Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 59 (2017) 172 – 177

www.elsevier.com/locate/procedia

The 5th International Conference on Through-life Engineering Services (TESConf 2016)

# Using big-data and surface fitting to improve aircraft safety through the study of relationships and anomalies

Duncan Wooder[a], Alan Purvis[a,*], Richard McWilliam[b]

[a]*School of Engineering and Computer Science, Durham University, Lower Mountjoy, South Road, Durham DH1 3LE, UK*
[b]*IBEX Innovations Ltd, Discovery 2 NETPark William Armstrong Way, Sedgefield, County Durham TS21 3FH*

* Corresponding author. Tel.: +44 (0) 191 33 42437. *E-mail address:* alan.purvis@durham.ac.uk

**Abstract**

The aim of this paper is to assess the utility of a Big-Data approach to fault detection for 'systems of systems', utilising the derivation of empirical relationships identified through surface fitting. So-called Big-Data Integrated Vehicle Health Management systems do currently exist, but tend to analyse the health of vehicle systems based on the behaviour of individual sensors and readings. This paper proposes that it is possible to consider vehicle systems with a 'macro' approach and identify relationships between key variables which may not be initially apparent. Used in this paper is the open source flight simulation software FlightGear which has previously been assessed for the development of fault detection systems with positive results. The relationships found can be combined into a model of expected results against which real-time data is tested. Surface fitting and the assessment of 'goodness of fit' is used to identify these relationships. It is proposed that this technique need not be limited to fault detection in vehicle systems but is also applicable to other vital systems which require redundancy and constant health analysis. This paper concludes that this method is a viable approach and that relationships can be successfully identified for fault detection purposes.

*Keywords:* FlightGear; fault detection; big-data; integrated vehicle health management; surface fitting

## 1. Introduction

Commercial air travel is an increaisly accessible form of transport and when considered on a fatalities per kilometre basis, proves to be the safest mainstream form of transport, at only 0.05 deaths per bn km. However, when compared in terms of fatalities per journey, travelling by air can be seen to be nearly 30 times more dangerous than bus travel at 117 deaths per bn journeys[1]. Of course, the average person does not take as many journeys by air as by bus, and, for this reason, the risk is often seen as reasonable. It could be said that aviation will always be unsafe, to some degree, due to the nature of the mode of travel. However, it is the ambition of many organisations, ranging from aircraft manufacturers to airline operators to reach an optimal level of safety in aviation, whilst maintaining the profitability of the form of travel. It is the aim of this project to develop theory in fault detection from Big-Data which could help secure improvements in both of these areas. Big-Data is the concept of the processing and analysis of extremely large, multi-variate data-sets, often polled at high frequencies.

It is consistently true that a large proportion of fatalities in aviation each year can be attributed, at least partially, to mechanical error[2]. Fault detection and prediction can be used to mitigate the risk to human lives caused by the potential for unexpected or undiscovered mechanical faults. It has been the opinion of Rolls Royce for some time that a large pool of data (Big-Data) can be used to both improve efficiency and safety[3] by finding or predicting the occurrence of faults. However, the detection of faults within complex engineering systems is a well-known challenge [4]. The engines which Rolls Royce produces are each equipped with hundreds of sensors which report in real-time to engineers based in the UK. During each flight, terrabytes of data are generated by the aircraft's engines. These data are analysed on-board, during the flight, and a distillation of these data is transmitted to the ground for maintenance action to either be scheduled in a few weeks time or for a ground crew to be dispatched immediately to the flight's destination. Upon landing, the entire data-set is available for download and analysis. Anomalies in pressure, temperatures and vibration measurements, amongst others, are investigated as potential indicators that an engine requires service. The expected values are generated from both simulated and experimental analysis. This paper suggests that a similar strategy of Big-Data analysis could be used on the aircraft's systems as a whole with similar aims.

Although proof of concept is provided by Rolls Royce, the complexity of modelling and analysing an entire aircraft's systems as compared to a single engine is far greater. It is noted that an engine is subject to relatively constant conditions and has limited variance in input but an aircraft and its systems experience a vastly varying set of conditions due to environmental or consequential factors ranging from the weather and cruising altitude to the duration and route of a flight. An aircraft is, of course, also a much larger system than a single engine and, is, in reality, a 'system of systems'.

This paper assesses the viability of a Big-Data approach to fault detection and prediction in aircraft systems and uses the open-source flight simulation software FlightGear[5] to provide the modelling of a Boeing 737. Ben Morris (2015) demonstrates the utility of *FlightGear as a Tool for Real-Time Fault Detection and Self-Repair*[6] and proves that its extensive input/output (I/O) features allow for it to be used for this purpose. Boeing, a leading commercial aircraft manufacturer, uses their Aircraft Health Management system (AHM system), a form of Integrated Vehicle Health Management (IVHM) on-board aircraft to attempt to detect or predict faults in order to increase safety and reduce the financial and logistical impacts of failure. This system monitors individual components on a micro scale to detect or predict failure on an element-by-element basis. The data generated from individual components is compiled into a system report, but the system's health is not analysed on a truly macro scale. It is suggested by this paper that a possible approach to this macro analysis is to use Big-Datasets from 'healthy' flights to form a model of how an aircraft is expected to behave. Individual, and potentially unexpected, relationships may then be drawn from these data-sets and flight data compared in real-time in an attempt to assess the health of the aircraft as a whole. The aim of this paper is to assess whether Big-Data from an open-source simulator may be used as a platform for the development of such a system.

## 2. Theory

### 2.1. FlightGear

Key to the operation of FlightGear is the Property Tree. The Property Tree is described as the 'central nervous system' in documentation[5]. It features a hierarchical tree-like structure of low-level variables. These variables range in function from user interface to Flight Dynamic Model (FDM), the modelled mechanics behind the behaviour of the aircraft. There are numerous different methods of viewing and modifying these variables. The folders entitled *instrumentation* and *engines* include many of the variables which would be expected to be found in an avionics computer of a real aircraft. There are approximately 1000 variables in these folders and it is these upon which this paper focuses. The input/output capabilities of FlightGear are comprehensive and designed for implementing configurable testbeds for a variety of applications [7]. I/O is configurable from the program launcher, with the ability for the use of standard or custom protocols. The Generic Protocol[8] allows for the design of custom packets with chosen fields. It is possible to select between reading and writing from/to file, serial, User Datagram Protocol (UDP) or Transmission Control Protocol (TCP). Since, by default, the FDM updates at 120Hz, it is possible to import/output variables at this rate. The configuration of protocols is facilitated by an easy to read XML file, examples of which are provided. FlightGear includes the option for the creation of multiple I/O protocols to allow for the concurrent transmission/receipt of different data sets.

### 2.2. Function Fitting

MATLAB includes objects, *cfit* and *sfit*, for the creation and analysis of empirical polynomial curve and surface functions fitted to a data set. It is possible for the user to select between curve and line fits as well as choosing the *fitType* from a variety of options including linear, quadratic or higher polynomial curves/surfaces. MATLAB offers polynomial fitting up to 9th order terms for curves and 5th order in each direction for surfaces in 3-dimensional space. With surface fitting, it is possible to select different orders of approximation for the $x$ and $y$ dimensions.[9] MATLAB's *fit* function uses the linear least-squares method to minimise the sum of the squares of the error between each data point and the proposed curve or surface. Equation 1 below shows how the number of coefficients, $k$, scales with $m$ (number of variables) and $n$ (order)[10]. It must be noted that MATLAB's *fit* function does not operate beyond 3-dimensions. Instead, *lsqnonlin* may be used.

$$\text{Number of terms, } k = \binom{n+m-1}{m-1} \qquad (1)$$

Equation 1 means that the expression for the number of coefficients needed to be found in a general case of n-dimensional surface fitting includes a factorial term with both the number of dimensions and the order of the desired polynomial. For this reason, the computation time for the finding of a least-squares fitted surface, which depends linearly on this number of coefficients to be found and linearly on the number of data points in the data set being fitted, will scale with factorial order.

The fit object includes the fit result (ie a list of function coefficients) as well as 'goodness of fit' statistics including the sum of squares due to error and the root mean squared error or deviation. A normalized root mean square error can be found by dividing the RMS error value by the difference between the maximum and minimum values of the dataset being fitted to (This method is referred to in this paper as Method A). However, in this case, it is also possible to normalise the RMS deviation value by the range in the $z$-direction of the fitted function over the $x, y$ plane in consideration by replacing the maximum and minimum values with the maximum and minimum heights of the surface. Approximated values for these can be found using the decimation of the surface in two dimensions. (This method is referred to in this paper as Method B).

## 3. Methods

### 3.1. Extracting Data from FlightGear

Before extracting a data-set from FlightGear, it was necessary to plan a 'healthy' test flight. For this study, a flight between London Heathrow and Toulouse Blagnac in a Boeing 737-200 was selected. The trip distance was chosen to minimise the size of the resultant data-set but maximise the cruise time of the flight and the aircraft was chosen as it includes the most developed auto-pilot in FlightGear. The -200 variant of the 737 was selected as it is optimal for the trip distance. A flight was made using the Boeing auto-pilot route planning facility and an Instrument Landing System approach and touchdown as is common with aircraft in commercial use. Fuel levels were accurately loaded and simulator realism was maximised, including randomising meteorological events. Random failures were disabled.

In order to extract the data-set from FlightGear, it was necessary to code a script in the language NASAL (an object oriented language which directly supports reading/writing property tree properties) which FlightGear utilises heavily. The NASAL console can be used to instruct FlightGear to output a .xml file with the hierarchical tree structure of the variables within the *instrumentation* branch of the property tree in FlightGear [11]. This structure is incompatible with the custom protocol .xml configuration files, which require a list of full paths (e.g */instrumentation/airspeed-indicator/indicated-speed-kt*) of each individual variable and, therefore, a custom script to compile a list of full paths from the tree structure was developed in MATLAB. The code was designed to detect the 'level' of the current branch and compose a full path using the '/' delimiter. The resulting list of full paths was retained for use in steps of analysis. At this stage, analysis proved that variables which included the text *'=n'* could be removed to minimise the data-set without compromising the accuracy of this relationship detection method. It was noted that the */instrumentation* branch does not include low level variables relating to engine operation (e.g. *N1* and *N2*) or pitot-static systems (e.g. *dynamic pressure* and *static pressure*). For this reason, individually selected variables from the branches */engines* and */systems* were added to the full paths list and custom protocol configuration for output.

The above mentioned test flight was flown with the start up argument *–generic=*params** (including parameters) to enable saving to file of the included custom protocol file at 1Hz. This generated a comma delimited text file which was imported into MATLAB as an array with columns corresponding to variables and rows corresponding to the sampled points over time. This array was temporally decimated by approximately 10 times to reduce computation time for correlation and fit functions. The variable property path names were also subjected to a keyword exclusion filter for terms such as 'freq' and 'encoder' to exclude variables which would not show any meaningful relationships. In order to test relationships in real-time, a UDP connection was coded in MATLAB to interface with current figures.

### 3.2. Finding Empirical Relationships

Ben Morris (2015)[6] shows that in an avionics system, there are ratios of variables which remain constant in healthy systems. This current paper suggests that these same relationships may be considered in a different manner. For a bi-variate dataset, if the ratio between variables remains constant temporally then when the variables are plotted as $x$ and $y$ respectively, a linear relationship will be observed, as can be seen with the relationship between barometric altitude and GPS altitude. The gradient ($dy/dx$) of this line is equal to a constant similar to that which Morris found and tested. The expected value of a variable plotted on the $y$ axis may be found using a linear equation. This mathematical relationship may be used in a fault detection routine to validate real-time data. However, this does not allow for an order of relationship higher than linear between the two variables and does not test variables which may depend on more than one other variable. In order to address the first point, a non-linear bi-variate Big-Data approach to relationship detection and evaluation is suggested.

It should be noted that the selected variable ratios in Morris' work are chosen using existing knowledge of flight systems and are specific to aviation. It is the aim of this current paper to present a Big-Data fault detection technique which is compatible with any system, given a 'healthy' data-set. For this reason, algorithms are used for each step of filtering and evaluating possible relationships.

#### 3.2.1. Bi-Variate Approach

Section 3.1 describes how a multi-variate, high frequency data-set was generated for a 'healthy' test flight. Using this data array, a list of all possible permutations of two variables was generated using two nested *for* loops. Permutations were selected rather than combinations because whether a variable is assigned to $x$ or to $y$ is significant in a bi-variate curve fitting. This looping structure was then used to compute fitted curves and their goodness of fit statistics (including RMS error and sum of squares error) for the bi-variate data. A *cfit* MATLAB object was accessed using structure notation to access coefficients and RMS error values. The coefficients are ordered such that $f(x) = p_1 \times x^2 + p_2 \times x + p_3$.

RMS data in its original form is dimensional (ie the RMS average error of identically well-fitting graphs with different $y$ dimensions will be different). Therefore, it is important to normalise this data for each bi-variate permutation. Section 2.2 demonstrates two methods of how to do so. In this 2D case, Method A is sufficient. Values of Normalised RMS Error (NRMSE) vary in magnitude between around $\times 10^{-10}$ to 0.5. This 2D array was sorted in MATLAB using the default *sort* function. Upon analysis of the sorted NRMSE array, it was clear that many of the low normalised RMS error values are due to direct coded correlations between variables in the simulator. For the purpose of this study, these are not of interest and were excluded or 'filtered'. Analysis allowed focus to be placed upon those relationships with significant quadratic terms and, following further filtering, a series of valid relationships of interest is found. The relationship between GPS Altitude and Barometric Altitude is an example of such a relationship.

#### 3.2.2. Tri-Variate Approach

Section 3.2.1 briefly explains the method of drawing bi-variate relationships using a Big-Data approach to fitting functions. Although it is possible to link this approach to the methods of Morris (2015)[6], the approach must be extended in dimensions in order for an automated process of relationship detection to be possible. Morris uses techniques of combining two or more variables in order to create normalised variables which may be monitored. As an example, a plot is used where the x-axis represents a ratio of variables $a$ and $b$ and the y-axis a ratio of variables $c$ and $d$. This may be considered instead as a fitted 4-dimensional function (ie one variable expressed as a function of three other variables). This current paper proposes that it is possible to extend this process to $m$-dimensions (or, into an $m$-variable case). The approximation order may also be extended to be of $n$th order.

In this current paper, the order of polynomial is limited to quadratic and a tri-variate approach is set out. This leads to the necessity to solve for six polynomial coefficients in each tri-variate combination of variables. MATLAB was also used in this case, with the use of the parallel processing command *parfor* to enable multi-processor support. It is important to note that in tri-variate analysis, for three variables, the order of variables assigned to both the $x$ and $y$ dimensions is irrelevant, but the variable chosen as the $z$ is significant. For this reason, a variant of combination theory, seen in Equation 2, may be used to calculate the number of combinations of variables which must be analysed in this way.

$$N = \binom{n}{3} \times 3 \text{ where n = number of variables} \tag{2}$$

*Surface Fitting.* Prior to the Big-Data function fitting process, it was important to exclude all variables of an either boolean (only two unique values) or highly discrete nature (few unique values) as these do not lead well to surface fitting. To do this, for each data-set, the number of unique values was calculated and analysed. Any variables which feature less than 20 unique values were excluded from the array. The resultant array was of 134 data-sets and with 680 sampled data points for each.

Due to the size in memory of *fit* and *gof* structures, it was important for the function fitting MATLAB code to be saved to disk regularly during the surface fitting process to prevent the saturation of RAM. It should be noted that this routine may be configured to be run on multiple nodes concurrently by changing starting and ending values. In this case, the availability of Durham University's *Hamilton* High Performance Computing cluster was exploited to expedite the process.

The resultant RMS error for each surface fit may be normalized individually using Method B in Section 2.2 and compiled with the six polynomial coefficients into a similar array structure to Section 3.2.1. The filtering and evaluation of fits in tri-variate, 3-dimensional cases is more complicated than in the above bi-variate example and requires a more detailed approach. As a comparison of numbers, with a data-set of 134 variables, as in this case, the bi-variate approach considers 17,956 combinations whereas the tri-variate approach considers 1,176,252.

In order to understand the reasoning for steps taken in filtering possible relationships, the desired characteristics of an ideal result must be appreciated. An ideal quadratic tri-variate relationship has very low error and, therefore, a high implied predictability and model reliability. It should be noted that the known data-points should be distributed relatively evenly across the surface and the maximum individual point error be minimal. In reality, it is unlikely that a bi-variate plane over which data-points are evenly distributed will be found. It was determined through the analysis of randomly chosen data-set combinations that a normalised RMS error of below approximately 0.05 is an acceptable safe threshold. This value ensures that no unnecessary data analysis is taking place without risking missing certain combinations which are of acceptable error. For this reason, after the sorting of combinations by NRMSE data, the primary combination filtering operation carried out is to exclude combinations with NRMSE values above 0.05. This immediately reduced the number of possible surfaces by 82.76%.

*Removing Linear Correlations.* Upon analysis of the data-set, it is clear that a large number of the combinations of variables result in straight lines in the $x, y$ plane due to linear relationships. These combinations are unwanted, as they suggest that the $x$ and $y$ variables are directly related and they do not produce surfaces with well distributed data points. Therefore, the second filtering procedure which was carried out is designed to remove data which follows a linear relationship in the $x, y$ plane. For this, a Pearson correlation coefficient for all possible bi-variate (2-dimensional) combinations is calculated. This is not a computationally intensive process as there are only $N = \binom{134}{2} = 8,911$ possible combinations of $x$ and $y$. Histograms allowed insight into the correct value of coefficient to be used as the threshold. In this case, all $x, y$ relationships which exhibited correlation coefficients greater than 0.9 were excluded. This operation reduced the number of surfaces under analysis by a further 7.78%.

*Removing Flat Surfaces.* Subsequently, graphical analysis of random samples from the array of combinations proved that a large number of surfaces under consideration were flat surfaces parallel to one axis. These surfaces are of little interest as they imply a relationship between one of $x$ or $y$ and $z$ but not both. Although this relationship can be expressed in 3-dimensions, it is not a true tri-variate relationship and for this reason should be discarded. In order to identify such flat surfaces parallel to an axis, but not unintentionally remove surfaces with quadratic-only relationships, the following method was used. If the z values at the extremities of a surface are too similar relative to the total range of the function in $z$, a flat surface is implied. Although this holds in strictly linear cases, in quadratic cases a more complex distinction must be found. In the case of a quadratic surface, the edge values may be equal but the centre value much different. For this reason, it was decided to sample the height in $z$ of the surface in three locations in each dimension. A normalised comparison between these values can detect a flat surface.

This filtering process reduced the number of combinations by a further 89.32%.

*Removing poorly distributed surfaces.* Many of the remaining surfaces do not have a sufficient distribution of data over the $x, y$ plane to enable a useful tri-variate relationship to be drawn. It is important to exclude these combinations of variables from analysis without discarding potentially interesting surfaces. In order to develop a normalised spread coefficient, the concept of discretisation was used. Each surface was discretised in each dimension $x, y$ into 14x14 'buckets' or cells. For each bucket, it was determined whether a data point existed within its coordinates. The ratio of cells with data to cells without data becomes the determinant for exclusion. Using the analysis of random samples, 15x15 points (14x14 buckets, as each bucket has two points in each dimension) was chosen as a suitable level of discretisation. Histograms were used to select a spread determinant of 0.2 as a threshold value. This filtering process reduced the number of combinations by a further 75.03%.

*Removing surfaces with high point errors.* In order for fault detection to be viable using empirically derived relationships, it is important that the false positive detection rate is low. For this to be the case, it is key that there is low individual point deviation of data points from the surface which represents the expected response of the system. Some relationships may have relatively low (acceptable) RMS error but maximum individual point deviation from the empirically found surface can be large. In order to exclude combinations by maximum point error, the values must be normalised. A conservative value of 0.3 was chosen as the threshold for normalised maximum point error by histogram analysis. This filtering reduced the number of combinations under consideration by a further 61.72%.

## 4. Results and Discussion

After filtering 99.84% of the original 1,176,252 tri-variate combinations, using the processes described in Section 3, 1,910 remained. As these relationships are empirical, the resultant models must be tested with more extensive data-sets to truly validate them (for example, by further flights). However, this is beyond of the immediate scope of this paper. For the remaining relationships, manual intervention was required to select the most interesting (relevant to fault detection) and viable combinations with the aid of available data and graphical visualisation. It is important during manual analysis to check the relationships which the above algorithms and processes select to ensure that they are physically accurate (i.e. not coincidental) and of interest. For some variables, it is also useful for the variable which is represented by the $z$ axis in a surface representation to be selected by manual analysis. Although, at this point, nearly 2,000 relationships remained under analysis, many of these were related to the same few variables, often air data. Also, due to the conservative choice of 0.05 NRMS error as a threshold value, many of these relationships are far too 'noisy' to be of use in reality. The methodology of choosing acceptable relationships involved sorting through the remaining array in the order of increasing NRMS error. The surface of Indicated Altitude vs GPS Ground Speed vs Total Pitot Pressure is an example of a relationship which appears to be near perfect, in terms of error, but in reality does not correspond to physical relationships. The indicated ground speed occasionally exceeds 1,000kt leading to a false surface being generated. It is clear upon manual analysis that this data is due to errors in the instantaneous GPS speed measurement, but when fitted to a surface, the result can be deceptive.

Figures 1 and 2 demonstrate two examples of relationships selected by manual analysis. The low RMS and individual point error of both surfaces should be noted. Figure 1 shows the relationship between Indicated Speed, Total Pitot Pressure measured and the True Speed of the aircraft. It is important to note that in an aircraft, both the indicated speed and the true speed are important. For example, the true speed corrected for wind speed is approximately the ground speed and is key for navigation. The indicated airspeed is important as the flight dynamics of the aircraft respond directly to it (this is key in stall speed calculations). The relationship between the two is related to the altitude of the aircraft, as confirmed by Figure 1. It is, therefore, likely that further testing would prove this tri-variate empirical relationship to be useful for the purpose of fault detection in the calculation of True Speed.

A second example of an empirical tri-variate relationship found by this paper is shown in Figure 2. Here, the relationship between Indicated Altitude, True Speed and Indicated Mach is

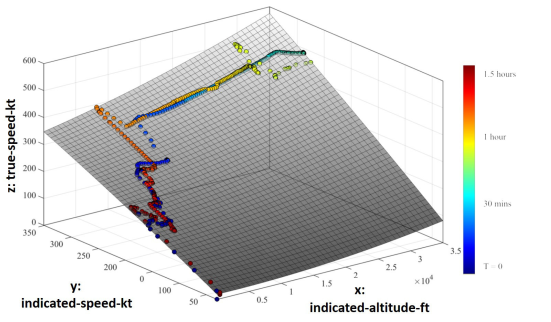

Fig. 1. Indicated Speed vs Indicated Altitude vs True Speed. Grey surface indicates fitted relationship. Coloured markers indicate original data-set. Marker colour scale indicates time into flight of data points. $z = -15.2 - 7.19 \times 10^{-4}x + 1.14y + 5.85 \times 10^{-8}x^2 + 1.48 \times 10^{-5}xy - 2.99 \times 10^{-4}y^2$

demonstrated. The Indicated Mach number is a ratio of the True Speed of the aircraft and the speed of sound at that specific altitude. The relationship between speed of sound and altitude is linear up to approximately the operating ceiling of a commercial airliner. Therefore, the gradient of the line connecting data points of True Speed and Indicated Mach should vary linearly with an increasing altitude. This is confirmed by the empirical relationship found and can also be used for fault detection in the calculation of Indicated Mach.

Multiple relationships similar to both of these example surfaces exist due to the characteristics of the air in which flight is undertaken.

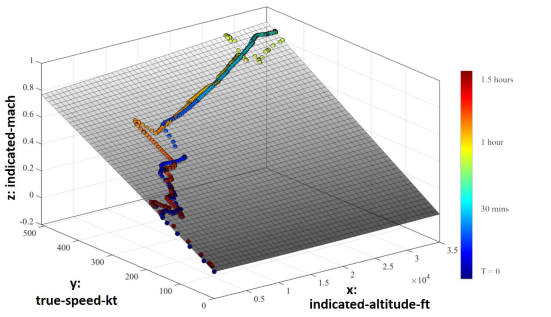

Fig. 2. Indicated Altitude vs True Speed vs Indicated Mach. Grey surface indicates fitted relationship. Coloured markers indicate original data-set. Marker colour scale indicates time into flight of data points. $z = -5.86 \times 10^{-4} - 3.61 \times 10^{-7}x + 1.53 \times 10^{-3}y + 1.57 \times 10^{-11}x^2 + 6.37 \times 10^{-9}xy - 4.49 \times 10^{-8}y^2$

Figure 3 demonstrates how the surfaces derived using the above methods can be used for fault detection. A drifting-type fault was injected into the calculation of Mach number at 15 minutes after take-off. The error between the actual value of Mach number and that injected was increased following a quadratic relationship with added noise. The deviation of each point from the expected surface was calculated, normalised and plotted against time, with healthy deviations included for reference. An error threshold of 1.5 times the maximum point error of the healthy data set was selected. The algorithm was set to flag a fault when two consecutive points breached this threshold in order to remove natural high frequency variation and discourage false positive results.

As seen in Figure 3, the fault detection algorithm detected a very slowly drifting fault within 30 minutes at an error value of approximately 0.5% from the healthy value. However, it should be noted that faults injected with faster drifts were detected in less time and instantaneously occurring faults detected within one second. A benefit of this detection method is that faults which occur either gradually or instantaneously can be detected. These results conclude that this method is not only viable but has produced meaningful, accurate results. It is suggested that reproducing the process on a more complex data set of a system involving higher order relationships could allow the process to be further developed.

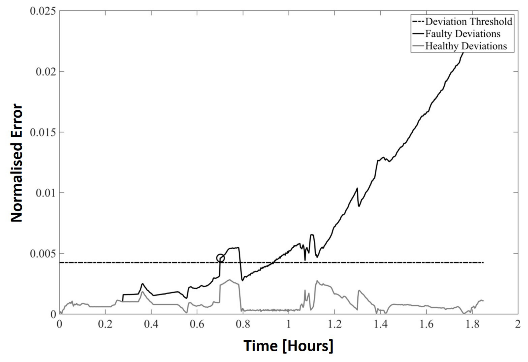

Fig. 3. A gradual drifting-type fault in the calculation of Mach number injected at 15 mins from take-off. Lines show normalised values of deviation from expected surface and error threshold. Healthy deviations included for reference.

## 5. Conclusions

This paper concludes that there is value in a Big-Data approach to fault analysis through the detection of empirical relationships by surface fitting. The filtering process used in this study was successful in excluding 99.84% of potential variable combinations automatically. It is observed that this process is scalable to higher orders, an increased number of dimensions and a larger number of variables. It is important to note, however, that the computation time of the learning process is a key factor in the viability of this process. Computation time scales with factorial terms of all three of these values and, therefore, for large models with complex relationships, the computation time can be extreme. The use of multiple computing nodes is an obvious solution to this issue although it assumes that a High Performance Computing cluster is available. In reality, simplification of the order and/or dependence of the relationships may produce acceptable results and should be attempted before increased complexity is introduced. Although not tested specifically in this paper, it is suggested that this approach could be easily adapted to systems outside of the field of aviation. This process would be well suited to systems of vital importance which require constant monitoring and fault detection. For use in the field of IVHM, it is proposed that relationships from the processes suggested in this paper may be included in fault detection routines. Computation power required on-board for analysis is minimal as the expected value for a function may be calculated with ease. The results from the fault detection implementation of the theory in this paper show that the relationships detected may be used to detect even slow drifting faults in a relatively short time. In the calculation of Mach number, errors of approximately 0.5% were detected by the algorithm. Furthermore, it is suggested that the analysis of the rate and direction of deviation away from the surface of the expected model could be used to determine the exact nature of the fault to develop a multi-class anomaly detection system. This information could be integrated into existing systems to alert the aircrew or the ground-crew, depending on the classification of the fault. Limitations of this paper include the use of simulated rather than real-world data for the generation of empirical relationships. A repeat of this study using similar methods but a real-world data-set would be of value. It should be formally noted that throughout this paper it is assumed that the highest order of relationships within the data-set analysed is quadratic and a limitation of 3-dimensional (tri-variate) analysis imposed. No relationships are found where there is insufficient data in the data-set considered. For example, if a relationship holds only at an altitude of 40,000ft, which the flight in the chosen data-set does not reach, then this will not present itself as a relationship found. Furthermore, this paper considers only those relationships which hold for the entirety of a flight (i.e. take-off, cruise and approach) and does not consider the temporal response of variables. In further work it could be possible to include a variable of time in the array of variables to seek these temporal relationships, or to separate the model into individual time-based responses.

Despite the limitations outlined above, this paper proposes that the detection of empirical relationships using the concept of Big-Data surface fitting is a viable approach to fault detection in vital systems.

## Acknowledgements

The authors would like to thank Dr Will Coombs and Dr Ian Jermyn of Durham University for their helpful suggestions.

## References

[1] "The risks of travel," http://www.numberwatch.co.uk/risks_of_travel.htm, (Accessed Oct 2015).
[2] "Cause of fatal accidents per decade," http://www.planecrashinfo.com/cause.htm, (Accessed Dec 2015).
[3] "How big data drives success at Rolls Royce," http://www.forbes.com/sites/bernardmarr/2016/06/01/how-big-data-drives-success-at-rolls-royce/, (Accessed Oct 2015).
[4] Z. Guo, G. Jiang, H. Chen, and K. Yoshihira, "Tracking probabilistic correlation of monitoring data for fault detection in complex systems," in *International Conference on Dependable Systems and Networks (DSN'06)*, June 2006, pp. 259–268.
[5] "The flightgear flight simulator project is an open-source, multi-platform, cooperative flight simulator development project," http://www.flightgear.org/, (Accessed Oct 2015).
[6] "Flightgear as a tool for real-time fault detection and self-repair," http://www.sciencedirect.com/science/misc/pii/S2212827115008926, (Accessed Oct 2015).
[7] A. Y. Javaid, W. Sun, and M. Alam, "Uavsim: A simulation testbed for unmanned aerial vehicle network cyber security analysis," in *2013 IEEE Globecom Workshops (GC Wkshps)*, Dec 2013, pp. 1432–1436.
[8] "Flightgear: Generic protocol," http://wiki.flightgear.org/Generic_protocol, (Accessed Oct 2015).
[9] "Mathworks: Fit," http://uk.mathworks.com/help/curvefit/fit.html, (Accessed Dec 2015).
[10] "Combinatorial analysis, properties," http://dlmf.nist.gov/26.4, (Accessed Feb 2016).
[11] J. Zhang, Q. Geng, and Q. Fei, "Uav flight control system modeling and simulation based on flightgear," in *Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on*, March 2012, pp. 2231–2234.