AIAA Infotech@Aerospace 2010
20 - 22 April 2010, Atlanta, Georgia

AIAA 2010-3419

Health Monitoring via Neural Networks

Daniel V. Uhlig* and Michael S. Selig† and Natasha Neogi‡
University of Illinois at Urbana-Champaign, Urbana, IL 61801

Monitoring of semi-autonomous systems increases situational awareness and is highly applicable to small remotely piloted aircraft. The complex dynamics of aircraft flight were abstracted by modeling the system with a neural network. The neural network accurately modeled the non-faulty dynamics for comparison to the actual system. Since, the system included unmodeled lag, the residual calculation needed to account for unmodeled lag. The method presented here decreased the effects of the lag on the residual. A ratio between the predicted and measured system outputs was used to understand the faults and understand how the future performance will be affected by the fault. A state machine abstraction offered a flexible fault-detection framework that can be adapted and designed to detect different faults without requiring a full dynamics model of each fault. Through using abstractions, the complex dynamics models required for fault detection were simplified and a state machine offered the flexibility to detect different suites of faults.

Nomenclature

c	=	counter
p	=	roll rate
q	=	pitch rate
r	=	yaw rate
R	=	residual
R_D	=	direct residual
R_L	=	time-lagged residual
T	=	threshold
t	=	time
V	=	airspeed
Y_p	=	predicted system output
Y_m	=	measured system output
α	=	angle of attack
β	=	sideslip angle
δ_a δ_e δ_r	=	aileron, elevator and rudder deflections
ϕ	=	roll angle
θ	=	pitch angle
ψ	=	yaw angle

I. Introduction

The proliferation of small uninhabited aerial vehicles (UAVs) operating semi-autonomously over long distances require local on-board monitoring of system performance. Monitoring allows a system operating with a degree of autonomy to alert operators or other systems when a fault is detected that degrades the system performance. Software-based monitoring does not require additional sensors or hardware and can easily be added to small UAVs. Software based fault monitoring may be key to increasing the safety and trust of small UAV systems so they can have wider use in civil airspace.[1] Fault monitoring and alerting allow the remote operator or remote pilot to quickly know when an aircraft flying autonomously needs additional attention.

In order to monitor system behavior and alert the operator when the system is not operating as expected, an accurate model of the system is needed. Aircraft flight dynamics is traditionally modeled as a six degree-of-freedom (6 DOF) system, that can be linearized at given angles of attack. The linearized model is only accurate close to the point of linearization, so the linear model will break down away from the point of linearization. A limited model of behavior

*Graduate Research Assistant, Department of Aerospace Engineering, AIAA Student Member.
†Associate Professor, Department of Aerospace Engineering, AIAA Senior Member.
‡Assistant Professor, Department of Aerospace Engineering, AIAA Member.

can be generated from the geometry, but higher-accuracy models require higher-fidelity data. This data can be acquired from wind tunnel tests, computational fluid dynamics and flight test. Flight testing generates the most realistic results, but with many limitations. In order to build accurate models for fault detection, model fitting must be done and the accuracy of the model must be verified. Beyond just modeling a set of complex decisions must be made to detect and diagnose different faults.

Instead of trying to fully model the complex system of an airplane, a number of levels of abstraction made the problem simpler and more tractable. Rather than developing complex continuous equations of motion, a black-box estimator was developed in the form of a neural network. This black-box abstracted the dynamics into a set of inputs and outputs that allowed residuals to be calculated. As the next level of abstraction, a state machine was used to monitor and observe the system. A state machine allowed complex decisions to be made on a small set of variables that resulted from the outputs of the black-box model. Since there was just one dynamics model, a more complex set of decisions was made within the state machine. Instead of spending time developing and checking numerous nominal and faulty dynamics models, a single nominal dynamics model and a well-designed state machine could accurately predict faults.

A. Background

An artificial neural network is a collection of nodes that are simple processors.[2–4] Each node takes a set of inputs and generates a set of outputs which are minimized to match the target outputs supplied during training. Through randomized iterative processing, the nodes converge to modeling the behavior of the input and output data. Neural networks can model non-linear relationships between input and output states. The performance of a neural network is based on training the neural network using existing test data, and they can be used to model aircraft behavior. Neural networks offer a way to rapidly model non-linear dynamics over a wide range of flight conditions that can be used for health monitoring.

A method was developed using neural networks for fault tolerant non-linear controllers.[5] An augmented model inversion controller was applied to a tilt rotor and an neural network was combined with linearized feedback to control the aircraft during different stages of flight and during failures. The authors explored the stability of single hidden-layer neural networks. Neural networks have been used to implement adaptive fault tolerant controllers.[3,6] The effectiveness of the controller depended on the selection of inputs and required tuning in order to be fault tolerant.[6] In the literature, the importance of the selection and structure of inputs and outputs was also noted.[3]

Neural networks map non-linear behavior well from limited data and have been used for system identification.[4,7] Neural networks have been used to identify nonlinear parameters over a wide range of flight conditions. They were used to model forces and moments on a turboprop over a wide range of angles of attack and sideslip angles.[8] The neural network accurately predicted the non-linear aerodynamic coefficients. Not only are neural networks accurate, but often, after the training, they are computationally efficient. Neural networks were investigated as a computationally more efficient method to model aircraft data used in flight simulators.[2,8] Once trained, the neural networks accurately simulated the non-linear aircraft dynamics much faster than a traditional coefficient-based simulator.

The wider problem of fault detection can be applied to the subset of aircraft health monitoring.[9] In general residuals are calculated as the prediction error. When the residuals exceed a threshold, a fault is detected. Depending on the set of observed changes in residual(s), a decision matrix (often very large) can be used to identify different failure modes of the system.[10] The threshold can be a static value or the threshold can vary dynamically depending on the system excitation.[11,12] A dynamic threshold allows the fault threshold to vary with the system excitation, making the fault detection work over a larger range of excitations. Instead of comparing modeled dynamics, the inputs can be reconstructed from the measured system response. The reconstructed control inputs are compared to the actual control inputs and mismatches indicate a fault.[13] In addition, instead of comparing against a model, multiple independent predictions can be used. An example in the literature used independent inputs to different neural networks to classify the maneuver that a helicopter was undergoing.[14] If there was disagreement between the independent neural networks then the system was assumed to be faulty. Many different modeling techniques and ideas can be used to generate residuals and a variety of techniques can be used to find when a fault has occurred.

B. Aircraft Models

Aircraft flight dynamics models have been developed to model aircraft motion as an inertial mass in translation and rotation with six degrees of freedom.[15–17] The six degree-of-freedom system has three translation and three rotation axes where the full nonlinear equations are generally linearized at the low angle of attack equilibrium position. In order to control the system, a number of actuators are available, and in this paper four basic actuators were used: ailerons, elevator, rudder and throttle.

Each control surface affects the system in a different manner. In order to understand how each fault will be observed in the system, the effect of different control surfaces must be understood. A transfer-function relationship between the control surfaces and the outputs for perturbed flight had been developed.[17] Each control surface primarily

Figure 1. 33%-scale Edge 540 aerobatic RC model used in the simulations.

affects one to three state variables, and the results are shown in Table 1. Both the ailerons and rudder affect roll angle and heading, but the aileron generates a roll rate while the rudder generates a sideslip rate to change heading and affect the roll through coupled dynamics. The elevator is used for pitch control and aircraft trim. Initially it drives the pitch rate, but then holds a steady condition. This knowledge forms the basis of developing the neural networks by selecting inputs and outputs that result in accurate fault detection performance.

Table 1. Control surface deflections and most correlated aircraft state output.

Aileron	$\Rightarrow$	$\phi, \dot{\phi}$
Rudder	$\Rightarrow$	$\dot{\beta}$
Elevator	$\Rightarrow$	$\alpha, \theta, \dot{\theta}$

C. Aircraft Simulation

In order to easily gather data similar to a flight test, a flight simulator was used. The data from the simulator was used to train the neural networks and investigate failure cases. FS One™ was developed to simulate small RC aircraft and can easily be applied to small scale UAVs.[18] The simulator models aerodynamics on a component-by-component basis over the linear and non-linear flight regimes. Forces and moments from each component, e.g., wing, horizontal tail, vertical fin, etc., are independently calculated and then summed for the total airplane. Because the model is built-up from components, any change in the aerodynamics of an individual component can easily be modeled without having to recalculate the force and moment coefficients for the entire aircraft.

The aircraft used within the simulator was a 33%-scaled version of the aerobatic Edge 540 produced and distributed by Horizon Hobby under the Hangar 9 brand.[19] The simulator model is based on an RC model that has a wingspan of 97 in (2.46 m) and a length of 88 in (2.24 m). The aerobatic RC airplane weighs approximately 25 lb (55 kg) and is powered by a 100-cc gas engine.

II. Models of Aircraft

Using data from FS One, a nominal neural network was trained on various inputs and outputs. The neural network was a single hidden-layer feed-forward neural network setup with 10 hidden nodes and a delay of 0.1 sec between the input state variables and the output state variables.[20] Ten nodes were selected because the nominal networks had ten inputs. The delay was selected to be long enough for the neural network to easily process the older inputs and make a prediction against the current time. A number of different neural network input and output sets were investigated. Initially, a single input-to-output mapping labeled as ‘angular rates’ in Table 2 was explored since there was just one neural network with a set of outputs being the angular rates. In evaluating the neural network applicability to fault detection, two performance criteria were used. First, the neural network needed to be able to accurately predict the nominal model, and second, the model needed to show a significant difference with the injected faults. The ‘angular rate’ model (see Table 2) accurately predicted the dynamics and showed a significant difference with aileron faults using the roll rate. Faults on the other surfaces only caused minor changes in the neural network predictions. In order

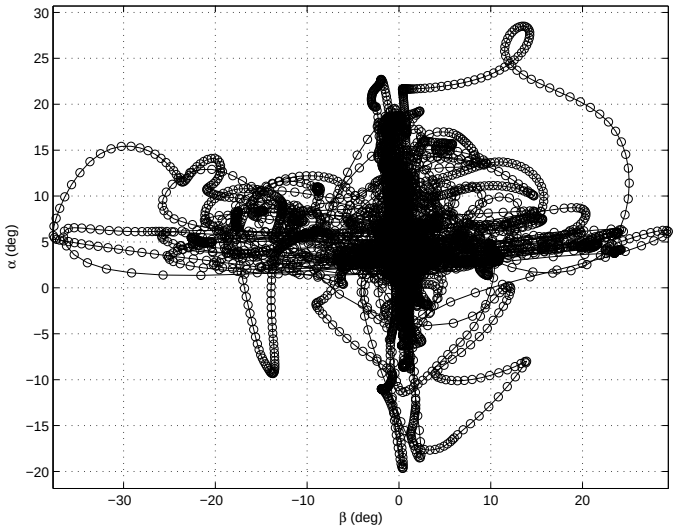

Figure 2. Angle of attack versus sideslip angle for the neural network training data from a 6.6-min flight simulation with the Edge 540.

to better observe faults for the different surfaces, individual neural networks were investigated. The neural networks were based on the effects of the control surfaces shown previously in Table 1.

Using the relationships in Table 1, additional neural networks were constructed and explored. The different neural networks described in this paragraph are listed in Table 2. The 'roll network' observed aileron faults through the roll rate. Rudder faults affected the yaw or sideslip rate which is the output of the 'yaw network'. The 'pitch network' observed elevator faults.

Table 2. Neural network inputs and outputs for different cases

Neural network	Inputs (0.1-sec delay)	Outputs
Angular Rates	α, β, V, ϕ, θ, ψ, δ_e, δ_a, δ_r δ_T	p, q, r
Roll Network	α, β, V, ϕ, θ, ψ, δ_e, δ_a, δ_r δ_T	p, ϕ, $\dot{\beta}$
Yaw Network	α, β, V, ϕ, θ, ψ, δ_e, δ_a, δ_r δ_T	$\dot{\beta}$
Pitch Network	β, V, ϕ, θ, ψ, δ_e, δ_a, δ_r δ_T	q, α, $\dot{\alpha}$

The neural networks were trained on the sets of input and output variables specified in Table 2. There was a 0.1-sec delay between the inputs and outputs in an attempt to model the system lag. The flight data came from an unplanned 6.6-min flight that covered a wide range of angles of attack and sideslip angles as shown in Fig. 2. The flight covered more than just the linear flight regimes and included stalls, inverted flight and flight with high angular rates. The deflections in elevator, ailerons and rudder were approximately $\pm$15, $\pm$20, and $\pm$27 deg, respectively. By covering a range of flight conditions during the neural-network training, the neural network matched some flight performance cases outside of the linear regime. The neural-network training covered a large enough range of non-linear flight conditions to be able to accurately predict performance in most flight regimes.

The trained neural network accurately simulated flight and predicted aircraft state variables based on previous state and control inputs. If the control inputs changed due to an unknown change in the control surfaces (a fault), the output of the neural network no longer matched the expected behavior. Each surface changed the output in a different manner, and the change in behavior could be used to investigate faults.

III. Observing Faults

The faults addressed in this paper deal with a continuum of control surface failures. Many control surface faults are manifested in a continuum from a small effect to large effect. In this paper, the suite of faults selected were under-responsive control surfaces which could represent a variety of underlying problems. The selected faults varied from a completely non-responsive control surface to one that only moved a percentage of the commanded amount. The non-responsive control surface was the most extreme fault, while a control surface that moved 50% was the least extreme. There are infinite specific faults between 0% and 100% non-responsive. The faults modeled as an under-responsive control surface represent a number of different failures.

The set of faults explored can be caused by a variety of problems. Damage to the control linkage, incorrect control

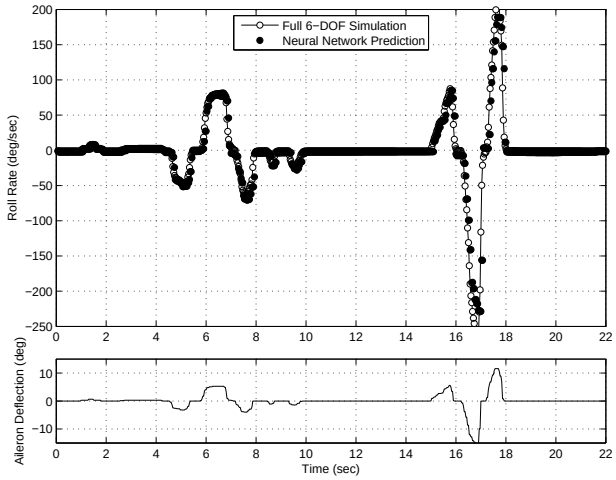

Figure 3. The full 6-DOF simulation and neural network predictions for the roll rate for the nominal system.

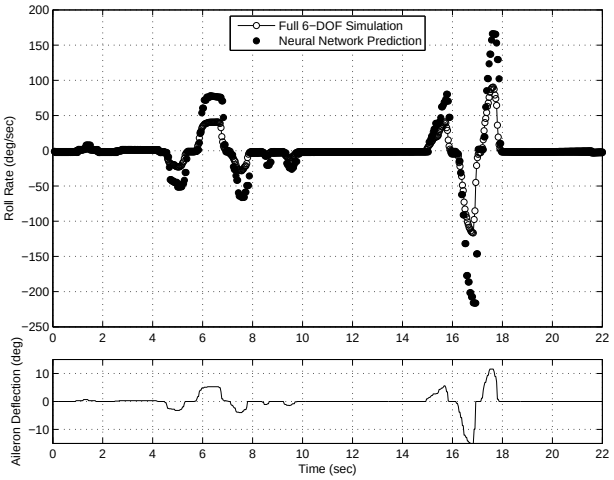

Figure 4. The full 6-DOF simulation and neural network predictions for the roll rate for a faulty system with one non-responsive aileron.

logic, and damaged or missing parts of control surfaces could all be the root cause of the fault. In-flight damaged such as battle damage or a bird-strike, can cause a control surface to loose effectiveness. The result would be the loss of a percentage of the nominal control effectiveness of a surface. A number of different test cases are used within this paper to explore the range of faults.

A. Modeling Control Surface Faults

The two ailerons can either fail individually, becoming frozen at a position or not deflecting as much as commanded. Figure 3 shows the behavior for a pair of aileron doublets with a left roll and then a right roll where the second doublet is larger than the first. The experiment with both ailerons working properly is shown in Fig. 3. As seen in Fig. 3 The neural network accurately predicted the roll rate of the aircraft during the maneuvers; the simulator data and neural network prediction practically overlap. In order to see if the neural network observed a fault, the same doublet experiment was repeated with one aileron locked at approximately neutral. In this case the airplane banked at a slower rate than the neural network expected (Fig. 4). The difference in roll rate allowed a residual to be calculated and faults to be identified (see following section “Calculating Residuals”).

The single rudder control surface can fail by not deflecting as far as commanded. Figure 5 shows the behavior of the yaw rate during a flight with a rudder fault injected at 12 sec. In non-faulty flight, the rudder caused a change in the yaw rate and the neural network accurately predicted the immediate change in yaw rate. When the rudder returned to neutral, the yaw rate decayed through under-damped oscillations, which the neural network predicted. When the rudder became half effective (after 12 sec) the immediate change in sideslip rate decreases. The neural network predicted higher yaw rate during the initial deflection while it still accurately predicted the sideslip during the decay (under-damped motion). Rudder faults can be detected by observing the yaw rate response as the rudder is deflected.

Finally, the pitch control is managed by a left and right elevator that move in parallel. Both sides can fail as a pair or just one side can fail alone. A failure can be represented by a degraded deflection or a frozen control surface. The airplanes pitch rate depends on airspeed and elevator deflection. In normal flight, the neural network accurately predicted pitch rate over a range of elevator deflections using the inputs to the ‘pitch-rate’ neural network (see Table 2). The result of the pitch-rate estimate for the elevator is shown in Fig. 6. The figure shows the nominal elevator before a fault is injected at approximately 8 sec. Before the fault was injected the neural network accurately predicted the pitch rate. After a fault was injected at 8 sec causing the elevators to deflect 33% of the commanded amount, the neural network over-estimated the pitch rate. The neural network system model showed the potential to detect faults by using the mismatch in the prediction and measured outputs.

The next case considered was an elevator failure where one side was frozen at 0-deg deflection. In this case, the pitch rate is expected to be lower as a result of having less elevator authority. Figure 7 shows the flight of the airplane with one elevator frozen at neutral and the prediction of the neural network. In Fig. 6 that fault was injected after the flight started, but in both cases the pitch rate showed how the neural network did not predict the pitch rate properly. Decreased effectiveness in the pitch axis can be observed by the mismatch between the neural network prediction and system response and can be used to detect faults by calculating a residual.

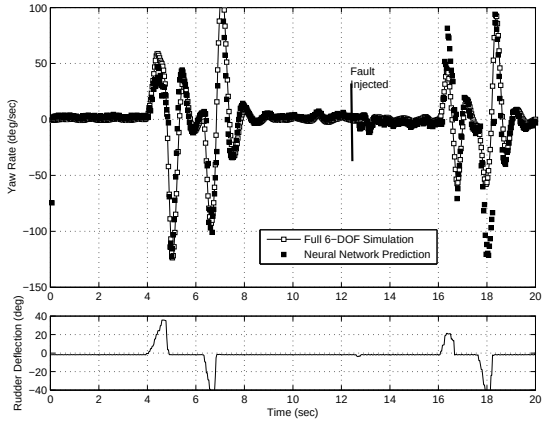

Figure 5. The full 6-DOF simulation and neural network predictions for the yaw rate for faulty and non-faulty systems with rudder deflections.

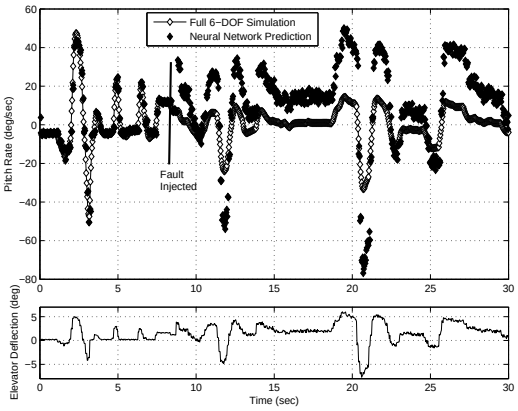

Figure 6. The full 6-DOF simulation and neural network predictions for the pitch rate for nominal elevator with a 33% effective elevator response fault injected at approximately 8 sec.

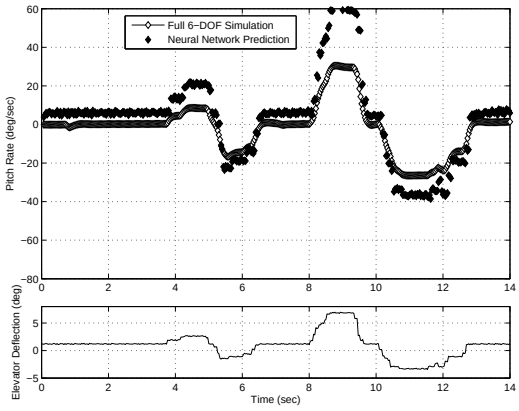

Figure 7. The full 6-DOF simulation and neural network predictions for the pitch rate for the right elevator locked fault.

B. Calculating Residuals

Residuals are the error between the neural network predictions and the actual system behavior at every timestep. A small residual occurs during the nominal non-faulty flight due to an imperfect neural network model. After a fault is injected the residual increases significantly because the actual system dynamics changed. Depending on the characteristics of the residual, specific faults can be identified within the system. Since the residual is key to identifying the faults, different methods to calculate the residuals were explored.

The residual was calculated two ways each represented the difference between the neural network system model and the actual system model. First, the direct residual, R_D, was calculated by comparing the neural network prediction of the output, $Y_p(t)$ and current output of the system, $Y_m(t)$:

$$R_D(t) = Y_p(t) - Y_m(t) \tag{1}$$

As shown in Fig. 8, the direct residual had spikes during the transient behavior as the control surface changed deflection. It was observed, in systems with unmodeled lag in the response, models will have prediction error at the instance when the inputs are changing. The comparison of neural network prediction to the actual model had unmodeled lag since the input only had one time step of delayed data–not a time history. The second method attempted to calculate the residual in the presence of unmodeled system lag.

In order to overcome the unmodeled lag, the calculation of the residual was modified to include a time history. Instead of just comparing the current time, t, a limited number of previous time steps, n, were included in the comparison. The current time estimate from the neural network was compared to the last few seconds of actual measurements as:

$$R_T(t) = \min_{i=1:n} \left\{ Y_p(t) - Y_m(t-i) \right\} \tag{2}$$

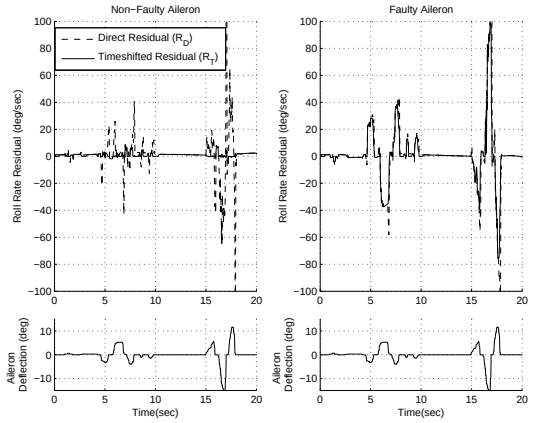

Figure 8. Comparison of the methods to calculate the residual for faulty and non-faulty ailerons.

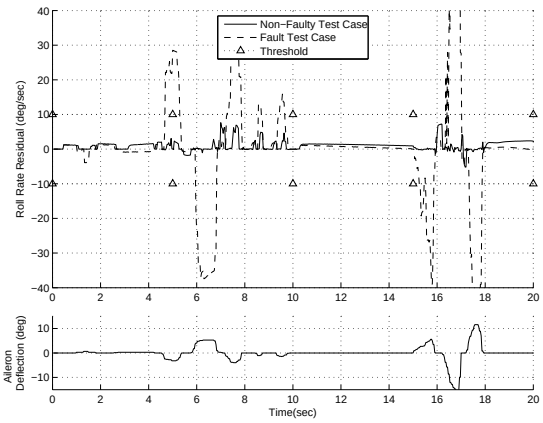

Figure 9. A residual trace comparison between nominal aileron and faulty aileron behavior.

The time-lagged residual, R_T, limited the spikes due to lag because as the system starts changing, the neural network was compared to a range of values from the real system, $Y_m(t-i)$, that included before and after the input changed. The range of data allowed the neural network predictions to have imprecise system lag and still model the system dynamics well. As shown in Fig. 8, the residual that allowed for unmodeled time delays had a closer match in error between predictions and the real system, but the time lagged residual still increased to similar values in the case of a fault. By allowing the shift in time when calculating residuals, the effect of any unmodeled lag in the system was decreased.

C. Residual Threshold

A threshold was used to differentiate the faults from non-faults. Instead of using a dynamic threshold which depends on the excitation at any given time to detect a fault, a set threshold value was used for the pitch, roll and yaw rate. A single set value was simpler to select than a dynamic threshold, but the ideas presented would be applicable to an adjustable threshold. Figure 9 shows the residual calculation during a flight with faulty ailerons. First, during the non-faulty flight the residual had some minor spikes, but always stayed below the threshold. Second, in the faulty flight, the fault was initially not detectable since the aileron was deflected and the roll rate was close to zero. All the control surface faults were latent or not observable when the control surface was not deflected and the angular rate was close to zero. Once the first doublet was started at 5 sec (Fig. 9), the residual spiked and a fault was detected. Finally, the residual continues to decrease as the roll rate decreases and spikes anytime the aileron is deflected, detecting the error. The constant threshold showed that the residual method detects faults when the faulty surface is deflected.

D. Tracking Errors

The residual was the difference between the prediction and actual response. By taking the ratio of the predicted to actual response an error ratio was calculated. The error ratio provided insight into the faults as illustrated in Fig. 10 which shows the error ratio for three different aileron faults. First, there was a regime of low excitation where the difference between faulty and nominal cannot be observed. This region of low excitation corresponded to the latent region. Second, the ratio depended on the percentage, the control surface was not moving. This allowed different faults to be distinguished and estimated. Beyond just identifying the control surface that was faulty, an understanding of the how the faulty surface was behaving increases the situational awareness and the increased understanding would allow supervisory and remote pilots to make better decisions.

By tracking the error ratio, the control surface effectiveness was estimated. Anytime a fault was observed (the residual was greater than the threshold), the ratio of predicted to actual response was calculated and stored. By storing the ratios during the faulty time-steps, the faulty surface effectiveness was calculated. The error ratio showed how the performance was changed in the presence of a fault. The result of calculating the error ratio during the periods of residual spikes, brought insight into what fault had occurred and would allow for decisions to be made that could include modifying the control inputs.

In the case of a locked aileron, the ratio between predicted and actual roll rate was approximately 50% which was expected since each aileron contributes half of the roll control. Similarly, when one aileron was half effective, the ratio was approximately 75%. Table 3 shows how different faults affected the ratio of predicted to actual response. By tracking the error ratio during periods when the residual was above the threshold, more information about the fault could be discovered.

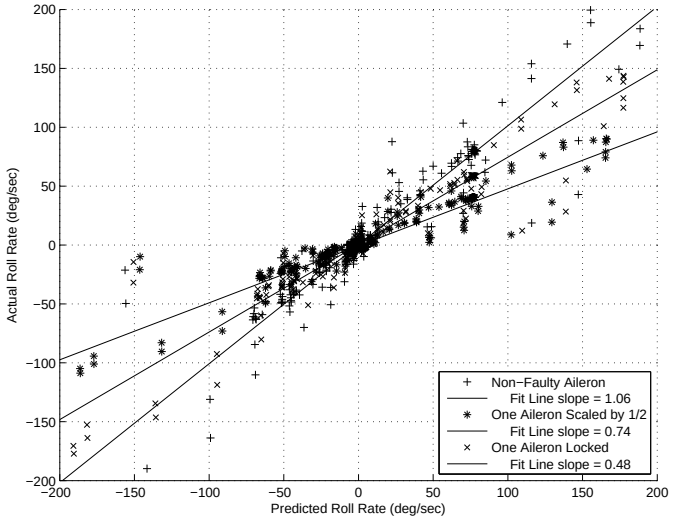

Figure 10. The slopes of the 6-DOF freedom roll rate versus the neural network prediction for roll rate for different aileron faults.

Table 3. A table showing the different control surface faults and their effects on the error ratio.

Fault	Percentage of nominal control surface response	Observed error ratio
Nominal aileron	100%	1.01
One aileron locked	50%	0.46
One aileron deflection scaled by 50%	75%	0.72
Nominal elevator	100%	1.11
One elevator locked	50%	0.54
Elevator deflection scaled by 33%	33%	0.29
Elevator deflection scaled by 50%	50%	0.56
Nominal rudder	100%	1.08
Rudder deflection scaled by 25%	25%	0.32
Rudder deflection scaled by 50%	50%	0.46

To model each of the faults individually would require numerous models since the control surface faults happen in a continuum range between completely non-responsive to partially-effective to nominal performance. Tracking the error ratio overcomes this problem through additional calculations on the output variables from the prediction and actual system. Instead of looking at a wide range of complex models, a single simple model of neural network abstracted dynamics, allow more complex decisions to be made on the relationship between the neural network predictions and the system outputs.

IV. Detection via State Machines

State machines can represent complex decision making processes through a set of states, transitions and actions within the states. From each state, changes in transition variables drive the state machine to a new state. Within in a state, a number actions are completed to alert the user, update internal calculations and finally select the next state. From the time evolution of the observed variables, the state machine makes discrete decisions at each time-step. Fault detection on a limited number of observation variables requires complex decision making which is easily implemented in a state machine framework. By changing the state depending on the outputs of the system and the neural network model of the system a fault detection state machine was developed.

The flexibility in designing the state machines response to different observations and the ease of redesigning without changing how the observations are generated make state machines applicable to fault detection. The transition decisions can be based on complex functions of the observed variables as well as the current state of the system. The functions used for transitions in the state machine can be changed as the suite of faults that need to be observed changes. Given different fault models, a state machine can be designed to detect different suites of faults without needing to alter the underlying system model. The flexibility allows additional types of faults to be added to the detection scheme as the faults are discovered.

State machines offer flexibility in the design of states and transitions to detect the different faults. Minimal knowledge about the fault is required before designing the system model since the state machine can be customized, a

posteriori, when detailed models of the faults are known. Instead of requiring numerous complex system dynamics models, the state machine offers a flexible framework to make complex decisions to detect a variety of faults using a single system model.

A. Application to UAV Fault Detection

In constructing the state machine, a few assumptions were made on the nature of the faults. The assumptions allowed a simple state machine to be constructed that could monitor the previously discussed faults. First, the aircraft was assumed to start in a non-faulty or nominal state. Second, the faults were assumed to be permanent which meant after a fault appeared no attempt was made to find when it left the system. The state machine designed here was a simple four-state state machine shown in Fig 11 and explained in Table 4. The model was used to predict faults using the residual and an internal counter.

The key transition variable was if the residual was greater than or less than the threshold. In order to avoid false positives, a minimum of three consecutive residuals must be greater than the threshold before the fault becomes permanent. While in the faulty state, the error ratio was calculated to provide information on the fault. As previously discussed, the error ratio provided information on how the fault was affecting the control surface. In the latent fault state, the error ratio was not calculated, but the system was still assumed to be faulty. The dashed line back to nominal requires a decision depending on if the faults can be proven to no longer exist. Proving a fault has left the system would require a decision depending on the excitation level being high enough for the fault to be seen, but the fault not appearing. However, since the faults were assumed to be permanent, the transition from faulty or latent to nominal was not allowed.

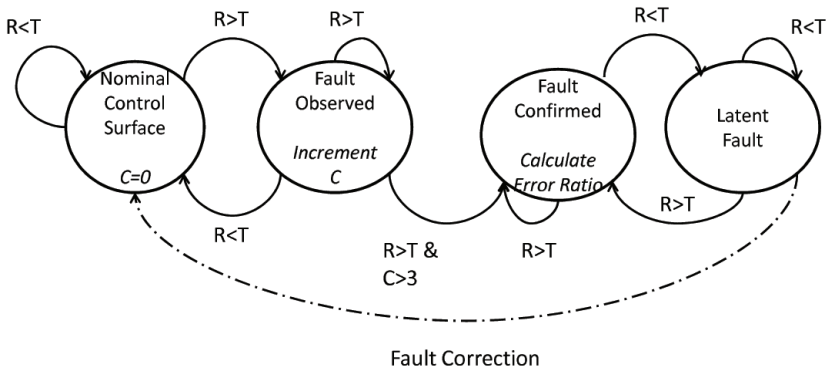

Figure 11. The simple four-state state machine used for fault detection.

B. Results for Fault Detection on UAV

In order to illustrate the state machine two example flights will be used. Figure 12 and 13 respectively show the behavior of the residual in the case of a faulty elevator and a faulty rudder. Starting with the elevator example (Fig. 12), the state machine started in the *nominal* state. The residual was well below the threshold for the first 8.5 sec and no transitions occurred. After the fault was injected (approximately 8.5 sec), the residual stayed below the threshold for a short period of time. Once the residual exceeded the threshold (approximately 9 sec) the transitioned to the *fault observed* state occurred. Since the residual continued to exceed the threshold, the counter incremented to three and then the state machine transitioned to *fault confirmed*. In this state the error ratio was calculated. Once the residual became lower than the threshold (approximately 9.5 sec) the state machine transitioned to the *latent fault* state. In this state, the fault was still present, but the ratio was no longer calculated. Just before 11 sec, when the residual again exceeded the threshold, the state machine transitioned back to *fault confirmed* and the error ratio continued to be calculated. The average error ratio measured during the flight was 0.29, which was close to the injected fault of 33% effective elevator. The transitions continued between latent fault and fault confirmed for the remainder of the flight.

In the second example shown in Fig. 13, the system also started in the nominal state. After the fault was injected, since the rudder was not excited, the fault stayed latent and unobserved. Once the rudder was deflected, residual spiked at 16.5 sec, and the state machine transitioned to *fault observed*. However, the sideslip rate excitation was brief, so the counter was less than three when the residual became less than the threshold. The state transitioned back to *nominal* and reset the counter to 0. At 17.5 sec, the state transitioned back to fault observed, and this time the counter exceeded three. Then the state transitioned to *fault confirmed* and the error ratio was calculated. With four seconds of being injected the fault was observed, but not confirmed. It was confirmed with a longer, larger deflection a couple of

Table 4. Finite state machine for the fault detection.

State: Nominal Control Surface

Entry Action: $counter = 0$

Next State: if $Residual < Threshold \rightarrow$ Nominal Control Surface

Next State: if $Residual > Threshold \rightarrow$ Fault Observed

State: Fault Observed

Entry Action: $counter = counter + 1$

Next State: if $Residual < Threshold \rightarrow$ Nominal Control Surface

Next State: if $Residual > Threshold \,\&\, counter < 3 \rightarrow$ Fault Observed

Next State: if $Residual > Threshold \,\&\, counter > 3 \rightarrow$ Fault Confirmed

State: Fault Confirmed

Entry Action: Calculate Error Ratio

Next State: if $Residual < Threshold \rightarrow$ Latent Fault

Next State: if $Residual > Threshold \rightarrow$ Fault Confirmed

State: Latent

Entry Action:

Next State: if $Residual < Threshold \rightarrow$ Latent Fault

Next State: if $Residual > Threshold \rightarrow$ Fault Confirmed

Next State: if *Fault corrected* $\rightarrow$ Nominal Control Surface

seconds later. The error ratio was calculated to be 0.46 during the periods in the *fault confirmed* state which was close to the injected fault of a 50% effective rudder. This example shows how latent faults can take longer to observe.

Both of these examples illustrate how the complex faults were accurately detected through the two layers of abstraction. First, the neural network modeled the system and allowed an accurate residual to be calculated. Second, a state machine abstraction was used to monitor the residual and detect the faults. The state machine used the residual as the main decision variable, and calculated an error ratio to further understand any detected fault. By using the error ratio the faults effect was calculated and then could be used to predict how the system performance will change.

Table 5 shows the average time to detect a fault after exciting the axis. For each control surface two different faults were used multiple times and specifics are listed in the table. The average time required to observe a fault depended on the excitation levels, the control surface and the dynamics model. The times listed the delay from the fault injection and the surface being excited to the fault detection.

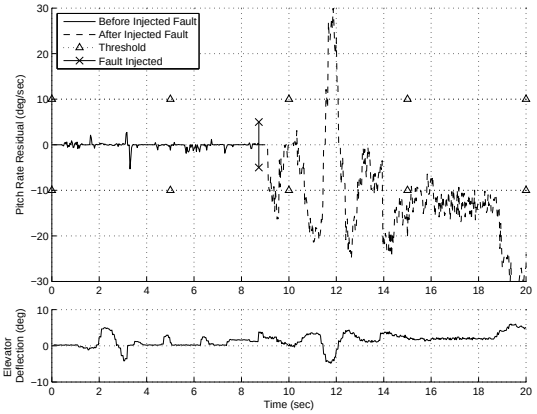

Figure 12. The residual traces and fault detection for a elevator fault (scaled by 33%).

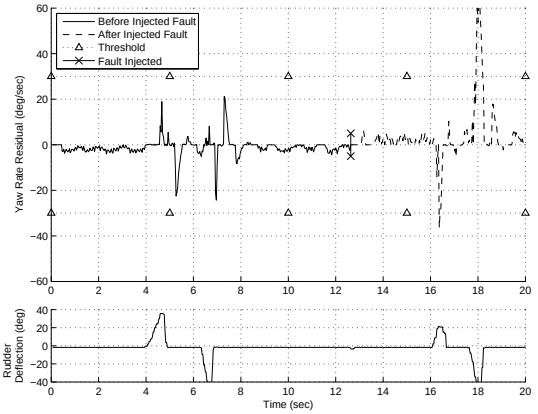

Figure 13. The residual traces and fault detection for a rudder fault (scaled by 50%).

Table 5. Fault detection results for neural network modeling in a state machine framework.

Control surface	Average time to observe fault (sec)	Test cases	Types of faults
Aileron	1.48	6	50% Left aileron
		6	Locked left aileron
Elevator	0.158	6	50% elevator
		6	33% elevator
Rudder	1.29	6	50% elevator
		5	25% elevator

V. Conclusions

Neural networks presented a way to simulate a physical system for health monitoring and fault detection. In order to observe the faults, the inputs and outputs of the neural network needed to be selected carefully to accurately model the system. Using non-linear aircraft models from existing flight simulators, the effects of the different control surfaces were found. By selecting inputs and outputs that were needed, faults on the aileron, rudder and elevator were observed using a neural network that performed well over a range of flight conditions. By comparing the predicted outputs and the measurement outputs a residual was calculated.

The residual was calculated using a set of current and older measured outputs to better handle the unmodeled lag in the system. Neural network predictions of flight parameters were good particularly outside of the instances of control surface movement. The unmodeled system lag limited the model accuracy in the short term when the control surfaces were deflected. By modifying the residual calculation a closer match between the predictions and measurements was achieved. The more accurate time-lagged residual was used by the state machine to detect faults within the system.

A simple state machine was constructed to track and make decisions on the fault state. If the residual exceeded the threshold, a fault was observed and once a fault was observed in a set of consecutive measurements the fault was confirmed. While in the faulty state, a ratio of the predictions to measurements showed how the fault was affecting the control surface effectiveness. The ratio placed the fault within the continuum of potential faults that the state machine had been designed to detect.

The state machine framework was selected because it represented a flexible framework that can be designed to detect the expected faults allowing the neural network dynamics models not to be redone for every new fault. Instead, the state machine could be updated as additional faults were discovered and understood. By using the state machine abstraction to monitor a system and detect faults, the underlying physics model could be simplified.

Detecting faults by using a complex dynamics model and comparing these models to actual outputs through a residual using state machines was presented here. Neural networks were trained to accurately predict the aircraft dynamics over a range of flight regimes. The state machine framework allowed flexibility in the fault detection without redesigning the dynamics models. Neural network predictions can be used with a state machine abstraction to observe control surface faults on an small airplane.

Acknowledgments

The authors would like to thank the Boeing Corporation for their support under the grant Boeing ITI RPS #15, and Dr. Jae Kim.

References

[1]Hughes, D., "UAVs Face Hurdles in Gaining Access to Civil Airspace," *Aviation Week and Space Technology*, February 11, 2007, pp. 49–55.

[2]Valmorbida, G., Wen-Chi, L., and Mora-Camino, F., "A Neural Approach for Fast Simulation of Flight Mechanics," *IEEE Computer Society, 38th Annual Simulation Symposium*, April 2005.

[3]Soloway, D. and Haley, P., "Aircraft Reconfiguration Using Neural Generalized Predictive Control," *American Control Conference*, 2001.

[4]Jategaonkar, R. V., *Flight Vehicle System Identification: A Time Domain Methodology*, AIAA Progress in Astronautics and Aeronautics, Reston, VA, 2006.

[5]Calise, A. J. and Rysdyk, R. T., "Fault Tolerant Flight Control Via Adaptive Neural Network Augmentation," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 98-4483, August, 1998.

[6]Soares, F., "A Flight Test Demonstration of On-Line Neural Network Applications in Advanced Aircraft Flight Control System," *AIAA Infotech@Aerospace Conference 2007 Conference and Exhibit*, AIAA Paper 2007-2942, May 2007.

[7]Amin, S. M., Gerhart, V., and Rodin, E. Y., "System Identification via Artificial Neural Networks:Applications to On-line Aircraft Parameter Estimation," *AIAA and SAE World Aviation Congress*, AIAA Paper 97-5612, October 1997.

[8]Manerowski, J., Zgrzywa, F., and Sibilski, K., "A Neural Model of Coefficients of Forces and Moments of Aerodynamic Forces for a Turboprop Aircraft," *AIAA Atmospheric Flight Mechanics Conference*, AIAA Paper 2006-6281, August 2006.

[9]Isermann, R., *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*, Springer, Berlin, 2006.

[10]Azam, M., Pattipati, K., Allanach, J., Poll, S., and Patterson-Hine, A., "In-flight Fault Detection and Isolation in Aircraft Flight Control Systems," *IEEE Aerospace Conference*, March 2005.

[11]Ortiz, A. and Neogi, N., "A Dynamic Threshold Approach to Fault Detection in Unihabited Aerial Vehicles," *AIAA Guidance, Navigation and Control*, AIAA Paper 2008-7420, August, 2008.

[12]Frank, P. M. and X., D., "A Survey of Robust Residual Generation and Evaluation Methods in Observer-based Fault Detection Systems," *Journal of Process Control*, Vol. 7, No. 6, 1997, pp. 403–424.

[13]Fu, H., Yan, J., Santillo, M. A., Palanthandalam-Madapusi, H. J., and Bernstein, D. S., "Fault Detection for Aircraft Control Surfaces Using Approximate Input Reconstruction," *IEEE American Control Conference*, June 2009.

[14]Oza, N. C., Tumer, K., Tumerand, I. Y., and Huff, E. M., *Lecture Notes in Computer Science*, Springer, 2003, pp. 160–170.

[15]Stevens, B. L. and Lewis, F. L., *Aircraft Control and Simulation*, Wiley-Interscience, New York, NY, 2003.

[16]Klein, V. and Morelli, E. A., *Aircraft System Identification: Theory And Practice*, AIAA Education Series, Reston, VA, 2006.

[17]Roskam, J., *Airplane Flight Dynamics and Automatic Flight Controls Part I and II*, DARcorporation, Lawrence, KS, 1995.

[18]Selig, M. S., "Modeling High Angle of Attack Aerodynamics of Small UAVs in Realtime," *AIAA Atmospheric Flight Mechanics Conference*, to appear, August, 2010.

[19]Hangar 9, "33% Edge 540 ARF," http://www.hangar-9.com/Products/Default.aspx?ProdID=HAN1150, Accessed April 2010.

[20]Demuth, H., Beale, M., and Hagan, M., *Neural Network Toolbox™ 6: Users Guide*, The MathWorks, Inc, Natick, MA, 2009.