A MACHINE LEARNING APPROACH TO PITOT STATIC ERROR DETECTION AND AIRSPEED PREDICTION

An Undergraduate Research Scholars Thesis

by

RENEE SWISCHUK

Submitted to the Undergraduate Research Scholars Thesis program
Texas A&M University
in partial fulfillment of the requirements for the degree of

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Douglas Allaire

May 2017

Major: Applied Mathematical Sciences

ABSTRACT

A Machine Learning Approach to Pitot Static Error Detection and Airspeed Prediction

Renee Swischuk
Department of Mathematics
Texas A&M University

Research Advisor: Dr. Douglas Allaire
Department of Mechanical Engineering
Texas A&M University

Aircraft guidance is dependent on various sensors which provide information on speed, altitude and location with respect to both the ground and the surrounding air. The pitot static system, global positioning system (GPS) and inertial navigation system (INS) are the main sources of information. The pitot static system measures total and static pressures to provide airspeed information. This system includes two ports located outside of the aircraft making them vulnerable to interference and failures. Autonomous aircraft software has not yet been developed to handle failures in this system. If an aircraft has access to redundant data streams, then it can be trained to autonomously recognize errors in the pitot static system and learn to correct them. In this work, we develop a novel machine learning approach to detecting pitot static system failures, identifying types of failures and predicting airspeed with the use of redundant flight data. Two running estimates of airspeed are kept during flight and major discrepancies between the two triggers an error identification system. This identification system computes the autocorrelation of the incoming pressure data to classify the state of the pitot static system. Exploratory dimensionality reduction and feature selection techniques are performed on the redundant

data to create a library of selected sensor output from previous flights. This library is used to train a k-nearest neighbors regression model to make online airspeed predictions in the event of a pitot static system failure. We demonstrate our methodology on sample flight data from a four engine commercial jet. Having this fault resistant guidance system for an aircraft makes it possible to remain in flight and continue critical missions.

ACKNOWLEDGMENTS

I would like to thank my professor, Dr. Douglas Allaire, for all his help and guidance over the past year. I would also like to thank Brian Burrows for all his encouragement and support no matter what corners of the country I drag him to. I look forward to all our adventures together. Thanks to my parents for helping me out all through my life. Last but not least big thanks to my cat, Whiskers Chicken Nugget, my dog, Daisy Muffins, and my roommate, Greg Krupit, for being awesome.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

Undergradute study was supervised by Professor Douglas Allaire of the Department of Mechanical Engineering.

Funding Sources

This work was partially supported by FA9550-16-1-0108, under the Dynamic Data-Driven Application Systems Program, Program Manager Dr. Frederica Darema.

NOMENCLATURE

UAV	Unmanned Aerial Vehicle
INS	Inertial Navigation System
GPS	Global Positioning System
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
ADIRU	Air Data Inertial Reference Units

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The airspeed indicator is directly connected to the pitot static system providing the main source of guidance for an aircraft. The pitot static system measures air pressure differences to determine the speed of the aircraft relative to the surrounding air. Combined with a Global Positioning System (GPS) on board, the aircraft is able to autonomously follow a flight path. Unfortunately, due to varying temperatures and pressures at high altitudes, or blockages of the pitot and static ports, the system can give erroneous readings. These faulty readings can lead to discrepancies, inaccuracies or even complete loss of speed and altitude data. This is the reason that larger, commercial aircrafts typically have three pitot static systems for the captain, first officer and a standby. If one system becomes compromised, pilots can resort to the standby system and safely continue. Due to weight restrictions and minimal surface area, there is typically only one pitot static system on smaller unmanned aerial vehicles (UAV) [1]. For these aircraft, there is no standby system to resort to so errors must be recognized and understood rapidly to continue flight. While numerous improvements to the performance of pitot tubes have been made, there have also been catastrophic results from failures.

The well studied crash of Air France flight 447 from Rio de Janeiro to Paris in June 2009 was determined to be the result of pitot tube icing. The incident occurred while flying through extremely dynamic weather found in the intertropical convergence zone [2], an area of converging winds from both sides of the equator which lead to erratic pressure changes and violent thunderstorms. The icing led to conflicting airspeed readings which in combination with pilot error, led to the plane eventually crashing into the Atlantic ocean [3]. At least a dozen other incidents involving obstructed pitot tubes occurred in the same area as the Air France flight, but with the help of back up systems and pilot intervention,

none of them were as catastrophic as flight 447.

Improvements have been made to help pitot tubes detect anomalies and recover from temporary loss of information. Using nuclear reactor technology, the Integrated Pitot Health Monitor was developed by the Analysis and Measurement Services Corporation in collaboration with NASA to detect obstructions in pitot tubes [4]. The monitor transforms pressure readings into electronic signals and senses when signals become static, an indication the pitot tube is no longer reacting to the change in air pressure. Fault tolerant air data inertial reference units (ADIRU) supported on board an aircraft calculate highly reliable ground speed readings [5]. With this ground speed, an accurate measure of wind speed must be made to estimate airspeed and help define the load factor and capability of an aircraft. By sampling weather forecasts at various altitudes and locations, an estimate of current wind speed can be found by interpolation. The P.I.L.O.T.'s software designed in Ref. [6] uses this estimate of wind speed in combination with ground speed to correct faulty readings from compromised pitot tubes. In an attempt to avoid ice accumulation and freezing of pitot tubes all together, a heating element has also been added to pitot tubes currently in use today. A heating design by Rosemount Aerospace places an electric coil inside the pitot tube cylinder to melt any ice that may accumulate [7]. These improvements have made commercial airliners an increasingly safe form of transportation. Having pilots readily available to interpret discrepancies with airspeed data provides an added level of assurance in flight. A precise understanding of the state of the aircraft must be made without human interaction in order for a UAV to stay in flight after a pitot static system failure. The aircraft must autonomously understand the errors and be able to correct itself.

While the pitot static system is the most accurate estimator of airspeed, there are additional sensors on an aircraft providing data that can be used to verify the integrity of the system. In this paper, we develop a novel machine learning approach to detecting pitot static system failures, identifying types of failures and predicting airspeed with the use of

redundant flight data. The inertial navigation system (INS) contains a gyroscope and an accelerometer completely contained within the aircraft which prevent their readings from being obstructed by changing weather patterns outside the aircraft. This accelerometer data is used to detect when the pitot static airspeed may be unreliable. The autocorrelation of two pressure streams are calculated in flight and compared to a collection of previous flight data. In the presence of these errors, the aircraft also has the ability to predict its true airspeed as long as the errors persist using pitot independent data streams. Simulated annealing is used for dimensionality reduction and feature selection of the sensor data to aid in making highly accurate airspeed predictions. This airspeed prediction is made using a k-nearest neighbors regression model that has been trained on a library of previous flight data. Our goal is to design a system that is capable of detecting and identifying pitot static errors within 30 seconds of failure as well as providing a corrected airspeed within 10 knots of the true airspeed for a UAV with the aim of keeping more critical missions on track and in flight. The rest of this paper is as follows. Section 2 discusses the details of the pitot static system and how failures can occur. Section 3 outlines the methodology used to teach an aircraft to identify errors and predict airspeed. Section 4 defines the problem setup for our demonstration as well as the results. Finally, in Section 5 we evaluate the performance and discuss future work.

2. BACKGROUND

The pitot static system has direct connections to the airspeed indicator, altimeter, vertical speed indicator and other devices. All of these mechanisms are necessary for an aircraft to stay in flight. Unlike a car or a boat, commercial aircraft and remotely piloted UAV's cannot be flown by looking out of the front windshield. Aircraft sensors are crucial to keeping an aircraft in flight. With many of these sensors being located outside the aircraft, they are susceptible to a number of different failures. This section details how the pitot static system produces airspeed and how a block within the system can effect this airspeed.

2.1 Pitot Static System

The pitot static system consists of three main components; the pitot probe, the static port and the pitot static instruments. The system is illustrated in Figure 2.1. The pitot probe is a cylindrical tube that faces forward, in the direction of flight, on the wing or under the plane. As the plane flies, the total pressure, P_t, a combination of dynamic and static pressure, is measured through this tube. The static port is oriented perpendicular to the direction of flight and is used to measure static pressure, P_s. The difference between the total pressure and the static pressure is the dynamic pressure, P_d. Airspeed is proportional to dynamic pressure using Bernoulli's Equation :

$$V^2 = \frac{2(P_t - P_s)}{r} = \frac{2P_d}{r}$$

where r is local air density and V is airspeed [9]. The pitot tube is attached to a diaphragm that is enclosed in a container vented to the static port. This diaphragm controls the airspeed indicator as shown in Figure 2.2a. As the speed of an aircraft increases, the total

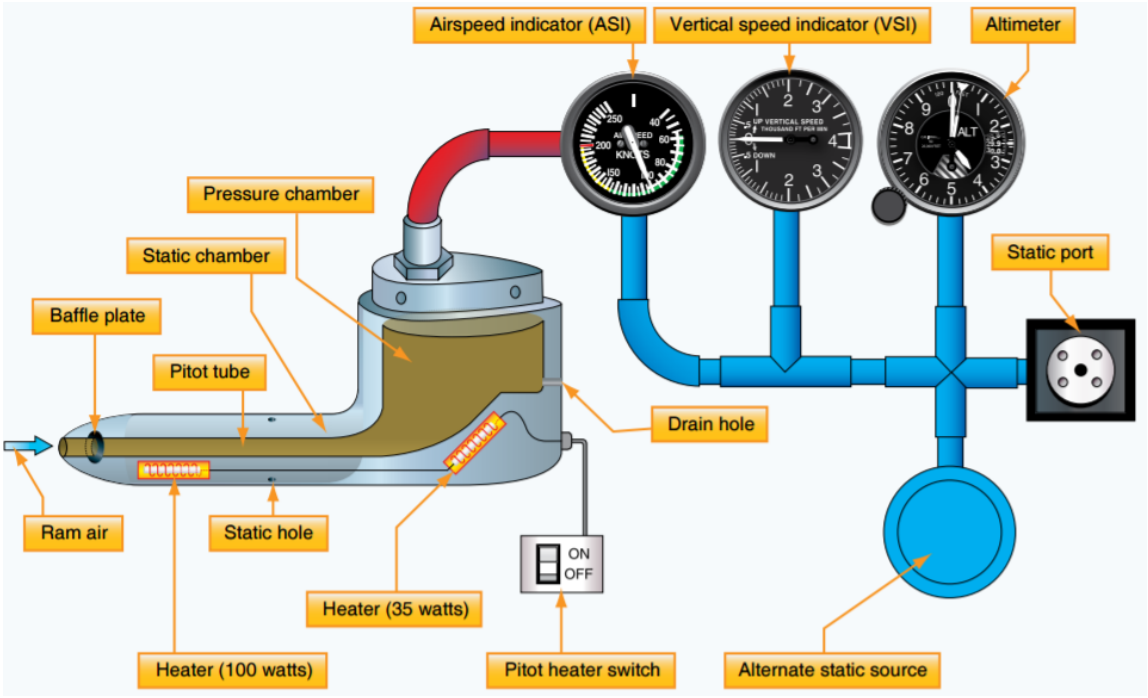
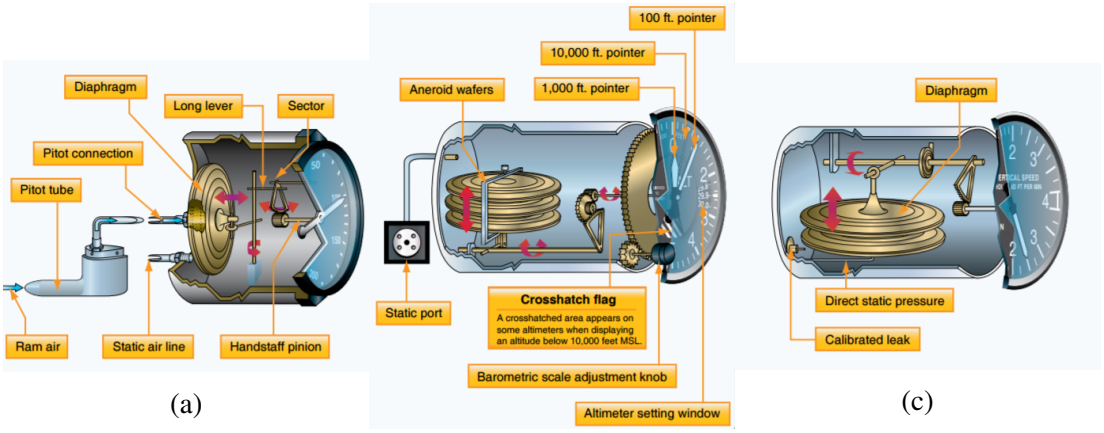


Figure 2.1: Pitot static system and instruments adapted from [8]

pressure increases, inflating the diaphragm against the static air which rotates the needle in the airspeed indicator.

The pitot static system also provides the altitude of the plane as shown in Figure 2.2b. The static port is connected to an air tight container enclosing an aneroid barometer which measures pressure and controls the altimeter [8]. As the aircraft climbs, the air density decreases and as it descends, density increases. Prior to flight, these barometers are calibrated to the pressure at mean sea level.

Another useful piece of flight information provided by the pitot static system is the vertical speed indicator. This indicator detects the rate of change in altitude and determines the orientation of the plane. This device, found in Figure 2.2c, contains a diaphragm that is enclosed in a container vented to the static port. Both the altimeter and the vertical speed indicator depend only on the static port itself and not the pitot tube [8]. As the aircraft



(a) (b) (c)

Figure 2.2: Airspeed Indicator (a) Altimeter (b) Vertical Speed Indicator (c) adapted from [8]

climbs in elevation, the air density from the static port decreases, allowing the diaphragm to expand and show an increased rate of climb on the vertical speed indicator. As the aircraft descends, the static air density increases causing the diaphragm to contract and indicate a rate of descent.

2.2 Pitot Static System Errors

The external ports that make up the pitot static system include a pitot tube which measures total pressure, the static port which measures static pressure, and the drain found on the pitot tube probe. Each of these systems can become compromised and have catastrophic effects on aircraft safety. Any combination of the three openings can become blocked and the results depend on the order in which the blockage occurs. The air measured by the pitot tube is referred to as ram air and the air measured by the static port is referred to as static air. In this paper, we focus on single blockages and their effects on the airspeed indicator. The remainder of this section outlines these effects as found in Ref. [8].

2.2.1 Pitot Tube Block

As stated earlier, the pitot tube is connected to a diaphragm which is sealed inside of a container linked to the static port opening. When the diaphragm inflates, the airspeed indicator increases. If the pitot tube becomes blocked, ram air is no longer entering the system, but the static port and the drain remain unobstructed. The diaphragm begins to compress and the air leaks out of the drain due to the pressure of the incoming static air. The diaphragm quickly deflates and the airspeed indicator drops to 0 and stays at 0 until the blockage is removed.

2.2.2 Static Port Block

When the static port becomes blocked, any air within the container becomes trapped. The density of the trapped air depends on the altitude at which the block occurred. If the aircraft ascends above the altitude of the block, the ram air becomes less dense than the static air surrounding it which results in a drop in airspeed, a reasonable result. However, since the static air at this altitude is not being collected, the pressure difference between ram air and static air is larger than it should be, which results in an under estimated airspeed. As the aircraft descends below the blockage altitude, the ram air becomes denser than its surrounding static air. Since the static air is trapped and no longer changing, the difference between ram air and static air is larger than it should be causing the diaphragm to inflate and overestimate the true airspeed. The effects of this problem are most apparent when the aircraft has a change in altitude.

2.2.3 Pitot Tube Drain Block

We consider the effects of the drain and the pitot tube becoming blocked together. This is a scenario common to pitot icing, the entire probe ices and becomes completely blocked. The air that was contained in the diaphragm at the time of the block becomes trapped. As

the aircraft ascends above the blockage altitude, the static air becomes less dense allowing the denser air trapped inside the diaphragm to expand and cause the airspeed indicator to increase as the altitude increases, opposite of the true airspeed. Similarly, as the aircraft descends below the blockage altitude, the static air becomes more dense causing the diaphragm to compress and show a decreasing airspeed as the altitude decreases, opposite of the true airspeed.

3. METHODOLOGY

In this chapter we present the methods used to develop an error detection system that is able to identify three types of pitot static system errors and predict a corrected airspeed. Our system uses an online/offline approach where the "offline" state refers to information collected or computations performed prior to flight and when the aircraft is in flight, it is considered "online". Two airspeed estimates are defined and an error threshold is set to detect when these estimates disagree. Following this, an error identification technique is presented which computes autocorrelation to classify the type of pitot static failure. We discuss how dimensionality reduction and feature selection techniques are performed on the high volume of sensor data using simulated annealing. Finally, we outline how this sensor data is combined to form an offline library which is used for predicting airspeed online in the event of a failure.

3.1 Error Detection

The system keeps a running comparison of two airspeed estimates; one from the pitot static system and one from the inertial navigation system's acceleration data. When the difference between the two airspeeds is more than our threshold, we trigger a system that uses autocorrelation to determine the type of failure. INS acceleration data contains small calibration errors and as we integrate this data, the information slowly drifts from the true value. The threshold we set is to allow this drift to occur. This section discusses the calculation of our two airspeed estimates and our approach to defining the error threshold.

3.1.1 Pitot Dependent Estimate

The airspeed indicator reports airspeed directly from the pitot static system as explained in section 2.1. Airspeed can also be calculated numerically from the pitot static

system data using Bernoulli's equation [9]. The total and static pressures (P_t and P_s, respectively) are measured and an estimate of airspeed, denoted as V^{pitot}, can be calculated as follows:

$$V^{pitot} = \sqrt{\frac{2(P_t - P_s)}{r}} \tag{3.1}$$

where r is the local air density. We would like to find a pitot independent estimate of this airspeed as a way to detect when the pitot static system is producing unreliable airspeed.

3.1.2 INS Estimate

The inertial navigation system (INS) consists of an accelerometer and a gyroscope. The accelerometer measures acceleration in three dimensions; longitudinal (x), latitudinal (y), and vertical (z) in the body coordinate system shown in Figure 3.1. The pitot tube points in the direction of flight, or the longitudinal (x) direction, therefore the integral of longitudinal acceleration should closely match the pitot static estimate of airspeed. The INS is completely self contained inside the aircraft, not susceptible to outside damage or radio interference like the pitot static system or a GPS might be. Although, due to calibration issues, accelerometer measurements tend to drift over time [10]. A simple integration of acceleration is evaluated at each time step to calculate velocity as follows:

$$V_t^{ins} = \int_{t-1}^{t} a_x dt + V_{t-1}^{ins} \tag{3.2}$$

where a_x is the longitudinal acceleration and V_t^{ins} is the INS velocity at time t. As these accelerations are integrated to estimate velocity, we add on small amounts of calibration error from the previous calculation producing an error that compounds with each time step. Despite these errors, this estimate of airspeed consistently follows the same trends as the pitot airspeed throughout flight. Each INS will have a different amount of error but, if we can determine an INS dependent error threshold that increases with time, we can use this

airspeed as an indicator of the integrity of the pitot static airspeed readings. This threshold is set to allow the INS airspeed to drift from the pitot static estimate by an amount that is considered "normal" during a safe flight. If the two airspeeds diverge by more than this threshold, then we have a good indicator of a possible failure in the pitot static system.

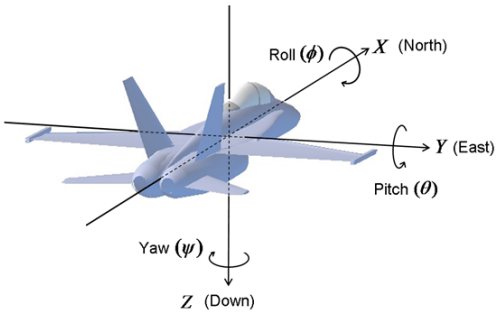


Figure 3.1: Body reference frame.

3.1.3 Error Threshold

If we would like to reduce the amount of computations that have to be done online, we need an indicator of when to check for a failure. At the end of flight, the INS has an estimate of the position of the aircraft (with a considerable amount of drift present). This estimate is then compared to a GPS measurement of position to determine the error accumulated during flight and reduce the error in its measurements. This means INS become more accurate with use so a general threshold should not be defined for all systems. Instead, a system specific threshold should be defined to capture the expected error. Due to the compounding integration drift, this threshold should be defined as an increasing function of time. Denote V^{thresh} as our threshold and if at any point during flight, $|V^{pitot} - V^{ins}| > V^{thresh}$, the error identification system will be triggered. While this

threshold does reduce the number online computations made to identify the type of errors occurring, it may be desirable to not have a threshold defined. If the aircraft is not changing altitude, pitot static blocks do not have a significant effect on airspeed and may not be detected if a threshold is set. It may be useful to know that errors are occurring even if they aren't currently having an effect on the airspeed. This is taken into consideration in later sections and our method is demonstrated both with and without a threshold.

3.2 Offline Error Library and Classification of Failure

To identify what type of failure may be occurring, the aircraft must have knowledge of how these failures effect flight data. Erroneous flight data was manually produced to simulate the effects of the three pitot static system blocks explained in Section 2.3. This data was created by changing the values of the total or static pressure streams and calculating airspeed using Bernoulli's equation (Eq 3.1). Each of the blocks were simulated for 100 seconds at four different times of flight and the resulting pressure and airspeed data is stored. The pitot tube block is detected when the airspeed drops to zero so this section focuses on a method to detect when a pitot drain and static port block are occurring. An offline library of faulty data is created and used as a reference for detecting these failures online.

The idea behind the offline library is to have a collection of error signatures produced during a failure and check if our online readings match any of these signatures. For example, the pitot drain and static port blocks cause the total and static pressure readings (respectively) to become constant. Therefore, the error signature for these blocks would be a constant pressure reading. When online, if our total pressure reading has become constant, we want to map that to the appropriate failure case in our offline library to signal a pitot drain block is occurring. Instead of storing a large number of pressure data from previous flights in this library, we compute the autocorrelation of these readings and store

those values instead. This requires much less information to be stored. Autocorrelation calculates the correlation between two points in a signal, where the second point is further in the signal, or delayed in time by a certain lag value[11]. The autocorrelation of a signal $Y = \{y_1, ..., y_n\}$ is defined as: [11]

$$ac_j = \frac{\sum_{k=1}^{n-j}(y_k - \bar{Y})(y_{j+k} - \bar{Y})}{\sigma_Y^2}, \tag{3.3}$$

where j is the lag value and $\bar{Y}$ and σ_Y^2 are the sample mean and variance of the signal. As shown, autocorrelation is undefined for constant valued signals since the sample variance would be zero. To account for this, we add 1 to the denominator, allowing the autocorrelation to exist while still keeping all values equivalently scaled. The autocorrelation values will quickly drop to zero for a constant signal whereas the values for noisy signal will take more time [12]. By collecting the first few lag values of autocorrelation, this method shows a pattern that is able to differentiate between constant signals and noisy signals.

Two libraries are stored offline, one for total pressure and one for static pressure. Each sample in the libraries consists of the autocorrelation for each pressure stream and one class label, denoting the state of the pitot static system. Class label 0 denotes a safe state, class label 1 denotes a pitot tube block, class label 2 denotes a pitot drain block and class label 3 denotes a static port block. The pitot drain block traps the air measuring total pressure, so it only has an effect on the total pressure data stream. Therefore, the autocorrelation values for the total pressure streams simulated from the pitot drain block receive a class label 2 while the static pressure values will receive a class label 0 indicating safe. On the other hand, the static port block traps the air measuring static pressure, so it only has an effect on the static pressure data streams. The autocorrelation values for static pressure streams simulated from the static port block receive a class label 3 while the total pressure values receives a class label 0. An outline of this error library is illustrated

in Figure 3.2. To detect a pitot tube block we simply detect when the airspeed indicator drops to zero so this error is not stored in our library but the class label 1 is reserved for this case. Total pressure and static pressure is collected online and the autocorrelation is taken using a moving window of 10 seconds. These autocorrelation values are compared to their respective libraries to find the offline cases that are most similar to them. At each instance, two class labels are gathered for each of the two pressure streams so specific patterns are searched for. The combinations of label predictions is shown in Table 3.1. If total pressure is assigned class label 2 and static pressure is assigned class label 0 then the pitot drain error is signaled. If static pressure is assigned class label 3 and total pressure is assigned class label 0 then the static port error is signaled. All other combinations are assumed to be a safe system state. We do not consider the case of a double block occurring. If a failure state is predicted, our pitot static airspeed is unreliable and a new airspeed must be calculated.

Total Pressure Prediction	Static Pressure Prediction	Final System Prediction
0	0	0
2	0	2
0	3	3
2	3	0

Table 3.1: Using autocorrelation as a measure, each pressure stream is given a predicted label. This label can be 0 (safe), 2 (pitot drain block), or 3 (static port block). Label 1 is assigned using a different method. The pitot drain block only effects the total pressure, so label 2 for total pressure and label 0 for static pressure indicates a pitot drain block is occurring. The static port block only effects static pressure, so label 0 for total pressure and label 3 for static pressure would indicate a static port block. All other combinations are considered noise and indicate a safe state.

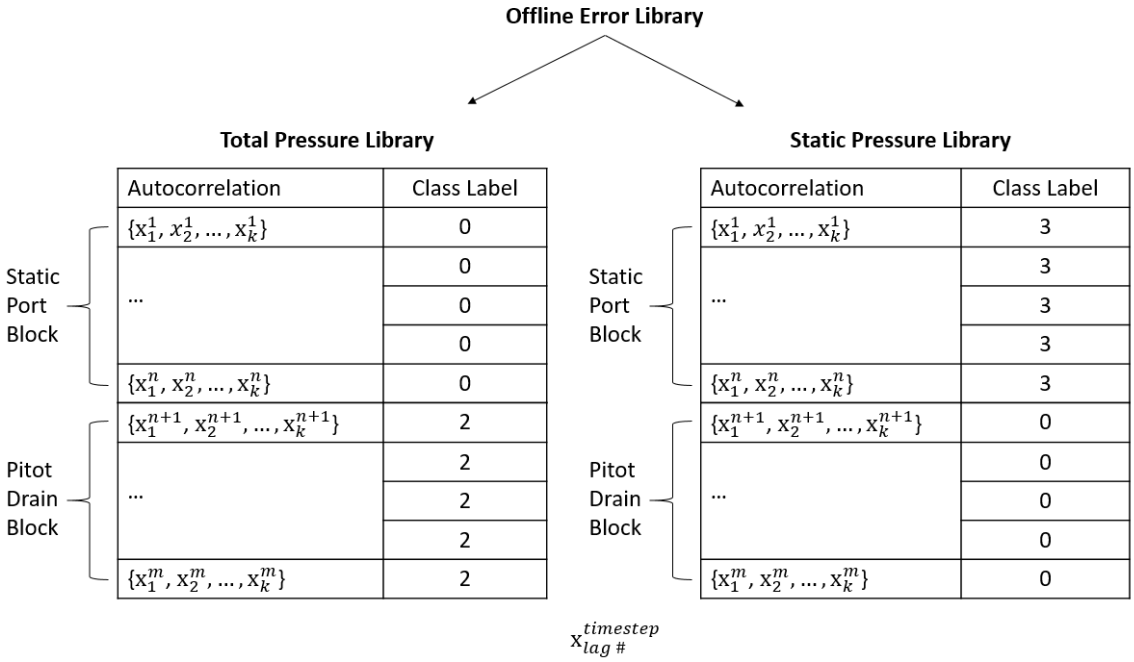


Figure 3.2: Pitot drain and static port blocks are simulated and the autocorrelation of the individual pressure streams are stored in two separate libraries. Each sample in the library consists of autocorrelation values and a label. The label denotes whether a block is effecting that pressure stream (0: safe, 2: pitot drain block, 3: static port block). In flight, autocorrelation is calculated for both pressure streams and they are mapped to their respective libraries.

3.3 Dimension Reduction and Feature Subset Selection

There are hundreds of different sensors on an aircraft that are constantly producing data. Most of this data is either too noisy or irrelevant for use in airspeed predictions. Too much data can also have a negative effect on predictions caused by issues such as over fitting a model, unintentional weighting, or simply being too computationally complex. This problem is known as the curse of dimensionality [13]. Airspeed predictions must be made rapidly to keep an airplane in flight, so we need to reduce the number of data streams that are collected and analyzed. To narrow down the number of data streams stored in our library and improve efficiency and accuracy in predictions, multiple exploratory

dimensionality reduction and feature subset selection techniques were employed.

An aircraft sensor produces a constant data stream which is also denoted as a "feature" in our data set. We can determine how many features are necessary and which ones are useful in preserving the most amount of information about the aircraft. The number of subsets of features is exponential in the number of available features so a direct feature subset selection over the entire data set results in a combinatorial optimization problem which is computationally too expensive. Initial dimension reduction is done using simulated annealing in conjunction with a distance based objective function to estimate a set of locally optimal dimensions. In this case, a locally optimal dimension is defined as the number of dimensions which preserve the most structure and overall information within the dataset. Feature subset selection is done using simulated annealing and two greedy choice algorithms on the set of optimal dimensions to find the most informative features for predicting airspeed. This section outlines the techniques used for dimensionality reduction and feature subset selection as well as a discussion of the simulated annealing and greedy choice algorithms.

3.3.1 Dimensionality Reduction

Non-linear dimensionality reduction (or manifold learning) is a technique for determining if there exists a lower dimensional manifold which defines the structure of a high dimensional space[14]. A common approach to non-linear dimension reduction is to define a non-linear mapping which projects the data from a high dimensional space to a lower dimension while minimizing a given mapping cost function. Linear dimensionality reduction techniques such as principal component analysis have a similar approach where instead, a lower dimensional space is made up of a linear combination of the original dimensions. Both of these methods produce new, projected data. Our goal is to make rapid, online predictions of airspeed so the added complexity of projecting raw data to low di-

mensional spaces is not ideal. Although, by taking the concepts of these dimensionality reduction techniques, we can take subsets of our original dimensions and apply a similar optimization without permuting our raw data.

Due to basic flight mechanics, airspeed will be correlated with various other data collected during flight such as accelerations, fuel flow and engine information. Intuitively, a subset of flight data that can be used to predict airspeed should exist. Our online predictions will be made by comparing sensor data to closely valued data in the offline library. A low dimensional space that preserves pairwise distances would help to reduce complexity while maintaining the metric of interest e.g, Euclidean distance between points. This need to preserve distances led to a classical distance based objective function called the Sammon's mapping function. Sammon's mapping is a multidimensional scaling method which quantifies how well a projection preserves the pairwise distances and overall structure of the data set [15]. The Sammon's mapping cost function is defined as follows:

$$E = \frac{1}{\sum_{i<j}^{n} d_{ij}^{*}} \sum_{i<j}^{n} \frac{[d_{ij}^{*} - d_{ij}]^2}{d_{ij}^{*}}, \tag{3.4}$$

where x_i and x_j are points in the original p dimensional space and y_i and y_j are the projections of those points into the new m dimensional space, respectively, where $m < p$ and $i, j \in \{1, ..., n\}$. d^* is denoted as the pairwise distance matrix of the p dimensional space and d as the pairwise distance matrix of the reduced m dimensional space. d_{ij}^{*} is the Euclidean distance between x_i and x_j in the original space and d_{ij} is the distance between points y_i and y_j in the projected space. This function is a measure of how the distances between points in the two spaces compare. From the numerator we can see that the function will be at zero when the data remains in its original space. By restricting the projection dimensionality to be strictly less than our original p dimensions and finding the minimum value of E, we can find the space which preserves, on average, the most distance between

points. Due to the high number of possible subsets of features available, this dimensionality reduction method is used to find a small set of optimal dimensions that we will further explore using simulated annealing subset feature selection.

3.3.2 Simulated Annealing Algorithm

Simulated annealing is a probabilistic optimization method based on the heating and cooling of materials proposed by Kirkpatrick, Gelett and Vecchi [16]. The randomness in movement along the objective function allows this algorithm to escape local optima and increase the probability of finding a global optimum[16]. The algorithm takes a random initial solution, $Y_0 = \langle y_1, ..., y_m \rangle$, and evaluates the objective function, or cost, at this solution as $cost_{old}$. Then, we randomly choose a new solution, Y_{new}, in the neighborhood of our current solution and evaluate the objective function at Y_{new} as $cost_{new}$. If the new solution satisfies an acceptance probability based on temperature (T), $cost_{new}$ and $cost_{old}$, then we give this solution the opportunity to take the current solutions place. Whether or not the new solution is accepted as the current solution is again a random decision. T is given a cooling rate and this method is repeated a defined number of times for each value of T until $T \leq T_{min}$. As T becomes smaller, the acceptance probability increases as costs decrease which allows the algorithm to narrow down the chosen solutions to those with the minimal cost.

Initially we use simulated annealing to determine a set of optimal, distance preserving dimensions using Sammon's Mapping function as our objective. Each dimension is given a final cost and is treated as a separate optimization problem. At the end of the algorithm, the lowest cost dimensions are chosen to be explored for feature subset selection. The flight data has t rows representing each second of flight (a sample) and p columns representing the total data streams (features) available. Instead of using the actual values found in the data set as possible solutions, we use the indices of our features as possible solutions

so that we do not have to project our data to a new space. For the k dimensional problem, our initial solution, Y_0, is k randomly chosen integers from 1 to p, representing a subset of the features. We extract the actual data using the indices from our solution to evaluate Sammon's mapping cost function, but this cost is associated with the indices as opposed to the values. A neighbor, Y_{new}, is chosen by replacing 2 of the k values in our current solution with randomly chosen new ones. The cost is evaluated for our neighboring solution and compared to the previous cost. Pseudo code for the algorithm is provided below. The following code is adapted from Reeves[17]

Algorithm 1 Simulated Annealing Dimensionality Optimization
y is a (t, dim) array and $train$ is the $(t, nattr)$ original data set

1: **function** SIMULATED ANNEALING(y)
2: $cost_{old} = \text{COST}(y)$
3: $T = 15$ ▷ Initial Temperature
4: $T_{min} = .00001$
5: **while** $T > T_{min}$ **do**
6: $i = 1$
7: **while** $i \leq 500$ **do**
8: $y_{new} = \text{NEIGHBOR}(y)$
9: $cost_{new} = \text{COST}(y_{new})$
10: $ap = \text{ACCEPTANCEPROBABILITY}(cost_{old}, cost_{new}, T)$
11: **if** $ap > random$ **then**
12: $y = y_{new}$
13: $cost_{old} = cost_{new}$
14: $i++$
15: $T = T_{initial}(.95^k)$ ▷ Cooling rate
16: **return** y, $cost_{old}$
17: **function** NEIGHBOR(y)
18: $a, b = random(0, dim)$ ▷ Randomly select which two components of y to replace
19: $x, y = random(0, nattr)$ ▷ Randomly select the index of 2 new columns
20: $y[a] = x$ ▷ New columns to analyze
21: $y[b] = y$
22: **return** y
23: **function** ACCEPTANCEPROBABILITY($cost_{old}$, $cost_{new}$, T)
24: $a = exp(\frac{cost_{old} - cost_{new}}{T})$
25: **return** a
26: **function** COST(y)
27: $sol = train[:, y]$ ▷ Evaluate cost of actual values
28: **return** SAMMONSMAPPING(sol)

As shown in line 11 of the Simulated Annealing algorithm, all solutions have a probability of being chosen, but lower cost solutions will have a much higher acceptance probability giving them a higher chance of actually being selected. This also means that high cost solutions have some probability of being chosen which is what allows the algorithm to escape local minimum and continue exploring. As the temperature parameter decreases,

the acceptance probability becomes larger, assuming we are moving to lower cost solutions. As the algorithm iterates, new, low cost solutions have an increasing chance of being selected which allows the function to settle into the neighborhood of a minimum.

This minimization is performed for each dimension and the lowest cost dimensions are chosen for feature selection. Sammon's Mapping function is minimized when the entire dataset is used and maximized when the fewest number of dimensions is used. There is a trade off between computational complexity and accuracy. Less stored data results in faster computations which is necessary when attempting to rapidly predict airspeed but this airspeed data is crucial so these predictions need to be accurate as well. This accuracy in predictions is dependent on the amount of information we have. There will be a point where the cost value will level out close to zero. Adding more data at this point results in very minimal amounts of information gained from the original data set. The dimension in which this leveling out occurs is ideally the optimal dimensionality; that which requires the least amount of features while still providing an adequate amount of structure and information from the original dataset. While Sammon's Mapping does a good job at determining a reduced set of dimensions that capture the majority of information about the aircraft, it does not produce dimensions that are directly correlated with airspeed. Therefore, the dimensions that are produced by minimizing this function don't necessarily provide good estimators of airspeed. Instead, they provide an indicator of the size of a reduced dimensionality that we should consider for these predictions.

A small set of optimal dimensions are chosen to be explored for feature selection. We again use simulated annealing to perform this process but our objective function is now directly based on prediction accuracy. We define our new objective function to be the prediction performance of airspeed using k-nearest neighbors regression. All parameters used for this optimization are increased to allow the method to dig deeper into the vast number of possible subsets of features and evaluate their prediction performance. As a comparison

to this method, we also performed two greedy choice feature selection techniques.

3.3.3 Max Error and MSE Greedy Choice Algorithms

Two greedy choice algorithms were used to perform sequential forward feature selection on the set of reduced dimensions [18]. The two approaches were similar in structure but varied in the metric used to add features to the collection. A subset of flight data was used as a training set and a separate subset used as a test set. Both methods were generalized using 10-fold cross validation. Our original dataset had $nattr$ total features.

Our first approach calculates the mean squared error across all of the test samples and the second approach calculates the maximum error. Initially, we iterate through each individual data stream giving each sample a single feature. In other words, our training and test sets were simply column vectors $(n\_samples, 1)$. We make predictions of airspeed for our test set using k-nearest neighbors regression. Then for each of our test samples we calculate the error, or the difference in value between the actual airspeed and the predicted airspeed. We store the single feature, f_1, that produces the minimum mean squared error and minimum max error for each approach and remove it from our list of available features. Then we iterate through each pair of features, that is, f_1 with each of the $nattr-1$ features, and store the pair that produces the minimum of the two metrics. This was continued until the desired number of features was chosen. These methods choose the locally optimal feature at each iteration but they are not guaranteed to produce the set of features that is a global minimum of either metric.

3.4 K-Nearest Neighbors Airspeed Prediction

The features that best satisfy our optimization criteria were chosen and stored in an offline library. These features represent the only sensor data that we look at when the airplane is in flight. This library was used to train a k-nearest neighbors regression model where Euclidean distance is used to determine the nearest neighbors. This method finds

the k closest valued samples in the offline library to our sample of interest and predicts an airspeed equal to the average of those k closest airspeeds. This method allows us to take advantage of the distance preserved by Sammon's mapping function. These predictions produce a noisy stream of airspeed that is then smoothed using a Kalman Filter. A Kalman filter is a common approach taken to smooth position data such as GPS and INS [19]. Denote z as a stream of noisy airspeed data, q as the prediction error, r as the sensor error, and $\hat{x}$ as the smoothed output stream.

$$\hat{x}[k] = \hat{x}[k-1] + K[k](z[k] - \hat{x}[k-1]), \tag{3.5}$$

where $K[k] = \frac{p[k-1]+q}{p[k-1]+q+r}$ and p is a measure of accuracy[20]. The smoothing effects of the Kalman filter are shown below in Figure 3.3.

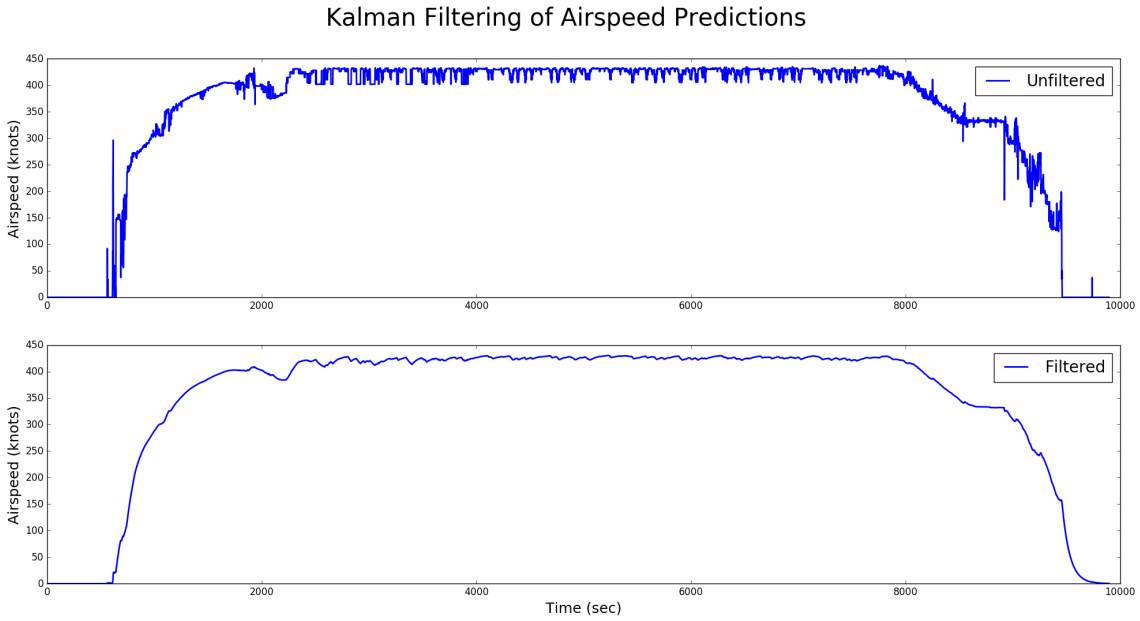


Figure 3.3: An illustration of the effects of a Kalman filter on airspeed data. Our airspeed predictions produce noisy data which is shown in the top plot. Some of this noise must be removed for this airspeed to be used as a source of guidance. The Kalman filter reduces the spikes and produces a much smoother airspeed reading throughout the flight.

4. DEMONSTRATION

Our method is demonstrated using approximately 90 hours of flight time from a four engine commercial jet collected from the NASA DASHlink data repository [21]. While commercial aircraft have three pitot static systems, we consider only one system to closely replicate the data collected from a UAV. Flights are randomly chosen from the repository to test our method. The aircraft data is evaluated every second to simulate an online flight data stream. This chapter outlines the problem setup for testing our method and discusses the results.

4.1 Problem Setup

The two airspeed estimates, V^{pitot} and V^{ins}, are calculated online at each time step as described in Section 3.1. Figure 4.1 shows the comparison of these two airspeeds and illustrates how the INS estimate closely follows the trends of the pitot estimate throughout flight. The error threshold is determined by averaging the error between the INS and pitot system estimates for a collection of safe flights. For our demonstration a linearly increasing function was defined as $V^{thresh} = .005t + 25$.

The library was created by holding the values of the total or static pressure constant and recalculating airspeed using Bernoulli's equation (Eq 3.1). Each of the blocks were simulated for 100 seconds at four different times of flight. For the pitot tube block, the total pressure will empty out of the drain on the pitot tube and the airspeed indicator will drop to zero. To simulate this, at each of the four sections of flight, the airspeed indicator drops linearly for 15 seconds until reaching zero. The indicator continues to read zero knots for 70 seconds at which point it increases linearly for 15 seconds until reaching the correct airspeed. The pitot drain block blocks the port and the drain, trapping the air which measures total pressure. To simulate this block, the total pressure at each of the four

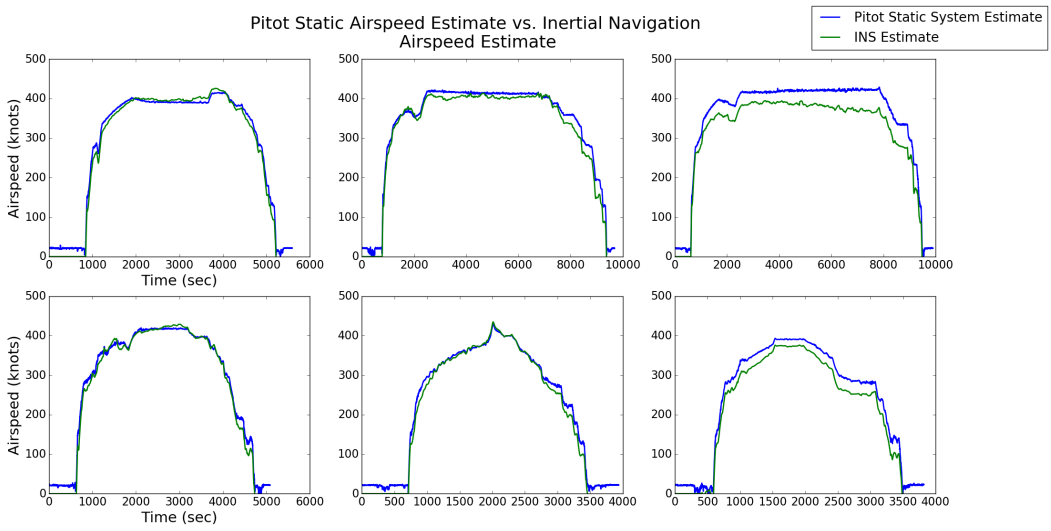


Figure 4.1: Comparison of two running airspeed estimates for 8 different flights. This illustrates how the INS estimate of airspeed (green line) closely follows the trends of the pitot static airspeed estimate (blue line). The increasing drift in the INS estimate is apparent in the far right plots.

sections of flight is held constant for 100 seconds. The static port block is simulated in exactly the same manner as the pitot drain block, except the static pressure is held constant for 100 seconds. A sample of the error produced in the pressure streams is shown in Figure 4.2. The effects of these blocks on airspeed can be seen in Figures 4.3 and 4.4.

Instead of storing the pressure streams in the library, we calculate the autocorrelation values for these streams using Equation 3.3. Autocorrelation is calculated for each of the 100 seconds of faulty pressure readings. Then using the class labels as noted in Section 3.2, the second autocorrelation lag value and a class label for each static pressure stream is stored in one offline library and the same is done for total pressure, stored in a separate offline library. A comparison of the autocorrelation values for safe flight vs. a block is shown in Figure 4.5. Autocorrelation for a constant signal quickly drops to zero while the autocorrelation for a noisy signal slowly drops to zero. The first few autocorrelation values for the two signals provide a good indicator of pitot static system failure.

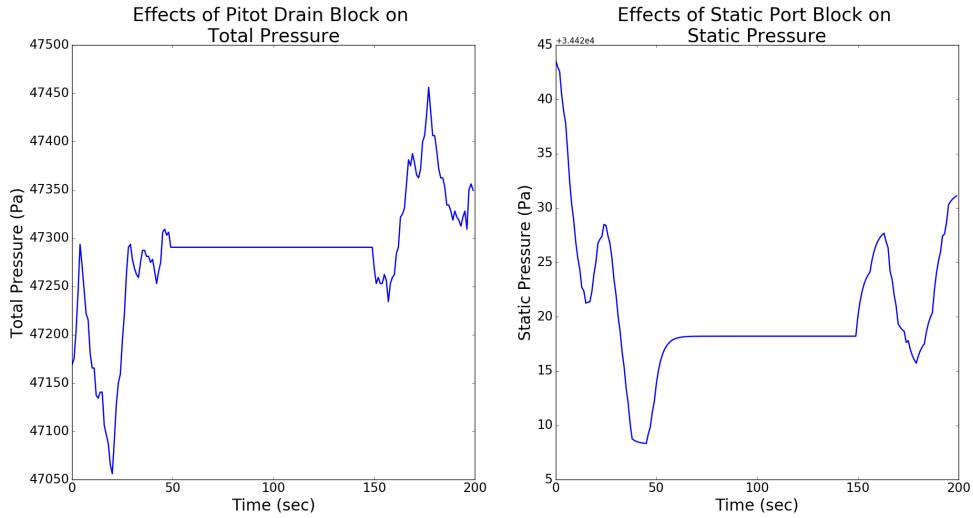


Figure 4.2: The manual data streams produced by holding pressure constant for 100 seconds at a time starting at 50 seconds. Left: Pitot drain block. Right: Static port block.

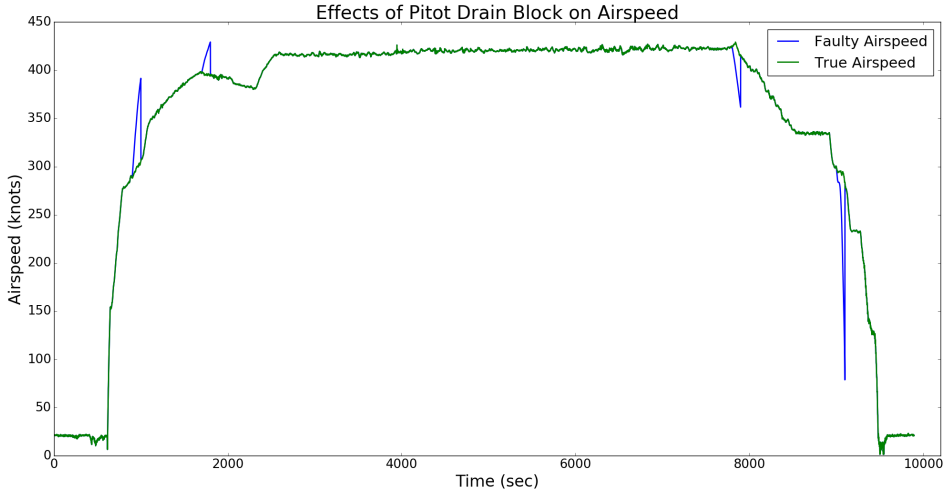


Figure 4.3: The effects of holding total pressure constant for 100 seconds during 4 sections of flight.

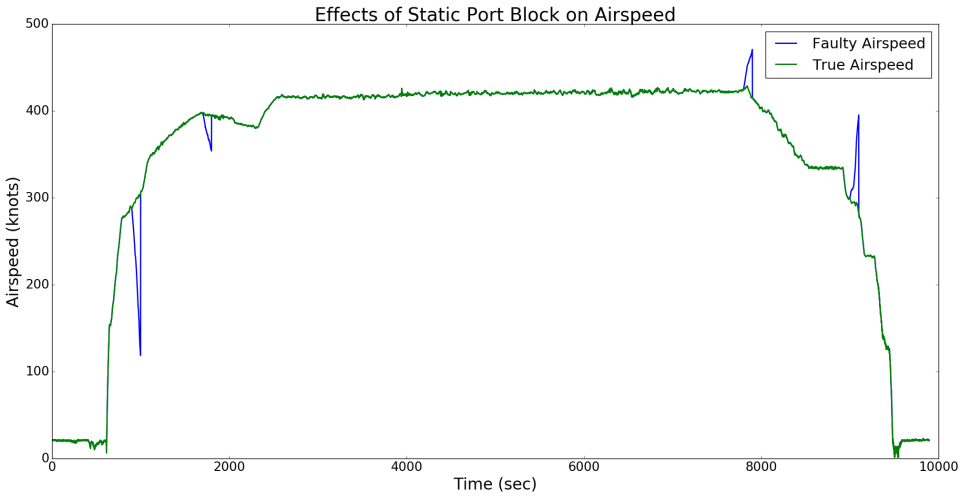


Figure 4.4: The effects of holding static pressure constant for 100 seconds during 4 sections of flight.

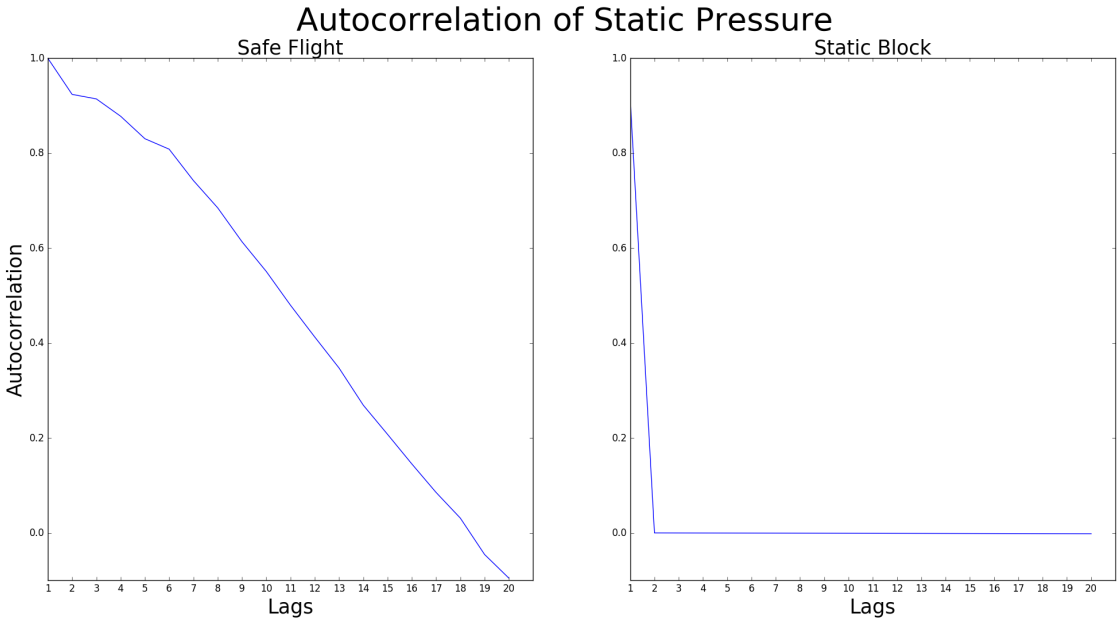


Figure 4.5: The autocorrelation values for the first 20 lags. Left: Calculated from a noisy (safe) static pressure stream. Right: Calculated from a static pressure stream that is constant due to a static port block. The autocorrelation quickly drops to zero for a constant signal. The first few lag values can differentiate a constant signal from a noisy one.

When the aircraft is in flight, it's measuring data every second. When the current airspeed error is higher than the defined threshold for ten consecutive seconds, the error identification system is activated. When activated, autocorrelation is computed for the previous ten seconds of total and static pressure and compared to the values we have stored in our offline library. Each online pressure stream is given a predicted state by finding the offline case that is closest to it using Euclidean distance. Once the closest case is found, the online pressure stream is assigned the state associated with that offline case. For example, if our online autocorrelation measure was closest to a case in the offline library that was labeled as a static port block, then that measurement would indicate a static port block is occurring. This online state prediction is continued until the pitot and INS airspeed estimates begin to agree again.

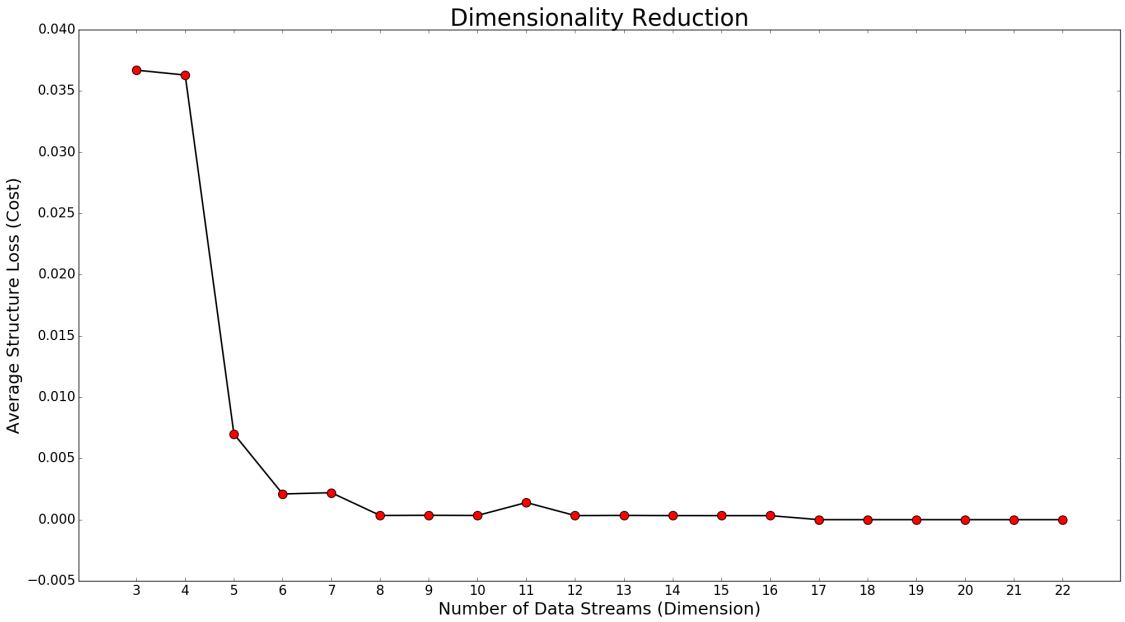
During a failure, airspeed readings become skewed and unreliable so we need to make a new prediction of the true airspeed. There are nearly 200 other sensors found on an aircraft which all produce data providing information about the aircraft. To narrow our search to those which best predict airspeed, we initially took a set of 24 data streams that are specific to the motion of the aircraft and the outside environment. The data streams are enumerated as follows.

0. Yaw
1. Pitch
2. Roll
3. Vertical Acceleration
4. Engine Core Speed
5. Engine Fan Speed
6. Power Lever Angle
7. Fuel Flow
8. Altitude Rate
9. 1^{st} angle of attack
10. 2^{nd} angle of attack
11. Body Longitudinal Acceleration

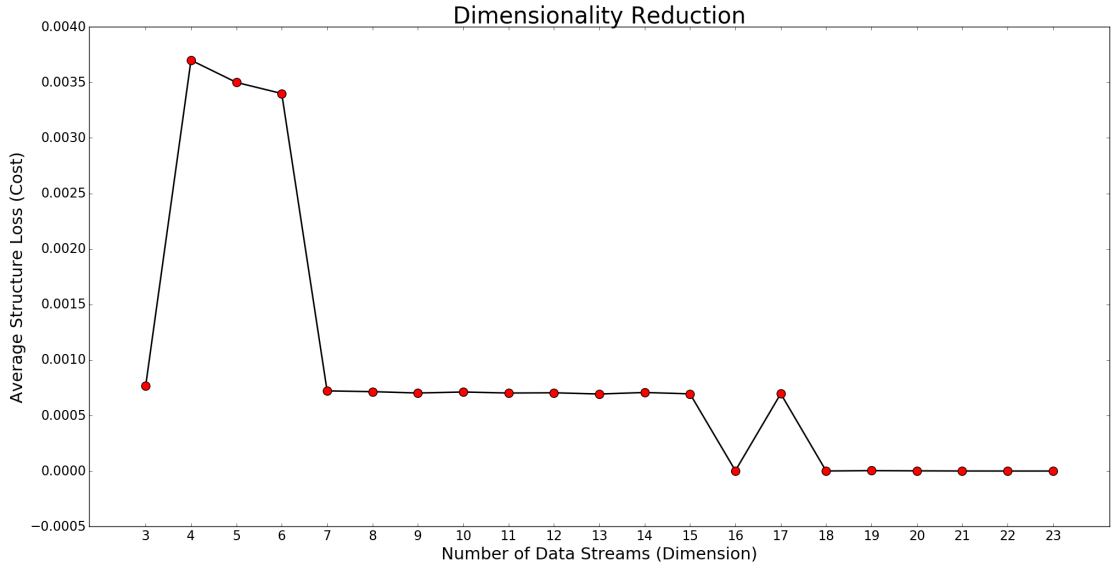
12. Cross Track Acceleration
13. Drift
14. Flight Path Acceleration
15. Inertial Vertical Speed
16. Latitudinal Acceleration
17. Longitudinal Acceleration
18. Magnetic Heading
19. Thrust Command
20. Thrust Target
21. Magnetic Track Angle
22. True Track Angle

Dimension reduction was completed on full flight datasets using Matlab's^{TM} simulated annealing function and Sammon's Mapping function as our objective. An objective function limit of 10^{-9} and an exponential cooling rate at 200 iterations were used. The results for two flights are shown in Figure 4.6. There exists a point in both of these flight datasets where a leveling out of cost occurs around zero. That is, a point where we have a maximum amount of information, or structure, with a minimal amount of dimensions. We chose the set $\{7, 8, 9\}$ as our optimal set of dimensions to perform feature subset selection on. As stated earlier, the dimensions produced by this method provides an indicator of the size of a reduced dimensionality that we should consider for predictions, not necessarily the specific features we should use.

Simulated annealing was also used for feature subset selection but the cost function is defined as the airspeed prediction accuracy. Three types of prediction accuracy were defined; mean squared prediction error over an entire flight, max prediction error over an entire flight and overall prediction score. The prediction score is defined as the R^2 score known as the coefficient of determination which measures the correlation between predicted airspeeds and actual airspeeds [22]. If $\hat{y}_i$ is the predicted value and y_i is the true



(a) Flight 1



(b) Flight 2

Figure 4.6: An illustration of how the Sammon's Mapping Cost function evaluates over various reduced dimensions for two different flight datasets. Note the areas at dimensions 7, 8 and 9 where a leveling out of cost around zero occurs. These are the "optimal" dimensions chosen to perform feature selection.

value for sample i, then the score for n predictions is [22]:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y})^2} \tag{4.1}$$

where $\bar{y}$ is the sample mean of the true airspeeds. The first two prediction measures are minimized, but the prediction score is maximized. Results for our simulated annealing feature selection as well as our greedy approaches are shown in Table 4.1. Simulated annealing produced better subsets of features than the greedy choice based on both max error and mean squared prediction error. To choose from these sets of features, we took those which produced the minimum error during the cruising portion of flight. Based on this performance, engine fan speed, fuel flow, 1st and 2nd angle of attack, longitudinal acceleration, thrust command and thrust target were selected as our most informative features for predicting airspeed and were stored in our offline library. The airspeed predictions using these features is shown in Appendix A. When online, this is the only sensor data read into our system for airspeed prediction. Airspeed predictions are made instantaneously when a failure in the pitot static system is occurring. Predictions are made using k-nearest neighbors regression and the number of neighbors is determined by evaluating the performance of the prediction using a varying number of neighbors. The number of neighbors, k, was determined to be six based on the results shown in Figure 4.7. Using more neighbors proves to have minimal gains when compared to the additional runtime costs included in consulting more neighbors.

Dimension	Features	Average Prediction Error
7	5, 7, 9, 10, 17, 19, 20	10.18
8	0, 4, 5, 7, 9, 10, 19, 20	9.88
9	0, 2, 5, 7, 9, 10, 16, 19, 20	10.08
Dimension	Features	Max Prediction Error
7	0, 1, 2, 12, 14, 20, 21	222.53
8	0, 4, 6, 10, 11, 12, 14, 20	217.73
9	0, 1, 4, 5, 6, 9, 16, 17, 20	223.73
Dimension	Features	Mean Squared Prediction Error
7	5, 7, 9, 10, 17, 19, 20	752800.50
8	4, 5, 9, 10, 11, 12, 15, 17	1539832.89
9	0, 1, 5, 7, 9, 10, 11, 19, 20	663530.85
Dimension	Features	Prediction Score
7	0, 6, 7, 10, 16, 17, 20	.9716
8	2, 3, 5, 6, 7, 10, 14, 20	.9562
9	2, 4, 7, 9, 11, 12, 13, 19, 20	.9733
Dimension	Features	Greedy Choice Max Error
7	19, 17, 20, 11, 2, 3, 12	391.9125
Dimension	Features	Greedy Choice Mean Squared Error
7	4, 7, 5, 20, 10, 19, 9	3028105.3550
8	4, 7, 5, 20, 10, 19, 9, 6	2911872.2453
9	4, 7, 5, 20, 10, 19, 9, 6, 3	2907245.4016

Table 4.1: The results of feature subset selection by six different methods. The first four sections show the results using simulated annealing with four different cost function. The last two sections show the results using our two greedy choice algorithms. Simulated annealing consistently found features which outperformed those found with the greedy approach. We chose the set out of these that best predicted airspeed during the cruising portion of flight.

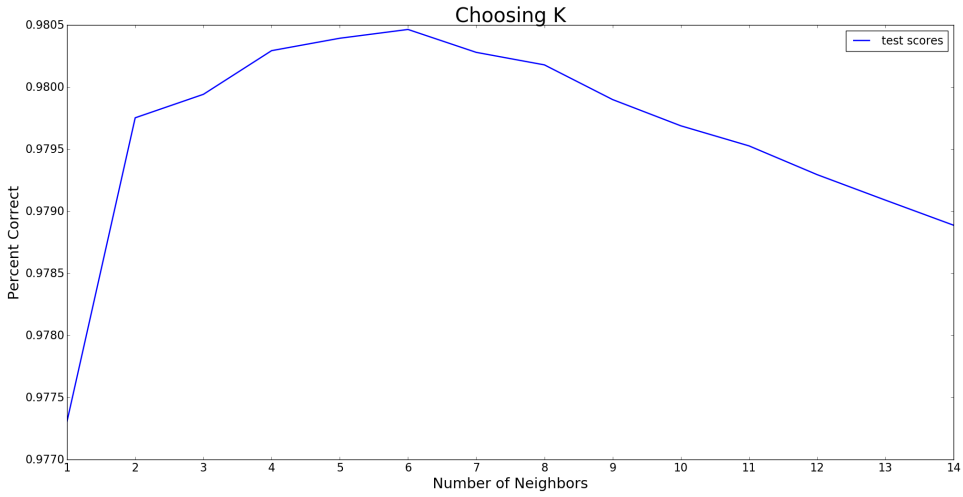


Figure 4.7: Test Score vs Number of neighbors. Test score is defined as the R^2 coefficient of determination.

4.2 Results

To test the accuracy of our error detection system, we encoded failures into the pressure streams and ran the flight data online. We ran the data as a simulation which produced and analyzed flight data every second. All three blocks were simulated at different sections of flight. The system was able to detect these errors and properly identify what type of errors they were within 20 seconds of the blocks occurring. We were then able to instantaneously predict a new airspeed during a pitot static system failure. This airspeed was within 30 knots of the true airspeed during the cruising portions of flight. The following figures show the results of our system on real flight data and are formatted as follows. The top plot shows the two running airspeed estimates, the green line is airspeed from the pitot static system and the blue line is airspeed from the accelerometers. The middle plot shows the error in those two airspeeds in blue and V_{thresh} in red. The bottom plot shows the error identification system. When we identify any type of error, our method predicts a new

airspeed which is denoted by the red line in the top plot. Figure 4.8 shows the accuracy during the entire flight, when $V_{thresh} = 0$. This illustrates that our method is able to correctly predict the state of the pitot static system during the entire flight. All of the errors were detected and correctly identified within 20 seconds of the initial block. This also shows that the method does indeed predict a safe flight correctly. Figure 4.9 shows the same scenario but instead $V_{thresh} = .005t + 25$, reducing the number of times the error classification needs to evaluate. The system was evaluated on a three more flights and the results are shown in Appendix B. During the cruising portion of flight, the aircraft is not changing altitudes enough for a static or pitot drain block to have an effect on airspeed readings. Due to this, the error between our running airspeeds will not diverge past the threshold. If we set the threshold to 0, our system is still able to detect these errors directly from pressure readings. This is also shown in Appendix B.

The error in airspeed produced by a blocked pitot static system can quickly reach above 100 knots. This error is mainly a function of change in altitude. An illustration of how quickly the error compounds is shown in Figures 4.10 and 4.11. This error jumps well above the 30 knots produced by our method and results in a very unsafe situation for both piloted and unmanned aircraft.

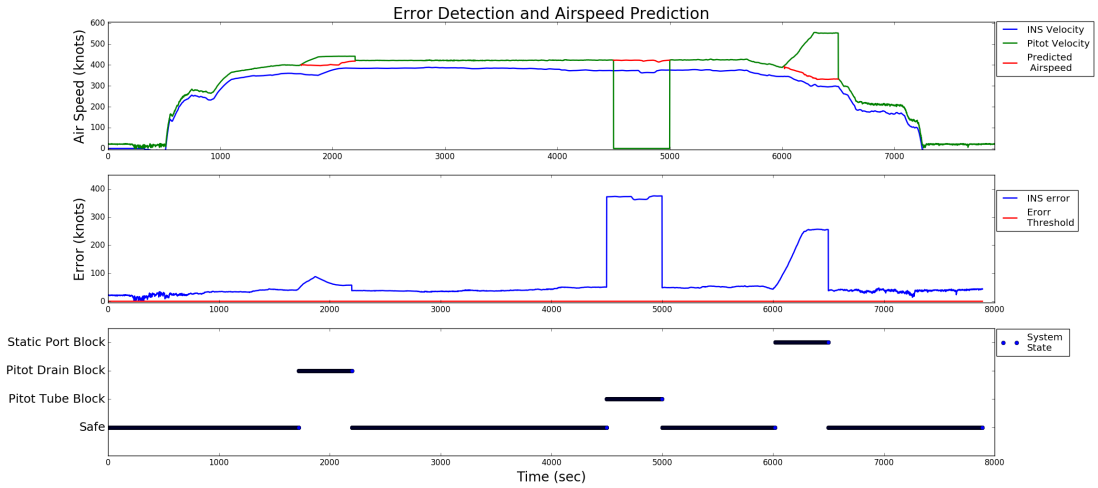


Figure 4.8: Demonstration of our full method. A pitot drain block was encoded from 1700 to 2200 seconds. A pitot tube block was encoded from 4500 to 5000 seconds. A static port block was encoded from 6000 to 6500 seconds. $V_{thresh} = 0$

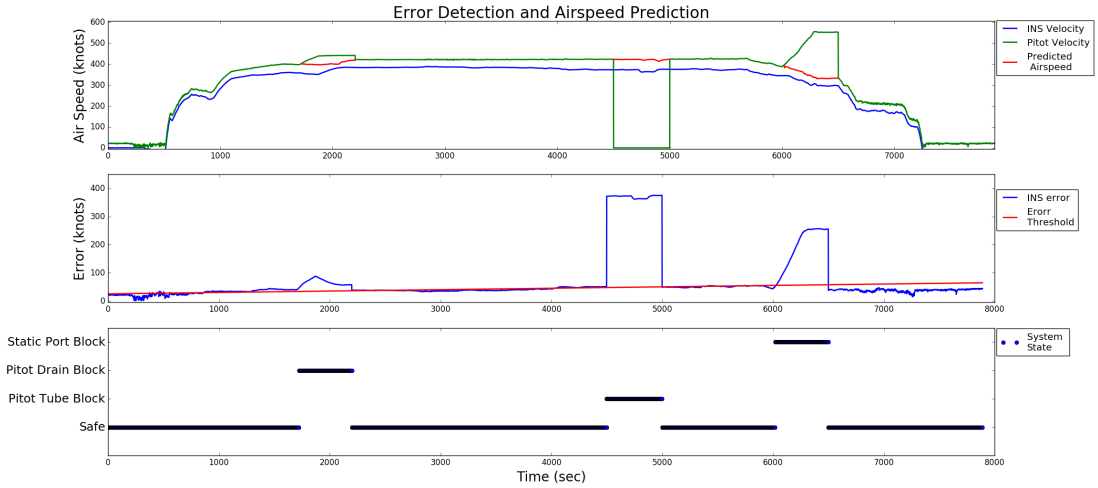


Figure 4.9: Demonstration of our full method. A pitot drain block was encoded from 1700 to 2200 seconds. A pitot tube block was encoded from 4500 to 5000 seconds. A static port block was encoded from 6000 to 6500 seconds. $V_{thresh} = .005t + 25$

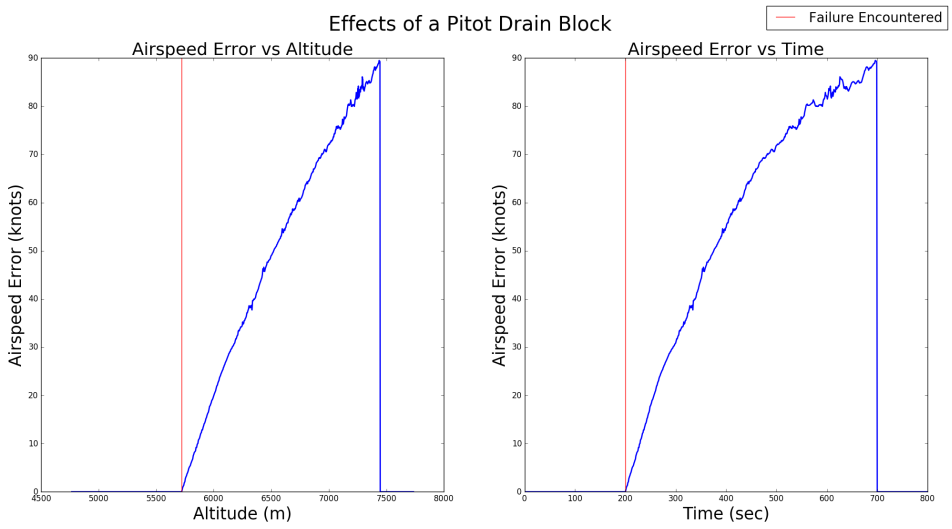


Figure 4.10: The airspeed error produced by the pitot static system as a result of a pitot drain block as a function of altitude (left) and time (right). The red line denotes the time at which the block occurs.

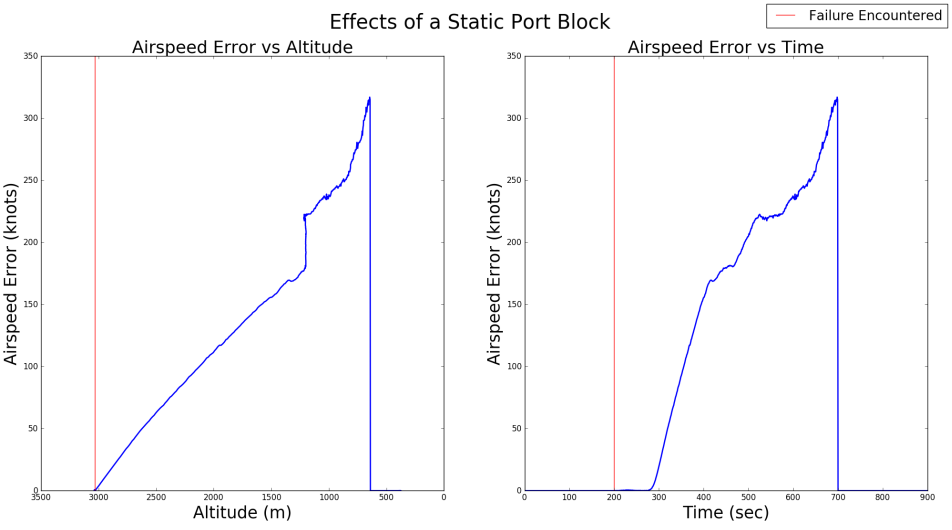


Figure 4.11: The airspeed error produced by the pitot static system as a result of a static port block as a function of altitude (left) and time (right). The red line denotes the time at which the block occurs.

5. CONCLUSION

This work proposed a machine learning approach to online pitot static system error identification and airspeed prediction. An offline library of failure data was created and autocorrelation was evaluated both offline and online to detect a constant pressure signal. Calculating autocorrelation online and comparing it to our offline library, we are able to accurately identify what failures, if any, are occurring within the pitot static system. Using manually produced errors in our test flights, our detection system was able to identify the correct failure mode within 20 seconds of the initial block. Dimensionality reduction and feature subset selection was performed on the high dimensional sensor output. From previous flight data, we collect and store the output of those selected data streams in an offline library. Then in flight, we read in those same streams and use them for airspeed predictions. Using k-nearest neighbors regression we were able to predict airspeed within 30 knots of the true airspeed during the cruising portions of the flight. When a block is encountered during flight, the fault in the airspeed readings will progressively worsen. The pitot drain and static port blocks produce error in airspeed that is proportional to the change in altitude. Having an airspeed prediction that can be incorrect by up to 30 knots is not ideal, but a block in the pitot static system can quickly produce airspeed error of over 50 knots in only 90 seconds.

Our initial goal was to have errors identified within 30 seconds of failure and an airspeed prediction within 10 knots of the true airspeed. This goal was set to motivate the research but these thresholds are important values that our system must incorporate. Future work should be done in analyzing the responsiveness and accuracy required to maintain aircraft safety. Along with this, a more in depth look at the data streams that we have available in flight may provide better predictions of airspeed. The data streams selected

are possibly the most important aspect of a functioning airspeed correction system. The system performed poorly at predicting airspeed during takeoff and landing. These portions of flight are where pitot static system failures have the most effect on airspeed data so it is even more critical to have corrected airspeed at these times. It may be beneficial to do separate feature selection at these portions of flight to produce a more accurate prediction. If this prediction technique can be significantly improved, it may be useful to have this predicted airspeed running throughout the flight.

Having this system on board an aircraft can potentially provide security against a pitot static system failure. Currently, commercial aircraft rely on pilots to make decisions in the event of a failure and more commonly than not, this results in planes going down and lives being lost. If pilots and UAV's can be made aware of unreliable pitot static system data and a more accurate prediction of airspeed is followed, poor decisions can be avoided and aircraft will be able to safely stay in flight.

REFERENCES

[1] S. Hansen, M. Blanke, and J. Adrian, "Diagnosis of uav pitot tube defects using statistical change detection," *IFAC Proceedings Volumes*, vol. 43, no. 16, pp. 485–490, 2010.

[2] D. J. Raymond, C. S. Bretherton, and J. Molinari, "Dynamics of the intertropical convergence zone of the east pacific," *Dynamics*, 2006.

[3] "Air france flight 447 final report," tech. rep., Bureau d'Enquetes et d'Analyses pour la securite de l'aviation civile.

[4] E. Schechter, "Detecting pitot tube obstructions," in *Aerospace America*, American Institute of Aeronautics and Astronautics.

[5] D. Carbaugh, "Flight instrument information-situations and guidance," *Aero Magazine*, vol. Third-quarter, 2003.

[6] R. S. Klockowski, S. Imai, C. Rice, and C. A. Varela, "Autonomous data error detection and recovery in streaming applications," in *Proceedings of the International Conference on Computational Science (ICCS 2013). Dynamic Data-Driven Application Systems (DDDAS 2013) Workshop*, pp. 2036–2045, May 2013.

[7] G. Seidel, T. Golly, and P. Johnson, "Ice resistant pitot tube," Dec. 8 2015. US Patent 9,207,253.

[8] F. A. Administration, ed., *Pilot's Handbook of Aeronautical Knowledge*, ch. 8.

[9] N. Hall, "Pitot-static tube - prandtl tube," 2015.

[10] K. R. Britting, "Inertial navigation systems analysis.," 1971.

[11] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[12] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014.

[13] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, pp. 1–58, 2006.

[14] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[15] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Trans. Comput.*, vol. 18, pp. 401–409, May 1969.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[17] C. R. Reeves, ed., *Modern Heuristic Techniques for Combinatorial Problems*. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[18] E. Cantú-Paz, S. Newsam, and C. Kamath, "Feature selection in scientific applications," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 788–793, ACM, 2004.

[19] S. Cooper and H. Durrant-Whyte, "A kalman filter model for gps navigation of land vehicles," in *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, vol. 1, pp. 157–163, IEEE, 1994.

[20] P. Zarchan, *Progress In Astronautics and Aeronautics: Fundamentals of Kalman Filtering: A Practical Approach*, vol. 208. Aiaa, 2005.

[21] N. DASHlink, “Sample flight data.”

[22] S. A. Glantz and B. K. Slinker, *Primer of applied regression and analysis of variance*. McGraw-Hill, Health Professions Division, 1990.

6. APPENDIX A

Airspeed predictions using engine fan speed, fuel flow, 1st and 2nd angle of attack, longitudinal acceleration, thrust command and thrust target.

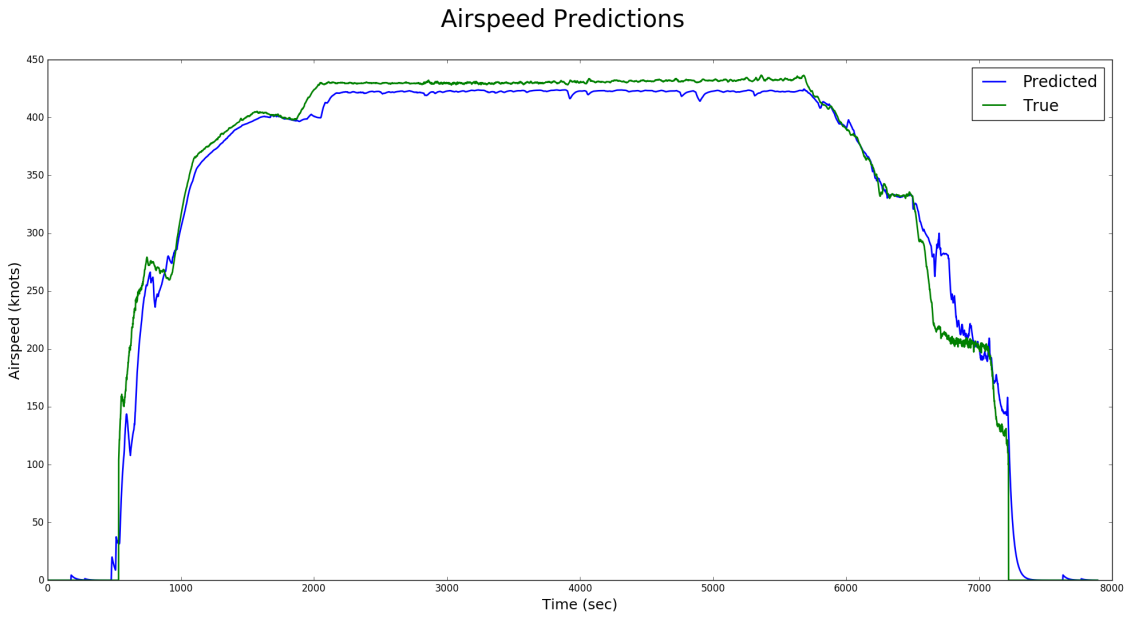


Figure 6.1: Airspeed prediction for flight 1.

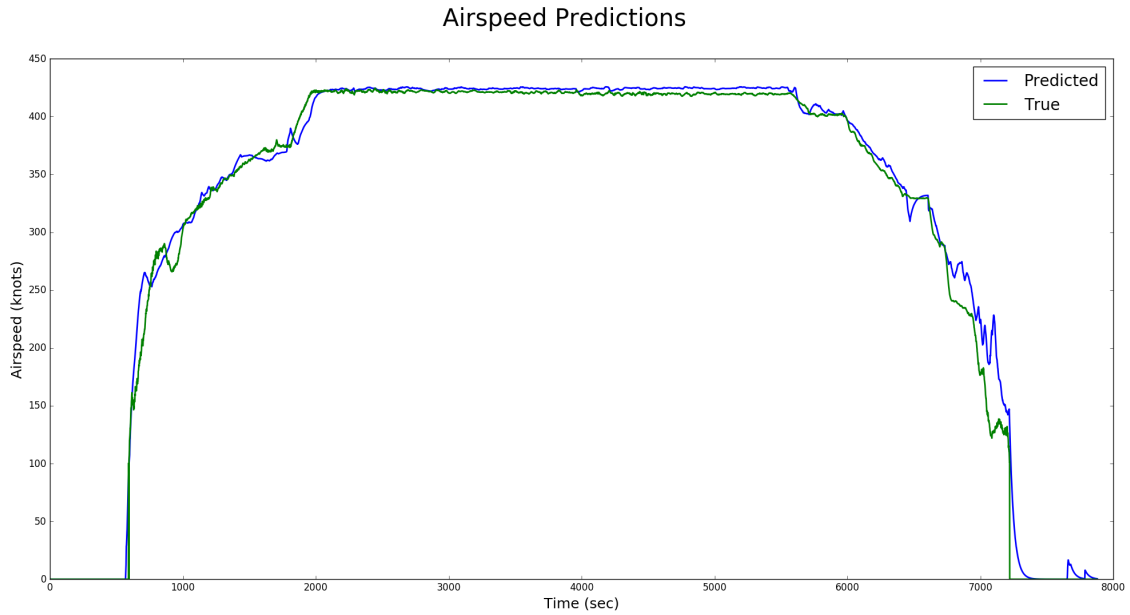


Figure 6.2: Airspeed prediction for flight 2.

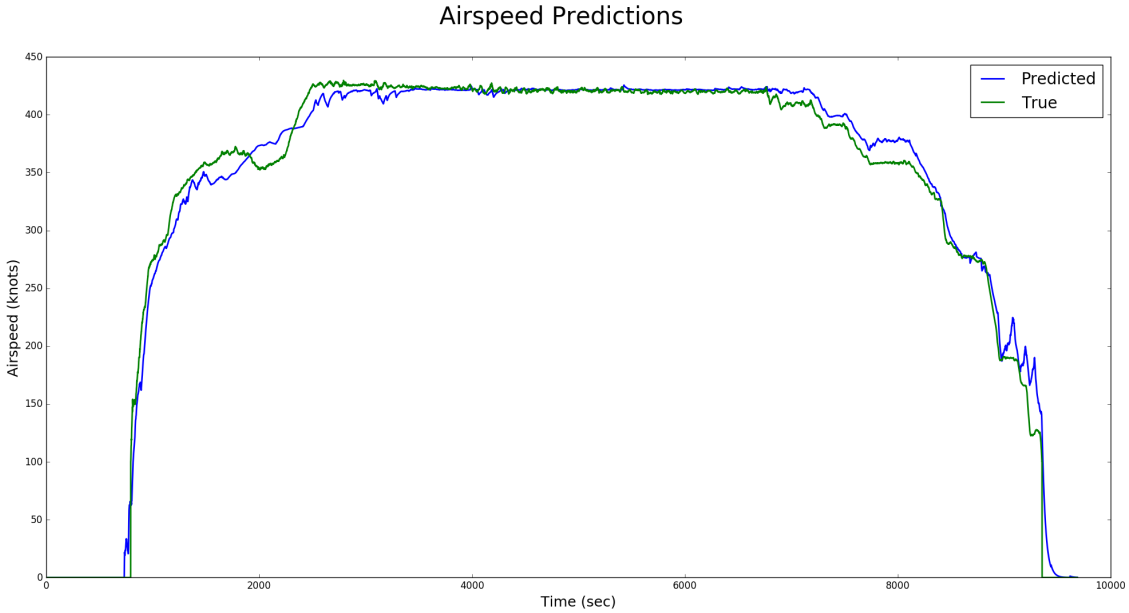


Figure 6.3: Airspeed prediction for flight 3.

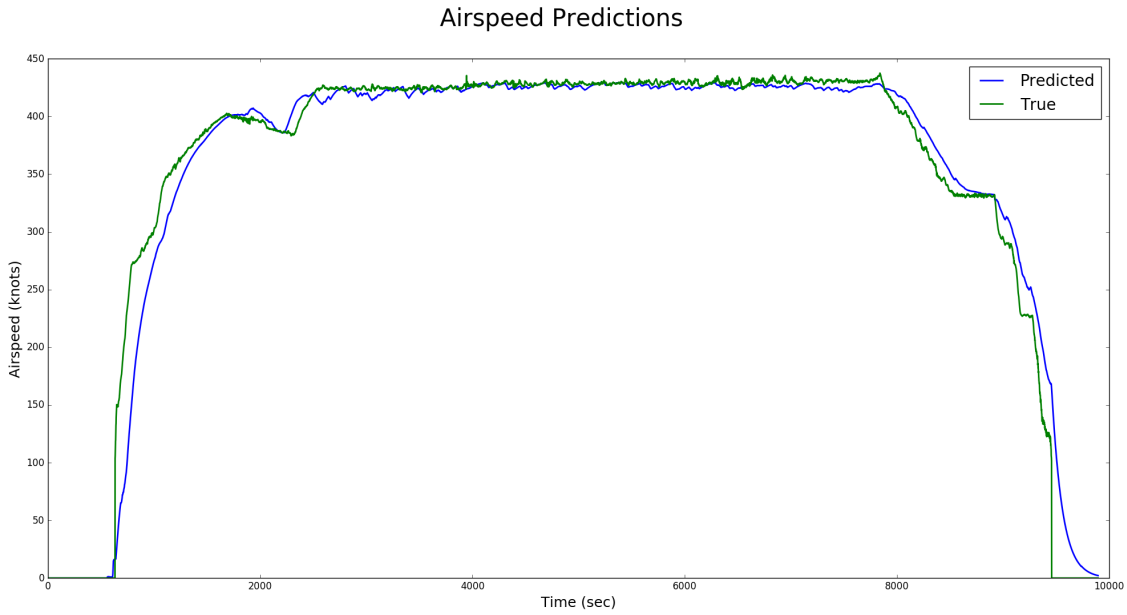


Figure 6.4: Airspeed prediction for flight 4.

7. APPENDIX B

The following figures show the results of our system on real flight data. All of the following figures are formatted as follows. The top plot shows the two running airspeed estimates, the green line is airspeed from the pitot static system and the blue line is airspeed from the accelerometers. The middle plot shows the error in those two airspeeds in blue and V_{thresh} in red. The bottom plot shows the error identification system. When we identify any type of error, our method predicts a new airspeed which is denoted by the red line in the top plot.

Full system performance on various flights with a threshold set to $V_{thresh} = .005t + 25$.

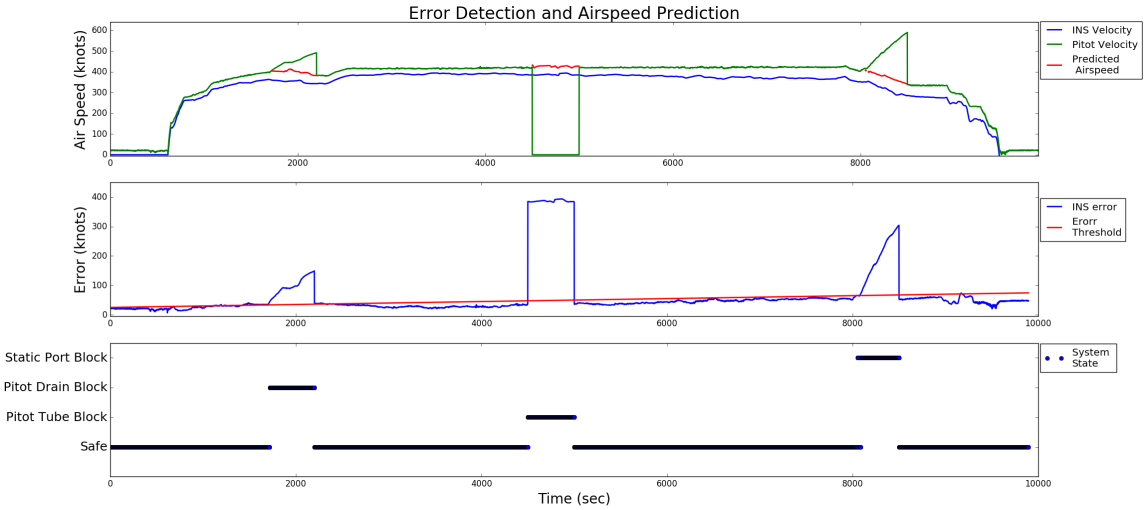


Figure 7.1: Our system performance where a pitot drain block is simulated from 1700 to 2200 seconds, pitot tube block from 4500 to 5000 seconds and a static port block from 8000 to 8500 seconds.

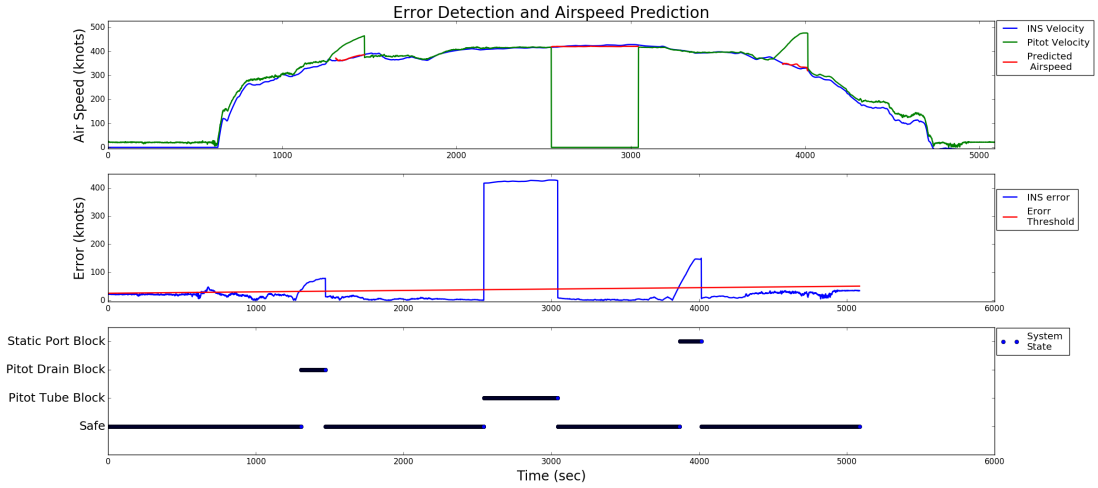


Figure 7.2: Our system performance where a pitot drain block is simulated from 1272 to 1472 seconds, pitot tube block from 2544 to 2744 seconds and a static port block from 3816 to 4016 seconds.

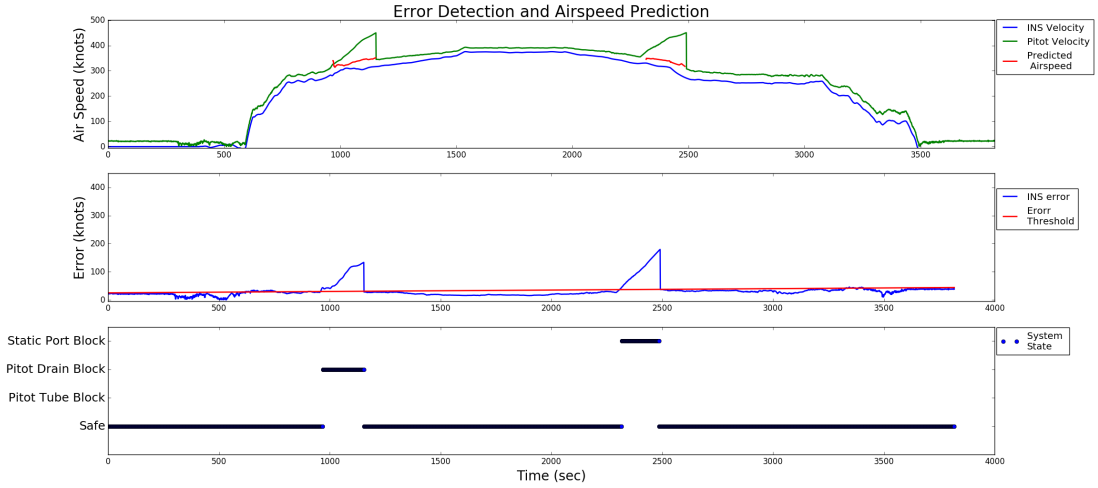


Figure 7.3: Our system performance where a pitot drain block is simulated from 955 to 1155 seconds and a static port block from 2292 to 2492 seconds.

Full system performance on various flights with no threshold set to show blocks may have little effect on airspeed during cruise but can still be detected.

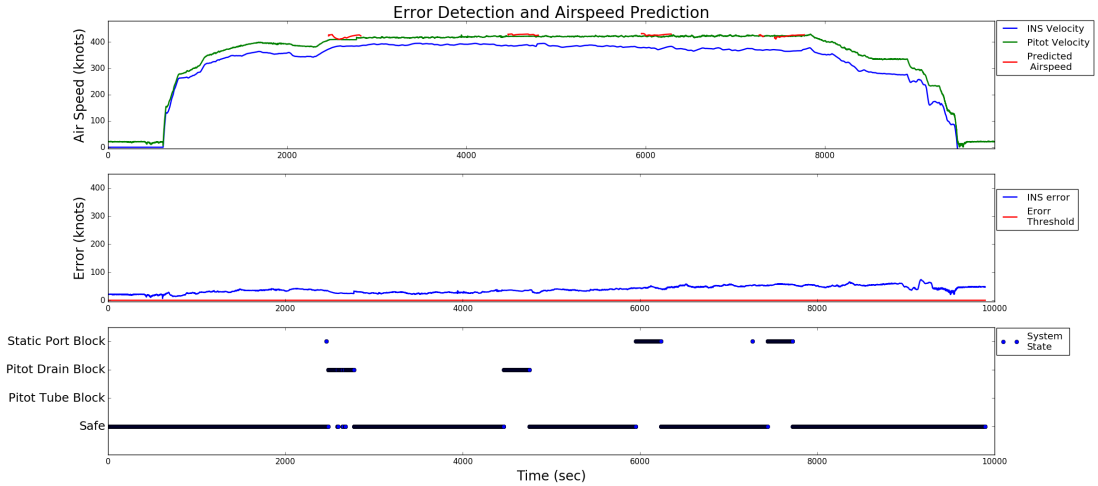


Figure 7.4: Our system performance where two pitot drain blocks are simulated from 2474 to 2774 seconds and 4453 to 4753 seconds and two static port blocks from 5937 to 6237 seconds and 7422 to 7722 seconds.

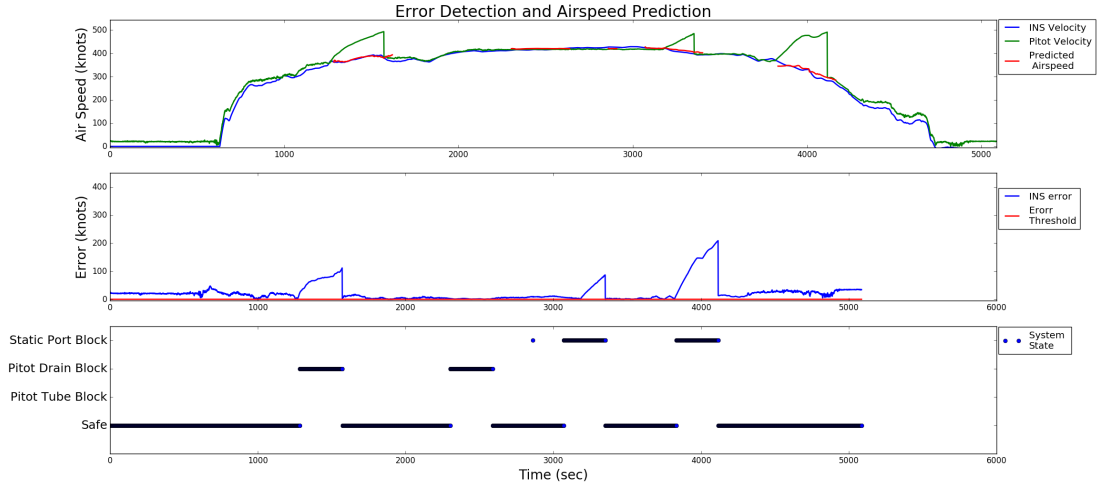


Figure 7.5: Our system performance where two pitot drain blocks are simulated from 1272 to 1572 seconds and 2289 to 2589 seconds and two static port blocks from 3052 to 3352 seconds and 3816 to 4116 seconds.

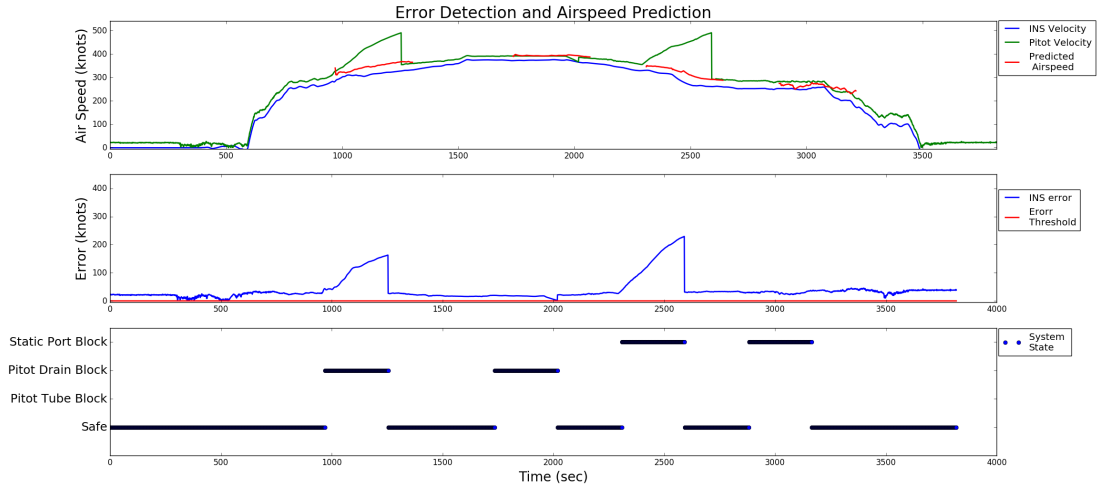


Figure 7.6: Our system performance where two pitot drain blocks are simulated from 955 to 1255 seconds and 1719 to 2019 seconds and two static port blocks from 2292 to 2592 seconds and 2865 to 3165 seconds.