Detection and Isolation of Propeller Icing and Electric Propulsion System Faults in Fixed-Wing UAVs

O. M. Haaland, A. W. Wenz, K. Gryte, R. Hann, T. A. Johansen

Center for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics,
Norwegian University of Science and Technology, Trondheim, Norway

***Abstract*—Fault in the propulsion system of UAVs is one of the main causes for incidents and loss of the aircraft. In electric propulsion systems typical faults are ball-bearing degradation leading to increased friction and losses, as well as propeller icing when operating in cold conditions. In this paper we propose a fault detection and isolation (FDI) framework that uses model-based estimators of the various faults, implemented with multiple Kalman and Bayes filters. The method is tested and shown to be effective in a simulation setup using a fixed-wing UAV simulator.**

I. INTRODUCTION

There are many key challenges and risks related to the increased utilization of unmanned aerial vehicles (UAVs) and their integration into non-segregated airspace. It is widely recognized that faults in the propulsion system is one of the main risks that may lead to loss of aircraft and damage to third parties. One particular risk is related to UAV propeller icing, which is a major concern as it is known that a propeller may loose 75% of its thrust after less than 2 minutes of operation in severe icing conditions [1], [2], [3]. Notably, the effects of icing are typically large on small UAVs because of slow airspeed and small airframe sizes [4]. Many missions that require beyond visual line of sight (BVLOS) operation are at risk of encountering icing conditions if the pilot cannot ensure that cloud environments are avoided [5], [24].

Electric propulsion system faults typically emerge gradually in the form of reduced efficiency or increased friction, e.g., due to ball-bearing wear. While detection and diagnosis of several types of propulsion system faults in UAVs is a relatively well studied topic, e.g., [8], [9], [10], [11], [12], [19], there is much less research on detection of propeller icing on UAVs.

The problem addressed here is how to automatically detect icing on the propeller on a fixed-wing UAV, and how to isolate this condition given that there could be several types of faults in the electric propulsion system. The main idea of this paper is to employ models of the propulsion system dynamics [17] to formulate models corresponding to degraded performance and failure modes. An algorithm for propeller icing detection, fault detection and isolation is built by combining multiple model Kalman filters in a Bayesian framework with a Markov model for fault hypotheses. The method uses measurements of the UAV's airspeed, and the propulsion motor's angular speed and electric current. It is assumed that the measurements are fault free, yet noisy, which is motivated by the reliable nature of sensors for angular speed and electric current, as well as the existence of methods that can detect and estimate faults on air speed sensors [19], [20], [21], [22]. This approach has some similarities to methods that have been used for detection of airfoil icing, which has been studied during the last years using various methods such as model-based estimation [16], multiple-model estimation [14], [13], and statistical fault diagnosis methods [15].

The main contribution and novelty in this paper is the use of aerodynamic propulsion performance models and estimation for propeller icing within an FDI framework. In order to enhance the ability to isolate the fault, we also employ a small change in airspeed.

The paper is organized as follows: Section II introduces the basic model of the propulsion system, that is used to formulate estimators, detection algorithms and diagnosis logic in Section III. The proposed method is tested using simulation of a fixed-wing UAV. The simulation setup is described in Section IV and results are given in Section V, before the conclusions in Section VI.

II. MODELING

This section introduces the electric propulsion model for a fixed-wing UAV. Then, a state space model is used to formulate different hypotheses about the propulsion faults and propeller icing. The section ends with a parameter estimation model that is formulated with a linear time-variant (LTV) state-space representation.

A. Propulsion Model

The propulsion model of the UAV is based on modelling of the electrical, mechanical and aerodynamic subsystems. We only state the resulting equations here, and refer to [17] for a more comprehensive description of the model:

$$\Theta\dot{\omega} = k_e(I_e - I_0) - c_v\omega - Q_a \tag{1}$$

where in the nominal case we have

$$Q_a = \rho\frac{\omega^2}{4\pi^2}D^5 C_Q(J) \tag{2}$$

$$C_Q(J) = C_{Q,0} + C_{Q,1}J + C_{Q,2}J^2 \tag{3}$$

$$J = 2\pi\frac{V_a}{D\omega} \tag{4}$$

Equation (1) gives the propeller torque balance, where ω is the angular speed, Θ is the moment of inertia of the

shaft and rotor including the propeller, I_e is the motor electric current, I_0 is the zero-load current, c_v is the viscous friction coefficient, and k_e is the motor constant. Equation (2) describes Q_a which is the aerodynamic torque created by propeller drag, where D is the propeller diameter and ρ is the air density. The thrust coefficient, $C_Q(J)$ is given by a second order polynomial. This polynomial is a function of the advance ratio J, where V_a is the airspeed.

Our investigation with different fault detection and identification methods has show that it is difficult, if not impossible, to get reliable online estimates of the three coefficients of the polynomial $C_Q(J)$ when they are estimated as independent parameters. The reason is that the natural variations in J are relatively small, and even with extensive airspeed changes and maneuvers designed to increase the observability of these parameters, it turns out to be difficult to get sufficiently accurate estimates to reliably detect icing and isolate it from faults.

Instead, we assume that icing has a linear scaling effect on $C_Q(J)$. The scaling is parameterized by the scalar variable θ_1 with nominal value $\theta_1^* = 1$. The model can then be described as

$$Q_a = Q_a^* \cdot \theta_1 \tag{5}$$

where $Q_a^* = \rho \frac{\omega^2}{4\pi^2} D^5 C_Q(J)$ is defined by the nominal (ice-free) values of the coefficients in $C_Q(J)$. Icing is then characterized by $\theta_1 > 1$.

Preliminary findings from icing wind tunnel tests suggest that the scalar multiplicative icing model, as given in equation (5), could be well suited to describe the icing dynamics when the changes in airspeed is small.

B. System Faults and Degradation due to Icing

We will now concretize the meaning of the fault concept. We shall then formalize *the fault states* of the system.

System faults and degradation occur whenever any of the parameters of the system deviate from its nominal value. To represent this, the nominal parameter vector $\boldsymbol{x}^*$ is introduced:

$$\boldsymbol{x}^* = [\theta_1^* \quad c_v^* \quad I_o^*]^T$$

The nominal vector $\boldsymbol{x}^*$ gives the parameter values of a fault-free and non-degraded system, which is related to the actual parameter vector $\boldsymbol{x}$ and the deviation $\boldsymbol{\epsilon}$ through the equation $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{\epsilon}$. Note that these vectors are partitioned into:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \boldsymbol{x}^{(3)} \end{bmatrix} \tag{6}$$

where $\boldsymbol{x}^{(1)}$ contains the first element, $\boldsymbol{x}^{(2)}$ contains the second- and $\boldsymbol{x}^{(3)}$ contains the third element. The same notation is also used for $\boldsymbol{\epsilon}$ and similar vectors.

A fault in the propulsion system is present whenever any element in $\boldsymbol{\epsilon}$ is sufficiently different from zero. However, a simultaneous occurrence of multiple faults would be exceedingly rare. Thus, it is assumed that only one error can occur at a time. The propulsion system has 4 possible states related to faults and degradation. The following will be referred to as the *fault states* m:

0) No fault: $\boldsymbol{x} = \boldsymbol{x}^*$
1) Propeller icing: $\boldsymbol{x}^{(1)} \neq \boldsymbol{x}^{(1)*}$
2) Change in viscose friction: $\boldsymbol{x}^{(2)} \neq \boldsymbol{x}^{(2)*}$
3) Change in static friction: $\boldsymbol{x}^{(3)} \neq \boldsymbol{x}^{(3)*}$

Note that we will also refer to the much simpler binary state: **healthy** vs. **faulty**. Both states are modeled as Markov processes. We refer to the two Markov processes as the

$$\text{Health Model: health} \in \{\text{Healthy, Faulty}\}$$
$$\text{Fault Model: fault} \in \{0,\ 1,\ 2,\ 3\}.$$

Thus, the *health state* reefer to the state of the health model. Similarly, the *fault state* reefer to the *Fault model.* The Markov chains are illustrated in Figure 1.

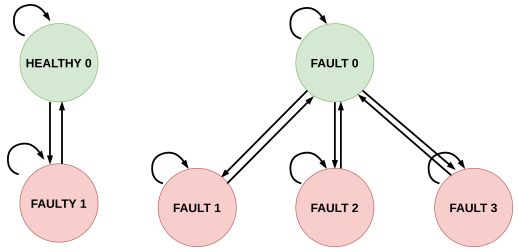

Fig. 1. Left: The Markov chain of the Health state. Right: The Markov chain of the Fault state.

Both models have associated transitions probabilities. These are defined by a transition matrix

$$\Pi^i_{k,k_\Delta} = \begin{bmatrix} \pi_{1,1} & \dots & \pi_{1,n} \\ \vdots & \dots & \vdots \\ \pi_{n,1} & \dots & \pi_{n,n} \end{bmatrix} \in \mathcal{R}^{n \times n} \tag{7}$$

where $i \in \{f, h\}$ refers to the given model, and n is the number of states. The *Health* state has $n = 2$ and the *Fault* state has $n = 4$. $\pi_{i,j} = p(m_k = j | m_{k_\Delta} = i)$ gives the transition probability from state i to j. The first row therefore gives the transition probability from a fault free state to any other state.

C. State Space Representation for Estimation

The parameters are assumed to evolve according to a random walk process. Thus, any change from time index k to $k+1$ is only attributable to process noise $\boldsymbol{v}_k$. This results in a process model given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_k \tag{8}$$

The measurement model of the system is derived from the torque balance in the system model given by equation (1). The airspeed V_a, angular velocity ω and the motor current I_e are assumed to be measured. Substituting equation (2), (3), (4) and the parameter vector $\boldsymbol{x}_k$ into equation (1) yields

$$y_k = [Q_a \quad \omega \quad k_e]\boldsymbol{x}_k + w_k \tag{9}$$

with $y_k = -k_e I_e + \Theta\dot{\omega}$. In cruise mode the UAV speed controllers will maintain an almost constant propeller speed. This motivates the simplifying assumption that

$$\dot{\omega} = 0 \tag{10}$$

which leads to $y_k = -k_e I_e$. This assumption will be used in the fault detection algorithms in this paper. The measurement model (9) forms the basis for the measurement matrices $\boldsymbol{C}_k^{(1)} = Q_a$, $\boldsymbol{C}_k^{(2)} = \omega$ and $\boldsymbol{C}_k^{(3)} = k_e$. This leads to

$$y_k = [\boldsymbol{C}_k^{(1)} \quad \boldsymbol{C}_k^{(2)} \quad \boldsymbol{C}_k^{(3)}] \begin{bmatrix} \boldsymbol{x}_k^{(1)} \\ \boldsymbol{x}_k^{(2)} \\ \boldsymbol{x}_k^{(3)} \end{bmatrix} + w_k \tag{11}$$

$$\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}), \quad w_k \sim \mathcal{N}(0, R) \tag{12}$$

Notice that y_k and $\boldsymbol{C}_k$ are *time dependant* functions of ω, V_a and I_e. This relationship is implicitly assumed throughout the paper. The variables ω, V_a and I_e will often be refereed to as a the measurement, $\boldsymbol{z_k} = [\omega, V_a, I_e]$.

III. FAULT DETECTION AND DIAGNOSIS

We present a framework for sequentially *detecting* faults and icing, and then correctly *identifying* the fault state. The framework will therefore be presented as modules. We start by an overview of the main algorithm and the formulation of multiple relevant hypotheses. The key elements of the algorithm are then described in more detail: *the Kalman filter* and *the Bayes filter*. Then, the detection and identification algorithms are presented.

The FDI framework presented here can be generalized and applied to other FDI problems. This allows the adoption of this algorithm to a wide range of UAV electric propulsion systems. However, some design decisions in this FDI algorithm are based on specific model assumptions.

A. Main idea

This section describes the system transitions between the detection step and the isolation step. This entails giving a descriptive overview of the relevant signals and how they propagates through the system.

The detection step is concerned with the Markov process, referred to as the *Health* model. It aims to detect transitions from *healthy* to *faulty*. It therefore provides a binary hypothesis, *transition to fault* or *no fault*. This is formalized with the detection hypothesis H_D:

$$H_D \in \{\text{true, false}\} \tag{13}$$

The isolation step is concerned with the *Fault* state. It attempts to determine witch transition is most likely to have occurred. The hypothesis space is therefore

$$H_I \in \{0, 1, 2, 3\} \tag{14}$$

where $H_I = 0$ is no fault, $H_I = 1$ is fault 1 (icing), $H_I = 2$ is fault 2, and $H_I = 3$ is fault 3.

The structure of the algorithm is shown in Figure 2. It is assumed that the UAV is initially in a fault free state. The detection algorithm is initialized with the nominal state $\boldsymbol{x}^{(*)}$ and processes a stream of measurements $\boldsymbol{z_k}$. The detection algorithm executes until a fault is detected. This can be seen in the feedback loop shown in Figure 2. If a fault is detected ($H_D =$ false), then the algorithm will start the identification algorithm and send a command for a small change in airspeed to the autopilot in order to enhance observability through excitation.

Inputs to the isolation algorithm are the measurement $\boldsymbol{z_k}$ and the nominal state $\boldsymbol{x}^{(*)}$. The isolation algorithm will after convergence output an hypothesis H_I, where $H_I = 0$ implies that the detection algorithm had a Type 1 error (false detection), whereas $H_I = i > 0$ imply that the system is in fault state i.

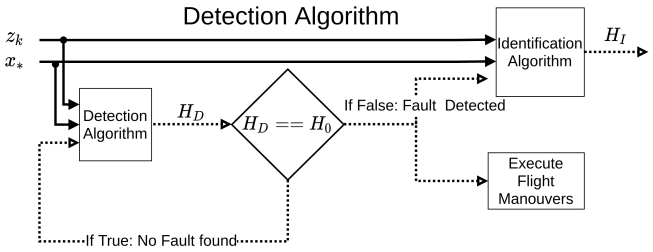

Fig. 2. The data flow of the FDI algorithm. It can be seen that the detection algorithm will launch the isolation algorithm if a fault is found. Dashed lines represents a binary/integer signal. Solid lines represent continuous values.

B. Estimation

The Kalman filter and the Bayes Filter will now be introduced [18]. These will be used by both the detection and the isolation step. We also formalize the concept of a static hypothesis models.

1) Kalman Filter: For each of the fault modes a Kalman filter is formulated. Since it is assumed that only one fault can occur at the time, the process noise is only affecting one of the sub-states $\boldsymbol{x^{(i)}}$, while the other states can be treated as constant. The state space model can therefore be rewritten as

$$\boldsymbol{x}_{k+1}^{(i)} = \boldsymbol{x}_k^{(i)} + \boldsymbol{v}_k^{(i)} \tag{15}$$

$$y_k^{(i)} = \boldsymbol{C}_k^{(i)} \boldsymbol{x}^{(i)} + [\boldsymbol{C}_k^{(j)} \quad \boldsymbol{C}_k^{(\ell)}] \begin{bmatrix} \boldsymbol{x}^{(j)*} \\ \boldsymbol{x}^{(\ell)*} \end{bmatrix} + w_k \tag{16}$$

where j and ℓ denote the two remaining modes other than i. The filters will have the same form for each mode $i \in \{1, 2, 3\}$. We refer to the Kalman filter of model i as $KF^{(i)}$. The prediction step is given by:

$$\hat{\boldsymbol{x}}_{k|k-1}^{(i)} = \hat{\boldsymbol{x}}_{k-1}^{(i)}, \qquad \in \mathcal{R}^{\delta_i} \tag{17}$$

$$\boldsymbol{P}_{k|k-1}^{(i)} = \boldsymbol{P}_{k-1}^{(i)} + \boldsymbol{Q}^{(i)}, \qquad \in \mathcal{R}^{\delta_i \times \delta_i} \tag{18}$$

$$\hat{y}_{k|k-1}^{(i)} = \boldsymbol{C}_k^{(i)} \hat{\boldsymbol{x}}_{k|k-1}^{(i)}, \qquad \in \mathcal{R}^1 \tag{19}$$

The update step is given by

$$\nu_k^{(i)} = y_k - \hat{y}_{k|k-1}^{(i)}, \qquad \in\mathcal{R}^1 \tag{20}$$

$$S_k^{(i)} = \boldsymbol{C}_k^{(i)} \boldsymbol{P}_{k|k-1}^{(i)} \boldsymbol{C}_k^{(i)T} + R, \qquad \in\mathcal{R}^{1\times 1} \tag{21}$$

$$\boldsymbol{W}_k^{(i)} = \boldsymbol{P}_{k|k-1}^{(i)} \boldsymbol{C}_k^{(i)T} (S_k^{(i)})^{-1}, \qquad \in\mathcal{R}^{\delta_i\times\delta_i} \tag{22}$$

$$\hat{\boldsymbol{x}}_k^{(i)} = \hat{\boldsymbol{x}}_{k|k-1}^{(i)} + \boldsymbol{W}_k^{(i)} \nu_k^{(i)}, \qquad \in\mathcal{R}^{\delta_i} \tag{23}$$

$$\boldsymbol{P}_k^{(i)} = (\boldsymbol{I} - \boldsymbol{W}_k^{(i)} \boldsymbol{C}_k^{(i)}) \boldsymbol{P}_{k|k-1}^{(i)}, \qquad \in\mathcal{R}^{\delta_i\times\delta_i} \tag{24}$$

where $\delta_i = \dim(\boldsymbol{x}^{(i)})$ is the dimensionality of the state $\boldsymbol{x}^{(i)}$. Thus, the dimensions of the equation (17) and (18) depend on the mode i. This is illustrated in Figure 3, which shows 3 filters running in parallel.

Also note that the covariance of the measurement noise R in equation (21) is the same for all modes. This follows from the fact that all filters depend on the same physical measurements.

2) Static Hypothesis Models: We will introduce the so-called *Static Model Hypotheses*, which are models that assume that the model parameters remain fixed. For the fault free case, this gives

$$\begin{aligned} Y_k^{(0)} &= y_k - [\boldsymbol{C}_k^{(1)*} \quad \boldsymbol{C}_k^{(2)*} \quad \boldsymbol{C}_k^{(3)*}] \begin{bmatrix} \boldsymbol{x}_k^{(1)*} \\ \boldsymbol{x}_k^{(2)*} \\ \boldsymbol{x}_k^{(3)*} \end{bmatrix} + w_k \\ &= y_k - \boldsymbol{C}_k^* \boldsymbol{x}_k^* + w_k \end{aligned} \tag{25}$$

Note that we will also make use of static models for fault state $i \in \{1,\ 2,\ 3\}$. In this case it is assumed that some estimate $\hat{\boldsymbol{x}}_{k_s}^{(i)}$ was sampled at time $k_s \leq k$. This gives rise to the static measurement model

$$Y_k^{(i)} = \boldsymbol{C}_k^{(i)} \hat{\boldsymbol{x}}_{k_s}^{(i)} + [\boldsymbol{C}_k^{(j)} \quad \boldsymbol{C}_k^{(\ell)}] \begin{bmatrix} x_k^{(j)*} \\ x_k^{(\ell)*} \end{bmatrix} + w_k \tag{26}$$

The static measurement models directly output the innovation $\nu_k^{(i)}$. The covariance $S_k^{(i)}$ of $\nu_k^{(i)}$ is given by the measurement noise:

$$\nu_k^{(i)} = Y_k^{(i)} \tag{27}$$

$$S_k^{(i)} = R \tag{28}$$

R will be the same for all static filters. In the reminder, the context (i.e block diagrams in Figures 3 and 4) should make it intelligible when $\nu_k^{(i)}$ and $S_k^{(i)}$ are taken from the static hypothesis model and when they are taken from a Kalman filter.

3) Bayes Filter: The Bayes filter allows us to directly compare the performance of various hypotheses $\mathcal{H}_i$ corresponding to modes $i \in \{0, 1, 2, 3\}$. The filter follows directly from Bayes Theorem [18]:

$$p(\mathcal{H}_i|\boldsymbol{z}_{0:k}) = \frac{\mathcal{N}(\nu_k^{(i)}, 0, S_k^{(i)}) p(\mathcal{H}_i|\boldsymbol{z}_{0:k-1}^{(i)})}{\sum_{j=0}^{M} \mathcal{N}(\nu_k^{(j)}, 0, S_k^{(j)}) p(\mathcal{H}_j|\boldsymbol{z}_{0:k-1}^{(j)})} \tag{29}$$

where $\mathcal{N}(\nu_k^{(i)}, 0, S_k^{(i)})$ is the (Gaussian) likelihood of the zero-mean innovation $\nu_k^{(i)}$ given covariance $S_k^{(i)}$. The Bayes filter can compare a set of filters $Y^{(i)}, i \in \{0, 1, 2, 3\}$ and $KF^{(j)}, i \in \{1, 2, 3\}$ based on the likelihood:

$$\ell^{(i)} = \mathcal{N}(\nu^{(i)}, 0, S) \tag{30}$$

where $\nu^{(i)}$ varies with the filters. This paper will be limited to using $S_k^{(i)} = R$ for $i \in \{0, 1, 2, 3\}$, where R is given by equation (28). Further, R will be the same for every model. This goes for both the Kalman- and Bayes filters.

The innovation covariance S determine Bayes Filter sensitivity. To see this, consider the zero-mean Gaussian probability density functions (PDF) defined by S. Thus, for to (different) innovations, $\nu^{(i)} > \nu^{(j)}$ would imply $\ell(\nu^{(i)}) < \ell(\nu^{(j)})$. The smaller the $S_k^{(0)}$ is, the more narrow the distribution will be. This will increase the probability differences between the innovations. For example, the Gaussian zero mean likelihoods $\ell_n(\cdot)$ and $\ell_m(\cdot)$ have variances, S_n and S_m. $S_n < S_m$ will then imply that $\ell_n(\nu^{(i)}) - \ell_n(\nu^{(j)}) > \ell_m(\nu^{(i)}) - \ell_m(\nu^{(j)})$. Thus, smaller values of S, make the algorithm *sensitive* to differences between the innovations.

The Bayes filter is recursive and initialized according to the prior $p(\mathcal{H}_i|\boldsymbol{z}_0^{(i)}) = p(\mathcal{H}_i)$. We always assume that the system starts in the fault free case. The prior is therefore given by the first row of the Markov matrix. Keep in mind that we are operating with two different Markov models. The *Health* model will initialize according to Π^h, while the *Fault* model will use Π^f.

The concept of a detection/isolation **window** L, can be introduced at this point. The window L determines the length of time (or number of samples) the Bayes Filter should process before returning an hypothesis. For example, $L = 10$ seconds would mean that the Bayes Filter would process the data from the last 10 seconds. The mode i, with the highest probability would then be returned as the hypotheses. This paper will make reefer to both a *detection-* and an *isolation window*. These are the windows used be the respective algorithms.

We conclude our discussion by noting how this filter is different from the Magill filter [23]. This implementation use that $S_k^{(i)} = R$. This yields a static measurement covariance. Contrarily, the Magill filter receive $S_k^{(i)}$ directly from Kalman filter $\boldsymbol{KF}^{(i)}$, as given in equation (21). Our reasoning is that using equations (21) for calculating $S_k^{(i)}$ would make filters probability differ substantially. This is because the process noise $\boldsymbol{Q}^{(i)}$ will directly affect $S_k^{(i)}$. This can be seen in equations (18) and (21). The effect is prominent since the process noise will differ, by orders of magnitude, between models. Thus, different values of $S_k^{(i)}$ is likely to introduce big variations in $\ell^{(i)}$. However, the magnitude of the process noise of one filter should should not make it more (or less) likely than other filters.

C. Fault Detection

The fault detection algorithm aims at detecting faults as defined in equation (13). A basic idea is to use a Bayes filter to compare the output of a Kalman filter, $KF^{(i)}$, against the

static zero hypothesis model, $Y_k^{(0)}$. A fault is detected if some Kalman filter, $KF^{(i)}$ $i \in \{1,2,3\}$ outperforms the fault-free static model $Y_k^{(0)}$ during the interval $\{k-L,...,k-1,k\}$, where the integer L is referred to as the detection window.

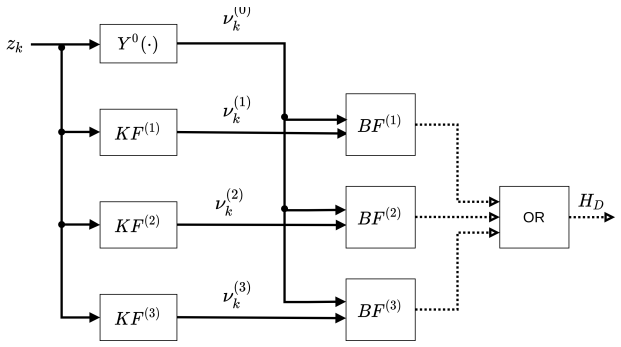

Fig. 3. Block diagram with detection algorithm.

A block diagram of the algorithm is given in Figure 3. It can be seen that the measurement $\boldsymbol{z}_k$ is propagated through three computational layers before a hypothesis H_D formed. The layers are as follows:

1) The Kalman filters and static model use the measurement $\boldsymbol{z}_k$ as input and outputs:
$$(\nu_k^{(i)}, S_k^{(i)}) \leftarrow KF^{(i)}(\boldsymbol{z}_k), \quad i \in \{1,2,3\}$$
$$(\nu_k^{(0)}, S_k^{(0)}) \leftarrow Y_k^{(0)}(\boldsymbol{z}_k)$$
2) Each Bayes Filter receives the data from *one* Kalman filter and the static hypothesis model. They each form a hypothesis based on the L last samples
$$\mathcal{H}_i \leftarrow BF^{(i)}((\nu_k^{(0)}, S_k^{(0)}), (\nu_k^{(i)}, S_k^{(i)})), \quad i \in \{1,2,3\}$$
3) The hypotheses are combined using a logical *OR* gate:
$$H_D = \mathcal{H}_1 \vee \mathcal{H}_2 \vee \mathcal{H}_3$$

Each Kalman filter, $KF^{(i)}$, is initialized to $\boldsymbol{x}^{(i)*}$. The estimate $\hat{\boldsymbol{x}}_k^{(i)}$ will then be updated as measurements $\boldsymbol{z}_k$ are made available.

Note that we use 3 different Bayes Filters. Each filter is initialized according to the transition probability given by the first row of Π^h. Each Bayes Filter is designed to detect different faults. This partitioning avoids the situation in which different filters $KF^{(i)}$ start competing for the probability space. This could occur because multiple Kalman filters often outperform the static model $Y^{(0)}$ when an error occurs. In practice, the detection window reinitializes the filter probabilities to the prior distribution.

D. Fault Isolation

The fault isolation algorithm aims at isolating the true system fault after some fault has been detected. The basic idea of the algorithm is to generate a set of static model hypotheses, and alter the airspeed to introduce a perturbation. The static model of the true model hypothesis will then outperform the false ones that will drift as a consequence of the perturbation. It is important that the models have static parameters because otherwise the parameter estimation will eventually mask the faults and not be helpful to isolate them. A block diagram of the algorithm is given in Figure 4.

The isolation algorithm works as follows

1) The airspeed V_a is increased. All Kalman filters estimates $\hat{\boldsymbol{x}}^{(i)}$ will quickly change due to the airspeed-change. The estimates are then given time to stabilize. This step can be seen in Figure 2.
2) The algorithm then samples the estimates $\hat{\boldsymbol{x}}_{k_S}^{(i)}$ at some time k_s. The time k_s is given by a clock signal, as shown in Figure 4. This is used to generate the static models
$$\boldsymbol{Y}^{(i)} \leftarrow \hat{\boldsymbol{x}}_{k_s}^{(i)}, \quad i \in \{1,2,3\}$$
3) The airspeed V_a is decreased, back to its original value.
4) The innovations $\nu^{(i)}$ of the static models are given to a Bayes filter. The filter generates a hypothesis based on the last L samples. The filter is initialized according to the first row of Π^f.
$$H_I \leftarrow BF^{(i)}(\nu_{k-L,\dots,k)}^{(i)}), \quad i \in \{1,2,3\}) \tag{31}$$

E. Algorithm Tuning

The success of the **FDI** framework depends on rigours tuning. The most important tuning aspects must therefore be discussed. We will specifically cover how to tune both the Kalman Filter and the Bayes filter.

1) Kalman filters: Tuning the Kalman filters is essential for achieving a good results. However, it is assumed that the reader is familiar with the tuning Kalman filters. The three most important will therefore be covered briefly.

1) The process noise covariance $\boldsymbol{Q}_k^{(i)}$ of Kalman filter $\boldsymbol{KF}^{(i)}$, should be commensurate with the magnitude and time constant of the fault dynamics. Furthermore, different faults, e.g $\boldsymbol{\epsilon}^{(i)}$ and $\boldsymbol{\epsilon}^{(j)}$, will tend to differ in these respects. Thus, the filters must be tuned independently.
2) The response time of a filter may be more important than its accuracy. Keep in mind that the purpose of the filters is to detect faults quickly. Furthermore, the isolation algorithm should not be executed before the Kalman filters have stabilized around some value. Thus, a quick filter is desirable. An example of this can be seen in Figure 8. The top plots shows the a noisy estimate of $\boldsymbol{x}^{(2)}$. The given filter was excellent at detecting faults quickly.
3) The goal is not to achieve a perfect estimate. The important point is that the correct filter outperforms all other filters. Figure 7 exemplify this. It is easy to see that the estimate of $\boldsymbol{x}^{(1)}$ is both slow and noisy. This is due to a low signal to noise ratio. However, the filter still performs adequately for both identification and isolation.

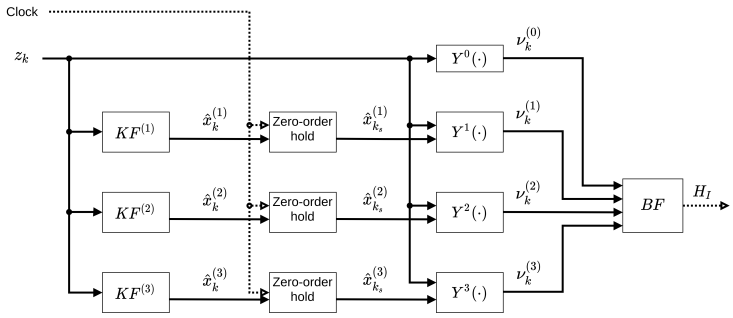

Fig. 4. Block diagram with fault isolation algorithm

2) Bayes filter: The tuning of the Bayes Filter determines both the success of the detection- and isolation step. The main tuning parameters are:

1) $\boldsymbol{S}_k^{(i)}$: The innovation variance of the Bayes filter (witch determine the sensitivity).
2) Π^f and Π^f: The Markov matrices (which determine the prior probability distributions)
3) L: The window length (which determine how many samples the filter should use)

There will always be an interplay between these parameters, which should be kept this in mind as we discuss them separately. For example, high sensitivity will allow for a shorter window length. The following will frequently refer to Figure 5. Note that this figure is the result of simulations, as covered in V.

The sensitivity is the most crucial aspect of the tuning process. Its effect can be seen in figure 5. The all the plots in the left column shows scenarios where the system is fault free. The right column show scenarios where the system is subjected to icing. Note that the icing has developed when the filter initialize. Thus, no change in θ_1 occurs during the execution. The plots in the top row has the highest sensitivity. The sensitivity then decrease down the rows. The innovation $\nu^{(1)}$, from Kalman filter $KF^{(1)}$ is exactly the same within each column.

It should be clear from the top left plot, that high sensitivity has obvious problems: The risk of false positives increase drastically. Notice that the red line, i.e the false hypothesis, almost surpass the blue line multiple times. High sensitivity also comes with a clear advantage: fast convergence rates. This can be seen in the top right. The correct hypothesis is isolated in about 10 seconds. The opposite extreme is found in the bottom row. The bottom left show that low sensitivity makes a false positive very unlikely. However, the bottom right shows that detecting the fault would take more than 3 minutes. A compromise is found in the middle row. This was the sensitivity level which functioned best in the simulations.

The detection window length L, must be adapted to the sensitivity. The horizontal lines of figure 5 show various options for L. Keep in mind that the algorithm will choose the hypothesis with the highest value. This happens when the lines reach the end of a detection window. It should be clear that we must chose a window that is long enough for the correct solution to be chosen. At the same time, the window must not be too long. This has the aforementioned problem that a fault would not be detected fast enough. Furthermore, a core model assumption is that the system does not change state while the Bayes Filter is running. For the simulations, a 100 second window was chosen.

The Transition probabilities also play an important role. All the plots in Figure 5 are initialized with the same prior distribution. However, it should be easy to imagine the effect on different priors. For example, the top left plot would in many instances return a false positive if the lines started closer together. Oppositely, the bottom right plot would return a true positive, if the lines started closer together.

It is straight forward to imagine more sophisticated approaches to determine both Π^h and Π^f. The transition probabilities could be, for example, varied according to environmental factors, such as power load, temperature and humidity. However, in the simulations, the Markov matrices, Π^h and Π^f were selected as simple as possible. Furthermore, only the first row of Π^h and Π^f are of interest. This follows from the fact that all simulations are initialized in a healthy state. The first row of both matrices gives the transition probabilities from healthy states. For the detection step, the first row of Π^h was set to:

$$\Pi_1^h = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix} \tag{32}$$

Thus, remaining in a healthy state is considered twice as likely as transitioning to a faulty state. The isolation step assumes a discrete uniform distribution:

$$\Pi_1^f = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \tag{33}$$

It is often assumed that the probability of remaining in the current state has the highest probability. However, it should be accounted for that Π^f is only used if the detection step has found a transition. This transition introduces a prior that should be accounted for in Π^f.

F. Evaluation Criteria

Specific evaluation criteria has been set to evaluate the algorithm. The *criteria* specify a baseline for what the algorithm must achieve in order to be considered functional.

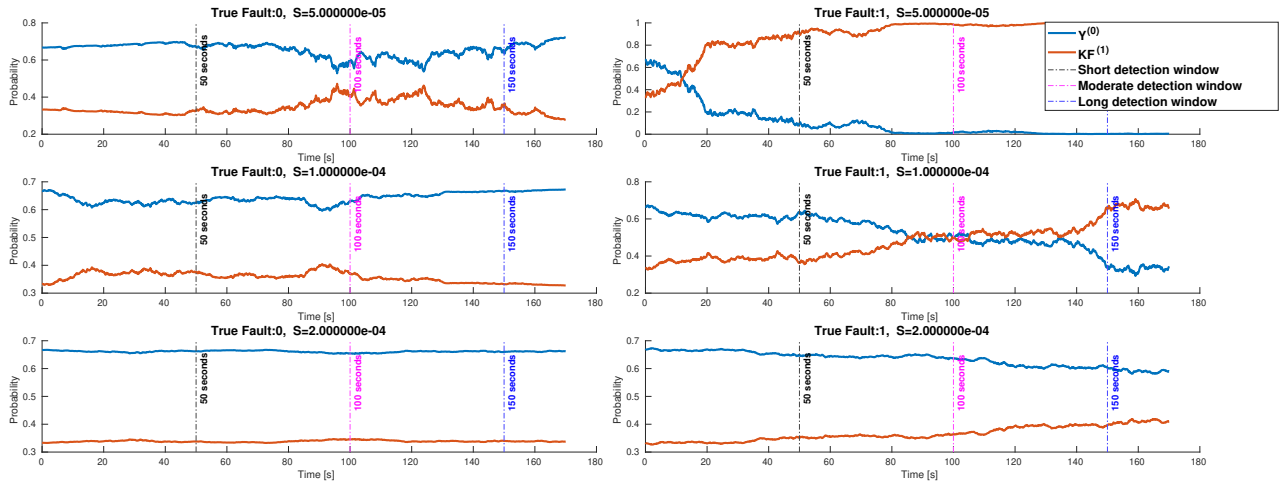

Fig. 5. All plots shows the output of a Bayes Filter during the detection step. The system is healthy, (fault = 0) in all the plots in the left column. The blue line therefore represents the true hypothesis in the left columb. The system subjected to is in fault state 1 in all the plots in the right column. The *sensitivity* of the filters is highest in the top row. The lowest sensitivity can be found in the bottom row. Thus, the red line represents the true hypothesis in the right column. The vertical dashed lines illustrate possible values of the detection window, L.

The evaluation criteria define a binary set of requirements: These are as follows:

1) False Alarm rate should be zero: The algorithm should not conclude that a healthy system has transitioned to a *faulty* state.
2) Fault detection rate should be 100%: The algorithm should always conclude that a faulty system has transitioned to a *faulty* state.
3) Fault Isolation should have 100% accuracy: The algorithm should always isolate the correct fault state of the system.

The presented algorithm is still at low technology readiness level. The presented results is therefore focused on satisfying these criteria using simulations. This allow us to determine proof of concept. Future work should introduce performance metrics for assessing an operational range. For example, what is the highest level of measurement noise the algorithm can function in? How quickly can a fault be detected?

IV. SIMULATION SETUP

The simulations were performed using the Ardupilot software-in-the-loop framework. This framework interfaces a flight dynamics simulator implemented in Simulink. The Simulink model includes the propulsion model (1) – (4) and aerodynamics of the X8 fixed-wing UAV based on [7], [6]. This work has been further developed by the inclusion of fault dynamics.

Only a subset of the possible transitions will be simulated. The system always start in a healthy state. The system will either stay in this state or transition to a faulty state. The implication of this is that the transitions from a faulty to a healthy state is neither modelled or simulated. This be visually understood by noting that the system always start in the green circles in Figure 1. Naturally, future work should encompass all possible transitions.

A. Simulated Fault dynamics

The fault dynamics are simulated using a sigmoidal function, gradually increasing between over 50 seconds. This occur after around 90 seconds, as illustrated in Figure 6. This is a pragmatic choice to model a system where a mathematical model is lacking. All the systems faults are simulated according to the same sigmoidal function. This goes for both the rise time and the relative growth of the fault.

For a given fault, the associated variable reach a steady state after a proportional change by a factor of 1.1. This is exemplified in figure 6. The figure shows the fault development of θ_1.

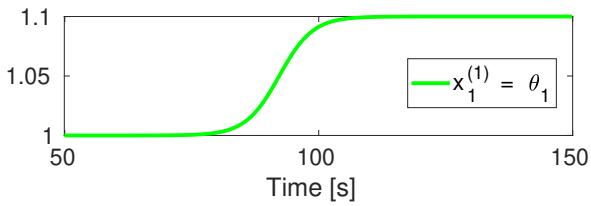

Fig. 6. This plot shows simulated fault dynamics of $\boldsymbol{x}^{(1)}$.

B. Measurement Noise

The framework relies on measuring ω, V_a and I_e. Naturally, these measurements will be subjected to noise, w_i, w_{V_a}, and w_ω. This noise has been simulated as additive, zero mean, Gaussian noise. The standard deviation of the given noise has been chosen to lie between 0.1% and 0.2% of the

mean signal values, i.e. ω has noise covariance $5 \cdot 10^{-1}$, V_a has covariance $7 \cdot 10^{-4}$, and I_e has covariance 10^{-5}.

Note that it is permissible for these noise levels to be less than that of real world sensors. This is because the model is not limited to raw measurements. For example, the FDI typically runs at a lower sampling rate than the raw measurements, which means that decimating (or averaging) several measurements would effectively reduce the noise. Moreover, the electrical model from [17] opens the door to estimating ω or I_e, or both, with a Kalman filter. The model could then make use of the less noisy $\hat{\omega}$ and $\hat{I}_e$.

The square terms in equations (3) and (4) will affect the noise. Specifically, the squaring of Gaussian noise components will result in the introduction of χ^2 noise terms. Thus, the distribution of the measurement noise w_k from equation (11), will be a mixture of Gaussian and χ^2 terms. Thus, the expected value of w_k will be positive. This will introduce a bias to the system. However, for the given noise levels, this bias is small enough to be ignored.

V. Simulation Results

Four different simulation scenarios are used to illustrate the performance of the presented fault detection and isolation system, one for each of the fault states listed in Section II-B. For the given implementation, the system satisfies the criteria we defined in section III-F for all four scenarios.

The results of the detection and isolation are covered separately. However, one should keep in mind that the FDI algorithm is only successful if both steps succeed. In the results we show a series of *probability* plots over time. Such plots can be seen in Figure 5, for example. These plots show the result of the Bayes filter iterating over a window of length L. The graph with the highest value at the end of the window is chosen as the most likely hypothesis. Note that the Bayes filter is generally not run over the entire data set. Instead, it is reinitialized to its priors and executed periodically.

A. Detection

It has been found that all fault scenarios can be detected using our method. We also found that false positives can be avoided through good tuning.

1) No-fault scenario: The algorithm was successful in avoiding false alarms. The algorithm did not show any false alarms in the chosen scenario. Note that this is the result of tuning for this scenario and should be analyzed over a wider range of scenarios.

2) Propeller icing scenario: The propeller icing $\boldsymbol{x}^{(1)}$ was successfully detected. This can be seen in the bottom row of Figure 7. The plots shows various Bayes Filter outputs throughout the simulation. The columns represent slices in time. Each row shows the probability of a particular Kalman filter. The goal here is for the red line to climb above the blue line (no fault). This signifies a successful detection. It can be seen that the it takes over 100 seconds (after the fault occur) before the error is detected. The slow detection time is in large part due to the slow convergence of the state estimate $\boldsymbol{x}^{(1)}$, as can be seen in the top plot of 7. It can be seen in the top plots of Figure 8 and 9 that $\hat{\boldsymbol{x}}^{(2)}$ and $\hat{\boldsymbol{x}}^{(3)}$ converge much faster. This results in shorter detection times.

An overview of $KF^{(1)}$ can be seen in the top plots of Figure 7. The top plot shows the true parameter and the Kalman filter estimates. The plot also show the Fault free (nominal) parameters. It should be evident that the filter estimates $\hat{\boldsymbol{x}^{(1)}}$ with moderate success. This is a result of the low signal to noise ratio between aerodynamic thrust coefficients and the measurement noise. The middle plot show that the error covariance $\boldsymbol{P}^{(1)}$ stops decreasing after about 300 seconds.

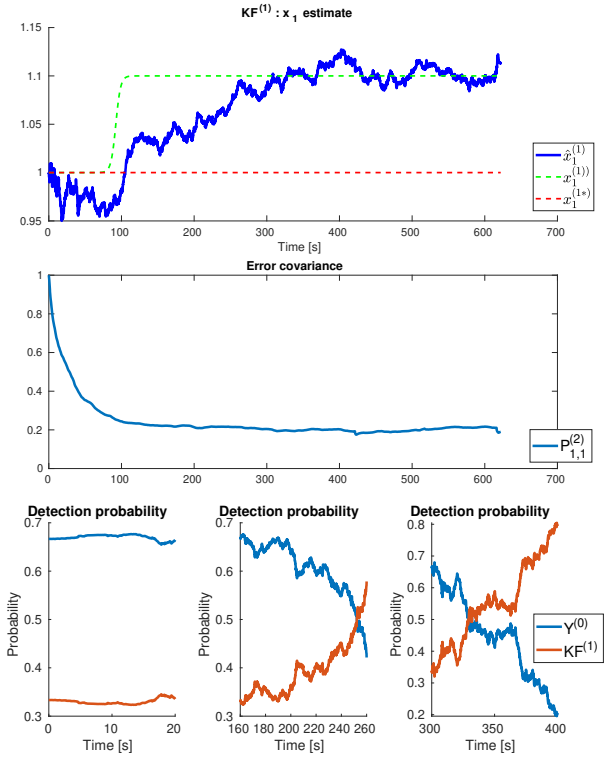

Fig. 7. Fault scenario 1: Top plots show state and its estimate. The middle plot show the error covariance $\boldsymbol{P}$. The bottom row display detection probabilities at different slices in time.

3) Change in viscous friction scenario: The detection algorithm detected the change in viscose friction $\boldsymbol{x}^{(2)}$ quickly. This can be seen in Figure 8. The top plot shows the dynamics of $\boldsymbol{x}^{(2)}$ and the estimate $\hat{\boldsymbol{x}}^{(2)}$. It can be seen that the estimate $\hat{\boldsymbol{x}}^{(2)}$ responds quickly to the change. This allows the fault to be detected very quickly. The fast detection can be seen in the 3 bottom plots of Figure 8. The Bayes filter becomes very confident 30 seconds after the error occurs.

Note that the estimate $\hat{\boldsymbol{x}}^{(2)}$ looks noisier than the estimate $\hat{\boldsymbol{x}}^{(3)}$ (see figure 9). However, $\boldsymbol{x}^{(2)}$ is many orders of magnitude smaller than $\boldsymbol{x}^{(3)}$. Furthermore, both estimates are subjected to the same measurement noise.

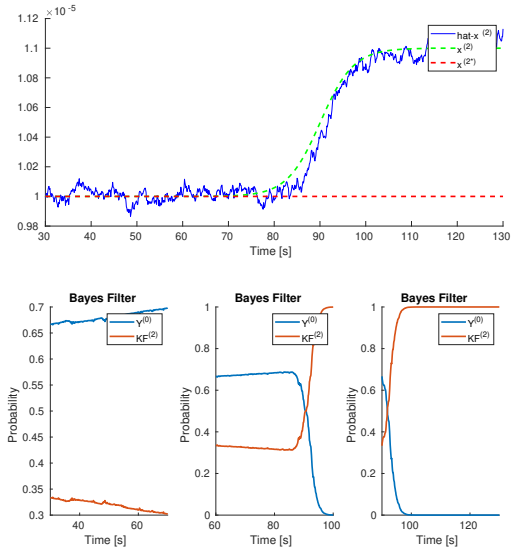

Fig. 8. Fault scenario 2. Top: True value and estimate of $\boldsymbol{x}^{(2)}$. Bottom: Probability plots from Bayes Filter at various time instances.

4) Change in static friction scenario: The change in static friction $\hat{\boldsymbol{x}}$ was successfully detected. The results of the estimation can be seen in Figure 9. Note that both the results and their plots are more or less the same as that of the change in viscous friction in Section V-A3.

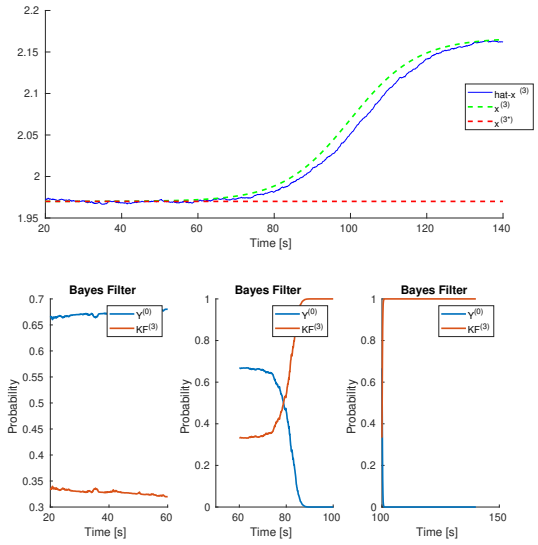

Fig. 9. Fault scenario 3. Top: True value and estimate of $\boldsymbol{x}^{(3)}$. Bottom: Probability plots from Bayes Filter at various time instances.

B. Isolation

The isolation algorithm manages to find the correct fault for all fault scenarios. This can be seen in Figure 10. The plot shows the output of the Bayes filter. It can be seen that the algorithm was successful in every case. However, it is clear that the icing fault is the hardest one to isolate. This can be seen in the top plot of Figure 10. Notice that $Y^{(1)}$ only marginally outperforms $Y^{(0)}$. The two plots below show $Y^{(2)}$ and $Y^{(3)}$. The Bayes filter was executed 40 seconds after the second air speed change (see Figure 11).

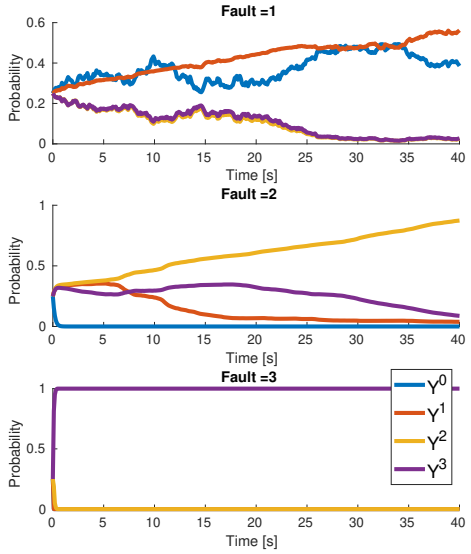

Fig. 10. Results of isolation algorithm for all faults. Left: True Fault = 1, Isolation hypothesis = 1. Middle: True Fault = 2, Isolation Hypothesis = 2, Right: True Fault = 3, Isolation Hypothesis = 3

The effectiveness of the isolation algorithm can also be seen in Figure 11. The true fault is given by $H_I = 2$. The left most bottom plot shows that finding the correct hypothesis, prior the identification step, is a challenging problem. In this plot, we see that the wrong hypothesis would have been chosen. One should also note the horizontal green lines in the two plots in the middle. The two horizontal green lines shown in the two plots indicate the parameter values chosen in the static models $Y^{(2)}$ and $Y^{(3)}$. These are generated right before the decrease in airspeed. Observe that the estimate $\hat{\boldsymbol{x}}_k^{(2)} \approx \hat{\boldsymbol{x}}_{k_s}^{(2)}$ and thus remains stable for all $k > k_s$. Contrary to this, it is found that both $\hat{\boldsymbol{x}}_k^{(3)}$ diverges from $\hat{\boldsymbol{x}}_{k_s}^{(3)}$ and $\hat{\boldsymbol{x}}_k^{(1)}$ diverges from $\hat{\boldsymbol{x}}_{k_s}^{(1)}$. It is these divergences that results from the perturbation of V_a that allows the isolation algorithm to find the true fault. The bottom right plot shows how the Bayes Filter confidently isolate returns $H_I = 2$. Note that this scenario, the fault dynamics only gave a 1% increase in $\boldsymbol{x}^{(2)}$.

VI. CONCLUSIONS

In this paper we propose a fault detection and isolation (FDI) framework for propeller icing and electric propulsion system faults. The method uses model-based estimators of the various faults, implemented with multiple Kalman and Bayes filters that are used to build separate detection and

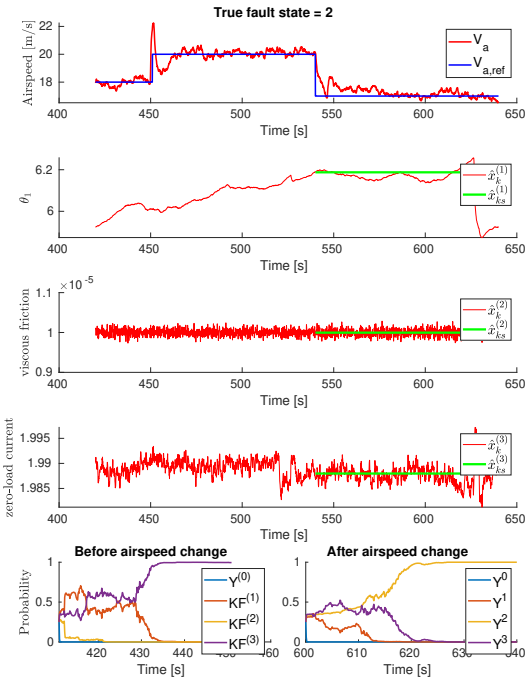

Fig. 11. Illustration of the isolation algorithm. The top plot shows the airspeed and its reference. The plots in the middle shows dynamic and static estimates. The bottom plots show outputs of the Bayes filter before and after the static filters are generated.

isolation algorithms. Clear evaluation criteria was defined for the method. The method is tested and shown to satisfy all the criteria in a simulation setup using a fixed-wing UAV simulator. Specifically, the method can detect and isolate faults corresponding to about 10 % change in parameter values.

ACKNOWLEDGEMENTS

This research was funded by the Research Council of Norway (RCN) through the Centres of Excellence funding scheme, grant number 223254 – NTNU AMOS, by RCN, Maritime Robotics and Equator Aircraft through the BIA FlightSmart project grant 282004, by RCN and UBIQ Aerospace through the BIA D*ICE Rotors project grant 296228, and a innovation scholarship provided by the Norwegian University of Science and Technology. We are grateful to Nicolas Müller at UBIQ Aerospace for sharing insights and experimental data on propeller icing.

REFERENCES

[1] Yang Liu, Linkai Li, Zhe Ning, Wei Tian and Hui Hu, Experimental Investigation on the Dynamic Icing Process over a Rotating Propeller Model *J. Propulsion and Power*, Vol. 34, pp. 933–946, 2018

[2] Yang Liu, Linkai Li, Wenli Chen, Wei Tian, Hui Hu, An experimental study on the aerodynamic performance degradation of a UAS propeller model induced by ice accretion process, *Experimental Thermal and Fluid Science*, vol. 102, pp. 101-112, 2019

[3] N. Müller, R. Hann, T. Lutz, The Influence of Meteorological Conditions on the Icing Performance Penalties on a UAV Propeller, Deutscher Luft- und Raumfahrtkongress (DLRK), 2020

[4] R. Hann, T. A. Johansen, UAV Icing: The Influence of Airspeed and Chord Length on Performance Degradation, *Aircraft Engineering and Aerospace Technology*, 2021

[5] B. C. Bernstein, C. A. Wolff, F. McDonough, An Inferred Climatology of Icing Conditions Aloft, Including Supercooled Large Drops. Part I, *Journal of Applied Meteorology and Climatology*, vol. 46, pp. 1857–1878, 2007

[6] A. Winter, R. Hann, A. Wenz, K. Gryte, T. A. Johansen, Stability of a Flying Wing UAV in Icing Conditions, 8th European Conference for Aeronautics and Space Sciences (EUCASS), Madrid, 2019

[7] K. Gryte, R. Hann, M. Alam, J. Rohác, T. A. Johansen, T. I. Fossen, Aerodynamic modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle, International Conference on Unmanned Aircraft Systems, Dallas, 2018

[8] G. Ducard, H. P. Geering, Efficient Nonlinear Actuator Fault Detection and Isolation System for Unmanned Aerial Vehicles, *J. Guidance, Control and Dynamics*, Vol. 31, pp.225-237, 2008

[9] E. Baskaya, M. Bronz, D. Delahaye, Fault detection and diagnosis for small UAVs via machine learning, IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), St. Petersburg, FL, 2017

[10] P. Freeman, R. Pandita, N. Srivastava and G. J. Balas, Model-Based and Data-Driven Fault Detection Performance for a Small UAV, *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 4, pp. 1300-1309, 2013

[11] H. M. Odendaal, T. Jones, Actuator fault detection and isolation: An optimised parity space approach, *Control Engineering Practice*, Vol. 26, pp. 222-232, 2014

[12] A. Hasan, T. A. Johansen, Model-Based Actuator Fault Diagnosis in Multirotor UAVs, International Conference on Unmanned Aircraft Systems, Dallas, 2018

[13] A. Cristofaro, A. P. Aguiar, T. A. Johansen, Icing Detection and Identification for Unmanned Aerial Vehicles using Adaptive Nested Multiple Models, *International Journal of Adaptive Control and Signal Processing*, Vol. 31, pp. 1584–1607, 2017

[14] M. M. Seron, T. A. Johansen, J. De Dona, A. Cristofaro, Detection and Estimation of Icing in Unmanned Aerial Vehicles using a Bank of Unknown Input Observers, Australian Control Conference, 2015

[15] K. L. Sørensen, M. Blanke, T. A. Johansen, Diagnosis of Wing Icing Through Lift and Drag Coefficient Change Detection for Small Unmanned Aircraft, Proc. 9th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Paris, 2015

[16] A. W. Wenz, T. A. Johansen, Icing Detection for Small Fixed Wing UAVs using Inflight Aerodynamic Coefficient Estimation, IEEE Aerospace Conference, Big Sky, Paper 2636, 2019

[17] E. M. L. Coates, A. W. Wenz, K. Gryte, T. A. Johansen, Propulsion System Modeling for Small Fixed Wing UAVs, International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, 2019

[18] S. Särkkä, Bayesian Filtering and Smooting. Cambridge University Press, 2013

[19] S. Hansen, M. Blanke, J. Adrian, A Framework for Diagnosis of Critical Faults in Unmanned Aerial Vehicles, IFAC Proceedings Volumes (IFAC World Congress), Vol. 47, pp 10555-10561, 2014

[20] S. Hansen and M. Blanke, Diagnosis of Airspeed Measurement Faults for Unmanned Aerial Vehicles, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, pp. 224-239, 2014

[21] T. A. Johansen, A. Cristofaro, K. L. Sørensen, J. M. Hansen, T. I. Fossen, On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors, International Conference on Unmanned Aircraft Systems, Denver, 2015

[22] A. W. Wenz, T. A. Johansen, Moving Horizon Estimation of Air Data Parameters for UAVs, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, pp. 2101-2121, 2020

[23] W. Chaer, R. Bishop, J. Ghosh, A mixture-of-experts framework for adaptive Kalman filtering, *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics*, vol. 27. pp. 452-64, 1997

[24] E. F. L. Narum, R. Hann, T. A. Johansen, Optimal Mission Planning for Fixed-Wing UAVs with Electro-Thermal Icing Protection and Hybrid-Electric Power Systems, Int. Conf. Unmanned Aircraft Systems, Athens, 2020