# Fuzzy Logic and Mahalanobis Distance Algorithms for Fault Detection in Fixed Wing UAVs

Ruth Gomez Quezada*
*University of Maryland, College Park, MD, 20742*

Lina M. Castano†
*Axient, Huntsville, AL 358062*

Huan Xu‡
*University of Maryland, College Park, MD, 20742*

**Control surface failures are one of the most common faults in flight on Uncrewed Aerial Vehicles (UAVs). Most fault detection methods focus on control and servo loops analysis. In this paper, two heuristic-based algorithms were developed for fault detection in stuck control surfaces on UAVs, both are computationally inexpensive plus no aerodynamic model is required. The first called Mahalanobis Distance Fault Detection Algorithm (MDFDA) uses said measurement to calculate the separation of state vector with respect to the average for nominal flight. The second called Fuzzy Logic Fault Detection Algorithm (FLFDA) implements this approach using state threshold values based on nominal flight limits. Fault isolation is possible with this algorithm. Overall, MDFDA performed better and faster, FLFDA performed better on faults larger than ±10 degrees, and both methods were the most successful in detecting aileron failures.**

## I. Nomenclature

| | | |
|---|---|---|
| $Coeff_{CS_{all}}$ | = | Sum of all possible coefficients in control surface |
| $CS_{ID}$ | = | Control surface ID |
| $Coeff_j$ | = | Coefficient of state j in control surface |
| $CS$ | = | Control surface |
| $CS_{lim}$ | = | Control surface limit |
| $F$ | = | System matrix |
| $G$ | = | Control matrix |
| $h$ | = | Relative altitude |
| $MD$ | = | Mahalanobis distance |
| $p$ | = | Roll angle rate |
| $\dot{p}$ | = | Roll angle acceleration |
| $q$ | = | Pitch angle rate |
| $\dot{q}$ | = | Pitch angle acceleration |
| $r$ | = | Yaw angle rate |
| $\dot{r}$ | = | Yaw angle acceleration |
| $S$ | = | Covariance matrix |
| $v_x$ | = | Velocity in the x axis |
| $v_y$ | = | Velocity in the y axis |
| $v_z$ | = | Velocity in the z axis |
| $\overrightarrow{x}$ | = | State vector |
| $\overrightarrow{\dot{x}}$ | = | State dynamics vector |
| $\overrightarrow{x_{FL}(t)}$ | = | State vector for fuzzy logic |

*Research Assistant, Department of Aerospace Engineering, Glenn L. Martin, College Park, MD 20742.
†UAS Program Lead, Arlington, VA, 22202
‡Associate Professor, Aerospace Engineering Department, 3180 Glenn L. Martin, College Park, MD 20742

| | | |
|---|---|---|
| $\overrightarrow{x_{MD}(t)}$ | = | State vector for Mahalanobis distance |
| $\delta$ | = | Input vector |
| $\overrightarrow{\mu(t)}$ | = | State vector averages |
| $\psi$ | = | Yaw angle |
| $\phi$ | = | Roll angle |
| $\theta$ | = | Pitch angle |

## II. Introduction

The increased usage of Uncrewed Aerial Vehicles (UAVs) for civilian and military operations call for a well suited hazard mitigation plan. According to Williams [1] who compared accidents in 5 different UAVs, around 33% to 67% of incidents are caused by an aircraft malfunction. These could have been prevented with a good mitigation plan.

A fast fault detection in UAVs is the first step to this plan, this could be definitive for the conservation of the aircraft as well as for people surrounding the flight area. Different faults in UAVs are found in motion sensors, motors, communication links with ground control, Global Positioning System (GPS), and additional sensors like LIDAR, cameras, or servos. This paper focuses on servo/actuator fault. There are different categories in servo failures: noise, overheating or electrical issues. However, these malfunctions of the servo lead to three different commanded vs real (physical) faults.

The first one is a weak signal servo which makes the amplitude of real deflection be less than the commanded. The second is an oscillating servo that is uncontrollable. The third one is a stuck servo that can not mechanically move. For the purpose of this paper, when referring to a control surface fault, said fault is a stuck servo. If the fault can be isolated into the corresponding control surface, a better mitigation plan can be developed.

Fault detection in UAVs could be expensive and difficult to implement as, generally, a control system must be designed to measure the difference in commanded vs actual signals. These methods are not fast enough because of their complexity. An example of these is the work of Zhang et al. [2] where a Kalman Filter is used to measure the residual sequences of each actuator signals and with this, detect and isolate faults. Another example is from Gheorghe et al.[3] where a servo loop model was developed to monitor the actuator performance, but again, a servo model has to be created. Some other authors Cork and Walker [4] for example, have created algorithms that used the knowledge of the nonlinear aerodynamic model, however, to obtain the coefficients of a nonlinear model there is a need for a system identification process which takes a lot of flight testing or wind tunnel tests. Heredia et al.,[5] developed a fault detection algorithm that combines GPS and vision sensors, but this could be expensive as well as add weight to the aircraft. Machine Learning algorithms are also being implemented in fault detection; however, it needs a vast flight database that can take months to gather. Algorithms based on heuristics were used in the past, like Douglas and Speyer [6] were the rule method was applied, although fault isolation was not reached at the time.

Research presented in this work, focused on creating Heuristics based algorithms that can detect and isolate a fault in stuck control surfaces i.e., aileron, rudder, and elevator. Section II defines the algorithms developed in this study. Section III contains the implementation for testing these algorithms. Section IV is the results from algorithms applied to flight test data. Section V talks about the conclusions derived from the experiments.

## III. Algorithms

Two algorithms were formulated to accomplish this. The first uses the Mahalanobis Distance inspired by Lin et al. [7] called the Mahalanobis Distance Fault Detection Algorithm (MDFDA). The second based on Fuzzy Logic called Fuzzy Logic Fault Detection Algorithm (FLFDA).

### A. MDFDA

Mahalanobis Distance is the value in standard deviation units (std units) that measure the distance from a point on a multi-dimensional space (state space in this case) to its covariance and averages. In MDFDA, MD is used to get a distance from the corresponding state vector to the distribution of the nominal flight, created with averages of each state, and a covariance matrix that describes the relationship between the states. MD was first used on Aircraft anomaly detection in 2001 by Brotherton [8]. Later it was used on UAVs by Lin et al. [7] for anomaly detection on the entire aircraft system. The main difference between this approach and MDFDA is the amount of states that are being considered. In this study we are only focusing on anomalies in control surfaces which means states like engine

temperature, mass, sensor readings, etc. will not be included when calculating the MD.

$$MD(\overrightarrow{x_{MD}(t)}) = \sqrt{(\overrightarrow{x_{MD}(t)} - \overrightarrow{\mu(t)})^T S^{-1} (\overrightarrow{x_{MD}(t)} - \overrightarrow{\mu(t)})} \tag{1}$$

The MD is calculated using Equation 1. Where $\overrightarrow{x_{MD}(t)}$ is the state vector at time $t$, $\overrightarrow{\mu(t)}$ is the vector corresponding to the state averages from start to time $t$, and $S$ is the covariance Matrix.

The states composing $\overrightarrow{x_{MD}(t)}$ are altitude ($h$), the Euler angles (i.e., roll ($\phi$), pitch ($\theta$),and yaw ($\psi$)), the angle rates (roll rate ($p$), pitch rate ($q$), and yaw rate ($r$)), angle accelerations (roll acceleration ($\dot{p}$), pitch acceleration ($\dot{q}$), and yaw acceleration ($\dot{r}$)), and components of velocity ($v_x$, $v_y$, and $v_x$). These states are used to calculate $\overrightarrow{x(t)}$, and the average of their values over time is $\overrightarrow{\mu(t)}$. $S$ is defined by gathering data from different time steps in a nominal flight, and calculating its covariance. This implies $S$ is a 13x13 matrix. The first step to define a covariance matrix is to find a 13x13 state matrix. The state vectors for this can be collected over one nominal flight in 13 different time steps. Another option is to select 13 state vectors from different flights in a nominal stage, which means all values had to be stable and should not be affected by a fault. For this study, the latest was used, permitting the state matrix to have data from different days with varying weathers. When the aircraft turns, yaw and roll changes drastically and the MD will increase. For this, adding state vectors captured during turning is highly recommended to add this maneuver into the state matrix.

Algorithm 1 is the pseudo code for MDFDA. Line 6 contains a check point that the algorithm will not start below altitude ($h$) of 20 meters. This is to eliminate possible false positives when the aircraft is in take off or landing stage. In line 10, the MD is compared to the threshold value $MD_{lim}$ that, after data analysis, was set to 1000.

**Algorithm 1** MDFDA($MD_{lim}, S^{-1}, t_{final}$)

$t_o \leftarrow 0$
**for** $t \leftarrow$from $t_o$ to $t_{final}$ **do**
  $\overrightarrow{x_{MD}} \leftarrow$ state vector at time t
  $\overrightarrow{\mu} \leftarrow$ vector with state averages from $t_o$ to $t$
5: $h \leftarrow x_{MD}(1)$ first state in $\overrightarrow{x_{MD}}$
  **if** $h(t) < 20$ **then**
    $Fault_t \leftarrow 0$ i.e. the aircraft is not flying
  **else**
    $MD_t \leftarrow \sqrt{(\overrightarrow{x_{MD}} - \overrightarrow{\mu})^T S^{-1} (\overrightarrow{x_{MD}} - \overrightarrow{\mu})}$
10: **if** $MD_t > MD_{lim}$ **then**
      $Fault_t \leftarrow 1$
    **end if**
  **end if**
**end for**

### B. FLFDA

The concept of Fuzzy Logic was created by Zadeh in the 1960s. See Ref.[9] for an article on the history of the acceptance of fuzzy logic in science. It is defined as the notion that there are more than binary truths. In a mathematical way, there is a scale where 0 is not true and 1 is the truth. In this work this can be called certainty, where 0 (or 0%) is certainly that there is not a fault on the aircraft, and 1 (or 100%) reflects that most certainty there is a fault on the aircraft. The states used in this application ($\overrightarrow{x_{FL}(t)}$ are altitude $h$, velocity in z-axis $v_z$, euler angles ($\phi$, $\theta$, and $\psi$), and angular rates ($p$, $q$, and $r$). In order to calculate the certainties, parameters must be delimited and weights of impact of each state in all control surfaces should be determined. The weights each state holds in the certainty that there is a fault in each actuator are the coefficients in Table 1, which are use to calculate certainties for all faults using equation 2.

Algorithm 2 is the pseudo code for FLFDA which describes its procedure in detail. At each time step: first, three variables are set to be determined ($Fault_{elevator}$, $Fault_{rudder}$, and $Fault_{aileron}$), the code starts testing the conditions in Table 1 and assigning the coefficient values to the designated fault variable. After all conditions for each control surface are compared, the certainty of each is calculated as stated in Eq. 2 where $j$ stands for the control surface ID. All certainties ($Aileron_{certainty}$, $Elevator_{certainty}$, and $Rudder_{certainty}$) are then compared and the largest one is set to be the predominant fault at that time step sending an output signal with the control surface ID of 1 for aileron fault, 2 for

**Table 1 FLFDA limit values and coefficients**

| Control Surface | Conditions | Coefficient |
|---|---|---|
| Elevator | $\|\theta\|$ > 20 deg | 3 |
| | $q$ > 50 deg/s | 3 |
| | $q$ < -40 deg/s | 3 |
| | $v_z$ > 10 m/s | 1 |
| Rudder | $\|\phi\|$ > 30 deg | 2 |
| | $\|r\|$ > 50 deg/s | 3 |
| | $\|q\|$ > 50 deg/s | 1 |
| | $v_z$ > 20 m/s | 1 |
| Ailerons | $\|\phi\|$ > 50 deg | 2 |
| | $v_z$ > 10 m/s | 1 |
| | $\|p\|$ > 50 deg/s | 3 |
| | $\theta$ < -15 deg | 1 |

elevator, and 3 for rudder. However, if the certainty is below 40%, the fault is set to zero as that is not enough certainty to confirm a fault. In addition to that, as with the MDFDA, if the altitude is below 20 meters the plane is not flying at an altitude where faults were injected, and the fault is set to zero. An example of how this works is if $\theta$ is greater than 20 degrees, and $v_z$ is higher than 10 m/s. The certainty for elevator fault is calculated adding the corresponding coefficients and dividing it by the sum of all coefficients collected from Table 1: (3 + 1)/9, the certainty is 44.4%. As this certainty is greater than 40%, the algorithm will signal an elevator fault (i.e. 2) with a certainty of 44.4%.

$$Certainty_j = \frac{\sum Coeff_{applicable}}{\sum Coeff_{j_{all}}} \tag{2}$$

**Algorithm 2** FLFDA($\overrightarrow{x_{FL}}, t$)

**for** Every $t$ **do**
  **for** Every $x_{FL}$ **do**
    **if** $x_{FL_i} > CS_{limit}$ **then**
      $Fault_{CS} \leftarrow Fault_{CS} + Coeff_j$ (Fault in control surface is the addition of the applicable coefficients)
    **end if**
  **end for**
  **for** Every $Fault_{CS}$ **do**
    $CS_{certainty} \leftarrow Fault_{CS} / \sum Coeff_{CS_{all}}$
  **end for**
  $Certainty \leftarrow$ Greater $CS_{certainty}$
  $Fault \leftarrow CS_{ID}$
  **if** $Certainty < 0.4$ **then**
    $Fault \leftarrow 0$
  **end if**
  **if** $h < 20$ **then**
    $Fault \leftarrow 0$
  **end if**
**end for**

## IV. Implementation

Heuristic based algorithms need a data set to be configured. That data was used to create the covariance matrix in MDFDA, and to select the limit values specific to this aircraft for FLFDA. This data was collected using a Carbon-Z

Cub model airplane with wing span of 2.1 m. Fig. 1, which was equipped with the following electronics:

- Baseline avionics
- Primary and secondary receivers
- 4 channel multiplexer
- Fault injection board
- Raspberry Pi for data collection
- Smart G2 LiPo Battery
- S3 LiPo Battery

**Fig. 1 Instrumented Carbon-Z Cub for flight testing**

Faults were injected on right and left ailerons (decoupled), elevators, and rudder over a range of degrees. Only one control surface fault was applied per flight. All flights were autonomous and controlled by the autopilot.

### A. Autopilot

The Pixhawk was the flight controller used for this project, this was connected to the actuators, motors, telemetry, and speed sensors. A multiplexer was used to change from manual mode (primary receiver) where the pilot was in control, to mission mode (secondary), where the autopilot was controlling the aircraft and following the mission plan designed with ground station software. While on a flight test, the pilot would take off, stabilize the aircraft and then switch to mission mode. After about 5 to 7 minutes which was a typical flight duration, the pilot would regain control by switching to manual mode and land the aircraft. The software used as ground station was QGroundControl, where a flight plan was determined beforehand in a specific area. The flight plan was a waypoint multi-point survey. The data captured by the autopilot had a ".ulg"extention, and was used to visualize in PX4 flight review, as shown in Fig. 2 with the flight path. Euler angles, angle rates, thrust, altitude from barometer, GPS, position, velocities, accelerations, ground speed, accuracy of position, voltage, temperature, magnetic field, and satellites are recorded at about 10 samples per second, and they are available download after each flight.

### B. Fault Injection

An Arduino board was used to simulate a fault, by cutting the signal from autopilot to actuators, and replacing that by sending the degree selected to the control surface. Algorithm 3 shows the code for fault injection in Arduino. For flight tests, the Fault IDs were 1 for Left Aileron, 2 for Right Aileron, 3 for Rudder, and 4 for Elevator (please note that these are different from the fault detection IDs in FLFDA). An Arduino board was installed between the Pixhawk and servo motors. The Arduino was then left to pass the signal through except when the channel was changed using a switch in the radio transmitter. Said switch can be in three positions, where the first one was set to no-fault, the second position to the first degree in $\overrightarrow{Fault_{deg}}$, and the third position to the second degree of control surface deflection. The code was adjusted before each flight, only one control surface and two different surface deflection degrees were tested on each flight test.

For ailerons and elevator the angle offset was of about 5 deg. However, for a negative rudder fault i.e., the trailing


**Fig. 2 Flight path seen from PX4**

edge of the rudder is deflected towards the left wing, the offset was of about 10 deg. This means that in reality, although the signal is set to -25 deg, physically the rudder is deflected about -15 deg.

### C. Data Collection

There was a need for an on-board computer to combine data from autopilot and Arduino to have a file with both flight data and fault injection data. A Raspberry Pi (RPi) was selected for this purpose, where an S3 LiPo battery was added to power the RPi. Through a USB, the RPi powered the Arduino and imported the fault data (i.e., $Fault_{ID}$ and $Fault_{deg}$). Another USB connection was installed from RPi to Pixhawk. From this, the flight data (i.e., altitude, attitude, GPS, etc.) was imported through MAVLink. The RPi data acquisition code combined both data sets to a single ".json" file choosing the lower incoming frequency to save them into a micro SD card.

**Algorithm 3** Fault Injection($Fault_{ID}, \overrightarrow{Fault_{deg}}$)

**if** $Fault_{ID} = ControlSurface_{ID}$ **then**
  Cut signal from PixHawk to $ControlSurface$
  $\delta_{ControlSurface} \leftarrow Fault_{deg}$
**end if**

### D. Data Visualization and Analysis

Data from a total of 23 flights was collected in a span of a year. An analysis of the flights was done to find critical state values for FLFDA and to select a maximum "nominal" MD for the MDFDA. The data from ".json" file was analyzed in MATLAB. Both Algorithms were coded in Simulink, where a true real-time simulation was done signaling the inputs in time steps of $dt \approx 0.06s$. Fig.3 show the Euler angles and angular rates through one flight with rudder fault at -20 deg and 20 deg. Notice that data from $\phi$ does not stay within the conventional range of $\pm 180$ deg, this is to prevent the algorithms to detect a fault while turning. The times in which the faults are inserted are shown in Fig.4.

All states are affected by all inputs. However, for FLFDA, after looking at the linear dynamics for a fixed wing airplane 3, each control surface was paired with the states that they affected the most. In Eq. 3, $\overrightarrow{\dot{x}}$ are the state dynamics, $\overrightarrow{x}$ the states, and $\overrightarrow{\delta}$ is the vector containing the input values i.e., throttle, elevators, ailerons, and rudder. Usually $\overrightarrow{\delta}$ includes only throttle, elevators and longitudinal stick, but because the tests were controlled by the autopilot, the inputs were actually the signals from the autopilot to each servo and the motor. The coefficients where then selected for each state anomaly depending on how much the this data changed in comparison to the other faults. After that, the behavior of each state was compared during a fault and normal flight, searching for spikes when the fault was injected. An example of this is at Fig. 3 and 4, where at second 95 a negative 25 deg fault was injected on rudder. $\theta$, $\phi$, $q$, and $r$ have pikes, and the same happens at the other faults. From analyzing these spikes throughout all flight data, the conditions in Table 1 were set for each fault.

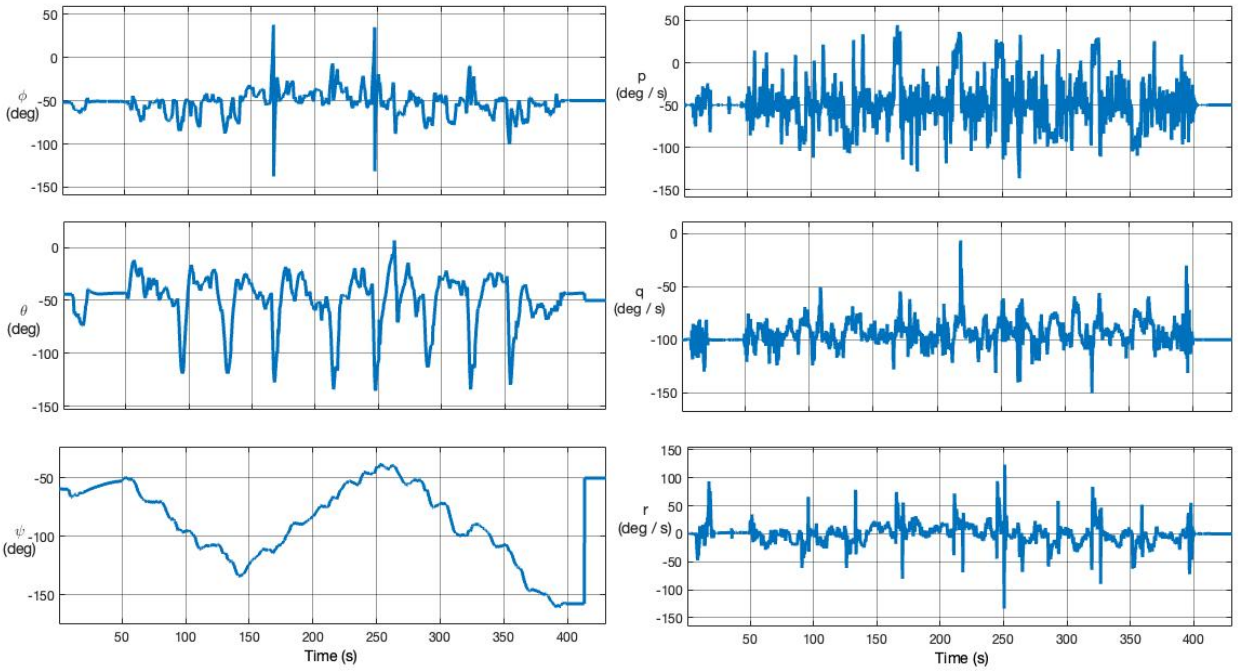


**Fig. 3 Data from flight test with rudder fault**

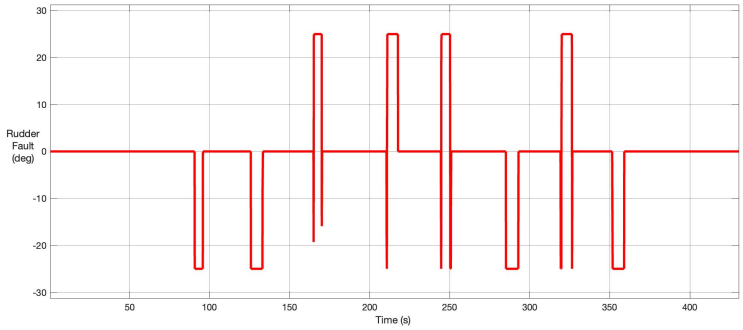


**Fig. 4 Rudder faults applied during flight with deflection signal**

$$\dot{\vec{x}} = F\vec{x} + G\vec{\delta} \tag{3}$$

For MDFDA, after the state matrix was created, and the covariance and inverse covariance matrix, tests were implemented in all flights data. This, in order to select a threshold value for the absolute MD that was the limit of nominal flight. In Section III, this value was set as 1000.

## V. Results

Statistics for both algorithms were computed using F-1 and Matthews Correlation Coefficient. F-1 score is the harmonic mean that combine the precision and the recalling of the model calculated via false positives, false negative, true positives, and true negatives. Mathews Correlation Coefficient (MCC in tables) uses the same variables as F-1 score, but MCC is used to measure the difference between the predicted and actual values of a model. Both algorithms send the output as impulse signals because of this, a fault was detected if there was an output pulse in the time where the fault was active. For FLFDA this is only the signal corresponding to the actual $Fault_{ID}$ is detected. For example, if the real fault was on rudder, only the $Fault_{ID}$ 3 was considered to detect the fault. Often times faults were tested on the

ground while data was being recorded. To account for this, a filter was added to the algorithms to erase the injected faults signal if altitude was less than 20 meters, which is a confidence interval of when. This was done specifically for the code running statistics, so the faults on the ground do not affect false negative rate.

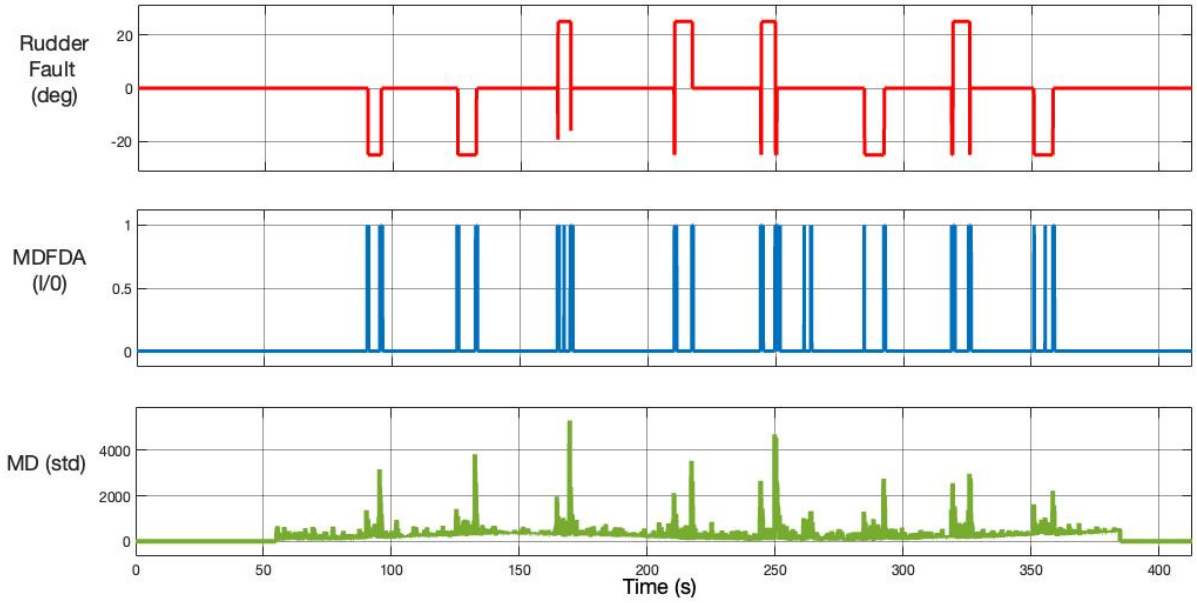


**Fig. 5 MDFDA for Rudder fault at ± 25 deg. First scope shows real fault, middle scope the detected fault, and last one shows MD at each time step**

Fig. 5 shows MDFDA output on the same flight the data in Fig.3 and 4 were collected, with rudder fault at ± 25. The value at the bottom scope is the MD at each time step, in the middle scope is the fault detected by the algorithm, and on the top is the actual fault. Table 2 contains the statistics for MDFDA such as average False Negative (FN) rate (%), False Positive (FP) rate(%), F-1 score, MCC, detection time (sec) that is the time it takes for the fault to be detected once it is present, and detection rate (%) from all 15 flights, flights with aileron faults (7 flights), rudder faults (5), and elevator faults (3). As well as two other categories for faults with deflection smaller or equal than 10deg (7), and greater than 10deg (11). It is worth to note that there were three flights in both categories, with deflections of +15deg and -5 deg.

**Table 2 MDFDA Statistics**

| | FP Rate (%) | FN Rate (%) | F-1 Score | MCC | Detection Time (sec) | Detection Rate (%) |
|---|---|---|---|---|---|---|
| Overall | 1 | 10 | 0.8622 | 0.8628 | 5.44 | 78 |
| Aileron Faults | 1 | 7 | 0.9489 | 0.927 | 2.91 | 90 |
| Rudder Faults | 1 | 2 | 0.9762 | 0.9687 | 4.68 | 86 |
| Elevator Faults | 0 | 34 | 0.7574 | 0.7708 | 14.45 | 62 |
| Faults smaller than \|10deg\| | 1 | 18 | 0.8723 | 0.8702 | 10.7 | 78 |
| Faults greater than \|10deg\| | 1 | 11 | 0.9125 | 0.9009 | 2.98 | 82 |

Fig.6 shows the FLFDA output for that same flight with ± 25 deg fault in rudder. The first scope shows the degree and the time in which faults were injected. The last scope is the higher certainty for each time step. Recall Algorithm 2, where the the three certainties (between 0 and 1) are calculated at each time step, and the one with higher value is the predominant. That certainty is plotted last in Fig.6. If the value reaches higher than 0.4, the fault ID will appear in the middle plot (3 for rudder, 2 for elevator, and 1 for ailerons). It is worth mentioning that there are times where there is zero probability the aircraft is experiencing actuator fault, which is showed at times in scope 3. Table 3 contains the

statistics of FLFDA (FP, FN, F-1 Score, MCC, Detection Time, and Detection Rate) for all flight tests, aileron, rudder and elevator faults separately, small degree fault and large degree faults.

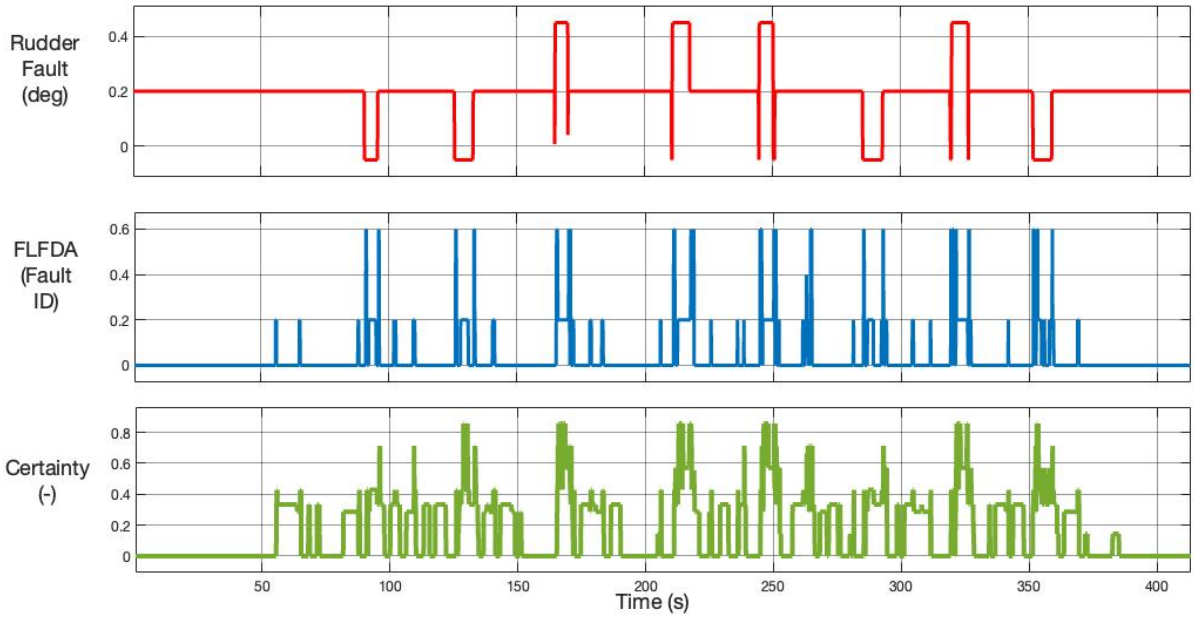


**Fig. 6 FLFDA for Rudder fault at ± 25 deg. First scope shows real fault, middle scope the detected fault ID, and last one shows certainty at each time step**

**Table 3 FLFDA Statistics**

| | FP Rate (%) | FN Rate (%) | F-1 Score | MCC | Detection Time (sec) | Detection Rate (%) |
|---|---|---|---|---|---|---|
| Overall | 8 | 26 | 0.6929 | 0.69112 | 14.13 | 64 |
| Aileron Faults | 2 | 9 | 0.9022 | 0.8734 | 8.4326 | 90 |
| Rudder Faults | 7 | 30 | 0.75472 | 0.67154 | 4.367 | 62 |
| Elevator Faults | 6 | 12 | 0.8695 | 0.8212 | 6.78 | 79 |
| Faults smaller than \|10deg\| | 4 | 19 | 0.8510 | 0.7973 | 9.832 | 72 |
| Faults greater than \|10deg\| | 5 | 30 | 0.7692 | 0.6924 | 5.2646 | 87 |

MDFDA was proved to be more effective overall, with a detection rate of 78%, F-1 score of 0.8622, and a MCC of 0.8628. It was faster to detect faults greater than 10 deg, than smaller, which aligns with the effects a larger deflection degree. MDFDA have a higher detection rate in elevators with 90%, rudder comes next with 86% detection rate and elevator faults has only 62% detection rate.

FLFDA did worse overall, only detecting 64% overall it also is slower to detect faults. However, it detected 90% of aileron faults, rudder and elevator faults detected 62%, and 79% of faults respectively. FLFDA performed better while detecting larger degree faults in comparison with smaller degree faults. In the case of FLFDA, the statistics were calculated only taking into account when the ID of the detected fault matched the injected fault. This is to say that if the algorithm detected an elevator fault (ID equals to 2), when the fault injected was rudder fault (ID, 3), then the algorithm failed to identify the actual fault leaving the code to indicate the fault was not detected.

With FLFDA, the algorithm was able to detect a fault exactly when it was switched in and when it was switched out, although it may not be the same ID that the fault injected. This is interesting to discuss further, this happens because the algorithm is detecting the change in state. When the fault is injected via the radio transmitter, the affected control surface servo moves with torque of about 65 oz/in and breaks the equilibrium of the aircraft. The UAV tends to oscillate

for the abrupt movement, considering the faults are switched in at straight and leveled flight. The same occurs when the fault is disabled and the servo jumps back to the deflection the autopilot is commanding.

## VI. Conclusion

Mahalanobis Distance Fault Detection Algorithm was found to be more effective in both detection rate and time with 78% and 5.44 seconds with a F-1 score of 0.8622. It was proved that this standard deviation measurement works for detecting control surfaces anomalies. The Fuzzy Logic Fault Detection Algorithm has higher detection time and a lower detection rate 14.13 seconds and 64%, but it predicts the faulty control surface with an F-1 score of 0.692. Both algorithms showed better results in aileron faults (90% of detection rate). This could be caused by the offset in servo angles discussed in Section IV. In future work, a combination of both algorithms appears to be promising. Using MDFDA to detect anomalies, and FLFDA to identify the control surface, and deploy the hazard mitigation plan accordingly. In summary, this study demostrated:

- Mahalanobis distance has a higher detection rate in stuck control surfaces.
- Fuzzy Logic has a potential in isolating faults in UAVs and it's worth to keep developing this idea.
- Both algorithms have a higher detection rate in aileron faults 90% for both.
- A combination of both algorithms would be beneficial for high detection rate and lower detection time with fault isolation.

## Acknowledgments

The authors would like to thank Mike Briggs and Axient corporation for their support of this project, as well as Jeriel Cortes from Univeristy of Maryland for piloting the Carbon Cub aircraft for flight testing. Also from University of Maryland, Miles Begley for his contributions to customization of the aircraft hardware/electronics and Nathaniel Wunderly for the data acquisition code on board the aircraft.

## References

[1] Williams, K. W., "A Summary of Unmanned Aircraft Accident/Inciden Data: Human Factors Implication," Tech. rep., Department of Transportation/ Office of Aerospace Medicine, 2004.

[2] Zhang, Q., Wang, X., Xiao, X., and Pei, C., "Design of a fault detection and diagnose system for intelligent unmanned aerial vehicle navigation system," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 233, No. 6, 2019, pp. 2170–2176. https://doi.org/10.1177/0954406218780508, URL https://doi.org/10.1177/0954406218780508.

[3] Gheorghe, A., Zolghadri, A., Cieslak, J., Goupil, P., Dayre, R., and Le Berre, H., "Model-Based Approaches for Fast and Robust Fault Detection in an Aircraft Control Surface Servo Loop: From Theory to Flight Tests [Applications of Control]," *IEEE Control Systems Magazine*, Vol. 33, No. 3, 2013, pp. 20–84. https://doi.org/10.1109/MCS.2013.2249417.

[4] Cork, L., and Walker, R., "Sensor Fault Detection for UAVs using a Nonlinear Dynamic Model and the IMM-UKF Algorithm," *2007 Information, Decision and Control*, 2007, pp. 230–235. https://doi.org/10.1109/IDC.2007.374555.

[5] Heredia, G., Caballero, F., Maza, I., Merino, L., Viguria, A., and Ollero, A., "Multi-Unmanned Aerial Vehicle (UAV) Cooperative Fault Detection Employing Differential Global Positioning (DGPS), Inertial and Vision Sensors," *Sensors*, Vol. 9, No. 9, 2009, pp. 7566–7579. https://doi.org/10.3390/s90907566, URL https://www.mdpi.com/1424-8220/9/9/7566.

[6] Douglas, R. K., and Speyer, J. L., "Robust fault detection filter design," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 1, 1996, pp. 214–218. https://doi.org/10.2514/3.21600, URL https://doi.org/10.2514/3.21600.

[7] Lin, R., Khalastchi, E., and Kaminka, G. A., "Detecting anomalies in unmanned vehicles using the Mahalanobis distance," *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 3038–3044. https://doi.org/10.1109/ROBOT.2010.5509781.

[8] Brotherton, T., and Mackey, R. M., "Anomaly detector fusion processing for advanced military aircraft," *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, Vol. 6, 2001, pp. 3125–3137 vol.6.

[9] Perry, T., "Lotfi Zadeh and the Birth of Fuzzy Logic]," *IEEE Spectrum*, Vol. 32, No. 6, 1995, pp. 32–35. https://doi.org/10.1109/6.387136.