AIAA Guidance, Navigation, and Control Conference and Exhibit
21 - 24 August 2006, Keystone, Colorado

AIAA 2006-6551

Aircraft Fault Detection, Isolation and Reconfiguration in the Presence of Measurement Errors

Subhabrata Ganguli, George Papageorgiou and Sonja Glavaški
Honeywell, Minneapolis, MN 55418

This paper discusses the effect of measurement errors on the fault detection, fault isolation and control law reconfiguration algorithms that Honeywell has been researching and developing together with NASA Langley Research Center (LaRC) under NASA's Aviation Safety and Security Program. In our previous papers, we developed fault detection, fault isolation, pilot cueing and control law reconfiguration algorithms for a civil transport aircraft, and evaluated the performance of our algorithms in piloted simulation in the Integration Flight Deck facility at NASA LaRC. However, we did not explicitly evaluate the effect of measurement errors (that is, sensor noise, bias, and dynamics) on the performance of our algorithms. In this paper, we add sensor models to LaRC's simulation model and evaluate the performance of our algorithms in the presence of measurement errors. Each algorithm is analyzed separately, and is enhanced by redesigning and/or re-tuned as necessary. The key contribution is that we provide theoretical justification for the architecture of our fault detection algorithm and discuss a systematic procedure for tuning its gains, and we adapt and re-tune our fault isolation algorithm so that it can cope with measurement errors. We also provide results from batch simulations that are representative of the achieved performance in the presence of imperfect measurements.

I. Introduction

In this paper we describe our research and development work done in fault detection, fault isolation and control reconfiguration under the Active Management of Aircraft Systems Failures (AMASF) contract with NASA Langley Research Center (LaRC). For a brief literature survey on fault detection, isolation and reconfiguration, please refer to our previous work and references therein.[1–4]

Previously, we developed fault detection, fault isolation, pilot cueing and control law reconfiguration algorithms for a civil transport aircraft, and evaluated the performance of our algorithms in batch simulation using a high fidelity mathematical model of LaRC's ARIES (Airborne Research Integrated Experiments System) aircraft, provided to us by LaRC. Our algorithms, referred to as the Control Upset Prevention and Recovery System (CUPRSys) algorithms, were successfully evaluated in piloted simulation at NASA LaRC.[4]

In our previous research work, we did not explicitly evaluate the effect of measurement errors (that is, sensor noise, bias, drift, dynamics and time delays) on the performance of our algorithms. In this paper, we add sensor models to LaRC's B-757 simulation model and evaluate the performance of the CUPRSys algorithms in the presence of measurement errors. Each algorithm is analyzed separately, and is enhanced by redesigning and/or re-tuned as necessary.

Most of our work has focussed on the fault detection and fault isolation algorithms. This is because the dynamic inversion based control law performs well with imperfect measurements and is very robust to parametric uncertainties. Therefore, if the fault isolation algorithm identifies a reasonable model of the faulted aircraft (we shall quantify reasonable later on in this paper), then the reconfigured control law will perform well with imperfect measurements.

The main contributions of this paper are:

1. We provide theoretical justification for the architecture of our fault detection algorithm;
2. We develop a systematic procedure for tuning the gains of the fault detection algorithm – the tuning procedure can handle measurement errors and is based on a support vector machine optimization that has statistical interpretation;
3. We adapt the fault isolation algorithm so that it handles measurement errors; and
4. We use the statistical interpretation of recursive linear least-squares estimation to develop a more systematic tuning procedure for the fault isolation algorithm's gains.

What follows is a brief outline of the paper. In section II, we briefly describe the aircraft model used in this research. Section III provides details on the sensor models used throughout this paper. The control law, the fault detection algorithm and the fault isolation algorithm are discussed in Sections IV, V and VI respectively. Some time histories showing the overall performance of CUPRSys in the presence of measurement errors are given in Section VII followed by conclusions in Section VIII.

II. Aircraft Model

This section presents a brief description of NASA LaRC's ARIES aircraft, which is used for this research. ARIES is a modified version of the Boeing 757-200 used for flight research programs by NASA. It is a mid-size transport aircraft with twin engines and with a maximum takeoff weight of approximately 250,000 lb. The longitudinal-axis control surfaces are a set of 2 (left/right) elevators and a horizontal stabilizer. Lateral/directional control is provided by a set of 2 ailerons (left/right) and a rudder. Further control of the aircraft is provided by flaps (1 inboard and 1 outboard per wing), slats (1 inboard and 4 outboard per wing) and spoilers (5 for flight and 1 for ground per wing). Note that LaRC has a very high fidelity batch simulation of ARIES at the Integration Flight Deck facility which can be used to conduct piloted simulations.

Initial development and evaluation of the CUPRSys algorithms was done with a MATLAB-based nonlinear 6-DOF model of the ARIES aircraft that includes the following components:

1. Rigid body aircraft dynamics (with respect to a rotating or flat earth);
2. An aerodynamics model that captures the effects of the stabilizer, average elevator, left and right ailerons, rudder, average left and right flaps, average left and right slats, average left and right inboard spoilers, average left and right outboard spoilers, and landing gear and ground spoilers;
3. Actuator models;
4. Left and right engine models;
5. A landing gear model; and
6. Atmosphere and Dryden turbulence models.

A number of aircraft failures have been modeled in the MATLAB simulation; these include stuck and floating control surfaces due to hydraulic/mechanical failures, reduced surface effectiveness due to surface loss, engine failures, landing gear failures and icing. However, in this paper we focus on only reduction of surface effectiveness. To simulate damaged surfaces, it is assumed that a change in the available control power can be modeled by reducing the surface commands before they enter the aerodynamic look-up tables (this is different from reducing the surface area of the damaged surface which is probably a more accurate way of modeling surface loss).

Although simulation studies are done at various flight conditions, in this paper we present results only from the Low Cruise (straight leveled flight trimmed at altitude 5000 ft, KCAS 250) flight condition for the sake of brevity.

III. Sensor Models

In this section, we present details about the sensor models used in our batch Matlab simulation in terms of their dynamics, their noise and their bias levels. The architecture presented in Fig. 1 shows the generic structure used to model the sensors. The signal y_m corresponds to an imperfect measurement of the signal y. The dynamics are modeled by a first order filter with the corner frequency represented by w. The variable $\tilde{n}$ represents band-limited white noise with unit intensity and with a randomly varying seed. Note that band-limited white noise produces normally distributed random numbers at a specific sampling rate and is useful to approximate white noise in simulations. The gain at the output of the band-limited white noise block is chosen so that the $\mathcal{H}_2$ norm of $gw/(s+w)$ is equal to σ, that is, $g = \sigma\sqrt{\frac{2}{w}}$. The variable $\tilde{b}$ represents a bias added to the uncorrupted signal y and is modeled by a uniformly distributed random number varying between $\pm\tilde{b}$. Based upon discussion with NASA LaRC, the parameters representing the noise and bias models for the sensor measurements are chosen as shown in Table 1. The sensor dynamics associated with all the measurements are considered as first order lags with corner frequencies of 30 rad/sec.

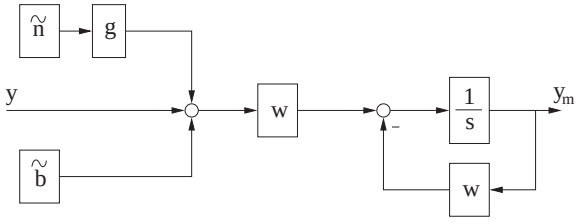

Figure 1. Sensor Model.

Sensor	σ	$\tilde{b}$
h	10 ft/s	10 ft/s
u_b, CAS, TAS	0.5 ft/s	2 ft/s
v_b, w_b	0.25 ft/s	0.5 ft/s
a_x, a_y, a_z	0.08 ft/s^2	0.1 ft/s^2
p, q, r	0.065 deg/s	0.02 deg/s
ϕ, θ, ψ	0.03 deg	0.2 deg/s
$\delta_{a,left}$, $\delta_{a,right}$, δ_e, δ_r, δ_{stab}	0.04 deg	0.2 deg

Table 1. Sensor noise and bias (notation for measured signals are standard in aerospace literature) .

IV. Control Law

CUPRSys implements a dynamic inversion-based control law. The key idea here is to invert the aircraft dynamics and to augment the resulting responses with a set of desired dynamics based on the performance requirements. The advantage of this framework is that the controller is parameterized in terms of the parameters of the on-board aircraft model. Thus, in the event of a failure, reconfiguration essentially involves updating the on-board plant model with its modified parameters. As mentioned before, the controlled variables for the CUPRSys control law are roll rate (p), C^* (a blend of pitch rate and normal acceleration) and angle-of-sideslip (β). Corresponding control surfaces used are aileron difference, average elevator and rudder. Our AIAA GNC 2005 paper[4] provides a detailed description of the control law.

Nonlinear simulation and linear frequency response analysis indicated that we did not need to modify our control law design to cope with measurement errors. The control law has integral action that compensates for sensor bias, and the closed-loop bandwidth and roll-off are such that sensor noise does not cause large control surface activity and ride discomfort. Finally, our control law is robust to phase lags (sensor dynamics).

Fig. 2 shows closed-loop time responses of the aircraft with and without sensor noise and bias. The pilot commands are a sequence of doublets in the three axes: a 3 deg sideslip doublet between 20-35 sec, a 3 deg/sec C^* doublet between 55-63 sec and a 3 deg/sec roll rate doublet between 78-84 sec. It is observed that the responses of the controlled variables (p, C^* and β) are almost identical for both cases confirming the robustness of the control law. With measurement error, the control activities in the three axes are slightly more compared to the control actions without any measurement error, but these are within acceptable limits. An alternative strategy is to low-pass filter the signals to reduced noise, but did not feel it was necessary for our case. The small drift in the elevator response after the C^* doublet corresponds to a drift in altitude from its trim position.

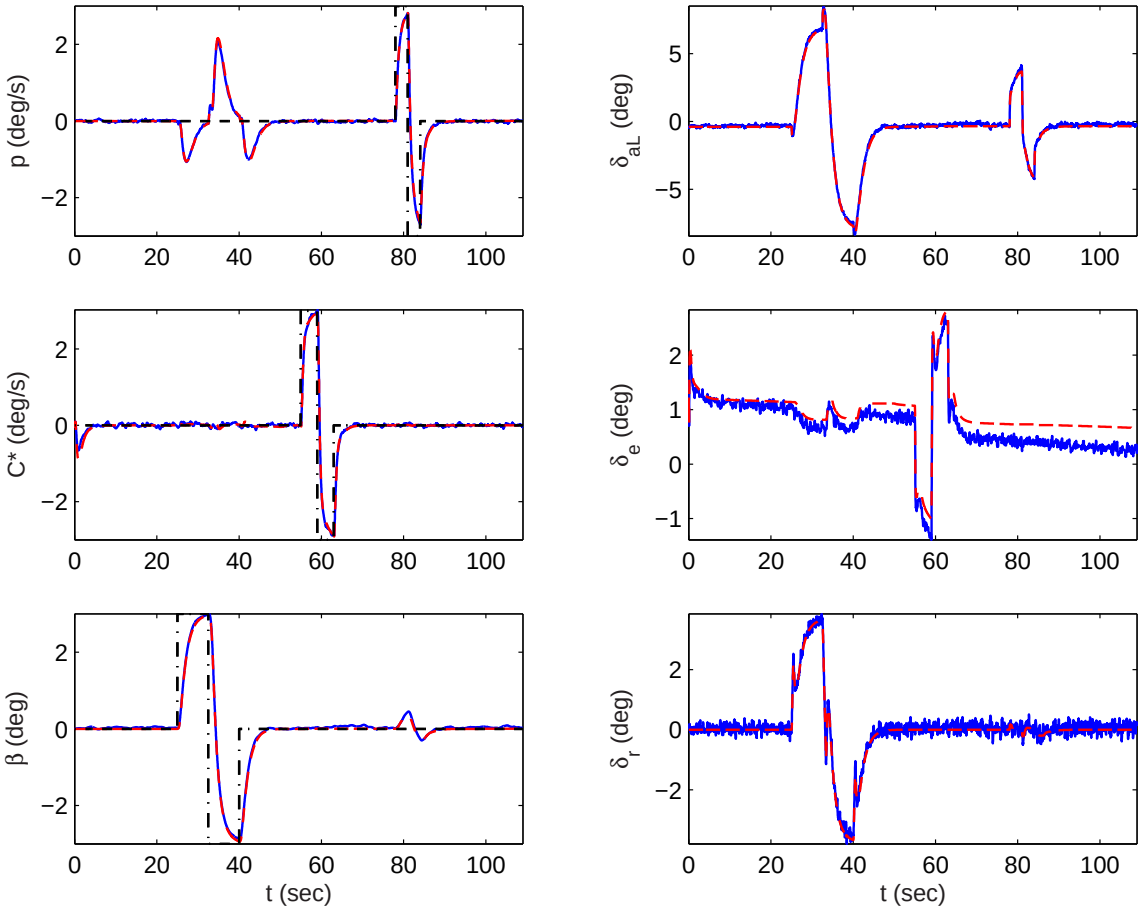

Figure 2. Time responses comparing control law performance with (solid) and without (dashed) measurement error.

V. Fault Detection

Assume that the true aircraft dynamics are given by:

$$\dot{x} = f(x,u), \quad \omega = Cx \quad \Rightarrow \quad \dot{\omega} = Cf(x,u),$$

where $x \in \mathbb{R}^n$ denotes the state vector, $u \in \mathbb{R}^m$ denotes the control input vector and $\omega \in \mathbb{R}^3$ denotes the angular velocity vector (note that C contains only zeroes and ones). Also assume that we have measurements of x and u:

$$x^{\mathrm{m}} = x + \delta x, \quad u^{\mathrm{m}} = u + \delta u,$$

where δx and δu denote the measurement errors. In this work, the two primary sources of measurement error are sensor noise and sensor bias (other sources are sensor drift, sensor dynamics and sensor time delay,

but these are not considered in this study). Our on-board model for the angular acceleration of the aircraft is given by:

$$\begin{aligned}\hat{\dot{\omega}} &= C\hat{f}\left(x^{\mathrm{m}}, u^{\mathrm{m}}\right) \\ &= -J^{-1}\omega^{\mathrm{m}} \times J\omega^{\mathrm{m}} + J^{-1}\left[\tau_{\mathrm{prop}}\left(x^{\mathrm{m}}, u^{\mathrm{m}}\right) + \tau_{\mathrm{aero}}\left(x^{\mathrm{m}}, u^{\mathrm{m}}\right)\right],\end{aligned} \tag{1}$$

where J is an estimate of the inertia matrix, τ_{prop} is an estimate of the moment generated by the propulsion system about the center-of-gravity of the aircraft and τ_{aero} is an estimate of the aerodynamic moment about the center-of-gravity. Also, in this work, our "measurement" of $\dot{\omega}$ comes from differentiating $\omega^{\mathrm{m}} = Cx^{\mathrm{m}}$ (alternatively, $\dot{\omega}^m$ could come from an arrangement of accelerometers as in gyro-free navigation):

$$\begin{aligned}\dot{\omega}^{\mathrm{m}} &= \frac{d(Cx^{\mathrm{m}})}{dt} \\ &= \frac{d(Cx + C\delta x)}{dt} = \dot{\omega} + \delta\dot{\omega}.\end{aligned}$$

Note that $\delta\dot{\omega}$ is only due to sensor noise since differentiating x^{m} removes the sensor bias.

The so-called *residual signal* is a signal that is used to decide whether or not a fault has occurred. For faults that affect the aerodynamic control surfaces (for example, the reduction in the effectiveness of a surface, and a stuck or floating surface), a very sensible choice for the residual signal is:

$$\begin{aligned}e &= \hat{\dot{\omega}} - \dot{\omega}^{\mathrm{m}} \\ &= \hat{\dot{\omega}} - \dot{\omega} - \delta\dot{\omega} \\ &= C\hat{f}(x + \delta x, u + \delta u) - Cf(x, u) - \delta\dot{\omega}.\end{aligned}$$

Let (x_0, u_0) be a trim condition of the aircraft, that is, $f(x_0, u_0) = 0$. By taking the Taylor series expansion of e about (x_0, u_0) and keeping only the first order terms:

$$e \approx \underbrace{C\hat{f}(x_0 + \delta x, u_0 + \delta u) - \delta\dot{\omega}}_{\text{trim + measurement errors}} + \underbrace{C\Delta A\Delta x + C\Delta B\Delta u}_{\text{model mismatch}}, \tag{2}$$

where $\Delta x = x - x_0$, $\Delta u = u - u_0$,

$$\Delta A = \left.\frac{\partial\hat{f}(x + \delta x, u + \delta u)}{\partial x}\right|_{\mathrm{trim}} - \left.\frac{\partial f(x, u)}{\partial x}\right|_{\mathrm{trim}}, \quad \Delta B = \left.\frac{\partial\hat{f}(x + \delta x, u + \delta u)}{\partial u}\right|_{\mathrm{trim}} - \left.\frac{\partial f(x, u)}{\partial u}\right|_{\mathrm{trim}}.$$

In an ideal world with perfect sensors and perfect models, when there are no faults, $e = 0$. In reality though, even when there are no faults present, e is not necessarily equal to 0 (see Eq. 2). Our fault detection strategy is based upon filtering e to remove the trim and measurement errors, and a threshold function that tightly bounds the model mismatch between the true aircraft dynamics and our aircraft model. If the filtered residual signal is greater than the threshold function, then we flag a fault.

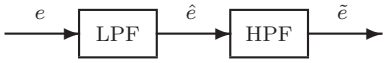

Figure 3. The band-pass filter – LPF stands for low-pass filter and HPF stands for high-pass filter.

The residual signal is filtered using a band-pass filter (see Fig.3). Typically, the low-pass part removes errors due to sensor noise and high-frequency un-modeled aircraft dynamics (for example, aircraft flexure), and the high-pass part removes the trim and sensor bias errors.

We will now derive an upper bound for the size of the model mismatch (between the true aircraft model and the on-board model), and this upper bound will be our threshold function. Note that in order to minimize the number of *false alarms* (that is, flagging a fault when one has not occurred) and the number of *missed detections* (that is, not flagging a fault when one has occurred), our threshold function must tightly bound

the size of the model mismatch. Denote the linear time-invariant transfer matrix from pilot commands (r_i) to filtered residuals ($\tilde{e}$) by $G(s)$ (in this work, the pilot commands are roll rate, C^* and angle-of-sideslip, and all commands are zero during steady wings-level flight). Our design procedure assumes the existence of a control law as we have not yet explored simultaneous design of control and fault detection algorithms. The transfer matrix $G(s)$ is computed using sensor bias but not sensor noise. Any sensor noise error not removed by the band-pass filter will be compensated for after we derive the upper bound.

Denote the impulse response of $G(s)$ by $g(t)$, and let $g_i(t)$ denote the i^{th} row of the impulse response matrix. Then the i^{th} element of $\tilde{e}$ is given by:

$$\tilde{e}_i(t) = \int_0^t g_i(\tau) r(t-\tau) d\tau.$$

Therefore, an upper bound for $|\tilde{e}_i(t)|$ can be derived as follows:

$$\begin{aligned}
|\tilde{e}_i(t)| &= \left| \int_0^t g_i(\tau) r(t-\tau) d\tau \right| \\
&\leq \int_0^t |g_i(\tau) r(t-\tau)| \, d\tau \\
&\leq \int_0^t \sum_{j=1}^{3} |g_{ij}(\tau) r_j(t-\tau)| \, d\tau \qquad \text{(equal to } |\tilde{e}_i(t)| \text{ when } r_j(t-\tau) = \operatorname{sgn}[g_{ij}^*(\tau)],\ \forall \tau \leq t) \qquad (3) \\
&\left(\text{choose } F_{ij}(s) \text{ with } \sum_{j=1}^{3} |g_{ij}(\tau)| \, |r_j(t-\tau)| \leq \sum_{j=1}^{3} f_{ij}(\tau) \, |r_j(t-\tau)| \,,\ \forall \tau \leq t \right) \qquad (4) \\
&\leq \int_0^t \underbrace{f_{ij}(\tau) \, |r_j(t-\tau)|}_{\text{threshold function for } \tilde{e}_i(t)} \, d\tau. \qquad (5)
\end{aligned}$$

It is important to note that Eq. 3 is typically a tight bound for $|\tilde{e}_i(t)|$. Therefore if desired, the designer can choose a filter $F_{ij}(s)$ according to Eq. 4 that tightly bounds $|\tilde{e}_i(t)|$.

Typically, when the control law achieves decoupled command tracking, the off-diagonal terms ($G_{ij}(s), i \neq j$) of $G(s)$ are small as compared to its diagonal terms ($G_{ii}(s)$). This assumption simplifies the implementation of the threshold function, and one can approximate the upper bound with three filters ($F_{ii}(s)$) corresponding to each of the three elements $|\tilde{e}_i(t)|$ of the error vector. In such cases, the simplified threshold function for $\tilde{e}_i(t)$ can be graphically depicted as in Fig.4. Note that this simplification should be viewed as an engineering approximation which works well for our work, but by no means is a limitation of this approach.

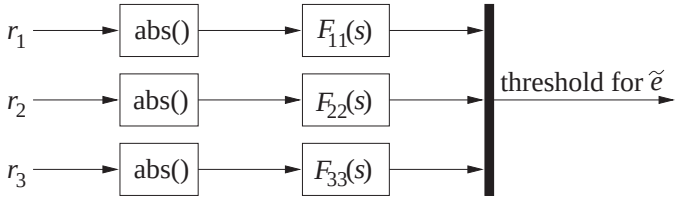

Figure 4. Simplified threshold function for $\tilde{e}_i$ – abs denotes the absolute value operator.

The fault detection algorithm is shown in Fig.5. Note that the threshold function has three filters, one for each axis, and that the output of the fault detection algorithm (`IDflag` in Fig.5) will go to one when a fault is detected (the condition for the presence of a fault is `IDsignal` > `IDthreshold`). The bias has been added to compensate for sensor noise errors, not attenuated sufficiently by the band-pass filter, and any other unmeasured disturbances (for example, atmospheric turbulence).[a]

[a]If we can estimate the effect of atmospheric turbulence on $\tilde{e}$ (for example, using linear acceleration measurements), then we may be able to derive a threshold function for turbulence similar to that derived for r.

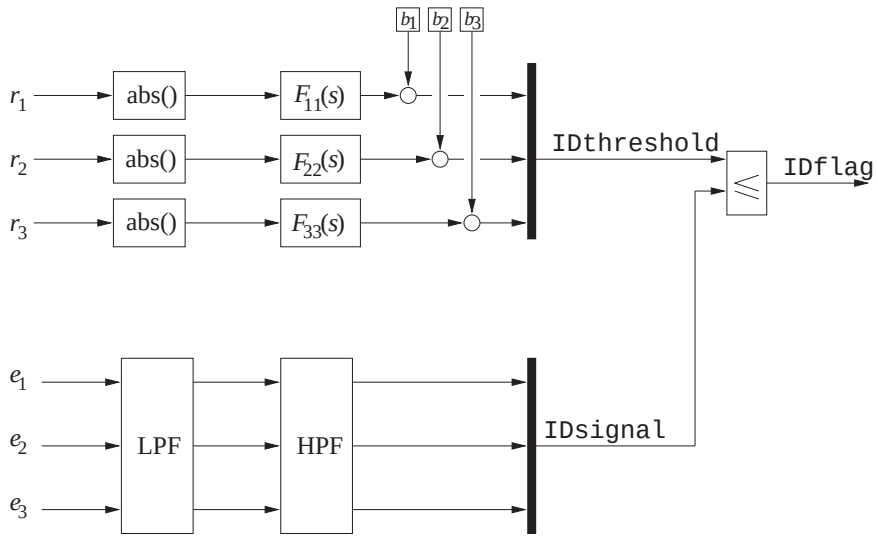

Figure 5. The fault detection algorithm – the output of the threshold function is `IDthreshold` and the output of the band-pass filter is `IDsignal`.

We will now discuss our design procedure for the fault detection algorithm and some specifics of the implemented design. Our design philosophy is to minimize the number of false alarms due to plant/model mismatch at the expense of the number of missed detections. The rationale behind this philosophy is that we only care about detecting faults (or equivalently plant/model mismatch) that require a control law reconfiguration, that is, in order to preserve the desired closed-loop performance we need to reconfigure the control law gains and/or reallocate the control power.

In summary, given the above architecture of the fault detection algorithm, the design consists of appropriately choosing $F_{ii}(s)$ and the bias for `IDthreshold`, and the band-pass filter for `IDsignal`. The first order filter

$$F_{ii}(s) = \frac{k_i}{T_i s + 1}, \text{ with } k_i, T_i \in \mathbb{R},$$

with $T_i = 2$ sec for all three axes was found to work well in practice. This is because it achieves a good trade-off between the robustness to modeling uncertainty and the number of missed detections. The robustness to modeling uncertainty is critical since our aircraft model (see Eq. 1) will be updated on-line and in real-time by the fault isolation algorithm. Therefore, the fault detection algorithm must be robust to any errors in the updated aircraft model.

The selection of the gain k_i, for the filter F_{ii}, and the bias b_i is formulated as a binary classification problem separating faulted data from the unfaulted for each axis. One of the challenges to the design of our fault detection algorithm is the expensive process of data generation (via nonlinear simulation of the aircraft with various types of faults). It is well known that lack of sufficient number of data-points can often lead to overfitting during classification, which in turn, can lead to bad generalization properties of the classifier. Our current pursuit is to use a systematic and structured approach in designing a fault detection classifier which addresses the above issues. The field of Statistical Learning Theory (SLT) provides a promising solution in this respect. The key idea is that the solution optimally separates the data by minimizing a cost function which penalizes both the misclassification error and the tendency of overfitting (by incorporating a measure of the complexity of the classifier function). Moreover, the results also provide a confidence level associated with the quality of classification based on a specified error tolerance level and the number of data-points. The latter provides the user the ability to calculate the minimum number of data-points required to optimally separate a binary data-set with a quantified statistical guarantee.

The next subsection provides a brief discussion highlighting a few key concepts in SLT.[5–7]

A. Statistical Learning Theory

Consider the problem of learning a binary classification problem given a dataset

$$(X, Y) = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\},$$

where the goal is to learn a function $y = f(x)$ that will correctly classify unseen examples[6,7] (by examples it is meant samples). A good measure of the performance of a candidate solution f is the *expected risk*:

$$R[f] = \int L(f(x), y) dP(x, y),$$

where R is the risk functional ranging over $[0,\ 1]$, $P(x, y)$ is the probability density function (pdf), and L is the loss function defined as

$$\begin{aligned} L(f(x), y) &= 0, \quad \text{if } f(x) = y \\ &= 1, \quad \text{if } f(x) \neq y \end{aligned}$$

Unfortunately, the pdf is often not well known, in which case the risk over the training set (*empirical risk*) is used:

$$R_{emp}[f] = \frac{1}{N} \sum_{i=1}^{N} L(f(x_i), y_i).$$

Empirical risk minimization (ERM) produces good results when N is large and the solution is close to that of minimizing the expected risk. However, for small sample size there is no such guarantee and a solution based on a small number of data-points can inadvertently lead to overfitting. This can be avoided if we can control the complexity (or the capacity) of the function f. One such measure of complexity of a classifier function is its Vapnik-Chervonenkis (VC) dimension.[5] The VC dimension of a classifier f measures the largest number of examples which can be modeled by the family f. The novelty of the VC dimension of a function is that it provides bounds on the expected risk as a function of the empirical risk and the number of available examples. In general, it can be shown with a probability $(1 - \eta)$, that the expected risk, $R[f]$, of classification by a function f is upper-bounded by the sum of the empirical risk, $R_{emp}[f]$, and a VC confidence function:

$$R[f] \leq R_{emp}[f] + \left[\frac{h \left(\log \left(\frac{2N}{h} \right) + 1 \right) - \log \left(\frac{\eta}{4} \right)}{N} \right]^{\frac{1}{2}}, \tag{6}$$

where h is the VC-dimension of f, N is the number of examples, and the second term on the right-hand side is the VC confidence function. Thus, to reduce expected risk in classification, a classifier should minimize both the empirical risk and the VC confidence. This is known as Structural Risk Minimization (SRM). Note that as N/h grows larger, the VC confidence term gets smaller and the expected risk becomes closer to the empirical risk. In other words, for a fixed size of the training dataset, the expected risk is reduced by reducing the VC dimension of the classifier. This motivates us to restrain ourselves to classifiers which have low VC-dimensions.

Note that the above-mentioned bound on the expected risk can be used to estimate the sample size which guarantees a desired degree of confidence in the classification results obtained by f. To do this, we first denote ϵ as the tolerance in the error between the estimated and empirical risks. Mathematically,

$$P\left(\sup(R[f] - R_{emp}[f]) < \epsilon\right) > 1 - \eta \tag{7}$$

Thus, any classification based on f is probably (with a confidence of $1-\eta$), approximately (within a tolerance of ϵ) correct. Based on Eqs. 6 and 7, we observe that

$$\epsilon < \left[\frac{h \left(\log \left(\frac{2N}{h} \right) + 1 \right) - \log \left(\frac{\eta}{4} \right)}{N} \right]^{\frac{1}{2}} \tag{8}$$

Thus, given values of η, ϵ and h, one can compute the size of the dataset N using Eq. 8. Typically, small values are used for ϵ and η (of the order of 0.01-0.10).[8] For example, using $\epsilon = 0.1$, $\eta = 0.01$, and $h = 3$ (VC dimension for 2-D hyperplane classifier, discussed later), the lower bound on N is computed as approximately 3200. Recent literature in SLT shows that this can be a conservative result. Using tighter bounds for small values of ϵ and η, N can be reduced by an order of magnitude (factor of 18 in the paper by Vu and Samad, 2005[8]).

An optimal separating hyperplane which maximizes the margin (the distance between the decision surface and the nearest data-point of each class) provides a good choice for linearly separable data from the viewpoint of SRM. This is because, it has been shown by Vapnik[5] that the VC dimension of a separating hyperplane with margin m is bounded as follows:

$$h \leq \min\left(\frac{R^2}{m^2}, n\right) + 1,$$

where n is the dimensionality of the input space, and R is the radius of the smallest hypersphere containing all the input vectors (i.e., samples). Thus, maximizing the margin minimizes the VC dimension, and since the separating hyperplane has zero empirical error (it correctly separates the data), maximizing the margin also minimizes the upper bound on the expected risk. The following briefly presents the optimization setup for separating hyperplanes.[5,6]

Consider the problem of separating the set of training vectors belonging to two separate classes

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\},\ x \in \mathbf{R}^n,\ y \in \{-1, 1\}.$$

with a hyperplane

$$\langle w, x\rangle + b = 0.$$

Without loss of generality it is appropriate to choose a canonical hyperplane where the parameters w and b are constrained by

$$\min_i |\langle w, x\rangle + b| = 1.$$

Thus, a separating hyperplane must satisfy the following constraints

$$y_i[\langle w, x_i\rangle + b]\ \geq 1,\ i \in \{1, 2, \ldots, N\}.$$

The margin is given by

$$\begin{aligned}
\rho(w,b) &= \min_{x_i:y_i=-1} \frac{|\langle w, x_i\rangle + b|}{||w||} + \min_{x_i:y_i=1} \frac{|\langle w, x_i\rangle + b|}{||w||} \\
&= \frac{1}{||w||}\left[\min_{x_i:y_i=-1} |\langle w, x_i\rangle + b| + \min_{x_i:y_i=1} |\langle w, x_i\rangle + b|\right] \\
&= \frac{2}{||w||}.
\end{aligned}$$

Thus, the hyperplane that optimally separates the data minimizes

$$\Phi(w) = \frac{1}{2}||w||^2.$$

Case I: If the data is linearly separable, then it can be formulated as the following optimization problem:

$$\max_{\alpha} \min_{w,b} \Phi(w, b, \alpha)$$

where

$$\Phi(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i \left(y_i[\langle w, x_i\rangle + b] - 1\right),$$

and α_i are Lagrangian multipliers with $\alpha_i \geq 0$. This can be cast into a QP problem using the dual formulation. The optimal separating hyperplane is given by

$$\begin{aligned} w^\star &= \sum_{i=1}^{N} \alpha_i y_i x_i, \\ b^\star &= -\frac{1}{2}\langle w^\star, x^r + x^s \rangle, \end{aligned}$$

where x^r and x^s are any support vectors from each class satisfying,

$$\alpha^r, \alpha^s > 0, \; y^r = -1, \quad y^r = 1.$$

The classifier function is given by

$$f(x) = sgn\left(\langle w^\star, x \rangle + b^\star\right)$$

Note that the hyperplane is defined only by a subset of the data for which the Lagrangian multipliers are nonzero. These points are called *support vectors*. Thus, a Support Vector Machine (SVM) can be used to remove the rest of the data without affecting the solution. Fig.6 shows a 2-D illustration of an optimal separating hyperplane with only a few points lying on the two extreme separating hyperplanes parallel to the optimal one.

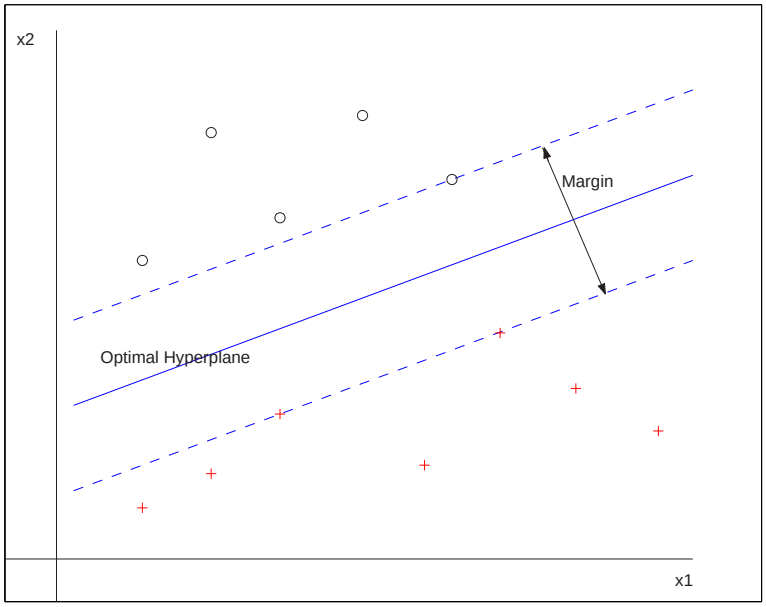

Figure 6. Separating Hyperplane.

Case II: If the data is linearly non-separable, then the constraints are modified by introducing slack variables $\xi_i \geq 0$:

$$y_i[\langle w, x_i \rangle + b] \geq 1 - \xi_i, \; i \in \{1, 2, \ldots, N\}.$$

Thus, the optimization problem can be reformulated as

$$\max_{\alpha,\beta} \min_{w,b,\xi} \Phi(w, b, \alpha, \xi, \beta)$$

where

$$\Phi(w, b, \alpha, \xi, \beta) = \frac{1}{2}||w||^2 + C\sum_{k=1}^{N} \xi_k - \sum_{i=1}^{N} \alpha_i \left(y_i[\langle w, x_i \rangle + b] - 1 + \xi_i\right) - \sum_{j=1}^{N} \beta_j \xi_j,$$

with Lagrangian multipliers $\alpha, \beta \geq 0$. The solution hyperplane is given by same expressions as in Case I, the only difference being, the Lagrangian multipliers are constrained as $0 \leq \alpha_i \leq C$. The cost C is a trade-off

between misclassification and capacity. Large values of C favor solutions with few misclassification errors, whereas small values denote a preference toward margin maximization and low complexity solutions. The parameter C is usually a choice of the user.

An alternate strategy for linearly non-separable data is to project the data into a higher-dimension space, called *feature space*, and to solve for an optimal hyperplane in the feature space. This can lead to higher VC dimension of the classifier (depending upon the choice of the mapping function) and is not relevant to our fault detection problem. For details, see Vanpik[5] and Gunn.[6]

Note that for both linearly separable and non-separable data, the parameters describing the optimal hyperplane classifier function directly correlates with the filter gains and bias values used in the threshold function — the optimal slope $w^\star$ corresponds to the filter gain k_i, and the optimal bias $b^\star$ corresponds to the threshold bias b_i.

B. Results

We will now present details of the optimal separating hyperplane classifiers for our fault detection algorithm. Note that for reduction-in-effectiveness type of faults, typically an elevator fault is observable during longitudinal maneuvers, while aileron and rudder faults can be observable both during roll rate and sideslip maneuvers. Thus, the relevant classification problems are:

P1: Aileron fault detection during sideslip command.

P2: Elevator fault detection during C^* command.

P3: Aileron fault detection during roll rate command.

P4: Rudder fault detection during sideslip command.

P5: Rudder fault detection during roll rate command.

Note that each of the above is a binary classification problem in a two dimensional space (command size $\times$ maximum `IDsignal`) and the corresponding separating hyperplane classifier is defined by two parameters — a slope term and a bias term. We choose three of the above five classification problems (**P2**, **P3**, **P4**) to select the three filter gains k_i (which correspond to the slopes of the classifiers) and the three bias terms in our fault detection setup (Fig. 5). The data is generated by running nonlinear simulations with varying sizes of doublet commands in the roll, pitch and yaw axes for both the unfaulted and the faulted aircraft. The faults simulated are 10%, 25% and 50% reduction in effectiveness of the ailerons, the average elevator and the rudder. Simulations for the same command size and same level of failure are repeated to account for the random variations in sensor noise levels and bias. The residual signal is generated as a difference between the estimated and the measured value of angular acceleration, and is then passed through an *approximate* band-pass filter (described below). The residual signal is first filtered using a first order low-pass filter with a corner frequency of 0.5 rad/sec to remove sensor noise. A first order high-pass filter with a corner frequency of 10 rad/sec is used to remove sensor bias and trim errors in the low-passed residual signal to generate `IDsignal`. Note that the high-pass filter is active only during zero pilot commands and the current estimation of the bias (difference between the input and output of the high-pass filter) is stored in memory. During non-zero pilot commands, the last stored value of the bias is subtracted off the residual to construct `IDsignal`. This is done to avoid unnecessarily high-passing excitations during pilot maneuvers since our objective is to remove only sensor bias and trim errors from the residual signal. This explains the usage of the work *approximate* in the text above.

The optimal hyperplane classifiers for the Low Cruise flight condition are shown in Figs. 7, 8, and 9. The data-points for the unfaulted aircraft are marked by 'x', while those corresponding to 25% and 50% surface effectiveness reduction are plotted by 'o' and '$\triangle$' respectively. Note that the data-points corresponding to the 10% failure level are removed before computing the optimal hyperplane classifiers since those are not linearly separable from the unfailed data-points. This is acceptable because, as mentioned earlier, our fault

	P2		P3		P4	
	SVM	Adjusted	SVM	Adjusted	SVM	Adjusted
Filter gain, k	0.011	0.011	0.005	0.005	0.019	0.020
Bias, $\hat{b}$	0.010	0.010	0.009	0.010	0.017	0.010
No. of false alarms	0	0	0	0	0	0
No. of missed detections	0	0	1	1	9	10
No. of data-points, N	156	156	156	156	156	156

Table 2. Threshold function parameters for fault detection at Low Cruise.

detection philosophy is to minimize false alarms at the expense of missed detections of low levels of fault. It can be argued that the robustness of the controller is sufficient to accommodate a small reduction in surface effectiveness. Observe that 25% and 50% reduction in surface effectiveness can be detected in all the three axes with a high degree of confidence using the linear hyperplane classifiers. The parameters defining the hyperplane classifiers for the three problems **P2**, **P3** and **P4** are presented in Table 2. The columns marked 'Adjusted' refer to minor changes made to the slopes and biases of optimal classifiers to arrive at a common bias term for all the three problems. This is done to make a simplification in the implementation of the fault detection architecture. Instead of comparing 3×1 sized residual and threshold vectors, the signals `IDsignal` and `IDthreshold` are transformed to scalar signals using the 1-norm. This requires three filter gains and a single bias term to construct `IDthreshold` as a function of the pilot commands and is found to work well in practice. For the rudder fault detection case, 25% and 50% level faults are linearly separable from the unfaulted data for the training dataset considered here. For the elevator and aileron fault detection cases, there are a few cases of missed detections for small command sizes. This is acceptable since we know that a reduction in surface effectiveness is observable only if the aircraft is sufficiently excited.

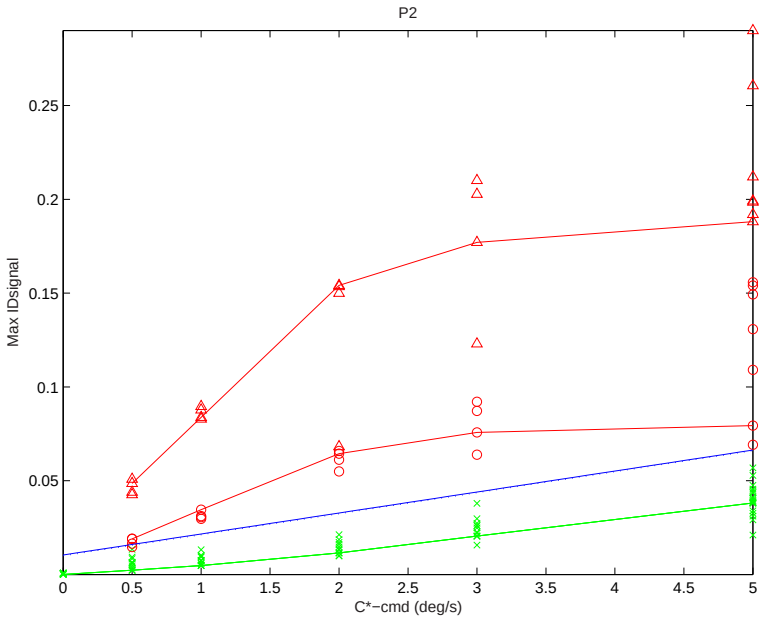

Figure 7. Hyperplane classifiers are shown for elevator fault detection for Low Cruise — optimal (solid), adjusted for common scalar bias (dashed). Un-faulted data-points are marked by 'x', 25% elevator fault data-points by 'o', and 50% elevator fault data-points by '△'. Data-points joined by solid lines represent data without sensor noise/bias.

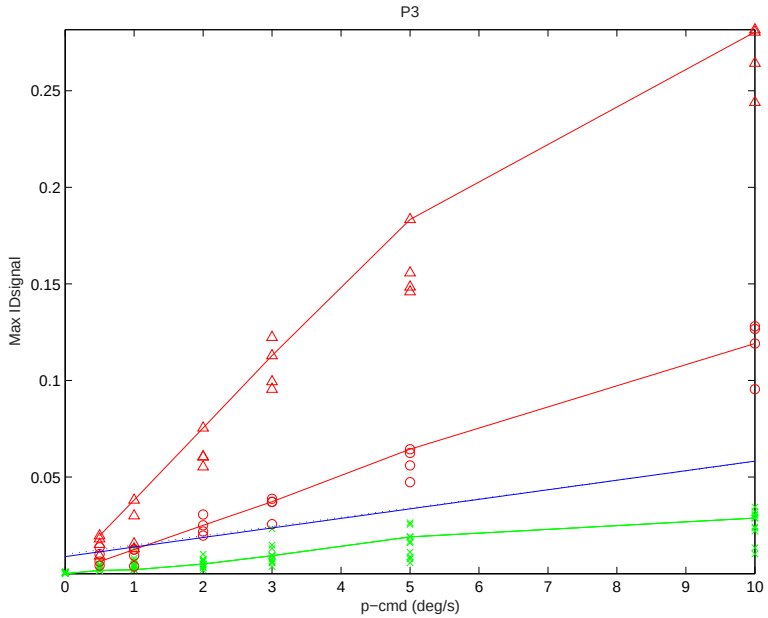

Figure 8. Hyperplane classifiers are shown for aileron fault detection for Low Cruise — optimal (solid), adjusted for common scalar bias (dashed). Un-faulted data-points are marked by 'x', 25% aileron fault data-points by 'o', and 50% aileron fault data-points by '△'. Data-points joined by solid lines represent data without sensor noise/bias.

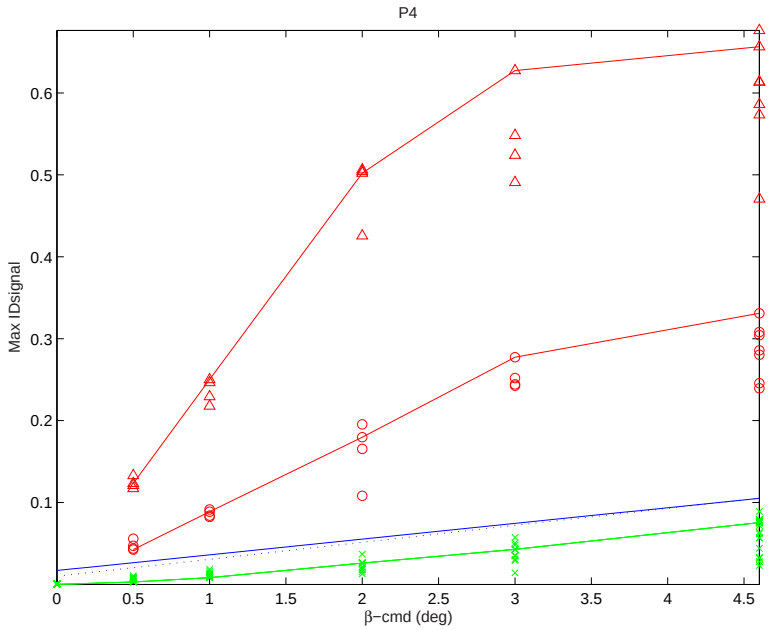

Figure 9. Hyperplane classifiers are shown for rudder fault detection for Low Cruise — optimal (solid), adjusted for common scalar bias (dashed). Un-faulted data-points are marked by 'x', 25% rudder fault data-points by 'o', and 50% rudder fault data-points by '△'. Data-points joined by solid lines represent data without sensor noise/bias.

VI. Fault Isolation

When a fault is detected, it is the job of the fault isolation algorithm to identify the fault (or faults). There are a number of different estimators that could be used for fault isolation. We chose a linear least-squares estimator because this type of estimator can be implemented recursively enabling on-line and real-time identification of faults. Also, linear least-squares estimation is theoretically mature and rich, and there have been a number of successful applications of recursive least-squares estimation reported in the literature.[9,10]

We will now state the equations of the recursive linear least-squares estimator. Assume that the aircraft dynamics are given by:

$$y(t) = \phi^T(t)\theta^0 + e(t), \tag{9}$$

where θ^0 are the "true" parameters and $\{e(t), t = 1, 2, \dots\}$ is a sequence of independent, equally distributed random variables with zero mean. It is also assumed that e is independent of ϕ. As will be discussed in the sequel, the parameters in this application are non-dimensional aerodynamic derivatives. Define

$$\Phi^T(t) = \begin{bmatrix} \phi(1) & \phi(2) & \dots & \phi(t) \end{bmatrix}.$$

From Åström and Wittenmark[9] (Theorem 2.3, p. 51), we have the following theorem:

Theorem 1 (Recursive least-squares estimation) *Assume that the matrix $\Phi(t)$ has full rank for all $t \geq t_0$. Given $\hat{\theta}(t_0)$ and $P(t_0) = [\Phi^T(t_0)\Phi(t_0)]^{-1}$, the least-squares estimate $\hat{\theta}(t)$ then satisfies the recursive equations:*

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + K(t)\left[y(t) - \phi^T(t)\hat{\theta}(t-1)\right], \\ K(t) &= P(t-1)\phi(t)\left[I + \phi^T(t)P(t-1)\phi(t)\right]^{-1}, \\ P(t) &= \left[I - K(t)\phi^T(t)\right]P(t-1). \end{aligned} \tag{10}$$

At every time sample, the mean of the least-squares estimate is $\hat{\theta}(t)$ and the covariance is $P(t)$. The rank supposition in the theorem is related to the excitation of Eq. 9, that is, the excitation of the dynamics that we seek to identify. It is insightful to note that the recursive linear least-squares estimator can be interpreted as a Kalman filter for the plant:

$$\theta(t+1) = \theta(t), \quad y(t) = \phi^T(t)\theta(t) + e(t), \quad \text{cov}[e(t)] = I,$$

where cov denotes the covariance of a random variable.

To be able to apply linear least-squares estimation, the parameters that we wish to identify must appear linearly in our aircraft model (see Eq. 1). Since in this work we only identify the effectiveness of the ailerons, the elevators and the rudder, we expand τ_{aero} as follows:

$$\tau_{\text{aero}} = \tau_{\text{aero},0}(x^{\text{m}}, u^{\text{m}}) + \underbrace{\bar{q}\, S\, \text{diag}\,(b, \bar{c}, b)\, H \begin{bmatrix} \delta_{\text{a}}^{\text{m}} & \delta_{\text{e}}^{\text{m}} & \delta_{\text{r}}^{\text{m}} \end{bmatrix}^T}_{\text{moment due to ailerons, elevators and rudder}},$$

where $\tau_{\text{aero},0}$ is an estimate of the aerodynamic moment about the center-of-gravity of the aircraft minus the contribution of the ailerons, the elevators and the rudder, $\bar{q}$, S, $\bar{c}$ and b are estimates of the dynamic pressure, the wing area, the mean aerodynamic chord and the wing span respectively, $\delta_{\text{a}}^{\text{m}}$, $\delta_{\text{e}}^{\text{m}}$ and $\delta_{\text{r}}^{\text{m}}$ are measurements of the displacement of the ailerons (difference), the elevators (average) and the rudder respectively, and H is the (constant) matrix of non-dimensional aerodynamic derivatives (for example, its $(3,1)$ element is $C_{N_{\delta_{\text{a}}}}$, that is, the non-dimensional yawing moment due to the ailerons). It is now straightforward to put our aircraft model in the form of Eq. 9 and apply linear least-squares estimation.

We will now discuss some implementation issues (see Section 11.5, p. 465–480 in Åström and Wittenmark[9] for an insightful discussion on estimator implementation). Much like in Section V, to remove trim and sensor

bias errors, and to attenuate sensor noise errors, $y(t)$ and $\phi(t)$ are filtered using identical band-pass filters (see Fig.3 for the block diagram of a band-pass filter). The recursive estimator then operates on the filtered signals.

To enable reliable estimation of the aerodynamic derivatives, every time a fault is detected, the estimator is reset and the aircraft is excited in all three axes using automated doublet injection. [b] The mean of the least-squares estimate is reset to its nominal value and the covariance is reset to a suitably large matrix. Statistically, a large covariance implies a poor estimate, and therefore, new data will significantly affect the mean of the estimate ensuring fast convergence of the estimator assuming there is sufficient excitation. Care must be taken not to choose the covariance too large because then the mean will oscillate.

Finally, a very important point that is often overlooked. The statistics of the estimator are only valid if the residual

$$e(t) = y(t) - \phi^T(t)\hat{\theta}(t)$$

is white. If the residual is not white, then it may be that the band-pass filter is not removing the measurement errors, or that our model is not rich enough to capture the true aircraft dynamics. Either way, the designer must make modifications that reduce the size of the residual.

We will now discuss the choice of band-pass filter and covariance matrix for our recursive least-squares (RLS) estimator. The low-pass part of the band-pass filter is selected as a first order filter with a corner frequency of 1 rad/sec. This choice is dictated by the trade-off between the need of sensor noise rejection and the need of sufficient excitation for fault isolation. The high-pass part is selected as a first order washout filter with a corner frequency of 0.5 rad/sec to remove sensor bias and low-frequency modeling error. As mentioned earlier, the choice of the initial/reset value of the covariance matrix (P_0) is crucial for the fault isolation algorithm to work satisfactorily. We select a diagonal structure for P_0 to reflect the fact that the reduction in surface effectiveness of the ailerons, the average elevator and the rudder would primarily correspond to changes in the diagonal elements of H. The diagonal elements for P_0 are selected in an *ad hoc* manner by running nonlinear simulations with faults in each axis (one at a time) and observing the bandwidth of the time responses of the corresponding diagonal elements of the estimated coefficient matrix H. Our criterion is to approximately identify the fault within 4 sec after the end of test signal injection (simultaneous doublet injection of 4 sec duration in each axis).

Fig. 10 shows the time responses of the diagonal elements of the H matrix for a 50% reduction in elevator effectiveness at the Low Cruise flight condition. The corresponding choice of P_0 is diag$[10^{-5}\ 10^{-5}\ 10^{-6}]$. The pilot command consists of three successive doublets: a 3 deg sideslip doublet between 20-35 sec, a 3 deg/sec C^* doublet between 55-63 sec and a 3 deg/sec roll rate doublet between 78-84 sec. The fault is detected at 56 sec and test signals are injected (simultaneous doublets of 4 sec duration in all three axes: 1 deg/sec roll rate, 1 deg/sec C^* and 2 deg sideslip). The covariance matrix $P(t)$ and the estimated H matrix are reset when the fault is detected. The estimated value of $H(2,2)$ after 8 sec (4 sec for test signal injection plus 4 sec delay for approximate convergence) is -24.5 which reflects a 56% reduction from its initial value of -55.7. The variations in $H(1,1)$ and $H(3,3)$ reflect changes in the effectiveness of the ailerons and the rudder. These variations are small compared to their nominal value and are considered acceptable for our work. The accuracy of the indentified elevator fault does improve with time (the overshoot decays) and the final estimated value of $H(2,2)$ is -27.4 reflecting a 49% effectiveness reduction. This is because of the presence of further excitation in terms of pilot commands in this example. Although it is possible to increase the response time of estimation by increasing the covariance matrix, a higher value of $P_0(2,2)$ leads to an oscillatory response for which the fixed convergence criterion of 8 sec delay after fault detection becomes inaccurate. On the other hand, the overshoot in the time response of $H(2,2)$ can be removed by decreasing $P_0(2,2)$, but that leads to a slow time response of the estimator which is unacceptable from a practical point of view. The two other diagonal elements of P_0 are chosen based on similar arguments by simulating faults in the ailerons and rudder respectively.

[b] Alternatively, instead of resetting the estimator, we could add a forgetting factor to the least-squares problem formulation. The forgetting factor smoothly phases out past data, see Åström and Wittenmark[9] (Theorem 2.4, p. 53). We tried this but found that for abrupt changes in the aerodynamic derivatives resetting the estimator performed better.

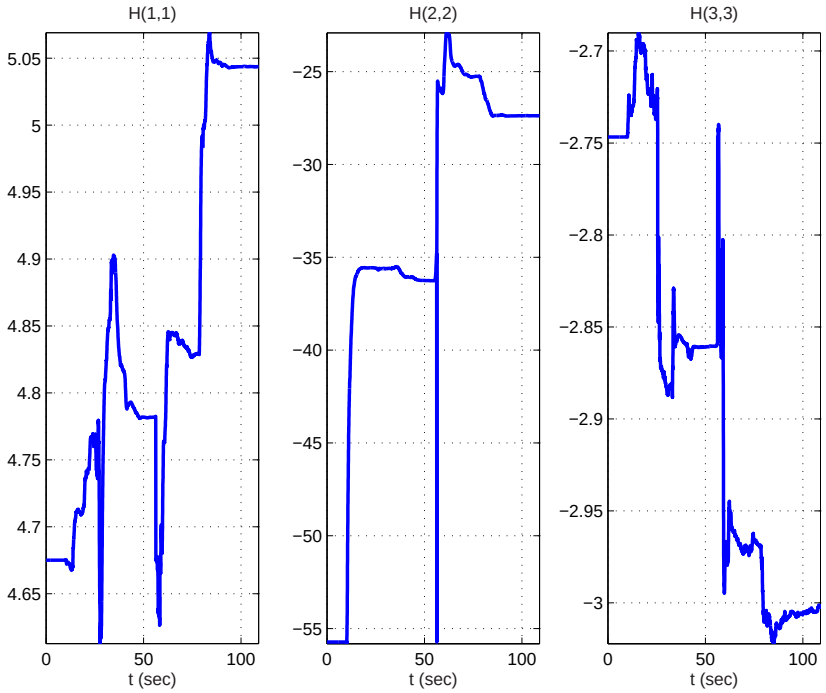

Figure 10. Fault isolation performance shown by time responses of the diagonal elements of the H matrix for a 50% reduction in elevator effectiveness at the Low Cruise flight condition.

Results

We analyze the performance of the RLS estimator by collecting statistics on the accuracy of estimation. Nonlinear simulations are run with varying sizes of commands (2-10 deg/s roll rate, 2-5 deg/s C^* and 2-4.6 deg sideslip) with 25% and 50% reduction in surface effectiveness in each axis, and runs for the same command size and fault levels are repeated to account for variations in sensor noise and bias levels. Smaller size commands than those mentioned above are not considered since fault detection is typically inefficient for small aircraft excitations. The fault identification data for Low Cruise are shown by box-and-whisker plots in Figs. 11, 12 and 13 using the MATLAB Statistical Toolbox command `boxplot`. The box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. The length of the whiskers are correlated with the interquartile range (IQR), which is defined as the difference between the upper quartile and lower quartile. The whiskers extend from each end of the box to the most extreme data value within 1.5×IQR of the box. If there are no data outside the whisker, then, there is a dot at the bottom whisker. If a data falls inside the box, we choose not to draw it. Outliers are data points with values beyond the ends of the whiskers. The novelty of the box-and-whisker plot is that the data is highlighted by its IRQ (middle 50% of the values) which is less influenced by extreme values. Each of the subplots in Figs. 11, 12 and 13 represent data from 28 different simulations.

The performance of the fault identification algorithm looks satisfactory based upon Figs. 11, 12 and 13. The median and the IQR matches with the simulated fault level to a reasonable degree of accuracy. In Fig. 11, for both 25% and 50% aileron faults, the identified faults are slightly lower than their equivalent simulated level, and a small level of failure is also represented in each of the other two axes. However, fault identification data for 25% and 50% aileron faults in *absence* of sensor noise and bias also show the same feature and correspond with the mean and IQR values of Fig. 11. The fact that the (minor) inaccuracies in the isolated fault levels both with and without measurement errors are similar, indicates possibility of a (minor) modeling error [c]. Similar analysis are done for the elevator and rudder fault isolation performances, and the data for

[c]The modeling error arises because the on-board aircraft model is an approximation of the true aircraft.

elevator and rudder faults (Figs. 12 and 13) are compared with corresponding data without measurement error. One such comparison is included here as an example, where we plot the time responses of the diagonal elements of the H matrix in Fig. 14 with and without sensor noise and bias for a 25% reduction in elevator effectiveness. In both simulations, the fault is detected at approximately 57 sec. The estimated values of the elevator effectiveness at 8 sec after fault detection represent a 39% reduction without measurement error and 37% with measurement error. Even the final estimated values (36% decrease without any measurement error and 35% decrease with sensor noise and bias) are higher than the simulated 25% level of failure. This indicates that the inaccuracies in the identified fault are essentially an artifact of modeling error; however, the fault identification algorithm is able to successfully reject the random variations in sensor noise and bias.

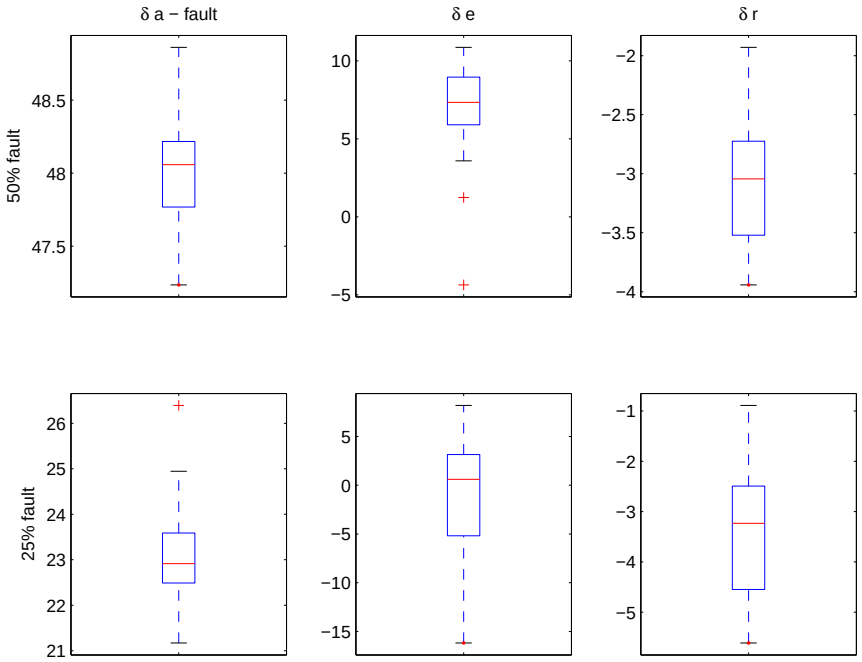

Figure 11. Box-and-whisker plot showing fault isolation performance for aileron fault at the Low Cruise flight condition. The y-axis in each subplot represent % decrease (increase if negative) in surface effectiveness.

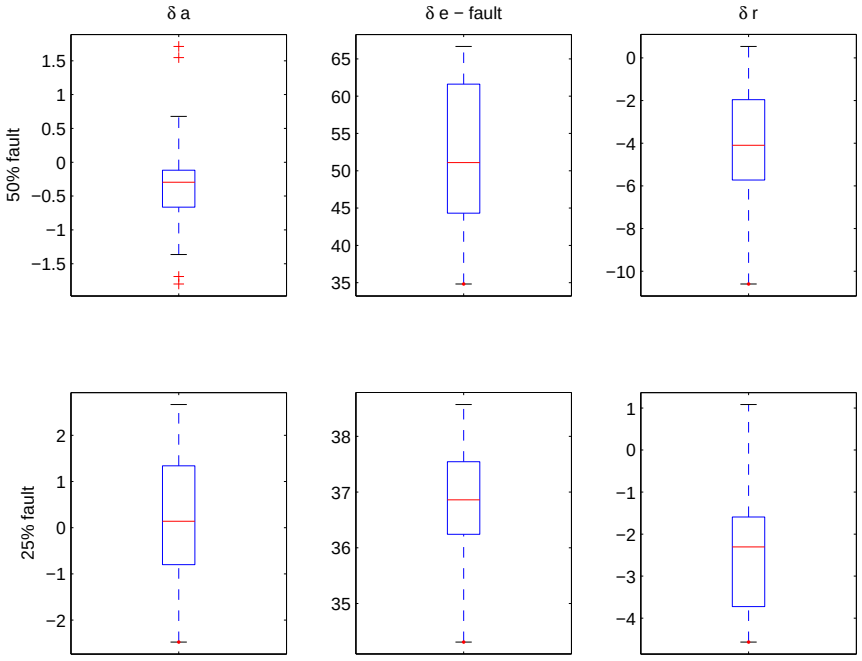

Figure 12. Box-and-whisker plot showing fault isolation performance for elevator fault at the Low Cruise flight condition. The y-axis in each subplot represent % decrease (increase if negative) in surface effectiveness.

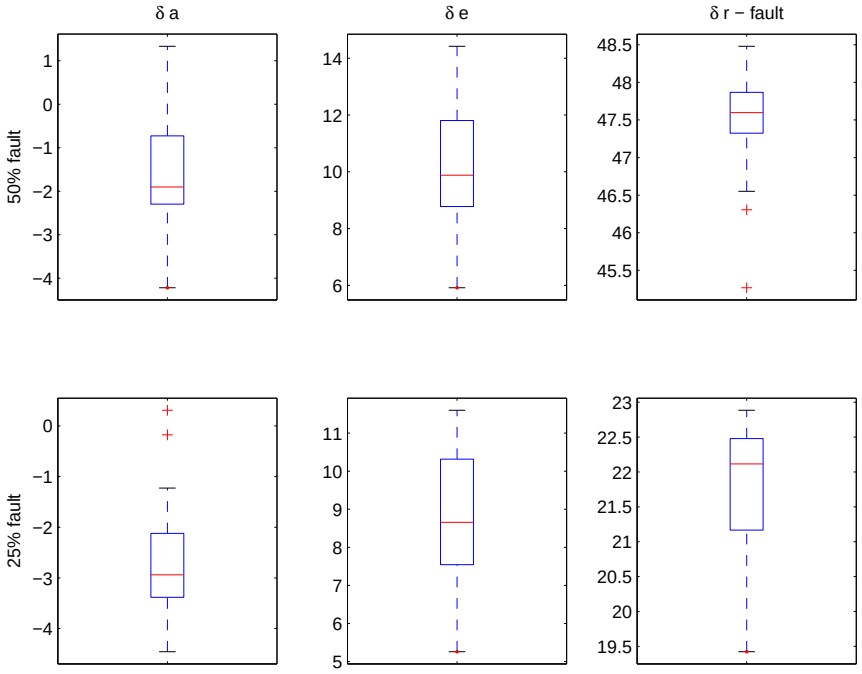

Figure 13. Box-and-whisker plot showing fault isolation performance for rudder fault at the Low Cruise flight condition. The y-axis in each subplot represent % decrease (increase if negative) in surface effectiveness.

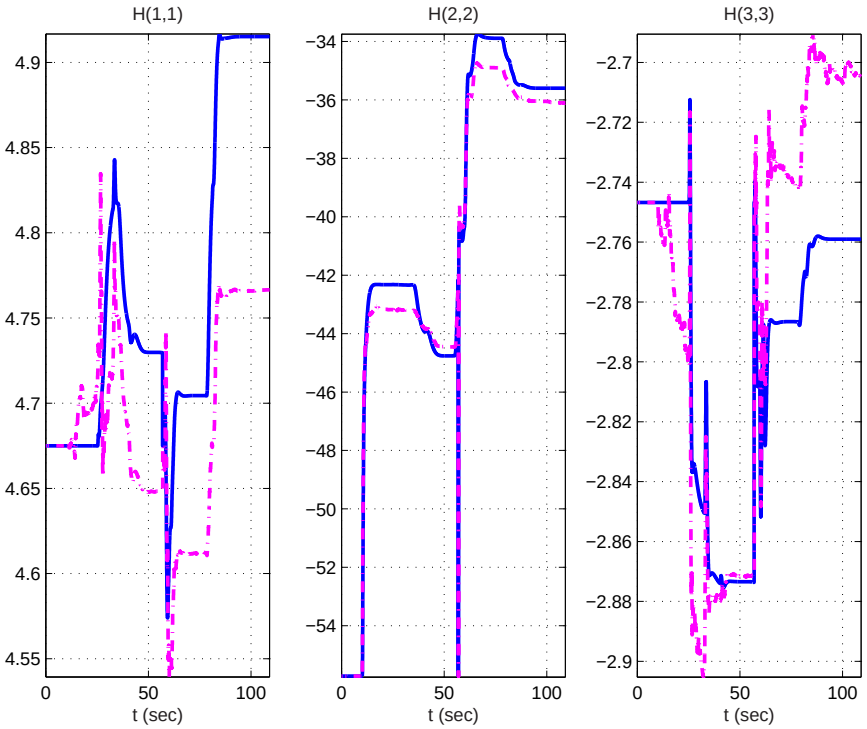

Figure 14. Fault isolation performance shown by time responses of the diagonal elements of the H matrix for a 25% reduction in elevator effectiveness at the Low Cruise flight condition. Responses with solid lines are with sensor noise and bias turned on, while those with dash-dotted lines are without any sensor noise or bias.

VII. Control Reconfiguration

This section presents the performance of the CUPRSys reconfigurable control law with sensor noise and bias turned on and, in turn, also illustrates the overall performance of the fault detection, isolation and reconfiguration algorithms put together in the presence of measurement error. In Fig. 15, we overlap plots from three different simulations. The first simulation represents aircraft responses for a C^* doublet without any fault (solid lines). The second simulation shows aircraft responses for a 50% reduction in elevator effectiveness (dash-dotted lines). Finally, the third simulation illustrates aircraft responses with the fault detection, isolation and reconfiguration algorithms turned on (dashed lines). The dotted lines represent a 3 deg/s doublet commanded by the pilot with a duration of 12 sec. The overlayed 4 sec C^* doublet command of 1 deg/s represents an automated test signal injected once the fault is detected at 15.2 sec. The aircraft responses with the faulty elevator is noticeably sluggish as compared with those for the healthy aircraft. Based on our fault isolation convergence criterion, the fault is isolated 8 sec after fault detection at 23.2 sec as a 53% reduction in elevator effectiveness and the control law is reconfigured immediately based on the estimated level of failure. We observe that the C^* response deviates from the slower response with the faulted aircraft right after reconfiguration and matches with the C^* response for the unfaulted aircraft. The elevator responses show larger activity after reconfiguration which is required to recover from the failure and achieve the same C^* response as that of the healthy aircraft.

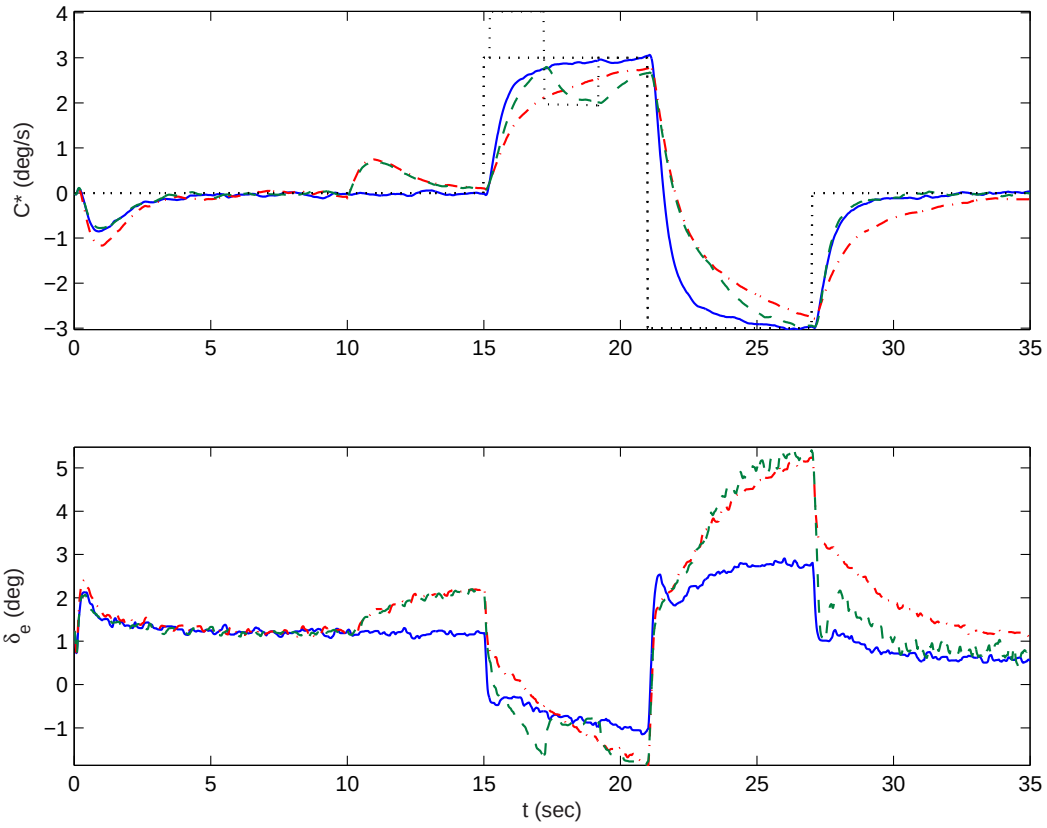

Figure 15. Time responses illustrating overall CUPRSys performance. Solid line shows aircraft responses for a C^* doublet with no fault. Dash-dotted lines denote responses for 50% elevator effectiveness reduction. Dashed lines shows responses with the reconfigured controller (with fault detection at 15.2 sec, isolation and reconfiguration at 23.2 sec). Dotted lines represent pilot command and a superimposed test signal for fault isolation.

VIII. Conclusions

This paper provides evidence that CUPRSys can perform well in the presence of measurement errors. Like in previous phases of the AMASF project, the control law did not require much modification and most of our effort focussed on the fault detection and fault isolation algorithms.

The performance of our fault detection algorithm is quite satisfactory in terms of number of false alarms and missed detections. This can be attributed to its architecture and the systematic tuning process that automates and shortens the gain selection process.

Overall, the fault isolation algorithm did not perform much worse in the presence of imperfect measurements. There are minor errors in the identified aircraft model in a few cases but analysis has shown that our control law is robust to the type and size of the errors encountered.

To date, we have demonstrated that CUPRSys can achieve good performance in the presence of modeling and measurement errors. However, we still need to evaluate its performance in the presence of atmospheric turbulence. Also, we need to revisit the structure of the on-board aircraft model. Currently, we consider each flight condition separately and have not attempted to schedule between flight conditions.

In conclusion, we believe that analyzing the effect of atmospheric turbulence on CUPRSys together with considering an expanded set of failures and flight conditions will help mature CUPRSys and ultimately provide a reliable fault detection, isolation and reconfiguration system for the next generation civil aircraft.

Acknowledgements

The authors would like to thank Patrick Murphy, Sungwan Kim, Stephen Derry, Gustav Taylor, Robert Rivers, Tom Bundick, Christine Belcastro, and Celeste Belcastro for many helpful discussions.

References

[1]Elgersma, M. and Glavaški, S., "Reconfigurable Control for Active Management of Aircraft System Failures," *Proceedings of the American Control Conference*, Arlington, VA, Jun 2001.

[2]Elgersma, M., Glavaški, S., Dorneich, M., and Lommel, P., "Failure Accomodating Aircraft Control," *Proceedings of the American Control Conference*, Anchorage, AK, May 2002.

[3]Glavaški, S., Elgersma, M., Dorneich, M., and Lommel, P., "Active Failure Management for Aircraft Control Recovery," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Monterey, CA, Aug 2002.

[4]Ganguli, S., Papageorgiou, G., and Glavaški, S., "Piloted Simulation of Fault Detection, Isolation and Reconfiguration Algorithms for a Civil Transport Aircraft," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, San Francisco, CA, Aug 2005.

[5]Vapnik, V., *Statistical Learning Theory*, Wiley, 1998.

[6]Gunn, S., *MATLAB SVM Toolbox*, University of Southampton, UK, 1998.

[7]Gutierrez-Osuna, R., "Lecture Notes on Pattern Recongnition and Intelligent Sensor Machines (PRISM)," Texas A & M University, `http://research.cs.tamu.edu/prism/lectures.htm`.

[8]Ha, V. and Samad, T., "Generalization Bounds for Weighted Binary Classification with Applications to Statistical Verification," *International Joint Conferences on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[9]Åström, K. J. and Wittenmark, B., *Adaptive Control*, Addison Wesley, 2nd ed., 1995.

[10]Ljung, L. and Söderström, T., *Theory and Practice of Recursive Identification*, Series in Signal Processing, Optimization, and Control, MIT Press, 1983.