2021

Comparative Analysis of Different Classes of On-line State Estimators for Aerodynamics Angles and True Airspeed Sensors for Applications to the Sensor Failure Problem

Alexandra Anne Augsberger
aaaugsberger@mix.wvu.edu

Comparative Analysis of Different Classes of On-line State Estimators for Aerodynamics Angles and True Airspeed Sensors for Applications to the Sensor Failure Problem

Alexandra A. Augsberger

Thesis submitted

to the Benjamin M. Statler 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

Marcello R. Napolitano, Ph.D., Chair

Jason N. Gross, Ph.D.

Mario G. Perhinschi, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia

2021

Keywords: Sensor Failure Detection, Identification, and Accommodation, On-line Estimation, Analytical Redundancy, Neural Network, Kalman Filter

Abstract

Comparative Analysis of Different Classes of On-line State Estimators for Aerodynamics Angles and True Airspeed Sensors for Applications to the Sensor Failure Problem

Alexandra A. Augsberger

Throughout aviation history, there have been numerous incidents due to sensor failure that have caused a range of issues from loss of control of the aircraft to crashes resulting in loss of human life. Although there are many hardware-based solutions to this problem, the threat of control hardware failure still exists. This work investigates the efficacy of implementing neural networks (NN) and Kalman filters (KF) to solve the accommodation portion of the sensor failure detection, identification, and accommodation (SFDIA) problem through on-line real-time estimation of specific aircraft dynamic parameters. The implementation of on-line estimation architectures into the aircraft flight control system provides multiple advantages such as cost effectiveness and drastic decrease in weight. The multilayer perceptron (MLP) NN, extended minimal resource allocation (neural) network (EMRAN), extended KF (EKF), and unscented KF (UKF) have been evaluated in this effort for the purpose of providing analytical redundancy (AR) for estimating the parameter of the 'failed' sensor in lieu of physical redundancy. Each NN-based and KF-based estimator was compared using preset criteria including estimation accuracy, time to perform, and complexity of the model. The overall results have shown that the NN-based sensor failure accommodation (SFA) schemes outperform the KF-based SFA schemes with no undetected faults nor false alarms and significantly smaller estimation errors. More specifically, the EMRAN-based neural estimator has the best performance of all four schemes followed by the MLP NN, UKF, and EKF, respectively. This research shows the great potential of analytical redundancy-based approaches as opposed to physical or hardware redundancy to improved aviation safety for preventing future crashes due to sensor failures.

Acknowledgements

This research effort would not have been possible without the patience and guidance of my graduate advisor and thesis chair, Dr. Marcello Napolitano. I will always be thankful for the opportunities he has provided me over the years as both an advisor and professor. This gratitude is extended toward my committee, Dr. Jason Gross and Dr. Mario Perhinschi, whose advice and teachings have enhanced my skills and knowledge as an engineer.

I would like to thank Dr. Giampiero Campa for providing essential neural network tools and resources. Additionally, I would like to thank Dr. Mario Fravolini for providing the YF-22 and Tecnam P92 flight data for this research.

I would like to thank my family and friends who have always provided love support throughout my academic career.

Finally, a special thanks to Nick, Logan, and Freddy, whose friendship I cherish every day and for making my graduate experience more rewarding than I could have ever imagined.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADS	Air Data Systems
AI	Artificial intelligence
AR	Analytical redundancy
aug	Augmented
BP	Back propagation
CG	Center of gravity
CLME	Conservation of linear momentum equation
DOF	Degrees of freedom
DSMS	Data stream management systems
DT	Fault detection time
EE	Estimation error
EKF	Extended Kalman filter
EMRAN	Extended minimal resource allocation network
FA	False alarm
FD	Fault detection
FDS	Flight dataset
FDIR	Fault detection, isolation, and reconfiguration
GRV	Gaussian random variable
KF	Kalman filter
LQE	Linear quadratic estimation

MIMO Multiple input – multiple output

MLP Multilayer perceptron

MRAN Minimal resource allocation network

NN Neural network

RBF Radial basis function

SD Standard deviation

SFA Sensor failure accommodation

SFDI Sensor failure detection and identification

SFDIA Sensor failure detection, identification, and accommodation

SGD Stochastic gradient descent

SIM Simulated flight dataset

SIMT Simulated flight training dataset

TAS True airspeed

UAV Unmanned air vehicle

UKF Unscented Kalman filter

UF Undetected faults

UT Unscented transform

List of Symbols (English)

a	Parameters to be updated, scaling factor, acceleration
A	New parameters
b	Scaling parameter
B	Control input model
c	Center value
e	Error, estimation error
E	Error function
F	State transition model matrix
g	Gravity
H	Observation model matrix
i, j, k	Interval, timestep
I	Identity matrix
K	Kalman gain
L	Dimension
n	Size
O	Sigmoid function derivative
p	Roll rotation rate
P	Prediction matrix, estimated accuracy of state estimate
q	Pitch rotation rate
Q	State noise covariance matrix
r	Yaw rotation rate
R	Measurement noise covariance matrix
S	Pre-fit residual covariance
u	X component of airspeed velocity
U	Control vector
v	Y component of airspeed velocity
w	Z component of airspeed velocity
W	Weight, weight vector, weight matrix
x	State

$\hat{x}$	State estimation
$\dot{x}$	State space
y	Output
$\hat{y}$	Output estimation
z	Observation of state

List of Symbols (Greek)

α	Angle of attack
β	Sideslip angle
Γ	Output with respect to neuron center
δ	Error signal
Δ	Difference
ξ	Overlap factor
η	Learning rate
θ	Euler pitch angle
Θ	Weighted bias value
κ	Secondary scaling parameter
λ	Scaling parameter
Λ	Activation function boundary limit
μ	Momentum
σ	Width
ϕ	Euler roll angle
Φ	Computing unit
χ	Sigma vector matrix
ψ	Euler yaw angle
Ψ	Matrix of output sigma points

Introduction

a. Motivation

On-line estimation has been a prominent control failure prevention method proposed for sensor failure scenarios in aircraft for many years. Control systems utilize on-line estimation to optimize both performance and safety. Neural networks and Kalman filters, two different classes of on-line estimators, are explored in this work to solve the sensor failure detection, identification, and accommodation (SFDIA) problem for both simulated and real aircraft data. The SFDIA problem was one of the first to address sensor failures through on-line estimation in works such as Samy *et al.* [1], Fravolini *et al.* [2], and Gururajan *et al.* [3]. SFDIA was a problem originally proposed for sensor failures on aircraft but can be applied to any control system. This paper will focus on the sensor failure accommodation (SFA) portion of the SFDIA. This topic has been researched strenuously as it can prove lifesaving in critical sensor failure situations.

b. Background

On-line estimation is a branch of on-line machine learning in which data becomes available in a sequential order and is used to update the best prediction for future data at each step [4]. An advantage of on-line estimation is its ability to accurately predict future data without analyzing the entirety of large sets of data. It also allows built-in algorithms to adapt to dynamic patterns commonly seen in aircraft data. Neural networks and Kalman filters provide two very different but effective approaches for the design of on-line estimators to accommodate sensor failures.

A NN is an artificial intelligence scheme composed of neurons organized in multiple layers that determines underlying relationships within large datasets. NNs have been used as nonlinear dynamic system controllers to address problems for which conventional approaches have been proven to be ineffective [5]; however, because the learning process is computationally demanding (either through on-line or off-line training), the practical use of NNs for on-line control schemes is limited, especially in areas such as flight controls [6, 7]. An additional issue that has prevented a much broader application of NN-based schemes (either as controllers or as estimators) is the lack of reliable validation tools for the certification process set by a regulating agency such as the Federal Aviation Administration. Hence, the problem of designing fast on-line learning algorithms for practical implementation of neural control schemes remains an active research topic [8].

KFs are a form of Bayesian on-line estimation first developed by Rudolf E. Kálmán and Richard Bucy [9] that utilize linear quadratic estimation (LQE). KFs, like NNs, use a series of measurements observed over time to produce estimates of unknown variables. By estimating a joint probability distribution over the variables for each timeframe, KFs tend to be more accurate than estimations based on a single measurement alone. Kalman filtering uses a system's dynamic model, known control inputs to that system, and multiple sequential measurements (such as sensors) to form an estimate of the system's varying states. As such, it is a common sensor fusion and data fusion algorithm.

This research will focus on the sensor failure of three specific variables, that is angle of attack (α), sideslip angle (β), and true airspeed (TAS). These specific variables have been selected because they are actively measured through sensors (aerodynamic vanes and pitot tubes) located outside the aircraft which are more likely to be damaged or interfered with than other sensors

(such as gyros and accelerometers) that are installed inside the aircraft. There have been multiple works in the past dedicated to the nonlinear estimation of each of these variables using flight data. Different on-line estimation schemes including the multilayer perceptron (MLP) NN and the unscented Kalman filter (UKF) have proven to be suitable approaches to the SFDIA problem [1, 10]. This work investigates and provides a comparative analysis of four different SFDIA schemes' abilities to accurately accommodate sensor failures within simulated and real aircraft.

c. Research Organization

The primary goal of this research is to determine the best on-line estimator to be used for the accommodation portion of the SFDIA problem. The comparative analysis will be performed using ad-hoc defined performance criteria. This thesis is organized as follows.

Chapter 1 provides an in-depth literature review of the sensor failure problem.

Chapter 2 discusses the challenges of this research, the selection of performance criteria, and expected performance of each on-line estimator.

Chapters 3 describes the two NNs in focus for this research, that is the multilayer perceptron (MLP) NN trained with the back propagation (BP) algorithm and the radial basis function (RBF)-based extended minimal resource allocation network (EMRAN). This chapter also provides a review of the original NN algorithms and their modifications over recent years.

Chapter 4 discusses the KFs used in this research, that is the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). This chapter provides the history and original algorithms of the Kalman filter and how the KF has developed over time to accommodate nonlinear systems.

Chapter 5 discusses the design of the sensor failure accommodation (SFA) schemes within the WVU flight simulator. The results of all four on-line estimators' analyses and accommodations using simulation data is discussed.

Chapters 6 and 7 analyze the results of each on-line estimator when tested with real flight data from the WVU YF-22 research aircraft and Tecnam P92 aircraft, respectively. These chapters provide evidence supporting which of the four on-line estimation schemes performs best when accommodating sensor failures for angle of attack, sideslip angle, and true airspeed.

Finally, Chapter 8 provides the overall conclusions of this comparative study as well as recommendations and plans for future work.

Figure I: WVU YF-22 Unmanned Research Aircraft. Photo courtesy of Dr. Marcello Napolitano.

Figure II: Tecnam P92 [11]

Chapter #1: Literature Review

On-line estimators have had multiple applications to real-world issues; specifically, over the last two decades, significant research efforts have been directed towards increasing operational safety of aircraft through developing fault tolerant flight control schemes. Aircraft sensor errors have been found to be the cause of numerous plane crashes over the course of flight history [5]. At times, crash investigations that concluded that pilot's error was the cause of the crash were later reversed when evidence proved the occurrence of sensor failures. Issues related to sensor failures were typically considered lower priority mainly due to the reliance on physical redundancy.

Physical redundancy defines the category of hardware in aircraft responsible for fault tolerance. The physical redundancy of the sensors, along with the implementation of Built-In Testing or voting schemes, have shown to be reliable approaches to manage failures for most of the flight control system sensors. For this reason, physical redundancy in the sensors to date has been the only accepted approach for manned aircraft by the FAA despite the proof provided by many research efforts regarding the success of analytical redundancy [3]. According to the FAA's Reliability, Maintainability, and Availability handbook section 7.1.4, the redundancy and fault tolerance first determinant must be the hardware architecture [12]. There must also be an "adequate number" of hardware elements in the aircraft dedicated to fault tolerance [12]. Analytical redundancy, due to these standards, does not currently meet FAA requirements.

Despite physical redundancy being the only internationally accepted fail-safe for any aircraft, it has not been completely risk or accident free. Sensors installed outside the aircraft that are dependent on physical redundancy are susceptible to external factors (such as bird) and environmental factors (such as the formation of ice on the external sensors). For this reason,

physical redundancy schemes do not allow the same level of robustness to sensor failures for external sensors compared to internal sensors. For example, if one Pitot tube fails due to environmental factors, it is statistically likely that the other Pitot tubes will experience the same failure; this is known as the "common mode" failure. Analytical redundancy has gained more popularity in recent years because it relies solely on the control system inputs and outputs of the aircraft, or the software, and is not impacted by external nor environmental factors.

Plane crashes in the past have proven that physical redundancy is not a completely foolproof method when accommodating external sensor failures. Notable crashes due to Air Data Systems (ADS) sensor failure include the NASA X-31 research aircraft and the Aeroperu B757 [3]. According to the aviation safety database, there is technical evidence of over ten ADS failures for all Pitot tubes over the past three decades leading to significant damage and/or fatalities [3, 13]. Other sensor failure-induced plane crashes in the past include the infamous Boeing 737 crashes caused by both inaccurate speed and angle of attack data [6], Korean Air Cargo Flight 8509 crashing due to incorrect data from the attitude director indicator [7], and Air France Flight 447's iced Pitot tubes that lead to an inevitable crash [8]. These cases displayed the flaws of physical redundancy through common mode failures and environmental factors that could have potentially been avoided with the use of analytical redundancy.

The first models used to accommodate sensor failure originated from state estimation, a set of mathematical models that reconstruct the state information of a system [13]. In this method, unmeasured state variables, or the unknowns of the problem, are estimated based on available model output predictions and noisy measurements. This is a crucial property of SFDIA; when a sensor fails, it is necessary to estimate what the actual sensor values must be based the sensors' inputs. Thomas Bayes was the first main contributor to the state estimation problem

with the development of Bayesian theory in the 1700's. This served as the foundation of statistical inference methods, which was the basis of on-line estimation schemes [14]. One of the more popular sensor failure schemes designed and utilized aside from SFDIA is the Data Stream Management Systems (DSMS) method. The DSMS scheme is more prevalent in transportation system designs but there have been applications of this method for aviation. As mentioned previously, this research centers around the SFDIA problem, also sometimes referred to as the Fault Detection, Isolation, and Reconfiguration (FDIR) problem that has been researched as early as the mid-1900's in works by Kolmogorov [15], Wiener *et al.* [16], and Kalman [17]. Since SFDIA was introduced into the aviation industry, the topic has been studied continuously to propose more reliable solutions to the sensor failure problems aircraft can face.

1.1 State of the Art of the SFDIA Problem

SFDIA has been utilized for numerous research applications over the past few decades and has proven to be capable of solving faulty sensor problems. The research efforts toward SFDIA have seen drastic improvements over time to where the state of the art of SFDIA is today.

The tasks of SFDIA systems today are to generate residuals when a sensor failure is detected, identify the sensor or fault, and accommodate the error based on the type of failure using preprogrammed data or recalculating the parameters on-line [2]. These approaches typically divide the process into two steps, that is sensor failure detection and identification (SFDI) and sensor failure accommodation (SFA). The SFDI will detect and identify the source of failure within the control system without physical redundancy. Once the SFDI is triggered, the SFA initiates and inputs a new estimate to the control system to replace the failed sensor data.

There have been two main approaches toward solving the SFDIA problem: model-based and not model-based. Both the NN and KF have been studied as potential permanent solutions to the SFDIA problem in aircraft control systems from the UAVs such as the YF-22 and aircraft such as the Tecnam P92. There have been many works focused on this issue in the past that have compared both the KF and NN approaches for various estimations of failed sensors.

At the most basic form, SFDIA models will compute the residual, or difference between the model estimate and actual sensor measurement. If the difference surpasses a user-implemented boundary, a fault is declared that initiates the SFA process. The model-based approaches consist of various KFs. Model-based schemes rely on a system of equations, or mathematical descriptions, of the system. The main model-based method used for SFDIA problems is the observer-based Kalman filter such as extended and unscented [18]. Although the KF is efficient in terms of cost and time to develop, there have been many issues observed in different works such as Samy *et al.* [1], Gururajan *et al.* [3], and Alexander *et al.* [14] including computational complexity, estimation discrepancies within the mathematical model compared to actual data, and robustness to nonlinearities. These difficulties associated with the KF have led to issues regarding SFDIA including false alarms (FA) that has diminished the probability of this analytical redundancy method being implemented into future aircraft control systems.

In contrast to the Kalman filter and other model-based methods, NNs have gained popularity over recent years as they do not rely on a strict mathematical model [18]. NNs have been favored in past works due to their reliance on actual data instead of systems of equations. By relying on data, the NNs have the capability to recognize input-output relationships within the data that are not fully captured by human observation nor mathematical equations. NNs have accurately estimated nonlinear functions with just one hidden layer and sufficient training data

[19, 20]. Although there have been KF models introduced in the past that have been successfully applied to other nonlinear systems like EKFs and UKFs, the NNs have no equal in terms of their structural simplicity, computational speed, and nearly unmatched fault tolerance. The NN has been implemented into SFDIA problems in recent years as they adaptively reduce modeling errors and FAs over time unlike the KF. Although it seems as though the NN is the clear choice to resolve SFDIA problems, there are many disadvantages associated with a seemingly reliable method. One of these disadvantages includes the amount of data required for training the network. A NN is nearly impossible to implement into brand new problems if there is no data available to train the network. For this same reason, the NN can be costly if there is a need for additional data. The NN can also require days to months to train depending on the amount of data whereas the KF can be used immediately after the completion of the design.

1.2 Related Work

The SFDIA problem has been researched in numerous works through different NNs, KFs, and other parameter estimation techniques to accommodate sensor failure on UAVs, aircraft, and spacecraft. Many of the schemes mentioned have also been compared to one another to determine the best on-line estimator.

NNs have been investigated thoroughly for the sensor failure problem in various works such as Fravolini *et al.* [21], Campa *et al.* [22], and Napolitano *et al.* [23]. Fravolini *et al.* displayed the competence of nonlinear NNs to characterize the nonlinear response in different phases of flight. The NN in this work was able to adapt to the changing data patterns of a Tecnam P92 over time. The NN was able to detect failures with low false alarm rates and high robustness despite the change in behavior of flights. In works by Campa *et al.* and Napolitano *et al.,* the extended minimal resource allocating NN (EMRAN), was investigated for aircraft

SFDIA and compared to the performance of a MLP NN. In this work, both the EMRAN and MLP were able to successfully detect, identify, and accommodate failed sensors in a 6 degree of freedom aircraft simulation. In this study, the EMRAN NN performed faster and more accurately than the MLP.

The KF has also been implemented into multiple sensor failure related studies including notable works by Rhudy *et al.* [24] and Wan *et al.* [10]. Modified versions of the KF such as the Extended Kalman filter (EKF) and Unscented Kalman filter (UKF) were utilized for nonlinear flight dynamics systems in these cases. In the work by Rhudy *et al.*, the nonlinear KF was the analytical redundancy-based approach of choice to accommodate the true aircraft speed without the Pitot probe. This was one of the first works to incorporate aviation data from a manned aircraft rather than an UAV. Two different stochastic wind models, the random walk and Gauss-Markov based on the nonlinear EKF, were successfully implemented and provided satisfactory results for airspeed estimation in this work. This work created a solid foundation for on-line estimation through an EKF-SFDIA scheme for manned air vehicles and proved it can be used as a standard technique within aircraft controls. The work by Wan *et al.* compared the EKF and UKF for the SFDIA problem. This work compared the EKF and UKF using datasets of multiple orders. The EKF was sufficient with first order systems but had large errors with higher order data. On the other hand, the UKF was capable of computing estimations with higher order data through unscented transform (UT). This work displayed the competence of both the EKF and UKF in terms of estimation capabilities. As of today, the UKF can compute higher-level order accuracy at the same degree of computational complexity as the EKF.

There are also recent works in which a comparative analysis was conducted between NNs and KFs to determine the best on-line estimator specifically for aircraft data. These comparative

works include Napolitano *et al.* [26] and Fravolini *et al.* [27]. Both studies compared the performance of the NN-SFDIA and KF-SFDIA schemes with nonlinear UAVs. The NNs outperformed the KFs in terms of estimation accuracy and false alarms in both studies but the KFs still performed satisfactorily.

Based on these past works, on-line estimators today have proven to be reliable implementations to control systems of all types but has yet to be installed permanently into any aircraft. In general, previous works have shown that although both the NN and KF provide satisfactory results and performance, the NN is favorable in terms of computational power and estimation accuracy. This research will provide more insight into each online estimator to determine which is the optimal scheme for each type of sensor failure in both simulated and real aircraft.

Chapter #2: Challenges and Criteria of the On-Line Estimators

This chapter discusses the challenges and the taken into consideration when undergoing this research. To increase the chances of successfully developing NNs and KFs for SFDIA, there were many potential challenges that needed to be noted before initiating the research. Further on in this chapter, the expected performance of each on-line estimator, based on the analysis criteria, was discussed.

2.1 Challenges

2.1.1 Robustness and Sensitivity

Crucial properties of on-line estimation schemes are their robustness and sensitivity to changes in behavior of the data. There is a wide variety of flight maneuvers and paths that need to be analyzed by each on-line estimator in this work. When the on-line estimator is properly sensitized to the training data, it will perform well when tested with new data. Lack of robustness or sensitivity can lead to poor estimation of new testing flight data. The SFA schemes were expected to handle any type of flight data with a low false alarm rate and low number of undetected faults [3]. To guarantee optimal sensitivity, it was critical to correctly characterize each scheme's uncertainty bounds, which presents a new challenge [28]. The testing of robustness and sensitivity is a challenge as there is no consistent or reliable algorithm for determining the appropriate parameters that will guarantee the capability to both be able to handle all data types and to also detect any fault severity. Any parameter within the NN or KF can further complicate this process as they each can have a positive or negative impact on both factors [21]. Neural architecture search algorithms are a relatively new tool designed for the purpose of optimizing robustness; however, these methods are not yet consistent nor reliable. Matrices in the NN and KF that are directly associated with robustness and sensitivity,

covariance and uncertainty, are typically defined through trial-and-error, which can be tedious to perform and inconsistently effective.

2.1.2 SFDIA: Sensor Failure Detection

Another challenge that affects both the NNs and the KFs is detecting minor failures within the flight data. The severity of a sensor failure can vary from unnoticeable to catastrophic. When considering more subtle sensor failures, although it may seem unimportant at first, it could become progressively worse throughout the flight; therefore, it is crucial for the on-line state estimators in this work to detect the sensor failure immediately no matter the severity. This could prove to be a challenge as the smaller sensor failures may still seem "correct" to the estimator if the incorrect values are still within what is considered a safe range of the variable.

2.1.3 SFDIA: Sensor Failure Identification

Sensor failure identification proves to be a challenge as many of the variables under observation are similar in terms of magnitude. In the case of the flight data, depending on the maneuver, for example, some of the angles may appear identical. If one of the identical sensors fails, the NN or KF could mistakenly try to accommodate the incorrect sensor. Although this research primarily focuses on the accommodation portion of the SFDIA problem, the potential issues with sensor identification are noteworthy as this step is still crucial to the overall SFDIA process.

2.1.4 SFDIA: Accommodation

The accommodation portion of the SFDIA problem is the core of this research. It is arguably the most important component of the SFDIA process as it can either completely resolve the sensor failure or exacerbate it. The estimation itself will present a challenge as there are

multiple variables the on-line estimators will need to consider while simultaneously deciphering a pattern within the nonlinear, dynamic flight data.

The uncertainty of the NN and KF is a challenging component of the designs as it can singlehandedly be the difference between an accurate estimation and a failed accommodation. Any parametrically distributed disturbance, such as a Gaussian noise, can result in a non-standard output measurement if uncertainty is not properly defined [14]. This uncertainty can also fluctuate depending on the variable. Uncertainty of the on-line estimator affects how accurately it recognizes patterns within a system and predicts future inputs. The main goal for quantifying uncertainty is to design a model that accurately represents the real dynamic stochastic process or find the state variables and the parameters of the system's model from available measurements [14].

NNs and KFs each have separate methods for quantifying the uncertainty. Developing high-fidelity uncertainty for NNs is not as critical as it is for KFs but is important to recognize for the sake of comparison. NNs typically apply various Gaussian processes to determine uncertainty. The most successful Gaussian process for the system is determined to be the method with the least false positives. A similar approach was used for the NNs in this research. KFs, on the other hand, have multiple well-tested methods to accurately quantify uncertainty. Most of these methods are stochastic to better suit the nature of the KF. To address the potential challenge of encountering parameter uncertainty, the EKF and UKF simultaneously estimate the parameters along with the states while augmenting the parameters as pseudo-state variables [13]. Determining uncertainty for these reasons proves to be a challenge as it is both difficult to define and can fluctuate in efficacy if other portions of the NN or KF are modified.

2.1.5 Neural Network Challenges

NNs are a widely used branch of artificial intelligence due to their ability to analyze large datasets compared to other types of AI; however, this also proves to be a disadvantage and challenge for a few reasons. The first challenge of the NN is the large amount of diverse data that is required for training and testing. Depending on the research, it can be difficult to gather enough data for the NN to learn or recognize patterns. The data provided to a NN must also be diverse to prevent the NN from 'memorizing' a single trend or individual training dataset. In the opposite cases where there is plentiful data, more data can greatly increase the time required to train or test the NN. RBF NNs in general can often suffer from what is known as "curse of dimensionality:" the more data presented, the more often the NN will require an exponentially increasing number of hidden neurons. This issue is addressed in works such as Samy *et al.* [1] and Li *et al.* [8]. The training will also require more data if there are more variables involved. NNs for these reasons are not always favored as they require more time to be designed and more time to analyze data. This will be a challenge in this research as there are large amounts of data to analyze multiple times over a short period of time. This research considers more than forty variables in each flight dataset and more than one hundred flights will be studied and analyzed by each NN.

The next challenge specific to the NN is the computational power required compared to traditional algorithms. The computational power needed for a NN depends not only on the size of the data but also on the complexity of the network. For example, a NN with one layer and 50 neurons will be much faster than a NN with 50 layers and 1000 neurons. More data should lead to lower estimation errors, but more data can be extremely time consuming and may not be feasible for the average computer to handle.

Finally, the main disadvantage of the NN unrelated to data is their "black box" algorithm. The "black box" of the NN is the term used to describe activity of the NN in between the input and final output [26]. Although there are outlines of the processes and algorithms within the NN, if the output is incorrect, there is almost never a way to pinpoint the exact issue as to why the output was inaccurate, rendering the system unadaptable. This leaves the main solution to this problem to be trial and error of readjusting certain aspects of the NN in hopes that these changes will fix the issue. This has been a fundamental reason for the FAA rejecting the use of analytical redundancy on aircraft.

2.1.6 Kalman Filter Challenges

Although the Kalman filter has plenty of benefits and applications to the real world, there are multiple fundamental issues within the KF that specifically affect nonlinear control systems. The overall challenge of KF algorithms is their finicky nature; a simple error in code or practice can deteriorate the optimal performance. Also, the simulation of nonlinear data creates more room for error as it was originally designed for linear systems. A similar issue arises if there is a mismatch between the state-space and measurement models for which the KF was designed and the realistic model, the KF will produce inaccurate results. Lastly, there are potential challenges when matching the actual system and measurements to the Kalman filter measurement noises.

Designing a Kalman filter has proven difficult due to the complex methods required to calculate covariances [14]. The covariance matrix of the on-line estimator is a square matrix that represents the deviations of two or more variables from their respective means. Essentially, this matrix numerically explains how the uncertainty of each variable is related to one another. By noting the relation between two variables, the on-line estimator can also predict how the two

variables will simultaneously change with respect to each other. Therefore, many state estimation techniques, including the techniques used in both NNs and KFs, are reliant upon accurate estimates of covariance for both measurements and noise. This matrix presents itself as a challenge because there are over forty variables taken into consideration for this research. According to Heffes [29], the need for accurate, consistent, and reliable estimation techniques has been noted in research as far back as the 1960's and is still a common modern research topic. An inaccurate covariance matrix will lead to higher peak variances, slower convergence, and suboptimal estimation performance [14]. Typically, in industry, there are random values selected for covariance models despite the clear downsides mentioned of inaccurate covariance; this is due to the complexity of quantifying both model and measurement uncertainty in real systems. In this research, this presents a challenge as the covariance matrix must essentially be solved through strenuous trial-and-error of different methods. The modern methods that best suited the SFDIA problems were Bayesian, maximum likelihood, and covariance matching [30, 31]. The best method will be determined by the overall performance of the on-line estimator; the better method should produce much more accurate results and perform quickly.

Lastly, a difficult challenge regarding the KF is its foundation of linear algorithms. The KF is arguably the best on-line estimator for linear systems but there are multiple modifications required to adapt the KF to a nonlinear system. This challenge creates room for error in terms of mismatching the realistic linear system and the KF linear system.

2.2 Criteria

The primary criteria by which each on-line estimator was evaluated were the following: estimation error (EE), fault detection time (DT), number of false alarms (FA), number of undetected faults (UF), and detectability ratio (DR). Secondary criteria that were taken into

consideration for this comparative analysis were computational power, overall complexity of design, and time required to train and accommodate the sensor failure.

The estimation error of each on-line estimator dictates the accuracy of the model when determining the proper values for the sensor failure accommodation. Larger estimation errors of the sensor failure could result in further issues with the control system such as providing pilots incorrect data or potential loss of control of the aircraft. The estimation error was calculated for each flight dataset tested in this research. The difference in actual and accommodated data should be minimal for optimal performance.

Fault detection time was considered a criterion for this research since it is of utmost importance in terms of safety that the failed sensor be accommodated immediately. Any extensive delay in accommodation could have serious consequences. If the sensor failure is left undetected, the pilot could lose control of the aircraft. The fault detection time was measured in this work by taking the difference between the initiation time of the SFA scheme and the actual time the sensor failure occurred. Shorter detection times are favorable in this work.

The number of false alarms was also considered a performance criterion for this work as it is crucial that the SFA schemes do not activate when there is no sensor failure present. If the SFA model is too sensitive, it could potentially trigger whenever there is any change in the flight data; this could further deteriorate performance for both the control system and the pilot. It is ideal to have no false alarms from any of the on-line estimators.

The undetected faults were also counted for each set of tests to determine whether the SFA schemes were sensitive enough to detect errors of any severity. An undetected fault is very clearly a sign of poor performance. If a sensor failure is not detected and accommodated

immediately, the severity of the scenario could worsen or become unfixable over time. It is preferred to have no undetected faults.

The last priority criteria taken into consideration for performance comparison was the detectability ratio of each SFA scheme. The DR quantifies the SFA scheme's residual sensitivity to the fault with respect to the residual noise. This value was calculated as the ratio between the maximum residual value before the time of the fault and the residual magnitude at the time of the fault detection. When there is an occurrence of a FA or UF, the DR provides more information as to how the sensitivity of the SFA influenced the incorrect response.

The results based on these criteria will determine the best on-line estimator and provide further analysis of the capabilities of each with regards to robustness, sensitivity, and efficiency.

Chapter #3: Review of Neural Network-Based Approaches

3.1 Basic Principles of Neural Networks

NNs are a branch of artificial intelligence that utilize layers of neurons that signal each other through connections, similar to a human brain, to analyze a current system with prior knowledge. They are used to develop robust algorithms and data structures to model more difficult problems that are not easily defined by equations [33]. NNs are implemented into multiple real-life aspects such as Google Maps and self-driving automobiles [34]. In this research, two different NNs were constructed to recognize a sensor failure by learning patterns from various flight datasets: the multilayer perceptron (MLP) NN trained with back propagation (BP) and the radial basis function (RBF)-based extended minimal resource allocation network (EMRAN).

The architecture of a NN is based on neurons, or nodes, that serve as connection points within the NN architecture. A row of neurons is referred to as a layer and the entire structure of neurons is referred to as the network topology [35]. Each neuron is a computational unit that has one or more input connections, an activation function that combines all inputs, and an output connection.

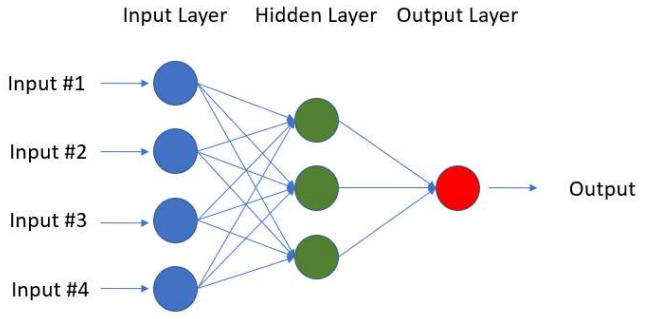

Figure 3.1: Basic model of neural network [36]

Figure 3.1 represents the basic outline of a NN. Each circle is a neuron, and the arrows represent the weighted connections between each neuron. As seen in Figure 3.1, each NN consists of at least three layers: the input layer, the hidden layer, and the output layer. Each layer serves a specific purpose to estimate a future input.

Initially, the input layer of the NN gathers the inputs and transfers the data to the first, or only, hidden layer. For this research, the input layer receives the flight data. The input layer is also called the visible layer as it is one of the exposed portions of the network. Often, a NN is drawn with a visible layer with one neuron per input value or column in the dataset [33, 35].

The layers of the NN are crucial to optimizing its prediction capability. Multiple layers allow the structure to represent certain features at different scales, resolutions, or weights, or combine them into higher-order features. This concept will be observed in this research as the NNs accommodate sensor failure through training with various flight data.

After the input layer is the first set of connections connecting to the hidden layer. The connections for the first iteration are all weighted randomly based on the NN's initial guess of the probability that a certain input is relevant when determining the output. For example, if input #1 has a greater influence on the final output compared to the other inputs, the connections from input #1 will have a greater weight. The weights are defined in every following iteration through a weight function. A bias is also added to the inputs by multiplying each input to their associated weight. The bias of each neuron is considered a 'pseudo-input' that always has the value of 1 and must be weighted [36]. For example, a neuron may have four inputs in which case it requires five weights: one weight for each input and one weight for the bias. Without the bias, the NN is prone to problems associated with an input pattern value of zero. The weight update is controlled by the learning rate (η) as well as the optimizer. The weights of each variable can stay constant

throughout the simulation, or in other cases, like this research, can change over multiple iterations to better estimate future inputs based on prior error.

The hidden layer of the NN performs most of the calculations that determine the final output. These layers are not directly exposed to the user like the input and output layers. Hidden layers allow for the function of a NN to be broken down into specific transformations of the data [37]. Deep learning AI schemes consist of NNs with more hidden layers; however, in the case of this research, there is only one hidden layer per NN. The NNs were limited to one hidden layer based on work by Cybenko [19] and Hornik *et al.* [20] that showed only one hidden layer was required in a NN to map any nonlinear function provided there is enough training data. Fewer hidden layers result in much quicker estimations and results from the NN. Each hidden layer function is specialized to produce a defined output either to the next hidden layer or to the output layer. The weighted sum of that neuron is then transferred to the activation function to produce the next output. The final hidden layer is what estimates the actual output through an application function. In this work, the single hidden layer performs each of these tasks.

Next, the output layer itself will then produce the output for the user. The next iteration will then initiate by adjusting the weights and transferring the new set of inputs from the output layer into the input layer. The weights themselves update based on the training models.

The final general step necessary for any NN is preparing the data for training and testing. The flight data for this research was categorized in a numerical set as to have a configuration understood by the network. For example, the first variable in the dataset for the flight simulation data, velocity, was labeled as '1.' Each column of flight data was assigned a number from 1 to *n,* depending on the size of the data as seen in Table 3.1.

Table 3.1: Number to Variable System used in Flight Simulation Datasets

1	v	10	x_e	19	$\dot{\phi}$	28	$\left(\frac{Pb}{2v}\right)$	37	elevators L
2	α	11	y_e	20	$\dot{\theta}$	29	$\left(\frac{qc}{v}\right)$	38	elevators R
3	β	12	z_e	21	$\dot{\varphi}$	30	$\left(\frac{rb}{v}\right)$	39	ailerons L
4	p	13	$\dot{v}$	22	$\dot{x_e}$	31	A_x	40	ailerons R
5	q	14	$\dot{\alpha}$	23	$\dot{y_e}$	32	A_y	41	rudder L
6	r	15	$\dot{\beta}$	24	$\dot{z_e}$	33	A_z	42	rudder R
7	Φ	16	$\dot{p}$	25	$\dot{u}$	34	a_x	43	flaps
8	θ	17	$\dot{q}$	26	$\dot{v}$	35	a_y	44	canard L
9	φ	18	$\dot{r}$	27	$\dot{w}$	36	a_z	45	canard R

Variables 4 through 45 were selected as the inputs for the NNs to estimate angle of attack, sideslip, and true airspeed.

The same variable numeric system was applied to the output layer. The code for this segment reverted said number of the variable back its name for the user facilitation. Not only did the names of each variable have to convert to a numerical format, the NNs required all inputs to be consistently scaled. For this research, not all flight datasets were the same size, but they had to be processed through the same network.

There were multiple factors taken into consideration when selecting the two NN models to be tested for the SFDIA problem. When considering NNs, the qualities considered were the following: execution time, structure size, overall complexity, and ability to generalize the data

[1]. The MLP NN and EMRAN NN were specifically selected due to their popularity in the research community and previous successes in other research efforts mentioned in Chapter 1.

3.2 Multilayer Perceptron Neural Network

The first NN designed and tested for this research was the MLP NN. The MLP is a completely feedforward supervised learning model that incorporates BP to improve estimation. This specific model is named for its multiple layers of perceptrons which are the building blocks of each layer of the NN [39]. A perceptron is a single neuron model that was a precursor to larger NNs [35]. A perceptron is essentially its own network with two layers: one input layer and one output layer. Each perceptron has a set of inputs, weights, bias, net sum, and activation function with output.

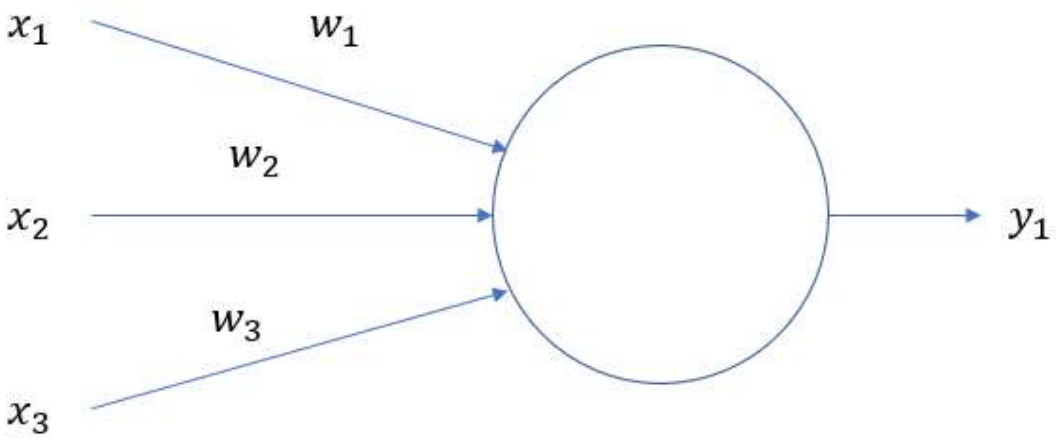

Figure 3.2: Perceptron Model by Minsky-Papert [40]

All perceptrons in the MLP work with the same inputs but will have different weights, bias, and activation functions. The NN uses the different information from the perceptrons to determine the final output.

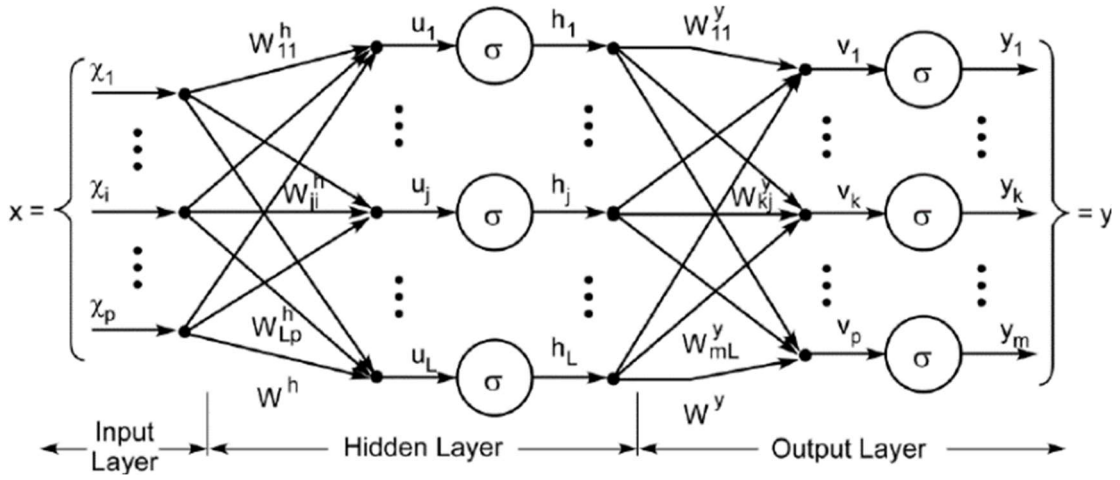

Figure 3.3: MLP NN Feedforward Structure [41]

Figure 3.3 displays the basic outline of the multilayer perceptron NN. As seen within the hidden layer in the figure, there are three perceptrons. Another detail to note from this figure is that all the connections are facing the same direction, resembling a feedforward system. Feedforward networks are typically easier to train than a recurrent network.

To minimize the estimation error of the output sensor, the BP method was implemented into the MLP NN. BP was originally developed in 1984 by James McClelland and David Rumelhart to train nonlinear NNs [42]. The BP method generalized the original perceptron learning algorithm to accommodate NNs with multiple layers. BP is still extensively used today and arguably the most general method of supervised training within multilayered NNs [43]. There have been many notable extensions to the BP algorithm over recent decades including the extended back propagation algorithm (EBPA) [43, 44]. Newer algorithms such as the EBPA improved certain drawbacks of the original BP algorithm such as its tendency to stick to the local minimum. By implementing BP into the NN, the weights of each connection between the input and output are constantly updated internally rather than updated in between intervals.

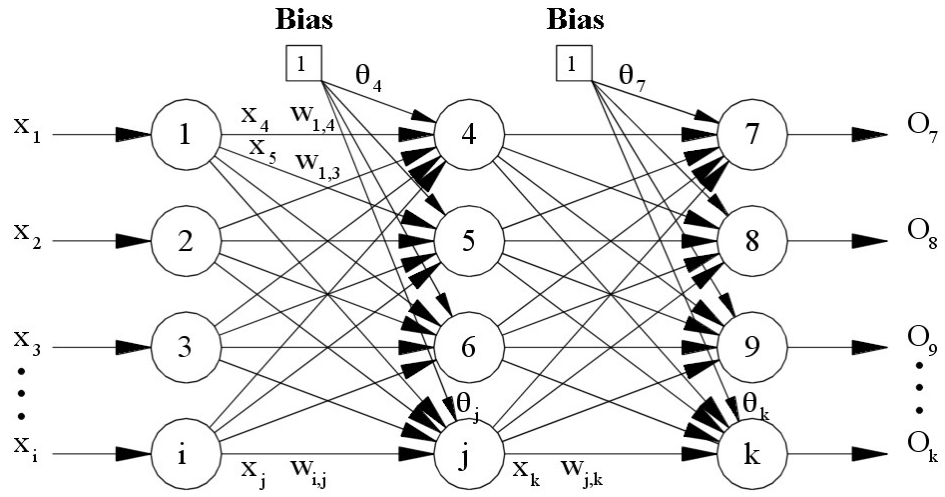

Figure 3.4 Back propagation within MLP NN [43]

In the case of a NN with three layers, like the example in Figure 3.4, the weights are updated internally when the signal reaches the next layer. By adjusting the weight values between layers, BP better approximates the nonlinear relationship between the input and output.

The MLP NNs accomplish the estimation process in two steps: feedforward and BP. In the feedforward step, an input pattern is applied to the input layer. This effect propagates through each layer of the network until an output is produced. This also improves the predictive ability of the NN; inputs not included in the training data can be further generalized with this method. The network's actual output value is then compared to the expected output, and an error signal is computed for each of the output nodes. Since all the hidden nodes have, to some degree, contributed to the errors evident in the output layer, the output error signals are transmitted backwards from the output layer to each node in the hidden layer that immediately contributed to the output layer. This process is then repeated, layer by layer, until each node in the network has received an error signal that describes its relative contribution to the overall error.

Once the error signal for each node has been determined, the errors are then used by the nodes to update the values for each connection weights until the network converges to a state that allows all the training patterns to be encoded. The algorithm evaluates the minimum value of the error function in weight space using a technique called stochastic gradient descent, or delta rule [45]. The error function of the NN is the performance function that calculates the average squared error between the network outputs and the target outputs. This is a crucial implementation required to further verify the estimation outputs of the system. The stochastic gradient descent only considers one row of inputs at a time; as the network processes each input, the neurons are simultaneously activating to eventually produce an output.

Once the error function values are 'approved' by the preset threshold standards of the NN, this solution to the learning problem, or in the case of this research, the sensor accommodation, is set to the weights that minimize the error function. The NN accuracy is tested by comparing the output of the NN to the actual results from the training data.

Possible consequences that may occur due to BP are over-training and under-training. Over-training occurs when a network is too complex; the results of these networks will fall outside the satisfactory range for the output data. This is caused by a NN containing too many nodes [46]. Under-training occurs when the NN is not complex enough to determine an accurate trend within the data. This occurs when there are not enough nodes present. It was desired for this work to develop an MLP with the optimal number of nodes to incorporate BP that can accurately estimate and accommodate the values for a faulty sensor.

3.2.1 Training the Multilayer Perceptron Neural Network

The performance of the MLP NN depends heavily on the training of the network. During the training process, the MLP 'learns' patterns of the angle of attack, sideslip angle, and true airspeed through both flight simulation data and actual flight data. If these patterns are accurate, the MLP can determine when a certain sensor is failing based on the difference between that failed sensor value and the MLP estimated value.

MATLAB and Simulink were used to train the MLP network. The simulated data and real flight data were both processed using these programs to train the MLP to recognize the patterns between each variable. In this process, the MLP also determines the number of hidden layers and neurons required to accurately accommodate each sensor failure type. The basis of this training scheme was inspired by work from Campa [47] seen in Figure 3.5.

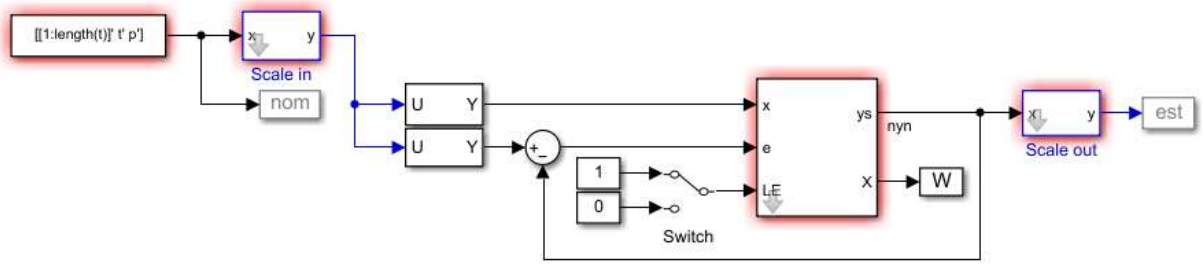

Figure 3.5: NN Training Scheme. Courtesy of Dr. Giampiero Campa.

This design was adjusted for each on-line estimator and sensor failure variable. This is discussed more in depth in Chapter 5.

3.2.2 Back Propagation Algorithm Implemented into the Multilayer Perceptron

The following explanations of the MLP algorithm trained by BP will focus on nodes *i*, *j*, and *k*, where *i* is the input node, *j* is the hidden layer node, and *k* is the output node. These notations were derived from Figure 3.4.

To initiate the estimation process, the first set of weights is calculated as soon as the training pattern is developed by the NN. The weighted sum of the input to the node *j* is given by the following equation

$$W_{net_j} = \sum W_{ij} * x_j + \Theta_j \tag{3.1}$$

where Θ_j is the weighted value from the bias node that is set to have a constant output of 1. Equation 3.1 calculates the aggregate input to the neuron *j*. The bias weights are included to prevent inputs having zero values; the NN cannot be trained without the biases for this reason [48].

The summed weighted inputs are then passed through an activation function. An activation function is a simple mapping of summed weighted input to the output of the neuron. It governs the threshold at which the neuron is activated and strength of the output signal. The activation function essentially determines whether the neuron should fire. The resulting output from the activation function becomes the input of the next layer.

There are several options of activation functions from which to select. Simple step activation functions were used in most past works. With a step activation function, if the summed input was above a preset threshold, for example .50, then the neuron would output a value of 1.00. Otherwise, it would output a 0.00. A nonlinear function allows the network to combine the inputs in more complex ways that better suits the nature of nonlinear flight dynamics equations. Specifically, for NNs with BP, the activation function must also be differentiable. One of the most popular nonlinear activation functions used for BP is the Sigmoid equation [49]. The equation is represented as

$$y_j = x_k = \frac{1}{1 + e^{-W_{net_j}}} \tag{3.2}$$

where *y* represents the output value. The Sigmoid equation sets the output of node *j* to the input of node *k*. This activation function works for BP as it is both nonlinear and differentiable with respect to the exponential function of the net weight. This function, however, is only effective with positive mapping ranging from 0 to 1 which was not favorable for this work. For this reason, Eq. 3.2 was modified using the extended back propagation algorithm proposed by Napolitano *et al.* [44].

$$y_j = x_k = \frac{\Lambda_{upper,j} - \Lambda_{lower,j}}{1 + e^{-W_{net_j}}} + \Lambda_{lower,j} \tag{3.3}$$

The modified Sigmoid function, Eq. 3.3, uses upper and lower bounds (Λ) to expand the original mapping of [0, 1] to [-1, 1] to accurately estimate negative values within the dataset such as sideslip angles. Other functions that can be taken into consideration for future work is the hyperbolic tangent function, inverse tangent function, and rectified linear unit function [50].

The next set of algorithms calculates the estimation errors and weight adjustments for the MLP. The error signal for the output of node *k* is calculated by

$$\delta_k = \Delta_k * O_k(1 - O_k) \tag{3.4}$$

where Δ_k is the difference between the actual output and expected output and O_k is the derivative of the activation function. Based on the stochastic gradient descent process, the change in weight connecting *j* and *k* is proportional to the error at *k* multiplied by the activation of *j* [48]. The formulas used to update the weights between *j* and *k* are the following:

$$w_{j,k} = w_{j,k} + \Delta W_{j,k} \tag{3.5}$$

$$\Delta W_{j,k} = \eta * \delta_k * x_k \tag{3.6}$$

The term $\Delta W_{j,k}$ is the change in weight between *j* and *k*. The learning rate η for this NN is a relatively low constant value that indicates the relative change in weights. The value for the learning rate selected for this research MLP was .1 for initial intervals and decreased to .01 gradually for later intervals. The value .01 was selected as a minimum learning rate as it is commonly used within related research efforts [48]. Values for the learning rate that are too small tend to slow the learning of the NN too much whereas higher values will cause the estimations to oscillate around the NN minimum.

There are many advantages to a changing learning rate such as decreasing the time the NN requires to learn and allows the NN to simultaneously converge to a solution with less chance of overshooting the optimal value. To implement this concept properly into the weight updates, Eq. 3.6 was updated to Eq. 3.7 to incorporate momentum. This allows the weight to change even if the error is decreasing over each interval or epoch.

$$\Delta W_{j,k}{}^{n} = \eta * \delta_k * x_k + \Delta W_{j,k}{}^{(n-1)} * \mu \tag{3.7}$$

By including a momentum term μ to the weight updates, the learning process is accelerated while simultaneously preventing the learning process from settling at a local minimum value. Another method similar to momentum that is commonly seen with BP is Learning Rate Decay. These formulas can also be applied to other layers, such as the hidden layer, if the equations reference the correct nodes, *i*, *j*, or *k*.

The last step of developing the MLP NN with BP is minimizing the error of the outputs though an error function. The following error function is used primarily for BP and was selected for use in this research [51]:

$$E = \frac{1}{2}\sum(\Sigma\Delta_k)^2 \tag{3.8}$$

Once the network topology, weights, and functions are constructed, the MLP is ready for training. After the MLP is trained, it can be tested with new, unseen flight data. The simulation data, YF-22 data, and Tecnam P92 did not contain the same number of variables in their datasets; for this reason, the MLP was designed to adjust its algorithms to accommodate any size of dataset. The MLP was trained independently for each test (simulated, YF-22, and Tecnam P92) then tested for each of the three sensor failures. The implementation of the MLP into the SFDIA scheme for the flight simulator and real flight data is discussed in Chapters 5-7.

3.3 EMRAN Neural Network

The second NN that was constructed and analyzed was the RBF-based EMRAN that is commonly used in aviation safety. It is a NN that develops estimates with no prior knowledge of the system, like the MLP. The general foundation of this NN is the radial basis function. This NN works by considering the distance between a certain point and a defined center. RBF networks consist of two layers; the hidden layer, which combines the features with the radial basis function, and the output layer, to which information is transferred from the hidden layer. The figure below displays a basic RBF NN.

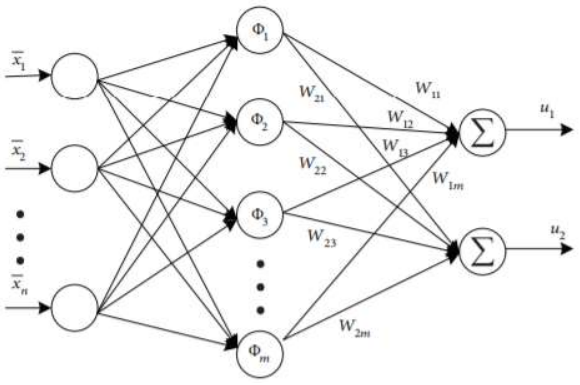

Figure 3.6: RBF NN Basic Structure [51]

As shown in Figure 3.6, the RBF NN is comprised of input signals, a hidden layer, and output layer. It has a similar general structure to the MLP.

Since the late 1980's, there has been considerable interest in RBF NNs due to their global generalization ability and simple network structure [52]. For most research efforts, Gaussian functions are selected as RBFs [8]. Gaussian functions have two parameters that need to be determined: center and width. The NN is trained by a user-selected algorithm to better estimate the states. For this work, the training scheme design was inspired by work from Campa *et al.* [22] to facilitate NN training seen in the earlier Figure 3.5.

When designing the RBF, the main characteristics that need to be taken into consideration are the number of hidden neurons, center and width sizes, and weight estimations. The number of neurons need to be set at a constant number before the training can be initiated. Multiple research efforts use one of the following two methods to determine the number of hidden neurons: stochastic gradient descent [53] and trial and error based on the number of inputs. The gradient descent algorithm works similarly to the MLP algorithm; the NN increases the number of hidden neurons from zero to the required number based on RBF parameter updates [8]. The second approach of trial and error, on the other hand, starts with as many hidden units as the number of inputs then reduces them using a clustering algorithm [9]. Once the number of hidden neurons is selected, the values of the centers and widths of the neurons are calculated based on information and properties of the input data.

Finally, the weights need to be estimated for the connections between the hidden and output layers. There are similar weight update methods that can be implemented into an RBF that are used for MLPs. A common approach for the weight calculations for RBFs is deriving values from a state space model [3, 8, 14].

3.3.1 RBF Algorithm

The following algorithms will be referencing variables from Figure 3.6 which has n_x number of inputs x and n_y number of outputs y. The hidden layer in the middle consists of N number of computing units Φ. These are connected to the output layer through N weight vectors w. The estimations of the output y are calculated by the following equation [8]

$$\hat{y} = w_0 + \sum_{n=1}^{N} w_n * \Phi_n \tag{3.9}$$

where y is a function of the input x. Like the MLP, the initial weight w_0 is given by the bias. The computing unit Φ is calculated by the following Gaussian function:

$$\Phi_n = e^{(-\|x-c_n\|^2/\sigma_n^2)} \tag{3.10}$$

In this equation, c represents the center value of the neuron n and σ is the width of the Gaussian function. The Euclidean norm is represented by the double magnitude value in the numerator. At this point, the RBF function is ready to be trained and tested.

The main issue that prevented RBF NNs being used by itself in this work is they are only suitable for batch type learning schemes and not on-line. For this reason, the MRAN RBF was considered next for development.

3.3.2 Development of MRAN from RBF

In 1991, John Platt proposed a sequential learning algorithm to modify the RBF to be compatible with on-line learning, that is the Resource Allocating Network (RAN) RBF [9]. The improvement to the RBF with this algorithm was how hidden neurons were added. This method used statistical parameters, such as skewness, median, and coefficient of variation, to determine the hidden neuron number in comparison to other parameters in the NN like weighted connections and output neurons. This was known as the LMS method (*lambda, mu, sigma*) [8].

The issue that arose from the RAN NN was an occasional excessive number of neurons being added to the system which led to overfitting of data. To accommodate this issue, a pruning strategy was introduced to remove the neurons that has less influence on the network output. The resulting network was the minimal resource allocation network (MRAN) [8].

3.3.3 MRAN Algorithm

The first critical difference within the MRAN algorithm compared to the standard RBF is that the hidden neuron count is initially set to $n = 0$. At this stage, there is only an input and output layer present in the NN. The network is developed over the course of receiving information or training data in a certain time index. The first step of the MRAN algorithm is setting criteria to add hidden neurons where e is the error, y is the output, x is the input, and E is the user-set threshold [8, 22].

$$\|e_i\| = \|y_i - \hat{y}_i\| > E_1 \tag{3.11}$$

$$e_{irM} = \sqrt{\sum_{j=i-(M-1)}^{i} \frac{e_j^2}{M}} > E_2 \tag{3.12}$$

$$\|x_i - c_{ir}\| > E_3 \tag{3.13}$$

In Eq. 3.11 through 3.13, r represents the center of the hidden unit in question of being added to the system and M is the number of outputs. Eq. 3.11 determines if the error specification made by the user is optimal for the current number of hidden neurons. Eq. 3.12 determines if the NN met the required squared error threshold for the past M outputs. Finally, the last criterion in Eq. 3.13 guarantees the new neurons will not be too similar in magnitude to the remaining existing neurons. If all thresholds E are met, a new hidden neuron is added. The new hidden neuron parameters are determined by the following values:

$$w_{N+1} = e_i\,, \qquad c_{N+1} = x_i\,, \qquad \sigma_{N+1} = \xi\|x_i - c_{ir}\|$$

The overlap factor ξ determines the overlap of responses of hidden neurons from the input space. These parameters remove the error caused by the prior lack of hidden neurons.

The last step of the MRAN algorithm is the implementation of the pruning strategy. In this step, the neurons that contribute less to the output compared to the average neuron are 'pruned;' in other words, the network is prevented from overgrowing in size. Matrix *y* in this case represents the matrix of outputs from the hidden layer and *W* is the weight matrix. These variables indicate the n^{th} neuron at *j* number of outputs [54].

$$y_{nj} = W_{nj} * \exp\left(-\frac{1}{\sigma_n^2}\|x - c_n\|^2\right) \tag{3.14}$$

To reduce any potential inconsistency caused by the absolute value of the output, the following equation is applied to normalize the distance from the center of the neuron to that of the highest output [54]. Here, the new output with respect to the neuron center is represented by Γ.

$$\Gamma_{nj} = \frac{y_{nj}}{max\,\{y_{1j}, y_{2j} \ldots y_{nj}\}} \tag{3.15}$$

Once the output is normalized, it is tested for *N* consecutive inputs. A neuron is pruned if its output *y* falls below a set threshold value for the set of consecutive inputs. At this point, the dimensionality of the matrices is adjusted to fit the new number of hidden neurons.

Although the MRAN could suffice for this research, the EMRAN was preferred for a few reasons. The first reason was that the EMRAN can recognize more accurate behavioral patterns within a more general set of data compared to the MRAN. Another benefit of the EMRAN is that the algorithm adopts the MRAN's pruning strategy which improves overall estimation accuracy. This also prevent the EMRAN from overgrowing [1]. With this property, the EMRAN remains a powerful tool that requires both less execution time and computer power. The following section

will further explain how the EMRAN was designed for this research and modified from the MRAN and RBF designs.

3.3.4 EMRAN Algorithm

The second NN-based on-line estimation algorithm selected for this work is the EMRAN. The EMRAN is preferrable for this work as it is designed for multiple input – multiple output (MIMO) nonlinear time invariant systems. Results from works by Campa *et al.* [22], Napolitano *et al.* [26], and Younis *et al.* [55] show that EMRAN is well suited for real-time implementation of nonlinear system identification, like the SFA problem.

The primary difference of the EMRAN compared to the MRAN is its adaptation of the pruning strategy; the EMRAN only updates the most activated neuron in a "winner takes all" type of strategy [22] whereas the MRAN updates the parameters of all neurons accordingly. This change in algorithm significantly reduces execution time and computation load with only a relatively small decrease in estimation performance estimation [3]. The "winner takes all" strategy reduces estimation error in the poorest performing areas of the network topology.

Overall, the EMRAN is built identically to the MRAN in terms of structure. There is an input layer, output layer, and eventually a hidden layer. Like the MRAN, there are no hidden neurons present in the beginning of the training. As the training data increases, the EMRAN adds hidden neurons as it sees fit by Eq. 3.11 through 3.13 and the pruning strategy.

Building off Eq. 3.9, the EMRAN output was calculated with the following equation [3]:

$$\hat{y}(x,a) = \sum_{i=1}^{M} w_i * \exp\left(\frac{||x - c_i||^2}{2 * \sigma_i^2}\right) \tag{3.16}$$

This output expression is a function of the inputs *x* and parameters *a* set to be updated by the NN. For a new hidden neuron to be added to the system, the same three criteria thresholds *E* from Eq.

3.11 through 3.13 must be met. The parameters of the newly added hidden neuron are the same as the MRAN method. If there are no new neurons created, the following stochastic gradient, or delta rule, function defines the relationship of the new parameters A at a time instant k [3]:

$$A(k+1) = A(k) - \eta \frac{\delta \hat{y}(k)}{\delta A(k)} * e(k) \tag{3.17}$$

In this equation, η is the learning rate and e is the estimation error.

With these algorithms in place within the MATLAB code and Simulink, the EMRAN NN is ready for training with the flight data. Similar to the MLP, since each flight dataset varies in size, the EMRAN was designed to accommodate the sensor failures based on the other information at hand no matter the size of the dataset. The EMRAN was trained for each aircraft type with the available data. The implementation of the EMRAN into the SFA scheme and flight simulator is discussed in Chapter 5.

Chapter #4: Review of Kalman Filter-Based Approaches

4.1 Basic Principles of a Kalman Filter

The KF was first proposed by Rudolf E. Kálmán in 1960 [17]. The foundation of a KF is a set of linear equations which are designed to produce optimal estimates of a vector of parameters by minimizing the mean-squared estimation error at each iteration [14]. The KF algorithm uses a series of measurements and dynamic equations over time, such as noise or inaccuracies, to produce estimates of unknown variables. KFs make estimations based on an implemented mathematical system rather than provided training data like the NNs. In this work, the KFs require a state model of the flight dynamics equations for each sensor and important flight variable.

Kalman filtering is divided into two phases, that is prediction and update, as seen in Figure 4.1. For the prediction phase, the KF produces estimates of the current state variables and their uncertainties. Once the outcome of the next measurement is produced, these estimates are updated using a weighted average of the estimated value and actual value. Like the NN, more weight is given to estimates with greater certainty. The algorithm is recursive in that it can operate in real time using only the present input measurements, the state calculated previously, and its uncertainty matrix.

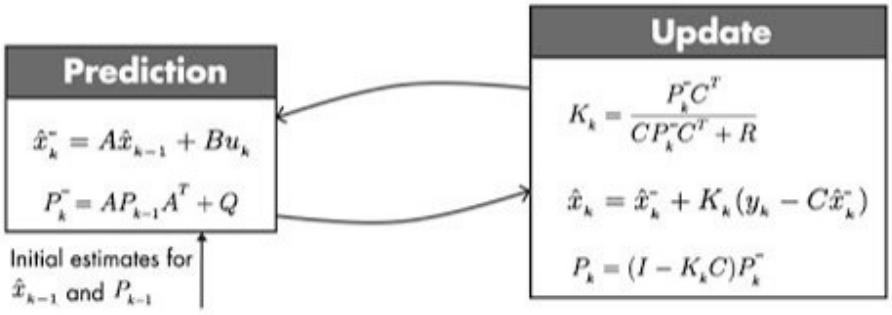

Figure 4.1: Representation of Kalman Filter Process [65]

The KF was groundbreaking at the time of its introduction as it was one of the only forms of control that could estimate unknowns with more than a single measurement. Since then, the KF has been applied to a broad range of problems such as the global positioning systems and controls engineering. KFs have also been widely applied to the fault detection and isolation/identification (FDI) field where the state estimations are directly used to generate a fault residual [56]. KF-based approaches have been applied to research efforts in the past to alleviate the effect of noise on residuals due to their robust nature in terms of residual generation techniques [3, 4, 10, 25]. In this work, the KF calculates the difference between sensor measurements and estimates for any unexpected deviations caused by sensor failures [57].

Various factors that limit the efficacy of the KF's estimation of a given state include noisy sensor data, the given dynamic system of equations implemented by the user, and other external factors. As mentioned, the KF algorithms that were originally designed for linear systems do not adapt to nonlinear systems smoothly. Although noise can diminish the accuracy of the KF, the noise is utilized to help handle uncertainty. Uncertainty is also, to some extent, maintained with random external factors.

Crucial components of the KF are the covariance matrices of the system noise and measurement noise, respectively Q and R. The covariance matrices are measurements of the estimated system state prediction uncertainty in terms of process noise, w, and observation noise, v, respectively. The process noises v and w are assumed to be zero-mean Gaussian with the covariances Q and R. This relationship is defined as the following [17].

$$w_{k-1} \sim \mathcal{N}(0, Q) \qquad v_k \sim \mathcal{N}(0, R)$$

The assigned covariance to a variable defines the joint variability of two random variables. If the greater values of one variable mainly correspond with the greater values of the

other variable, and the same relationship holds for the lesser values, the covariance is positive. Lower covariance values correlate to less noise in the system.

When performing the actual calculations for the KF, the state estimate and covariances are coded into matrices since there are multiple dimensions involved in a single set of calculations. The matrices allow for a representation of linear or nonlinear relationships between different state variables in any of the transition models or covariances. A common example of this type of relationship is position, velocity, and acceleration.

The weighted average calculated is a new state estimate that lies between the predicted and measured state. This new value has a better estimated uncertainty than either predicted or measured values alone. The estimate is updated using a state transition model, *F*, and prior measurements [26]. This process is repeated for every time interval using the new estimate and its associated covariance.

One of the most common methods of assigning weights to measurements is the use of a Kalman gain as seen in works by Kalman [17], Napolitano *et al.* [26], and Stepanov *et al.* [58]. The gain is a weight assigned to both measurements and the current state estimate that can be changed at any iteration depending on the overall performance. Similar to any weight function in a KF, higher gains are associated with more weight whereas lower gains represent smaller weights. When a larger gain is assigned to a value, the KF conforms to the most recent measurements more responsively. Lower gains will cause the KF to conform more closely to the model predictions. Although each gain value has its own advantages, a gain that is set too high or low will have serious consequences on the KF. If the gain value is set too high, the estimations will be more ecstatic or emphasized throughout the trajectory of the state. If the gain is set too

low, such as a value of zero, the noise will be smoothed out but the KF will have minimal to no responsiveness to the system.

The last step of the KF design process is determining the relationship behavior of the errors. To optimize the performance of the system, there is an assumption implemented that errors have a normal, or Gaussian, distribution to manage both uncertainty and noise [17]. Physical random phenomena can be implemented into the KF as random noise that excites dynamic systems. The primary sources of this noise are independent Gaussian random processes with zero mean and the dynamic systems being linear [17]. Regardless of Gaussian distribution, if the process and measurement covariances are known, the KF is notably one of the best possible linear estimators in the minimum mean-square-error sense [59].

There has been a large variety of nonlinear KF designs proposed over the years. This work focuses on two of these extensions, the extended KF and unscented KF, which were both developed to analyze nonlinear systems. The foundation of each method is a hidden Markov model [60]. In these schemes, the state space of the latent variables is continuous, and all variables have Gaussian distributions. Other notable KF designs that were considered for this work were the Dempster-Shafer theory [60], "simple" KF [17], the Kalman–Bucy filter [61, 62], the information filter [60], and a variety of square-root filters [62].

4.2 Methodology of the Kalman Filter

This section discusses the mathematics of the simple KF and how it was modified for the nonlinear flight dynamics equations utilized for the sensor failure accommodations. Both the extended and unscented KFs are designed using this model as a foundation. The following equations are written in terms of the current timestep *k* and previous timestep *k-1*.

4.2.1 Flight Dynamics Equations for α, β, and TAS Used in the State Model

Each variable that will be analyzed (angle of attack, sideslip angle, and true airspeed) has unique input requirements the KF uses to determine whether the sensor has failed. For this reason, three separate KFs were designed for each aircraft, one for each sensor failure, that have the same foundations, but different inputs, systems of equations, and outputs. The on-line estimation code will utilize the inputs and outputs of the specific variable that has failed.

For the angle of attack, the KF will analyze the body components of airspeed velocity (*u, v, w*), rotation rates (*p, q, r*), and the inertial position of the aircraft with respect to the angle of attack ($x_\alpha, y_\alpha, z_\alpha$). Similarly, for the sideslip angle, the KF will analyze the body components of airspeed velocity, rotation rates, and inertial coordinates with respect to the sideslip angle ($x_\beta\ y_\beta\ z_\beta$). Lastly, the true air speed will be estimated with the inputs of the body components for airspeed and the flight angles for roll, pitch, and yaw ($\phi\ \theta\ \psi$). The following formulas were used to determine which variables should be selected as inputs. The KF also uses these equations to measure the residual between the actual measurement and the estimation [63].

$$\alpha = \tan^{-1}\left(\frac{w - qx_\alpha + py_\alpha}{u}\right) \tag{4.1}$$

$$\beta = \sin^{-1}\left(\frac{v + rx_\beta - pz_\beta}{\sqrt{u^2 + v^2 + w^2}}\right) \tag{4.2}$$

$$TAS = \sqrt{u^2 + v^2 + w^2} \tag{4.3}$$

Each KF required the following matrices to be defined: state-transition model F_k, observation model H_k, covariance models Q_k and R_k, and control input model B_k. The KF calculates its first estimate at *k* from the previous state *k-1* using the following equation [3]

$$x_k = F_k * x_{k-1} + B_k * U_k + w_k \tag{4.4}$$

where *U* is the control vector and *w* is the process noise. The process noise for these KFs is assumed to be derived from a zero mean multivariate normal distribution. Next, the observation *z* of the state *x* is calculated with the following equation

$$z_k = H_k * x_k + v_k \tag{4.5}$$

where *v* is the observation noise. The observation noise is assumed to be zero mean Gaussian distribution. Both the initial state and noise vectors in each of the equations are mutually independent.

Eq. 4.4 and 4.5 were modified for the EKF and UKF to accommodate nonlinear data. By adjusting these equations, the performance quality will increase since KFs heavily depend on the modeled dynamics of the system. Unmodeled or inaccurate dynamics like Eq. 4.4 and 4.5 can cause the system in this work to become unstable.

As mentioned earlier, the KF is a recursive estimator; the KF only considers the estimated states from the previous and current timesteps and current measurements. The first phase of the KF process is the prediction phase. In this phase, the state estimate $\hat{x}$ and estimated accuracy of the state estimate *P* are defined by the following relations where *k|k-1* represents "from time *k-1* to time *k*:"

$$\hat{x}_{k|k-1} = F_k * \hat{x}_{k-1|k-1} + B_k * U_k \tag{4.6}$$

$$P_{k|k-1} = F_k * P_{k-1|k-1} * F_k^T + Q_k \tag{4.7}$$

With equations 4.1 through 4.3, the state vectors for each variable were determined as the following.

$$x_\alpha = [u\ v\ w\ p\ q\ r\ x_\alpha\ y_\alpha\ z_\alpha]^T$$

$$x_\beta = \left[u\ v\ w\ p\ q\ r\ x_\beta\ y_\beta\ z_\beta\right]^T$$

$$x_{TAS} = [u\ v\ w\ \phi\ \theta]^T$$

To aid in the prediction of these states, an input vector containing aircraft body acceleration, $a_x\ a_y\ a_z$, and angular rates, p, q, r, is used:

$$U = \left[a_x\ a_y\ a_z\ p\ q\ r\ \right]^T$$

Using the above definitions, the state dynamics can be defined using the three Conservation of Linear Momentum Equations (CLMEs) from *Aircraft Dynamics: From Modeling to Simulation* [63] and kinematic equations.

$$\dot{u} = rv - qw - g\sin\theta + a_x \tag{4.8}$$

$$\dot{v} = pw - ru + g\cos\theta\sin\phi + a_y \tag{4.9}$$

$$\dot{w} = qu - pv + g\cos\theta\cos\phi + a_z \tag{4.10}$$

$$\dot{\phi} = p + q\,sin\,\phi\,tan\,\theta + r\,cos\,\phi\,tan\,\theta \tag{4.11}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{4.12}$$

Additionally, the body component accelerations (a_x, a_y, a_z) were included via the simulated and real flight data directly as it was assumed there was no sensor failure in terms of acceleration.

Following the prediction phase is the update phase. The update is initiated by calculating the pre-fit residual $\tilde{y}$, the difference between the KF estimated value and true value.

$$\tilde{y}_k = z_k - H_k * \hat{x}_{k|k-1} \tag{4.13}$$

The measurements calculated by the KFs are angle of attack, sideslip angle, true airspeed, pitch, and roll.

$$\tilde{y} = [\alpha\ \beta\ TAS\ \phi\ \theta]^T$$

Next, the pre-fit residual covariance *S* is calculated in terms of the observation model, state estimate accuracy, and observation noise covariance.

$$S_k = H_k * P_{k|k-1} * H_k^T + R_k \tag{4.14}$$

The optimal Kalman gain *K* for the timestep *k* is calculated using the following relation from Walrand *et al.* [64]:

$$K_k = P_{k|k-1} * H_k^T * S_k^{-1} \tag{4.15}$$

At this point, now that the Kalman gain is defined, the final state estimate for the timestep *k* can be updated with the following equation

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k * \tilde{y}_k \tag{4.16}$$

which was derived from [64]

$$\hat{x}_{k|k} = (I - K_k * H_k) * \hat{x}_{k|k-1} + K_k * (H_k * z_k + v_k) \tag{4.17}$$

Similarly, the covariance of the state estimate is updated with the following equation in terms of the Kalman gain:

$$P_{k|k} = (I - K_k * H_k) * P_{k|k-1} \tag{4.18}$$

Finally, the process for timestep *k* concludes with the calculation of the post-fit residual. This is solved in terms of the current state estimate to finalize the updated value.

$$\tilde{y}_{k|k} = z_k - H_k * \hat{x}_{k|k} \tag{4.19}$$

The final output was the same structure for each KF. No matter which sensor failure was accommodated, the KF output all three variables of interest and the flight angle rates for roll, and pitch.

$$y = [\alpha\ \beta\ TAS\ \phi\ \theta]^T$$

4.2.2 Estimation of the Noise Covariances Q_k and R_k

The last step in the overall design is the noise covariance matrices Q and R updates. The There are multiple proposals that studied which method is best in terms of estimating covariance matrices but there is no single way that works for every problem type. In this research, after trial and error of various Q and R matrices, the method selected was the autocovariance least-squares method used in works by Julier *et al.* [25], Napolitano *et al.* [26], Duník *et al.* [30], and Odelson et al. [31]. The values calculated by this method provided the most accurate results when tested with simulated flight data. The final Q and R matrices for both the simulated and real flight data are shown in Chapters 5-7.

4.3 Extended Kalman Filter

The EKF is one of the most popular approaches for nonlinear applications used in SFA schemes and, more generally, sensor fusion schemes like navigation systems and GPS [3, 26, 66]. The EKF was the first nonlinear KF designed shortly after the introduction of the original KF. The EKF adapted techniques from multivariate Taylor series to linearize a model about a working point [67].

Similar to the linear KF, the EKF initiates by calculating the initial prediction of the state estimate and observation estimate. Contrary to the KF, both estimates need to be defined by nonlinear relationships derived from the flight equations. The general forms of these estimations are written as follows [17]:

$$x_k = f(x_{k-1}, U_k) + w_k \tag{4.20}$$

$$z_k = h(x_k) + v_k \tag{4.21}$$

In these equations, *f* and *h* represent the nonlinear, differentiable functions that define the estimated predictions of the states and observation in terms of all other variables *x* and the control input *U*.

$$x_{k+1} = \begin{bmatrix} \tan^{-1}\left(\frac{w - qx_\alpha + py_\alpha}{u}\right) \\ \sin^{-1}\left(\frac{v + rx_\beta - pz_\beta}{\sqrt{u^2 + v^2 + w^2}}\right) \\ \sqrt{u^2 + v^2 + w^2} \\ q\cos\phi - r\sin\phi \\ p + q\,sin\,\phi\,tan\,\theta + r\,cos\,\phi\,tan\,\theta \end{bmatrix} + w_k \tag{4.22}$$

$$z_k = x_k * U_k + v_k \tag{4.23}$$

In contrast to the linear KF, these equations for states and observation cannot be applied directly to the covariance; the solution to this issue is computing a Jacobian matrix of partial derivatives [68]. Due to the KF's linear nature, the Jacobian is required to linearize the nonlinear function around the current estimate. At each time step, the Jacobian is computed in MATLAB with the current predicted states which are implemented into the prediction and update phases of the EKF.

The EKF design required the following matrices mentioned in the original KF: *F, H, Q,* and *R*. The state transition and observation matrices *F* and *H* were derived through Jacobian partial derivatives. Each equation is defined as [17, 68]

$$F_k = \frac{\partial f}{\partial x}|_{\hat{x}_{k-1|k-1}, U_k} \tag{4.24}$$

$$H_k = \frac{\partial h}{\partial x}|_{\hat{x}_{k-1|k-1}} \tag{4.25}$$

The remaining process for the prediction and update phases follows Eq. 4.13 through 4.19 from the simple KF.

4.4 Unscented Kalman Filter

The second KF designed in this work is the UKF proposed by Julier and Uhlman [25]. The UKF was originally designed to improve the estimation shortcomings of the EKF. Most linearization techniques used in the EKF typically yield the incorrect evaluation of the mean or covariance. The UKF introduced a new approach to this issue and the overall design of the nonlinear KF.

In the EKF, the state distribution is approximated by the Gaussian random variable which is then propagated analytically through the first-order linearization of the nonlinear system. As mentioned, this can introduce large errors which may lead to sub-optimal performance and sometimes divergence of the filter. The UKF solution to this problem is implementing a deterministic sampling approach. The state distribution is approximated by the Gaussian random variable, similar to the EKF, but is now represented using a minimal set of sigma points. These sample points accurately determine the true mean and covariance of the Gaussian random variable. When these values are propagated through the actual nonlinear system, it captures the posterior mean and covariance accurately to the 3^{rd} order using Taylor series expansion [25]. The EKF on the other hand can only approximate first-order accuracy. The UKF is a notable innovation within KFs as it is not only more powerful than the EKF itself but is no more complex [10]. This is achievable by unscented transform (UT).

The UT method calculates the statistics of a random variable that is processed through nonlinear transformation. This process develops sigma vectors and their corresponding weights. By implementing the UT sigma point propagation, both the Jacobian and Hessian are evaluated without having to perform any analytic differentiation. The following algorithm originally

designed by Wan *et al.* was adjusted and implemented into the SFA flight data scheme to designate the values within the sigma vectors for the UKF in this work [10].

The UKF was selected for this analysis due to ease of implementation and success in past works by Wan *et al.* [10], Napolitano *et al.* [23], and Fravolini *et al.* [27]. Similar to the EKF, there are straightforward calculations implemented into the KF to accurately estimate the unknown variable. To initiate the UKF process, the state space $\dot{x}$ is created as a matrix, or function, of the state variables and control inputs.

$$\dot{x} = f(x, U) \tag{4.26}$$

$$y = G(x, U) \tag{4.27}$$

After the initial state space is created, the matrix is discretized through the following algorithm.

$$x_k = x_{k-1} T_S f(x_{k-1}, U_k) = f(x_{k-1}, U_k) \tag{4.28}$$

$$x_k = f(x_{k-1}, U_k + w_k) \tag{4.29}$$

In the UKF, an augmented version of the matrix is implemented into the KF algorithm with the state values and weight values.

$$x_{aug} = [x^T \quad w^T]$$

Due to the state augmentation, the state covariance matrix is augmented accordingly to include the noise characteristics of the process noise states, as in

$$P_{k_{aug}} = \begin{bmatrix} P_k & 0 \\ 0 & Q_k \end{bmatrix}$$

The UKF uses the unscented transformation to obtain the estimates of mean and covariance. The first step of the UKF implementation is to calculate $2l$+1 sigma points based on

the square-root of the augmented state covariance matrix where the total dimension of the augmented state vector is denoted by l [3].

$$X_{k-1_{aug}} = \begin{bmatrix} X_{k-1}^{x} \\ X_{k-1}^{w} \end{bmatrix}$$

$$= \left[\hat{X}_{k-1|k-1_{aug}} \quad \hat{X}_{k-1|k-1_{aug}} + \eta \sqrt{P_{k-1|k-1_{aug}}} \quad \hat{X}_{k-1|k-1_{aug}} \right.$$

$$\left. + \eta \sqrt{P_{k-1|k-1_{aug}}} \right]$$

The χ is a matrix of sigma points, and η is the sigma point spread parameter, given by

$$\eta = \sqrt{1 + \lambda} \tag{4.30}$$

$$\lambda = l(\alpha^2 - 1) \tag{4.31}$$

where λ is the compound sigma point parameter, and α is the primary sigma point scaling parameter, which is suggested to vary between 0.001 and 1 [10, 16].

The following algorithm implemented for the remaining processes within the UKF was derived from Gururajan *et al*. [3], Wan *et al.* [10], and Rhudy *et al.* [24]. Like the EKF and original KF, the overall process of the UKF can be divided into the prediction and update phases.

As mentioned, the primary modification to the UKF compared to other KFs is the implementation of the sigma points χ. The equations from the original KF (Eq. 4.13 through 4.19) were adjusted to accommodate the sigma points. The state prediction equation in terms of the sigma points initiates the prediction process:

$${}^{i}X_{k|k-1}^{x} = f({}^{i}\chi_{k-1}^{x}, U_k + {}^{i}\chi_{k-1}^{w}) \tag{4.32}$$

Eq. 4.30 derives a function for the sigma points in terms of the control input, initial sigma states, and the weighted initial sigma states. This equation is used as a foundation to derive the actual state and prediction estimates which are calculated in the following equations.

$$\hat{x}_{k|k-1} = \sum_{i=0}^{2l} w_i^m \ {}^i\chi_{k|k-1}^x \tag{4.33}$$

$$P_{k|k-1} \sum_{i=0}^{2l} w_i^c \left({}^i\chi_{k|k-1}^x - \hat{x}_{k|k-1}\right)\left({}^i\chi_{k|k-1}^x - \hat{x}_{k|k-1}\right)^T \tag{4.34}$$

Both Eq. 4.31 and 4.32 are based on the original KF algorithm. In this case, the difference is taken from the sigma matrix and state estimates to calculate a more accurate prediction. The next step is the development of the observation function Ψ.

$${}^i\Psi_{k|k-1} = G\left({}^i\chi_{k|k-1}^x, U_k\right) \tag{4.35}$$

After the sigma matrix is calculated, the remainder of prediction phase initiates. The estimated output *y* is now defined in terms of both weight and the sigma matrix. The output covariance and cross-covariance matrices *P* are also calculated with the estimated sigma matrix and *y*.

$$\hat{y}_{k|k-1} = \sum_{i=0}^{2L} (w_i^m * \ {}^i\Psi_{k|k-1}) \tag{4.36}$$

$$P_k^{yy} = R_k + \sum_{i=0}^{2L} w_i^c \left({}^i\Psi_{k|k-1} - \hat{y}_{k|k-1}\right)\left({}^i\Psi_{k|k-1} - \hat{y}_{k|k-1}\right)^T \tag{4.37}$$

$$P_k^{xy} = \sum_{i=0}^{2L} w_i^c \left({}^i\chi_{k|k-1} - \hat{x}_{k|k-1}\right)\left({}^i\Psi_{k|k-1} - \hat{y}_{k|k-1}\right)^T \tag{4.38}$$

With the covariance matrices *P*, the Kalman gain is calculated with the following equation.

$$K_k = P_k^{xy}\left(P_k^{yy}\right)^{-1} \tag{4.39}$$

After the Kalman gain is calculated, the update phase can initiate. Once the new estimated values for the next input states *x* and the new *P* are derived, the next time interval of the KF can start. This process repeats until the end of the time series.

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k\left(y_k - \hat{y}_{k|k-1}\right) \tag{4.40}$$

$$P_k = P_{k|k-1} - K_k P_k^{yy} K_k^T \tag{4.41}$$

To initiate the UT, a matrix of the sigma vectors χ_0 is constructed and set equal to the mean $\bar{x}$ of the inputs *x*. The sigma vectors are adjusted and updated in accordance with a set scaling parameter λ. This scaling factor is calculated in terms of the dimension *L*, and secondary and tertiary scaling factors *a* and *κ*.

$$\lambda = a^2 * (L + \kappa) - L \tag{4.42}$$

The value of *a* is set to a very small value; this determines the spread of the sigma points around the mean. The secondary value *κ* is set to zero in this research as the UKF did not require further scaling in the SFA scheme. The sigma matrix is updated for each time step *i* where

$$\chi_i = (\bar{x} + (\sqrt{(L + \lambda) * P_x})_i \qquad i = 1, \dots, L \tag{4.43}$$

or

$$\chi_i = (\bar{x} - (\sqrt{(L + \lambda) * P_x})_{i-L} \qquad i = 1, \dots, 2L \tag{4.44}$$

The weights of each sigma vector are calculated with the following equations:

$$W_{0_m} = \frac{\lambda}{L + \lambda} \tag{4.45}$$

$$W_{0_c} = \frac{\lambda}{L+\lambda} + (1 - a^2 + b) \tag{4.46}$$

$$W_{i_m} = W_{i_c} = \frac{\lambda}{2 * (L+\lambda)} \tag{4.47}$$

The subscripts *c* and *m* represent the covariance and mean, respectively. The weights are evaluated in terms of the dimensions *L*, scaling parameter *a,* and scaling parameter *b.* The value *b* represents the system's prior knowledge of the distribution of the inputs *x.* For most Gaussian distributions, the optimal number for *b* is 2, which was also used for this work.

Once the weight matrices are developed based on the sigma values, the sigma vectors are input into the nonlinear flight dynamics equations for angle of attack, sideslip, and true airspeed to produce the final output *y.* The mean and covariance of the sigma values are solved by the following equations:

$$\bar{y} = \sum_{i=0}^{2L} W_{i_m} * y_i \tag{4.48}$$

$$P_y = \sum_{i=0}^{2L} W_{i_c} * (y_i - \bar{y}) * (y_i - \bar{y})^T \tag{4.49}$$

These equations have shown to be accurate for approximations to the third order for Gaussian inputs for all nonlinearities in past works mentioned by Wan *et al*. [10] and Napolitano *et al*. [23]. Adjustments are made through the values of *a* and *b* for non-Gaussian values [69]. The UKF in this process has redefined the Gaussian random variable to accommodate for higher order nonlinear functions.

After designating each initial matrix, the UKF was ready for testing with actual flight data. In the case of the KFs compared to the NNs, there was no adjusting needed when testing simulated or real flight data; the KFs were designed to use variables that were provided by every

flight dataset. The system of equations used by the KFs to calculate estimations and accommodate the failed sensor remained the same for the simulated, YF-22, and Tecnam P92 data. Unlike the NNs, there was no training required for the KF. Depending on the variable of the sensor failure, however, the input matrix x changes based on the matrices earlier in the chapter.

Chapter #5: Performance Analysis Using Simulation Data

This project incorporated flight data generated by WVU's flight simulator to train and test the NNs and KFs. The simulated sensor failure data for angle of attack, sideslip angle, and true airspeed were used as the first evaluation in this work. The simulator consists of two joysticks for the human pilot input and monitors for simulator visualization.

Figure 5.1: West Virginia University High-Performance Military Aircraft Flight Simulator via FlightGear [71]

Within this simulation, the user or pilot was able to control the aircraft with the provided joysticks and observe the behavior of the aircraft in real time through the FlightGear program pictured in Figure 5.1. The simulation was designed in Simulink by recreating the flight system of a high-performance military aircraft through sets of subsystems seen in Figure 5.2. This Simulink scheme combined with MATLAB served as the source of the training and testing data for the NNs and KFs in this chapter.

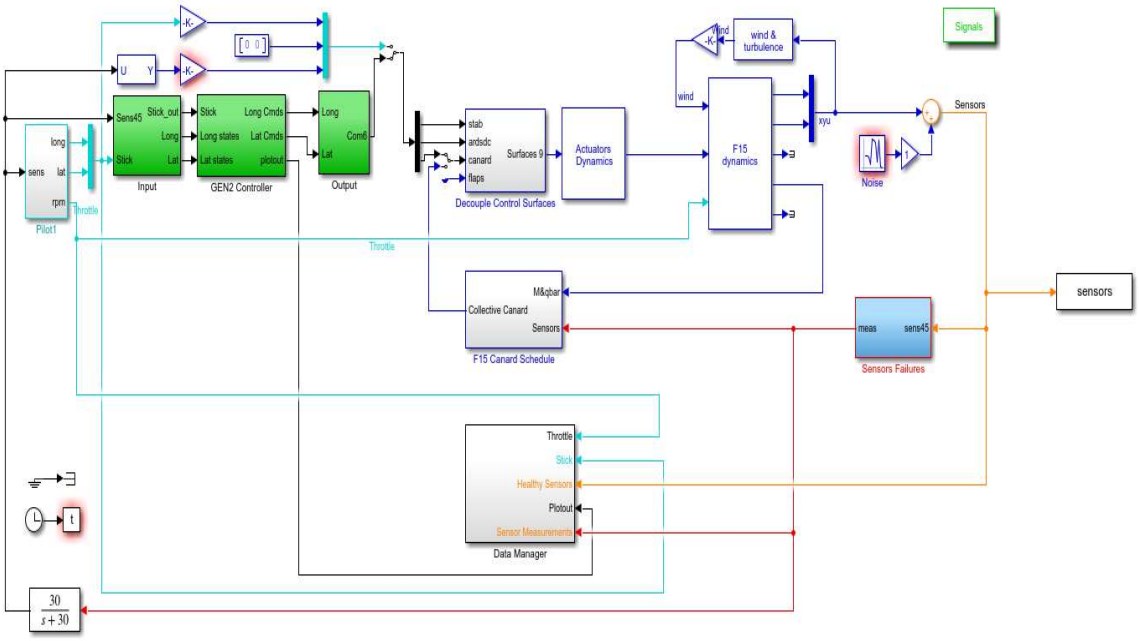

Figure 5.2: Simulation Aircraft Dynamics Simulation Block Diagram (Simulink)

Figure 5.2 displays the original Simulink scheme used to generate fault-free training data for the neural networks. This scheme also served as the foundation for the design of the Simulink model that produced the testing data, or faulty data. Each simulated flight was limited to 1500 seconds with a sampling rate of 0.02 seconds to facilitate time-series modeling. For each flight, the human pilot would perform random maneuvers at cruise altitude in the given time frame to provide a larger variety of data for the NN training. The training data produced for this work consisted of twenty simulated fault-free flights, all with random, unique maneuvers. Another ten simulated fault-free flights were generated to be used as the testing data. The sensor failure was implemented into these ten flights through the sensor failure mechanisms discussed later in this section.

After each flight, the data from the simulator was extracted through Simulink and MATLAB. These training and testing datasets were saved as .mat files so they could be accessed

at any point during the research process. The NN was trained with the twenty training datasets through MATLAB and Simulink.

The training datasets provided by the simulation were used by the NNs to determine the pattern of α, β, and TAS with respect to the other variables. The NNs applied this "knowledge" to the testing data to detect the sensor failure and used the acquired patterns to accommodate that failed sensor with accurate estimations of that variable. The KFs on the other hand continuously calculated α, β, and TAS over time with the implemented system of equations (4.1 through 4.3 and 4.8 through 4.12) using the test data values and compare its estimation to the actual value to determine the residual, or whether a sensor has failed. The KFs used the covariance matrices from Chapter 4 to accurately estimate each of the three sensor failures. The angle of attack sensor, sideslip angle sensor, and true airspeed sensor were accommodated by the KFs with their respective derived Q and R covariance matrices from Chapter 4.

The simulation provided the data of 45 sensors in real time that were used by the NNs and KFs to calculate residuals and estimations for the failed sensor. Table 5.1 displays the variables measured in the simulation.

Table 5.1: Variables Measured in Simulated Aircraft

v	(m/s)	x_e	(m)	f_i	(rad/s)	$\left(\frac{Pb}{2v}\right)$	(rad)	***elevators***	L(rad)
α	(rad)	y_e	(m)	$\dot{\theta}$	(rad/s)	$\left(\frac{qc}{v}\right)$	(rad)	***elevators***	R(rad)
β	(rad)	z_e	(m)	$\dot{\phi}$	(rad/s)	$\left(\frac{rb}{v}\right)$	(rad)	***ailerons***	L(rad)
p	(rad/s)	$\dot{v}$	(m/s^2)	$\dot{x_e}$	(m/s)	A_x	(g)	***ailerons***	R(rad)
q	(rad/s)	$\dot{\alpha}$	(rad/s)	$\dot{y_e}$	(m/s)	A_y	(g)	***rudder***	L(rad)
r	(rad/s)	$\dot{\beta}$	(rad/s)	$\dot{z_e}$	(m/s)	A_z	(g)	***rudder***	R(rad)
Φ	(rad)	$\dot{p}$	(rad/s^2)	$\dot{u}$	(m/s^2)	a_x	(g)	***flaps***	(rad)
θ	(rad)	$\dot{q}$	(rad/s^2)	$\dot{v}$	(m/s^2)	a_y	(g)	***canard***	L(rad)
φ	(rad)	$\dot{r}$	(rad/s^2)	$\dot{w}$	(m/s^2)	a_z	(g)	***canard***	R(rad)

5.1 Developing and Distinguishing Training and Testing Data

Both training and testing datasets were developed in the simulation for the first evaluation of the NNs and KFs. Each dataset developed within the flight simulator was labeled *SIMT* or *SIM* with a respective number. Flight datasets labeled *SIMT* are datasets created specifically for training. Flight datasets labeled *SIM* were generated solely for testing.

This chapter will discuss the results for four flight simulations used for testing in this work, respectively SIM 001, SIM 002, SIM 003, and SIM 004. These four simulations were selected to be displayed out of the ten potential testing datasets as they are completely different from each other in terms of flight maneuvers. The results from the four selected testing

simulations provide sufficient performance results of each of the on-line estimators. The flight maneuvers of each SIM selected are shown below, specifically the roll, pitch, and yaw data.

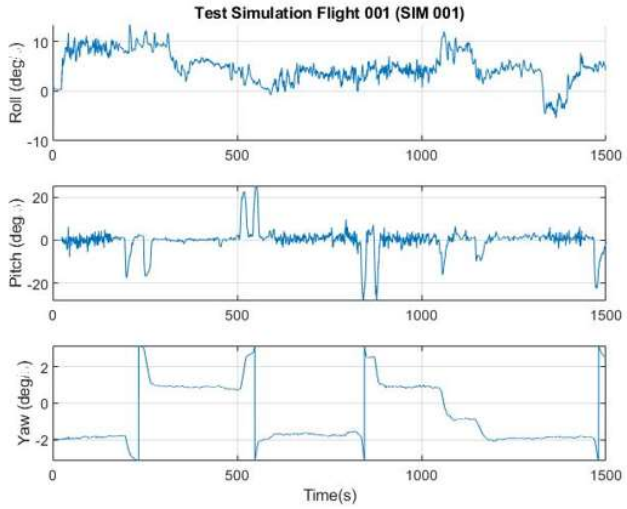

Figure 5.3: Testing Simulation 001 Flight Maneuver Data

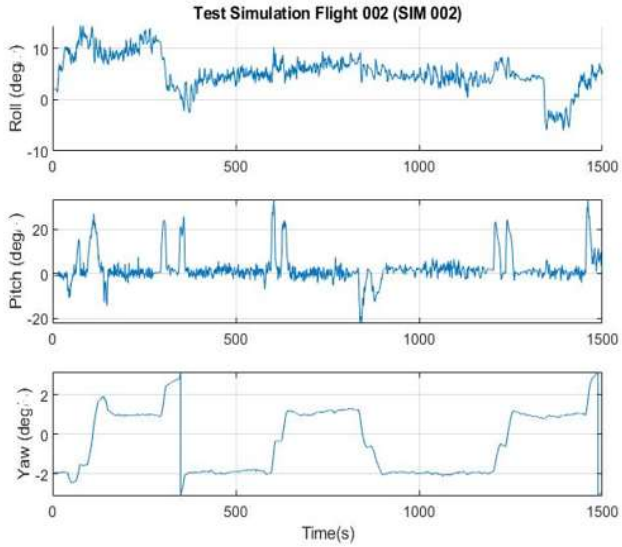

Figure 5.4 Testing Simulation 002 Flight Maneuver Data

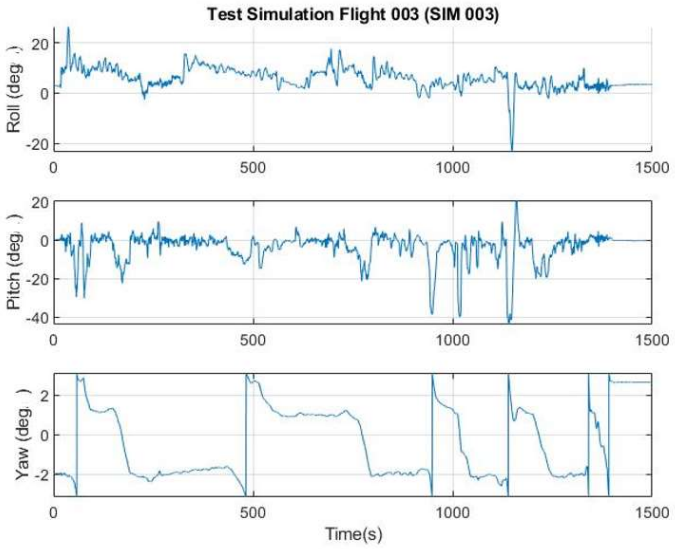

Figure 5.5: Testing Simulation 003 Flight Maneuver Data

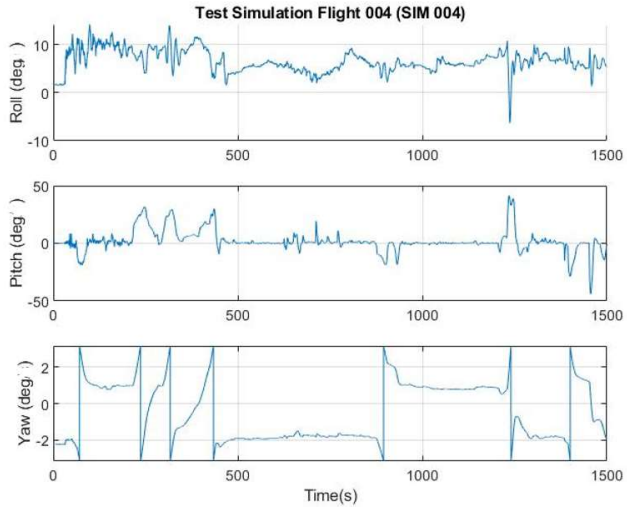

Figure 5.6: Testing Simulation 004 Flight Maneuver Data

This work also displays the training results of the NNs using four selected training datasets, SIMT 001, SIMT 002, SIMT 003, and SIMT 004. These training datasets provided the best results in training for both the MLP and EMRAN when tested against the four SIM flights. The flight maneuvers of each training flight (SIMT 001 through SIMT 004) selected are shown below, specifically the the roll, pitch, and yaw data.

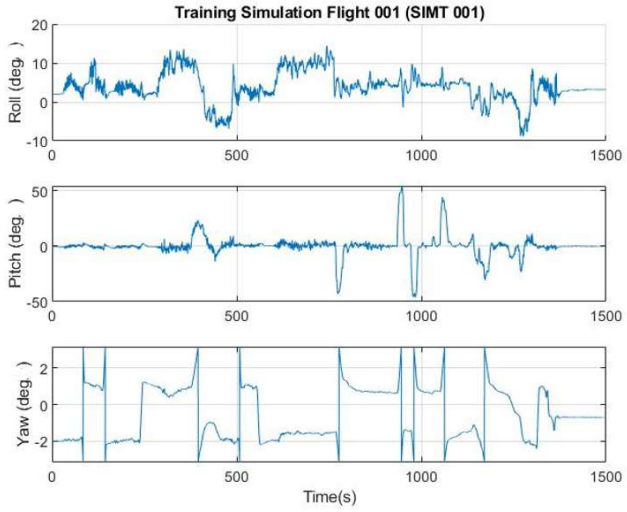

Figure 5.7: Training Simulation 001 Flight Maneuver Data

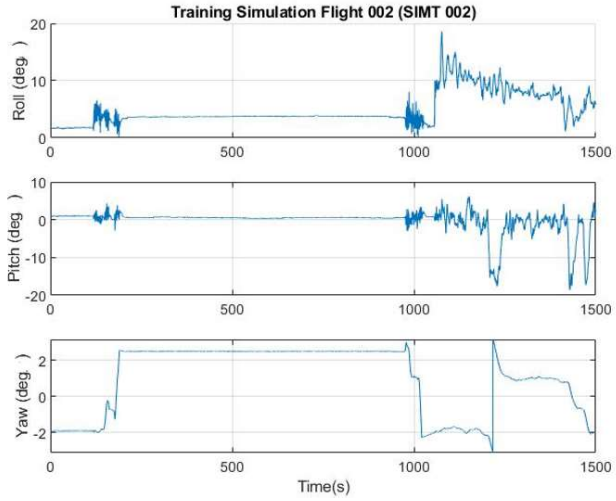

Figure 5.8: Training Simulation 002 Flight Maneuver Data

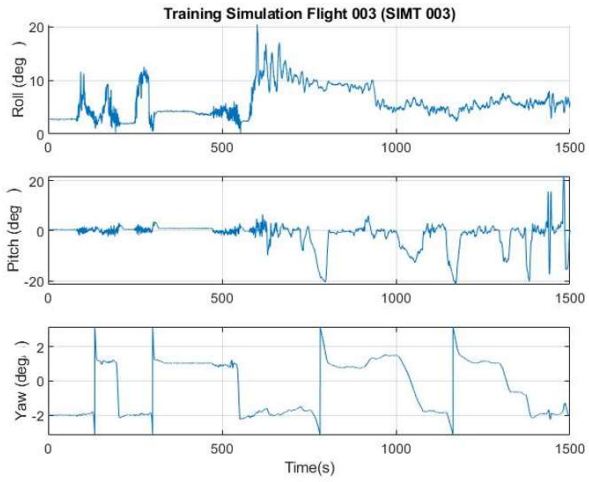

Figure 5.9: Training Simulation 003 Flight Maneuver Data

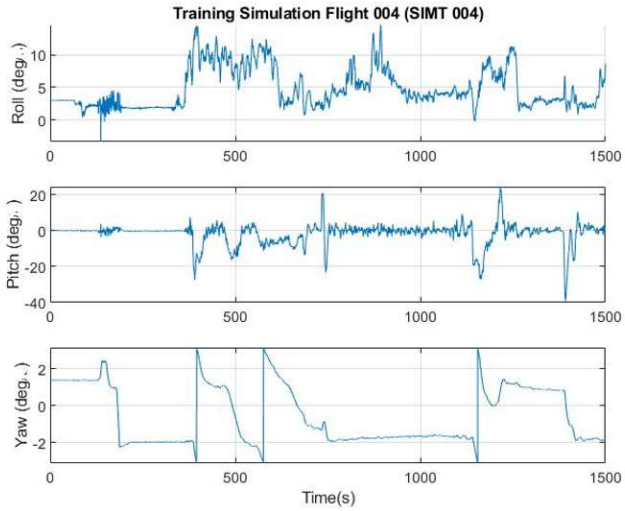

Figure 5.10: Training Simulation 004 Flight Maneuver Data

5.2 Implementing the Sensor Failure into Testing Flight Data

The ten simulated flight datasets developed for testing in the original simulation required additional processing through a sensor failure-inducing scheme to implement the sensor failure of angle of attack, sideslip angle, or true airspeed. The subsystem Sensor Failures from the original Simulink model as seen below was implemented into the second Simulink model for this purpose.

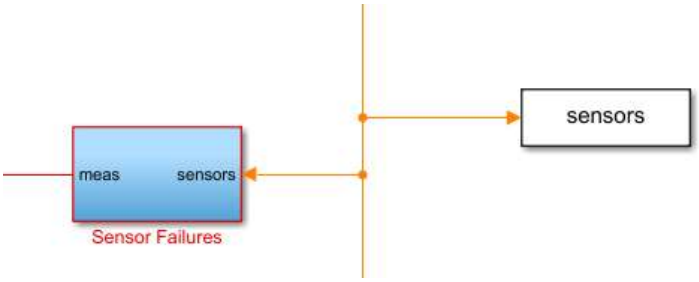

Figure 5.11: Original Sensor Failure Block without SFA

Various changes were made to this design not only to simulate multiple types of failures but also to incorporate an SFA scheme to test the NNs and KFs against the testing data.

The Sensor Failures block skews the true measurements from the testing simulation datasets to create faulty data for a specific sensor selected by the user. The testing flight datasets developed in the original scheme in Figure 5.1 were processed a second time through this sensor failure block to implement the sensor failure that the on-line estimators needed to accommodate. The user defined the sensor failure and severity of the failure for each test flight through this subblock.

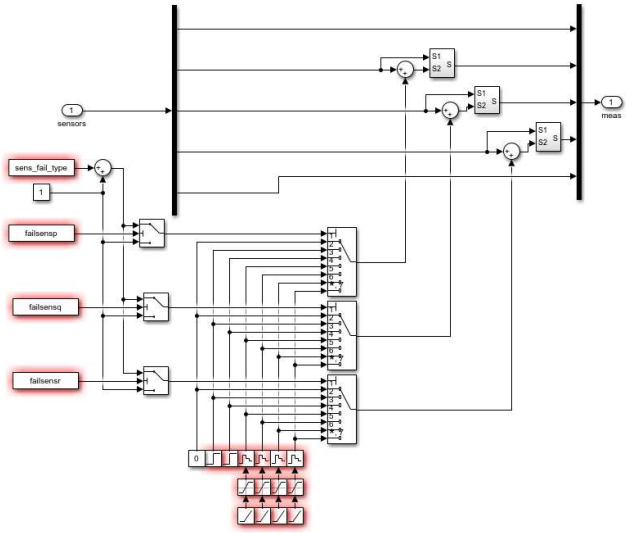

Figure 5.12: Inner Mechanics of Sensor Failure Block

Figure 5.12 displays the mechanics of the sensor failure subsystem. As seen in the figure, the sensor that will fail in the simulation can be designated by the user prior to reprocessing the testing data. This allowed the user to implement a sensor failure at a specific user-indicated time. For this specific simulation, the port number corresponding to each of the relevant variables (*α, β, TAS*) were 2, 3, and 1, respectively. The figure below shows the system channeling port 1, so the sensor that was designed to fail in this specific case was the true airspeed.

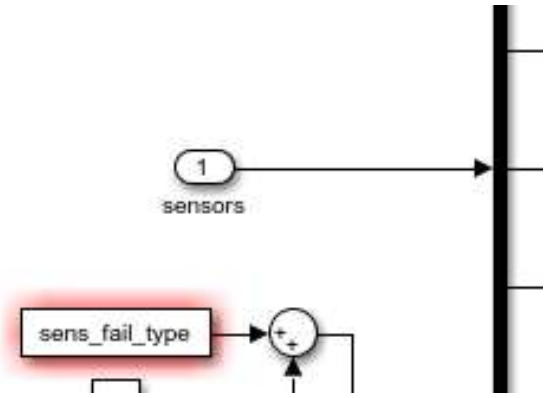

Figure 5.13: Port Number of Sensor Selected to Fail in Simulation

Since the true airspeed is the sensor selected to fail in this case, the channel "fail_sens_TAS" is input into the sensor failure block scheme seen in Figure 5.4. As seen in Figure 5.13 toward the left, the sensor failure type (small bias or large bias) is selected and input through MATLAB. The severity of the failure is determined by the switch mechanisms and fault inputs; essentially, the user or code can dictate which inputs or failure factors are considered when the testing data is processed.

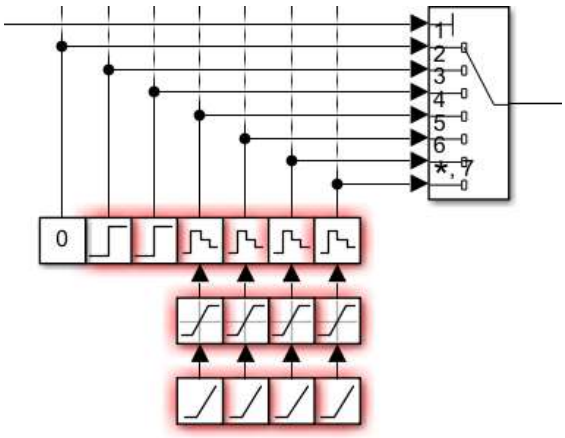

Figure 5.14: Zero-Order Holds, Saturations, Sensor Biases as Faulty Inputs for Switches

Figure 5.14 shows the variety of input selections used to determine fault severity in the simulation. The inputs are constructed as series of holds, saturations, and sensor biases to influence the type of sensor failure. The system defaults to the first input switch option of 0, or fault-free. These mechanisms were applied to each sensor failure type in this research. The levels of severity were divided into two categories: large bias and small bias errors. The magnitudes of each error, no matter the type of error, were varied within the flight datasets to determine the robustness of each on-line estimator. Large errors were greater than 5% difference in value while small errors were lower than 5%. Tables at the end of Chapters 5-7 display the results of each of the two error categories.

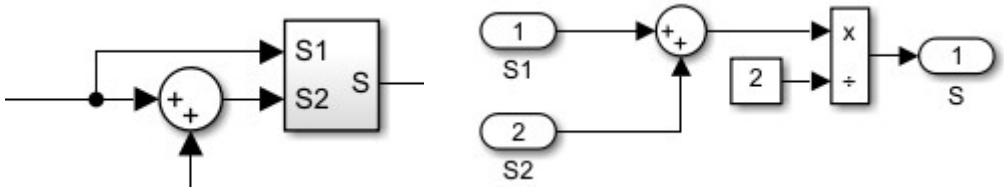

Figure 5.17: Voting Scheme and Its Inner Mechanics

Lastly, voting schemes were implemented into this subblock to simulate a more realistic output of a faulty sensor. For each input (angle of attack, sideslip, or true airspeed), the newly calculated faulty data was averaged with the original data to generate the final faulted output.

Each SIM dataset was processed through this sensor failure scheme to implement small and large sensor failures for angle of attack, sideslip angle, and true airspeed. All SIM flights had a failure of each type and severity implemented. For example, the first testing dataset, SIM 001, was processed through this simulation six separate times to generate six different failure scenarios for testing: large angle of attack error, small angle of attack error, large sideslip angle error, small sideslip angle error, large true airspeed error, and small true airspeed error. This allowed the user to test the same flight dataset for multiple scenarios. This method was also used in Chapters 6 and 7 with the YF-22 and Tecnam P92 data.

5.3 Designing and Implementing SFA Schemes

The following SFA schemes were initially designed for the flight simulator and later used for the YF-22 and Tecnam P92 testing datasets. As mentioned, the SFA scheme will accommodate sensor failure data when the residual calculated surpasses the input limits. To successfully input this mechanism within the simulator scheme, the SFA and on-line estimator was placed in a position to intake the failed flight data. The overall concept of the SFA for this work was inspired by work originally done by Napolitano *et al.* [70] shown in the following image.

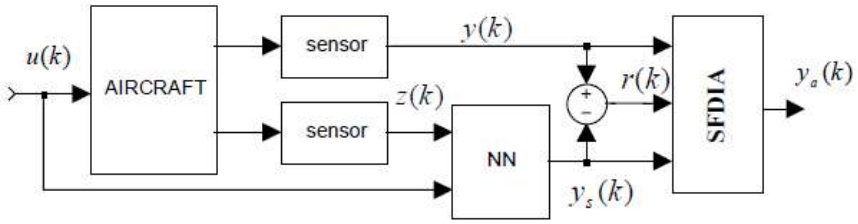

Figure 5.16 General SFDIA Scheme [70]

In this work, rather than implementing the scheme directly into a real aircraft, the SFA scheme was implemented into the flight simulator. In the cases of the YF-22 and Tecnam P92, the preexisting flight datasets were processed through the sensor failure block and SFA to mimic a realistic analytical redundancy scenario. The figure below displays the generic outline used for this work within Simulink.

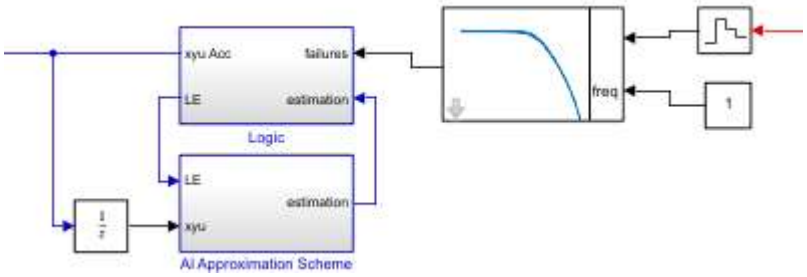

Figure 5.17: Logic and Approximation Subblocks

As seen in Figure 5.17, the flight data from the simulator (the red line on the right) is input through a filter to calculate residual. The data is then processed through the SFA scheme. If the residual exceeds the designated threshold, the fault alarm is triggered, initiating the accommodation process; the on-line estimator in place will begin estimating the new data at this point. Figures 5.18 and 5.19 below display residual graphs from flight datasets that triggered false alarms.

The Logic block follows the flowchart in Figure 5.18.

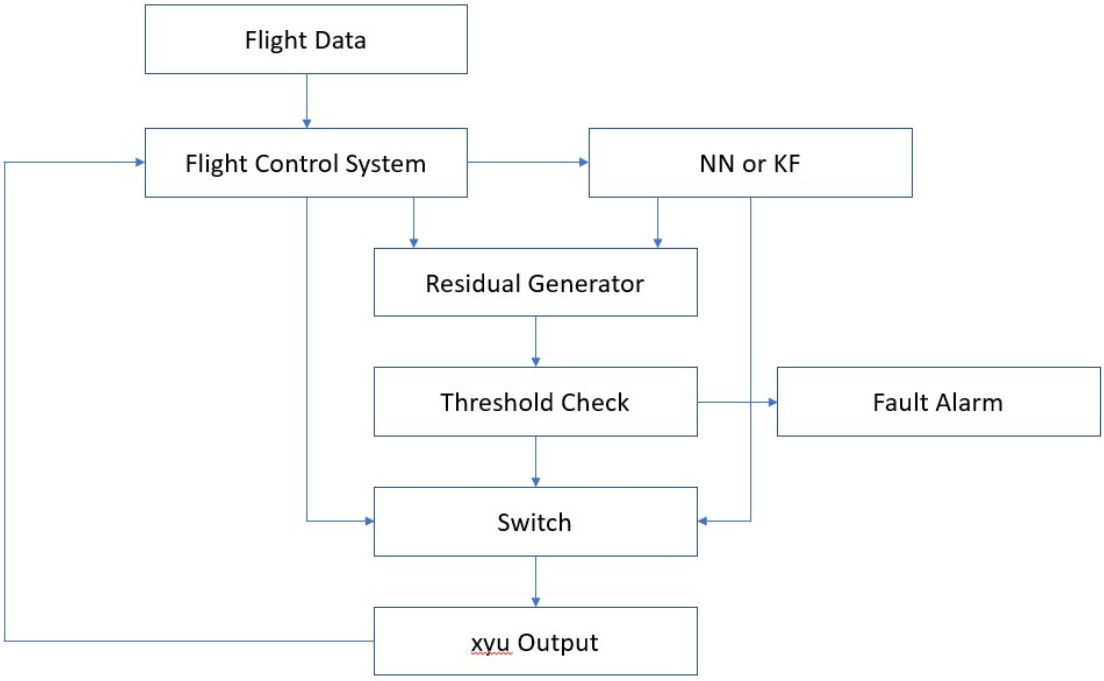

Figure 5.18: Sensor Failure Accommodation Logic Box Flowchart

The SFA Logic block initiates its cycle with the input of the flight data. Both the flight control system values and the NN or KF estimation values are input to the Logic block for comparison purposes. The residual from both datasets is calculated and compared to the user-implemented threshold. If the threshold is exceeded, a fault alarm is triggered and the switch in the next step initiates the usage of the accommodated data instead of the actual flight data. This accommodated data is then input back to the flight control system for the next timestep. Examples of the residual generated in this scheme is seen in the following figures.

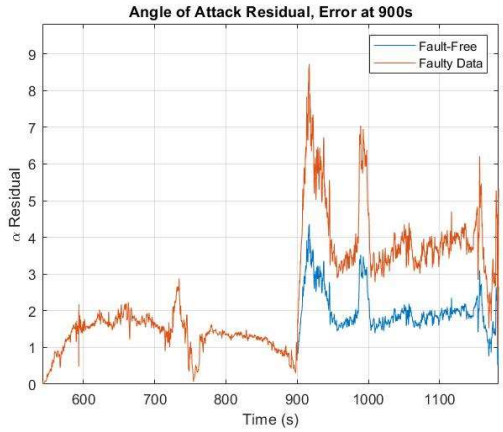

Figure 5.19: Fault-Free vs. Faulty MLP Residual for Angle of Attack for SIM 001

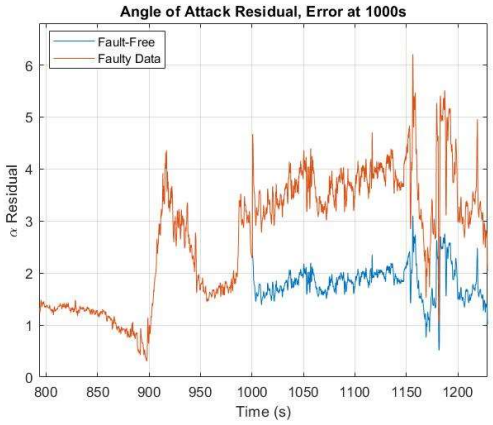

Figure 5.20: Fault-Free vs. Faulty MLP Residual for Angle of Attack on SIM 002

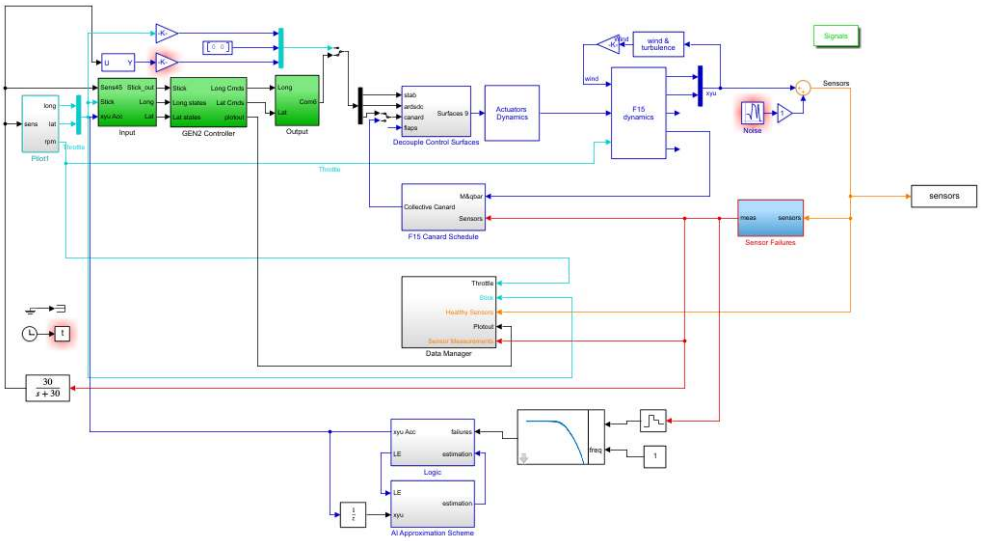

Figure 5.21: General Outline of SFA Scheme within Flight Simulator

Figure 5.21 shows the SFA schematic within the flight simulator. This setup allowed the SFA scheme to work and accommodate failure data in real time with the simulator. This was a similar case to the preexisting flight data; the other flight datasets for the YF-22 and Tecnam P92 were input into the SFA scheme as a loop in time series to allow the SFA scheme to be tested and work in real time.

The NNs and KFs were tested with both the faulted testing data and fault-free training data to better determine the legitimacy of each on-line estimator; in other words, fault-free data was tested to observe whether any of the NN or KF schemes would fire a false alarm for a failure that was not present. The testing data also determined whether present failures were detected.

The following results show a visual comparison of the actual fault-free data to the failure accommodated data. Each SIM used for testing had a fault implemented at a certain point for the SFA to detect and accommodate. With these two sets of values, actual and accommodated, the estimation error, mean, and standard deviations of the values were calculated to numerically represent the results. Additional tables were provided for the NNs to display training information and statistics from the SIMT datasets. It was expected for the accommodated data to have a low mean and standard deviation; in other words, the accommodated data should be almost identical to the original fault-free data.

The display format for the results was inspired from works by Samy *et al.* [1] and Gururajan *et al.* [3]. The overall results at the end of this chapter provide the numerical values for each criterion from Chapter 2 based on all the simulated data.

The MLP and EMRAN were trained by each SIMT dataset and tested against the SIM datasets to determine the best training data for this work with the simulated data. The NN was

tested against each SIM dataset 100 times. Tables 5.2 through 5.13 in this section provide the result averages for each SIMT training flight dataset selected. Based on these values for the 20 simulated datasets, the optimal dataset designated as SIMT 001 was selected as the training data for each sensor due to having the lowest estimation mean for all the sensor failure possibilities: angle of attack, sideslip angle, and true airspeed. The following sections display the MLP-SFA accommodation data and EMRAN-SFA accommodation data compared to the fault-free data for each of the four SIM datasets.

5.4 MLP Results

5.4.1 Angle of Attack

Tables 5.2 and 5.3 display the numerical reasoning behind selecting SIMT 001 as the training dataset for angle of attack in the simulation data. SIMT 001 provided the lowest mean from the actual value. It also required fewer neurons to train the NN.

Table 5.2: MLP Angle of Attack Estimation Error Statistics for Each Simulation Flight Dataset – Training and Testing

	SIM 001		**SIM 002**	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
SIMT 001	$\mathbf{2.394e^{-3}}$	***.2896***	$\mathbf{.9834e^{-2}}$	***.2965***
SIMT 002	.7928	.5620	.8654	.3532
SIMT 003	.6025	.9862	.0625	.9635
SIMT 004	.3965	.8621	.5259	.5954
	SIM 003		**SIM 004**	
SIMT 001	.2759	.1865	.8024	.1593
SIMT 002	.3865	.9465	.0875	.6492
SIMT 003	$\mathbf{2.462e^{-2}}$	***.0956***	.2865	.0164
SIMT 004	.2747	.6492	$\mathbf{8.165e^{-2}}$	***.0252***

Table 5.3: Angle of Attack Error Statistics from MLP Training for Simulated Flight Data

	SIMT 001	SIMT 002	SIMT 003	SIMT 004
Number of active neurons	75	82	92	101
Number of training epochs	1000	1000	1000	1000
Mean of training error	-.0204	.0163	.0749	-.0198
Standard deviation of training error	.3845	.2836	.4721	.2689

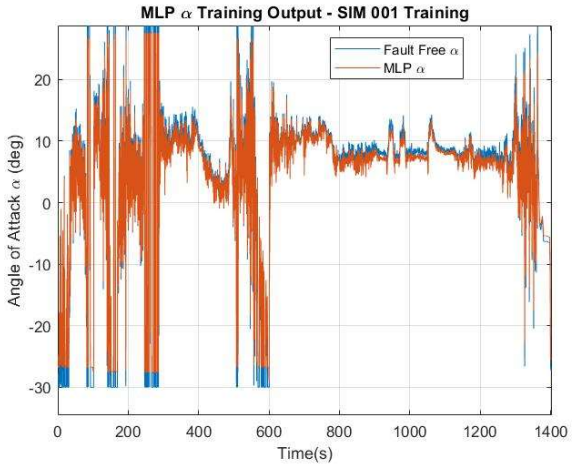

Figure 5.22: MLP Angle of Attack Accommodation vs. Fault Free Data (SIMT 001 Training)

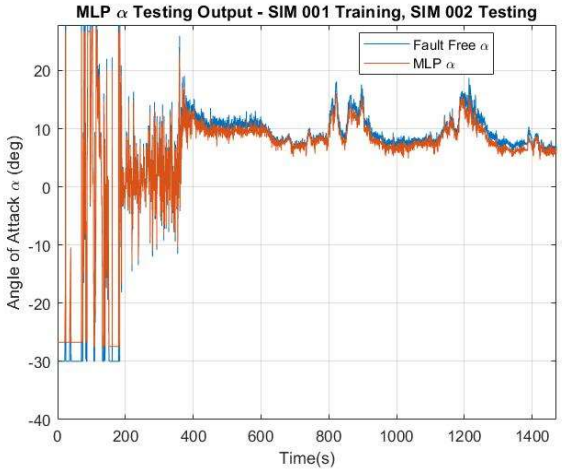

Figure 5.23: MLP Angle of Attack Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 002 Testing)

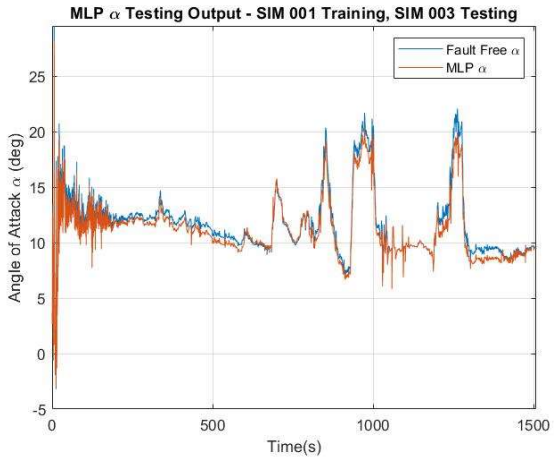

Figure 5.24: MLP Angle of Attack Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 003 Testing)

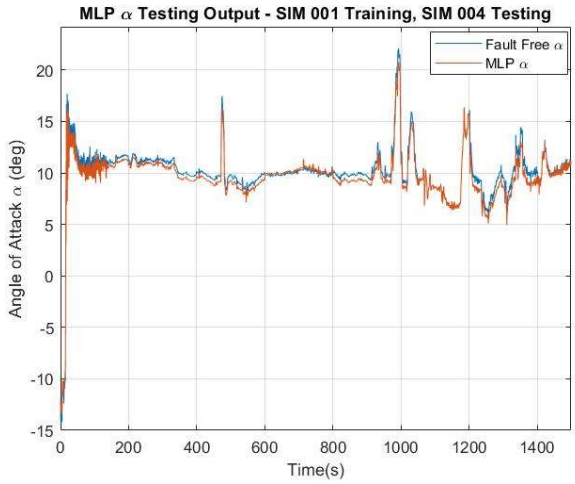

Figure 5.25: MLP Angle of Attack Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 004 Testing)

5.4.2 Sideslip Angle

Tables 5.4 and 5.5 display the numerical reasoning behind selecting SIMT 001 as the training dataset for sideslip angle in the simulation data. SIMT 001 provided the lowest mean from the actual value. It also required fewer neurons to train the NN.

Table 5.4: MLP Sideslip Angle Estimation Error Statistics for Each Simulation Flight Data Set – Training and Testing

	SIM 001		SIM 002	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
SIMT 001	$\mathbf{1.742e^{-4}}$	***.1426***	.8452	.1974
SIMT 002	.1632	.1246	$\mathbf{3.462e^{-2}}$	***.1046***
SIMT 003	.7592	.8275	.0275	.2559
SIMT 004	.3759	.2946	.2682	.6843
	SIM 003		**SIM 004**	
SIMT 001	$\mathbf{7.472e^{-2}}$	***.4927***	.9207	.1496
SIMT 002	.3865	.9465	.3753	.2349
SIMT 003	.2546	.4768	.5839	.2846
SIMT 004	.3776	.9863	$\mathbf{4.452e^{-3}}$	***.1846***

Table 5.5: Sideslip Error Statistics from MLP Training for Simulated Flight Data

	SIMT 001	SIMT 002	SIMT 003	SIMT 004
Number of active neurons	63	65	90	98
Number of training epochs	1000	1000	1000	1000
Mean of training error	-.0384	-.0274	-.0187	.0539
Standard deviation of training error	.7402	.4720	.3856	.2974

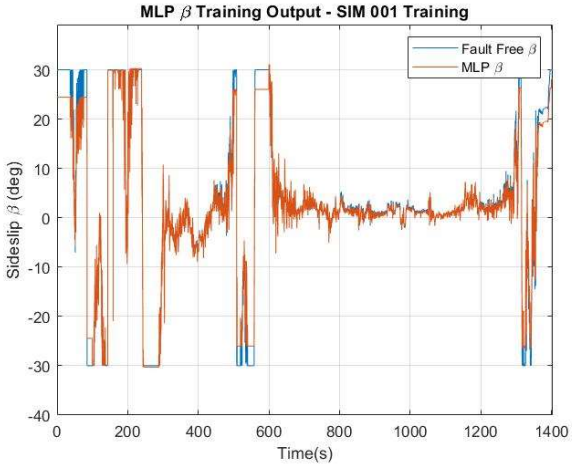

Figure 5.26: MLP Sideslip Accommodation vs. Fault Free Data (SIMT 001 Training)

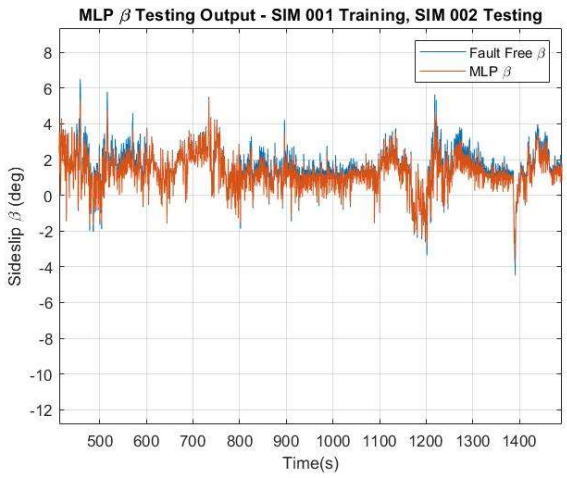

Figure 5.27: MLP Sideslip Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 002 Testing)

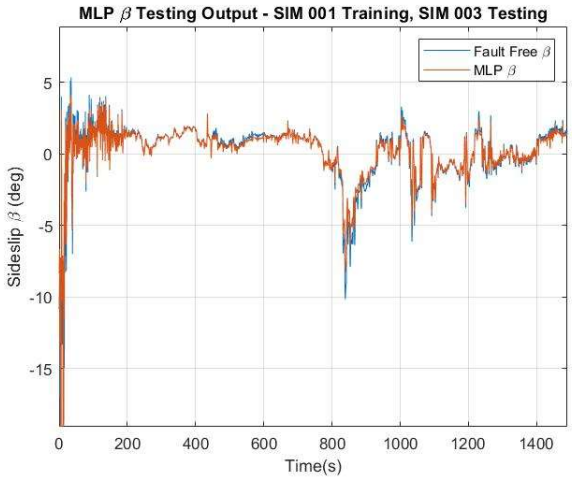

Figure 5.28: MLP Sideslip Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 003 Testing)

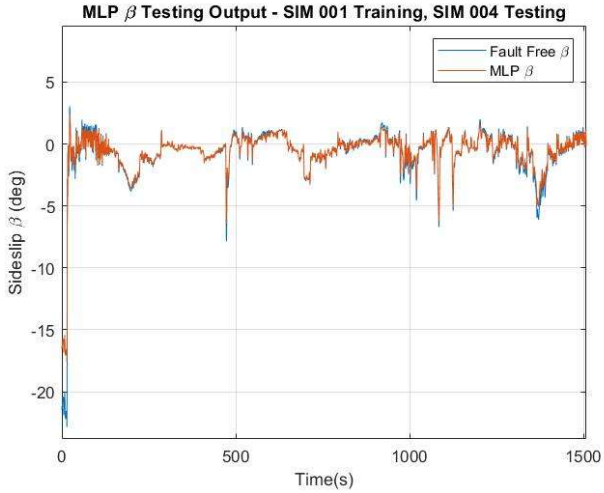

Figure 5.29: MLP Sideslip Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 004 Testing)

5.4.3 True Airspeed

Similar to the angle of attack and sideslip, the SIMT 001 training dataset was used as the training data for the NNs.

Table 5.6: MLP True Airspeed Estimation Error Statistics for Each Simulated Flight Data Set – Training and Testing

	SIM 001		SIM 002	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
SIMT 001	$\mathbf{3.829e^{-6}}$	***.1192***	.3721	.7402
SIMT 002	.3382	.2974	$\mathbf{1.370e^{-3}}$	***.2721***
SIMT 003	.2847	.0972	.4928	.9773
SIMT 004	.4742	.4907	.2974	.4997
	SIM 003		**SIM 004**	
SIMT 001	.4972	.9621	.9372	.3445
SIMT 002	.3335	.5921	.4171	.7638
SIMT 003	$\mathbf{1.211e^{-4}}$	***.3613***	.3186	.3962
SIMT 004	.1737	.4943	$\mathbf{2.667e^{-4}}$	***.4917***

Table 5.7: True Airspeed Error Statistics from MLP Training for Simulated Data

	SIMT 001	SIMT 002	SIMT 003	SIMT 004
Number of active neurons	79	112	97	109
Number of training epochs	1000	1000	1000	1000
Mean of training error	-.0184	.0926	.09352	-.0364
Standard deviation of training error	.5962	.4388	.3443	.4256

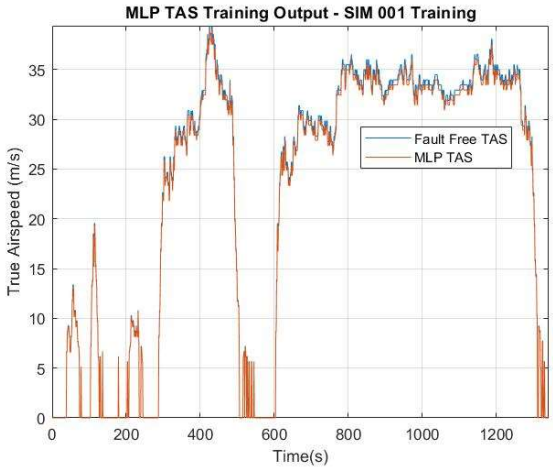

Figure 5.30: MLP True Airspeed Accommodation vs. Fault Free Data (SIMT 001 Training)

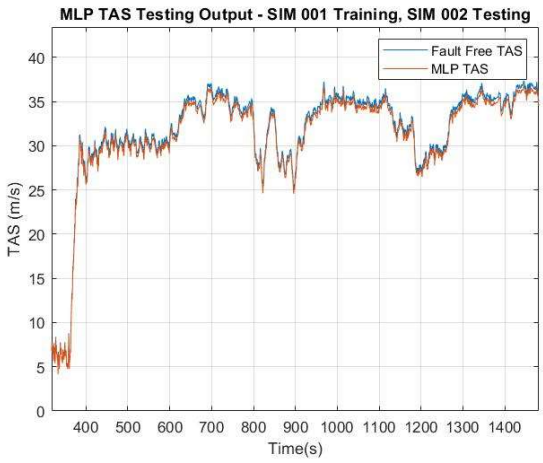

Figure 5.31: MLP True Airspeed Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 002 Testing)

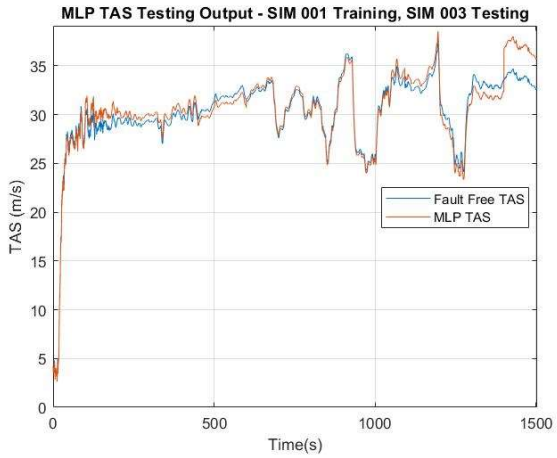

Figure 5.32: MLP True Airspeed Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 003 Testing)

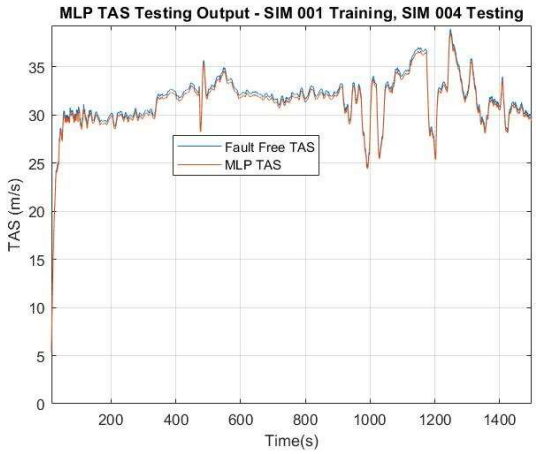

Figure 5.33: MLP True Airspeed Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 004 Testing)

5.5 EMRAN Results

The EMRAN was trained with the SIMT 001 data like the MLP. The training provided similar results of SIMT 001 producing the lowest estimation means and tended to require fewer active neurons. SIMT 001 was used for angle of attack, sideslip, and true airspeed.

5.5.1 Angle of Attack

Table 5.8: EMRAN Angle of Attack Estimation Error Statistics for Each Simulated Flight Data Set – Training and Testing

	SIM 001		SIM 002	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
SIMT 001	**.0394**	***.5733***	**.0472**	***.3275***
SIMT 002	.8333	.4662	.5678	.8947
SIMT 003	.2456	.1174	.3356	.9864
SIMT 004	.3728	.3614	.3352	.8855
	SIM 003		**SIM 004**	
SIMT 001	.8532	.1368	**.0118**	***.3883***
SIMT 002	.3468	.4657	.0875	.4967
SIMT 003	**.0962**	***.0797***	.2164	.4443
SIMT 004	.8643	.6747	.4738	.7839

Table 5.9: Angle of Attack Error Statistics from EMRAN Training for Simulated Data

	SIMT 001	SIMT 002	SIMT 003	SIMT 004
Number of active neurons	67	101	89	72
Number of training epochs	1000	1000	1000	1000
Mean of training error	.0164	.0112	.0136	.0271
Standard deviation of training error	.3529	.1640	.2737	.3622

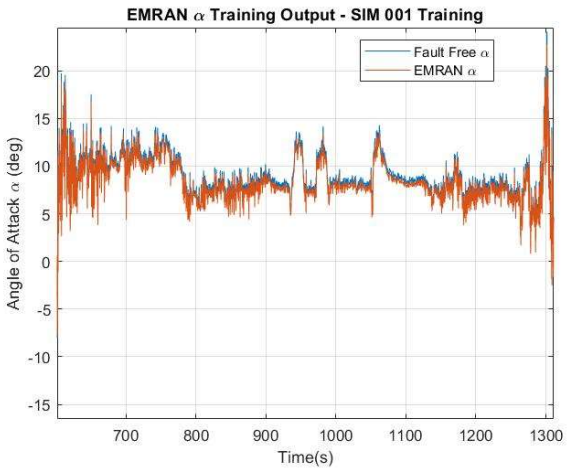

Figure 5.34: EMRAN Angle of Attack Accommodation vs. Fault Free Data (SIMT 001 Training)

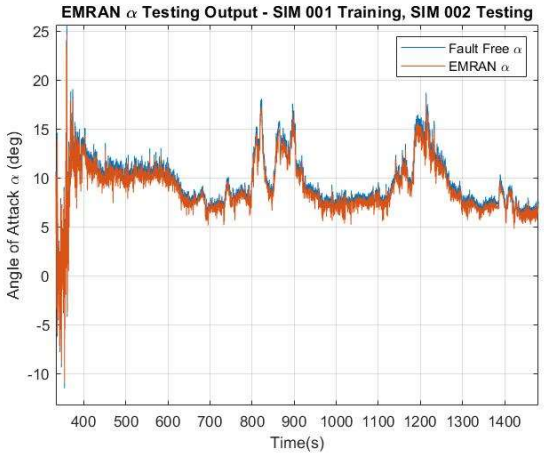

Figure 5.35: EMRAN Angle of Attack Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 002 Testing)

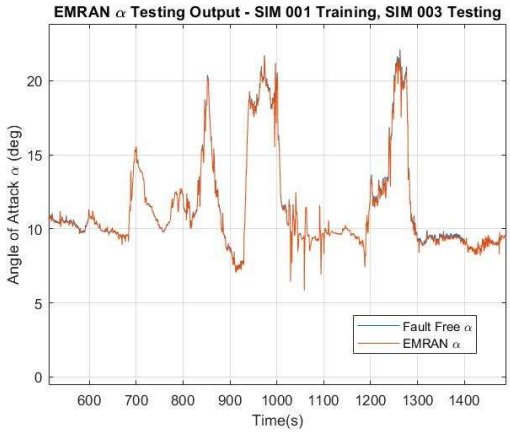

Figure 5.36: EMRAN Angle of Attack Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 003 Testing)

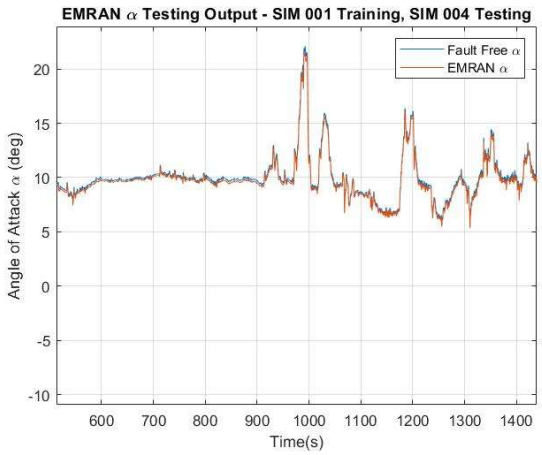

Figure 5.37: EMRAN Angle of Attack Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 004 Testing)

5.5.2 Sideslip Angle

Table 5.10: EMRAN Sideslip Angle Estimation Error Statistics for Each Simulated Flight Data Set – Training and Testing

	SIM 001		SIM 002	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
SIMT 001	***.0482***	***.2975***	***.0759***	***.1986***
SIMT 002	.3852	.3642	.8833	.7654
SIMT 003	.5839	.4774	.9626	.9746
SIMT 004	.5756	.3947	.3740	.2740
	SIM 003		**SIM 004**	
SIMT 001	***.0233***	***.5026***	.2865	.7402
SIMT 002	.4224	.4335	.4926	.9264
SIMT 003	.4658	.7859	.8644	.8575
SIMT 004	.3725	.5447	***.0187***	***.2542***

Table 5.11: Sideslip Error Statistics from EMRAN Training for Simulated Data

	SIMT 001	SIMT 002	SIMT 003	SIMT 004
Number of active neurons	89	96	92	112
Number of training epochs	1000	1000	1000	1000
Mean of training error	.0174	-.0233	-.0455	.0927
Standard deviation of training error	.1272	.1462	.1749	.1255

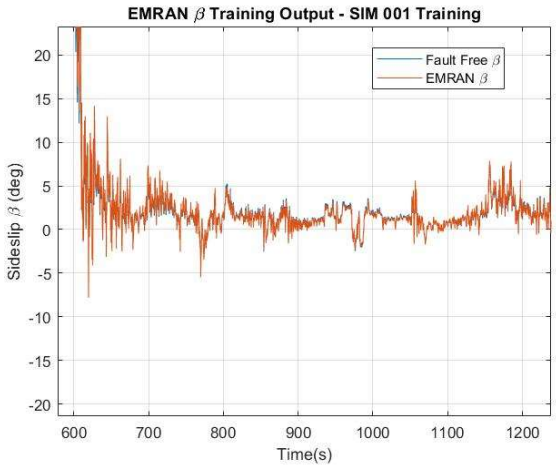

Figure 5.38: EMRAN Sideslip Accommodation vs. Fault Free Data (SIMT 001 Training)

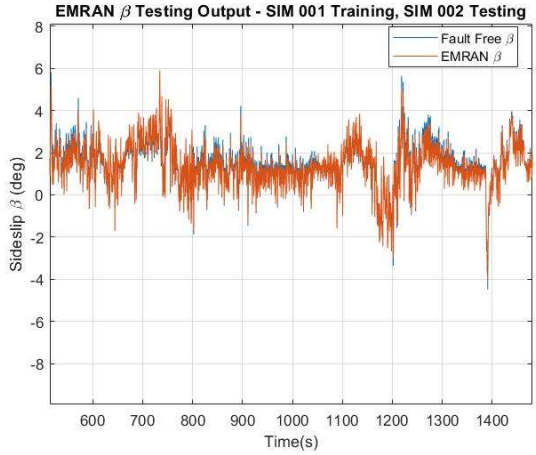

Figure 5.39: EMRAN Sideslip Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 002 Testing)

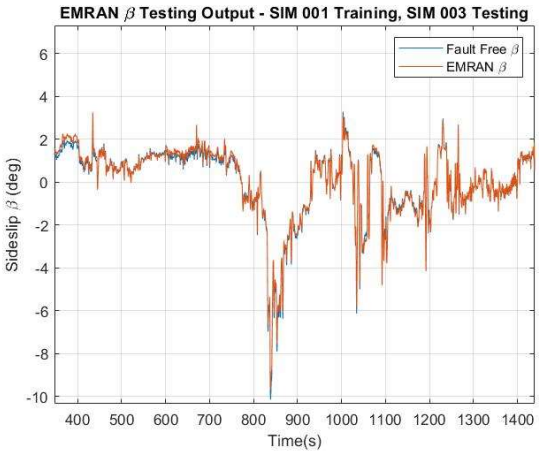

Figure 5.40: EMRAN Sideslip Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 003 Testing)

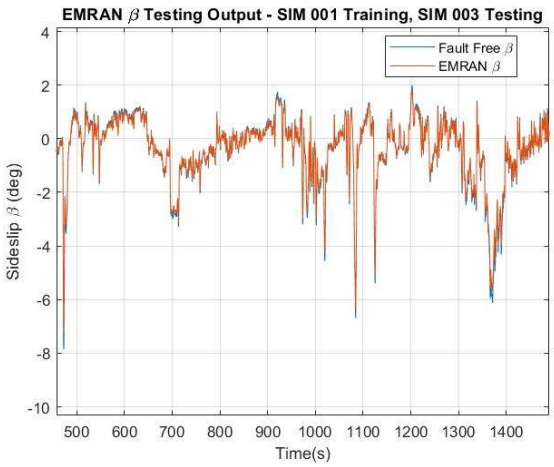

Figure 5.41: EMRAN Sideslip Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 004 Testing)

5.5.3 True Airspeed

Table 5.12: EMRAN True Airspeed Estimation Error Statistics for Each Simulated Flight Data Set – Training and Testing

	SIM 001		SIM 002	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
SIMT 001	***.0262***	***.0926***	.4597	.7573
SIMT 002	.4377	.6748	**.0355**	***.3335***
SIMT 003	.3659	.5583	.2975	.3975
SIMT 004	.9597	.9483	.4972	.3077
	SIM 003		**SIM 004**	
SIMT 002	.3855	.5112	***.0563***	***.1242***
SIMT 003	.4995	.5972	.1145	.4475
SIMT 004	***.0422***	***.3840***	.3722	.2888
SIMT 005	.4989	.5362	.2100	.7533

Table 5.13: True Airspeed Error Statistics from EMRAN Training for Simulated Data

	SIMT 001	SIMT 002	SIMT 003	SIMT 004
Number of active neurons	86	108	103	114
Number of training epochs	1000	1000	1000	1000
Mean of training error	.0373	-.0277	-.0394	-.0119
Standard deviation of training error	.1836	.1242	.1974	.2929

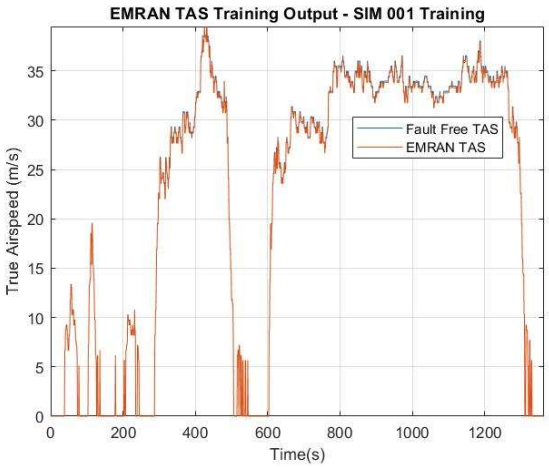

Figure 5.42: EMRAN True Airspeed Accommodation vs. Fault Free Data (SIMT 001 Training)

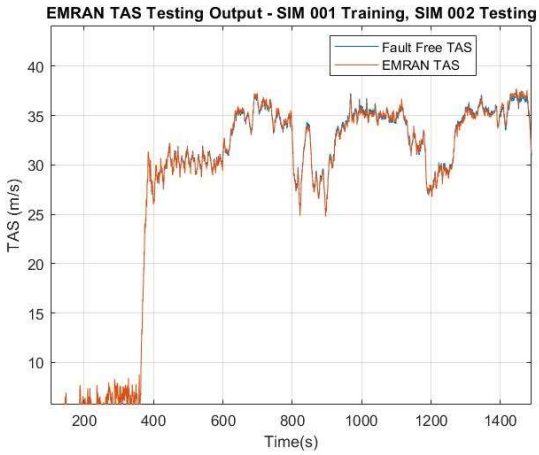

Figure 5.43: EMRAN True Airspeed Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 002 Testing)

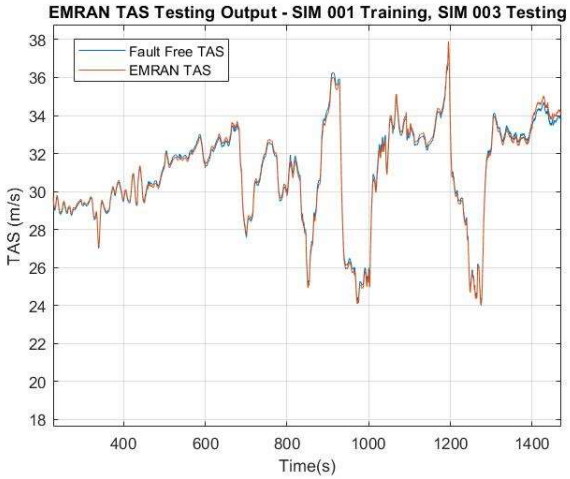

Figure 5.44: EMRAN True Airspeed Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 003 Testing)

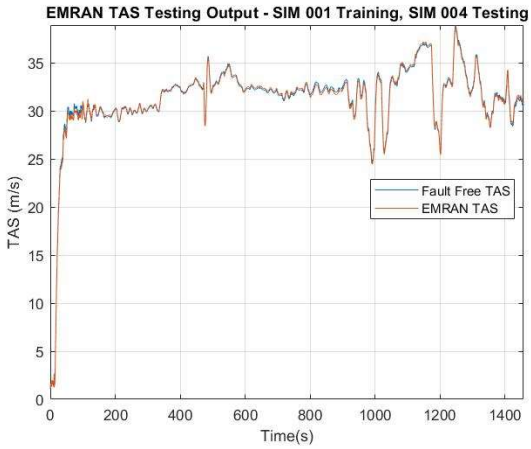

Figure 5.45: EMRAN True Airspeed Accommodation vs. Fault Free Data (SIMT 001 Training, SIM 004 Testing)

5.6 EKF Results

The following graphs display the performance of the EKF when exposed to the simulated data. The EKF used the covariance matrix values designated below.

$Q_{k,\alpha}$

$= diag[5.01 * 10^{-4} \quad 0.99 * 10^{-3} \quad 7.88 * 10^{-4} \quad 2.02 * 10^{-7} \quad 2.74 * 10^{-7} \quad 4.67 * 10^{-7} \quad \cdots]$

$[\cdots \quad 3.53 * 10^{-6} \quad 3.53 * 10^{-6} \quad 3.53 * 10^{-6}]$

$R_{k,\alpha} = diag[1.84 * 10^{-5} \quad 1.96 * 10^{-5} \quad 9.93 * 10^{-6} \quad 5.77 * 10^{-6} \quad .48 * 10^{-5}]$

$Q_{k,\beta}$

$= diag\,[4.56 * 10^{-4} \quad 1.83 * 10^{-3} \quad 1.24 * 10^{-4} \quad 2.34 * 10^{-7} \quad 1.87 * 10^{-7} \quad 7.92 * 10^{-7} \quad \cdots]$

$[\cdots \quad 1.99 * 10^{-6} \quad 8.16 * 10^{-6} \quad 4.11 * 10^{-6}]$

$R_{k,\beta} = diag[1.19 * 10^{-5} \quad 2.11 * 10^{-5} \quad 2.66 * 10^{-6} \quad 5.75 * 10^{-6} \quad 6.98 * 10^{-6}]$

$Q_{k,TAS}$

$= diag[5.01 * 10^{-4} \quad 0.99 * 10^{-3} \quad 7.88 * 10^{-4} \quad 2.02 * 10^{-7} \quad 2.74 * 10^{-7} \quad 4.67 * 10^{-7} \quad 5.33 * 10^{-6}]$

$R_{k,TAS} = diag[1.08 * 10^{-5} \quad 1.09 * 10^{-5} \quad 9.39 * 10^{-6} \quad 5.10 * 10^{-6} \quad 7.12 * 10^{-5}]$

The units within the Q and R matrices are of m/s^2 for accelerations, *deg/s* for angular rates, and *deg* for angles.

As mentioned, the KFs did not require training like the NNs so they were tested directly with the simulation flight datasets.

5.6.1 Angle of Attack

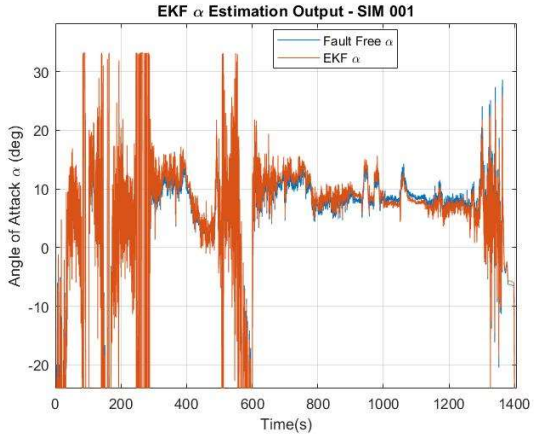

Figure 5.46: EKF Angle of Attack Accommodation vs. Fault Free Data (SIM 001 Testing)

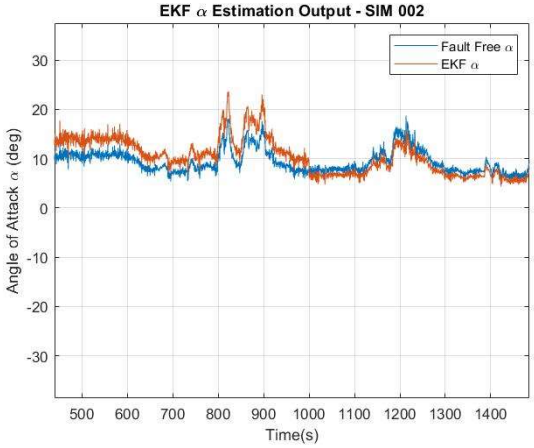

Figure 5.47: EKF Angle of Attack Accommodation vs. Fault Free Data (SIM 002 Testing)

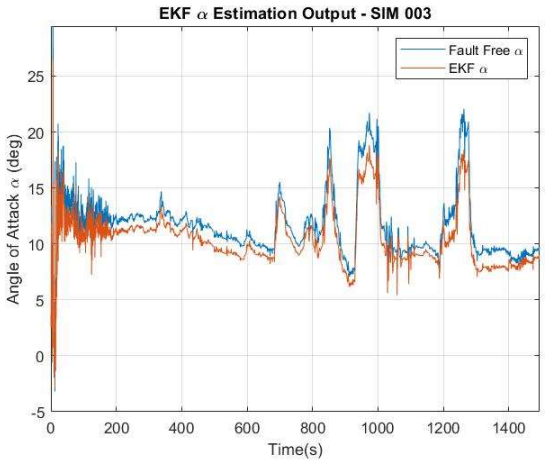

Figure 5.48: EKF Angle of Attack Accommodation vs. Fault Free Data (SIM 003 Testing)

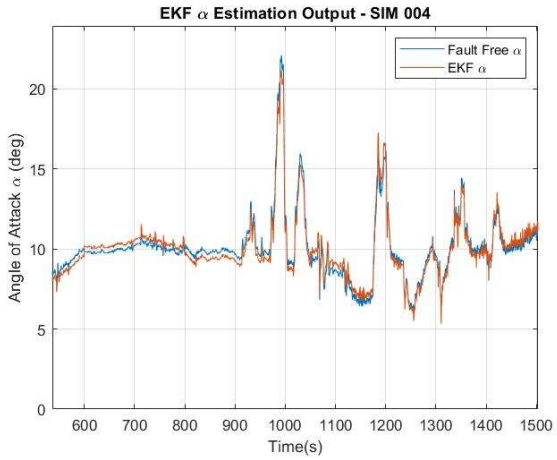

Figure 5.49: EKF Angle of Attack Accommodation vs. Fault Free Data (SIM 004 Testing)

5.6.2 Sideslip Angle

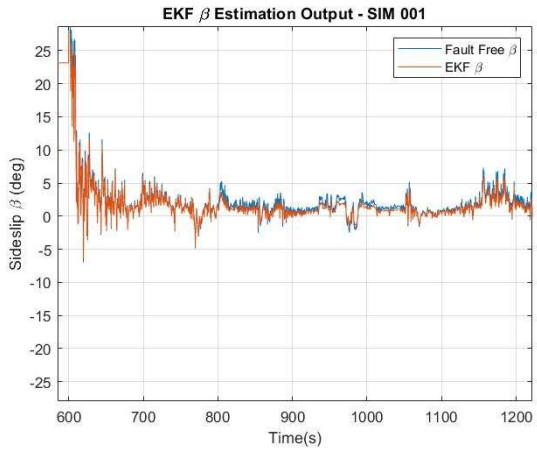

Figure 5.50: EKF Sideslip Accommodation vs. Fault Free Data (SIM 001 Testing)

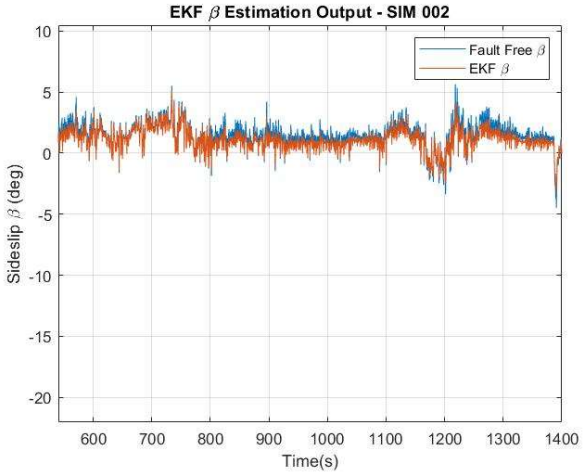

Figure 5.51: EKF Sideslip Accommodation vs. Fault Free Data (SIM 002 Testing)

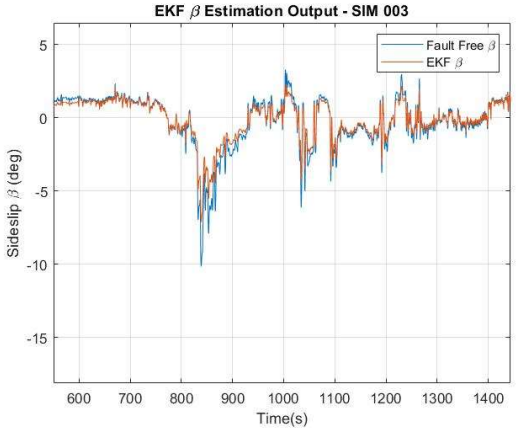

Figure 5.52: EKF Sideslip Accommodation vs. Fault Free Data (SIM 003 Testing)

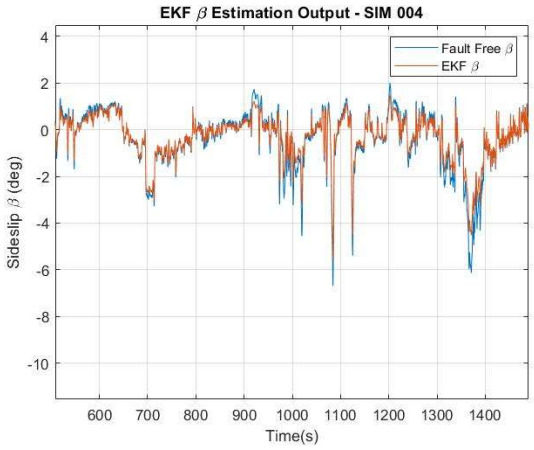

Figure 5.53: EKF Sideslip Accommodation vs. Fault Free Data (SIM 004 Testing)

5.6.3 True Airspeed

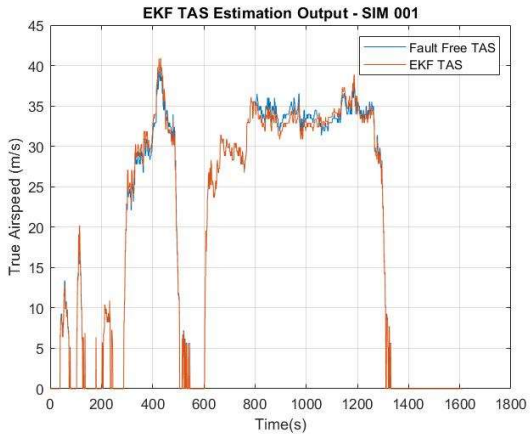

Figure 5.54: EKF True Airspeed Accommodation vs. Fault Free Data (SIM 001 Testing)

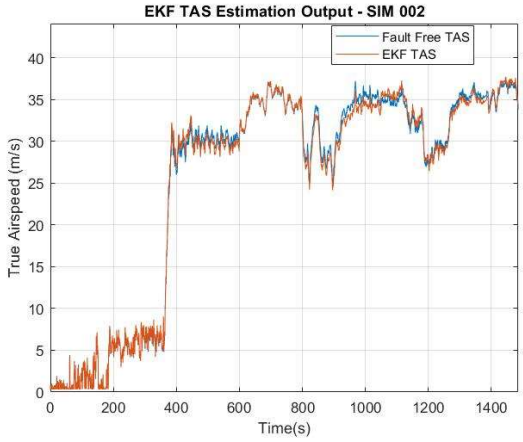

Figure 5.55: EKF True Airspeed Accommodation vs. Fault Free Data (SIM 002 Testing)

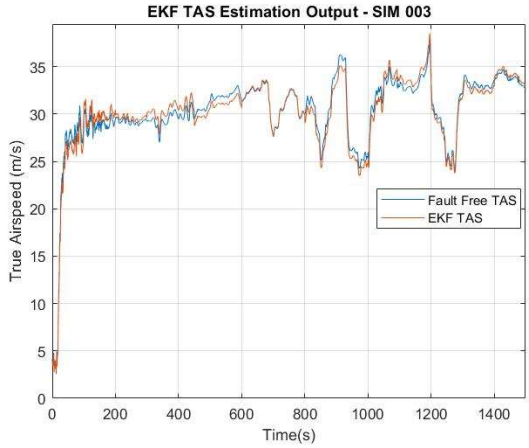

Figure 5.56: EKF True Airspeed Accommodation vs. Fault Free Data (SIM 003 Testing)

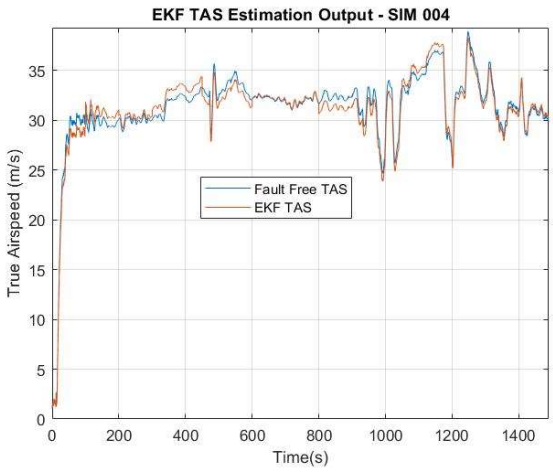

Figure 5.57: EKF True Airspeed Accommodation vs. Fault Free Data (SIM 004 Testing)

5.7 UKF Results

The UKF used the same covariance matrices as the EKF that were denoted Section 5.6 to accurately estimate and accommodate the sensor failure. The following graphs display the resulting performance of the UKF when exposed to the simulation data.

5.7.1 Angle of Attack

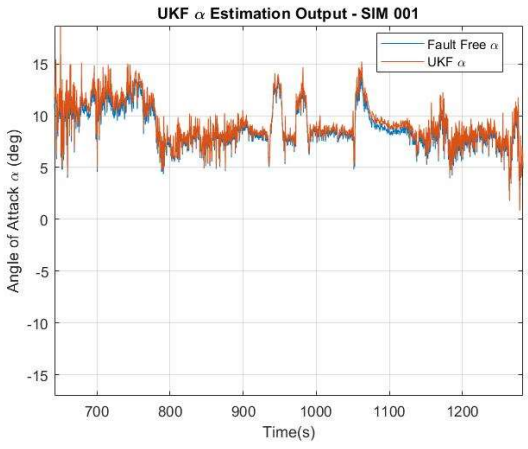

Figure 5.58: UKF Angle of Attack Accommodation vs. Fault Free Data (SIM 001 Testing)

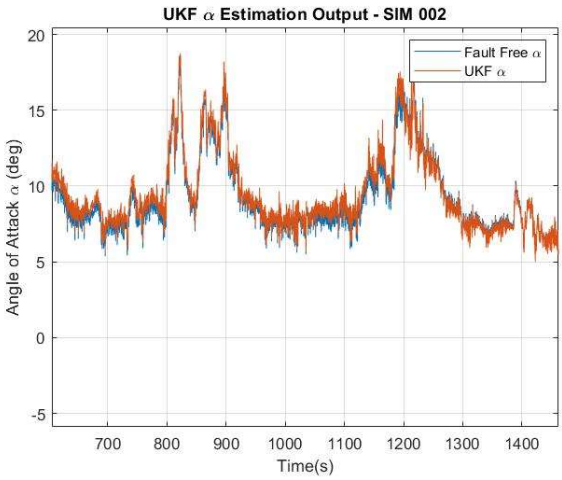

Figure 5.59: UKF Angle of Attack Accommodation vs. Fault Free Data (SIM 002 Testing)

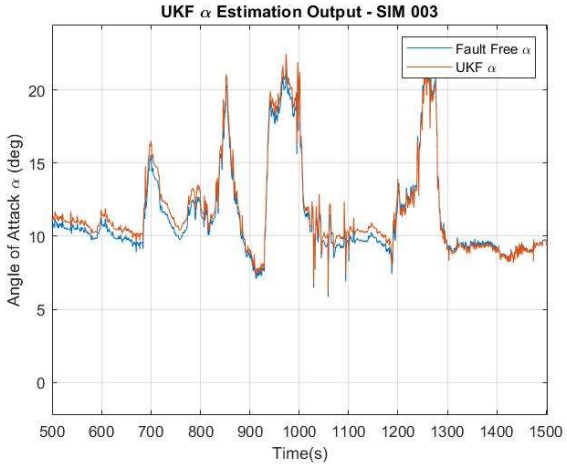

Figure 5.60: UKF Angle of Attack Accommodation vs. Fault Free Data (SIM 003 Testing)

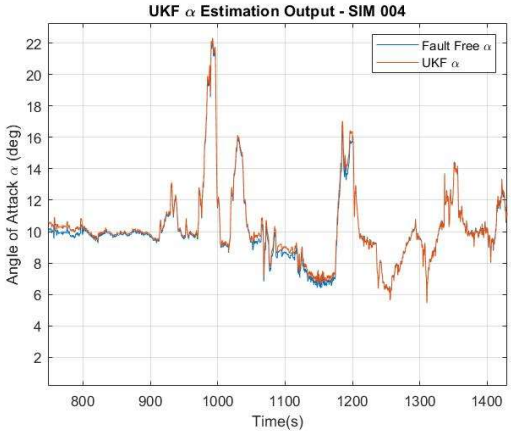

Figure 5.61: UKF Angle of Attack Accommodation vs. Fault Free Data (SIM 004 Testing)

5.7.2 Sideslip Angle

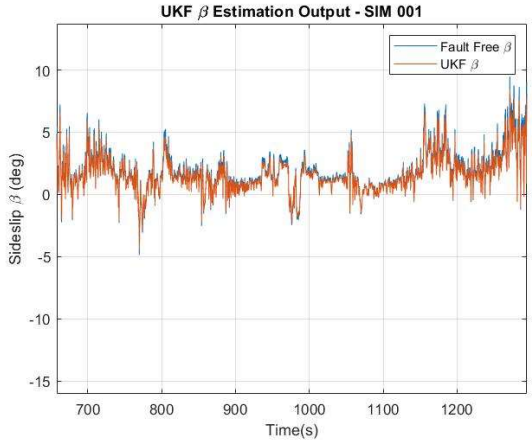

Figure 5.62: UKF Sideslip Accommodation vs. Fault Free Data (SIM 001 Testing)

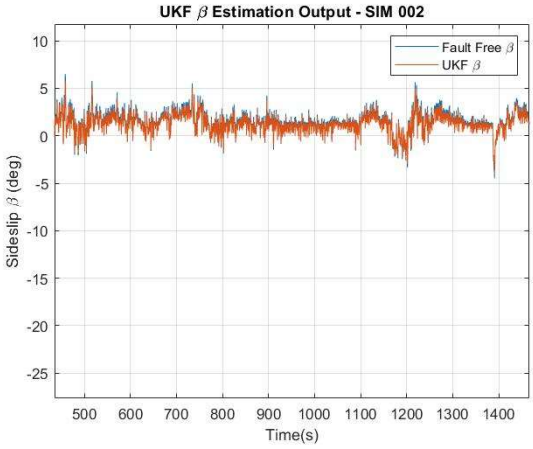

Figure 5.63: UKF Sideslip Accommodation vs. Fault Free Data (SIM 002 Testing)

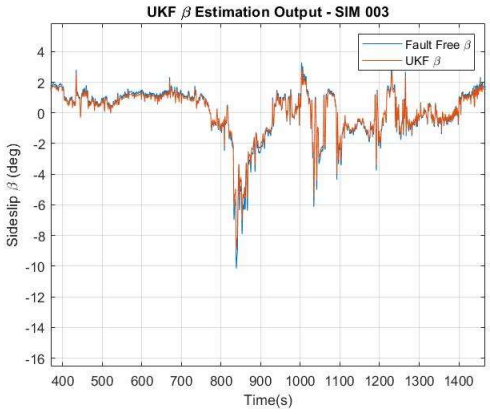

Figure 5.64: UKF Sideslip Accommodation vs. Fault Free Data (SIM 003 Testing)

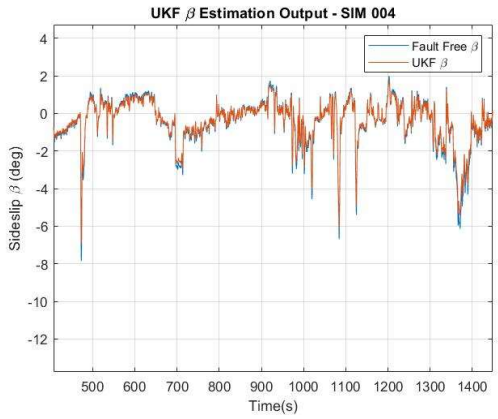

Figure 5.65: UKF Sideslip Accommodation vs. Fault Free Data (SIM 004 Testing)

5.7.3 True Airspeed

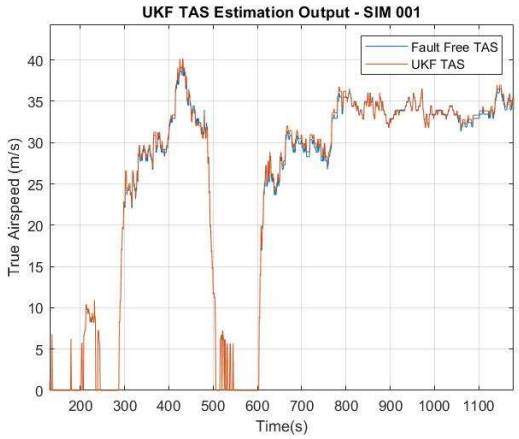

Figure 5.66: UKF True Airspeed Accommodation vs. Fault Free Data (SIM 001 Testing)

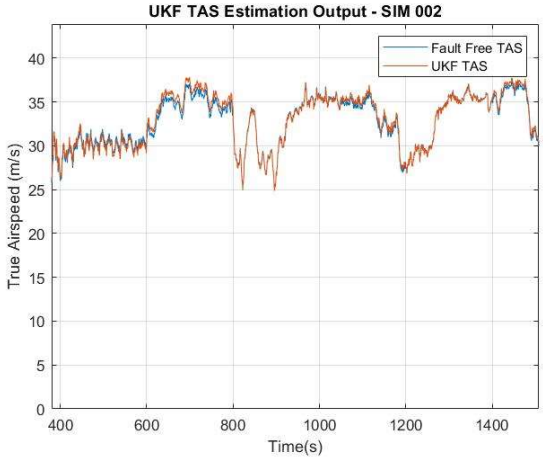

Figure 5.67: UKF True Airspeed Accommodation vs. Fault Free Data (SIM 002 Testing)

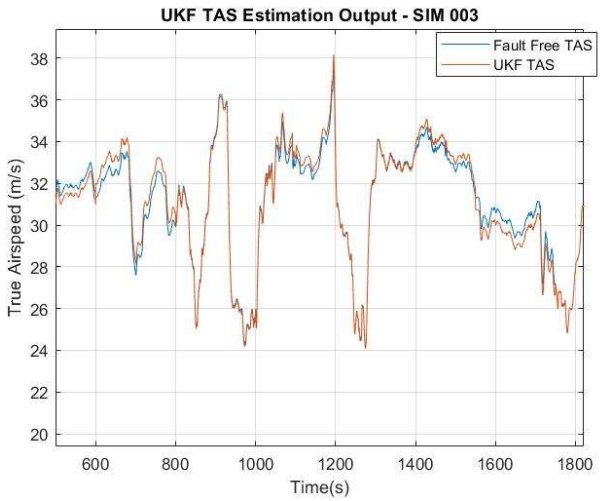

Figure 5.68: UKF True Airspeed Accommodation vs. Fault Free Data (SIM 003 Testing)

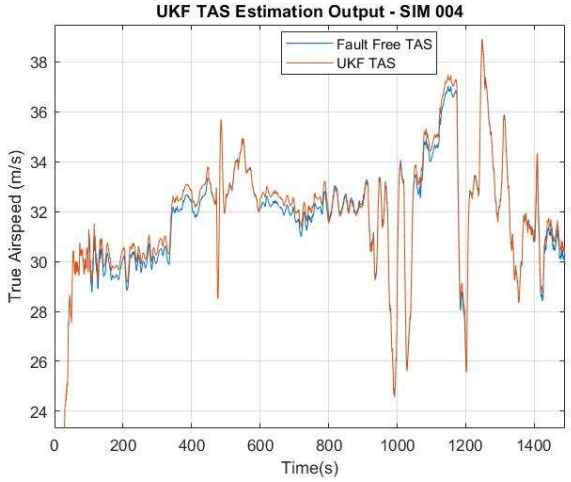

Figure 5.69: UKF True Airspeed Accommodation vs. Fault Free Data (SIM 004 Testing)

5.8 Summary of Simulation Results

The following tables display the numerical averages for mean and standard deviation of the accommodation data for each of the SFA schemes when tested with the selected four simulated datasets. Means closer to zero with lower standard deviations were correlated to more accurate accommodations.

Table 5.14: Comparison of All SFA Approaches for Angle of Attack Simulation Data

SIM	MLP		EMRAN		EKF		UKF	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
001	.0284	.2075	.0147	.1984	.4942	.4729	.3746	.2842
002	.0372	.2173	.0107	.1992	.2085	.2564	.2857	.5222
003	-.0363	.4937	-.0285	.1876	.1964	.2483	.2740	.8462
004	-.0236	.2344	.0229	.2001	.1642	.8372	.4830	.3795

Table 5.15: Comparison of All SFA Approaches for Sideslip Angle Simulation Data

SIM	MLP		EMRAN		EKF		UKF	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
001	.0382	.2974	-.0272	.1344	.3373	.4729	.1175	.3792
002	.0221	.4792	-.0742	.1111	.3829	.5887	.2862	.3375
003	-.0283	.2247	.0266	.2482	.2742	.5729	.2114	.4722
004	.0422	.4483	.0138	.2811	.1964	.2974	.1864	.4282

Table 5.16: Comparison of All SFA Approaches for True Airspeed Simulation Data

SIM	MLP		EMRAN		EKF		UKF	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
001	.0765	.7492	-.0244	.6534	.7305	.3573	.3387	.2543
002	.0365	.3463	.0264	.3865	.6264	.7593	.2642	.7425
003	.0975	.8645	.0265	.3543	.6775	.9765	.3629	.8543
004	.0355	.7736	.0554	.4739	.8463	.7503	.4325	.1253

The following tables provide a numerical summary of the performance from each SFA scheme for all 20 simulated flights.

Table 5.17: Results of Simulated Data with Large Errors

	α	β	*TAS*
Estimation Error			
MLP	.3291	.1214	.7723
EMRAN	.0182	.0338	.4979
EKF	.7498	.2976	.9754
UKF	.4825	.2118	.8979
Fault Detection Time			
MLP	.2897	.2939	.3825
EMRAN	.1973	.2844	.2976
EKF	.5722	.9874	.6935
UKF	.4791	.5727	.5580
False Alarms			
MLP	0	0	0
EMRAN	0	0	0
EKF	1	0	1
UKF	0	0	0
Undetected Faults			
MLP	0	0	0
EMRAN	0	0	0
EKF	0	0	0
UKF	0	0	0
Detectability Ratio			
MLP	120.65	117.59	165.33
EMRAN	198.53	186.96	195.99
EKF	.04	34.68	.05
UKF	443.53	203.87	267.12

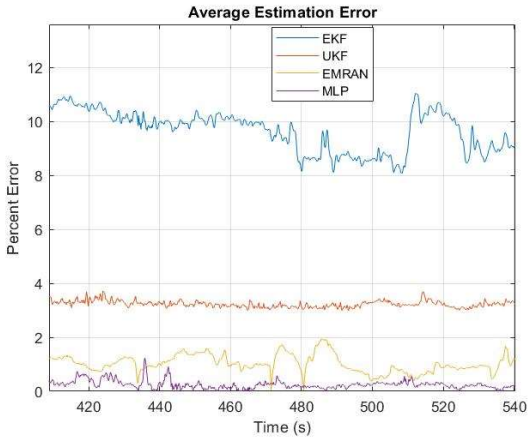

Figure 5.70: Visual Representation of Large Estimation Error Comparison over Time for True Airspeed SIM 002

Table 5.18: Results of Simulated Data with Small Errors

	α (deg)	*β (deg)*	*TAS (m/s)*
Estimation Error			
MLP	.3291	.1212	.8723
EMRAN	.2934	.0285	.6835
EKF	.4920	.8768	.3971
UKF	.3740	.2883	.3755
Fault Detection Time			
MLP	.374	.344	.432
EMRAN	.281	.317	.332
EKF	.638	1.02	.873
UKF	.621	.614	.722
False Alarms			
MLP	0	0	0
EMRAN	0	0	0
EKF	2	1	0
UKF	0	0	0
Undetected Faults			
MLP	0	0	0
EMRAN	0	0	0
EKF	0	1	1
UKF	0	0	1
Detectability Ratio			
MLP	146.38	173.93	164.27
EMRAN	189.48	133.85	190.02
EKF	.011	.026	306.96
UKF	402.11	379.32	327.08

The results in this chapter show that each SFA scheme implemented into the flight simulator performed well to an extent. The EMRAN NN provided the most accurate accommodation data with the lowest estimation error, no false alarms nor undetected faults, and very low standard deviation from the actual values of the angle of attack, sideslip, and true airspeed. The MLP also performed well but was not as accurate as the EMRAN in terms of estimation error. The UKF was sufficient in terms of a KF; the system of equations and covariance matrices implemented into the system were able to help provide accurate estimations

with lower estimation error than the EKF. The KFs had an advantage over the NNs in terms of the DR; the KFs were keener to detecting the failures but had more false alarms as a result.

Chapter #6: Performance Analysis Using YF-22 Flight Data

There was a total of four flight datasets for the YF-22 analyzed in this chapter. The NNs and KFs were adjusted to accommodate data specifically from the YF-22 as mentioned in Chapter 5. This case was unique as there was no sensor for true airspeed provided. The values of the true airspeed were calculated within the MATLAB code using the other provided variables to calculate a set of values that can be compared to the SFA accommodations. The datasets were provided as .m files so they could be processed immediately in MATLAB.

The angle of attack flight data from the YF-22 was not considered due to a malfunction of the potentiometer of the angle of attack vane for several flights. The sideslip angle and true airspeed provided sufficient data for analysis in this chapter.

The MLP and EMRAN were trained by each dataset and tested against the remaining datasets to determine the best training data for this work. The NN was tested against each dataset 100 times. The YF-22 only had four flights recorded so there were less training trials to perform compared to the simulator trials in Chapter 5. The YF-22 flights were designated FDS 002, FDS 003, FDS 004, and FDS 005. The optimal datasets for sideslip angle and true airspeed were FDS 004 and FDS 002, respectively. The following figures display the MLP-SFA accommodation data and EMRAN-SFA accommodation data compared to the fault-free data for each of the four flight datasets. Both KFs were tested against every dataset to determine their estimation capabilities.

6.1 MLP Results

6.1.1 Sideslip Angle

The dataset selected to train the NNs for sideslip of the YF-22 was FDS 004. FDS 004 provided the lowest mean values compared to the other flight datasets. This is shown in the following tables.

Table 6.1: MLP Sideslip Angle Estimation Error Statistics for Each YF-22 Flight Data Set – Training and Testing

	FDS 002 (test)		**FDS 003 (test)**	
Training FDS	**Mean (deg)**	**SD (deg)**	**Mean (deg)**	**SD (deg)**
FDS 002	$\mathbf{.3347e^{-3}}$	***.1345***	.8147	.9058
FDS 003	.9246	.8624	$\mathbf{1.543e^{-4}}$	***.2456***
FDS 004	.3408	.3456	.1270	.8472
FDS 005	.2075	.4762	.6365	.2347
	FDS 004 (test)		**FDS 005 (test)**	
FDS 002	.9134	.6324	.9274	.7255
FDS 003	.2785	.5469	.9836	.2750
FDS 004	$\mathbf{3.525e^{-5}}$	***.3516***	.1486	.1284
FDS 005	.9575	.7194	$\mathbf{.0972e^{-4}}$	***.1197***

Table 6.2: Sideslip Error Statistics from MLP Training for YF-22

	FDS 002	**FDS 003**	**FDS 004**	**FDS 005**
Number of active neurons	68	70	74	82
Number of training epochs	1000	1000	1000	1000
Mean of training error	.0479	.0395	.0375	.0885
Standard deviation of training error	.9257	.2076	.2133	.1753

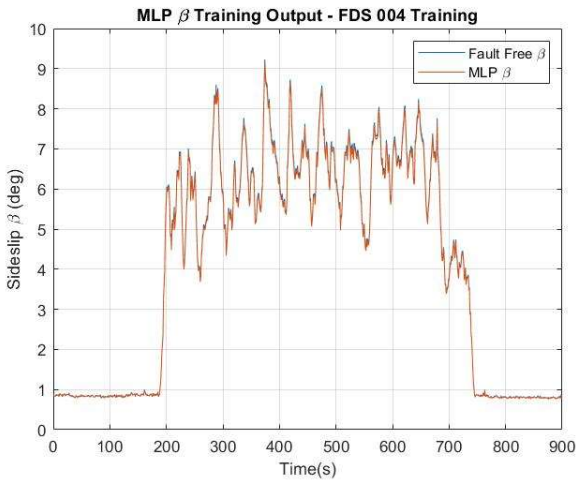

Figure 6.1: MLP Sideslip Accommodation vs. Fault Free Data (FDS 004 Training)

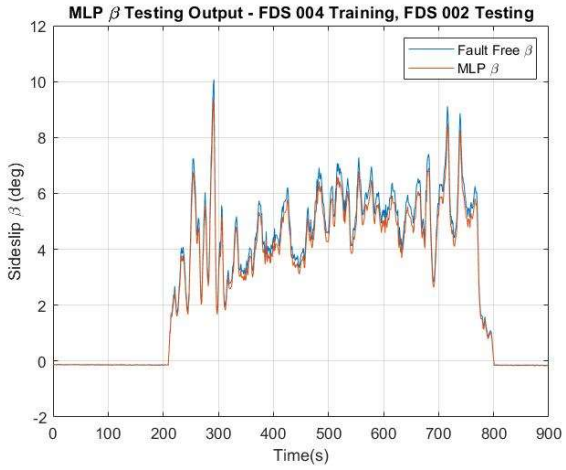

Figure 6.2: MLP Sideslip Accommodation vs. Fault Free Data (FDS 004 Training, FDS 002 Testing)

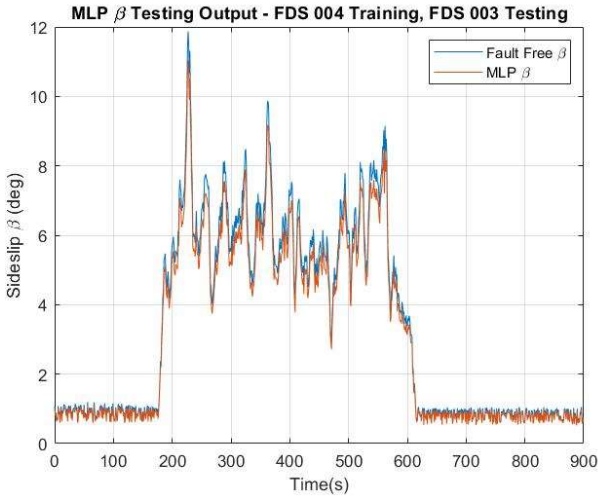

Figure 6.3: MLP Sideslip Accommodation vs. Fault Free Data (FDS 004 Training, FDS 003 Testing)

6.1.2 True Airspeed

The training dataset selected for true airspeed of the YF-22 was FDS 002. This dataset resulted in the optimal training of the NNs for the true airspeed. The mean values upon which this conclusion was based can be seen in the following tables 6.5 and 6.6.

Table 6.3: MLP True Airspeed Estimation Error Statistics for Each YF-22 Flight Data Set – Training and Testing

	FDS 002 (test)		**FDS 003 (test)**	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
FDS 002	$\mathbf{.3347e^{-6}}$	***.5314***	.9264	.1469
FDS 003	.6566	.7232	$\mathbf{1.543e^{-4}}$	***.1037***
FDS 004	.2745	.4775	.2764	.7253
FDS 005	.4819	.4629	.8203	.8016
	FDS 004 (test)		**FDS 005 (test)**	
FDS 002	.2046	.6104	.7896	.2748
FDS 003	.5026	.6072	.8265	.1946
FDS 004	$\mathbf{3.106e^{-5}}$	***.3343***	.9362	.4973
FDS 005	.3856	.7194	$\mathbf{.0972e^{-5}}$	***.2975***

Table 6.4: True Airspeed Error Statistics from MLP Training for YF-22

	FDS 002	**FDS 003**	**FDS 004**	**FDS 005**
Number of active neurons	101	82	104	115
Number of training epochs	1000	1000	1000	1000
Mean of training error	.0124	.0749	.0264	.0962
Standard deviation of training error	.2562	.2566	.3472	.3752

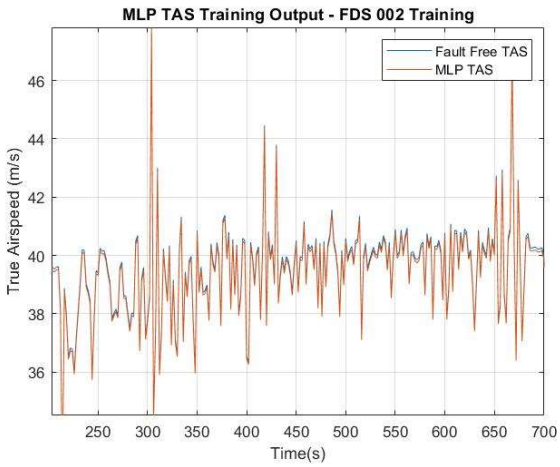

Figure 6.4: MLP True Airspeed Accommodation vs. Fault Free Data (FDS 002 Training)

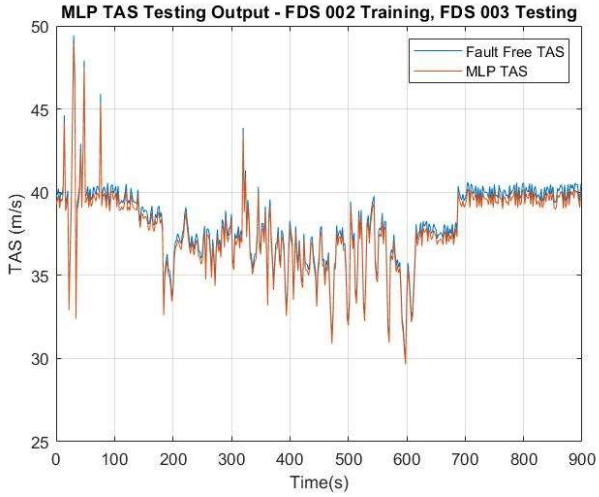

Figure 6.5: MLP True Airspeed Accommodation vs. Fault Free Data (FDS 002 Training, FDS 003)

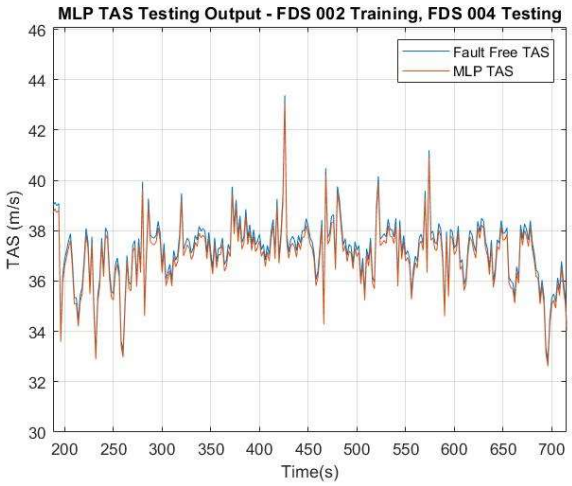

Figure 6.6: MLP True Airspeed Accommodation vs. Fault Free Data (FDS 002 Training, FDS 004)

6.2 EMRAN Results

The EMRAN was trained with the same training datasets selected for the MLP due to the lower mean values and fewer active neurons. As seen in tables 6.7 through 6.12, FDS 005 provided the lowest means for the YF-22 angle of attack, FDS 004 was the optimal training data for sideslip angle, and FDS 002 was selected as the best training data for true airspeed.

6.2.1 Sideslip Angle

Table 6.5: EMRAN Sideslip Angle Estimation Error Statistics for Each YF-22 Flight Data Set – Training and Testing

	FDS 002 (test)		**FDS 003 (test)**	
Training FDS	**Mean (deg)**	**SD (deg)**	**Mean (deg)**	**SD (deg)**
FDS 002	***.0163***	***.1535***	.2754	.5567
FDS 003	.3785	.2553	***.0522***	***.2754***
FDS 004	.3554	.6322	.2776	.2219
FDS 005	.2255	.7642	.2654	.4463
	FDS 004 (test)		**FDS 005 (test)**	
FDS 002	.2654	.4435	.5489	.2906
FDS 003	.2557	.4282	.2699	.3997
FDS 004	***.0133***	***.2545***	.2667	.7472
FDS 005	.2461	.3527	***.0345***	***.2648***

Table 6.6: Sideslip Error Statistics from EMRAN Training for YF-22 Data

	FDS 002	**FDS 003**	**FDS 004**	**FDS 005**
Number of active neurons	96	77	107	91
Number of training epochs	1000	1000	1000	1000
Mean of training error	.0183	-.0345	-.0147	-.0537
Standard deviation of training error	.1764	.3628	.2542	.2018

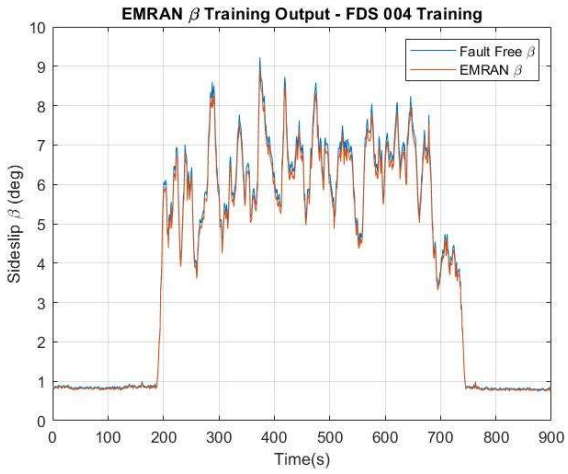

Figure 6.7: EMRAN Sideslip Accommodation vs. Fault Free Data (FDS 004 Training)

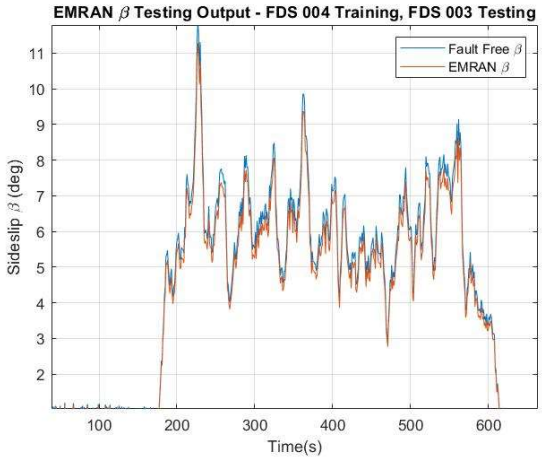

Figure 6.8: EMRAN Sideslip Accommodation vs. Fault Free Data (FDS 004 Training, FDS 003 Testing)

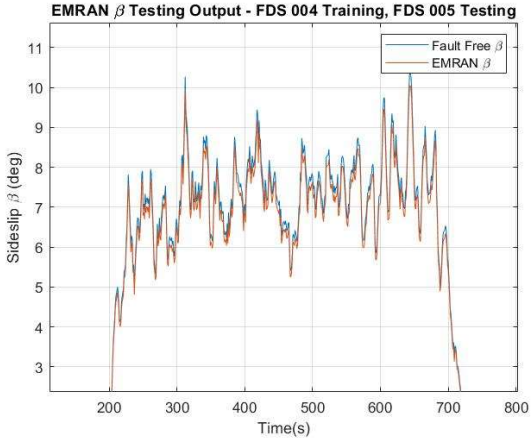

Figure 6.9: EMRAN Sideslip Accommodation vs. Fault Free Data (FDS 004 Training, FDS 005 Testing)

6.2.2 True Airspeed

Table 6.7: EMRAN True Airspeed Estimation Error Statistics for Each YF-22 Flight Data Set – Training and Testing

	FDS 002 (test)		FDS 003 (test)	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
FDS 002	**.0199**	***.2924***	.4987	.9274
FDS 003	.3770	.2975	***.0275***	***.2975***
FDS 004	.2745	.4775	.9725	.5732
FDS 005	.4907	.5310	.2075	.1975
	FDS 004 (test)		**FDS 005 (test)**	
FDS 002	.9865	.2749	2975	.6482
FDS 003	.4739	.8563	.2474	.7462
FDS 004	***.0244***	***.2841***	.1974	.3972
FDS 005	.4829	.2740	***.0937***	***.4792***

Table 6.8: True Airspeed Error Statistics from EMRAN Training for YF-22 Data

	FDS 002	FDS 003	FDS 004	FDS 005
Number of active neurons	72	78	88	90
Number of training epochs	1000	1000	1000	1000
Mean of training error	.0173	.0749	.0183	.0971
Standard deviation of training error	.9173	.2864	.8621	.4672

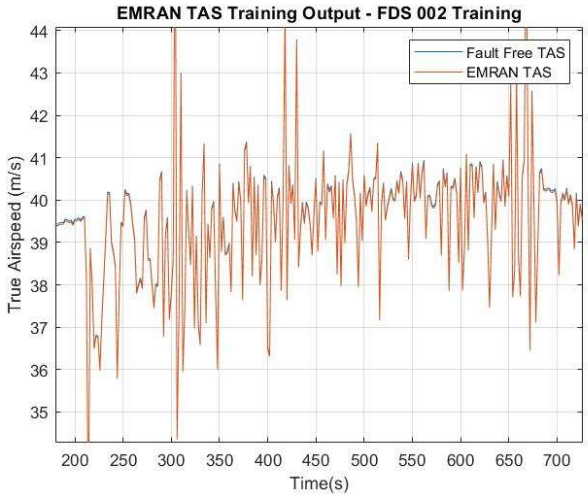

Figure 6.10: EMRAN True Airspeed Accommodation vs. Fault Free Data (FDS 002 Training)

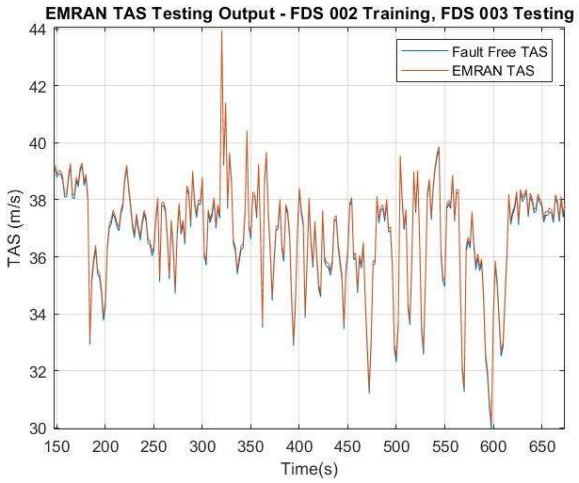

Figure 6.11: EMRAN True Airspeed Accommodation vs. Fault Free Data (FDS 002 Training, FDS 003 Testing)

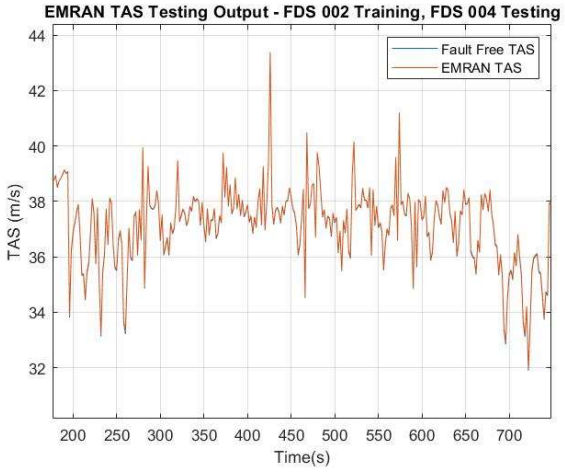

Figure 6.12: EMRAN True Airspeed Accommodation vs. Fault Free Data (FDS 002 Training, FDS 004 Testing)

6.3 EKF Results

Unlike the NN, the KFs did not require initial training. The initial matrices designated below for Q and R were applied to the EKF and UKF for each test. The following results display the accommodation of the EKF when tested with each YF-22 FDS.

$Q_{k,\beta}$

$$= diag\,[2.41 * 10^{-2} \quad 0.367 * 10^{-1} \quad 4.45 * 10^{-3} \quad .616 * 10^{-5} \quad 4.27 * 10^{-2} \quad 8.57 * 10^{-5} \quad \cdots]$$

$$[\cdots \quad 6.49 * 10^{-4} \quad 2.35 * 10^{-3} \quad 3.03 * 10^{-3}]$$

$$R_{k,\beta} = diag[1.55 * 10^{-5} \quad 1.62 * 10^{-4} \quad 5.22 * 10^{-4} \quad 8.71 * 10^{-5} \quad 7.19 * 10^{-4}]$$

$Q_{k,TAS}$

$$= diag[5.21 * 10^{-2} \quad 9.13 * 10^{-4} \quad 8.77 * 10^{-5} \quad 3.14 * 10^{-1} \quad 5.98 * 10^{-5} \quad 2.26 * 10^{-3} \quad 4.32 * 10^{-5}]$$

$$R_{k,TAS} = diag[9.33 * 10^{-4} \quad 5.11 * 10^{-5} \quad 1.49 * 10^{-6} \quad 2.43 * 10^{-5} \quad 0.916 * 10^{-1}]$$

The units within the *Q* and *R* matrices are of m/s^2 for accelerations, *deg/s* for angular rates, and *deg* for angles.

6.3.1 Sideslip Angle

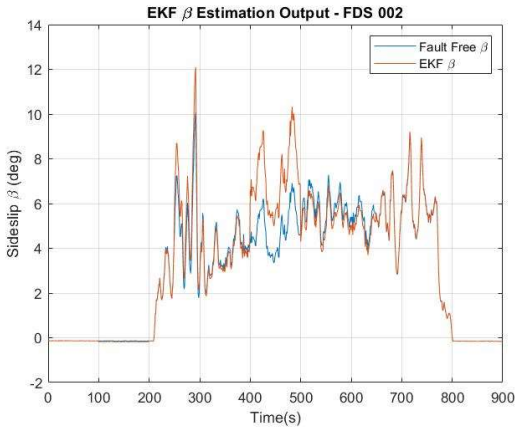

Figure 6.13: EKF Sideslip Accommodation vs. Fault Free Data (FDS 002 Testing)

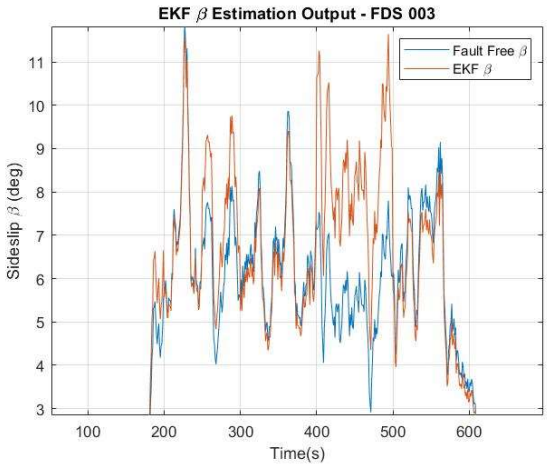

Figure 6.14: EKF Sideslip Accommodation vs. Fault Free Data (FDS 003 Testing)

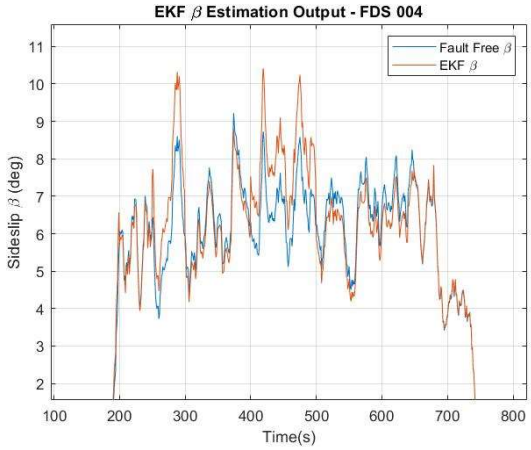

Figure 6.15: EKF Sideslip Accommodation vs. Fault Free Data (FDS 004 Testing)

6.3.2 True Airspeed

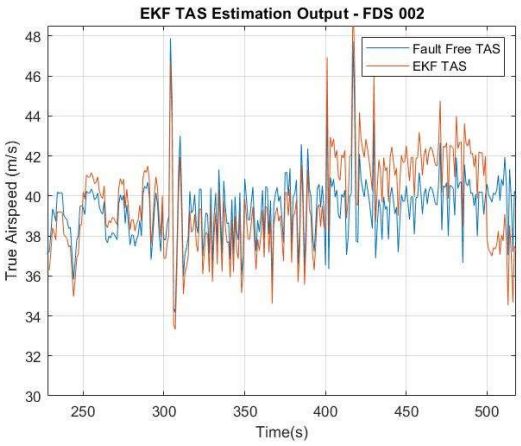

Figure 6.16: EKF True Airspeed Accommodation vs. Fault Free Data (FDS 002 Testing)

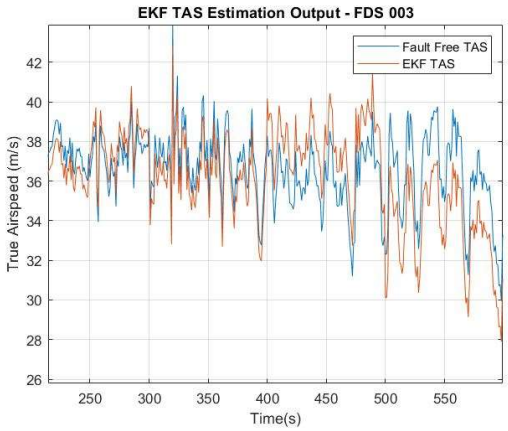

Figure 6.17: EKF True Airspeed Accommodation vs. Fault Free Data (FDS 003 Testing)

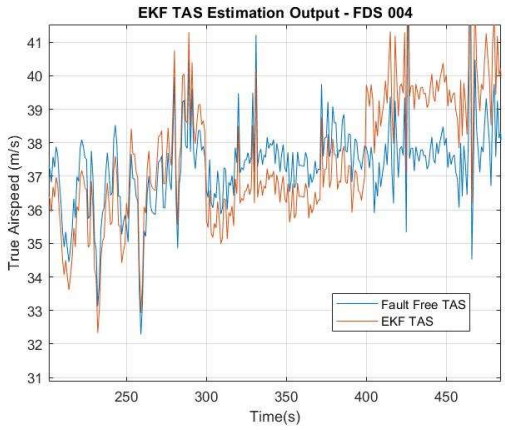

Figure 6.18: EKF True Airspeed Accommodation vs. Fault Free Data (FDS 004 Testing)

6.4 UKF Results

As with the EKF, the UKF was tested with each YF-22 flight dataset using the Q and R matrices from Section 6.3. The results are shown below.

6.4.1 Sideslip Angle

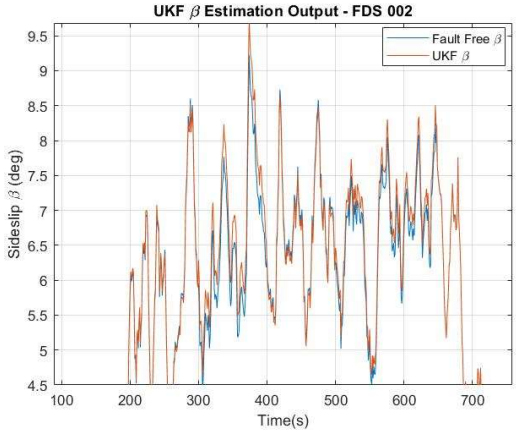

Figure 6.19: UKF Sideslip Accommodation vs. Fault Free Data (FDS 002 Testing)

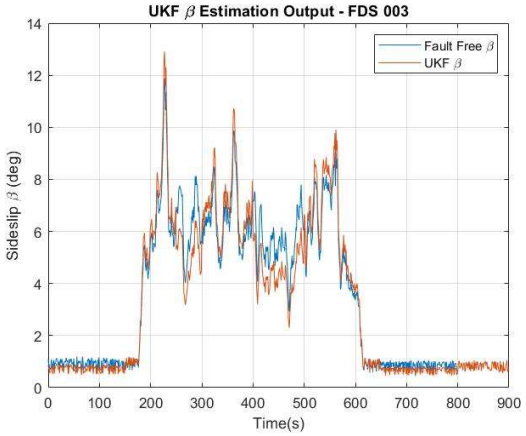

Figure 6.20: UKF Sideslip Accommodation vs. Fault Free Data (FDS 003 Testing)

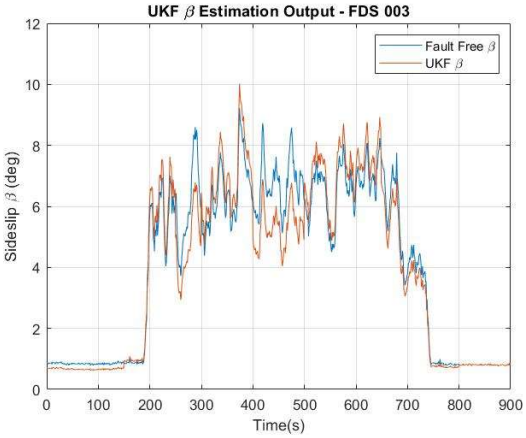

Figure 6.21: UKF Sideslip Accommodation vs. Fault Free Data (FDS 004 Testing)

6.4.2 True Airspeed

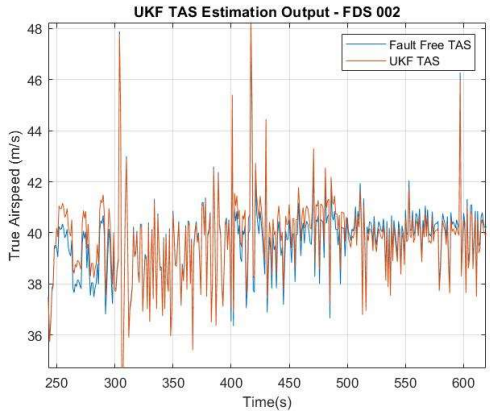

Figure 6.22: UKF True Airspeed Accommodation vs. Fault Free Data (FDS 002 Testing)

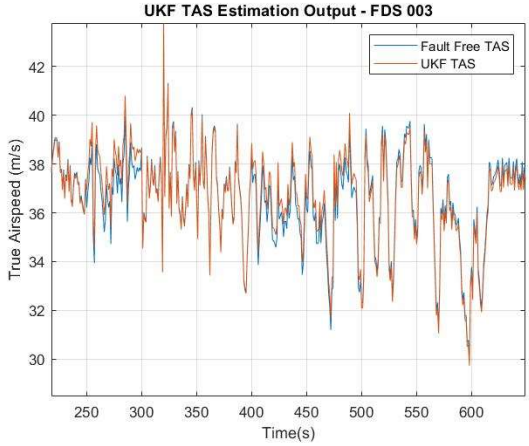

Figure 6.23: UKF True Airspeed Accommodation vs. Fault Free Data (FDS 003 Testing)

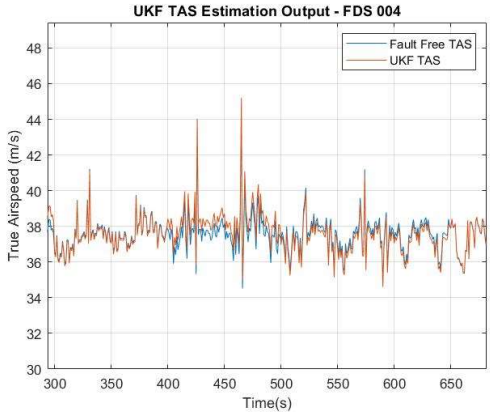

Figure 6.24: UKF True Airspeed Accommodation vs. Fault Free Data (FDS 004 Testing)

6.5 Summary of YF-22 Results

The following tables display the numerical averages for mean and standard deviation of the accommodation data for each of the SFA schemes when tested with the selected four YF-22 datasets. Means closer to zero with lower standard deviations were correlated to more accurate accommodations.

Table 6.9: Comparison of All SFA Approaches for Sideslip Angle YF-22 Data

FDS	MLP		EMRAN		EKF		UKF	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
002	-.2754	.7635	.2177	.1896	.3554	.1624	.1390	.3618
003	.2872	.0888	.2773	.2778	.2367	.3347	.6432	.8462
004	.2765	.2756	.1545	.3354	.7643	.7732	.2466	.2662
005	.4738	.3673	.4421	.3764	.7832	.6688	.4678	.6633

Table 6.10: Comparison of All SFA Approaches for True Airspeed YF-22 Data

FDS	MLP		EMRAN		EKF		UKF	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
002	.1778	.4579	.1554	.6638	.6999	.2664	.3554	.4563
003	.3478	.5552	.2664	.3648	.3727	.5577	.3527	.6672
004	.2143	.2668	.4625	.3557	.3884	.2466	.3322	.6322
005	.2277	.3628	.2779	.2442	.4662	.5789	.2999	.9466

Table 6.11: Results of YF-22 Data with Large Errors

	β	*TAS*
Estimation Error	*deg*	*m/s*
MLP	.1215	.3234
EMRAN	.3326	.2291
EKF	.6569	2.001
UKF	.5577	1.256
Fault Detection Time (s)		
MLP	.6593	.8692
EMRAN	.4928	.5879
EKF	1.479	1.374
UKF	.9764	.9924
False Alarms		
MLP	0	0
EMRAN	0	0
EKF	0	0
UKF	0	0
Undetected Faults		
MLP	0	0
EMRAN	0	0
EKF	0	0
UKF	0	0
Detectability Ratio		
MLP	103.28	123.81
EMRAN	112.95	124.46
EKF	325.99	378.13
UKF	409.65	447.97

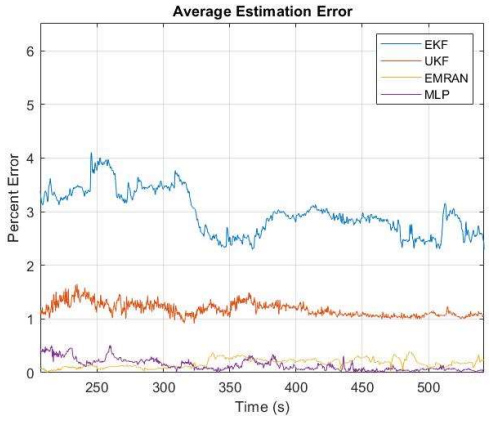

Figure 6.25: Average Estimation Error Comparison for Sideslip Angle in YF-22 FDS 002

Table 6.12: Results of YF-22 Data with Small Errors

	β	***TAS***
Estimation Error	*deg*	*m/s*
MLP	.3685	.8462
EMRAN	.3728	.5673
EKF	.9763	1.790
UKF	.4628	.8872
Fault Detection Time		
MLP	.6721	.6987
EMRAN	.5231	.7112
EKF	1.564	1.238
UKF	1.273	1.652
False Alarms		
MLP	0	0
EMRAN	0	0
EKF	1	0
UKF	0	0
Undetected Faults		
MLP	0	0
EMRAN	0	0
EKF	0	0
UKF	0	0
Detectability Ratio		
MLP	109.47	117.65
EMRAN	174.62	158.92
EKF	.009	381.09
UKF	457.12	432.77

The overall results of the YF-22 accommodation were similar to the outcome from the simulated data. All SFA schemes performed slightly better than in the simulated environment. This is most likely due to the training exposure to actual data instead of simulated data. The EMRAN again performed the best in terms of estimation. The accommodation data was nearly identical to the actual data values for every sensor failure case. The MLP, UKF, and EKF followed the EMRAN in that order, respectively. The EKF was the only SFA scheme that

produced a false alarm with the YF-22 data but the KFs in this case had much better detectability ratios when there was no false alarm.

Chapter #7: Performance Analysis Using Tecnam P92 Flight Data

The same training process of the NNs for the YF-22 was used for the Tecnam P92. Due to the higher number of flights and to maintain brevity, only 4 of the 25 flights were highlighted in these results. These four flights provide a satisfactory overview of the range of performances and different training results from the entire data collection. Flight 020 was the best flight to use overall for training followed closely by Flight 003. Flight 011 was the worst flight for training, and Flight 017 was average compared to the other 25 flights for training.

7.1 MLP Results

As seen in the following tables, SIM 020 provided the best results from the training process for angle of attack of the Tecnam P92.

7.1.1 Angle of Attack

Table 7.1: MLP Angle of Attack Estimation Error Statistics for P92 Flight Data Sets – Training and Testing

	FDS 020 (Best test)		FDS 003 (2nd Best test)	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
FDS 020	$\mathbf{.1763e^{-2}}$	***.2974***	.3322	.6355
FDS 003	.2211	.3288	$\mathbf{2.372e^{-2}}$	***.3552***
FDS 017	.2262	.3622	.3782	.2274
FDS 011	.4322	.2551	.3329	.2998
	FDS 017 (Average test)		**FDS 011 (Worst test)**	
FDS 020	.3374	.3662	.5521	.2536
FDS 003	.4425	.4566	.2718	.3547
FDS 017	$\mathbf{.9778e^{-2}}$	***.5421***	.3445	.6452
FDS 011	.1535	.7255	$\mathbf{2.472e^{-3}}$	***.2553***

Table 7.2: Angle of Attack Error Statistics from MLP Training for Tecnam P92

	FDS 020	FDS 003	FDS 017	FDS 011
Number of active neurons	78	64	68	73
Number of training epochs	1000	1000	1000	1000
Mean of training error	-.0131	.0981	.7381	.1271
Standard deviation of training error	.2635	.4673	.3627	.3488

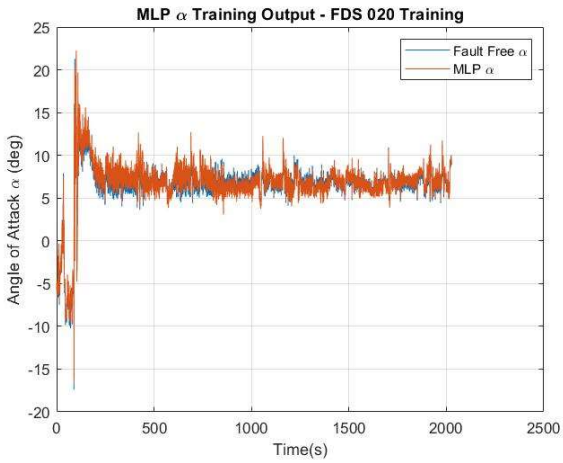

Figure 7.1: MLP Angle of Attack Accommodation vs. Fault Free Data (FDS 020 Training)

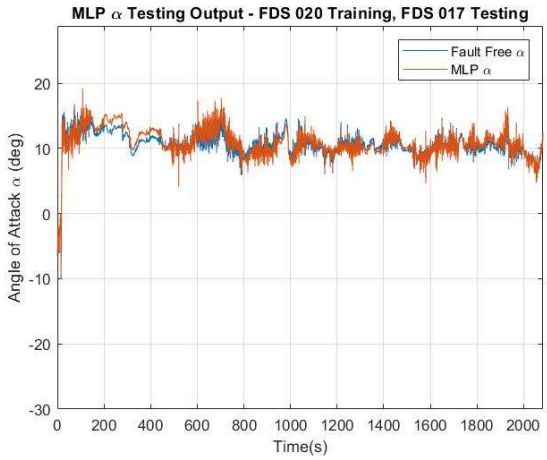

Figure 7.2: MLP Angle of Attack Accommodation vs. Fault Free Data (FDS 020 Training, FDS 017 Testing)

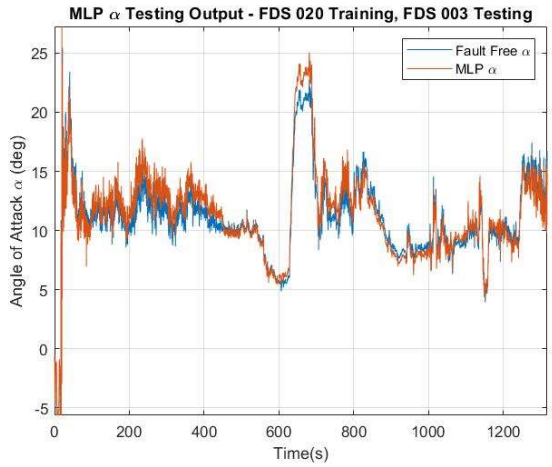

Figure 7.3: MLP Angle of Attack Accommodation vs. Fault Free Data (FDS 020 Training, FDS 003 Testing)

7.1.2 Sideslip Angle

The FDS 020 provided optimal sideslip angle training for the NNs in the Tecnam P92. There are lower means for each estimation and fewer active neurons required.

Table 7.3: MLP Sideslip Angle Estimation Error Statistics for Each P92 Flight Data Sets – Training and Testing

	FDS 020 (Best test)		**FDS 003 (2nd Best test)**	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
FDS 020	$\mathbf{1.242e^{-2}}$	***.3749***	.4872	.2974
FDS 003	.1739	.3729	$\mathbf{1.274e^{-2}}$	***.3526***
FDS 017	.2367	.3728	.2188	.3772
FDS 011	.2137	.3526	.1728	.3718
	FDS 017 (Average test)		**FDS 011 (Worst test)**	
FDS 020	.2647	.3791	.4722	.1636
FDS 003	.3288	.4658	.8262	.1772
FDS 017	$\mathbf{1.998e^{-2}}$	***.1277***	.2743	.2217
FDS 011	.5266	.3551	$\mathbf{.8879e^{-3}}$	***.1122***

Table 7.4: Sideslip Error Statistics from MLP Training for Tecnam P92

	FDS 020	**FDS 003**	**FDS 017**	**FDS 011**
Number of active neurons	71	80	93	98
Number of training epochs	1000	1000	1000	1000
Mean of training error	.2612	.3628	.4352	.2717
Standard deviation of training error	.3692	.2794	.1846	.3740

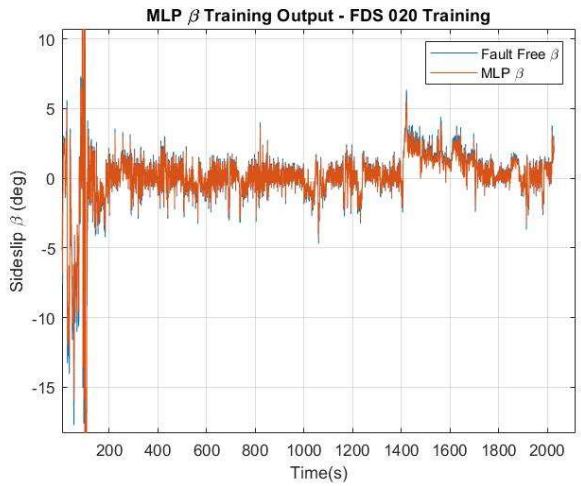

Figure 7.4: MLP Sideslip Accommodation vs. Fault Free Data (FDS 020 Training)

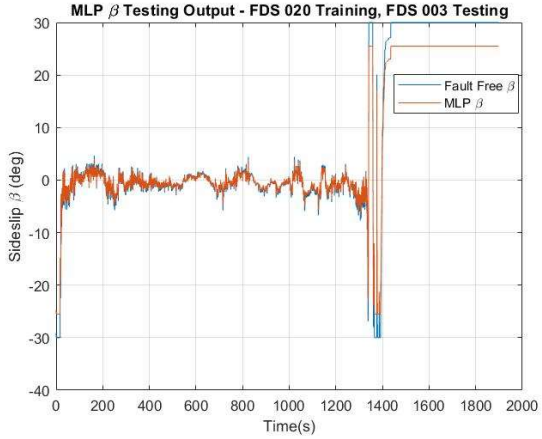

Figure 7.5: MLP Sideslip Accommodation vs. Fault Free Data (FDS 020 Training, FDS 003 Testing)

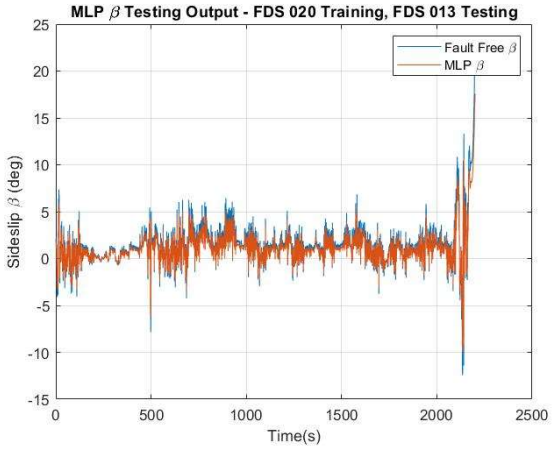

Figure 7.6: MLP Sideslip Accommodation vs. Fault Free Data (FDS 020 Training, FDS 011 Testing)

7.1.3 True Airspeed

Lastly, the FDS 020 was used as training for true airspeed for its low mean and standard deviation values.

Table 7.5: MLP True Airspeed Estimation Error Statistics for Each P92 Flight Data Set – Training and Testing

	FDS 020 (Best test)		**FDS 003 (2nd Best test)**	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
FDS 020	$\mathbf{.276e^{-3}}$	***.2466***	.2633	.2778
FDS 003	.2271	.3740	$\mathbf{2.373e^{-2}}$	***.2647***
FDS 017	.4772	.3889	.9927	.2743
FDS 011	.2738	.2664	.3772	.4738
	FDS 017 (Average test)		**FDS 011 (Worst test)**	
FDS 020	.2477	.3622	.2744	.3772
FDS 003	.1779	.2883	.4632	.2839
FDS 017	$\mathbf{1.778e^{-3}}$	***.2861***	.2773	.4665
FDS 011	.2747	.3739	$\mathbf{4.377e^{-3}}$	***.2864***

Table 7.6: True Airspeed Error Statistics from MLP Training for Tecnam P92

	FDS 020	**FDS 003**	**FDS 017**	**FDS 011**
Number of active neurons	76	87	66	100
Number of training epochs	1000	1000	1000	1000
Mean of training error	-.0228	.0182	.0283	.0338
Standard deviation of training error	.1828	.2974	.2858	.2283

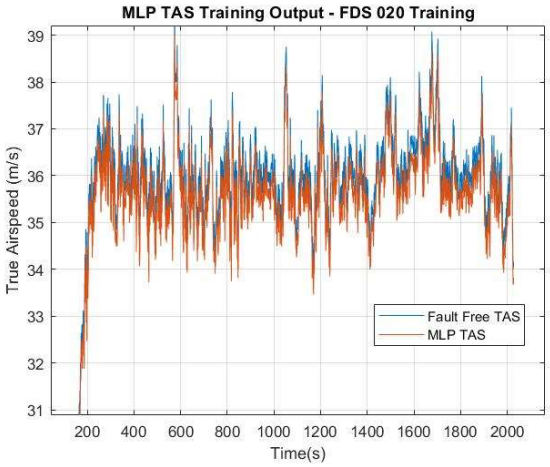

Figure 7.7: MLP True Airspeed Accommodation vs. Fault Free Data (FDS 020 Training)

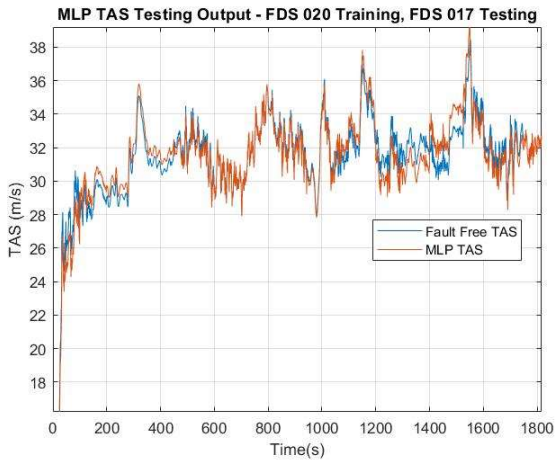

Figure 7.8: MLP True Airspeed Accommodation vs. Fault Free Data (FDS 020 Training, FDS 017 Testing)

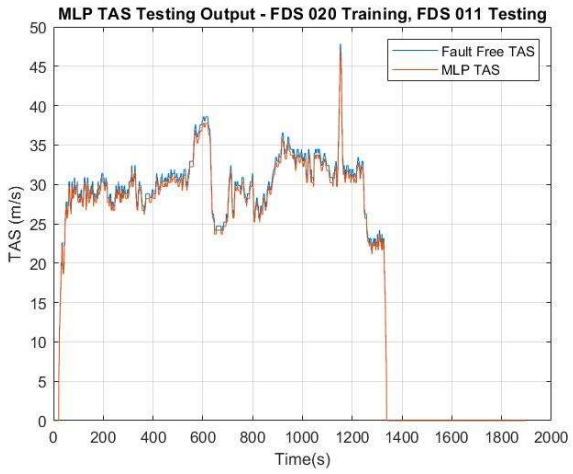

Figure 7.9: MLP True Airspeed Accommodation vs. Fault Free Data (FDS 020 Training, FDS 011 Testing)

7.2 EMRAN Results

The EMRAN NN used the same flight dataset FDS 020 for angle of attack, sideslip angle, and true airspeed. The following tables and figures display the efficacy of utilizing FDS 020 as the Tecnam P92 training data for the EMRAN.

7.2.1 Angle of Attack

Table 7.7: EMRAN Angle of Attack Estimation Error Statistics for Each P92 Flight Data Sets – Training and Testing

	FDS 020 (Best test)		FDS 003 (2nd Best test)	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
FDS 020	***.0193***	***.3974***	.2898	.8361
FDS 003	.1836	.2746	***.0264***	***.2746***
FDS 017	.2866	.4775	.9277	.2748
FDS 011	.2332	.2974	.2976	.2299
	FDS 017 (Average test)		**FDS 011 (Worst test)**	
FDS 020	.2973	.3888	.4665	.3874
FDS 003	.2974	.2249	.2018	.2973
FDS 017	***.0329***	***.2884***	.2081	.2974
FDS 011	.3792	.2947	***.0281***	***.2974***

Table 7.8: Angle of Attack Error Statistics from EMRAN Training for Tecnam P92

	FDS 020	FDS 003	FDS 017	FDS 011
Number of active neurons	83	84	99	75
Number of training epochs	1000	1000	1000	1000
Mean of training error	.0143	-.0196	.0263	.0366
Standard deviation of training error	.1964	.2874	.2844	.1747

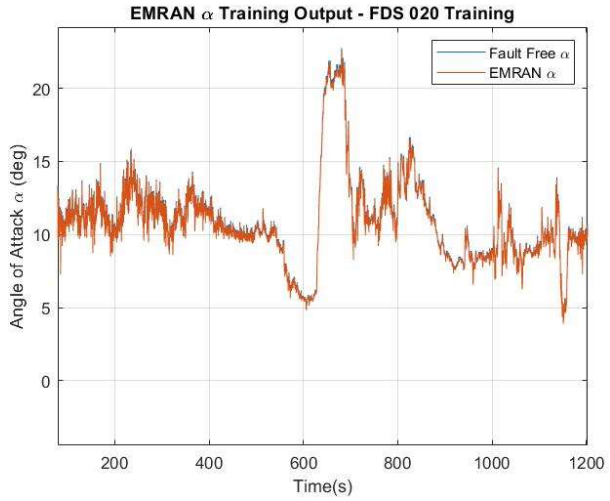

Figure 7.10: EMRAN Angle of Attack Accommodation vs. Fault Free Data (FDS 020 Training)

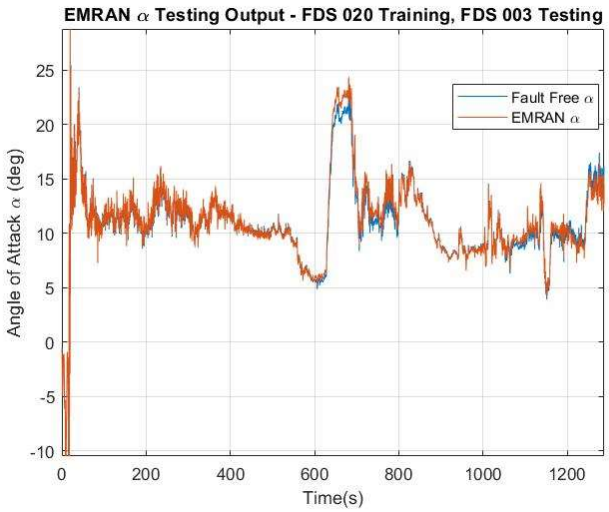

Figure 7.11: EMRAN Angle of Attack Accommodation vs. Fault Free Data (FDS 020 Training, FDS 003 Testing)

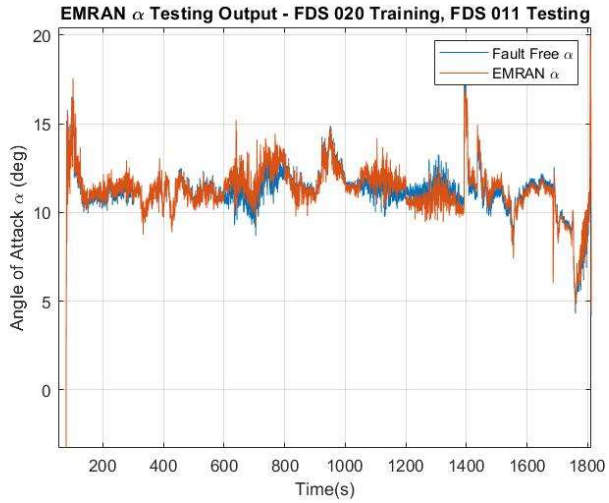

Figure 7.12: EMRAN Angle of Attack Accommodation vs. Fault Free Data (FDS 020 Training, FDS 011 Testing)

7.2.2 Sideslip Angle

Table 7.9: EMRAN Sideslip Angle Estimation Error Statistics for Each P92 Flight Data Sets – Training and Testing

	FDS 020 (Best test)		FDS 003 (2nd Best test)	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
FDS 020	***-.0186***	***.1232***	.9733	.2933
FDS 003	.3872	.3244	***.0274***	***.2874***
FDS 017	.2873	.0373	.2732	.2864
FDS 011	.2983	.7392	.2974	.9272
	FDS 017 (Average test)		**FDS 011 (Worst test)**	
FDS 020	.3732	.3748	.2863	.4637
FDS 003	.2973	.2747	.3746	.2748
FDS 017	***.0221***	***.2974***	.2634	.2747
FDS 011	.3777	.2986	***-.0488***	***.1033***

Table 7.10: Sideslip Error Statistics from EMRAN Training for Tecnam P92

	FDS 020	FDS 003	FDS 017	FDS 011
Number of active neurons	103	84	90	87
Number of training epochs	1000	1000	1000	1000
Mean of training error	-.0237	.0583	.0374	-.0336
Standard deviation of training error	.1228	.1473	.1863	.2862

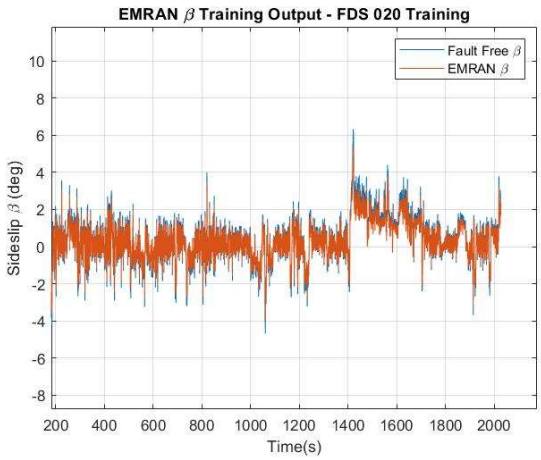

Figure 7.13: EMRAN Sideslip Accommodation vs. Fault Free Data (FDS 020 Training)

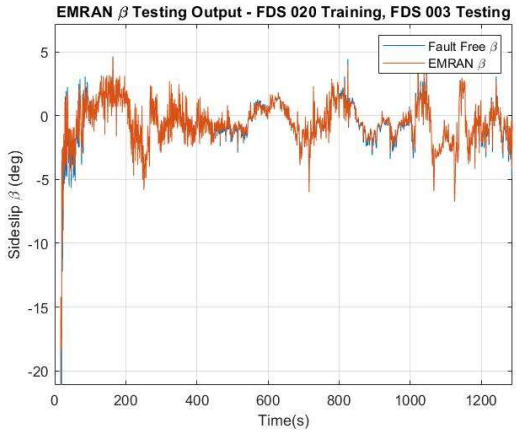

Figure 7.14: EMRAN Sideslip Accommodation vs. Fault Free Data (FDS 020 Training, FDS 003 Testing)

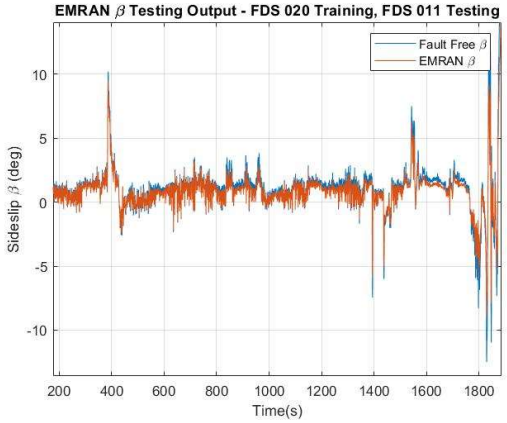

Figure 7.15: EMRAN Sideslip Accommodation vs. Fault Free Data (FDS 020 Training, FDS 011 Testing)

7.2.3 True Airspeed

Table 7.11: EMRAN True Airspeed Estimation Error Statistics for Each P92 Flight Data Sets – Training and Testing

	FDS 020 (Best test)		FDS 003 (2nd Best test)	
Training FDS	**Mean (m/s)**	**SD (m/s)**	**Mean (m/s)**	**SD (m/s)**
FDS 020	***.0371***	***.2801***	.4972	.1397
FDS 003	.5221	.4786	***.0711***	***.4961***
FDS 017	.2343	.9889	.2864	.2974
FDS 011	.2683	.3590	.2975	.2335
	FDS 017 (Average test)		**FDS 011 (Worst test)**	
FDS 020	.9735	.3975	.3477	.7717
FDS 003	.4442	.6904	.4623	.9623
FDS 017	**.0997**	***.2578***	.3722	.1226
FDS 011	.2835	.0749	***.0516***	***.1064***

Table 7.12: True Airspeed Error Statistics from EMRAN Training for Tecnam P92

	FDS 020	FDS 003	FDS 017	FDS 011
Number of active neurons	67	69	73	87
Number of training epochs	1000	1000	1000	1000
Mean of training error	-.0231	-.0345	-.0776	-.0655
Standard deviation of training error	.1749	.1863	.2719	.2962

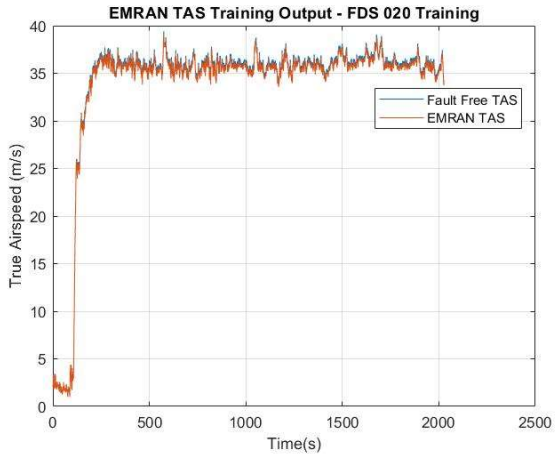

Figure 7.16: EMRAN True Airspeed Accommodation vs. Fault Free Data (FDS 020 Training)

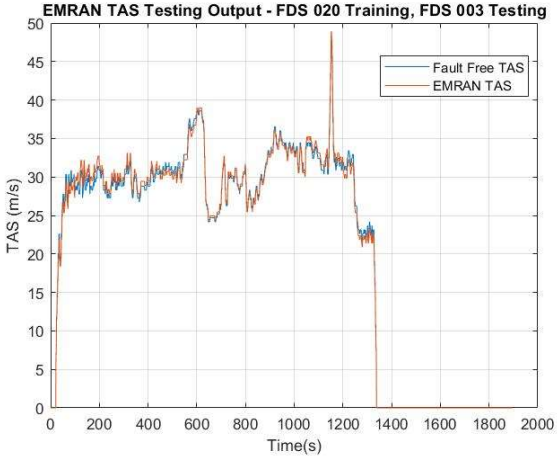

Figure 7.17: EMRAN True Airspeed Accommodation vs. Fault Free Data (FDS 020 Training, FDS 003 Testing)

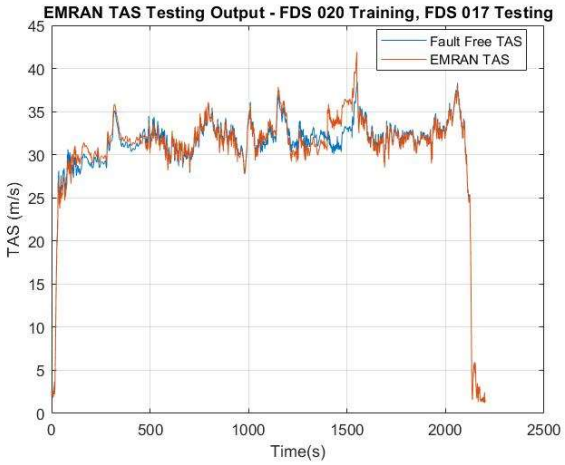

Figure 7.18: EMRAN True Airspeed Accommodation vs. Fault Free Data (FDS 020 Training, FDS 017 Testing)

7.3 EKF Results

The KFs were tested in the same way as for the simulated flight datasets and the YF-22 datasets. Since the KFs did not require training, they were each tested directly with each dataset for all sensor failure types.

The Q and R matrices used for the Tecnam P92 data with the EKF and UKF are as follows.

$$Q_{k,\alpha} = diag[1.34 * 10^{-3} \quad 9.09 * 10^{-4} \quad 5.43 * 10^{-5} \quad 7.78 * 10^{-3} \quad 4.73 * 10^{-5} \quad 7.64 * 10^{-5} \quad \cdots]$$

$$[\cdots \quad 3.55 * 10^{-4} \quad 6.80 * 10^{-4} \quad 4.40 * 10^{-2}]$$

$$R_{k,\alpha} = diag[4.81 * 10^{-4} \quad 6.91 * 10^{-2} \quad 7.55 * 10^{-4} \quad 4.55 * 10^{-3} \quad 5.69 * 10^{-2}]$$

$$Q_{k,\beta} = diag\,[8.22 * 10^{-2} \quad 3.18 * 10^{-4} \quad 4.21 * 10^{-2} \quad 4.09 * 10^{-6} \quad 7.13 * 10^{-2} \quad 2.97 * 10^{-3} \quad \cdots]$$

$$[\cdots \quad 6.96 * 10^{-4} \quad 6.18 * 10^{-2} \quad 1.11 * 10^{-3}]$$

$$R_{k,\beta} = diag[1.16 * 10^{-3} \quad 1.87 * 10^{-3} \quad 6.71 * 10^{-3} \quad 8.96 * 10^{-3} \quad 3.17 * 10^{-4}]$$

$Q_{k,TAS}$

$$= diag[9.41 * 10^{-5} \quad 5.65 * 10^{-4} \quad 8.56 * 10^{-4} \quad 5.02 * 10^{-6} \quad 3.98 * 10^{-4} \quad 4.53 * 10^{-3} \quad 3.88 * 10^{-4}]$$

$$R_{k,TAS} = diag[3.75 * 10^{-3} \quad 1.86 * 10^{-3} \quad 4.10 * 10^{-4} \quad 2.15 * 10^{-4} \quad 6.78 * 10^{-2}]$$

The units within the Q and R matrices are of m/s^2 for accelerations, *deg/s* for angular rates, and *deg* for angles.

7.3.1 Angle of Attack

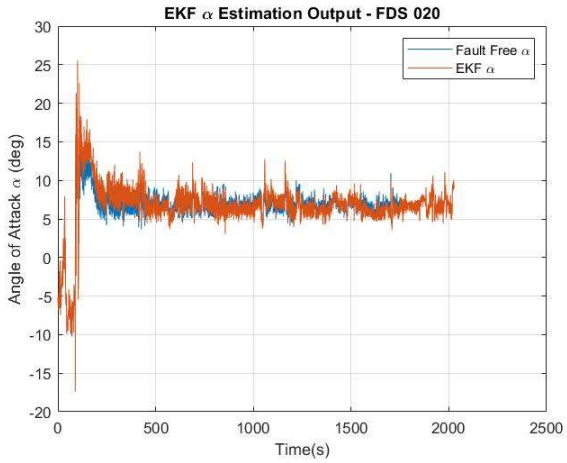

Figure 7.19: EKF Angle of Attack Accommodation vs. Fault Free Data (FDS 020 Testing)

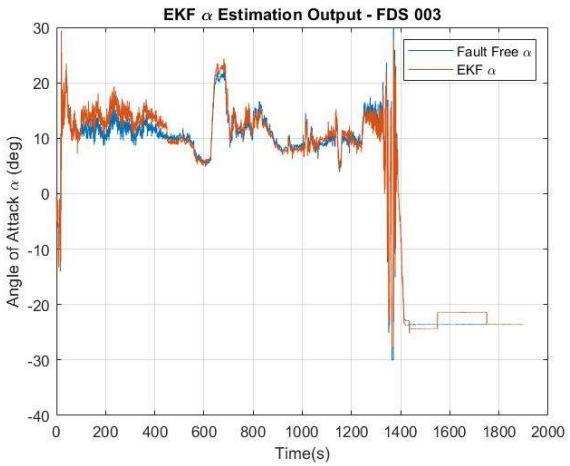

Figure 7.20: EKF Angle of Attack Accommodation vs. Fault Free Data (FDS 003 Testing)

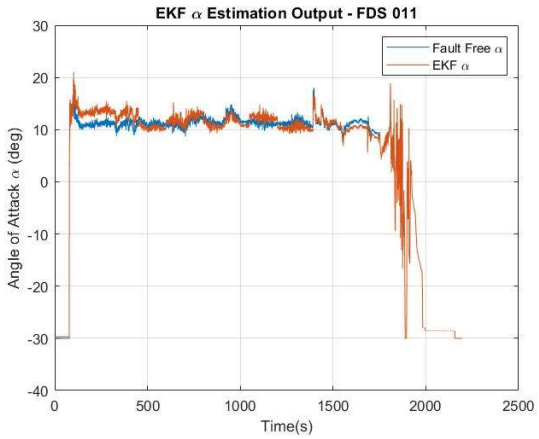

Figure 7.21: EKF Angle of Attack Accommodation vs. Fault Free Data (FDS 011 Testing)

7.3.2 Sideslip Angle

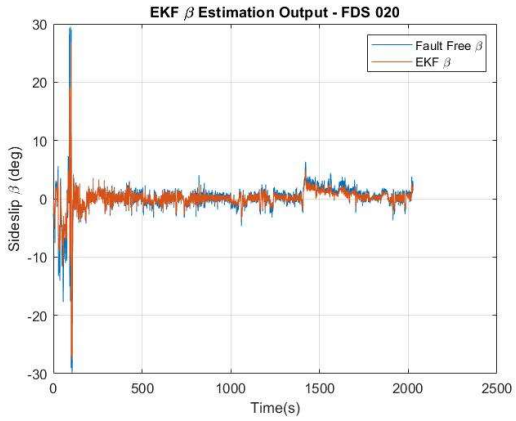

Figure 7.22: EKF Sideslip Accommodation vs. Fault Free Data (FDS 020 Testing)

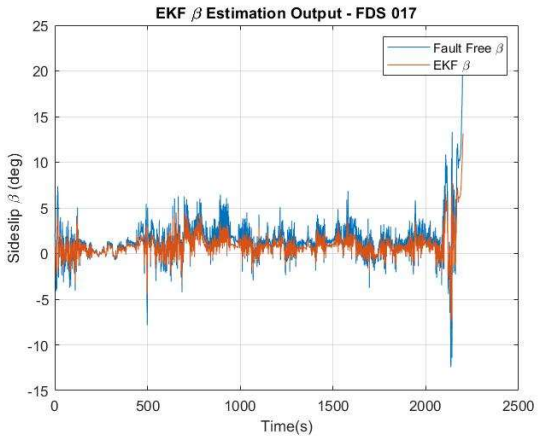

Figure 7.23: EKF Sideslip Accommodation vs. Fault Free Data (FDS 017 Testing)

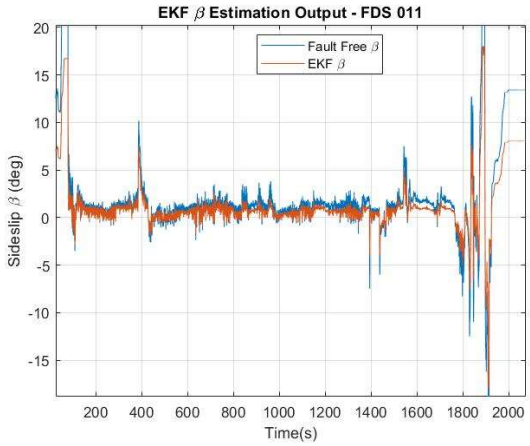

Figure 7.24: EKF Sideslip Accommodation vs. Fault Free Data (FDS 011 Testing)

7.3.3 True Airspeed

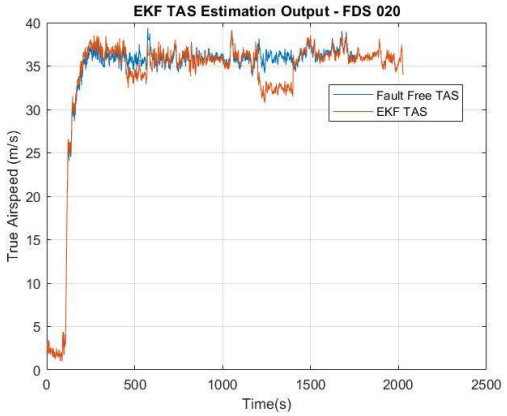

Figure 7.25: EKF True Airspeed Accommodation vs. Fault Free Data (FDS 020 Testing)

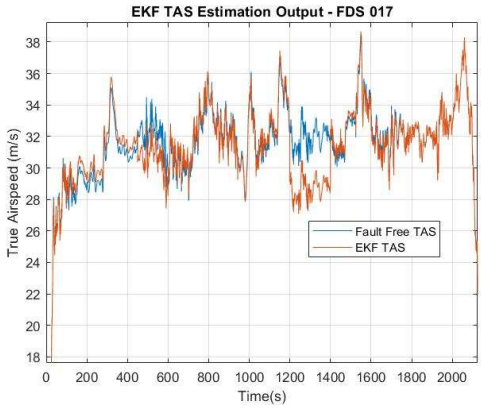

Figure 7.26: EKF True Airspeed Accommodation vs. Fault Free Data (FDS 017 Testing)

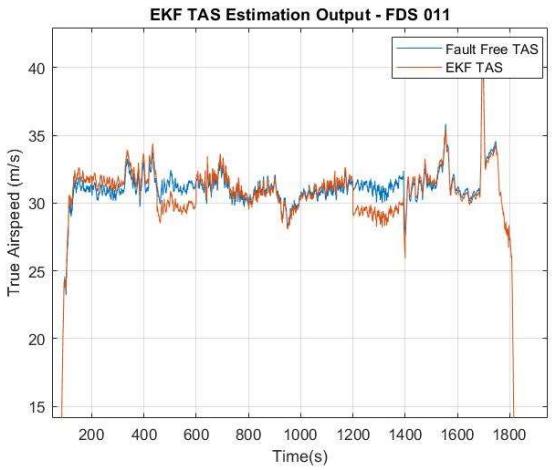

Figure 7.27: EKF True Airspeed Accommodation vs. Fault Free Data (FDS 011 Testing)

7.4 UKF Results

The UKF was tested with each FDS like the EKF. No training was required so the UKF was tested as soon as the system of equations and covariance matrices from Chapter 4 were implemented.

7.4.1 Angle of Attack

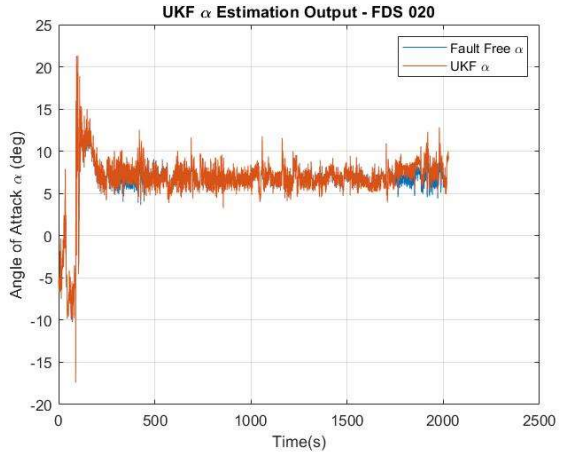

Figure 7.28: UKF Angle of Attack Accommodation vs. Fault Free Data (FDS 020 Testing)

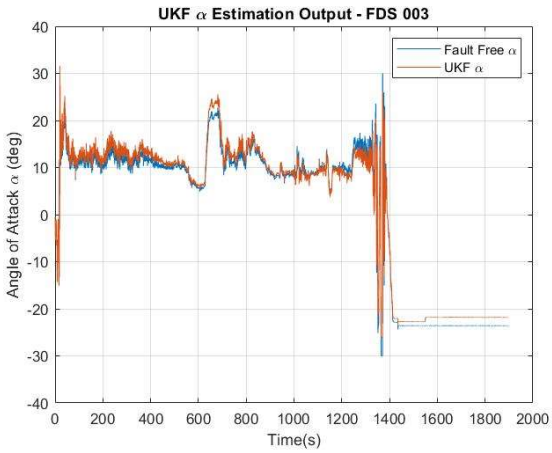

Figure 7.29: UKF Angle of Attack Accommodation vs. Fault Free Data (FDS 003 Testing)

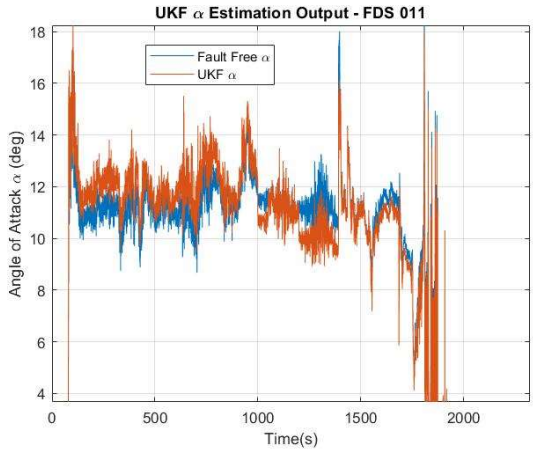

Figure 7.30: UKF Angle of Attack Accommodation vs. Fault Free Data (FDS 011 Testing)

7.4.2 Sideslip Angle

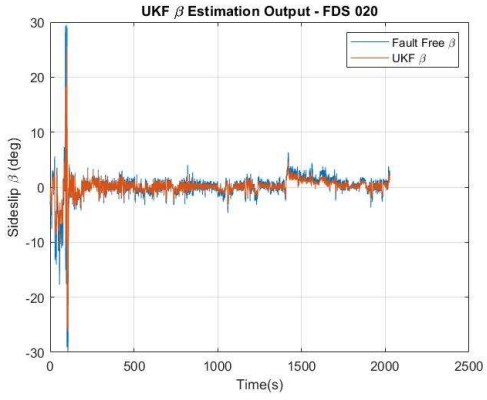

Figure 7.31: UKF Sideslip Accommodation vs. Fault Free Data (FDS 020 Testing)

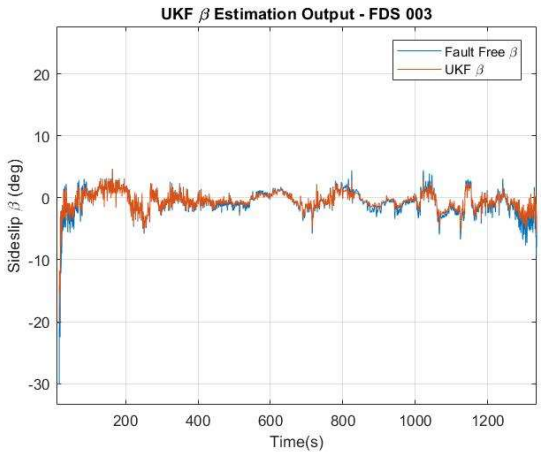

Figure 7.32: UKF Sideslip Accommodation vs. Fault Free Data (FDS 003 Testing)

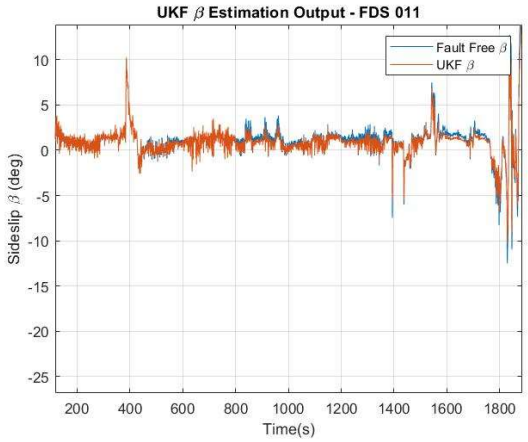

Figure 7.33: UKF Sideslip Accommodation vs. Fault Free Data (FDS 011 Testing)

7.4.3 True Airspeed

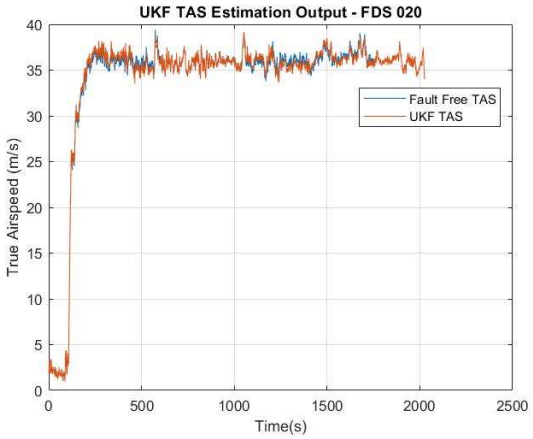

Figure 7.34: UKF True Airspeed Accommodation vs. Fault Free Data (FDS 020 Testing)

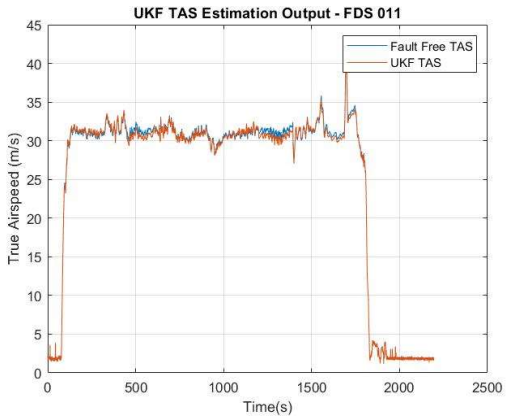

Figure 7.35: UKF True Airspeed Accommodation vs. Fault Free Data (FDS 011 Testing)

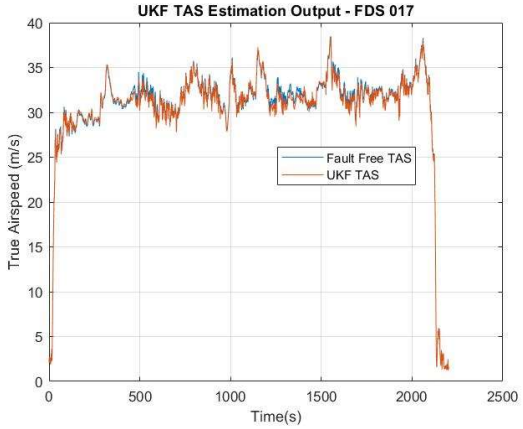

Figure 7.36: UKF True Airspeed Accommodation vs. Fault Free Data (FDS 017 Testing)

7.5 Summary of Tecnam P92 Results

Table 7.13: Comparison of All SFA Approaches for Angle of Attack P92 Data

FDS	MLP		EMRAN		EKF		UKF	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
020	.0274	.2793	.0197	.3864	.2498	.7423	.1836	.2896
003	.0399	.2497	-.0244	.3975	.3728	.8338	.3628	.4728
017	.0548	.2757	.0381	.4729	.8374	.4783	.1264	.6482
011	-.0638	.4628	-.0183	.5638	.6388	.4649	.2378	.2864

Table 7.14: Comparison of All SFA Approaches for Sideslip Angle P92 Data

FDS	MLP		EMRAN		EKF		UKF	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
020	-.0343	.1336	.0135	.3432	.6532	.4363	.3532	.3499
003	.0357	.3628	.0272	.3648	.5672	.8352	.2836	.4566
017	-.0233	.3378	-.0245	.1273	.3627	.5372	.2718	.4332
011	.0372	.3622	.0199	.1375	.5632	.3722	.3625	.4632

Table 7.15: Comparison of All SFA Approaches for Sideslip Angle P92 Data

FDS	MLP		EMRAN		EKF		UKF	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
020	.0367	.2754	.0272	.2763	.3648	.2874	.1863	.2355
003	.0285	.2874	-.0114	.1737	.2648	.4763	.2637	.4334
017	-.0437	.3695	.0174	.1564	.3784	.3762	.2833	.5546
011	.0324	.3645	.0193	.2837	.4373	.2864	.2846	.5738

The following tables provide a numerical summary of the performance from each SFA scheme for all 25 Tecnam P92 flight datasets.

Table 7.16: Results of P92 Data with Large Errors

	α	***β***	***TAS***
Estimation Error	*deg*	*deg*	*m/s*
MLP	.2297	.1214	.8722
EMRAN	.1073	.2225	.7829
EKF	.7290	1.027	1.758
UKF	.6356	.8720	1.284
Fault Detection Time *(s)*			
MLP	.286	.372	.286
EMRAN	.197	.297	.174
EKF	.370	.450	.445
UKF	.302	.411	.392
False Alarms			
MLP	0	0	0
EMRAN	0	0	0
EKF	1	0	0
UKF	1	0	0
Undetected Faults			
MLP	0	0	0
EMRAN	0	0	0
EKF	0	0	0
UKF	0	0	0
Detectability Ratio			
MLP	122.65	194.27	177.27
EMRAN	209.85	239.27	256.83
EKF	.19	271.92	287.92
UKF	.72	298.56	366.23

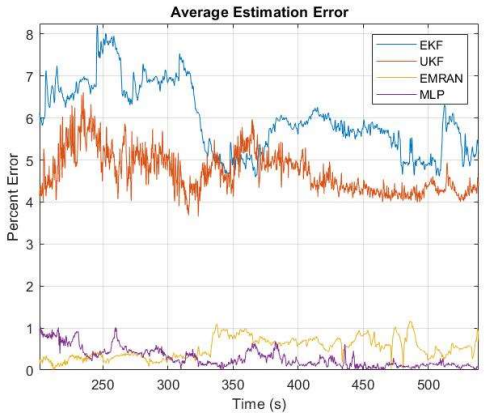

Figure 7.37: Estimation Error Comparison for Tecnam P92 True Airspeed

Table 7.17: Results of P92 Data with Small Errors

	α	β	*TAS*
Estimation Error	*deg*	*deg*	*m/s*
MLP	.3629	.2341	.5875
EMRAN	.3528	.2618	.7830
EKF	.6492	.3749	1.479
UKF	.5803	.3556	1.003
Fault Detection Time *(s)*			
MLP	.382	.332	.462
EMRAN	.352	.322	.401
EKF	.472	.649	.538
UKF	.468	.566	.578
False Alarms			
MLP	0	0	0
EMRAN	0	0	0
EKF	0	0	1
UKF	0	0	0
Undetected Faults			
MLP	0	0	0
EMRAN	0	0	0
EKF	0	0	2
UKF	0	1	0
Detectability Ratio			
MLP	286.12	253.18	267.77
EMRAN	375.48	471.46	401.26
EKF	362.44	376.35	.20
UKF	476.22	415.91	389.63

In the case of the Tecnam P92 testing, the MLP performed slightly better than the rest of the SFA schemes. The EMRAN was able to accommodate the angle of attack more accurately but the MLP provided more accurate accommodations for sideslip angle and true airspeed. The UKF performed best with the Tecnam P92 data compared to its performance with the simulated and YF-22 data testing; however, the UKF still did not perform as well as the NN. The UKF had one undetected fault within the failed Tecnam P92 datasets. The EKF again performed the worst of all

SFA schemes in terms of estimation error and false alarms but there was a high detectability ratio which is beneficial for SFA.

Chapter #8: Conclusions, Recommendations, and Future Work

After testing each SFA scheme on simulated data and real aircraft data from the YF-22 and Tecnam P92, it was observed that all four on-line estimation techniques provided satisfactory results in terms of accurate estimations, low number of false alarms, and performance. Overall, the NNs performed more accurately and faster than the KFs with fewer false alarms.

The evidence from Chapters 5-7 show that the best on-line estimator in this work was the EMRAN NN. There were multiple criteria taken into consideration to determine the best performance of all the SFA schemes. The EMRAN in most of the cases provided the most accurate accommodation data for the sensor failure. Not only was the data more accurate, but the EMRAN was able to detect failures more quickly compared to the other SFA schemes. Based on the figures within the results, the analytical redundancy method for the sensor failure problem is more than sufficient in terms of detecting and accommodating failed data. The MLP NN was a close second in performance after the EMRAN. The MLP did not detect faults as quickly as the EMRAN but provided other benefits outside of the criteria such as requiring fewer active hidden neurons in some cases. Compared to the EMRAN, the MLP required more time to train and develop the appropriate number of hidden layers and neurons. Neither the MLP nor EMRAN NN triggered false alarms nor failed to detect a sensor failure.

The UKF performed well considering its limitations such as not having training and complete dependency on a system of equations provided by the user. The UKF was able to detect over 97% of the failures, both large and small, within the given datasets. Although it did not perform as well as the NNs, the UKF could be implemented in an analytical redundancy scheme as a more cost-effective option for sensor failure problems lacking preexisting data. The UKF

also provides the benefit to users who may not have powerful computers to process large amounts of data. The EKF performed the most poorly out of all the SFA schemes. The EKF would be a much stronger contender for linear control systems. It can adapt decently to nonlinear systems but there are still improvements that can be made to the algorithm. The EKF was notably very sensitive to incorrect or inaccurate initial inputs; any incorrect or inaccurate initial estimate of the state caused the system to diverge quickly due to its linearization, resulting in a failed accommodation. On the other hand, both the EKF and UKF were more likely to detect faults as this sensitivity improved their detectability ratio.

It should be noted that although the NNs performed better overall, there were multiple limitations in the work that decreased the performance of the KFs. The KFs were heavily dependent on the tuning and tuning processes implemented by the user whereas the NNs were able to train themselves. The KFs were also not exposed to the entire dataset like the NNs which limits the accuracy the KF can estimate the unknown value. Methods to improve the bias within comparative analysis between the NNs and KFs should be explored in future work.

8.1 Recommendations

It is recommended to investigate more types of AI for SFA problems. Although the four schemes tested in this work are sufficient for the SFA problem, AI schemes are improved each day; there are potentially many more untested models that could be better suited for the SFA problem. It would also be recommended to have or gain access to more flight datasets. Providing more training opportunities to the NNs can drastically improve performance in terms of estimation and further reducing risk of false alarms and undetected failures.

8.2 Future Work

There are many directions this research can take regarding future work. An interesting topic to research would be testing the same sensor failures but on a wider variety of aircraft. Each aircraft is unique in the sense of what range of values are acceptable for certain variables. This work should also evolve to detect and accommodate multiple failures at once within the same dataset. It would also be beneficial to develop a form of AI-SFA scheme that can accommodate different sensors without any adjustment or input from a user. It is hopeful that analytical redundancy can be tested in flight in real time, but it is unlikely this will be permitted in the near future.

References

[1] Samy, I., Postlethwaite, I. Gu, D.W. A comparative study of NN- and EKF-based SFDA schemes with application to a nonlinear UAV model, International Journal of Control, 83:5, 1025-1043, 2010. DOI: 10.1080/00207170903552059.

[2] Fravolini, M.L., Brunori, V., Campa, G., Napolitano, M.R., and La Cava, M. Structural analysis approach for the generation of structured residuals for aircraft FDI. IEEE Transactions on Aerospace and Electronic Systems, Vol. 45, 4 (Oct. 2009), 1466–1482.

[3] Gururajan, S., Rhudy, M., Fravolini, M.L., Chao, H., Napolitano, M.R. Failure Detection and Accommodation Approaches for the Airspeed Sensor on a Small UAV. Mediterranean Conference on Control and Automation. Platanias-Chania, Crete, Greece. June 2013. DOI:10.1109/MED.2013.6608784.

[4] Chen, X., Hazan, E. Edited by Mikhail Belkin, Samory Kpotufe. Black-Box Control for Linear Dynamical Systems. Conference on Learning Theory, (COLT) 2021, 15-19 August 2021, Boulder, Colorado, USA. Proceedings of Machine Learning Research, vol. 134, pp. 1114-1143. PMLR, 2021.

[5] Imai, S., Blasch, E., Galli, A., Zhu, W., Lee, F., Varela, C.A. Airplane flight safety using error-tolerant data stream processing. In IEEE Aerospace and Electronic Systems Magazine, vol. 32, no. 4, pp. 4-17, April 2017, doi: 10.1109/MAES.2017.150242.

[6] United States. Federal Aviation Administration. (2019). Boeing 737-Max Crash Reports. U.S. Dept. of Transportation, Federal Aviation Administration.

[7] Jacobson, S. Aircraft loss of control causal factors and mitigation challenges. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, 2010.

[8] Li, Y., Sundararajan, N., Saratchandran, P. Analysis of minimal radial basis function network algorithm for real-time identification of nonlinear dynamic systems. IEE Proceedings-Control Theory and Applications. Vol. 147, Issue 4, pp. 476-484. IET Digital Library, 2000.

[9] Lauritzen, S.L. Time series analysis in 1880. A discussion of contributions made by T.N. Thiele. International Statistical Review. 49 (3): 319–331. doi:10.2307/1402616. JSTOR 1402616. (Dec. 1981).

[10] Wan, E.A., van der Merwe, R. The Unscented Kalman Filter for Nonlinear Estimation. Oregon Graduate Institute of Science & Technology. 2000.

[11] Tecnam/Monica Castellani VFR Aviation. 2020. Flying Magazine. Digital image. Accessed Oct. 19, 2021. https://www.flyingmag.com/story/aircraft/tecnam-p92-echo-mkii-enters-new-era/.

[12] United States. Federal Aviation Administration. (2008). Reliability, Mantainability, and Availability Handbook. U.S. Dept. of Transportation, Federal Aviation Administration.

[13] Dunbar, B. State Estimation. NASA Ames Technology Capabilities and Facilities. nasa.gov. March, 29, 2008.

[14] Alexander, R., Campani, G., Dinh, S. Lima, F.V. Challenges and Opportunities on Nonlinear State Estimation of Chemical and Biochemical Processes. Processes 2020, 8(11), 1462; https://doi.org/10.3390/pr8111462.

[15] Kolmogorov, A.N., Interpolation and Extrapolation of Stationary Sequences. Ser. Math. 1941, 5, 3–14.

[16] Wiener, N. Extrapolation, Interpolation, and Smoothing of Stationary Time Series with Engineering Application; MIT Press: Cambridge, MA, USA, 1964.

[17] Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems. J. Basic Eng. 1960, 82, 35–45.

[18] Isserman, R., Balle, P. Trends in the application of model-based fault detection and diagnosis of technical processes, Control. Eng. Practice, 5(5): 709-719.

[19] Cybenko, G. Approximation by superpositions of a sigmoidal function. Math. Control Signal Systems 2, 303–314 (1989). https://doi.org/10.1007/BF02551274.

[20] Hornik, K., Stinchcombe, M. White, H. Multilayer feedforward networks are universal approximators, Neural Networks, Volume 2, Issue 5, 1989, Pages 359-366, ISSN 0893-6080, https://doi.org/10.1016/0893-6080(89)90020-8.

[21] Papa, U., Fravolini, M., Napolitano, M.R., Del Core, G. (2018) Experimental interval models for the robust Fault Detection of Aircraft Air Data Sensors. Control Engineering Practice. 78. 196-212. 10.1016/j.conengprac.2018.07.002.

[22] Campa, G., Fravolini, M.L., Napolitano, M.R. (2001). Extended Minimal Resource Allocating Neural Networks For Aircraft SFDIA.

[23] Napolitano, M.R., "Development of Formation Flight Control Algorithms Using 3 YF-22 Flying Models," AFOSR Final Report, AFOSR Grant F49620-01-1-0373, April 2005.

[24] Rhudy, M.B., Fravolini, M.L., Porcacchia, M., Napolitano, M.R. Comparison of wind speed models within a Pitot-free airspeed estimation algorithm using light aviation data. Aerospace Science and Technology, Volume 86, 2019, Pages 21-29, ISSN 1270-9638, https://doi.org/10.1016/j.ast.2018.12.028.

[25] Julier, S.J., Uhlmann, J.K. A New Extension of the Kalman Filter to Nonlinear Systems. In Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls., 1997.

[26] Napolitano, M.R., Windon, D., Casanova, J., and Innocenti, M. (1996), 'A Comparison Between Kalman Filter and Neural Network Approaches for Sensor Validation', AIAA Report No. AIAA-1996-3894, AIAA Guidance, Navigation and Control Conference, San Diego, CA, USA, 29–31 July 1996.

[27] Fravolini, M.L., Campa, G., Napolitano, M.R., La Cava, M. Comparison of Different Growing Radial Basis Functions Algorithms for Control Systems Applications. Proceedings of the 2002 American Control Conference, Anchorage AK, May 2002.

[28] Calafiore, G., Campi, M. (2006). The Scenario Approach to Robust Control Design. Automatic Control, IEEE Transactions on. 51. 742 - 753. 10.1109/TAC.2006.875041.

[29] Heffes, H. The effect of erroneous models on the Kalman filter response. IEEE Transactions on Automatic Control, vol. 11, no. 3, pp. 541-543, July 1966, doi: 10.1109/TAC.1966.1098392.

[30] Duník, J., Kost, O., Straka, O., Blasch, E. State and Measurement Noise in Positioning and Tracking: Covariance Matrices Estimation and Gaussian Assessment. In Proceedings of the IEEE/ION Position, Location and Navigation Symposium (PLANS), Monterey, CA, USA, 23–26 April 2018.

[31] Odelson, B., Rajamani, M., Rawlings, J.A. New Autocovariance Least-Squares Method for Estimating Noise Covariances. Available on-line: https://sites.engineering.ucsb.edu/~jbraw/jbrweb-archives/tech-reports/ twmcc-2003-04.pdf.

[32] McCloskey, M., Cohen, N.J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. Psychology of Learning and Motivation. 24. pp. 109–165. doi:10.1016/S0079-7421(08)60536-8. ISBN 978-0-12-543324-2.

[33] Hassoun, M.H. Fundamentals of Artificial Neural Networks. MIT Press, 1995. Cambridge, MA, United States. ISBN:978-0-262-08239-6.

[34] Aggarwal, C.C. Neural Networks and Deep Learning: A Textbook. New York City: Springer International Publishing, 2018.

[35] Neglia, G., Calbi, G., Towsley, D., Vardoyan, G.. (2019). The Role of Network Topology for Distributed Machine Learning. 2350-2358. 10.1109/INFOCOM.2019.8737602.

[36] Hyndman, R.J., & Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2.

[37] Zhang, C., Hou, Y., Song, D., Ge, L. Yao, S. Redundancy of Hidden Layers in Deep Learning: An Information Perspective. arXiv:2009.09161v1 [cs.LG] 19 Sep 2020.

[38] Powell, M.J.D. (1987). Radial Basis Function for Multivariable Interpolation: A Review. Algorithms for Approximation, eds. J.C. Mason and M.G. Cox, Oxford, UK: Clarendon Press, pp. 143–167.

[39] Esposito, M., Tse, T., Bheemaiah, K. (May 2017). You've heard about it, but do you understand? everything you need to know about machine learning, [Online]. Available: https://www.weforum.org/ agenda/2017/05/what-is-machine-learning. accessed: 013.10.2021.

[40] Minsky, M., Papert, S. Perceptrons: An Introduction to Computational Geometry, The MIT Press, Cambridge MA, ISBN 0-262-63022-2. 1972 (2nd edition with corrections, first edition 1969).

[41] Share price time series forecasting for effective supply chain information exchange - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Multi-layer-perception-network-topology_fig1_264837175 [accessed 9 Nov, 2021].

[42] Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986) Learning internal representation by error propagation. In: Rumelhart, D. E., McClelland, J. L. and the PDP Research Group (eds) Parallel distributed processing, vol 1, chap 8. MIT Press, Cambridge, MA.

[43] Innocenti, M., Napolitano, M.R. Neural Networks and other Techniques for Fault Identification and Isolation of Aircraft Systems. Intelligent Systems for Aeronautics. June 1, 2003.

[44] Napolitano, M.R., Silvestri, G., Windon II, D.A., Casanova, J.L., Innocenti, M. Sensor Validation Using Hardware-Based On-Line Learning Neural Networks. IEEE Transactions on Aerospace and Electronic Systems, Vol. 34, No. 2. April 1998.

[45] Rojas, R. Neural Networks - A Systematic Introduction, Chapter 7, The Back propagation Algorithm, 1996. http://www.inf.fu-berlin.de/~rojas/neural/chap7.ps.

[46] Barry, R. Artificial Neural Network Prediction of Wavelet Sub-bands for Audio Compression, Thesis for Bachelor of Science, Dept. of Electrical & Computer Engineering, 2000. http://www.ilos.net/~rbarry/Thesis.pdf.

[47] Campa, G. (July 2007) MATLAB. R11.1-R14. [Adaptive Neural Network Library]. Mathworks.

[48] Fröhlich, J. Neural Net Components in an Object Oriented Class Structure, Neural Networks with Java, 1996-97. http://rfhs8012.fh-regensburg.de/~saj39122/jfroehl/diplom/e-index.html.

[49] Tveter, D.R. The Backprop Algorithm, Chapter 2, 2001.

[50] Li, F.F. (2021). Convolutionl Neural Networks for Visual Recognition. Stanford University CS231n. Retrieved September 9, 2021, from https://cs231n.github.io/neural-networks-1/.

[51] Mitchell, T.M., Machine Learning, Chapter 4, (1997) , McGraw-Hill.

[52] Broomhead, D.S.; Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks (Technical report). RSRE. 4148.

[53] Bottou, L. (1998). Online Algorithms and Stochastic Approximations. Online Learning and Neural Networks. Cambridge University Press. ISBN 978-0-521-65263-6.

[54] Sundararajan, N., Saratchandran, P., Wei, L.Y. Radial Basis Function Neural Networks with Sequential Learning: MRAN and Its Applications. 1999.

[55] Younis, B.T., Saied, B.M. (2006) Friction Compensation in a Controlled One Link Robot Arm with Observer using Emran RBF-NN.

[56] Clark, R.N., Setzer, W. Sensor Fault Detection in a System with Random Disturbances. IEEE Transactions on Aerospace and Electronic Systems, vol. AES-16, no. 4, pp. 468-473, July 1980, doi: 10.1109/TAES.1980.308976.

[57] Koenig, D., Patton, R.J. New Design of Robust Kalman Filters for Fault Detection and Isolation. IFAC Proceedings Volumes, Volume 32, Issue 2, 1999, Pages 7926-7931, ISSN 1474-6670, https://doi.org/10.1016/S1474-6670(17)57352-8.

[58] Stepanov, O.A. (15 May 2011). Kalman filtering: Past and present. An outlook from Russia. (On the occasion of the 80th birthday of Rudolf Emil Kalman). Gyroscopy and Navigation. 2 (2): 105. doi:10.1134/S2075108711020076. S2CID 53120402.

[59] Humpherys, J. (2012). A Fresh Look at the Kalman Filter. Society for Industrial and Applied Mathematics. 54 (4): 801–823. doi:10.1137/100799666.

[60] Satish, L., Gururaj, B.I. (April 2003). Use of hidden Markov models for partial discharge pattern classification. IEEE Transactions on Dielectrics and Electrical Insulation.

[61] Bucy, R.S., Joseph, P.D., Filtering for Stochastic Processes with Applications to Guidance. John Wiley & Sons, 1968; 2nd Edition, AMS Chelsea Publ., 2005. ISBN 0-8218-3782-6.

[62] Jazwinski, A.H., Stochastic processes and filtering theory, Academic Press, New York, 1970. ISBN 0-12-381550-9.

[63] Napolitano, M.R. (2012). Aircraft Dynamics: From Modeling to Simulation. Hoboken, NJ: J. Wiley.

[64] Walrand, J., Dimakis, A. (August 2006). Random processes in Systems -- Lecture Notes (PDF). pp. 69–70.

[65] MATLAB. (2021). version 9.10.0 (R2021a). Natick, Massachusetts: The MathWorks Inc.

[66] Courses, E., Surveys, T. (2006). Sigma-Point Filters: An Overview with Applications to Integrated Navigation and Vision Assisted Control. Nonlinear Statistical Signal Processing Workshop, 2006 IEEE. pp. 201–202. doi:10.1109/NSSPW.2006.4378854. ISBN 978-1-4244-0579-4. S2CID 18535558.

[67] Smith, G.L., Schmidt, S.F., McGee, L.A. (1962). Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle. National Aeronautics and Space Administration.

[68] Jacobian - Definition of Jacobian in English by Oxford Dictionaries. Oxford Dictionaries - English. Archived from the original on 1 December 2017. Retrieved 2 September 2021.

[69] Julier, S.J. The Scaled Unscented Transformation. Automatica, February 2000.

[70] Napolitano, M.R., An, Y., Seanor, B.A. A fault tolerant flight control system for sensor and actuator failures using Neural Networks. Aircr. Des. 3, 103-128, doi:10.1016/S1369-8869(00)00009-4.

[71] Perhinschi, M.G., Beamer, F. (2012). Flight simulation environment for undergraduate education in aircraft health management. Computers in Education Journal. 22. 50-62.