A Machine Learning Approach to Aircraft Sensor Error Detection and Correction

Renee Swischuk* and Douglas Allaire†

Texas A&M University, College Station, Texas, 77843, USA

The pitot static system provides critical airspeed information and consists of two ports located outside of the aircraft making them vulnerable to interference and failures. If an aircraft has access to redundant sensor output, then it can be trained to autonomously recognize errors in faulty sensors and learn to correct them. In this work, we develop a novel machine learning approach to detecting aircraft sensor failures and predicting corrected flight data using an offline/online paradigm. We demonstrate our methodology on flight data from a four engine commercial jet that contains failures in the pitot static system to show the safety benefits of our system in flight. Autocorrelation of incoming pressure data is used to classify the state of the pitot static system. Feature selection is performed on the high dimensional sensor output to create an offline library for predicting airspeed. This library is used to train a regression model to make real time corrections to airspeed data in the event of a pitot static system failure.

I. Introduction

Aircraft performance and guidance is entirely dependent on sensor systems that provide in-flight data. A self-aware aircraft must be able to detect if these sensors become damaged or unreliable and have the ability to decide which sensors can be trusted as a back up source. The goal of this work is to design a machine learning approach for detecting when the pitot static system has become damaged or blocked and

*Department of Mathematics, AIAA Student Member, rswischuk3@tamu.edu.

†Assistant Professor, Department of Mechanical Engineering, AIAA Member, dallaire@tamu.edu.

to determine a set of pitot static independent sensors that can predict a new, more reliable airspeed. This is accomplished by building an offline library of previous flight data that is used to train the aircraft to make critical decisions online regarding the integrity of certain sensor systems. With a system like this in place, damage to sensors can be overcome and aircraft will be able to confidently continue flight.

Improvements have been made to help aircraft autonomously detect sensor data anomalies and recover from temporary loss of information. Using nuclear reactor technology, the Integrated Pitot Health Monitor [1] transforms pressure readings into electronic signals and senses when signals become static to detect obstructions in pitot tubes. It is shown in [2] that fault tolerant air data inertial reference units (ADIRU) supported on-board an aircraft are able to calculate highly reliable ground speed readings, which in conjunction with an accurate measure of wind speed can 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 as done in the P.I.L.O.T.'s software designed by Klockowski et al. [3] to help correct faulty readings from compromised pitot tubes. Failure detection and identification (FDI) is a heavily studied technique involving online prediction or estimation of sensor output to detect failures and damage as show in [4–6]. This method often incorporates a neural network or Kalman filter approach that is able to update online and make estimates of "normal" sensor trajectories to detect when discrepancies occur [4–6]. Although these improvements have made commercial and unmanned aircraft an increasingly safe form of transportation, a system that can also correct this air data could provide an additional level of protection.

These previous works often involve complex nonlinear modeling of an aircraft as well as updating this model online. In a dynamic data-driven approach, we show that a low-dimensional, localized method can be used rapidly online to not only detect anomalies in sensor data but also learn to predict corrected data using redundant sensor data entirely contained on the aircraft. Our approach relies on the hypothesis that when two aircraft observe similar sensor output values, then those aircraft are experiencing similar flight conditions (i.e, similar airspeeds). This hypothesis implies that an aircraft can have the ability to rapidly determine its state by comparing its current flight data to previous flight data. We demonstrate our methodology on failures found within the pitot static system using flight data from a four engine commercial jet. In Section II we discuss the problem setup, including the effects of pitot static system failures and the sensor data that is analyzed for airspeed prediction. In Section III we explain how autocorrelation is used to detect errors

in flight sensors and we outline a feature selection technique used offline to help make improved flight data predictions. The approach is applied to sample flight data containing pitot static system failures and results are presented in Section IV. In Section V we draw conclusions on the performance of the approach and discuss areas of future work, including how this approach can be generalized to a variety of sensor systems on-board an aircraft.

II. Problem Setup

Sensor failure is a common problem faced by both piloted aircraft and unmanned aerial vehicles (UAV's). Military UAV's can be prone to structural damage and commercial aircraft are often susceptible to interferences due to weather. These issues can lead to partial or complete failure of external sensor systems and as a result, in-flight data can become skewed and unreliable. In this section, we explain two types of failures that can occur in the pitot static system and the effects they have on airspeed. We also discuss the redundant sensor output that is analyzed for predicting a corrected airspeed.

II.a. Pitot Static System Failures

The pitot static system consists of two ports; the pitot tube and the static port, both located outside the aircraft. The pitot tube measures total pressure and the static port measures static pressure, and together they determine an aircraft's airspeed. Airspeed can also be calculated analytically using Bernoulli's equation and is proportional to the difference between these pressures as shown below:

$$V^2 = \frac{2(P_t - P_s)}{r} = \frac{2P_d}{r}, \tag{1}$$

where V is airspeed, r is air density, P_t is total pressure, P_s is static pressure and P_d is dynamic pressure. These ports are completely exposed and vulnerable to damage. Damage to the pitot static system can often lead to faulty airspeed measurements and, in certain situations, this airspeed data can be lost entirely. We consider two types of failure; 1) the pitot tube becoming blocked and 2) the static port becoming blocked. For the first case, if the pitot tube has become completely blocked, total pressure will remain constant as no air will be moving in or out of the port, preventing the pitot tube from measuring any new air press. As the aircraft increases in altitude, air pressure will decrease and due to the pitot tube block, this will only have

an effect on the static pressure. This results in a dynamic pressure that is higher than the true value, which then produces an airspeed that is higher than the true value. As the aircraft decreases in altitude, the air pressure will increase resulting in a dynamic pressure that is lower than the true value and thus an airspeed that is lower than the true value. For the second case, when the static port is blocked, the opposite situation occurs. When the aircraft increases in altitude the airspeed produced is lower than the true value and when the aircraft decreases in altitude, the airspeed produced is higher than the true value. An intuitive way to detect if these blocks are occurring is to detect when either of these pressure streams become constant. If we do detect some type of failure, our airspeed will become unreliable and a new, more accurate airspeed should be calculated. In the next section we discuss the redundant aircraft data that is considered for predicting this new airspeed.

II.b. Redundant Sensor Data

If a certain sensor is experiencing a failure, the data produced by that sensor is unreliable. If we have independent sensor data available, we may be able to correct this faulty data with the help of these other sensors. In the case of a pitot static system failure, we need a collection of sensors that are unrelated to the pitot static system, and therefore unaffected by the failure, that can be used to accurately predict airspeed as a back up. Although, the sensors chosen must have some relationship to airspeed, otherwise predictions made by these sensors will be just as unreliable. The task is to determine which of the sensors on an aircraft, aside from the pitot static system, have a relationship to airspeed. Flight dynamics can give us an insight as to which sensors are reasonable to consider first, then we can further analyze the relationship to airspeed mathematically. In the remainder of this section, we discuss some physical relationships that exist between aircraft sensor output and airspeed. We also visualize these relationships with data collected onboard a single type of four engine aircraft that has been made available by NASA [7].

Airspeed is dynamically changing during flight but is shown to be related to actions the aircraft performs such as changes in thrust, angle of attack, acceleration, etc. We can analyze relationships to this motion by considering the aircraft as a simple 3 dimensional object. With the aircraft's center of gravity as the origin, an aircraft can be modeled as shown in Fig. 1. The axis parallel to the nose, in the direction of flight is the longitudinal (x) axis, the axis perpendicular to flight along the wingspan is the latitudinal (y) axis and the

axis facing out of the bottom of the aircraft, perpendicular to the other axes is the vertical (z) axis.

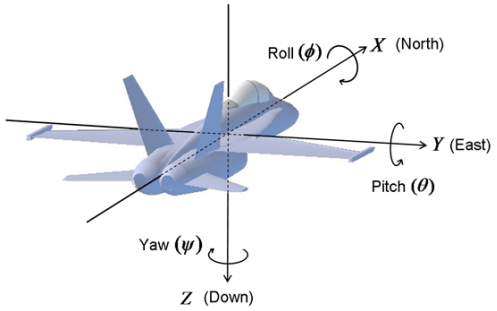


Figure 1. The coordinate system of an aircraft [8]

Airspeed is defined as the difference between the aircraft velocity vector (ground speed) and the wind speed vector. The velocity can be explained by the forces applied to the aircraft, such as thrust from the engine, and the angle of these forces. Another value related to this velocity is acceleration. We can visualize how changes in acceleration correspond to changes in airspeed in Fig. 2(a), which shows how low values of longitudinal acceleration correspond to high values of airspeed during the climb portion of flight. Accelerometers on an aircraft are contained within the inertial navigation system and are safe from outside interference. Thus, they generally produce robust data with a strong relationship to airspeed. Another useful set of data collected by the inertial navigation system is aircraft attitude in these three dimensions. Unless the aircraft is moving completely horizontally, the airspeed must be broken down into the components that describe the direction of flight such as angle of attack, yaw, magnetic heading, altitude rate, pitch and roll [9]. Angle of attack is of particular interest due to its relationship to airspeed. As explained in [9], angle of attack must be adjusted with changing airspeed to maintain level flight, therefore changes in angle of attack should provide some indicator of changes to airspeed. This is further visualized in Fig. 2(b), which shows low degrees of angle of attack corresponding with high airspeeds.

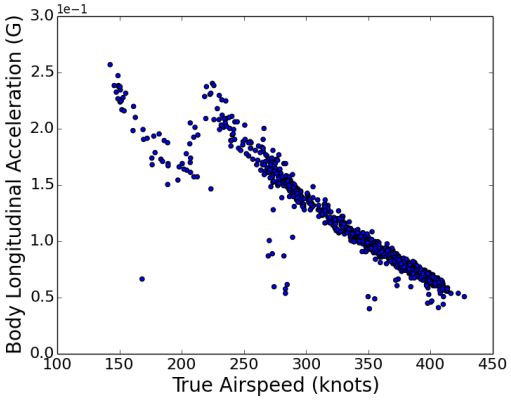
The force that initiates the motion of the aircraft is thrust from the engine. The data we consider in this work is from an aircraft with four turbofan engines that generate this thrust. As the aircraft flies, air is passed through the fan and mixed with fuel to create combustion that sends exhaust through the engine core and fan turbines producing this thrust [10]. So, thrust, and therefore aircraft motion, can also be explained with information on fuel flow, engine core speed, engine fan speed and the thrust power lever angle. Figure

2(c) shows how an increase in thrust command during cruise creates an increase in airspeed. Figure 2(d) shows a decrease in fuel flow during climb corresponds to an increase in airspeed as the aircraft reaches cruising speed. Figure 2(e) shows how a decrease in altitude during approach corresponds to a decrease in airspeed as the aircraft slows to prepare for landing. The effects of wind speed on an aircraft should also be considered. Although wind speed cannot be collected directly in flight, the drift angle can be recorded to determine the effect of wind on the direction of flight.

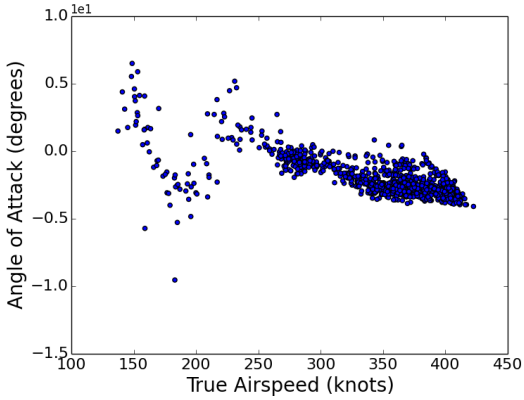
Based on this analysis, we chose 24 sensors whose output can potentially capture an aircraft's changing airspeed, shown in Table 1. We collect these sensors output for multiple past flights and store them in a library offline. While Fig. 2 shows a relationship between aircraft sensor output and airspeed, none of these alone are capable of predicting airspeed throughout an entire flight. A specific sensor's output may be correlated with airspeed during one portion of flight, but uncorrelated during another. As a result, our hypothesis is that airspeed can be predicted using a combination of sensor outputs that are selected specifically for use during a particular portion of flight.

Table 1. The 24 sensor outputs considered for airspeed predictions. Units shown in parenthesis.

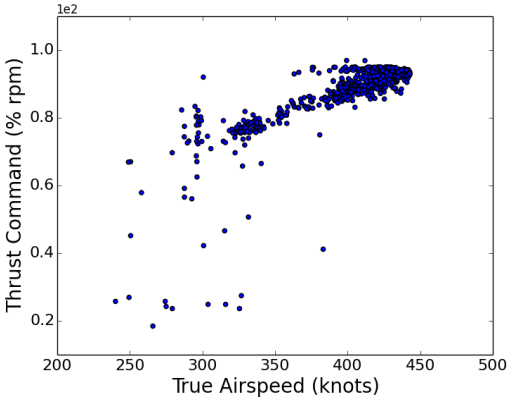
Yaw (degrees)	Pitch (degrees)	Roll (degrees)
Vertical Acceleration (G)	Engine Core Speed (%RPM)	Engine Fan Speed (%RPM)
Power Lever Angle (degrees)	Fuel Flow (pounds/hour)	Altitude Rate (feet/minute)
1^{st} Angle of Attack (degrees)	2^{nd} Angle of Attack (degrees)	Body Longitudinal Acceleration (G)
Cross Track Acceleration (G)	Drift (degrees)	Flight Path Acceleration (G)
Inertial Vertical Speed (feet/minute)	Latitudinal Acceleration (G)	Longitudinal Acceleration (G)
Magnetic Heading (degrees)	Thrust Command (%RPM)	Thrust Target (%RPM)
True Track Angle (degrees)	Magnetic Track Angle (degrees)	Phase (climb,cruise,approach)



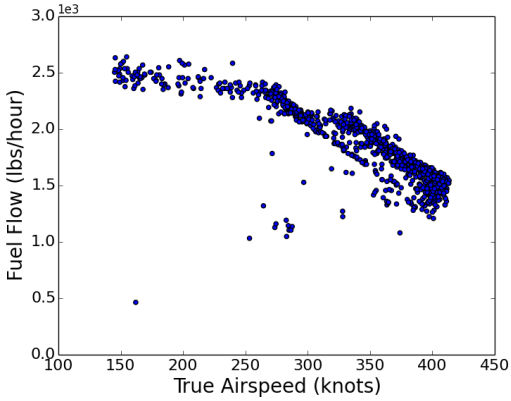
(a) Body longitudinal acceleration vs. airspeed during climb.



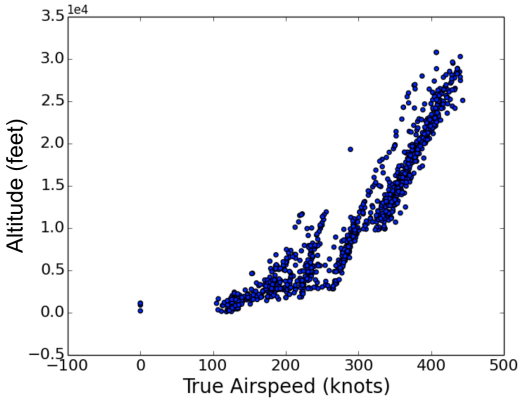
(b) Angle of attack vs. airspeed during climb.



(c) Thrust command vs. airspeed during cruise.



(d) Fuel flow vs. airspeed during climb.



(e) Altidude vs. airspeed during approach.

Figure 2. Relationship between various sensor outputs during different portions of flight.

III. Approach

For an aircraft to autonomously detect sensor failures, it must first learn how a failing sensor behaves. Similarly, if the aircraft is to make predictions of corrected flight data it must be able to decide which sensors can make accurate predictions of this data. In this section we present the methods used to develop an error detection system that is able to identify two types of pitot static system errors and predict a corrected airspeed. Our system uses an offline/online approach as developed in Ref. [11–13].

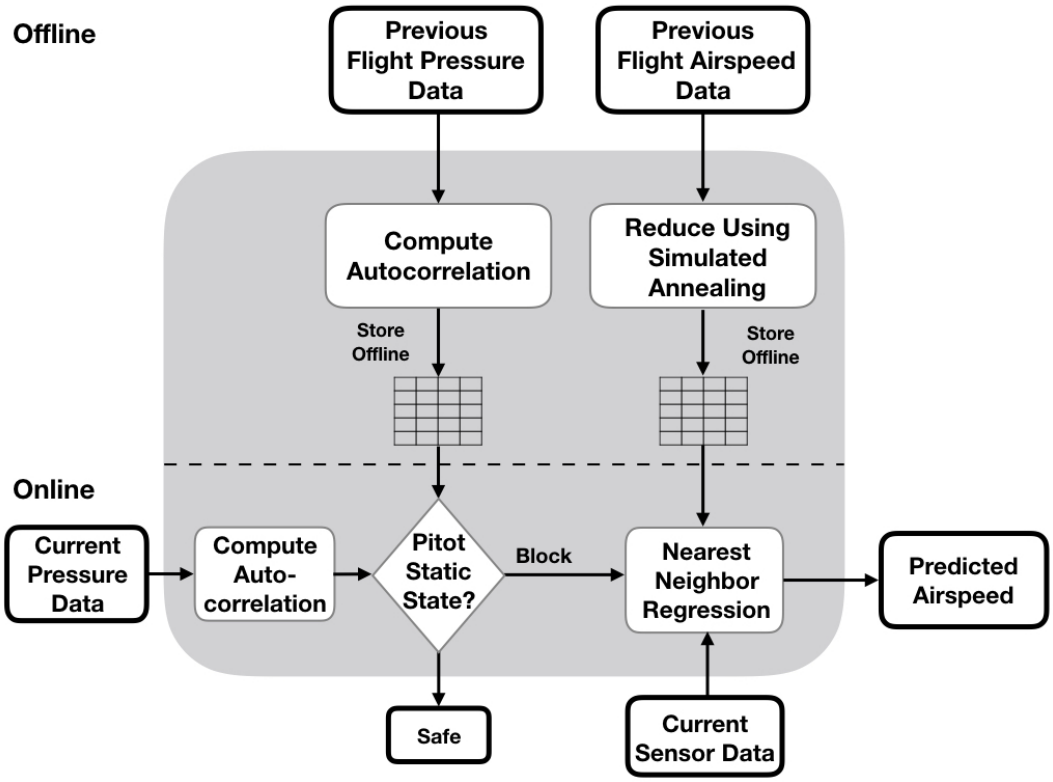


Figure 3. The offline (upper) and online (lower) portions of our approach.

An illustration of our approach is shown in Fig. 3. The offline state occurs prior to flight, when data is collected, analyzed and stored in offline libraries. During the online state, when the aircraft is in flight, decisions about the status of the aircraft are made by comparing current flight data to that found in the offline libraries. If a pitot static block is occurring, sensor information is collected to make rapid predictions of current airspeed.

This section describes the approach taken for error detection and airspeed prediction. We explain how autocorrelation of pressure data can be calculated to determine the state of the pitot static system. Following

this, we discuss the feature selection algorithm that is performed on the high volume of sensor data and how this sensor data is combined to form an offline library used for predicting airspeed online in the event of a failure.

III.a. Error Detection of Sensor Data

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 two pitot static system failures described in Section II. This data was created by changing the values of the total or static pressure streams and calculating airspeed using Bernoulli's equation (Eq. 1). When either of the two failures occur, the respective pressure streams become constant as air is no longer flowing through the ports. The sensor data should inherently be noisy in a safe situation due to the high speeds of the aircraft, therefore, a system that can differentiate between a constant and noisy signal would provide a good indicator of when one of these failures is occurring.

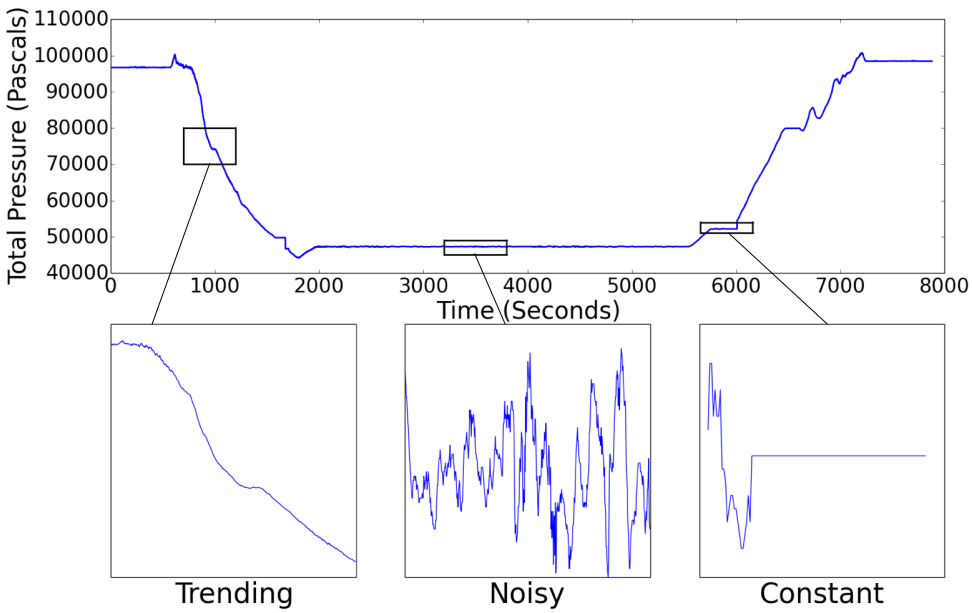


Figure 4. Three general ways a total pressure stream behaves during flight.

During flight, the stream of pressure data only behaves in a few distinct ways. Figure 4 illustrates the total pressure throughout an entire flight. We can see that the pressure will be trending with slight amounts

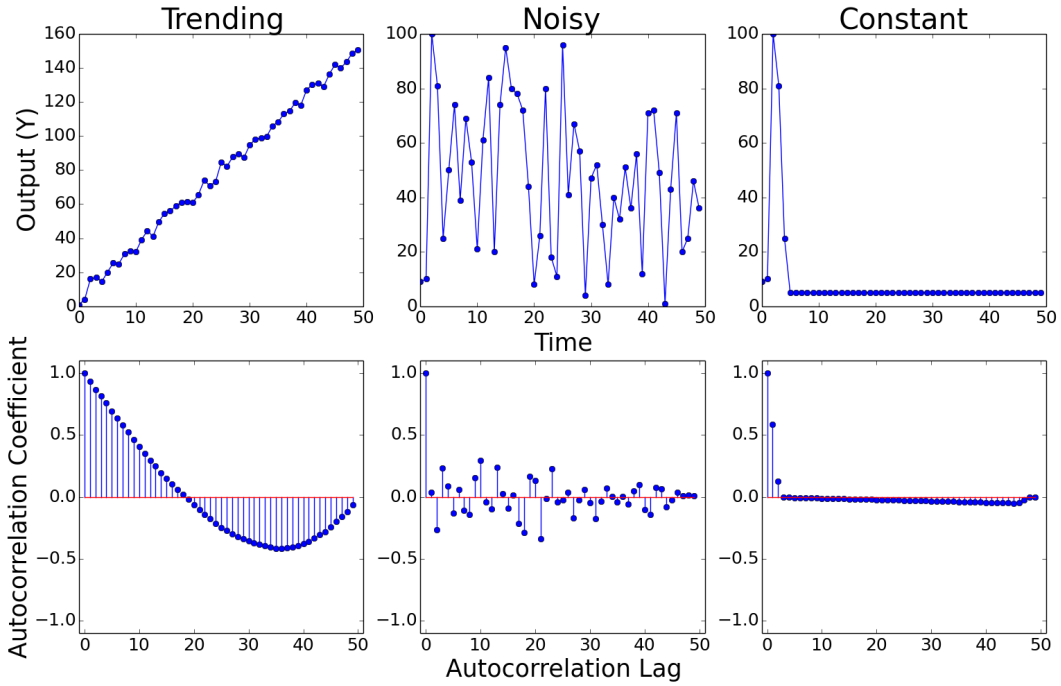


Figure 5. Autocorrelation behavior of three types of data signals.

of noise during takeoff and approach, noisy but somewhat steady during cruise or constant when a block is occurring. A common approach to detecting constant signals is to calculate the autocorrelation. The statistical definition of autocorrelation of a set of points $Y = \{y_1, ..., y_n\}$ is defined as:

$$ac_j = \frac{\sum_{k=1}^{n-j}(y_k - \bar{y})(y_{j+k} - \bar{y})}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{2}$$

where j is the autocorrelation lag index and $\bar{y}$ is the sample mean of the signal, respectively.[14] As shown, autocorrelation is undefined for constant valued signals since the denominator will be zero for completely blocked ports. Since we do not consider partial blocks where the port can still allow air in, producing noisy data, we account for this division by zero by adding 1 to the denominator. This allows the autocorrelation to exist while still keeping all values equivalently scaled.

We compare the autocorrelation behavior for a trending, noisy and constant signal in Fig. 5. There is a clear distinction between the constant signal vs. a trending or noisy signal. This will allow us to define a certain characteristic to detect when a block may be occurring, namely, the autocorrelation values.

Autocorrelation lag values for total and static pressure are calculated for both types of failures and safe flights. The lag values for static pressure are stored in one library and the lag values for total pressure are stored in a different library. Each entry in the libraries is assigned a label of pitot tube block, static port block or safe. The two types of blocks we consider only effect one pressure stream at a time. During a pitot tube block, only total pressure is effected and during a static port block only static pressure is effected. Thus, during a pitot tube block, the label associated with the total pressure autocorrelation lags is a pitot tube block and the corresponding static pressure lags are assigned a label of safe since this data is uneffected by the block. Similarly, during a static port block, the total pressure autocorrelation lags are assigned a label of safe and the static pressure lag values are assigned a label of static port block.

Online, while the aircraft is in flight, autocorrelation lag values of incoming total and static pressure are calculated in a moving window of size n. Those values are compared to the autocorrelation lags found in the two offline libraries where we determine the two samples that are closest to them in value using Euclidean distance (i.e., the nearest neighbor). Each of our online samples receive the label of their nearest neighbor. In the event of a pitot tube block, the total pressure label should be pitot tube block and the static pressure label should be safe. Thus, the only time we indicate that there is a pitot tube block is when our online total pressure lags return an output label of pitot tube block and our online static pressure lags return a label of safe. Similarly, we only indicate a static port block when our online total pressure lags return a label of safe and our online static pressure lags return a label of static port block. All other outputs are considered safe.

This calculation and comparison can be done online in linear time, $\mathcal{O}(tnm)$, where t is the total time of flight, n is the size of the moving window, and m is the size of the offline library. To reduce overall computational costs, it may be beneficial to make a second estimate of airspeed from the accelerometer and define an error threshold to determine when to check for failures as opposed to checking throughout the entire flight. Accelerometer data is prone to drift so this threshold would allow the accelerometer airspeed to diverge from the pitot static system airspeed without raising alarm [15]. Autocorrelation would be calculated and the state of the pitot static system would be determined only in the case that the two airspeeds diverge by more than the threshold. Although this would reduce runtime online, it would only be reduced by a constant factor and would require a system specific threshold to be calculated.

III.b. Online Flight Data Correction

In the event of sensor failure, flight data will become unreliable or incorrect. If an aircraft determines that a failure is occurring, it must be able to estimate new flight data and decide which sensors to follow. The pitot static system controls the airspeed indicator so a failure in the system will result in faulty airspeed readings. Sensors that are independent of the pitot static system and contained within the aircraft may be able to provide reliable information about the current airspeed without directly measuring it. We currently have a collection of 24 sensor outputs, which is a somewhat large amount of data to store offline and to collect and process online. With the high dimensionality of this dataset, there is the possibility of a lower dimensional subspace existing that is able to capture the trends in the data. In the following subsections, we discuss our rationale for reducing the dimensionality of the dataset using a multidimensional scaling method. We also explain our implementation of an offline feature selection technique that decides the most informative sensors for predicting airspeed using previous flight data and how these predictions are made online.

III.b.1. Sammon's Mapping

The hundreds of sensors found on an aircraft are narrowed down to those mentioned in Section II.b that capture information about the motion of the aircraft. The output data from each of these sensors is denoted as a feature and stored in a data set. The data set consists of a set of previous flight information and with this data, in the event of a pitot static system failure, our goal is to make accurate predictions of airspeed continuously until the failure has been resolved. If we are able to relate our online aircraft state (current fuel flow, engine power, accelerations, etc.) to offline flights, we can use the airspeed from these offline flights to aid in making predictions of airspeed online. Based on the relationship between aircraft sensor data and airspeed found in Section II.b, the similarity measure we will use to relate current sensor state to offline data is Euclidean distance.

The 24 features chosen provide useful information about the motion of the aircraft. However, this generates a high dimensional data set and too much data can often have negative effects on predictions. According to Aggarwal et al. [16], in a high dimensional data set, the ratio of distances between nearest and farthest neighbors approaches 1. Thus, as discussed in [16], Euclidean distance can lose meaning in high dimensions, resulting in an ill defined nearest neighbors problem. Another drawback to high dimensional

data is the compute time for predictions. Airspeed predictions must be made rapidly to keep an aircraft safely in flight, so we need to minimize the compute time as well as maintain accuracy of these predictions. One way to reduce the compute time and ensure the accuracy of predictions is to reduce the dimensionality of our dataset.

To determine if the dataset has the potential to be reduced, we consider multidimensional scaling [17], which is a method used to visualize high dimensional data by projecting it into lower dimensions in a way that preserves similarities between points. Recall that our hypothesis was that similar flight data corresponds to similar airspeed, therefore, any reduction in the dimensionality of our data set should preserve relative distances between points. Sammon's mapping is one multidimensional scaling method that quantifies how well a projection preserves pairwise distances and overall structure of the data set [18]. The Sammon's mapping cost function is defined as follows:

$$Cost = \frac{1}{\sum_{i<j}^{n} d_{ij}^*} \sum_{i<j}^{n} \frac{[d_{ij}^* - d_{ij}]^2}{d_{ij}^*}, \tag{3}$$

where d_{ij}^* is the Euclidean distance between x_i and x_j in the original space and d_{ij} is the Euclidean distance between points y_i and y_j in the projected space. Additionally, 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 $i, j \in \{1, ..., n\}$. This function is a measure of how the distances between points in the two spaces compare. When the function evaluates to a small value, the distances are being preserved in the reduced space (i.e. $|d_{ij}^* - d_{ij}| \approx 0$). When the function evaluates to a large value, then the distances between points are not being preserved (i.e. $|d_{ij}^* - d_{ij}| > 0$). For our problem, a lower dimensional projection is defined by removing certain features from our data set. For example, to reduce from 24 features to 20, we ignore 4 features of our dataset and redefine our coordinates for each point by removing these 4 dimensions. Figure 6 shows the Sammon's mapping function evaluated on our data set when reduced from dimension 24 to dimensions 3 through 22. The cost at each dimension was minimized using an iterative steepest descent algorithm [17]. This process was not an exhaustive minimization, thus the cost at each dimension may differ if performed again using different stopping criteria. The function evaluation value is denoted on the y-axis as "Average Structure Loss (Cost)" where small values indicate pairwise distances being preserved. The x-axis denotes the size of the reduced dimension we are projecting onto. There is a degree of freedom in choosing what values of cost are acceptable. The cost is able to remain under .01

with a dimension as low as 5. This indicates a minimal amount of distortion between points in these lower dimensions. On the other hand, when we project our data into dimensions less than 5, we can see that we quickly begin changing these pairwise distance values. High dimensional data points may be far apart in their original dimension but when we reduce the dimensionality of these points so drastically, they can end up being almost identical in this lower dimension. It is ideal to find the middle ground where our dimensionality is low but still preserves adequate distance between points. We can see from Fig. 6 that there does exist lower dimensions that preserve essentially as much distance as our full dimensional case. This result therefore provides rationale for performing feature selection to reduce the dimensionality of the data.

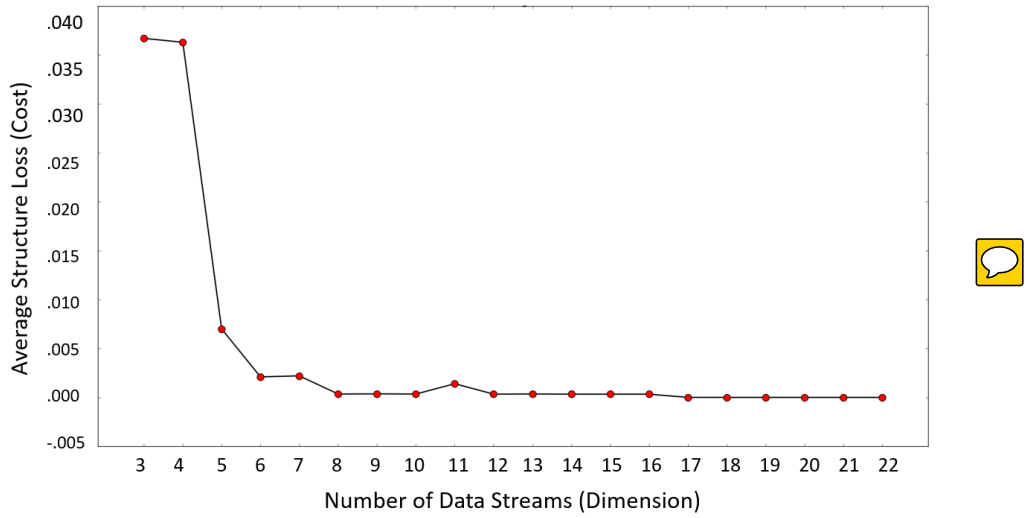


Figure 6. Minimization of Sammon's Mapping function vs. dimensionality.

III.b.2. Feature Selection

Due to the differences in engine power necessary in different portions of flight, we divide our offline flight data into three sections; climb, cruise and approach. The climb portion of flight is the time frame in between takeoff and cruising when the aircraft is gaining altitude and speed. The cruising portion of flight has a mostly constant airspeed and altitude. The approach portion is immediately after cruise when the aircraft begins to descend and decrease airspeed. The airspeeds are changing at very different rates for each of these sections of flight so a generic feature selection over the entire flight will produce suboptimal airspeed predictions. To assess the accuracy of our airspeed predictions, we define an error measure. During flight, an

inaccurate airspeed can cause major issues even if the inaccuracies last only seconds. For this reason, we set our error as the maximum difference at any time between the predicted airspeed and the true airspeed over the entire flight. An airspeed prediction that performs well on average can potentially produce very poor predictions at times. The following error bound will allow us to quantify our performance with a confident upper bound as opposed to an average:

$$error_{prediction} = \max_{t} |v_t - \hat{v}_t|, \tag{4}$$

where v_t is the true airspeed and $\hat{v}_t$ is the predicted airspeed at time t. Minimizing this error requires finding the set of features that can produce a minimal prediction error. This results in a combinatorial optimization problem. This is due to the fact that we are keeping our features physical meaning as opposed to methods such as Principal Component Analysis [19] that produce new features that are linear combinations of the original features. Our solutions to this optimization problem will be discrete sets of features and the function we want to minimize is the prediction error (Eq. 4). To solve this problem we must search through the finite set of possible features and move towards sets that produce low prediction error.

We found the set of features that minimized the prediction error using simulated annealing. Simulated annealing was founded on the idea of heating and cooling of materials proposed by Kirkpatrick, Gelatt and Vecchi [20]. The method randomly moves along the objective function, calculating costs and assigning acceptance probabilities to solutions, then searches in a "neighborhood" of these solutions to determine if there is a "nearby" solution that produces a lower cost. A neighborhood around a point $\vec{x}$ is defined as the collection of points that share all but two components with $\vec{x}$. Any point in this collection can be chosen as a nearby point. The randomness that the method employs distinguishes it from gradient based methods and allows it to escape local optimum, increasing the chances of producing the globally optimal solution. The solutions to our optimization problem are the indices of the features that minimize the error from Eq. 4. This error is calculated by predicting airspeed with the chosen set of features using a nearest neighbor regression model. The regression model is explained in detail in the next section. Each dimension size, d, is treated as its own optimization problem, where the solution domain consists of all sets of indices of size d. Due to the large computational complexity of this problem, we chose dimensions 7,8, and 9 to perform simulated annealing feature selection over as a result of our Sammon's mapping analysis.

III.b.3. Prediction Method

To quantify the performance of different sets of features and produce corrected flight data, we define a prediction algorithm. Each feature represents the output data from a given sensor located on the aircraft. This data comes from individual sensors and thus, we make the assumption that each feature is independent. However, any other assumptions about this data, such as the distribution of noise, are not plausible without further information on the specific sensors used. In Section II.b, we showed that similar sensor values resulted in similar airspeed values. To take advantage of this distance based relationship between sensors and airspeed and to refrain from making any assumptions about the sensor data leads us to the non-parametric, distance based, k-nearest-neighbors approach. Using a non-parametric method allows us to avoid making any assumptions about the underlying distribution of sensor data and the distance based prediction method has the potential to capture the relationship we found between sensor values and airspeed.

The k-nearest-neighbors regression algorithm works by finding the k closest training inputs to a test input, and predicts the output of the test case to be the weighted average of the k training outputs. This is in essence a localized linear regression model. Our local region is defined by the number of neighbors chosen and our output can be defined by $\sum_{i=1}^{k} \frac{v(i)}{|d_i|}$, where $v(i)$ is the airspeed associated with the i^{th} neighbor and d_i is the distance between the test point and the i^{th} neighbor. In our case, an input is a vector in $\mathbb{R}^p$ that contains the output of p different sensors at a given time, t, and an output is the airspeed value at the same time t corresponding to the sensor values. An example input could be $s_t = [Yaw_t, FuelFlow_t, ThrustCommand_t]$ representing the output from the sensors that measure yaw, fuel flow and thrust command, respectively, at time t. A corresponding example output could be 350 knots representing the airspeed at time t. Our training input set would consist of a collection of vectors in $\mathbb{R}^p$ containing output from the p sensors at different times of flight and the training output set would contain a collection of airspeed values that correspond to the inputs. These training data are stored in another offline library.

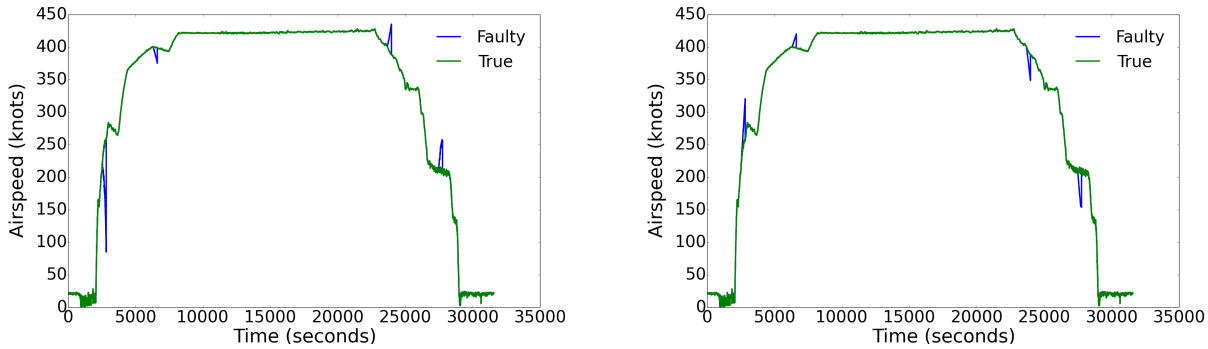
To find the airspeed for a given test input, $\vec{s}_{test} \in \mathbb{R}^p$, we find the k closest training inputs in our offline library using Euclidean distance in $\mathbb{R}^p$. The Euclidean distance between two points, $\vec{s}_i$ and $\vec{s}_j$, is defined as

$$\sqrt{\sum_{k=1}^{p} (s_{i,k} - s_{j,k})^2}.$$

The predicted airspeed is the average of the airspeeds that correspond to those k closest neighbors.

IV. Demonstration and Results

Our method is demonstrated using data from a four engine commercial jet collected from the NASA DASHlink [7] data repository that contains manually produced errors from pitot static system failures. The aircraft data is evaluated every second to simulate an online flight data stream. This section begins with a discussion of the two pitot static failures that were implemented and the results of the failure detection system. Following this, the results of our airspeed predictions are shown using the sensors found with simulated annealing feature selection. Finally, we present the entire method, which simultaneously detects pitot static failure and predicts a new airspeed.



(a) Static Port Block.

(b) Pitot Tube Block

Figure 7. The effects of holding static pressure (left) and total pressure (right) constant for 100 seconds during 4 sections of flight.

We manually created two offline libraries of faulty pitot static system data to simulate these blocks. The library was created by holding the values of the total or static pressure constant and recalculating airspeed using Bernoulli's equation (Eq. 1). Each of the blocks were simulated for 100 seconds at four different times of flight. For a pitot tube block, we hold total pressure constant and for a static port block, we hold static pressure constant. The effects of these blocks on airspeed can be seen in Fig. 7.

We calculate the autocorrelation lag values for each of the 100 seconds of faulty pressure readings using Eq. 2 and store the lag values in the two libraries. Then the lags in each library are assigned labels according to the description in Section III.a.

Feature selection was performed with simulated annealing as described in Section III and the features chosen for each section of flight are listed in Table 2. The predictions for a single flight using these features

Table 2. Features selected for each phase of flight and the prediction error for each.

Flight Phase	Features	Max Error (knots)
Climb	Yaw, Roll, Engine Core Speed, Power Lever Angle, Fuel Flow, 1^{st} Angle of Attack, 2^{nd} Angle of Attack, Body Longitudinal Acceleration, Flight Path Acceleration, Longitudinal Acceleration, Magnetic Heading, Thrust Command, Magnetic Track Angle, True Track Angle	40.5
Cruise	Yaw, Pitch, Engine Core Speed, Engine Fan Speed, 1^{st} Angle of Attack	32.4
Approach	Thrust Command, Thrust Target, Drift Angle, Fuel Flow	84.5

is shown in Fig.8. The predicted airspeed was within 41 knots of the true airspeed during climb and cruise while it reaches an error of 84.5 knots in the approach portion when tested on various flights. A topic for future work is understanding the airspeed error associated with the approach portion of flight. One thing to consider is that k-nearest neighbors is a *lazy* learning algorithm, according to Ref. [21]. That is, generalizations are not made about the training data offline and all of the data is analyzed online for each prediction. It is shown in Ref. [22] that an *eager* learning method, such as a support vector machine, that generalizes the training data by only storing its support vectors, is able to store less training data and make generalized predictions about the data. During the approach portion of flight, many changes are being made to engine power and attitude that may be difficult for a lazy learner to understand. Eager learning methods may be able to capture dynamic patterns that a lazy learner may miss by making generalizations about the data set. This approach may also reduce the dependency on the particular training set used.

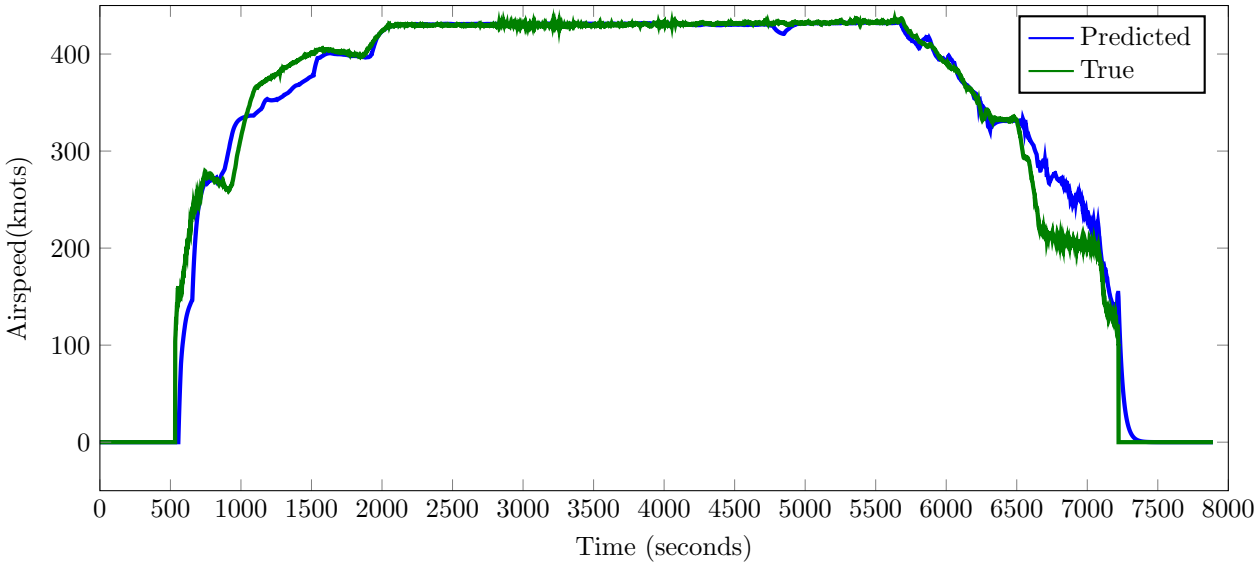


Figure 8. Airspeed prediction for a single flight.

The performance of the overall system can be found in Fig. 9. In this figure, the top plot shows the error detection system, which indicates a pitot tube block (PTB), a static port block (SPB) or safe. Two 300 second pitot tube blocks and one 300 second static port block were simulated during flight and these blocks were detected within 20 seconds of the time they were encountered. The true time frame of the block is denoted by the shaded segments. Anytime this system predicts a block is occurring, a new airspeed is predicted. The bottom plot shows the error in the airspeed produced by the pitot static system compared to the predicted airspeed during each of the blocks.

When the aircraft is changing altitude, error in airspeed produced by a blocked pitot static system can quickly reach above 100 knots. An illustration of how quickly the airspeed error compounds with altitude is demonstrated in Fig. 10, which shows the error produced by the pitot static airspeed and the predicted airspeed for a pitot tube block lasting 510 seconds. This error jumps well above the error produced by our prediction method and results in a very unsafe situation for both piloted and unmanned aircraft.

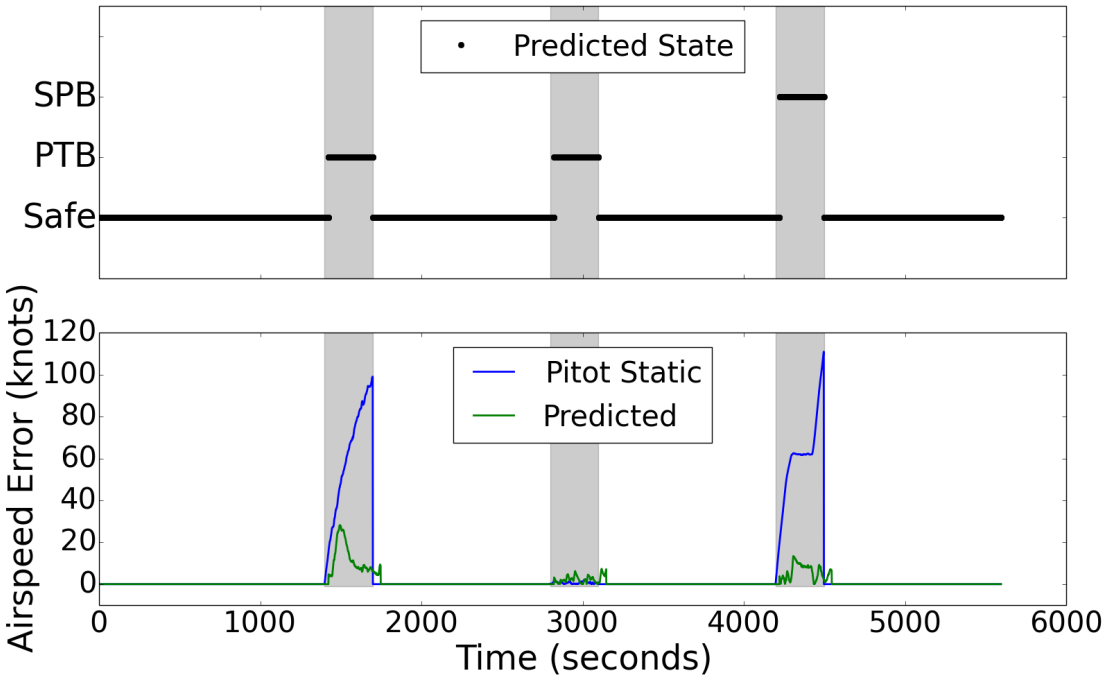


Figure 9. Upper: Detection of two pitot tube (PTB) and one static port block (SPB). Lower: Airspeed error. Shaded columns denote actual duration of block.

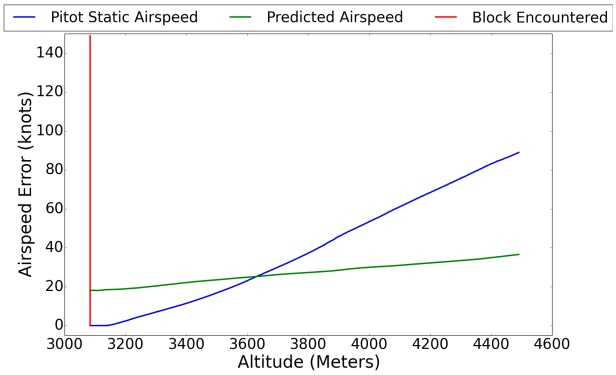


Figure 10. Error produced by the pitot static and predicted airspeeds as a function of altitude for a pitot tube block.

V. Conclusions and Future Work

This work proposed a machine learning approach to aircraft sensor error detection and flight data correction. The method was demonstrated on real flight data containing pitot static system failures. By calculating autocorrelation online and comparing it to our offline library, the aircraft is 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. Feature subset selection was performed on the high dimensional sensor output to be used for airspeed prediction. From previous flight data, we collect and store the output of those selected data streams in an offline library. Then in flight, the aircraft reads in those same streams and uses them for airspeed predictions. Using k-nearest-neighbors regression, the aircraft is able to predict airspeed within 41 knots of the true airspeed during the climb and cruise portions of flight. When a block is encountered during flight, the fault in the airspeed readings will progressively worsen. Having an airspeed prediction that can be incorrect by up to 41 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. One possibility for an increase in accuracy would be to use more flight data. This may increase the presence of certain trends and help reduce the effects of abnormalities in the small number of flights currently used.

This work can be generalized to various other sensors found on an aircraft. First, we must be able to define the behavior of a failing sensor and determine what flight data it has an effect on. With this information, an aircraft can learn to classify these failures and detect them online by referencing an offline library of sensor failure signatures. Once this is completed, a set of independently functioning sensors can be selected that have some relationship to the unreliable flight data we would like to correct. Using the output of these sensors as the features of our dataset, feature selection can be performed and the faulty flight data can be corrected using a prediction model.

Future work should be done in analyzing the responsiveness and accuracy required to maintain aircraft safety. A more in-depth look at the sensor data that we have available in flight may provide better predictions of airspeed. The sensor data selected is possibly the most important aspect of a functioning airspeed correction system. These predictions may also benefit from a different regression model. Due to the dimensionality of the data set, visualization of certain hidden trends in the data may not be possible. With a global regression algorithm, we may be able to pick up on these trends and produce better airspeed predictions than those from our local algorithm. If the prediction technique can be significantly improved, it may be useful to have a predicted airspeed running throughout the entire flight.

We have shown that there is potential in a machine learning approach to aircraft sensor error detection

and correction, which employs an offline/online paradigm. With future research into this methodology, pilots and UAV's can be confidently made aware of unreliable sensor output and follow a more accurate prediction of flight data. 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 too often, this results in aircraft being lost or damaged. This advance in technology will help avoid poor decisions and safely keep aircraft in flight.

Acknowledgments

This work was supported by FA9550-16-1-0108, under the Dynamic Data-Driven Application Systems Program, Program Manager Erik Blasch.

References

[1]Schechter, E., "Detecting Pitot Tube Obstructions," *Aerospace America*, edited by B. Iannotta, Vol. 52, American Institute of Aeronautics and Astronautics.

[2]Carbaugh, D., "Flight Instrument Information-Situations and Guidance," *Aero Magazine*, , No. 16, 2003, pp. 485–490.

[3]Klockowski, R., Imai, S., Rice, C. L., and Varela, C. A., "Autonomous Data Error Detection and Recovery in Streaming Applications," *Procedia Computer Science*, Vol. 18, 2013, pp. 2036–2045.

[4]Napolitano, M. R., Windon, D. A., Casanova, J. L., Innocenti, M., and Silvestri, G., "Kalman filters and neural-network schemes for sensor validation in flight control systems," *IEEE Transactions on Control Systems Technology*, Vol. 6, No. 5, Sep 1998, pp. 596–611.

[5]Napolitano, M. R., Chen, C. I., and Naylor, S., "Aircraft failure detection and identification using neural networks," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 6, nov 1993, pp. 999–1009.

[6]Massoumnia, M.-A., Verghese, G., and Willsky, A., "Failure detection and identification," *IEEE Transactions on Automatic Control*, Vol. 34, No. 3, mar 1989, pp. 316–321.

[7]NASA, "DASHlink Sample Flight Data," .

[8]CHRobotics, "Understanding Euler Angles," .

[9]Administration, F. A., *Pilot's Handbook of Aeronautical Knowledge*, chap. 5,8.

[10]Linke-Diesinger, A., *Systems of commercial turbofan engines: an introduction to systems functions*, Springer Science & Business Media, 2008.

[11]Allaire, D., Chambers, J., Cowlagi, R., Kordonowy, D., Lecerf, M., Mainini, L., Ulker, F., and Willcox, K., "An Offline/Online DDDAS Capability for Self-Aware Aerospace Vehicles," *Procedia Computer Science*, Vol. 18, No. 0, 2013, pp. 1959 – 1968, 2013 International Conference on Computational Science.

[12]Burrows, B. J. and Allaire, D., "A Comparison of Naive Bayes Classifiers with Applications to Self-Aware Aerospace Vehicles," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017, p. 3819.

[13]Burrows, B., Isaac, B., and Allaire, D. L., "A Dynamic Data-Driven Approach to Multiple Task Capability Estimation for Self-Aware Aerospace Vehicles," *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2016, p. 4125.

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

[15]Woodman, O. J., "An introduction to inertial navigation," Tech. Rep. UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, Aug. 2007.

[16]Aggarwal, C. C., Hinneburg, A., and Keim, D. A., *On the Surprising Behavior of Distance Metrics in High Dimensional Space*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 420–434.

[17]Kruskal, J. B., "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, Vol. 29, No. 1, Mar 1964, pp. 1–27.

[18]Sammon, J. W., "A Nonlinear Mapping for Data Structure Analysis," *IEEE Trans. Comput.*, Vol. 18, No. 5, May 1969, pp. 401–409.

[19]Wold, S., Esbensen, K., and Geladi, P., "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, Vol. 2, No. 1, 1987, pp. 37 – 52, Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[20]Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, 1983, pp. 671–680.

[21]Aha, D. W., *Artificial Intelligence Review*, Vol. 11, No. 1/5, 1997, pp. 7–10.

[22]Kotsiantis, S. B., "Supervised Machine Learning: A Review of Classification Techniques," *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, IOS Press, Amsterdam, The Netherlands, The Netherlands, 2007, pp. 3–24.