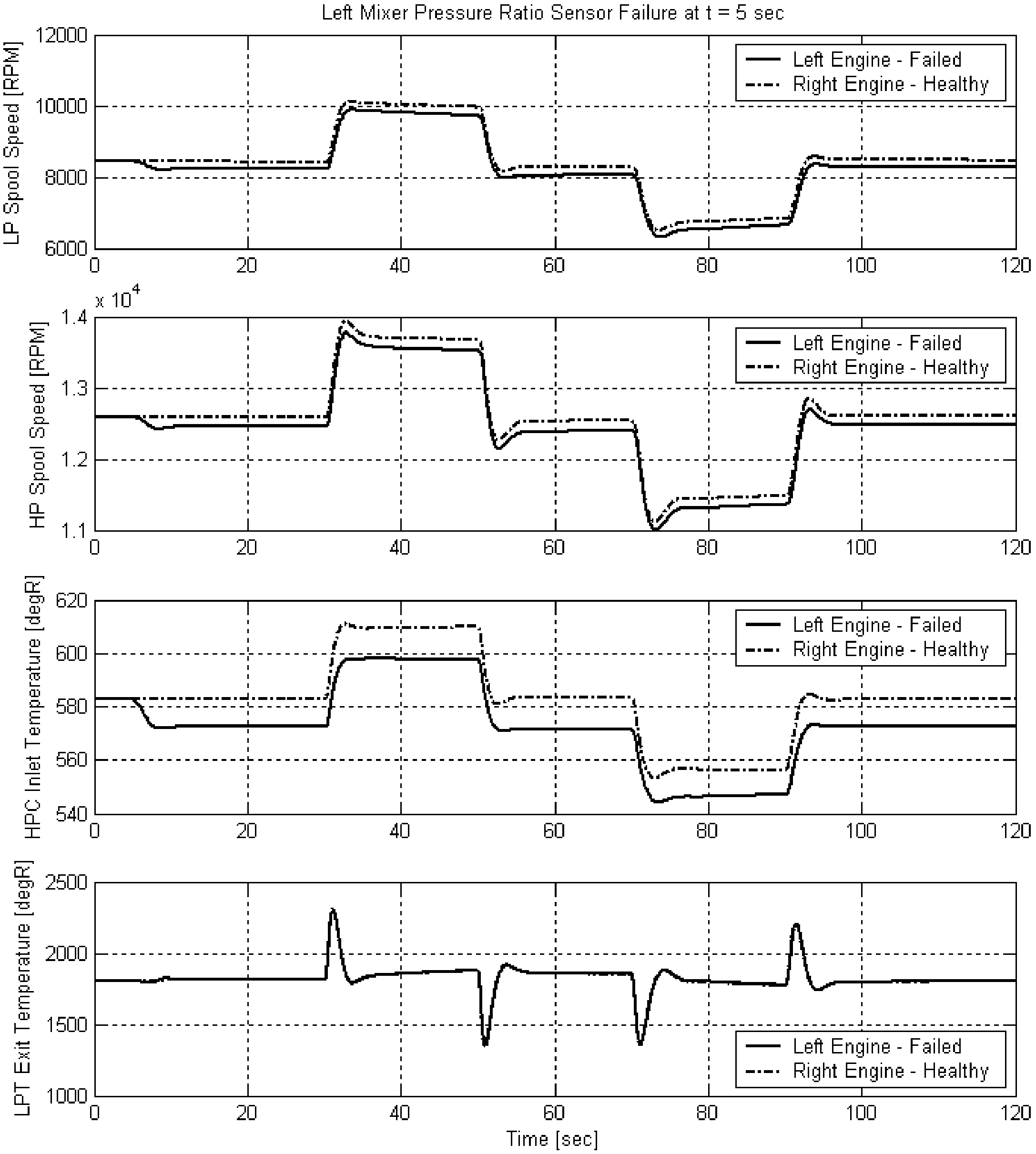


Figure A.44: Standard Throttle Input Response with a Mixer Pressure Ratio Max. Failure (4)

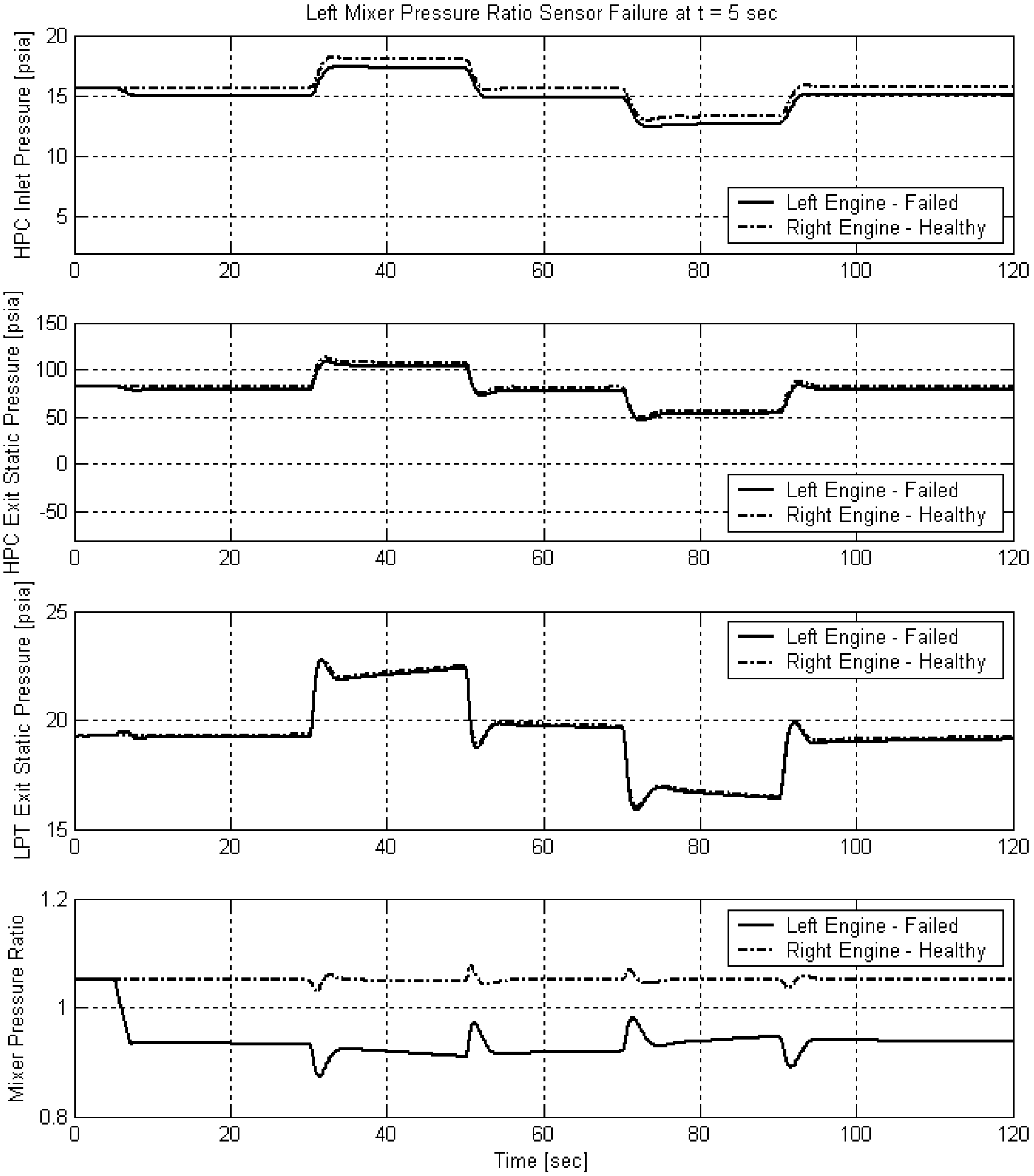


Figure A.45: Standard Throttle Input Response with a Mixer Pressure Ratio Max. Failure (5)

Left Engine Mixer Pressure Ratio Sensor at Minimum of 1.02:

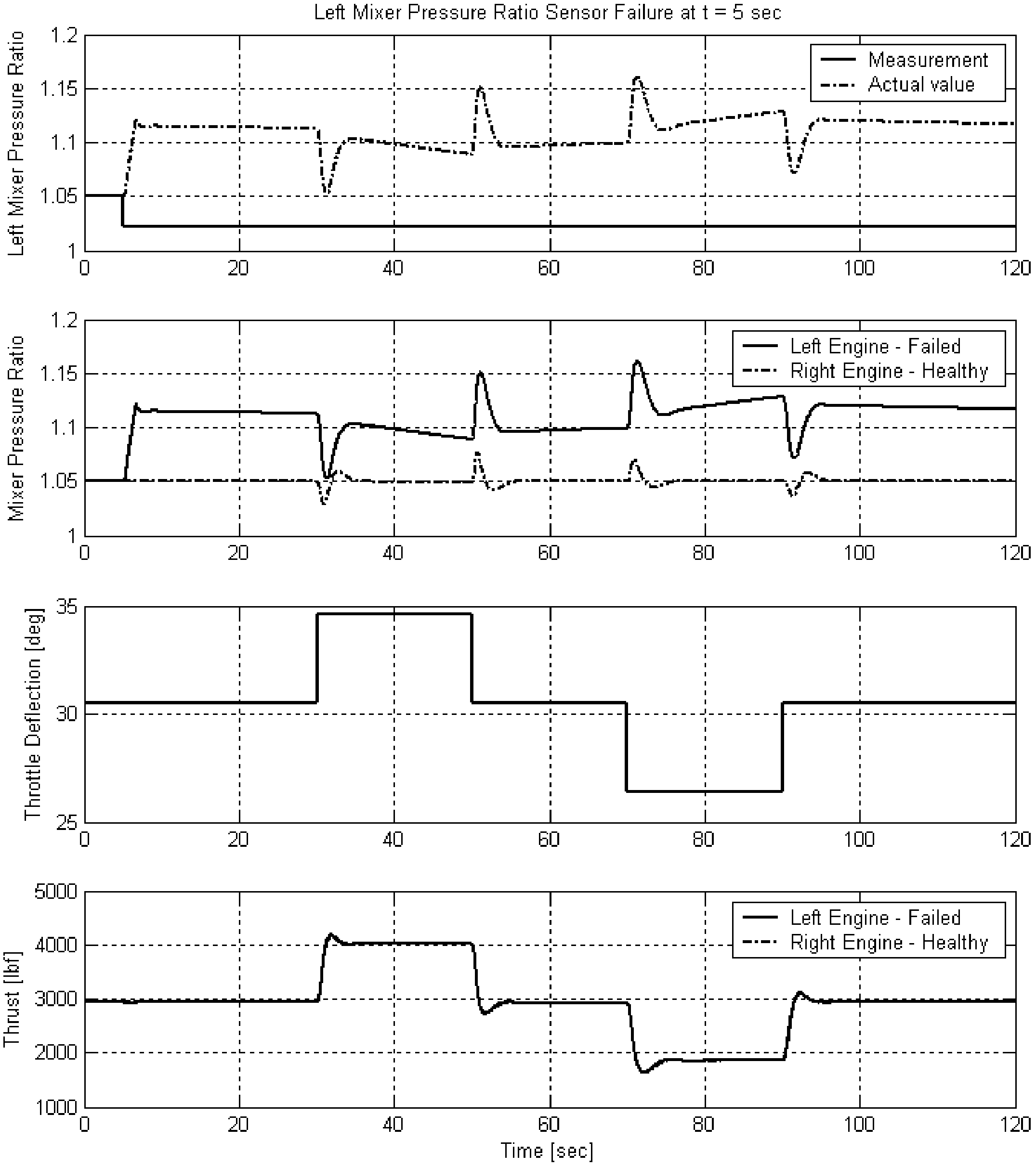


Figure A.46: Standard Throttle Input Response with a Mixer Pressure Ratio Min. Failure (1)

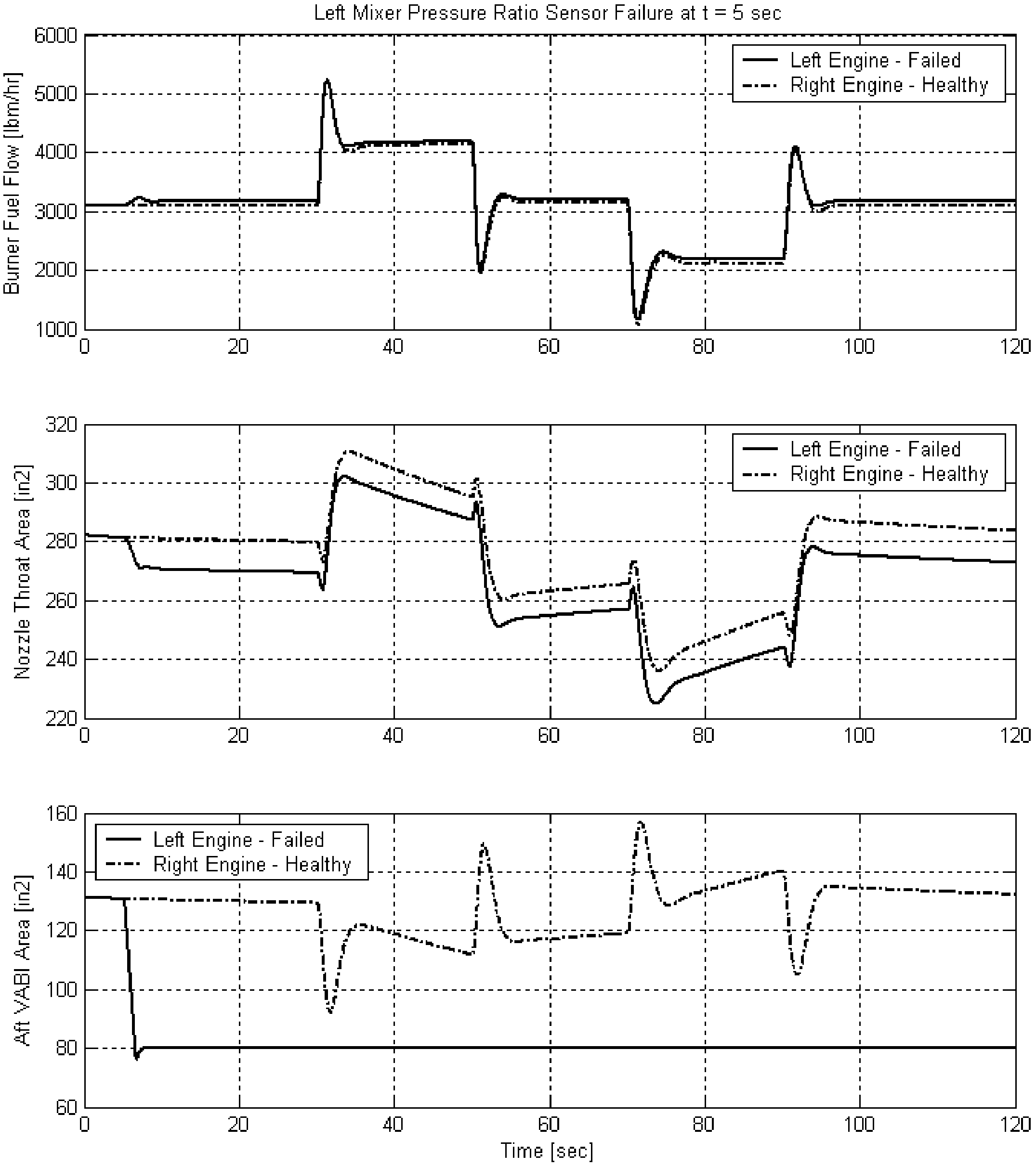


Figure A.47: Standard Throttle Input Response with a Mixer Pressure Ratio Min. Failure (2)

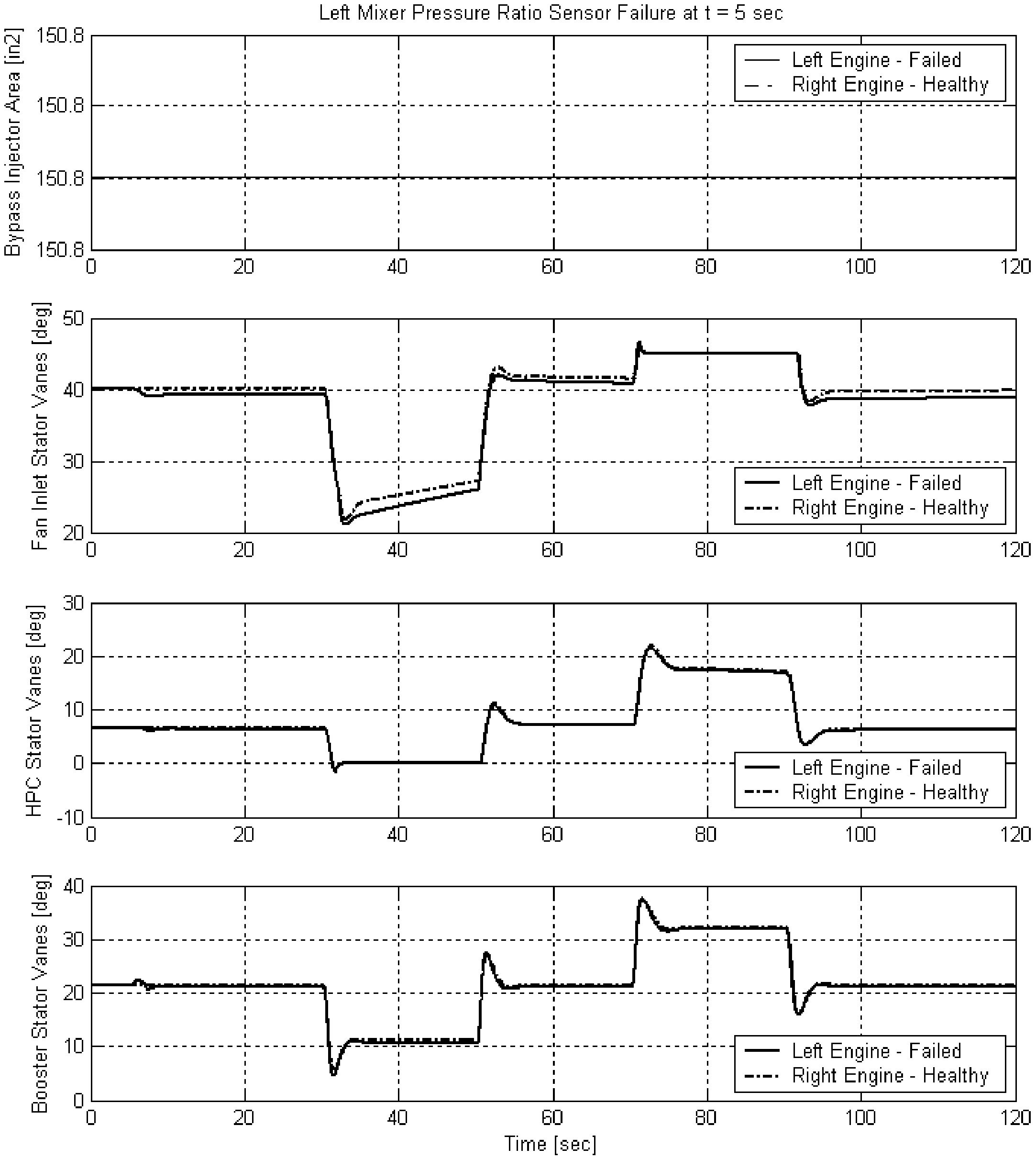


Figure A.48: Standard Throttle Input Response with a Mixer Pressure Ratio Min. Failure (3)

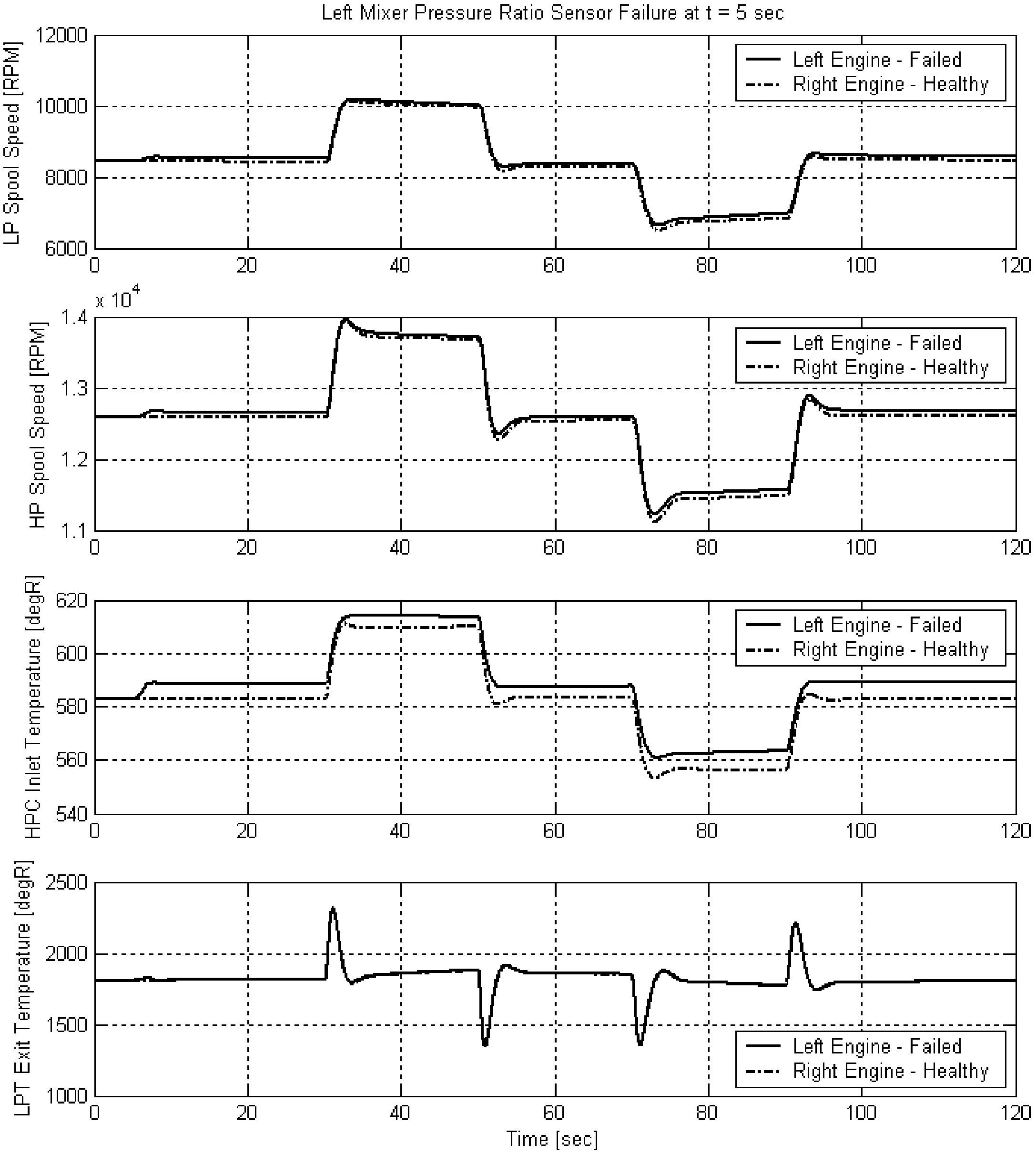


Figure A.49: Standard Throttle Input Response with a Mixer Pressure Ratio Min. Failure (4)

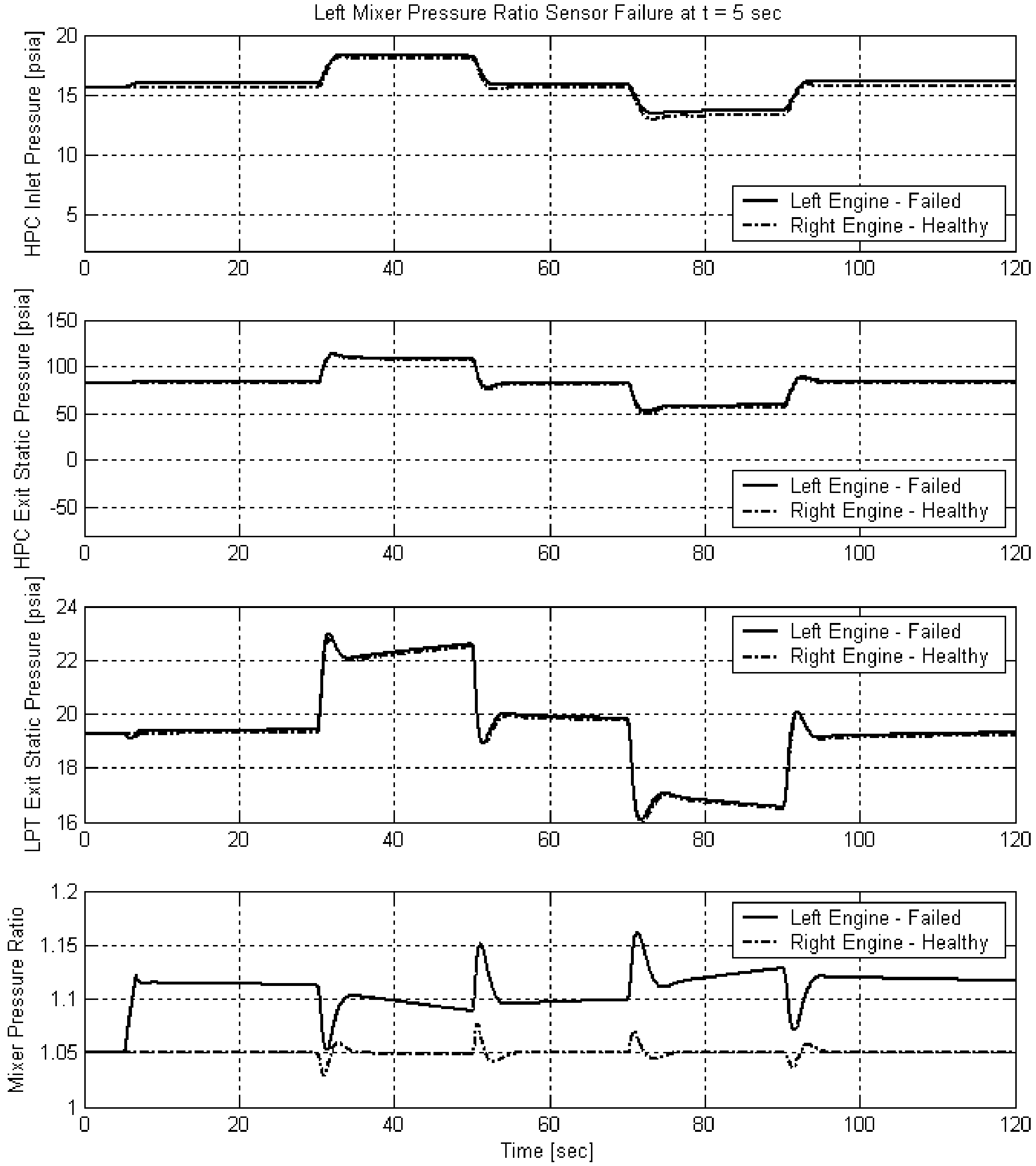


Figure A.50: Standard Throttle Input Response with a Mixer Pressure Ratio Min. Failure (5)