Blind Fault Identification of Air Data Sensors

Data-driven approach to fault diagnosis

Nirupama Sai Ramesh

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Blind Fault Identification of Air Data Sensors

Data-driven approach to fault diagnosis

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Nirupama Sai Ramesh

October 19, 2023

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Blind Fault Identification of Air Data Sensors

by

Nirupama Sai Ramesh

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: October 19, 2023

Supervisor(s):

Prof.Dr.Ir. Michel Verhaegen

Ir. Jacques Noom

Reader(s):

Dr.Ir. Coen de Visser

Dr. Nitin J Myers

Abstract

Model-based fault diagnosis methodologies rely on an accurate mathematical representation of a system's dynamics to effectively detect and localize faults. However, creating such models can be challenging, particularly for complex systems operating under diverse conditions. Furthermore, faults affecting the system can also modify its dynamics.

Given the limitations of model-based fault diagnosis, this study introduces a data-driven approach within the Blind System Identification framework. This approach can identify both the fault and the linear-time invariant model simultaneously. The mathematical formulation of this problem is expressed as a constrained least squares problem involving rank and sparsity constraints. To illustrate the application of this methodology, we demonstrate its effectiveness in diagnosing structured faults in Air Data Sensors using actual flight data obtained from the Cessna Citation II aircraft.

Table of Contents

List of Figures

List of Tables

Preface and Acknowledgements

In an age marked by a data explosion, control engineering trends have evolved with a pronounced shift towards data-driven techniques. Real-time data holds invaluable insights, mainly when a fault influences the system dynamics.

This thesis focuses on developing and validating data-driven fault identification of air data sensors. It has led to exciting research in blind system identification, specifically in rank and sparsity-constrained optimisation. This report documents the contributions to the field of data-driven fault diagnosis of sensors and provides avenues for further research.

I wish to convey gratitude to my thesis supervisor, Prof.Dr.Ir. Michel Verhaegen, for his guidance and contributions that have significantly elevated the quality of my research. I am profoundly thankful to Ir. Jacques Noom for his unwavering support and remarkable patience during all our interactions. Our discussions have allowed me to refine my perspective and gain clarity.

Additionally, I am grateful to Dr.Ir. Coen de Visser for his collaboration in the project, particularly for providing insights into fault diagnosis within the aerospace domain and supplying crucial datasets.

Delft, University of Technology — Nirupama Sai Ramesh
October 19, 2023

Glossary

List of Acronyms

ADS	Air Data Sensor
TAS	True Airspeed
AOA	Angle-Of-Attack
UAV	Unmanned Aerial Vehicle
FDI	Fault Detection and Isolation
FTC	Fault Tolerant Control
KF	Kalman Filter
MMAE	Multiple-Model Adaptive Estimation
DMAE	Double-Model Adaptive Estimation
BSI	Blind System Identification
SVD	Singular Value Decomposition
RISRO	Recursive Importance Sketching for Rank-Constrained Optimisation
BFI	Blind Fault Identification
FIR	Finite Impulse Response
VARX	Vector Auto-Regressive with eXogenous inputs
PM	Pseudo-Measurement

Chapter 1

Introduction

This preliminary chapter puts forth the main motivation and factors directing the chosen research trajectory. A comprehensive overview of the subsequent document structure is provided, followed by the introduction of the notational framework utilised throughout the remainder of this thesis.

1-1 Research background

Technological advancements in the aircraft industry are often shaped by catastrophic accidents that necessitate crucial safety improvements. A case in point is the Boeing 737 Max crash, attributed to inaccurately high readings from the Angle-Of-Attack (AOA) sensor, and the Airbus A330 accident linked to improper airspeed measurements caused by ice obstruction in the pitot tubes [6][5]. These incidents are currently propelling significant progress in the field of Air Data Sensor (ADS) fault detection, a sensor suite that provides True Airspeed (TAS) and AOA, among other measurements, to the aircraft (or Unmanned Aerial Vehicle (UAV)) control unit.

Fault Detection and Isolation (FDI) forms a crucial component of active Fault Tolerant Control (FTC) relying on accurate fault information for reconfiguring control strategies. FDI encompasses fault detection, isolation and estimation. Currently, aircraft FDI systems rely on the operational divergence between the aircraft and its mathematical model, also known as the digital twin, to detect and diagnose faults. However, dependence on digital twins severely stunts fault diagnosis capabilities due to the complexity of accurately representing such systems under dynamic operating conditions [1]. Conversely, data-driven fault diagnosis leans on pattern recognition, necessitating vast amounts of labelled data.

These motivations have brought forth the formulation of the research question:

Research Question

'Can Air Data Sensor(ADS) faults be identified using a computationally simple, model-free, data-driven approach based solely on input-output data and knowledge of the temporal behaviour of the fault without the need for extensive historical training data for specific operating conditions?'

In light of the shortcomings of model-based and knowledge-based fault diagnosis schemes, fault identification in the Blind System Identification (BSI) framework, requiring only input-output data and exploiting fault behaviour knowledge, shows promise as a fault identification tool [25].

Employing this framework for ADS fault identification forms the primary theme of this thesis. This report presents thorough validation results for the algorithm and emphasises its validity and effectiveness through a comparison with model-based fault diagnosis schemes.

1-2 Thesis Outline

The document is structured as follows.

Chapter 2 and Chapter 3 intend to introduce the necessary topics about the thesis. The modelling of ADS faults, along with an overview and limitation of model-based fault diagnosis schemes, are presented in Chapter 2. Chapter 3 lays the foundation for the proposed algorithm. It also introduces the concept of sparsity-constrained optimisation, facilitating its use as a fault diagnostic framework.

Chapter 4 elaborates on the Blind Fault Identification framework, deriving the mathematical formulation of the problem statement and presenting an algorithm for the same.

Chapter 5 presents the performance results with actual flight data. Algorithm validation results and comparison against benchmark FDI schemes are presented. Furthermore, a comprehensive description of the computation of fault start time, along with its limitations, is furnished through illustrative examples.

Finally, Chapter 6 summarises this thesis's conclusions and presents recommendations for future work, directing towards challenges that remain to be addressed.

1-3 Notations and Preliminaries

The following notations have been employed throughout. Scalars are denoted by uppercase and lowercase letters (e.g., C, a, b). Vectors are represented by bold lowercase letters or Greek symbols (e.g., $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}$). The boldface distinguishes sets of vectors, such as $\boldsymbol{x_1}, \boldsymbol{x_2}$ from the elements of a vector, $\boldsymbol{x} \in \mathbb{R}^n$, such as $x_1, x_2, .., x_n$. For a vector $\boldsymbol{v} \in \mathbb{R}^n$, $||\boldsymbol{v}||_q = (|v_1|^q + |v_2|^q + .. + |v_n|^q)^{\frac{1}{q}}$ denotes its l_q norm, for $1 \leq q \leq \infty$. Additionally, the l_0 norm is given as $||\boldsymbol{v}||_0 = \#\{j; v_j \neq 0\}$. For two vectors $\boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^n$, their inner product is denoted by $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{n} \boldsymbol{u_i} \boldsymbol{v_i}$.

The notations and operations on matrices are introduced next. Matrices are denoted by bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$). For a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\mathbf{X}_{ij}$ denotes the entry at $i-$th

row and $j-$th column with $\mathbf{X}_{i,:}$ and $\mathbf{X}_{:,j}$ denoting the $i-$th row and $j-$th column of $\mathbf{X}$ respectively. The transpose and inverse are represented as $\mathbf{X^T}$ and $\mathbf{X}^{-1}$ respectively. The Moore-Penrose inverse is denoted by $\mathbf{X}^{\dagger}$. The operation 'vec' transforms a matrix into a vector by vertically stacking its columns. $\mathbf{I}_r$ denotes the identity matrix of dimensions $r \times r$. The Frobenius norm of a matrix is denoted by $||\mathbf{X}||_F = \sqrt{\sum_{i=1}^{p} \sigma_i^2(\mathbf{X})}$, where $p = \min(m, n)$ and $\sigma_i(\mathbf{X})$ are the singular values of the matrix. The nuclear norm of a matrix is defined as $||\mathbf{X}||_* = \sum_{i=1}^{p} \sigma_i(\mathbf{X})$. The singular value decomposition decomposition is computed as $\sum_{i=1}^{p} \sigma_i(\mathbf{X}) \boldsymbol{u_i} \boldsymbol{v_i^T}$, with $\sigma_1(\mathbf{X}) \geq \sigma_2(\mathbf{X}) \geq .. \geq \sigma_p(\mathbf{X})$. The condition number of a matrix is computed as the ratio between the largest and smallest singular value, $\frac{\sigma_{max}}{\sigma_{min}}$.

The Kronecker product of two matrices, $\mathbf{A} \in \mathbb{R}^{m_1 \times m_2}$ and $\mathbf{B} \in \mathbb{R}^{n_1 \times n_2}$ is represented as $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{m_1 n_1 \times m_2 n_2}$.

Additional notations specific to the topic are introduced in their corresponding sections.

Chapter 2

Air Data Sensor Fault Modelling and Diagnosis

This chapter derives from the Literature Survey conducted as a preliminary to this thesis. It briefly presents the main concepts of fault diagnosis of Air Data Sensors. In particular, the modelling of the various fault modes that the sensor can encounter and its effect on the model are presented. An overview of the model-based diagnosis of ADS faults is presented, focusing on its limitations and challenges.

2-1 Fault Modelling

Faults in sensors manifest as additive or multiplicative deviations from the nominal parameter being measured. The temporal behaviour of the deviation characterises the nature of the fault. The Air Data Sensor (ADS) consists of a pitot-static tube that measures the differential pressure between the static port and the stagnation(pitot) inlet, as illustrated in Figure 2-1. In addition to the pitot inlet and the static pressure port, it also provides a drain hole to prevent moisture accumulation from the air stream entering the pitot. This differential pressure measurement is used to compute the True Airspeed (TAS), which gives the velocity of the aircraft (V_{TAS}).

The total pressure at the stagnation inlet, p_t and the pressure at the static port, p_s are related by the dynamic pressure, p_d as below:

$$p_d = p_t - p_s \ . \tag{2-1}$$

The physical relationship between the TAS, V_{TAS} and the pressure measurement is expressed using the compressible form of Bernoulli's equation [31] [14]:

$$V_{TAS} = \sqrt{\frac{2\gamma}{\gamma - 1} R_g T_s \left(\frac{p_d}{p_s} + 1 \right)^{\frac{\gamma - 1}{\gamma}} - 1}, \tag{2-2}$$

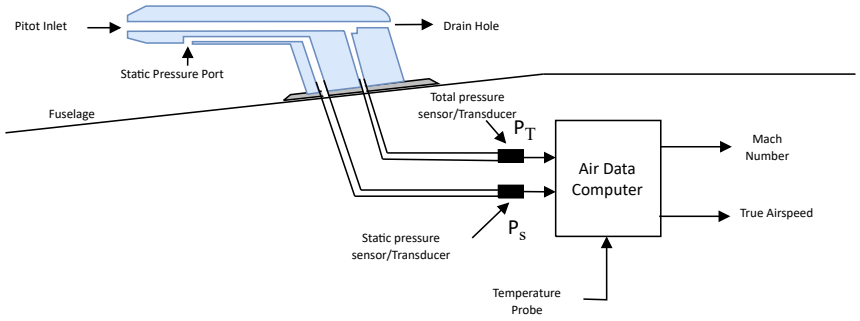


Figure 2-1: Schematic diagram of a Pitot Tube Air Data System

where, γ is the specific heat ratio of air, R_g is the ideal gas constant and T_s is the static temperature.

To comprehend the effect of pressure variations on the computed V_{TAS}, Equation 2-2 is reduced to obtain the expression of V_{TAS} solely in terms of the measured pressure.

Substituting the specific heat ratio of air, $\gamma = 1.4$, and the expression for the speed of sound, $A_o = \sqrt{R_g T_s \gamma}$ in Equation 2-2, the airspeed formula is expressed as

$$V_{TAS} = A_o \sqrt{5\left(\frac{p_d}{p_s} + 1\right)^{\frac{2}{7}} - 1}\,. \tag{2-3}$$

Let the air data equation be expressed as a function of total and static pressure,

$$f(p_s, p_t) = A_o \sqrt{5\left(\frac{p_t - p_s}{p_s} + 1\right)^{\frac{2}{7}} - 1}\,. \tag{2-4}$$

Equation 2-4 is linearised around specific trim conditions aiming to express the variation in the monitored variable, V_{TAS_m}, as an additive deviation. Let $\bar{p} = (\bar{p}_s, \bar{p}_t)$ be the trim conditions. Subsequently, anomalies introduced in the pressure measurements can be treated as deviations from the equilibrium(represented by Δ) and the resulting change in V_{TAS} can be expressed as an additive disturbance as follows [8]:

$$\begin{aligned} \Delta V_{TAS} &= \begin{bmatrix} a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} \Delta p_s \\ \Delta p_t \end{bmatrix}, \\ \text{where} \qquad a_{11} &= \left.\frac{\partial f(p_s, p_t)}{\partial p_s}\right|_{p_s=\bar{p}_s, p_t=\bar{p}_t}, \\ a_{12} &= \left.\frac{\partial f(p_s, p_t)}{\partial p_t}\right|_{p_s=\bar{p}_s, p_t=\bar{p}_t} \end{aligned} \tag{2-5}$$

and $\left.\frac{\partial f(p_s,p_t)}{\partial p_*}\right|_{\bar{p}}$ is the partial derivative with respect to p_*, evaluated at $\bar{p}$.

Thus,

Fault	Mathematical Representation
Abrupt blockage	Bias
Gradual blockage	Drift
Partial water blockage	Sinusoidal

Table 2-1: Mathematical Modelling of Pitot Probe fault scenarios

$$\begin{aligned} V_{TAS_m} &= V_{TAS} + \Delta V_{TAS}, \\ V_{TAS_m} &= V_{TAS} + \begin{bmatrix} a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} \Delta p_s \\ \Delta p_t \end{bmatrix}, \end{aligned} \tag{2-6}$$

where V_{TAS_m} is the faulty measurement and V_{TAS} is the nominal parameter.

2-2 Air Data Sensor Fault modes

The temporal dynamics of the perturbations observed in pressure measurements depend on the specific fault in the pitot tube. The pitot tube extending from the aircraft fuselage is vulnerable to blockage by insects, water, and, particularly, ice formation at high altitudes. Although commercial pitot tubes incorporate heating mechanisms to guarantee the proper functioning of the instrument, two main issues emerge. Firstly, there is a vulnerability to malfunctions, and secondly, the rapid heating risks diminishing the instrument's longevity [13].

Precise mathematical formulation and impact of heater faults on V_{TAS} remain unclear in the literature; hence, the scope of this study is limited to blockage faults. Experimental analysis of ice accretion on pitot tube inlet in tubes without drain holes shows that the gradual blocking of the pitot inlet leads to a drop in the measured total pressure. However, in the presence of drain holes, the area of the drain hole plays a significant role in the pressure decrease [30] [22]. The reader is directed to the *Literature Survey* for a detailed summary of the effects of blockage on each of the ports and their impact on the measured pressure variable [28].

Wind tunnel experiments have shown that increased noise, signal spikes, and false increase/decrease characterise blockages of pitot inlet, static port inlet and drain hole[30] [22] [8]. Additionally, the first comprehensive investigation of ice accretion effects in realistic atmospheric icing conditions confirmed an abrupt decrease in the airspeed [11]. Nonetheless, the scope of this thesis is confined to structured faults. The mathematical modelling of the considered faults is presented in Table 2-1 [10].

Rewriting Equation 2-4 in terms of the fault, $\boldsymbol{f(k)}$, at time instance, k, and the fault matrix specific to the linearisation, $\mathbf{F}$,

$$\begin{aligned} V_{TAS_m}(k) &= V_{TAS}(k) + \begin{bmatrix} a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} \Delta p_s(k) \\ \Delta p_t(k) \end{bmatrix}, \\ V_{TAS_m}(k) &= V_{TAS}(k) + \mathbf{F}\boldsymbol{f(k)} \\ \text{where} \qquad \mathbf{F} &= \begin{bmatrix} a_{11} & a_{12} \end{bmatrix}, \quad \boldsymbol{f(k)} = \begin{bmatrix} \Delta p_s(k) \\ \Delta p_t(k) \end{bmatrix}. \end{aligned} \tag{2-7}$$

The fault signals, $f(k)$, i.e. the perturbation in the pressure parameters, are expressed as the basis signals (or a scaled sum of thereof) as

$$\boldsymbol{f(k)} = \boldsymbol{\Theta(k)z}, \tag{2-8}$$

where $\boldsymbol{\Theta(k)} \in \mathbb{R}^{n_f \times n_z}$ is the dictionary of fault basis functions, mentioned in Table 2-1 and $\boldsymbol{z} \in \mathbb{R}^{n_z}$ contains the magnitude of each basis function.

2-3 Model-based Fault Diagnosis

The body of academic literature about the fault diagnosis of aircraft sensors is extensive. The majority of ADS fault diagnosis methods can be subsumed into model-based fault diagnosis. In recent times, Neural Network(NN)-based methods have gained traction, primarily because they prevent the need for an accurate mathematical model [15] [39]. However, it is worth noting that the methods rely on an NN model trained extensively on historical data at multiple operating points and can suffer from being system-specific.

Since NN-based methods are still developing, this discussion focuses on model-based techniques relying on established mathematical models.

Model-based fault diagnosis has evolved from the General Observer Scheme(GOS), wherein a fault is detected by evaluating the residual between the actual system measurement and predicted measurements. In such a scenario, the mathematical model of the system assumes a central role. The most commonly employed model is the linearised longitudinal model of the aircraft characterised by the states, TAS, V_{TAS}, Angle-Of-Attack (AOA), α, pitch angle, θ and rotational rate, q. The linearization of the aerodynamic equations of an aircraft around specific trim velocity and altitude provides the linear system of equations, represented as

$$\begin{bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{\theta} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} x_V & x_\alpha & x_\theta & 0 \\ z_V & z_\alpha & z_\theta & z_q \\ 0 & 0 & 0 & 1 \\ m_V & m_\alpha & m_\theta & m_q \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{\theta} \\ \dot{q} \end{bmatrix} + \begin{bmatrix} x_{\delta_e} & x_{\delta_t} \\ z_{\delta_e} & z_{\delta_t} \\ 0 & 0 \\ m_{\delta_e} & m_{\delta_t} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_{th} \end{bmatrix}, \tag{2-9}$$

where δ_e and δ_{th} are the system's elevator and engine thrust inputs. The system parameters denoted as $x_{V,\alpha,\theta}, z_{V,\alpha,\theta,q}, m_{V,\alpha,\theta,q}$ are known as the stability derivatives. Meanwhile, the input matrix contains the dimensionless control derivatives denoted by $x_{\delta_e,\delta_t}, z_{\delta_e,\delta_t}, m_{\delta_e,\delta_t}$.

These models are used in constructing observers, commonly the Kalman Filter (KF), or a modification such as the extended or unscented KF, to detect a deviation from the nominal performance. Fault diagnosis is accomplished by constructing 'elemental' filters, each specifically designed to model distinct faults. The overarching principle guiding this methodology asserts that the filter with the minimum residual error best captures the system dynamics and is thus deemed the system representative. This is known as the Multiple-Model Adaptive Estimation (MMAE)[7] [37] [23] [16].

However, MMAE suffers from being computationally complex as $K + 1$ filters need to run online parallelly to detect K distinct faults. The modelling of partial faults entails the addition of elemental filters to the existing set. This is combatted by the Double-Model Adaptive

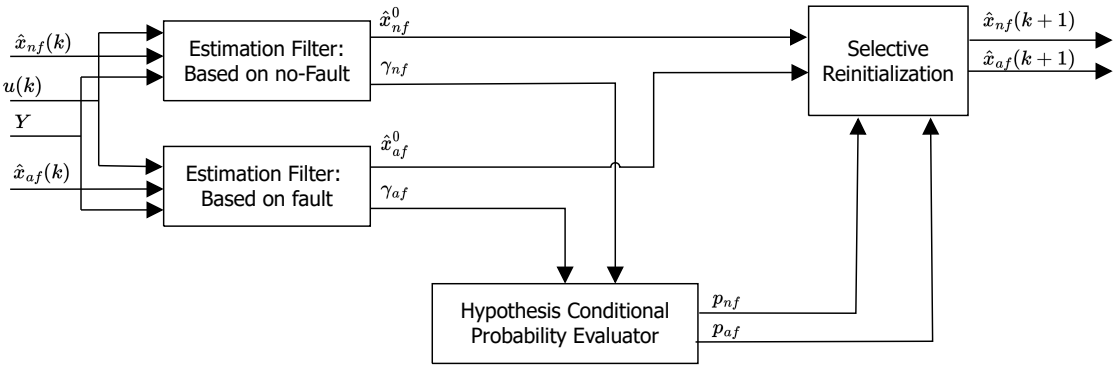


Figure 2-2: Double Model Adaptive Estimator Block Diagram [17] [18]

Estimation (DMAE), which uses only two filters, one for no-fault and the other incorporating all faults [17][20]. The states of each of the respective filters are given as follows:

$$\begin{aligned} \boldsymbol{x}_{nf}(k) &= \begin{bmatrix} x_1(k) & x_2(k), ..., x_{nx}(k) \end{bmatrix}^T, \\ \boldsymbol{x}_{af}(k) &= \begin{bmatrix} x_1(k) & x_2(k), ..., x_{nx}(k), f_1(k), f_1(k), ..., f_{n_f}(k) \end{bmatrix}^T, \end{aligned} \tag{2-10}$$

where the subscripts '$_{nf}$' and '$_{af}$' denote no-fault and augmented fault respectively. n_x represents the state dimension and n_f, the number of faults acting on the system. The estimates from the filters are processed through a hypothesis conditional probability module as shown in Figure 2-2.

The DMAE algorithm, Algorithm 1, requires the state-space model, $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{B} \in \mathbb{R}^{n_x \times n_u}$ for the Kalman Filter. While the no-fault filter uses the model as is, the augmented fault system extends these models to accommodate the fault estimation in the augmented fault filter.

The DMAE boasts lower false alarms than MMAE and unbiased estimates of faults and states. This attribute has positioned it as the established benchmark against which the outcomes of the proposed solution are juxtaposed in Chapter 5. Details about the benchmark code utilised for the comparison are also provided in Chapter 5.

2-4 Concluding Remarks

This chapter examined the principles and derivations behind the mathematical modelling of fault as additive disturbances on measured quantities. The temporal behaviour of deviations in the presence of blockage faults was reviewed and summarised. However, the literature is deficient in investigating the deviations introduced due to heater faults. Consequently, the main model-based fault diagnosis methods were explained, their shortcomings described, and the state-of-the-art DMAE was selected as the benchmark for evaluating fault diagnosis schemes proposed in this thesis.

Algorithm 1 Double Model Adaptive Estimator(DMAE) Pseudo Code

Inputs:

$\boldsymbol{u}(k) \in \mathbb{R}^{n_u},\ \boldsymbol{y}(k) \in \mathbb{R}^{n_y},\ \boldsymbol{A} \in \mathbb{R}^{n_x \times n_x},\ \boldsymbol{B} \in \mathbb{R}^{n_x \times n_u},\ \boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}$

Initialisation:

$\mathbf{Q}_k \in \mathbb{R}^{n_x \times n_x},\ \mathbf{R}_k \in \mathbb{R}^{n_y \times n_y},\ \boldsymbol{x}_{af}^0 \in \mathbb{R}^{n_x+n_f}$

$\mathbf{x}_{ff}^0 \in \mathbb{R}^{n_x},\ \mathbf{P}_{af}^0 \in \mathbb{R}^{(n_x+n_f)\times(n_x+n_f)},\ \mathbf{P}_{ff}^0 \in \mathbb{R}^{n_x \times n_x}$

for $k = 1, 2, ..., N$ **do**

  Update $\hat{\boldsymbol{x}}_{af}^k$,$\hat{\mathbf{P}}_{af}^k$ using fault-augmented KF

  Update $\hat{\boldsymbol{x}}_{ff}^k$,$\hat{\mathbf{P}}_{ff}^k$ using fault-free KF

  Compute conditional probabilities p_{af},p_{ff} using

$$\mathbf{Y}_{k-1} = \{\boldsymbol{y}(0), \boldsymbol{y}(1), ..., \boldsymbol{y}(k-1)\}$$

  Find index of the filter with maximal probability

$$\boldsymbol{i}_{max} \in \{\mathbf{P}_k = \max(p_{af}, p_{ff})\}$$

  Re-initialization of fault and no-fault filter

  if $i_{max} = 1$ **then**

    Reinitialise fault filter.

  else if $i_{max} = 2$ **then**

    Reinitialize fault-free filter.

  end if

end for

Chapter 3

Blind System Identification for Fault Diagnosis

This chapter lays the groundwork for the proposed algorithm by introducing the concept of Blind System Identification (BSI) and elaborating on its use as a fault diagnostic framework. Using a Finite Impulse Response (FIR) model, the problem of identifying model parameters and input signal is presented as a rank-constrained optimisation problem. A rank-constrained least squares framework based on algebraic sketching is reviewed, and the notion of sparsity constraints is discussed regarding determining the active fault.

The preceding chapter provided a concise overview of model-based fault diagnosis approaches, emphasising the critical role of precise mathematical models in these methods. Data-driven fault diagnosis methods essentially derive the model parameters from vast amounts of data from various operating conditions and use these models to diagnose faults. Identifying model parameters and diagnosing the fault simultaneously motivate the employment of BSI for fault diagnosis.

Where System Identification refers to the procedure of deducing the mathematical model of a system through an analysis of the correlation between the input and output of the system, BSI is the process of extracting the model parameters and the input signal from the output measurement alone. It is particularly sought in specific domains like seismic signal processing and celestial body imaging, where the input signal and system model are unknown.

A few areas of applications of BSI include but are not limited to, wireless communication, adaptive optics, speech recognition, and medical imaging [2] [33].

In the case of fault diagnosis, the fault is treated as an additional unknown input acting on the system output for sensor faults or input for actuator faults.

3-1 Formulation as Rank Constrained Optimisation Problem

The concept and formulation of BSI as a rank-constrained least squares problem is explained using a simplistic FIR model. Doing so provides the reader with fundamental principles upon

which the foundation of the fault diagnostic framework in Chapter 4 is constructed.

Consider the output of an FIR model of order p_1, characterised by unknown model parameters, $b_1, b_2, .., b_{p_1}$, for a single-input single-output process given by

$$\hat{y}(k) = (b_1 q^{-1} + b_2 q^{-2} + ...b_{p_1} q^{-p_1}) u(k), \tag{3-1}$$

where $\hat{y}(k) \in \mathbb{R}$ is the estimated output and $u(k) \in \mathbb{R}$ is the unknown input at time instance k. q^{-i} represents the shift parameter. Rewriting in terms of past inputs,

$$\hat{y}(k) = b_1 u(k-1) + b_2 u(k-2) + ... + b_{p_1} u(k-p_1). \tag{3-2}$$

Given a set of N observations, $\{y(k)\}_{k=1}^{N}$, the objective of BSI is to find the system parameters, $b_1, b_2, ..., b_{p_1}$, and the unknown input $u(k)$ from the output, $y(k)$. It is easy to see that without further assumptions, the problem is rather ill-posed. To this end, the input signal is assumed to be drawn from a known subspace of signals, called the data dictionary, composed of m basis signals, $\boldsymbol{\theta}(k) \in \mathbb{R}^m$ and expressed as

$$\begin{aligned} u(k) &= \underbrace{\begin{bmatrix} \theta_1(k) & \theta_2(k) & .. & \theta_m(k) \end{bmatrix}}_{\boldsymbol{\theta}(k)} \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ . \\ . \\ z_m \end{bmatrix}}_{\mathbf{z}}, \\ u(k) &= \boldsymbol{\theta}(k)\mathbf{z}. \end{aligned} \tag{3-3}$$

In Equation 3-3, the vector, $\boldsymbol{\theta}(k) \in \mathbb{R}^m$ represents the values of each of the m basis signals, at time instance k. $\boldsymbol{z} \in \mathbb{R}^m$ contains the magnitude of each basis signal and is unknown.

For simplicity of representation, let the model order $p_1 = 3$, i.e. the model parameters, represented by $\mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$ then Equation 3-2 can then be written in terms of the data dictionary and its corresponding magnitude as:

$$\hat{y}(k) = b_1 \boldsymbol{\theta}(k-1)\boldsymbol{z} + b_2 \boldsymbol{\theta}(k-2)\boldsymbol{z} + b_3 \boldsymbol{\theta}(k-3)\boldsymbol{z}, \tag{3-4}$$

$$\hat{y}(k) = \boldsymbol{\theta}(k-1) b_1 \boldsymbol{z} + \boldsymbol{\theta}(k-2) b_2 \boldsymbol{z} + \boldsymbol{\theta}(k-3) b_3 \boldsymbol{z}. \tag{3-5}$$

With the introduction of the Kronecker product between the model parameters, $\mathbf{B}$ and the input magnitudes, $\boldsymbol{z}$, denoted by $\mathbf{B} \otimes \boldsymbol{z} \in \mathbb{R}^{p_1 n_z}$, the observed output is expressed as

$$\hat{y}(k) = \begin{bmatrix} \boldsymbol{\theta}(k-1) & \boldsymbol{\theta}(k-2) & \boldsymbol{\theta}(k-3) \end{bmatrix} \mathbf{B} \otimes \boldsymbol{z}, \tag{3-6}$$

where $\boldsymbol{\theta}(k) \in \mathbb{R}^{n_z}$ and $\boldsymbol{z} \in \mathbb{R}^{n_z}$.

The data equation can now be expressed as:

$$\underbrace{\begin{bmatrix} \hat{y}(k) \\ \hat{y}(k+1) \\ \hat{y}(k+2) \\ . \\ . \\ . \\ \hat{y}(k+n) \end{bmatrix}}_{\hat{\mathbf{Y}}} = \underbrace{\begin{bmatrix} \boldsymbol{\theta}(k-1) & \boldsymbol{\theta}(k-2) & \boldsymbol{\theta}(k-3) \\ \boldsymbol{\theta}(k) & \boldsymbol{\theta}(k-1) & \boldsymbol{\theta}(k-2) \\ \boldsymbol{\theta}(k+1) & \boldsymbol{\theta}(k) & \boldsymbol{\theta}(k-1) \\ . & . & . \\ . & . & . \\ . & . & . \\ \boldsymbol{\theta}(k+n-1) & \boldsymbol{\theta}(k+n-2) & \boldsymbol{\theta}(k+n-3) \end{bmatrix}}_{\mathbf{T}_{\boldsymbol{\theta}}} \mathbf{B} \otimes \mathbf{z}, \tag{3-7}$$

$$\hat{\mathbf{Y}} = \mathbf{T}_{\theta}\mathbf{B} \otimes \boldsymbol{z}. \tag{3-8}$$

The estimation of $\mathbf{B}$ and $\boldsymbol{z}$ gives rise to the bilinear least squares problem,

$$\min_{\mathbf{B},\mathbf{z}} ||\mathbf{Y} - \mathbf{T}_{\theta}(\mathbf{B} \otimes \mathbf{z})||_2^2. \tag{3-9}$$

The bilinearity is characterised by the presence of the Kronecker product of the optimisation variables, $\mathbf{B}$ and $\boldsymbol{z}$. However, it is apparent that the vector product of $\mathbf{B}$ and $\boldsymbol{z}^T$ is rank deficient by construction,

$$\text{rank}\left(\begin{bmatrix} b_1 z_1 & b_1 z_2 & ... & b_1 z_m \\ b_2 z_1 & b_2 z_2 & ... & b_2 z_m \\ b_3 z_1 & b_3 z_2 & ... & b_3 z_m \end{bmatrix} \right) = 1. \tag{3-10}$$

Therefore, the bilinear least squares optimisation problem in Equation 3-9 can be recast as a rank-constrained least squares problem as in Equation 3-11,

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{z}} ||\mathbf{Y} - \mathbf{T}_{\theta}(\mathbf{B} \otimes \mathbf{z})||_2^2 \\ &\text{s.t. } \text{rank}(\mathbf{B}\mathbf{z}^T) = 1. \end{aligned} \tag{3-11}$$

The blind identification of models has been extended to state-space systems with a variation of objective functions that are non-convex [35]. However, the solutions can often get stuck in local minima owing to the non-convex nature of the optimisation problem [12] [32].

The cost function Equation 3-11 is convex; however, the rank constraint introduces non-convexity to the optimisation problem. An $m \times n$ matrix is characterised by its column and row space. A frequently utilized convex relaxation technique for the rank constraint involves computing the nuclear norm of a variable. Minimizing the nuclear norm promotes rank deficiency in the variable while concurrently ensuring the satisfaction of the objective function and adherence to other constraints [12] [29].

Advances in the area of low-rank approximation include the imposition of rank constraints by sketching the optimisation variable to low-rank subspaces [34] [21]. An initial research investigation aimed at assessing the applicability of Recursive Importance Sketching for Rank-Constrained Optimisation (RISRO) for BSI, in comparison to the convex relaxation strategy of nuclear norm minimisation, confirmed the superior performance of RISRO [27]. This superiority is evident in terms of estimation accuracy and computational execution time.

This superiority is evident in terms of estimation accuracy and computational execution time. It is noteworthy that convergence to the true solution is assured as long as the initialisation falls within a specific threshold of the true value and the regressor adheres to the Restricted Isometry Property (RIP) [27] [21]. Notably, the convergence towards the true solution is significantly influenced by the condition number of the regressor, perhaps due to the reliance on a linear least squares solution. A brief explanation of RISRO is presented here as it is the choice of a rank-constrained optimisation framework for the fault identification framework presented subsequently in Chapter 4.

RISRO: Solution of Rank-Constrained Least Square Problems

Recursive Importance Sketching for Rank-Constrained Optimisation (RISRO) is a technique to solve rank-constrained optimisation problems by sketching the regressor to lower-rank subspaces of the optimisation variable and performing a linear least squares regression on the updated regressor and regressand. Its core sketching techniques have been derived from ISLET: Importance Sketching for Low-Rank Estimation for Tensors [34]. The sketched matrices are updated recursively to satisfy the rank constraint and the objective function.

This section elucidates the application of RISRO by considering the following optimisation problem:

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^{p_1\times p_2}} \; & ||\boldsymbol{y}-\boldsymbol{\mathcal{A}}(\mathbf{X})||_2^2, \\ \text{s.t. } & \operatorname{rank}(\mathbf{X}) = r \end{aligned} \tag{3-12}$$

where $\boldsymbol{y}\in\mathbb{R}^n$ is the measured data and $\boldsymbol{\mathcal{A}}\in\mathbb{R}^{p_1\times p_2}$ is a linear mapping given as

$$\boldsymbol{\mathcal{A}}(\mathbf{X}) = [\langle \mathbf{A}_1, \mathbf{X}\rangle, ..., \langle \mathbf{A}_n, \mathbf{X}\rangle]^T, \qquad \mathbf{A}_i \in \mathbb{R}^{p_1\times p_2}, \qquad i = 1, 2, ..., n$$

with the inner product, $\langle \mathbf{A}_i, \mathbf{X}\rangle = \sum_{1\le j\le p_1, 1\le k\le p_2} \mathbf{A}_{i_{j,k}} \mathbf{X}_{j,k}$.

The algorithm admits an initial estimate, $\mathbf{X}^0 \in \mathbb{R}^{p_1\times p_2}$ with the singular value decomposition(SVD) defined as $\mathbf{X} = \mathbf{U}^0\boldsymbol{\Sigma}^0\mathbf{V}^{0T}$, with $\mathbf{U}^0 \in \mathbb{R}^{p_1\times r}, \boldsymbol{\Sigma}^0 \in \mathbb{R}^{r\times r}, \mathbf{V}^0 \in \mathbb{R}^{p_2\times r}$.

Step 1: Importance Sketching
The singular subspace approximations of $\mathbf{X}$ are characterised by the subspaces spanned by $\mathbf{U}^0$ and $\mathbf{V}^0$ and their orthonormal complements, $\mathbf{U}^0_\perp$ and $\mathbf{V}^0_\perp$, onto which each $\mathbf{A}_i$ is sketched as

$$\begin{aligned} (\boldsymbol{\mathcal{A}}_B)_i &= \mathbf{U}^{tT}\mathbf{A}_i\mathbf{V}^t, \\ (\boldsymbol{\mathcal{A}}_{D_1})_i &= \mathbf{U}^{tT}_\perp\mathbf{A}_i\mathbf{V}^t, \\ (\boldsymbol{\mathcal{A}}_{D_2})_i &= \mathbf{U}^{tT}\mathbf{A}_i\mathbf{V}^t_\perp, \end{aligned} \tag{3-13}$$

where $\boldsymbol{\mathcal{A}}_B : \mathbb{R}^{r\times r} \to \mathbb{R}^n$, $\boldsymbol{\mathcal{A}}_{D_1} : \mathbb{R}^{(p_1-r)\times r} \to \mathbb{R}^n$ and $\boldsymbol{\mathcal{A}}_{D_2} : \mathbb{R}^{r\times(p_2-r)} \to \mathbb{R}^n$.

Step 2: Dimension-reduced Least Squares
Next, the dimension-reduced unconstrained least squares problem, expressed below, is solved:

$$\underset{\mathbf{B}\in\mathbb{R}^{r\times r}, \mathbf{D}_i\in\mathbb{R}^{(p_i-r)\times r}, i=1,2}{\arg\min} \; ||\boldsymbol{y} - \boldsymbol{\mathcal{A}}_B(\mathbf{B}) - \boldsymbol{\mathcal{A}}_{D_1}(\mathbf{D}_1) - \boldsymbol{\mathcal{A}}_{D_2}(\mathbf{D}_2^T)||_2^2 \tag{3-14}$$

Step 3: Inverse Projection and Update of sketching matrices
From the computed $\boldsymbol{\gamma}$ in Step 2, $\hat{\mathbf{X}}$ has to be estimated by inverting the projection carried out in Step 1, similar to Equation 2.4 of ISLET [34]. The estimate is retrieved using the following formula:

$$\begin{aligned} \mathbf{X}_U^{t+1} &= \mathbf{U}^t\mathbf{B} + \mathbf{U}_\perp^t\mathbf{D}_1, \\ \mathbf{X}_V^{t+1} &= \mathbf{V}^t\mathbf{B}^T + \mathbf{V}_\perp^t\mathbf{D}_2, \\ \hat{\mathbf{X}}^{t+1} &= \mathbf{X}_U^{t+1}(\mathbf{B})^\dagger\mathbf{X}_V^{t+1}, \end{aligned} \tag{3-15}$$

The sketching projection matrices $\mathbf{U}^{t+1}$ and $\mathbf{V}^{t+1}$ are obtained by QR orthogonalization of $\mathbf{X}_U^{t+1}$ and $\mathbf{X}_V^{t+1}$.

$$\begin{aligned} \mathbf{U}^{t+1} &= QR(\mathbf{X}_U^{t+1}), \\ \mathbf{V}^{t+1} &= QR(\mathbf{X}_V^{t+1}). \end{aligned} \tag{3-16}$$

The steps are repeated for each iteration for n iterations or till a satisfactory error value is reached as defined by the user.

3-2 Blind System Identification for Fault Diagnosis

In the context of fault identification, the objective of BSI is to identify the presence of faults and construct the model concurrently. Blind deconvolution methods have been extensively studied and applied to diagnose faults in rotating machinery [24]. The accurate representation using transfer function for complex rotating machinery in industrial settings poses a formidable challenge. Consequently, the process of extracting fault-related features from measured signals without a transfer function is called Blind Deconvolution. In the same vein, albeit independently, and more recently, the blind identification of actuator faults has been developed in a rank and sparsity-constrained framework [25]. It employs a state-space Vector Auto-Regressive with eXogenous inputs (VARX) representation of inputs and outputs alone. The objective function in the formulated optimisation problem seeks to minimise the Frobenius norm of the error between the predicted and available measurements while enforcing rank and sparsity conditions on the optimisation variable.

Sparsity is utilised to restrict the concurrent activation of multiple faults. The data dictionary is structured to represent various potential fault scenarios. Nevertheless, assuming that all faults could be simultaneously active is not practical. Achieving precise isolation of solely active faults necessitates the imposition of a sparsity constraint, which forces the majority of entries of the fault magnitude vector to be zero [36].

Sparse Estimation using Constrained Kalman Filter

The concept of sparse signal representation originated within Compressed Sensing(CS). This signal-processing technique leverages the sparsity inherent in signals to efficiently capture and reconstruct them using a limited set of measurements or samples. The same algorithms used for sparse representation of signals can be utilised for sparse estimation [36].

Sparsity is generally studied from the viewpoint of a norm minimisation scheme. Typically, enforcing sparsity translates to l_0, $l_p(0 \leq p \leq 1)$ or l_1 norm minimisation [38]. The p-norm of a vector, $\boldsymbol{v} = \begin{bmatrix} v_1 & v_2 & ... & v_n \end{bmatrix}$, for $1 \leq p \leq \infty$, is computed as

$$||\boldsymbol{v}||_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}}. \tag{3-17}$$

The sparsity of a vector is related to its l_0-norm as it gives the number of non-zero elements in the vector, computed as in Equation 3-17, with $\lim_{p \to 0}$,

$$||\boldsymbol{v}||_0 = \lim_{p \to 0} \sum_{i=1}^{n} |v_i|^p. \tag{3-18}$$

However, the l_0-norm is non-convex; hence, the convex non-smooth minimisation of the $l_1$norm is preferred.

The Kalman Filter (KF) is a simple yet powerful estimation algorithm which can be extended to the estimation of sparse signals for dynamic compressed sensing utilising a Pseudo-Measurement (PM) technique [4].

To understand the PM technique, let us consider the following optimisation problem:

$$\begin{aligned} \min_{\hat{\boldsymbol{z}}(k)} \mathbb{E}\big[||\boldsymbol{z}(k) - \hat{\boldsymbol{z}}(k)||_2^2\big] \\ \text{s.t.} ||\hat{\boldsymbol{z}}(k)||_1 \leq \boldsymbol{\epsilon}' \end{aligned} \tag{3-19}$$

where $\mathbb{E}[.]$ represents the expectation operator, $\hat{\boldsymbol{z}}(k) \in \mathbb{R}^n$ is the vector to be estimated at time instance k, and $\boldsymbol{\epsilon}$ is the noise. The unconstrained optimisation problem can be solved using the traditional KF. To deal with the inequality constraint, the PM technique considers a fictitious measurement and translates the inequality constraint to

$$\begin{aligned} 0 &= \bar{\mathbf{H}}\boldsymbol{z}(k) - \boldsymbol{\epsilon}', \\ \text{where } \bar{\mathbf{H}} &= \begin{bmatrix} \text{sign}(z_1(k)) & \text{sign}(z_2(k)) & ... & \text{sign}(z_n(k)) \end{bmatrix}. \end{aligned} \tag{3-20}$$

The newly defined state-dependent $\bar{\mathbf{H}}$ is used in the PM stage in a KF framework as explained in Algorithm 2.

Algorithm 2 Pseudo-Measurement Stage

Initialisation:

$\mathbf{P}^1 \in \mathbb{R}^{n \times n}$, $\hat{\boldsymbol{z}}^1 \in \mathbb{R}^n$, $\mathbf{R} \in \mathbb{R}$

1: **for** $m = 0, 1, ..., M$ **do**

2: $\quad \bar{\mathbf{H}^m} = \begin{bmatrix} \text{sign}(z_1(k)) & \text{sign}(z_2(k)) & ... & \text{sign}(z_n(k)) \end{bmatrix}$

3: $\quad \mathbf{K}^m = \mathbf{P}^m \bar{\mathbf{H}^m} (\bar{\mathbf{H}^m} \mathbf{P}^m \bar{\mathbf{H}^m}^T + \mathbf{R})^{-1}$

4: $\quad \hat{\boldsymbol{z}}^{m+1}(k) = (\mathbf{I} - \mathbf{K}^m \bar{\mathbf{H}^m}) \hat{\boldsymbol{z}}^m(k)$

5: $\quad \mathbf{P}^{m+1} = (\mathbf{I} - \mathbf{K}^m \bar{\mathbf{H}^m}) \mathbf{P}^m$

6: **end for**

The estimation is carried out using the traditional unconstrained KF framework wherein the state covariance matrix, $\mathbf{P}(k+1|k+1)$ and the updated state, $\hat{\boldsymbol{z}}(k+1|k+1)$ are obtained. These, along with the tunable parameter, $\mathbf{R}$, form the inputs to the PM Stage. The PM stage builds on the KF framework and drives the elements of the state estimate to zero to enforce sparsity.

In place of l_1-norm minimisation, the PM technique can also accommodate the quasi-norm, i.e. l_p-norm$(0 \leq p \leq 1)$ and approximating the l_0-norm using an exponential function. The l_0-norm can be approximated as [4]

$$||\boldsymbol{z}(k)||_0 = n - \sum_{i=1}^{n} \exp(-\alpha |z_j(k)|), \tag{3-21}$$

where α is a tunable parameter.

In this case, the steps 2 and 4 in the PM stage in Algorithm 2 are modified as below:

$$\begin{aligned} \bar{\mathbf{H}}_j^m &= \begin{cases} -\alpha \exp(-\alpha z_j^m, & \text{if } \ z_j^m > 0 \\ \alpha \exp(\alpha z_j^m, & \text{if } \ z_j^m \leq 0 \end{cases}, \\ \hat{\boldsymbol{z}}^{m+1} &= \hat{\boldsymbol{z}}^m - \mathbf{K}^m \Big[n - \sum_{j=1}^{n} \exp(-\alpha |z_j^m|) \Big]. \end{aligned} \tag{3-22}$$

where $j = 1, 2, ..n$ represents the index of the state $\hat{\boldsymbol{z}}^m$. $\bar{\mathbf{H}}_j$ is the j^{th} element of the fictiotious measurement matrix on the m^{th} iteration of the PM stage. α is a tunable parameter.

3-3 Chapter Summary

This chapter presents concepts relevant to applying BSI as a framework for data-driven fault diagnosis. The formulation of BSI as a rank-constrained least squares problem is presented for the case of an FIR model. Notably, incorporating the rank constraint introduces non-convexity into the optimisation problem. A recursive sketching-based algorithm tailored for rank-constrained least squares optimisation is reviewed, demonstrating its suitability for solving the BSI rank-constrained problem.

The essence of applying BSI for fault diagnosis hinges on constructing an exhaustive fault data dictionary that encompasses all possible fault scenarios. The optimisation algorithm then identifies the active fault and estimates its magnitude. In this context, the imposition of sparsity constraints on the magnitude vector emerges as a crucial consideration, allowing only select faults to be concurrently active, mirroring real-world fault scenarios. Sparse estimation within a constrained KF framework leveraging norm minimisation techniques is reviewed in that regard.

This chapter presents specific concepts and algorithms, forming the foundation for the comprehensive data-driven fault diagnosis within the BSI framework.

Chapter 4

Additive Sensor Fault Diagnosis for Linear Stochastic Systems

In this chapter, we introduce the primary innovations of this thesis, which include a novel approach for identifying faults in Air Data Sensor (ADS). This method eliminates the requirement for precise system information and prior training data. The task of fault identification is framed as an optimisation problem characterised by constraints related to rank and sparsity. The rank-constrained optimisation is achieved using Recursive Importance Sketching for Rank-Constrained Optimisation (RISRO), and the sparsity constraint is satisfied by incorporating an exponential approximation of l_0-norm.

4-1 Air Data Sensor Blind Fault Identification

Consider a discrete linear stochastic system representing the longitudinal dynamics of the aircraft given as

$$\begin{aligned} \boldsymbol{x}(k+1) &= \mathbf{A}(\mu)\boldsymbol{x}(k) + \mathbf{B}(\mu)\boldsymbol{u}(k) + \boldsymbol{w}_1(k)), \\ \boldsymbol{y}(k) &= \mathbf{C}(\mu)\boldsymbol{x}(k) + \boldsymbol{w}_2(k)), \\ \boldsymbol{y}_m(k) &= \boldsymbol{y}(k) + \boldsymbol{f}(k), \end{aligned} \tag{4-1}$$

where $\mathbf{A}(\mu)$, $\mathbf{B}(\mu)$ and $\mathbf{C}(\mu)$ represent the state, input and output matrices of an aircraft as a function of aerodynamic parameters. The process noise $\boldsymbol{w}_1(k)$ and measurement noise $\boldsymbol{w}_2(k)$ are statistically independent, with zero mean white noise. The true output, $\boldsymbol{y}(k)$ is perturbed by a fault, $\boldsymbol{f}(k)$ forming the measured output available from the sensor, $\boldsymbol{y}_m(k)$.

Given the pair $(\mathbf{A}(\mu), \mathbf{C}(\mu))$ is observable, the state-space observer model with observer gain $\mathbf{K}$, is represented as

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= \mathbf{A}(\mu)\hat{\boldsymbol{x}}(k) + \mathbf{B}(\mu)\boldsymbol{u}(k) + \mathbf{K}(\mu)(\boldsymbol{y}_m(k) - \hat{\boldsymbol{y}}(k)), \\ \hat{\boldsymbol{y}}(k) &= \mathbf{C}(\mu)\hat{\boldsymbol{x}}(k), \end{aligned} \tag{4-2}$$

The state vector $\hat{\boldsymbol{x}}(k) \in \mathbb{R}^{n_x}$ provides information on the longitudinal states of the system, and the observer output is represented as $\hat{\boldsymbol{y}}(k) \in \mathbb{R}^{n_y}$. Elevator deflection, δ_e is the input to the system, $\boldsymbol{u}(k)$. Let the ADS measurements of True Airspeed (TAS), V_{TAS_m} and Angle-Of-Attack (AOA), α_m, be represented as $\boldsymbol{y}_m(k)$. They are contaminated by an additive disturbance, acting on the nominal parameter, and the measured variable at a time instance, k, is expressed as

$$V_{TAS_m}(k) = V_{TAS}(k) + \underbrace{\begin{bmatrix} a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} \Delta p_1(k) \\ \Delta p_2(k) \end{bmatrix}}_{\boldsymbol{d}(k)}, \tag{4-3}$$

where a_{11} and a_{12} are constants and $\Delta p_1(k)$ and $\Delta p_2(k)$ are the deviations in pressure measurements of the ADS. The fault is expressed as an unknown disturbance, $\boldsymbol{d}(k)$. In the case of the longitudinal aircraft model, the effect of the disturbance, $\boldsymbol{d}(k) \in \mathbb{R}^{n_d}$, on the measured variables are determined by a fault distribution matrix, $\mathbf{F} \in \mathbb{R}^{n_y \times n_d}$, and the relation is given by

$$\begin{bmatrix} V_{TAS_m}(k) \\ \alpha_m(k) \end{bmatrix} = \begin{bmatrix} V_{TAS} \\ \alpha \end{bmatrix} + \underbrace{\begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}}_{\mathbf{F}} \boldsymbol{d}(k). \tag{4-4}$$

The fault distribution matrix, along with the disturbance, together express the fault in the variables,

$$\boldsymbol{f}(k) = \mathbf{F}\boldsymbol{d}(k). \tag{4-5}$$

Employing the Blind System Identification (BSI) framework for the identification of these unknown disturbances necessitates an assumption on the subspace in which the fault sequences exist [25] [29] [32].

Consequently, they are characterized as a scaled sum of basis functions derived from a fault data dictionary,

$$\boldsymbol{d}(k) = \boldsymbol{\theta}(k)\boldsymbol{z}(k), \tag{4-6}$$

where $\boldsymbol{\theta}(k) \in \mathbb{R}^{n_d \times n_z}$ represents the fault data dictionary and $\boldsymbol{z} \in \mathbb{R}^{n_z}$, the corresponding magnitudes. Let the state-space matrices $\mathbf{A}(\mu)$, $\mathbf{B}(\mu)$ and $\mathbf{C}(\mu)$ be denoted by $\mathbf{A}$,$\mathbf{B}$ and $\mathbf{C}$, respectively, for concision in representation.

Incorporating the effect of the additive fault in the observer output, $\hat{\boldsymbol{y}}(k)$, to model the deviation from nominal behaviour in $\boldsymbol{y_m}(k)$,

$$\hat{\boldsymbol{x}}(k+1) = \mathbf{A}\hat{\boldsymbol{x}}(k) + \mathbf{B}\boldsymbol{u}(k) + \mathbf{K}(\boldsymbol{y}_m(k) - \hat{\boldsymbol{y}}(k)), \tag{4-7}$$

$$\hat{\boldsymbol{y}}(k) = \mathbf{C}\hat{\boldsymbol{x}}(k) + \mathbf{F}\boldsymbol{d}(k). \tag{4-8}$$

The output equation, Equation 4-8 can be expressed in terms of the previous state by replacing $\hat{\boldsymbol{x}}(k)$ by the corresponding state equation, Equation 4-7 at time step k, yielding

$$\begin{aligned} \hat{\boldsymbol{y}}(k) &= \mathbf{C}(\mathbf{A}\hat{\boldsymbol{x}}(k-1) + \mathbf{B}\boldsymbol{u}(k-1) + \mathbf{K}(\boldsymbol{y}_m(k-1) - \hat{\boldsymbol{y}}(k-1))) + \mathbf{F}\boldsymbol{d}(k), \\ \hat{\boldsymbol{y}}(k) &= \mathbf{C}((\mathbf{A} - \mathbf{KC})\hat{\boldsymbol{x}}(k-1) + \mathbf{B}\boldsymbol{u}(k-1) + \mathbf{K}\boldsymbol{y}_m(k-1)) + \mathbf{F}\boldsymbol{d}(k). \end{aligned} \tag{4-9}$$

Extending the output equation and writing it in terms of past s time steps, the following expression is obtained

$$\begin{aligned}\hat{\boldsymbol{y}}(k) = \mathbf{C}(\mathbf{A}-\mathbf{KC})^s\hat{\boldsymbol{x}}(k-s) + \sum_{i=1}^{s}\mathbf{C}(\mathbf{A}-\mathbf{KC})^{i-1}\mathbf{B}\boldsymbol{u}(k-i) + \\ \sum_{i=1}^{s}\mathbf{C}(\mathbf{A}-\mathbf{KC})^{i-1}\mathbf{K}\boldsymbol{y}_m(k-i) + \sum_{i=1}^{s}\mathbf{C}(\mathbf{A}-\mathbf{KC})^{i-1}\mathbf{KF}\boldsymbol{d}(k-i) + \mathbf{F}\boldsymbol{d}(k).\end{aligned} \tag{4-10}$$

For $s \geq n_x$, the state estimate error will converge to zero if $\mathbf{A}-\mathbf{KC}$ is asymptotically stable and $\mathbf{K}$ is a deadbeat observer gain. Consequently, the current output in Equation 4-10 can be written only in terms of its past inputs and outputs as [26]

$$\begin{aligned}\hat{\boldsymbol{y}}(k) \approx \sum_{i=1}^{s}\mathbf{C}(\mathbf{A}-\mathbf{KC})^{i-1}\mathbf{B}\boldsymbol{u}(k-i) + \sum_{i=1}^{s}\mathbf{C}(\mathbf{A}-\mathbf{KC})^{i-1}\mathbf{K}\boldsymbol{y}_m(k-i) \\ + \sum_{i=1}^{s}\mathbf{C}(\mathbf{A}-\mathbf{KC})^{i-1}\mathbf{KF}\boldsymbol{d}(k-i) + \mathbf{F}\boldsymbol{d}(k).\end{aligned} \tag{4-11}$$

Representing $\mathbf{C}(\mathbf{A}-\mathbf{KC})^{i-1}\mathbf{B} = \boldsymbol{\mathcal{B}}_i$ and $\mathbf{C}(\mathbf{A}-\mathbf{KC})^{i-1}\mathbf{K} = \boldsymbol{\mathcal{K}}_i$, the output equation is written as:

$$\hat{\boldsymbol{y}}(k) \approx \sum_{i=1}^{s}\boldsymbol{\mathcal{B}}_i\boldsymbol{u}(k-i) + \sum_{i=1}^{s}\boldsymbol{\mathcal{K}}_i\boldsymbol{y}_m(k-i) + \sum_{i=1}^{s}\boldsymbol{\mathcal{K}}_i\mathbf{F}\boldsymbol{d}(k-i) + \mathbf{F}\boldsymbol{d}(k). \tag{4-12}$$

Expressing the disturbance as derived from the fault dictionary and segregating the knowns from the unknowns, the fault signal can be rewritten as

$$\boldsymbol{f}(k) = (\mathbf{F}\otimes\boldsymbol{z}^T)\mathrm{vec}(\boldsymbol{\theta}^T(k)). \tag{4-13}$$

For ease of representation, the following notation will be followed:

$$\mathbf{F}(\boldsymbol{z}) = \mathbf{F}\otimes\boldsymbol{z}^T. \tag{4-14}$$

Substituting Equation 4-13 in Equation 4-12,

$$\hat{\boldsymbol{y}}(k) \approx \sum_{i=1}^{s}\boldsymbol{\mathcal{B}}_i\boldsymbol{u}(k-i) + \sum_{i=1}^{s}\boldsymbol{\mathcal{K}}_i\boldsymbol{y}_m(k-i) + \sum_{i=1}^{s}\boldsymbol{\mathcal{K}}_i\mathbf{F}(\boldsymbol{z})\mathrm{vec}(\boldsymbol{\theta}^T(k-i)) + \mathbf{F}(\boldsymbol{z})\mathrm{vec}(\boldsymbol{\theta}^T(k)). \tag{4-15}$$

Taking $\boldsymbol{\mathcal{M}}_i = \boldsymbol{\mathcal{K}}_i\mathbf{F}(\boldsymbol{z})$,

$$\hat{\boldsymbol{y}}(k) \approx \sum_{i=1}^{s}\boldsymbol{\mathcal{B}}_i\boldsymbol{u}(k-i) + \sum_{i=1}^{s}\boldsymbol{\mathcal{K}}_i\boldsymbol{y}_m(k-i) + \sum_{i=1}^{s}\boldsymbol{\mathcal{M}}_i\mathrm{vec}(\boldsymbol{\theta}^T(k-i)) + \mathbf{F}(\boldsymbol{z})\mathrm{vec}(\boldsymbol{\theta}^T(k)). \tag{4-16}$$

Given N observations, $\boldsymbol{y}(k)_{k=1}^{N}$, the past inputs, outputs and the data dictionary signals can be written in terms of their Toepltiz matrices, $\mathbf{T}_u$, $\mathbf{T}_{y_m}$ and $\mathbf{T}_\theta$ respectively. The measurements and fault dictionary at the current time step are arranged as

$$\mathbf{Y}_m = \begin{bmatrix}\boldsymbol{y}_m^T(k+s)\\ \boldsymbol{y}_m^T(k+s+1)\\ \vdots\\ \boldsymbol{y}_m^T(k+N)\end{bmatrix} \text{ and } \quad \boldsymbol{\Theta} = \begin{bmatrix}\mathrm{vec}(\boldsymbol{\theta}^T(k+s))^T\\ \mathrm{vec}(\boldsymbol{\theta}^T(k+s+1))^T\\ \vdots\\ \mathrm{vec}(\boldsymbol{\theta}^T(k+s+1))^T\end{bmatrix}, \tag{4-17}$$

and the unknown parameters $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\mathcal{M}}$ defined respectively as

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \boldsymbol{\mathcal{B}}_1^T \\ \boldsymbol{\mathcal{B}}_2^T \\ \vdots \\ \boldsymbol{\mathcal{B}}_s^T \end{bmatrix}, \quad \boldsymbol{\mathcal{K}} = \begin{bmatrix} \boldsymbol{\mathcal{K}}_1^T \\ \boldsymbol{\mathcal{K}}_2^T \\ \vdots \\ \boldsymbol{\mathcal{K}}_s^T \end{bmatrix}, \text{ and } \boldsymbol{\mathcal{M}} = \begin{bmatrix} \boldsymbol{\mathcal{M}}_1^T \\ \boldsymbol{\mathcal{M}}_2^T \\ \vdots \\ \boldsymbol{\mathcal{M}}_s^T \end{bmatrix}; \tag{4-18}$$

the output observer equation is given by

$$\hat{\mathbf{Y}} \approx \begin{bmatrix} \mathbf{T}_u & \mathbf{T}_{y_m} & \mathbf{T}_\theta & \boldsymbol{\Theta} \end{bmatrix} \begin{bmatrix} \boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{K}} \\ \boldsymbol{\mathcal{M}} \\ (\mathbf{F}(\boldsymbol{z}))^T \end{bmatrix}. \tag{4-19}$$

The objective is to find system parameters $\boldsymbol{\mathcal{B}}$,$\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\mathcal{M}}$, and fault magnitudes, $\mathbf{F} \otimes \boldsymbol{z}^T$ such that the error between the observer output, $\hat{\mathbf{Y}}$, and the measured output, $\mathbf{Y}_m$ is minimised.

Mathematically, it is expressed using the following optimisation scheme,

$$\begin{aligned} \min_{\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{K}},\boldsymbol{\mathcal{M}},\mathbf{F}\otimes\boldsymbol{z}^T} & \left\| \mathbf{Y}_m - \begin{bmatrix} \mathbf{T}_u & \mathbf{T}_{y_m} & \mathbf{T}_\theta & \boldsymbol{\Theta} \end{bmatrix} \begin{bmatrix} \boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{K}} \\ \boldsymbol{\mathcal{M}} \\ (\mathbf{F}(\boldsymbol{z}))^T \end{bmatrix} \right\|_2^2 \\ s.t. \quad & \operatorname{rank}(\operatorname{vec}(\mathbf{F}^T)\boldsymbol{z}^T) = 1, \\ & \operatorname{rank}\left(\begin{bmatrix} \boldsymbol{\mathcal{M}}_1 & \boldsymbol{\mathcal{K}}_1 \\ \boldsymbol{\mathcal{M}}_2 & \boldsymbol{\mathcal{K}}_2 \\ \vdots & \vdots \\ \boldsymbol{\mathcal{M}}_s & \boldsymbol{\mathcal{K}}_s \\ (\mathbf{F}(\boldsymbol{z}))^T & I \end{bmatrix} \right) = n_y. \end{aligned} \tag{4-20}$$

Additionally, a sparsity constraint is imposed on the fault magnitude vector, $\boldsymbol{z}$, as only a few faults would be active simultaneously from all the possible faults expressed in the dictionary. The sparsity constraint is represented as the minimisation of the l_0-norm and is expressed in the optimisation framework as

$$\begin{aligned} \min_{\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{K}},\boldsymbol{\mathcal{M}},\mathbf{F}\otimes\boldsymbol{z}^T} & \left\| \mathbf{Y}_m - \begin{bmatrix} \mathbf{T}_u & \mathbf{T}_{y_m} & \mathbf{T}_\theta & \boldsymbol{\Theta} \end{bmatrix} \begin{bmatrix} \boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{K}} \\ \boldsymbol{\mathcal{M}} \\ (\mathbf{F}(\boldsymbol{z}))^T \end{bmatrix} \right\|_2^2 + \|\mathbf{F} \otimes \boldsymbol{z}^T\|_0 \\ s.t. \quad & \operatorname{rank}(\operatorname{vec}(\mathbf{F}^T)\boldsymbol{z}^T) = 1, \\ & \operatorname{rank}\left(\begin{bmatrix} \boldsymbol{\mathcal{M}}_1 & \boldsymbol{\mathcal{K}}_1 \\ \boldsymbol{\mathcal{M}}_2 & \boldsymbol{\mathcal{K}}_2 \\ \vdots & \vdots \\ \boldsymbol{\mathcal{M}}_s & \boldsymbol{\mathcal{K}}_s \\ (\mathbf{F}(\boldsymbol{z}))^T & I \end{bmatrix} \right) = n_y. \end{aligned} \tag{4-21}$$

4-2 Achieving Rank and Sparsity Constraints

The implementation of RISRO is achieved by considering a Single-Input Single-Output(SISO) system, characterised by the elevator deflection, δ_e as the input and the TAS measurement from the ADS, V_{TAS_m} as the system output.

Recall Equation 4-1 for SISO case,

$$\begin{aligned}\hat{\boldsymbol{y}}(k) \approx \ & \begin{bmatrix}\boldsymbol{u}(k-1) & ... & \boldsymbol{u}(k-s)\end{bmatrix}\begin{bmatrix}\boldsymbol{\mathcal{B}}_1^T \\ \vdots \\ \boldsymbol{\mathcal{B}}_s^T\end{bmatrix} + \begin{bmatrix}\boldsymbol{y}_m(k-1) & ... & \boldsymbol{y}_m(k-s)\end{bmatrix}\begin{bmatrix}\boldsymbol{\mathcal{K}}_1^T \\ \vdots \\ \boldsymbol{\mathcal{K}}_s^T\end{bmatrix} \\ & + \begin{bmatrix}\text{vec}(\boldsymbol{\theta}^T(k-1))^T & ... & \text{vec}(\boldsymbol{\theta}^T(k-s))^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\mathcal{M}}_1^T \\ \vdots \\ \boldsymbol{\mathcal{M}}_s^T\end{bmatrix} + \text{vec}(\boldsymbol{\theta}^T(k))^T(\mathbf{F}(\boldsymbol{z}))^T,\end{aligned} \tag{4-22}$$

where $\boldsymbol{y}_m(k) \in \mathbb{R}$, $\boldsymbol{u}(k) \in \mathbb{R}$, $\boldsymbol{\theta}(k) \in \mathbb{R}^{n_d \times n_z}$, $\mathbf{F} \in \mathbb{R}^{1 \times n_d}$, $\boldsymbol{z} \in \mathbb{R}^{n_z}$, $\boldsymbol{\mathcal{B}}_i \in \mathbb{R}^{n_y \times n_u}$, $\boldsymbol{\mathcal{K}}_i \in \mathbb{R}^{n_y \times n_y}$ and $\boldsymbol{\mathcal{M}}_i \in \mathbb{R}^{n_y \times n_y n_d n_z}$.

Recollect that in Equation 4-22, the system parameters characterised by $\boldsymbol{\mathcal{B}}_i$, $\boldsymbol{\mathcal{K}}_i$ and $\boldsymbol{\mathcal{M}}_i$ and the fault parameter, $\mathbf{F}(\boldsymbol{z})$ are the unknowns and the input signal($\boldsymbol{u}(k)$), output signal($\boldsymbol{y}_m(k)$) and dictionary subspace($\boldsymbol{\theta}(k)$) are known.

Restructuring for RISRO(rank-constraint)

The estimated output can be expressed as an inner-product, as in Equation 3-12 such that, at each time step, k,

$$\hat{\boldsymbol{y}}(k) \approx \begin{bmatrix}\boldsymbol{u}(k-1) & ... & \boldsymbol{u}(k-s)\end{bmatrix}\begin{bmatrix}\boldsymbol{\mathcal{B}}_1^T \\ \vdots \\ \boldsymbol{\mathcal{B}}_s^T\end{bmatrix} + \boldsymbol{\mathcal{A}}_k(\mathbf{X}), \tag{4-23}$$

where $\text{rank}(\mathbf{X}) = 1$,

$$\boldsymbol{\mathcal{A}}(\mathbf{X}) = [\langle \mathbf{A}_1, \mathbf{X}\rangle, ..., \langle \mathbf{A}_N, \mathbf{X}\rangle]^T, \mathbf{A}_k \in \mathbb{R}^{p_1 \times p_2}, k = 1, 2, ..., N$$

and

$$\boldsymbol{\mathcal{A}}_k(\mathbf{X}) = \langle \mathbf{A}_k, \mathbf{X}\rangle,$$

where $p_1 = n_d n_z + 1$ and $p_2 = s + 1$. Equation 4-22 can be restructured as

$$\begin{aligned}\hat{\boldsymbol{y}}(k) \approx & \begin{bmatrix}\boldsymbol{u}(k-1) & ... & \boldsymbol{u}(k-s)\end{bmatrix}\begin{bmatrix}\boldsymbol{\mathcal{B}}_1^T \\ \vdots \\ \boldsymbol{\mathcal{B}}_s^T\end{bmatrix} + \\ & \Bigg\langle \underbrace{\begin{bmatrix}\text{vec}(\boldsymbol{\theta}^T(k-1))^T & ... & \text{vec}(\boldsymbol{\theta}^T(k-s))^T & \text{vec}(\boldsymbol{\theta}^T(k))^T \\ \boldsymbol{y}_m(k-1) & ... & \boldsymbol{y}_m(k-s) & 0\end{bmatrix}}_{\mathbf{A}_k} \cdot \underbrace{\begin{bmatrix}\boldsymbol{\mathcal{M}}_1^T & ... & \boldsymbol{\mathcal{M}}_s^T & (\mathbf{F}(\boldsymbol{z}))^T \\ \boldsymbol{\mathcal{K}}_1^T & ... & \boldsymbol{\mathcal{K}}_s^T & 1\end{bmatrix}}_{\mathbf{X}} \Bigg\rangle,\end{aligned} \tag{4-24}$$

where $\text{rank}\left(\begin{bmatrix} \boldsymbol{\mathcal{M}}_1^T & ... & \boldsymbol{\mathcal{M}}_s^T & (\mathbf{F}(\boldsymbol{z}))^T \\ \boldsymbol{\mathcal{K}}_1^T & ... & \boldsymbol{\mathcal{K}}_s^T & 1 \end{bmatrix} \right) = 1.$

The workhorse of RISRO is a linear least squares formulation; hence, Equation 4-24 is split into a linear least squares and a rank-constrained least squares problem as in Equation 3-12 and facilitates the application of RISRO for solving it.

The implementation is described in Algorithm 3.

Sparsity Constraint

Sparsity in $\boldsymbol{z}$ is attained by treating it as a Pseudo-Measurement (PM) which exploits the well-known Kalman Filter (KF) framework [4]. The l_0-norm is approximated using an exponential function, and Algorithm 2 with accordingly modified PM stage is utilised.

Recalling the exponential approximation of l_0-norm, as in Equation 3-21, at time instance, k, for $\boldsymbol{z}(k) \in \mathbb{R}^n$

$$||\boldsymbol{z}(k)||_0 = n - \sum_{i=1}^{n} \exp(-\alpha |z_i(k)|).$$

The fictitious measurement matrix, $\bar{\mathbf{H}}$ in Equation 3-22, for the m^{th} iteration of constrained KF is constructed using the fault magnitude matrix, $(\mathbf{F}(\boldsymbol{z}))^T \in \mathbb{R}^{n_d n_z}$ as

$$\bar{\mathbf{H}}_j^m = \begin{cases} -\alpha \exp(-\alpha (\mathbf{F}(\boldsymbol{z})^T)_j^m), & \text{if } (\mathbf{F}(\boldsymbol{z})^T)_j^m > 0 \\ \alpha \exp(\alpha (\mathbf{F}(\boldsymbol{z})^T)_j^m), & \text{if } (\mathbf{F}(\boldsymbol{z})^T)_j^m \leq 0 \end{cases}, \; for \; j = 1, 2, ...n_d n_z. \tag{4-25}$$

In the equation above, $\bar{\mathbf{H}}_j^m$ represents the j^{th} element of the matrix on the m^{th} PM iteration.

The rank constraint is achieved using RISRO and sparsity using the constrained KF setup. The l_0-norm minimisation can be expressed as a Constrained KF if the regression matrix follows the Restricted Isometry Property(RIP). This condition is also a pre-requisite for RISRO. The combination of the algorithms is designed such that after each iteration of RISRO, sparsity is enforced.

The BFI algorithm output,

$$\widehat{\mathbf{X}} = \begin{bmatrix} \hat{\boldsymbol{\mathcal{K}}}^T \widehat{\mathbf{F}(\boldsymbol{z})^T} & \widehat{\mathbf{F}(\boldsymbol{z})^T} \\ \hat{\boldsymbol{\mathcal{K}}}^T & 1 \end{bmatrix},$$

provides the bilinear variable $\widehat{\mathbf{F}(\boldsymbol{z})^T}$, and the parameters, $\widehat{\boldsymbol{\mathcal{K}}}_1, ...\widehat{\boldsymbol{\mathcal{K}}}_s$. Additionally, the least squares output in Step 4 of Algorithm 3 produces $\hat{\boldsymbol{\mathcal{B}}} = \begin{bmatrix} \hat{\boldsymbol{\mathcal{B}}}_1 \\ \vdots \\ \hat{\boldsymbol{\mathcal{B}}}_s \end{bmatrix}$ simultaneously.

Algorithm 3 Blind Fault Identification Pseudo-Code

Inputs:

$s \in \mathbb{N}, p_1, p_2, \boldsymbol{u} \in \mathbb{R}^N, \boldsymbol{y} \in \mathbb{R}^N, \theta \in \mathbb{R}^{n_d \times n_z}, \alpha \in \mathbb{R}, \mathbf{P} \in \mathbb{R}^{n_z \times n_z}, \mathbf{R} \in \mathbb{R}$

Initialisation:

$\mathbf{X}^1 = \mathbf{U}^1 \mathbf{\Sigma}^1 \mathbf{V}^{1T}$

Compute $\mathbf{T}_u, \mathbf{A}_1, ..., \mathbf{A}_N$

for i=1,...,N_{iter} **do**

Perform Importance Sketching on $\boldsymbol{\mathcal{A}}$ to obtain covariates $\boldsymbol{\mathcal{A}}_B$, $\boldsymbol{\mathcal{A}}_{D_1}$,$\boldsymbol{\mathcal{A}}_{D_2}$

Solve linear least squares problem

$$(\boldsymbol{\mathcal{B}}^i, \mathbf{B}^i, \mathbf{D}_1^i, \mathbf{D}_2^i) = \arg \min_{\boldsymbol{\mathcal{B}}, \mathbf{B}, \mathbf{D}_1, \mathbf{D}_2} \|\boldsymbol{y} - \mathbf{T}_u \boldsymbol{\mathcal{B}} - \boldsymbol{\mathcal{A}}_B(\mathbf{B}) - \boldsymbol{\mathcal{A}}_{D_1}(\mathbf{D}_1) - \boldsymbol{\mathcal{A}}_{D_2}(\mathbf{D}_2^T)\|_2^2$$

Compute

$$\mathbf{X}_U^{i+1} = \mathbf{U}^i \mathbf{B} + \mathbf{U}_\perp^i \mathbf{D}_1$$
$$\mathbf{X}_V^{i+1} = \mathbf{V}^i \mathbf{B}^T + \mathbf{V}_\perp^i \mathbf{D}_2$$
$$\hat{\mathbf{X}}^{i+1} = \mathbf{X}_U^{i+1} (\mathbf{B})^\dagger \mathbf{X}_V^{i+1}$$

Compute $\mathbf{F}(\boldsymbol{z})^T$, $\boldsymbol{\mathcal{K}}^T$

$$\mathbf{F}(\boldsymbol{z})^T = \hat{\mathbf{X}}^{i+1}_{1:n_d n_z, end}$$
$$\boldsymbol{\mathcal{K}}^T = \hat{\mathbf{X}}^{i+1}_{end, 1:s}$$

if $i \neq N_{iter}$ **then** ▷ To enforce sparsity constraint

for $m = 0, 1, ..., M$ **do**

$$\bar{\mathbf{H}}_j^m = \begin{cases} -\alpha \exp(-\alpha \mathbf{F}(\boldsymbol{z})_j^{T^m}), & \text{if } \mathbf{F}(\boldsymbol{z})_j^{T^m} > 0 \\ \alpha \exp(\alpha \mathbf{F}(\boldsymbol{z})_j^{T^m}), & \text{if } \mathbf{F}(\boldsymbol{z})_j^{T^m} \leq 0 \end{cases}, j = 1, 2, ..n_d n_z$$

$$\mathbf{K}^m = \bar{\mathbf{H}^m} \mathbf{P}^m (\bar{\mathbf{H}^m} \mathbf{P}^m \bar{\mathbf{H}^m}^T + \mathbf{R})^{-1}$$

$$\mathbf{F}(\boldsymbol{z})^{T^{m+1}} = \mathbf{F}(\boldsymbol{z})^{T^m} + \mathbf{K}^m \left[n_d n_z - \textstyle\sum_{j=1}^{n_d n_z} \exp(-\alpha |\mathbf{F}(\boldsymbol{z})_j^{T^m}|) \right]$$

$$\mathbf{P}^{m+1} = (\mathbf{I} - \mathbf{K}^m \bar{\mathbf{H}}^m) \mathbf{P}^m$$

end for

end if

Update

$$\hat{\mathbf{X}}^{i+1} = \begin{bmatrix} \boldsymbol{\mathcal{K}}^T \mathbf{F}(\boldsymbol{z})^{T^M} & \mathbf{F}(\boldsymbol{z})^{T^M} \\ \boldsymbol{\mathcal{K}}^T & 1 \end{bmatrix}$$

Update $\mathbf{U}^{i+1}$,$\mathbf{V}^{i+1}$ using QR orthogonalisation:

$$\mathbf{U}^{i+1} = \mathrm{QR}(\mathbf{X}_U^{i+1}),\ \mathbf{V}^{i+1} = \mathrm{QR}(\mathbf{X}_V^{i+1})$$

end for

Chapter 5

Application to Air Data Sensor faults

This chapter succinctly encapsulates the primary findings of this thesis. It presents results about the robustness and accuracy of Blind Fault Identification (BFI) in detecting and isolating additive Air Data Sensor (ADS) faults using real flight data. The proposed scheme has been compared against the state-of-the-art model-based fault diagnosis method, Double-Model Adaptive Estimation (DMAE), explained in Chapter 2. Additionally, a scheme to detect the start time of a fault is presented and evaluated.

Fault identification encompasses detection, diagnosis and isolation of faults. Distinctly, detection pertains to identifying the presence of a fault within a system, diagnosis entails estimating the nature and magnitude of the fault, and isolation typically involves specifying the fault location. In the context of BFI, detection, diagnosis, and isolation are achieved by estimating the magnitude of the active fault(s) from an extensive dictionary comprising all possible fault scenarios. The objectives of the experiments focus on the fault identification aspect rather than system identification and hence, have been designed accordingly.

5-1 Validation Objectives and Metrics

The validation objective is to evaluate the robustness and accuracy of fault detection capabilities offered by the BFI in comparison to DMAE(as elaborated in Chapter 2) when utilising real flight data. The validation approach involves introducing an artificial fault into the measured parameter and applying both BFI and DMAE to identify this fault. A fundamental distinction between these two methodologies is that BFI relies on an extensive and meticulously designed fault data dictionary. The DMAE benchmark code is explained in Section 5-2. To facilitate the analysis, we have considered the following scenarios:

Case 1: Injected fault is precisely represented in the data dictionary. The average fault detection performance is determined by conducting Monte Carlo simulations of 100 different fault magnitudes for varying numbers of active faults. The magnitude error convergence and fault isolation rate are analysed for the same.

Case 2: The injected fault is a perturbed variation of the fault present in the data dictionary (the perturbation is not accounted for in the dictionary). The fault detection performance for increasing perturbation magnitude has been compared with DMAE.

Case 3: The injected fault is a time-shifted version of the fault represented in the data dictionary.

Validation Metrics

1. Frobenius norm of fault magnitude estimation error is calculated on the bilinear variable, $\mathbf{F}(\boldsymbol{z}) = \mathbf{F} \otimes \boldsymbol{z}^T$ as
$$= \frac{||\widehat{\mathbf{F}(\boldsymbol{z})} - \mathbf{F}(\boldsymbol{z})||_F}{||\mathbf{F}(\boldsymbol{z})||_F}, \tag{5-1}$$
where $\mathbf{F}(\boldsymbol{z})$ and $\mathbf{F}(\hat{\boldsymbol{z}})$ are the true and estimated variables respectively.

2. Fault and output reconstruction are validated by analysing the Variance Accounted For (VAF), calculated as
$$\text{VAF} = \left[1 - \frac{||\boldsymbol{y} - \hat{\boldsymbol{y}}||_2^2}{||\boldsymbol{y}||_2^2}\right] 100\%, \tag{5-2}$$
where $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ represent the true and estimated signal vector respectively. The experiments below use VAF to analyse the measurement and fault reconstruction using BFI and DMAE. For BFI the fault is estimated as the product of the estimated magnitudes and the fault subspace, knowledge of which is assumed.

3. Fault isolation success rate is quantified by determining how often, out of 100 trials, the faults are accurately detected. Detection is considered accurate when the non-zero components of $|\hat{\boldsymbol{z}}| > \epsilon$, with $\epsilon = 10^{-3}$ correspond to non-zero fault magnitudes in $\boldsymbol{z}$ respectively. This corresponds to zero false alarms and zero missed detections. In this analysis, $\hat{\boldsymbol{z}}$ is the normalised vector obtained by performing an Singular Value Decomposition (SVD) on the bilinear optimisation variable, $F(z)$.

5-2 Experiment Framework

Flight Data

The flight test data was obtained during Aerodynamic Model Identification(AMI) experiments on the Cessna Citation II aircraft housed at the Faculty of Aerospace Engineering at the Delft University of Technology. The dataset used in the experiments for this thesis has been obtained under cruise flight conditions, without turbulence.

The signal of interest for fault identification of ADS sensors is the True Airspeed (TAS), denoted by V_{TAS_m}, a longitudinal parameter. To this end, the data used consists of persistently exciting control input, '3-2-1-1', on the elevators to excite the longitudinal modes of the aircraft. The use of this data in BFI can identify the longitudinal dynamics alone. The dataset characteristics have been summarised in Table 5-1 and shown in Figure 5-1.

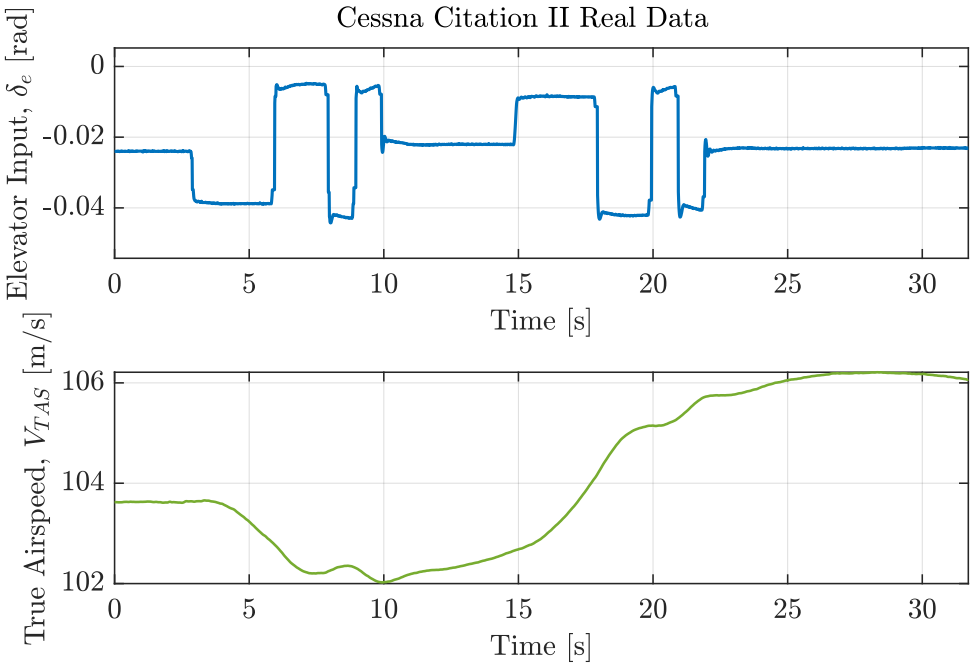


Figure 5-1: Flight Data: Control Input and Air Data Sensor Measurement

Parameter	Value	Unit	Notation
Input	Elevator Deflection	rad	δ_e
Output	True Airspeed	m/s	V_{TAS}
Altitude	≈ 5200	m	h
Data Length(Samples)	3172	-	N
Sampling Interval	0.01	s	dt
Operation Condition	Cruise	-	-

Table 5-1: Flight Data Characteristics

It is essential to emphasise that handling sensor biases and anomalies through Flight Path Reconstruction(FPR) techniques when utilising raw data is imperative. Nevertheless, using ADS measurements exclusively obviates the need for flight path reconstruction, as they are inherently not as prone to sensor biases and anomalies as the Inertial Measurement Units(IMUs).

Semi-real fault data has been created by artificially injecting faults in the otherwise 'clean' measurements obtained during AMI experiments. The nature of induced faults and their representation in the dictionary have been detailed next.

Fault Signals and Data Dictionary

The fault signal, $f(k)$, acts additively on the true measurement, $V_{TAS}(k)$ and is represented as a scaled sum of known basis functions which constitute the fault dictionary(subspace), $\boldsymbol{\theta}(k) \in \mathbb{R}^{n_z}$. Using the fault magnitude vector, $\boldsymbol{z} \in \mathbb{R}^{n_z}$, the fault at time step k is expressed as

$$\begin{aligned} V_{TAS_m}(k) &= V_{TAS}(k) + f(k), \\ V_{TAS_m}(k) &= V_{TAS}(k) + \underbrace{[F_1]}_{\mathbf{F}} \underbrace{\begin{bmatrix} \theta_1(k) & \theta_2(k) & .. & \theta_{n_z}(k) \end{bmatrix}}_{\boldsymbol{\Theta(k)}} \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n_z} \end{bmatrix}}_{\boldsymbol{z}}. \end{aligned} \tag{5-3}$$

The choice of fault subspace signals is based on laboratory experiments studying the deviations introduced in TAS measurements upon blockage of pitot probes [9] [11] [30] [22]. In line with the findings, different faults are represented as biases, drifts and sinusoidal signals, tabulated in Table 2-1.

The BFI method of fault diagnosis requires the design of a fault dictionary encompassing all possible fault scenarios. For the experiments in this chapter, the dictionary comprises seven signals, tabulated in Table 5-2 and depicted in Figure 5-2.

Caution should be exercised while constructing the fault dictionary, as the workhorse of the BFI framework is the Recursive Importance Sketching for Rank-Constrained Optimisation (RISRO) algorithm. Since it uses an unconstrained least squares solution at its core, the condition number of the Toeplitz matrices of the fault dictionary is a crucial factor in ensuring the successful reconstruction of the fault signal and estimation of system parameters.

Longitudinal Aircraft Model for diagnosis using Double Model Adaptive Estimation

DMAE is a model-based fault diagnosis approach that has two Kalman Filter (KF)s running in parallel to detect and diagnose the presence of a fault. The DMAE software used for the analysis has been obtained from the author's online repository [19]. The software available online implements DMAE for a toy second-order system mentioned in [20]. In this study, the second-order model has been replaced by the fourth-order longitudinal model of the aircraft and supplied with actual flight data(δ_e, V_{TAS}) from the Cessna Citation II. It is to be noted that certain modifications were introduced to remove the estimation of disturbances and have only one augmented fault(on the V_{TAS} measurement) in the augmented filter.

Dictionary Signal	Fault Type	Start Time	Expression
$\theta_1(t)$	Sine	0.2 s	$\sin(4.5\pi t)$
$\theta_2(t)$	Bias	13.8 s	1
$\theta_3(t)$	Sine	6.35 s	$\sin(2.3\pi t)$
$\theta_4(t)$	Sine	10.58 s	$\cos(2.25\pi t)$
$\theta_5(t)$	Sine	5.3 s	$\sin(0.7\pi t)$
$\theta_6(t)$	Drift	7.93 s	$0.23t$
$\theta_7(t)$	Bias	6.35 s	1

Table 5-2: Fault Subspace Signals as a function of time, t

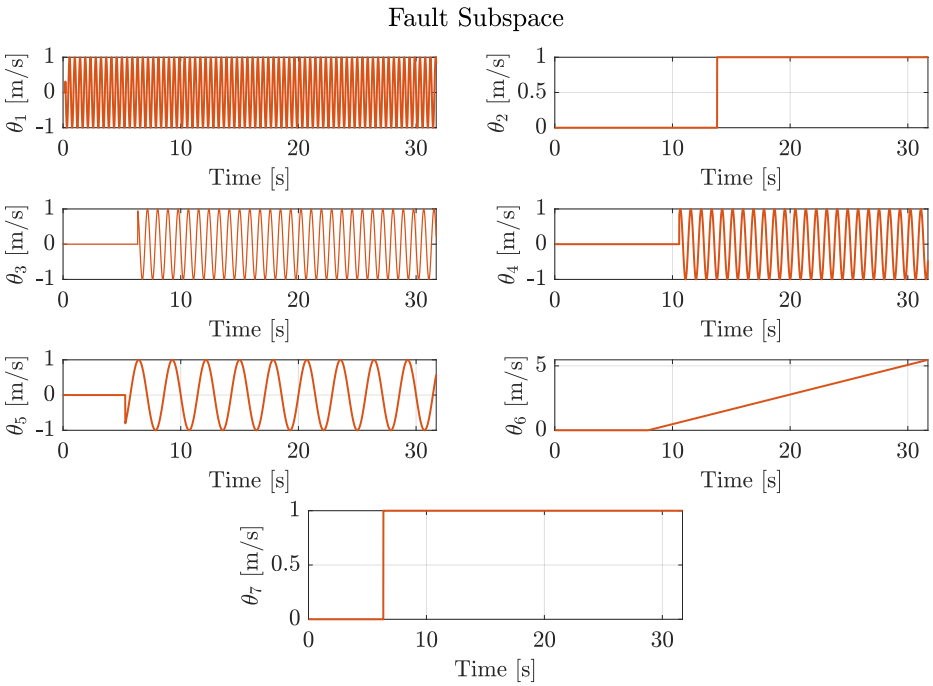


Figure 5-2: Fault Data Dictionary Signals

The dataset under consideration for this study comprises the elevator deflection input signal and the corresponding airspeed measurement obtained from ADS. These signals pertain to the longitudinal aspects of an aircraft; hence, the aerodynamic longitudinal model of the aircraft is the appropriate model to characterise the dynamics.

The continuous-time linearised longitudinal model, at trim conditions of airspeed $V_{TAS} = 100m/s$ and altitude, $h = 5000\ m$, is as below:

$$\begin{bmatrix} \dot{V_{TAS}} \\ \dot{\alpha} \\ \dot{\theta} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.0179 & 3.4177 & -9.7911 & -0.2036 \\ -0.0020 & -1.1443 & 0 & 0.9777 \\ 0 & 0 & 0 & 1 \\ 0.0017 & -7.1848 & 0 & -1.6400 \end{bmatrix} \begin{bmatrix} V_{TAS} \\ \alpha \\ \theta \\ q \end{bmatrix} + \begin{bmatrix} -0.4525 \\ -0.1186 \\ 0 \\ -11.4642 \end{bmatrix} \begin{bmatrix} \delta_e \end{bmatrix}. \quad (5\text{-}4)$$

The model has been discretised using sampling interval, $dt = 0.01s$, to obtain the following model:

$$\mathbf{A}_{lin} = \begin{bmatrix} 0.9834 & 0.0000 & -0.0708 & -0.0000 \\ -0.0023 & 0.9998 & 0.0341 & -0.0979 \\ 0.0096 & -0.0000 & 0.9883 & 0.0000 \\ 0.0099 & 0.0000 & -0.0004 & 1.0000 \end{bmatrix}, \quad \mathbf{B}_{lin} = \begin{bmatrix} -0.1136 \\ -0.0044 \\ -0.0017 \\ -0.0006 \end{bmatrix}. \quad (5\text{-}5)$$

5-3 Fault Diagnosis Performance Comparison

In the experiments detailed below, the parameters for BFI as in Algorithm 3 are $s = 4$, $\alpha = 1$, $M = 1$, $\mathbf{P} = \mathbf{I}_{n_z \times n_z}$ and $\mathbf{R} = 10$, unless mentioned otherwise.

5-3-1 Case 1: Precise Representation of Fault in Dictionary

This section first demonstrates the performance of magnitude estimation error and fault isolation rate of BFI algorithm using Monte-Carlo simulations of different fault magnitudes. The estimated fault magnitude vector, $\hat{\boldsymbol{z}}$ appears in the bilinear optimisation variable, $\widehat{\mathbf{F}(\boldsymbol{z})}$; hence it can be estimated only upto a scalar multiple. Therefore, to facilitate estimation error analysis, the bilinear term, $\widehat{\mathbf{F}(\boldsymbol{z})}$ is considered against the ground truth.

The mean estimation error has been averaged over 100 simulations with a randomly chosen magnitude vector for varying numbers of active faults.

The mean estimation error is significantly lower for one active fault than for a higher number of active faults due to the regularisation achieved by the sparsity constraint. With an increasing number of active faults, the sparsity constraint pushes elements of the magnitude vector towards zero, increasing the error. However, this increase in estimation error does not lead to missed detections or false alarms, as seen in Figure 5-4.

The normalised $\hat{\boldsymbol{z}}$ is obtained through the SVD of $\mathbf{F}(\boldsymbol{z})$. The criteria for successful fault isolation is as mentioned previously in the Validation Metrics.

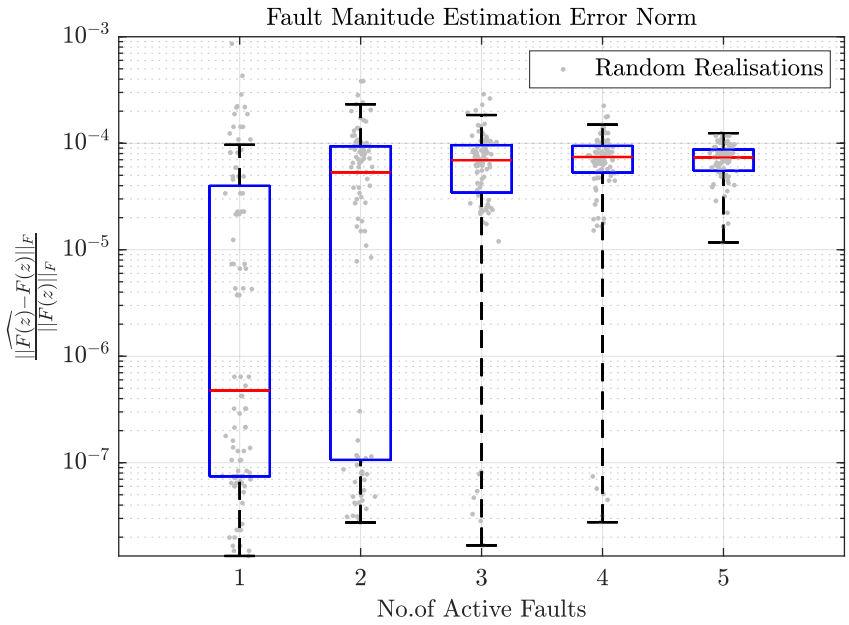


Figure 5-3: Magnitude Estimation Error against the number of active faults(out of 7), averaged over 100 simulations of randomly chosen entries of the fault magnitude vector.

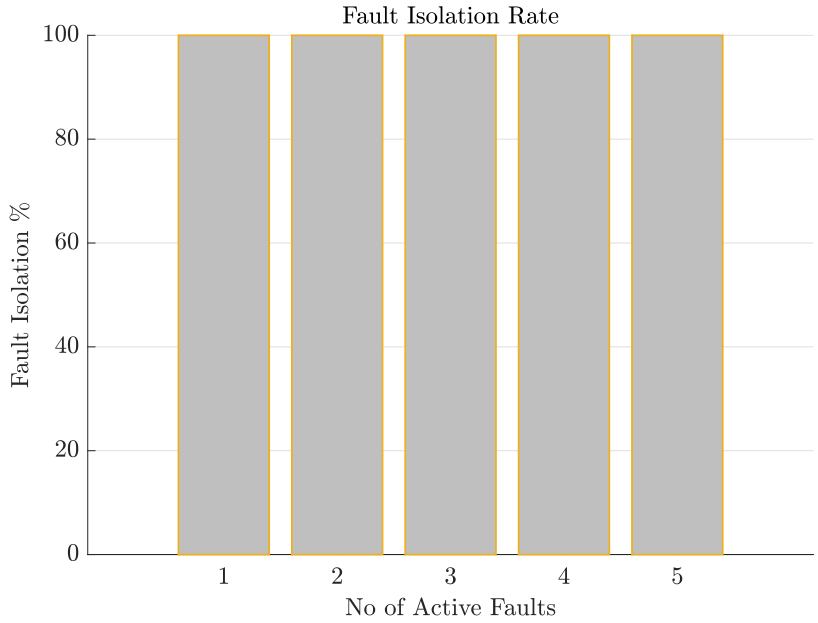


Figure 5-4: Fault Isolation rate per 100 simulations for each number of active fault

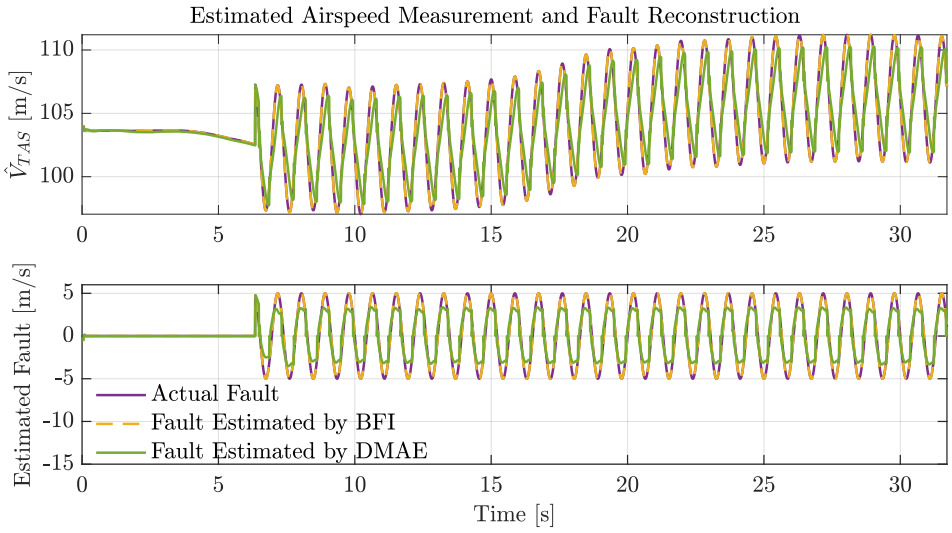


Figure 5-5: Reconstruction of faulty measurement and fault using BFI(yellow) and DMAE(green). The fault reconstruction by BFI enjoys a higher VAF due to its exact representation in the dictionary.

Comparison of BFI fault detection with DMAE

To compare fault detection performance, the fault reconstruction performance for a sinusoidal fault, expressed in Equation 5-6, is analysed in terms of the fault and measurement VAF. The fault signal and the faulty measurement are given as

$$\begin{aligned} f(k) &= 5\sin(2.3\pi t(k)), \\ V_{TAS_m}(k) &= V_{TAS}(k) + \underbrace{\begin{bmatrix} 0 & 0 & 5 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{F}(z)} \boldsymbol{\theta}^T(k), \end{aligned} \tag{5-6}$$

with fault start instance, $k = 635$, i.e. fault start time $t_f = 6.35s$.

The DMAE fault and measurement estimation is obtained using the (un)augmented longitudinal aerodynamic model with process and measurement noise covariance matrices tuned at

$$\mathbf{Q}_k = 10^{-5}\text{diag}\begin{bmatrix} 0.4 & 0.4 & 0.4 & 0.4 \end{bmatrix}, \quad \mathbf{R}_k = 0.01,$$

where diag[.] represents a diagonal matrix with its principal diagonal elements. The fault reconstruction VAF for BFI and DMAE are 99% and 98% respectively, whereas the measurement estimation VAF for BFI and DMAE are at 100% and 97% respectively. The performance of both algorithms was compared for a slow-moving fault,

$$f(k) = 4\sin(0.7\pi t(k)), \tag{5-7}$$

with fault start time, $t_f = 5.3s$. In the case of DMAE, the fault dynamics were absorbed in the state estimate, and the fault is mis-detected as a negative bias of the same magnitude. Nonetheless, the availability of a fault dictionary facilitates the identification of a slow-moving sinusoidal signal using BFI, provided that said signal is included in the dictionary.

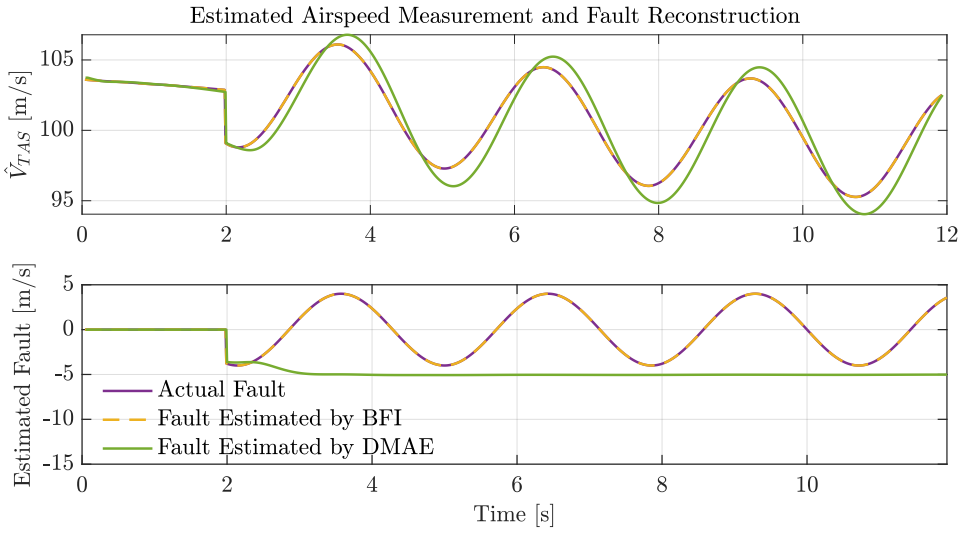


Figure 5-6: Reconstruction of fault measurement and fault by BFI(yellow) and DMAE(green). DMAE detects the abrupt start of fault; however, the slow dynamics of the fault are not captured, leading to correct detection of the fault but incorrect diagnosis.

5-3-2 Case 2: Actual Fault is a perturbed version of the dictionary signal

The fault detection and diagnosis are studied for a case where the actual fault affecting the measurement is a perturbed version of the fault signal present in the dictionary. The following fault signal is considered:

$$f(k) = 4 + \kappa \cos(4.5\pi t(k)), \tag{5-8}$$

where κ is a perturbation constant, the cosine perturbation is chosen to be orthogonal to the fault subspace. The faulty measurement is shown in Figure 5-7, and can be expressed as

$$V_{TAS_m}(k) = V_{TAS}(k) + \underbrace{\begin{bmatrix}0 & 4 & 0 & 0 & 0 & 0 & 0\end{bmatrix}}_{\mathbf{F}(z)} \boldsymbol{\theta}^T(k) + \kappa \cos(4.5\pi t(k)). \tag{5-9}$$

It is observed that the fault reconstruction using BFI suffers as κ increases. However, the measurement estimation enjoys a high VAF against increasing κ. This could imply the existence of multiple solutions; however, the high VAF of the estimated output with the faulty measurement hints at a tendency to absorb the characteristics of the perturbations by overfitting.

The specific case of $\kappa = 0.2$ is studied in detail by analysing the fault detection, diagnosis and isolation performance with 's', ranging from $s = 3$ to $s = 80$, shown in Figure 5-9. It is observed that at $s = 5$, the fault magnitude estimation error decreases significantly, increasing the fault reconstruction VAF; however, the fault is not accurately identified as per Equation 5-9, which has only one active fault, i.e. the bias fault. Increasing 's' improves the isolation performance and the fault diagnosis, achieving its best performance at $s = 45$, after which the VAF, estimation error and isolation deteriorate.

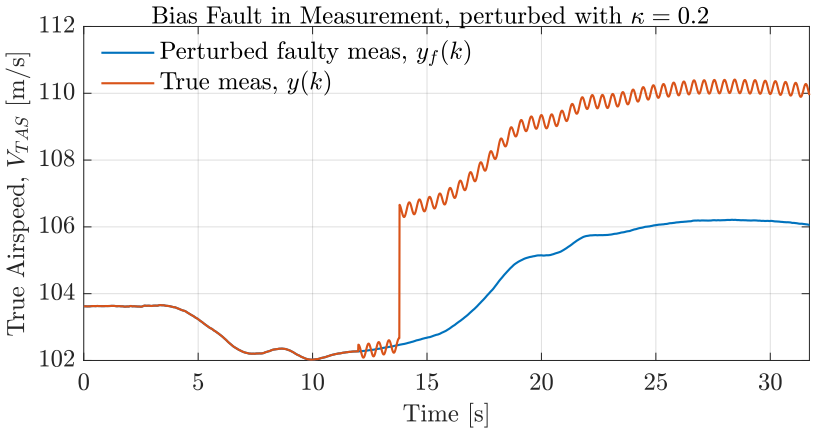


Figure 5-7: Representation of faulty airspeed measurement perturbed by cosine signal Equation 5-9 with $\kappa = 0.2$. The perturbation starts before the fault at $t = 12s$.

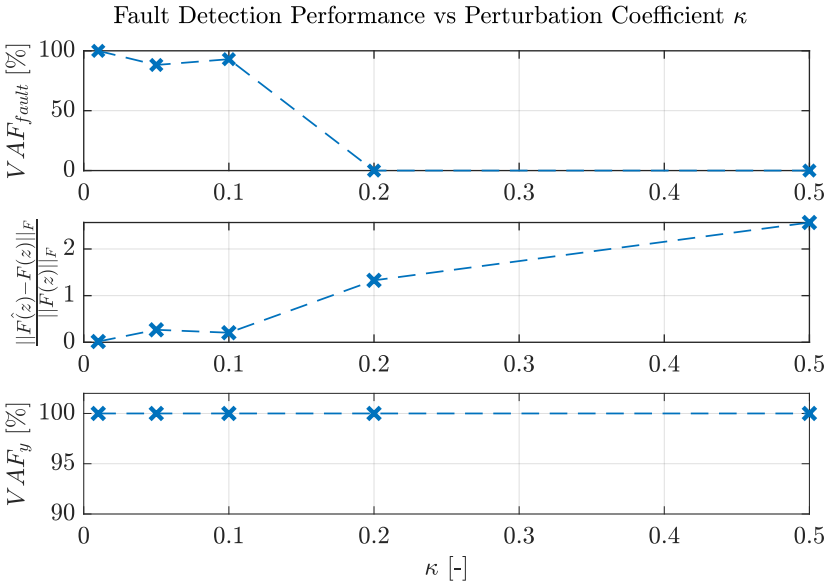


Figure 5-8: Fault Identification performance and measurement estimation in terms of VAF, against perturbation constant, κ, for $s = 4$

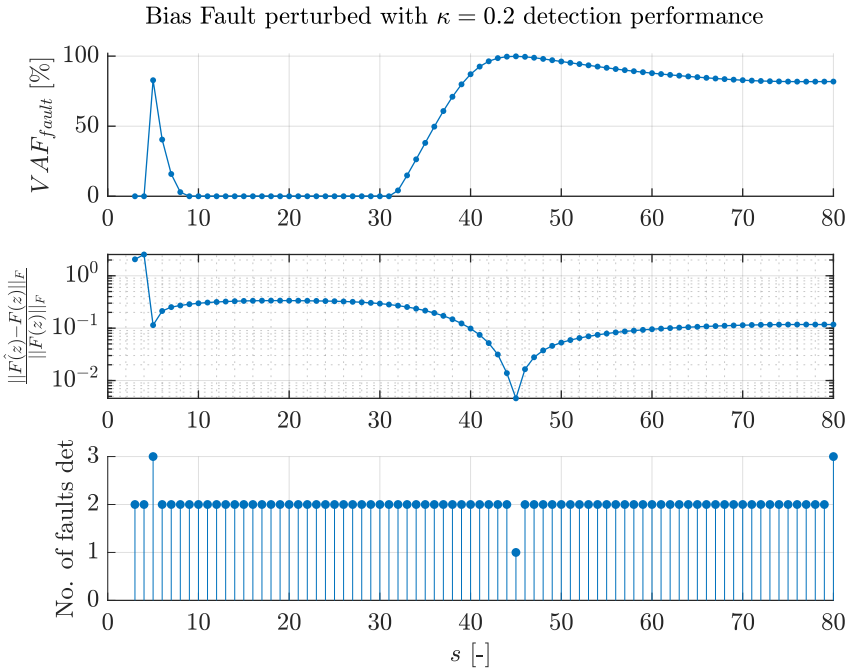


Figure 5-9: Fault Identification performance of BFI against 's', in terms of fault VAF, magnitude estimation and isolation (no.of faults detected). The actual fault is a bias signal perturbed by a cosine signal of magnitude $\kappa = 0.2$.

In the VARX input-output formulation, in the absence of noise,

$$(\mathbf{A} - \mathbf{KC})^s \approx 0, \qquad s \geq n_x \tag{5-10}$$

where n_x is the model order and the input-output relationship defined by Equation 4-12 is exact. However, a sufficiently large s is required in the presence of perturbations. Theoretically, $s >> n_x$ for Equation 5-10 to hold true [26].

5-3-3 Case 3: Actual Fault is a time-shifted version of dictionary signal

The third case investigates the fault scenarios where the fault is represented in the dictionary but shifted in time. Let the fault, $f(k)$ be a sinusoidal signal, with start time, $t_f = 15.86s$ expressed as

$$f(k) = \sin(2.3\pi t(k)). \tag{5-11}$$

The dictionary includes a sinusoidal signal with the same frequency, but the start time is $6.35s$. The fault reconstruction performance of BFI and DMAE are shown in Figure 5-11. The DMAE enjoys superior performance(fault VAF = 98%) compared to DMAE in fault detection and diagnosis as it relies only on residuals and does not assume fault information. BFI detects the presence of the sinusoidal component but also gives a false alarm for drift fault with 0% fault VAF. However, the output estimation using BFI has a VAF of 100%, implying the existence of multiple solutions.

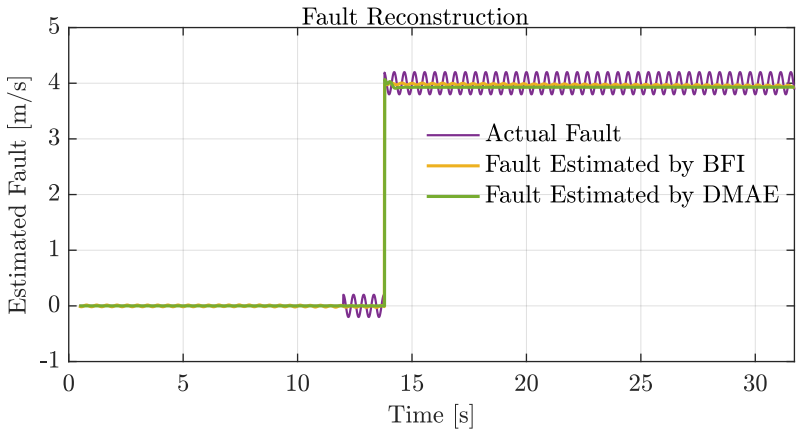


Figure 5-10: Fault Reconstruction performance of BFI with $s = 45$, against DMAE with $\mathbf{Q}_k = 10^{-4}\text{diag}\begin{bmatrix}0.24 & 0.24 & 0.24 & 0.24\end{bmatrix}$, $\mathbf{R}_k = 0.005$. Increasing 's' allows BFI to detect and isolate perturbed faults. It is important to factor in that while BFI is detecting an imprecisely represented fault, DMAE is provided with a true representation of the system dynamics around the trim condition.

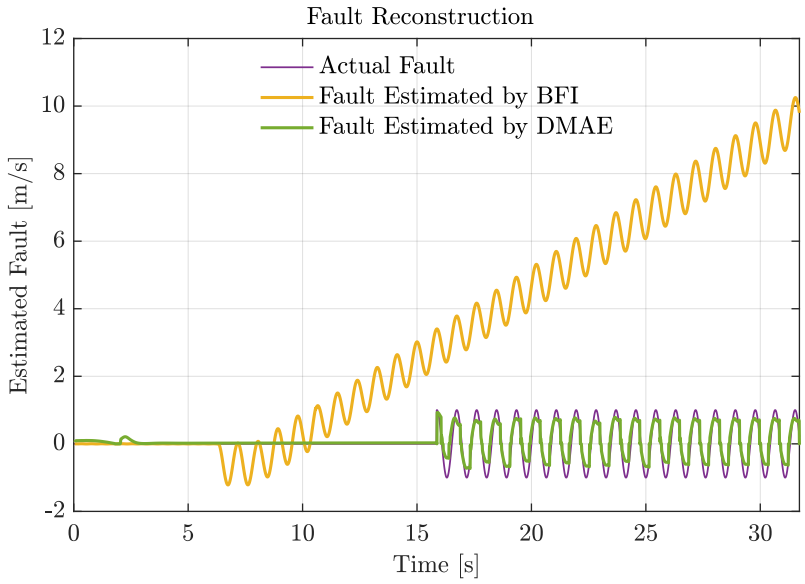


Figure 5-11: Fault Reconstruction performance of BFI with $s = 4$, against DMAE with $\mathbf{Q}_k = 10^{-6}\text{diag}\begin{bmatrix}0.4 & 0.4 & 0.4 & 0.4\end{bmatrix}$, $\mathbf{R}_k = 10^{-3}$ for a time-shifted sinusoidal fault.

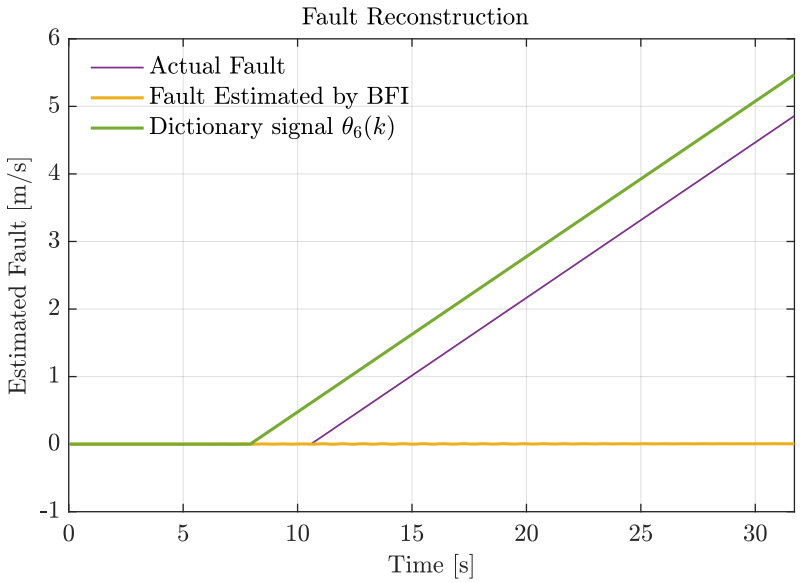


Figure 5-12: Fault Reconstruction performance of BFI with $s = 45$ for a time-shifted drift signal.

Detecting time-shifted drift signals provides an interesting case wherein a fault is not detected, leading to a missed detection, as shown in Figure 5-12. It implies that unless the true fault matches the fault start-time of the dictionary, no detection is raised. Based on this, the fault start-time detection framework is built, explained next.

5-4 Fault Start-Time Detection

The fault data dictionary in the experiments above assumes knowledge of the start time of the fault, as shown in Figure 5-2; however, the fault start time is a parameter to be ascertained and can not be predetermined.

The findings from Subsection 5-3-3 provide insights into how the fault-start time can be computed. It was observed that the measurement VAF maintained high accuracy even when the faults were imprecise or shifted in time. Consequently, the VAF does not provide information regarding a fault's presence or absence. In contrast, the magnitude estimates denoted as $\mathbf{F}(\hat{\boldsymbol{z}})$ displayed variations and can be harnessed for detecting the initiation times of faults.

The case of complete blockage resulting in bias and drift faults is considered here. Let the fault dictionary comprise only one step signal, starting at the time, $t_f = 15s$, and affecting the output, V_{TAS}, shown in Figure 5-13.

$$\begin{aligned} f(k) &= 5, \\ V_{TAS_m}(k) &= V_{TAS}(k) + f(k), \end{aligned} \tag{5-12}$$

The fault data dictionary does **not** assume knowledge of the fault start-time and has an arbitrary start time. Let us assume that the dictionary comprises a step signal with a start time $= 1s$. $\boldsymbol{\theta}(k) \in \mathbb{R}^{n_z}$, with $n_z = 1$.

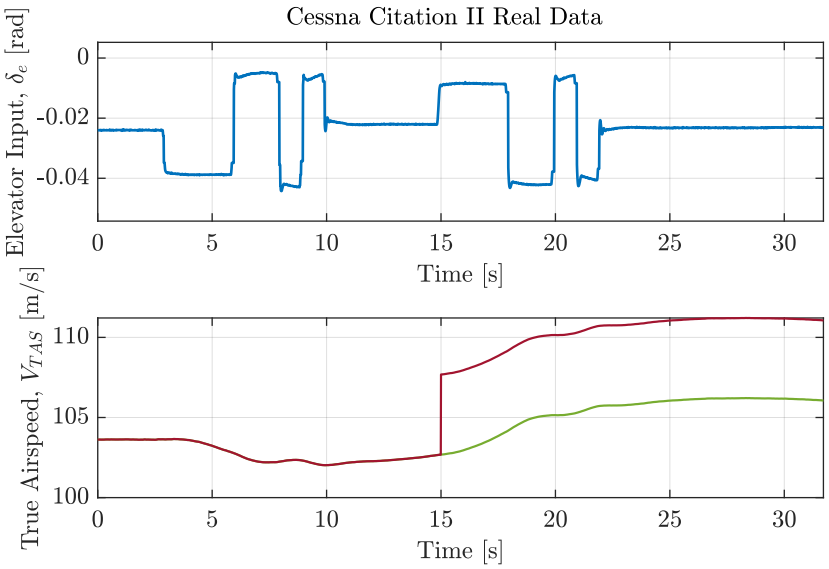


Figure 5-13: Real Flight Elevator input along with true(green) and faulty(maroon) measurement with a bias fault injected at $t_f = 15s$.

Instead of executing the BFI algorithm once for a batch of data, it is run 'N' times, iteratively, each time using a different data dictionary. In each iteration, the data dictionary assumes a different start time of the fault, expressed as

$$\boldsymbol{\theta}(k) = \begin{cases} 0, & \text{if } k < k_f \\ 1, & \text{if } k \geq k_f \end{cases}, \tag{5-13}$$

where k_f is the fault start time, varied from $k_f = 1$ to N. The batch optimisation is conducted using the dictionary of time-shifted bias signals. The fault magnitude estimate, $\widehat{\mathbf{F}(\boldsymbol{z})}$ is computed for each iteration, and its peak represents the fault start time as shown in Figure 5-14.

The same has been conducted for a drift fault using a time-shifted drift signal, shown in Figure 5-15.

In both cases, the fault dictionary consisted of only one signal, $n_z = 1$, with the exact nature of fault as was injected. In the experiment below, the reverse is investigated. The magnitude is estimated for a bias-injected fault measurement by shifting the drift dictionary in time and vice versa, shown in Figure 5-16.

Ideally, we would like the $\widehat{\mathbf{F}(\boldsymbol{z})}$ estimate to ≈ 0 when using switched dictionaries. However, as observed from Figure 5-16a, using a time-shifted drift dictionary in the presence of a bias leads to false alarms, often erroneously high in magnitude. The estimation of drift fault using time-shifted bias dictionaries displays ideal behaviour with $\widehat{\mathbf{F}(\boldsymbol{z})}(k) \approx 0$, for all k. In practice, the methodology that can be adopted to ensure accurate fault detection is first performing BFI with time-shifted bias dictionaries and then consecutively following it with time-shifted drift dictionaries.

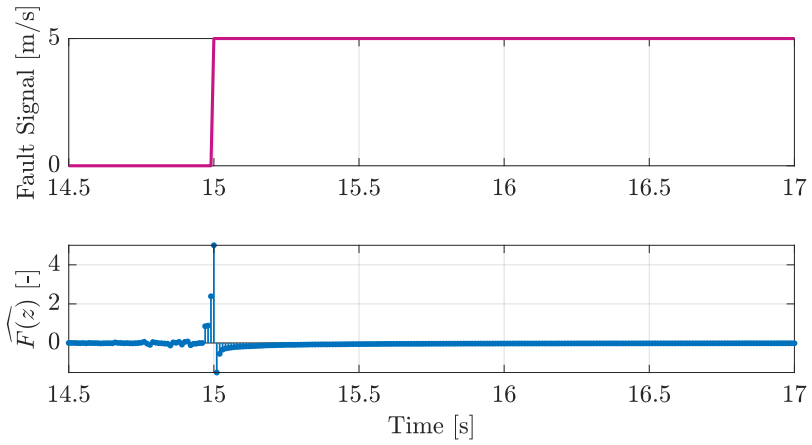


Figure 5-14: (Above)Plot of the true Bias fault(pink) starting at 15 sec.(Below)Fault magnitude estimate, $\widehat{\mathbf{F}(z)}$ against time, as a function of t_f, where t_f represents the fault-start time incorporated in the fault dictionary.

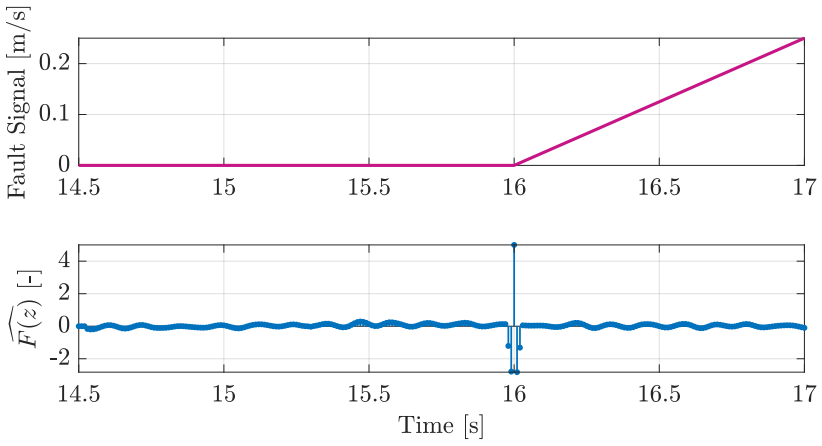
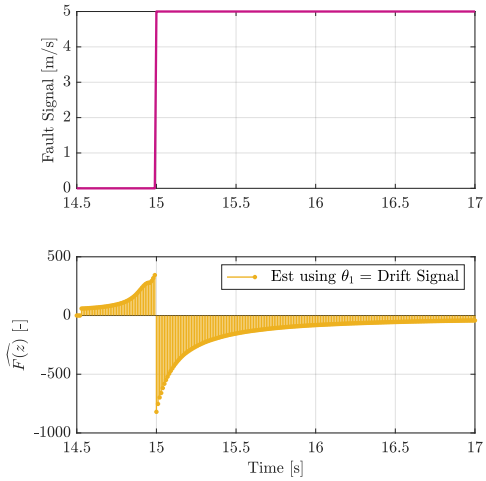
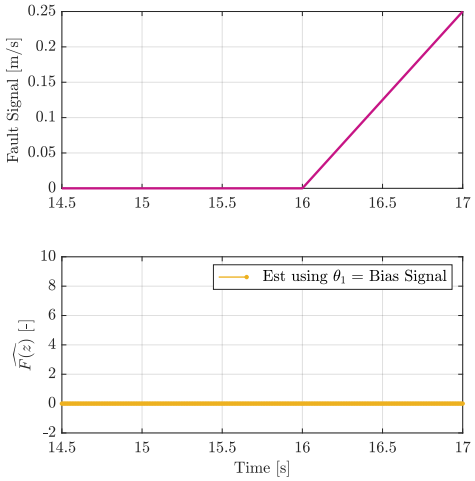


Figure 5-15: (Above)Plot of the actual Drift fault(pink) starting at 16 sec.(Below)Fault magnitude estimate, $\widehat{\mathbf{F}(z)}$ against time, as a function of t_f, where t_f represents the fault-start time incorporated in the fault dictionary.



(a) (Above) Injected true Bias fault. ((Below)Fault magnitude estimate, $\widehat{\mathbf{F}(z)}$ against time, as a function of t_f, where t_f represents the fault-start time incorporated in the **drift** fault dictionary.



(b) (Above) Injected true drift fault. ((Below)Fault magnitude estimate, $\widehat{\mathbf{F}(z)}$ against time, as a function of t_f, where t_f represents the fault-start time incorporated in the **bias** fault dictionary.

Figure 5-16: Fault Magnitude Estimation using switched drift and bias time-shifted dictionaries.

5-5 Results Summary

In this chapter, the BFI framework developed earlier was validated using real flight data and its performance, analysed for different cases. The first case examined is where the fault data dictionary exactly represents the actual fault. To assess the robustness, the algorithm performance was analysed in the face of uncertainties in accurately modelling the fault in the dictionary – first as perturbations and second as time-shifted versions of the dictionary signals.

The analysis has led to interesting results; specifically, in the case of perturbed signals, the model order denoted as 's' emerged as a critical factor influencing the accuracy of fault estimation and isolation. Additionally, it is noteworthy that employing this framework when the fault signal does not precisely match the dictionary can lead to missed detections and false alarms. Nevertheless, the framework maintains a consistently high measurement Variance Accounted For (VAF) of approximately 100%, indicating multiple optimisation solutions.

In the context of time-shifted bias and drift signals of the fault dictionary, high missed detections and low false alarms were observed. This observation allows for determining the fault start-time, presented and evaluated in this chapter. Similar to the previous scenario, the high measurement VAF across all cases prompts an exploration of fault magnitude estimation peaks as a parameter for computing the fault start time. It is important to note that this method, still in its rudimentary stages, is applicable solely to step and drift faults, as its use with sinusoidal faults can result in false alarms.

Chapter 6

Conclusions

The concluding chapter offers an overview of the thesis work and its alignment with addressing the research question. Moreover, it outlines suggestions for future research while underlining the challenges and constraints inherent in the proposed methodology.

6-1 Thesis Summary

The research objective of this thesis was to develop and investigate a fault diagnosis methodology for detecting Air Data Sensor (ADS) sensor faults that neither suffered from model dependency nor necessitated intensive training with vast historical data. In a preliminary investigation, the innovative Blind Fault Identification (BFI) framework emerged as a promising candidate for fault diagnosis where the fault and system parameters can be identified simultaneously. This thesis formulates the fault identification problem in the BFI framework for sensor fault diagnosis. The BFI approach inherently relies on a priori knowledge of the subspace encompassing all potential fault scenarios, thus necessitating the imposition of sparsity constraints to limit the simultaneous occurrence of specific faults. Upon verification, BFI successfully detects and isolates the fault even under unaccounted perturbations.

The central focus of this thesis lies in providing an alternate reliable method of fault diagnosis in sensors. Nevertheless, it also contributes by expanding the sketching-based rank-constrained framework by incorporating l_0 norm approximations to enforce sparsity. The empirical findings validate the efficacy of the BFI framework and the algorithm in detecting the presence of faults and estimating their magnitude.

It is crucial to differentiate between the concept of BFI and the impact of the chosen optimization algorithm on the outcomes. For instance, the reliance of the selected algorithm, Recursive Importance Sketching for Rank-Constrained Optimisation (RISRO), on linear least squares regression necessitates well-conditioned Toeplitz matrices, which is otherwise not essential for convex approaches.

The framework is still in its early stages and offers opportunities for improvement. Recommendations for future research in this direction are elaborated in the next section.

6-2 Recommendations for Future Work

Currently, the BFI strategy primarily applies to structured faults where prior knowledge of the fault dictionary subspace is assumed. Attempts were made to extend fault diagnosis to unknown faults using a fault dictionary composed of Rectified Linear Unit (RelU) basis functions, which can approximate functions [3]. However, it was observed that a sufficiently rich set of RelU functions could potentially model the fault and the system's output, leading to trivial solutions for the system parameters. This also brings to the fore an essential discussion about ascertaining additional conditions required for the fault to be diagnosed, apart from the linear independence of the columns.

Another critical aspect is determining the fault start time without assuming prior knowledge within the fault dictionary. Some faults may also cease to exist during operation, necessitating their detection to prevent false alarms. The solution proposed in this thesis is limited to bias and drift faults. Extending the computation of a fault's start and end time when combining various signals in the dictionary, such as sinusoidal signals with varying frequencies, remains an open research area.

Lastly, the RISRO algorithm relies on an inner-product formulation between the Toeplitz matrices and the regressor, which confines the implementation in this thesis to scalar output signals. Extending this framework to multi-output systems and considering multiple rank constraints on the optimization variable is a necessary avenue for further research.

Appendix A

Additional Results

A-1 Impact of Sparsity Constraint in Magnitude Estimation

This section highlights supplementary results that emphasize the contrast in estimation error and fault isolation rate with and without the incorporation of sparsity in the Blind Fault Identification (BFI) algorithm.

The fault magnitude estimation error analysis has been averaged over 100 simulations of varying fault magnitudes and for different numbers of active faults.

The fault dictionary comprising $n_z = 7$ fault signals is tabulated in Table 5-2 and represented graphically in Figure 5-2. The input and output parameters considered are real flight data in Chapter 5, detailed in Table 5-1. In the experiments below, the model order is $s = 4$.

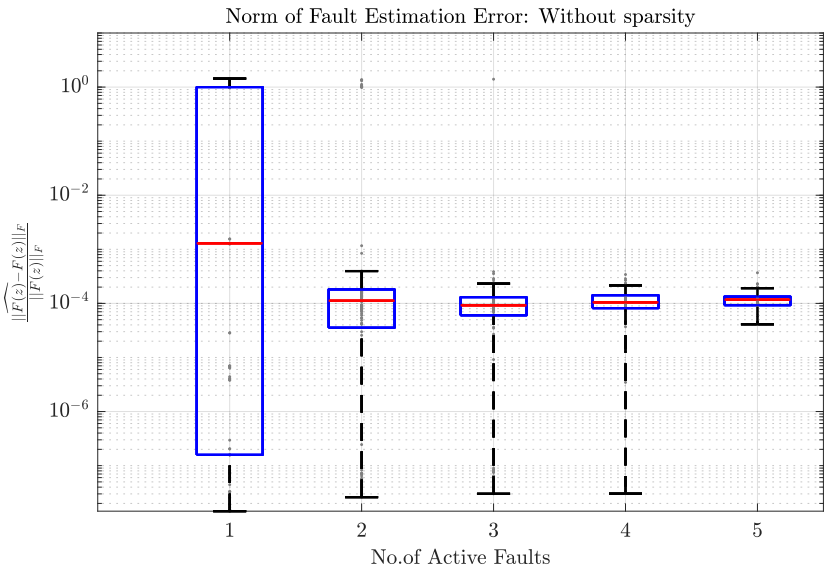


Figure A-1: Magnitude Estimation error with only rank-constrained least squares, RISRO. As the count of active faults decreases, the estimation error tends to be relatively greater compared to situations with a higher number of active faults.

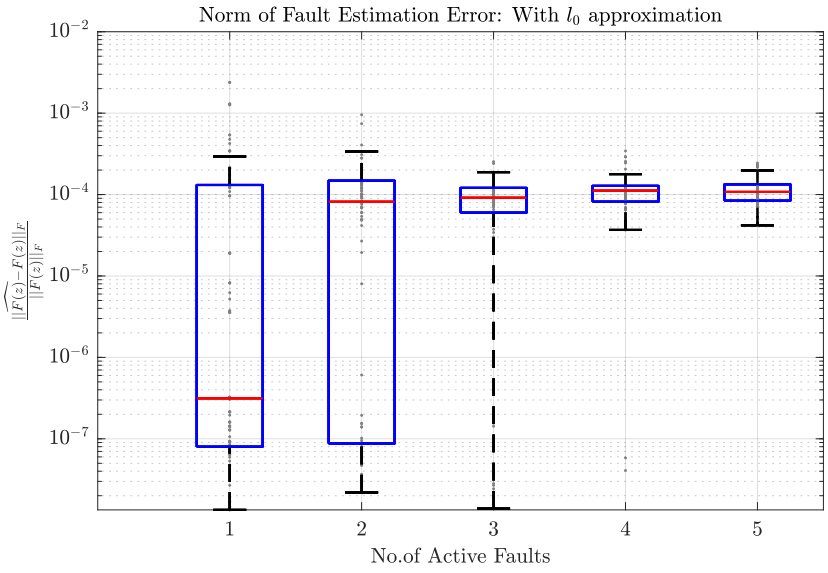


Figure A-2: Magnitude Estimation error with rank and sparsity constrained least squares. Sparsity constraint is imposed using the Pseudo-Measurement (PM) stage of a constrained Kalman Filter (KF) with $M = 1$, $\alpha = 1$, $P = I_{n_z \times n_z}$ and $R = 10$ in Algorithm 3. In contrast to the previous graph, with fewer active faults, the accuracy of magnitude estimation is higher and decreases with an increase in active fault count.

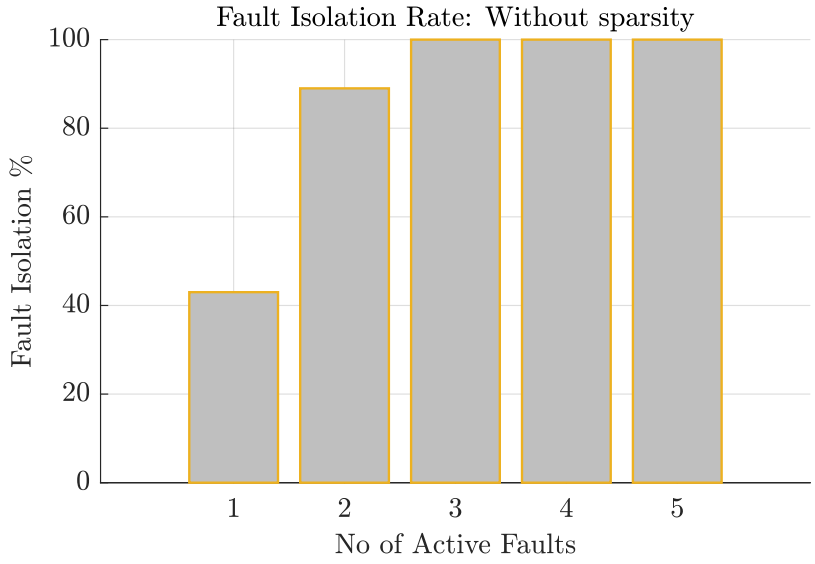


Figure A-3: In line with the magnitude estimation error, the fault isolation improves with an increase in the number of faults. This is directly attributed to the non-sparse solution presented without sparsity constraints.

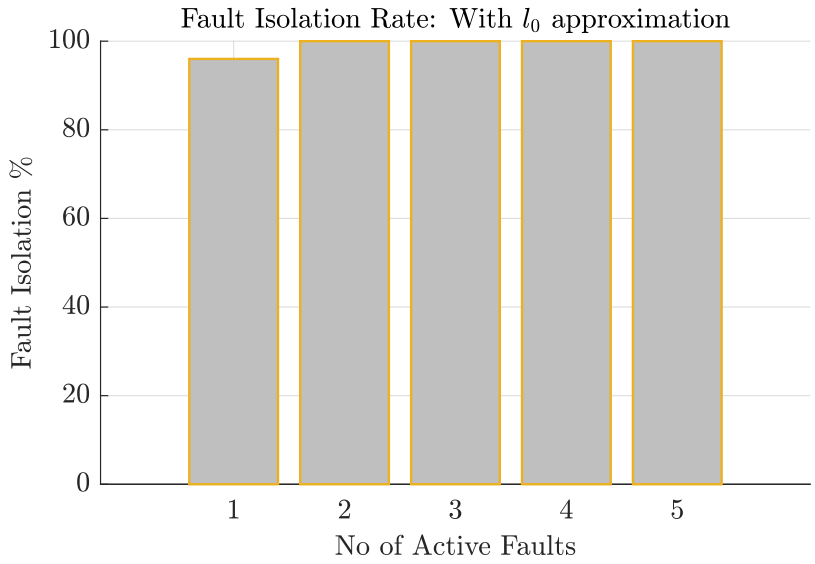


Figure A-4: The Fault Isolation Rate enhances compared to the preceding plot, supporting the notion that sparsity is necessary for a reduced number of active faults.

Appendix B

Software Implementation

The scripts employed in the experiments have been documented in this section. The codes have been written using MATLAB 2021a and do not employ any additional toolboxes.

Filename	Description
`BFI_DataGen.m`	Loads Models(for simulated data), real data and defines Fault Dictionary based on user inputs.
`BFI_FaultIdent_MC.m`	Conducts Monte Carlo Simulations of 100 different fault magnitudes for varying numbers of active faults and reports average performance for chosen model and dictionary.
`BFI_FaultIdent_Spec.m`	Reports the analysis carried out on a specific fault with perturbations for varying values of s.
`BFI_SOF_Det.m`	Time of Start of Fault Detection using time-shifted dictionary signals for a bias fault and dictionary
`risro_fd_sparse2.m`	Algorithm based on RISRO and constrained KF sparsity norm, as detailed in Algorithm 3

Table B-1: Summary of included files and their descriptions

B-1 Script `BFI_DataGen.m`

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Title          : BFI_DataGen.m
% Description    : Load Models, Data and define Data Dictionary
% Author         : Nirupama Sai Ramesh(5320402)
% Date Created   : 7th August, 2023
% Mentors        : Prof. Michel Verhaegen, Jacques Noom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
8
9  %% Cleaning Environment
10
11 close all;
12 clear all;
13 clc;
14
15 %% Sample Discrete System and Data Generation:
16
17 prompt0 = "Enter"+newline+"#1 for Simulated Longitudinal Data"+newline
       +"#2 for FCR"...
18              +newline+"#3 for Short-Period Simulated Data"+newline;
19 load_data = input(prompt0);
20
21 dt      = 0.01;
22
23 if load_data ==1
24
25     load('Cessna_ct_V100.mat');
26     t_end       = 20;                                          % in seconds
27     t           = 0:dt:t_end;
28     N           = length(t);
29
30     A           = Cessna_ct.A;
31     B           = Cessna_ct.B;
32     C           = [0,1,0,0];
33     D           = 0;
34     Cessna_sys  = ss(A,B,C,D);
35     Cessna_dt   = c2d(Cessna_sys,dt,'tustin');
36
37     obs         = rank([Cessna_dt.C;
38         Cessna_dt.C*Cessna_dt.A;
39         Cessna_dt.C*(Cessna_dt.A^2);
40         Cessna_dt.C*(Cessna_dt.A^3)]);
41
42     Amp         = .5;
43     u_3211      = [zeros(1,20),Amp*ones(1,300),-Amp*ones(1,200),Amp*ones
           (1,100),-Amp*ones(1,100),zeros(1,100)];
44     inp_u       = [u_3211,zeros(1,N-length(u_3211))];
45
46     [y,~,x]     = lsim(Cessna_dt,inp_u,t);
47     fn_dt_add = '_Sim';
48     nx = 4;
49
50 elseif load_data ==2
51
52     load('FCRData/r1.mat');
53     t           = tsim'-tsim(1);
54     N           = length(t);
55     t           = t(1:N);
56     inp_u       = delE(1:N);
57     y           = VTAS(1:N);
58     fn_dt_add   = '_FCR';
```

```
59      nx          = 4;
60
61  elseif load_data ==3
62
63      load('Cessna_sp_V100.mat');
64      t_end       = 100;                                          % in seconds
65      t           = 0:dt:t_end;
66      N           = length(t);
67
68      A           = Cessna_sp.A;
69      B           = Cessna_sp.B;
70      C           = [0,1];
71      D           = 0;
72      Cessna_sys  = ss(A,B,C,D);
73      Cessna_dt   = c2d(Cessna_sys,dt,'tustin');
74
75      obs         = rank([Cessna_dt.C;
76          Cessna_dt.C*Cessna_dt.A;
77          Cessna_dt.C*(Cessna_dt.A^2);
78          Cessna_dt.C*(Cessna_dt.A^3)]);
79
80      Amp         = .5;
81      u_3211      = [zeros(1,20),Amp*ones(1,300),-Amp*ones(1,200),Amp*ones
            (1,100),-Amp*ones(1,100),zeros(1,100)];
82      inp_u       = [u_3211,zeros(1,N-length(u_3211))];
83
84      [y,~,x]     = lsim(Cessna_dt,inp_u,t);
85      fn_dt_add   = '_Sim';
86      nx          = 2;
87
88  end
89
90  ny          = 1;
91  nu          = 1;
92
93  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
94
95  %% Fault Dictionary Definition
96
97  prompt1 = "What should the fault dictionary comprise of?";
98  prompt1 = prompt1 + newline + "4. Mixed Structured - 7 signals"+ newline
        +...
99          "5. Mixed Structured - 3 signals"+newline;
100 f_dict = input(prompt1);
101
102 if f_dict == 4
103     theta1              = [zeros(1,20),sin(4.5*pi*t(21:end))];
104     theta2              = [zeros(1,ceil(N/2.3)),ones(1,N-ceil(N/2.3))];
105     theta3              = [zeros(1,ceil(N/5)),sin(2.3*pi*t(ceil(N/5)+1:
            end))];
106     theta4              = [zeros(1,ceil(N/3)),cos(2.25*pi*t(ceil(N/3)+1:
            end))];
```

```
107     theta5              = [zeros(1,ceil(N/6)),sin(0.7*pi*t(ceil(N/6)+1:
            end))];
108     theta6              = [zeros(1,ceil(N/4)),0.23*t(1:N-floor(N/4))];
109     theta7              = [zeros(1,ceil(N/5)),ones(1,N-ceil(N/5))];
110
111     nz = 7;
112
113 elseif f_dict == 5
114     theta1              = [zeros(1,500),ones(1,N-500)];
115     theta2              = [zeros(1,200),0.05*t(1:N-200)];
116     theta3              = [zeros(1,150),sin(4.5*pi*t(151:end))];
117
118     nz = 3;
119     Dict = '5_ThreeDict';
120
121 end
122
123 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B-2 Script BFI_FaultIdent_MC.m

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Title          : BFI_FaultIdent_MC.m
3  % Description    : Average Performance of BFI using Monte Carlo
       Simulations
4  % Author         : Nirupama Sai Ramesh(5320402)
5  % Date Created   : 7th August, 2023
6  % Mentors        : Prof. Michel Verhaegen, Jacques Noom
7  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8
9  %%
10 % 1. Use BFI_FSimDataGen.m to create simulated data or load real flight
       data.
11 % 2. Change the number of theta and value of 's' in the next section to
12 % match the laoded data dictionary and model order.
13 % 3. Run BFI_FaultIdent_MC.m to produce plots for Monte-Carlo Simulations
       .
14
15 %%
16 clear theta_k;
17 clear T_th_i T_u
18 clear T_th_k T_yf;
19
20 s = 4;
21
22 % Comment out theta not in the dictionary or add more if required.
23 for itr=1:1:length(y)
24     theta_temp              = [theta1(itr),...
25                                theta2(itr),...
26                                theta3(itr),...
```

```
27 %                                    theta4(itr),...
28 %                                    theta5(itr),...
29 %                                    theta6(itr),...
30 %                                    theta7(itr),...
31                                     ];
32
33     theta_k(itr,:)              = theta_temp;
34 end
35
36 T_th_k       = theta_k(s+1:end,:);
37
38 for itr = 1:1:(N-s)
39             temp          = flip((theta_k(itr:itr+s-1,:))',2);
40             T_th_i(itr,:) = temp(:)';
41 end
42
43 for itr = 1:1:(N-s)
44             T_u(itr,:) = flip(inp_u(itr:itr+s-1));
45 end
46
47 %% Fault Diagnosis
48
49 col_list     = ['#0072BD','#D95319','#EDB120','#7E2F8E','#77AC30',...
50                 '#4DBEEE','#A2142F','#C71585','#48CCCD','#D296BA'];
51
52 % % % Parameter Setting % % %
53
54 acf_iter         = nz-1;
55 num_iter         = 100;
56 f_mag_max        = 10;
57 Z_err            = zeros(num_iter,acf_iter);
58 Z_isol           = zeros(num_iter,acf_iter);
59 VAF_fault        = zeros(num_iter,acf_iter);
60 RMSE_fault       = zeros(num_iter,acf_iter);
61
62
63 nf_max           = nz;
64
65 for acf = 1:1:acf_iter
66
67     nd = [ones(1,acf),zeros(1,nf_max-acf)];
68
69     for nr = 1:1:num_iter
70
71         % % % Fault Magnitudes % % %
72
73         z        = randi(f_mag_max,[nz,1]);
74         nf       = nd(randperm(nz));
75          z       = z.*nf';
76
77
78         % % % Fault Induction % % %
79         for itr=1:1:length(y)
```

```
80          fault_tru(itr,:)        = theta_k(itr,:)*z;
81          yf(itr,:)               = y(itr,:) + fault_tru(itr,:);
82      end
83
84      yfm     = yf;
85
86
87      %% Construction of Toeplitz matrices of input and output:
88
89      for itr = 1:1:(N-s)
90          temp        = flip((yfm(itr:itr+s-1,:))',2);
91          T_yf(itr,:) = temp(:)';
92      end
93
94      Yf          = yfm(s+1:end,:);
95      T_th_k      = theta_k(s+1:end,:);
96
97      %% RISRO for solving the rank constrained least squares
98
99      p1          = nz+1;
100     p2          = s+1;
101     A_T         = zeros(p1,p2,(N-s));
102
103     for itr = 1:1:(N-s)
104         temp_th_i       = -flip((theta_k(itr:itr+s-1,:))',2);
105         temp_y          = flip((yf(itr:itr+s-1,:))',2);
106         temp_th         = flip((theta_k(itr+s,:))',2);
107         A_T(:,:,itr)    = [temp_th_i,   temp_th;
108                            temp_y,          0];
109     end
110
111     % % % RISRO Initialisation % % %
112     r_app           = 1;
113     [X0,U0,S0,V0]   = X0_init(r_app,[T_yf -T_th_i T_th_k], yf(s+1:end
            ), p1, p2);
114
115     % % % RISRO ALgorithm Call % % %
116     n_iter          = 50;
117     Xt              = zeros(p1,p2,n_iter);
118     dimB            = nu*s;
119     [TB,Xt1]        = risro_fd_sparse2(n_iter,U0,V0,A_T,Yf,r_app,p1,
            p2,Xt,T_u,dimB,nf_max);
120     fn_algo_add     = '_Sp';
121
122     err_val         = zeros(n_iter+1,1);
123     X_str           = z;
124
125     for iter = 1:1:n_iter+1
126         if iter == 1
127             err_val(iter,1) = norm(X0(1:end-1,end)-X_str,'fro')/norm(
                    X_str,'fro');
128         else
```

```
129                 err_val(iter,1) = norm(Xt1(1:end-1,end,iter-1)-X_str,'fro
                        ')/norm(X_str,'fro');
130             end
131         end
132
133         %% Error and Fault Isolation Analysis
134         Xhat        = Xt1(:,:,end);
135         Zhat        = Xt1(1:end-1,end,end);
136
137
138         [U,S,V]     = svd(Zhat','econ');
                                                % normalised value of z
139         zabs        = abs(V)>0.001;
                                                    % 0.001= tunable
                threshold for FIR
140         Zhat_i      = zabs.*Zhat;
141
142         if(logical(Zhat_i)==logical(z))
143             Z_isol(nr,acf) = 1;
144         else
145             Z_isol(nr,acf) = 0;
146         end
147
148         fault_hat       = T_th_k*Zhat_i;
149         fault_gt        = T_th_k*z;
150
151         Z_err(nr,acf)   = norm(Zhat_i-z,'fro')/norm(z,'fro');
152         [VAF_fault(nr,acf),RMSE_fault(nr,acf)] = VAF_RMSE(fault_hat,
                fault_gt);
153     end
154 end
155
156 %% Plot: Norm of Magnitude Estimation Error
157 figure(1);
158 for acf = 1:1:acf_iter
159     semilogy(acf,Z_err(:,acf),'.','Color',[0.5,0.5,0.5],'MarkerSize',5);
            grid on;
160     hold on;
161     boxplot(Z_err,'symbol','');
162     hold on;
163     hline = findobj(gca,'Type','line');
164     set(hline, 'LineWidth', 1);
165 end
166
167 ylabel('$\frac{||\widehat{F(z)}-{F(z)}||_F}{||F(z)||_F}$','Interpreter','
        Latex','FontSize',24);
168 xlabel('No.of Active Faults','FontSize',16);
169 title('Norm of Fault Estimation Error: Without sparsity','FontSize',16);
170
171 hfig = figure(1);
172 picturewidth = 20;
173 hw_ratio = 0.65; % height-width ratio
174 set(findobj(gca,'type','line'),'linew',1.2)
```

```
175 set(findall(hfig,'-property','FontSize'),'FontSize',12);
176 set(findall(hfig,'-property','Interpreter'),'Interpreter','Latex');
177 set(findall(hfig,'-property','TickLabelInterpreter'),'
    TickLabelInterpreter','Latex');
178 set(hfig,'Units','centimeters','Position',[3 3 picturewidth hw_ratio*
    picturewidth]);
179 pos = get(hfig,'Position');
180 set(hfig,'PaperPositionMode','Auto','PaperUnits','centimeters','PaperSize
    ',[pos(3) pos(4)]);
181
182 %% Plot: Fault Isolation Rate
183 Z_isol_sum = sum(Z_isol,1);
184 figure(2);
185 hold on;
186 grid on;
187  bar(Z_isol_sum,'EdgeColor','#EDB120','FaceColor',[0.75 .75 .75],'
     LineWidth',1)
188 xticks([1 2 3 4 5]);
189 xlabel('No of Active Faults');
190 ylabel('Fault Isolation $\%$');
191 title('Fault Isolation Rate: Without sparsity');
192
193 hfig = figure(2);
194 picturewidth = 15;
195 hw_ratio = 0.65; % height-width ratio
196 set(findobj(gca,'type','line'),'linew',1.2)
197 set(findall(hfig,'-property','FontSize'),'FontSize',12);
198 set(findall(hfig,'-property','Interpreter'),'Interpreter','Latex');
199 set(findall(hfig,'-property','TickLabelInterpreter'),'
    TickLabelInterpreter','Latex');
200 set(hfig,'Units','centimeters','Position',[3 3 picturewidth hw_ratio*
    picturewidth]);
201 pos = get(hfig,'Position');
202 set(hfig,'PaperPositionMode','Auto','PaperUnits','centimeters','PaperSize
    ',[pos(3) pos(4)]);
203 %%
204 %%%%%%%%%%%%%%%%%%%%%% Definition of Functions %%%%%%%%%%%%%%%%%%%%%%
205
206 % % % % % % % % % % Risro Initialisation Function % % % % % % % % % % %
    %
207 % Desc      : Initialisation using truncated spectral initialisation
208 % r_app     : Rank Approximation
209 % A         : Subspace of regressor
210 % y         : Observation signal/ measurement
211 % n         : Signal length
212 % p1,p2     : Dimension of X (model order, number of disturbance vectors)
213
214 function [X0,U0,S0,V0]  = X0_init(r_app, A, y, p1, p2)
215
216 n               = length(y);
217 temp            = [y'*A,n];
218 X0              = reshape(temp,[p1,p2])/n;  % A' for adjoint of A
219
```

```
[U0,S0,V0]          = svd(X0);
U0                  = U0(:,1:r_app);
V0                  = V0(:,1:r_app);
S0                  = S0(1:r_app,1:r_app);

end
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% % % % % % % % VAF and RMSE Calculation Function % % % % % % % % % % %
% Desc   : VAF Calculation for comparing estimated output with true output
% y_hat : estimated output from the model
% y_gt   : actual ground truth output
function[VAF,N_rms]     = VAF_RMSE(y_hat,y_gt)
N        = size(y_gt,1);
err      = y_hat - y_gt;
ey_per   = 1 - ((norm(err,2)^2)/(norm(y_gt,2)^2));
VAF      = max(0,ey_per*100);
RMSE     = sqrt((norm(err,2)^2)/N);
GT_RMS   = sqrt(norm(y_gt,2)/N);
N_rms    = RMSE/(GT_RMS);
end
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

B-3 Script BFI_FaultIdent_Spec.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Title          : BFI_FaultIdent_Spec.m
% Description    : To generate data for identification and validation
% Author         : Nirupama Sai Ramesh(5320402)
% Date Created   : 3rd April, 2023
% Mentors        : Prof. Michel Verhaegen, Jacques Noom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
% 1. Use BFI_FSimDataGen.m to create simulated data or load real flight data.
% 2. Change the number of theta and value of magnitude vector, z in the
% next section to match loaded dictionary.
% 3. Run BFI_SpecFault.m

% ** To run for precise sepicific fault, specific 's', comment lines 65,67
% ** 155-165. Set value of parameter 's' in line 67
%% Faul Magnitude Setting
% % % Parameter Setting % % %
% % % Fault Magnitudes % % %
z                        = [  0;
                              4;
                              0;
                              0;
```

```
                                0;
                                0;
                                0;
                                ];
nz = length(z);
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%% Fault Injection to True Measurement

% Change the number of theta based on chosen dictionary
for itr=1:1:length(y)

    theta_temp              = [  theta1(itr),...
                                 theta2(itr),...
                                 theta3(itr),...
                                 theta4(itr),...
                                 theta5(itr),...
                                 theta6(itr),...
                                 theta7(itr),...
                                ];
    theta_k_2               = theta_temp';
    theta_k(itr,:)          = theta_k_2(:)';
    fault_tru(itr,:)        = theta_temp*z;
    yf(itr,:)               = y(itr,:) + 1*fault_tru(itr,:);
end

% Introduce Perturbations for Imprecise Fault
amp_dist         = 0.2;
fault_pert       = (amp_dist*[zeros(1,1200),cos(4.5*pi*t(1201:end))]);
fault_mod        = fault_tru + fault_pert';
yf               = yf+fault_pert';
yfm              = yf;

%% Plot for Real and fault-induced Data

hfig = figure(3);
subplot(211);
plot(t,delE,'LineWidth',1.5);
grid on;
ylabel('Elevator Input, $\delta_e$ [rad]','Interpreter','Latex','FontSize'
    ',14);
xlabel('Time [s]');
xlim([0 t(end)]);
ylim([min(delE)-0.01 max(delE)+0.01]);
ax = gca;
ax.XAxis.LineWidth = 0.75;
ax.YAxis.LineWidth = 0.75;
subplot(212);
plot(t,VTAS,'Color','#77AC30','LineWidth',1.25,'DisplayName','true output
    , $y_{true}$');
xlim([0 t(end)]);
% ylim([70 95]);
grid on; hold on;
```

```
75  plot(t,yf,'Color','#A2142F','LineWidth',1.25,'DisplayName','faulty
        measurement, $y_{m}$');
76  ylabel('True Airspeed, $V_{TAS}$ [m/s]');
77  xlabel('Time [s]','Interpreter','Latex','FontSize',13);
78  % legend('true output, $y_{true}$','faulty measurement, $y_{m}$','
        Interpreter',...
79  %         'Latex','FontSize',13,'Location','Best');
80  sgtitle('Cessna Citation II Real Data');
81  % legend('Location','best','Box','off');
82  ax = gca;
83  ax.XAxis.LineWidth = 0.75;
84  ax.YAxis.LineWidth = 0.75;
85
86  picturewidth = 20;
87  hw_ratio = 0.65; % height-width ratio
88  set(findall(hfig,'-property','FontSize'),'FontSize',14);
89  set(findall(hfig,'-property','Interpreter'),'Interpreter','Latex');
90  set(findall(hfig,'-property','TickLabelInterpreter'),'
        TickLabelInterpreter','Latex');
91  set(hfig,'Units','centimeters','Position',[3 3 picturewidth hw_ratio*
        picturewidth]);
92  pos = get(hfig,'Position');
93  set(hfig,'PaperPositionMode','Auto','PaperUnits','centimeters','PaperSize
        ',[pos(3) pos(4)]);
94
95  %% Construction of Toeplitz matrices of input and output:
96
97  % close all;
98  % For loop for performance for varying 's' for imprecise fault
99  clear s_arr, nr= 0;
100
101 for i = 3:1:80
102
103     clear T_u T_yf T_y T_th_i A_T
104     s       = i;
105
106     for itr = 1:1:(N-s)
107         T_u(itr,:) = flip(inp_u(itr:itr+s-1));
108     end
109
110     for itr = 1:1:(N-s)
111         temp        = flip((yfm(itr:itr+s-1,:))',2);
112         T_yf(itr,:) = temp(:)';
113     end
114
115     for itr = 1:1:(N-s)
116         temp        = flip((y(itr:itr+s-1,:))',2);
117         T_y(itr,:) = temp(:)';
118     end
119
120     for itr = 1:1:(N-s)
121         temp        = flip((theta_k(itr:itr+s-1,:))',2);
122         T_th_i(itr,:) = temp(:)';
```

```
123     end
124
125     Yf          = yfm(s+1:end,:);
126     Y           = y(s+1:end,:);
127     T_th_k      = theta_k(s+1:end,:);
128
129     %% RISRO for solving the rank constrained least squares
130
131     p1      = nz+1;
132     p2      = s+1;
133     A_T     = zeros(p1,p2,(N-s));
134
135     for itr = 1:1:(N-s)
136         temp_th_i       = -flip((theta_k(itr:itr+s-1,:))',2);
137         temp_y          = flip((yfm(itr:itr+s-1,:))',2);
138         temp_th         = flip((theta_k(itr+s,:))',2);
139         A_T(:,:,itr)    = [temp_th_i,   temp_th;
140             temp_y,         0];
141     end
142
143     % % % RISRO Initialisation % % %
144     r_app           = 1;
145     [X0,U0,S0,V0]   = X0_init(r_app,[T_yf -T_th_i T_th_k], yfm(s+1:end),
            p1, p2);
146
147     % % % RISRO ALgorithm Call % % %
148     n_iter          = 50;
149     Xt              = zeros(p1,p2,n_iter);
150     dimB            = nu*s;
151     nf_max          = nz;
152     [TB,Xt1]        = risro_fd_sparse2(n_iter,U0,V0,A_T,Yf,r_app,p1,p2,Xt
            ,T_u,dimB,nf_max);
153     fn_algo_add     = '_Sp';
154
155     err_val = zeros(n_iter+1,1);
156     X_str = z;
157     for iter = 1:1:n_iter+1
158         if iter == 1
159             err_val(iter,1) = norm(X0(1:end-1,end)-X_str,'fro')/norm(
                    X_str,'fro');
160         else
161             err_val(iter,1) = norm(Xt1(1:end-1,end,iter-1)-X_str,'fro')/
                    norm(X_str,'fro');
162         end
163     end
164
165     %% Estimates of Fault Magnitude
166     Zhat    = Xt1(1:end-1,end,end);
167     err_val(end);
168     Xhat    = Xt1(:,:,end);
169     Zhat    = Xt1(1:end-1,end,end);
170
171     %% Computing Estimate of measurement
```

```
172     clear yhat;
173     for ir = 1:1:length(A_T)
174         yhat(ir) = T_u(ir,:)*TB+sum(A_T(:,:,ir).*Xhat,'all');
175     end
176 
177     %% Computing normalised 'z' using SVD for Fault Isolation
178     fault_hat   = T_th_k*Zhat;
179     fault_gt    = fault_mod(s+1:end);
180 
181     [U,S,V]     = svd(Zhat','econ');
182     zabs        = abs(V)>0.01;
183     Zhat_i      = zabs.*Zhat;
184 
185     %% Fault and Measurement VAF Computation
186     [VAF_fault,~] = VAF_RMSE(fault_hat,fault_gt);
187     [VAF_y,~] = VAF_RMSE(yhat,Yf');
188 
189     %% Populating Analysis Metrics for different 's'
190     nr = nr+1;
191     s_arr(nr,1)=s;
192     s_arr(nr,2)=VAF_fault;
193     s_arr(nr,3)=err_val(end);
194     s_arr(nr,4)=sum(zabs);
195     s_arr(nr,5)=VAF_y;
196 
197 end
198 %% Plots for analysis against 's'
199 hfig = figure(7);
200 
201 subplot(311)
202 grid on; hold on;
203 plot(s_arr(:,1),s_arr(:,2),'.-','Color','#0072BD','LineWidth',0.75,'
        MarkerSize',10);
204 ylabel('$VAF_{fault}\; [\%]$');
205 ax = gca;
206 ax.XAxis.LineWidth = 0.75;
207 ax.YAxis.LineWidth = 0.75;
208 ylim([0 102]);
209 
210 subplot(312)
211 semilogy(s_arr(:,1),s_arr(:,3),'.-','Color','#0072BD','LineWidth',0.75,'
        MarkerSize',10);
212 grid on; hold on; grid minor;
213 ylabel('$\frac{||\hat{F(z)}-{F(z)}||_F}{||F(z)||_F}$');
214 yticks([10^-2 10^-1 10^0]);
215 ax = gca;
216 ax.XAxis.LineWidth = 0.75;
217 ax.YAxis.LineWidth = 0.75;
218 
219 subplot(313)
220 grid on; hold on;
221 stem(s_arr(:,1),s_arr(:,4),'.-','Color','#0072BD','LineWidth',0.5,'
        MarkerSize',15);
```

```
222 ylabel(' No. of faults det');
223 xlabel('$s$ [-]');
224 sgtitle('Bias Fault perturbed with $\kappa = 0.2$ detection performance')
    ;
225
226 picturewidth = 20;
227 hw_ratio = 0.75; % height-width ratio
228 set(findall(hfig,'-property','FontSize'),'FontSize',14);
229 set(findall(hfig,'-property','Interpreter'),'Interpreter','Latex');
230 set(findall(hfig,'-property','TickLabelInterpreter'),'
    TickLabelInterpreter','Latex');
231 set(hfig,'Units','centimeters','Position',[3 3 picturewidth hw_ratio*
    picturewidth]);
232 pos = get(hfig,'Position');
233 set(hfig,'PaperPositionMode','Auto','PaperUnits','centimeters','PaperSize
    ',[pos(3) pos(4)]);
234
235 %%
236 %%%%%%%%%%%%%%%%%%%%%%% Definition of Functions %%%%%%%%%%%%%%%%%%%%%%%
237
238 % % % % % % % % % Risro Initialisation Function % % % % % % % % % %
239
240 % Desc      : Initialisation using truncated spectral initialisation
241 % r_app     : Rank Approximation
242 % A         : Subspace of regressor
243 % y         : Observation signal/ measurement
244 % n         : Signal length
245 % p1,p2     : Dimension of X (model order, number of disturbance vectors)
246 function [X0,U0,S0,V0]  = X0_init(r_app, A, y, p1, p2)
247 n = length(y);
248 temp = [y'*A,n];
249 X0              = reshape(temp,[p1,p2])/n;  % A' for adjoint of A
250 % X0              = randn(p1,p2);
251 [U0,S0,V0]      = svd(X0);
252 U0              = U0(:,1:r_app);
253 V0              = V0(:,1:r_app);
254 S0              = S0(1:r_app,1:r_app);
255 end
256 % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
257
258
259 % % % % % % % VAF and RMSE Calculation Function % % % % % % % % % %
260
261 % Desc  : VAF Calculation for comparing estimated output with true output
262 % y_hat : estimated output from the model
263 % y_gt  : actual ground truth output
264 function[VAF,N_rms]    = VAF_RMSE(y_hat,y_gt)
265 N       = size(y_gt,1);
266 err     = y_hat - y_gt;
267 ey_per  = 1 - ((norm(err,2)^2)/(norm(y_gt,2)^2));
268 VAF     = max(0,ey_per*100);
269 RMSE    = sqrt((norm(err,2)^2)/N);
270 GT_RMS  = sqrt(norm(y_gt,2)/N);
```

```
N_rms    = RMSE/(GT_RMS);
end
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

B-4 Script BFI_SOF_Det.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Title          : SOF_Det.m
% Description    : Start of Fault Detection using time shifted signals
% Author         : Nirupama Sai Ramesh(5320402)
% Date Created   : 28th August, 2023
% Mentors        : Prof. Michel Verhaegen, Jacques Noom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Fault Magnitude Settin


z = [5];               % Set Fault Magnitude
nz = length(z);

theta1                 = [zeros(1,1500),ones(1,N-1500)];
% theta2                 = [zeros(1,1600),0.05*t(1:N-1600)];

%%

index_st     = 1400;
index_fn     = 1800;
t_batch      = t(index_st:index_fn);
N_batch      = length(t_batch);
inp_u_batch = inp_u(index_st:index_fn);
y_batch      = y(index_st:index_fn);

%% Fault Injection
clear theta_k;

for itr=1:1:length(y)
    theta_temp                 = [theta1(itr)];

    theta_k(itr,:)             = theta_temp;
    fault_tru(itr,:)           = theta_temp*z;
    yf(itr,:)                  = y(itr,:) + fault_tru(itr,:);
end

yf_batch     = yf(index_st:index_fn);
yfm          = yf;
%%
yfm_batch    = yfm(index_st:index_fn);

%%
clear z_hat_nt
clear theta_k_batch
```

```
45 nt = 50:1:N_batch-50;
46
47 % Change Dictionary for every j_ind iteration, shifted by j_ind
48 for j_ind = 5:1:length(nt)
49
50     theta_k_batch(:,1)      = [zeros(1,nt(j_ind)),ones(1,N_batch-nt(j_ind
       ))];
51     %     theta_k_batch(:,1)      = [zeros(1,nt(jj)),0.05*t(1:N_batch-nt(
       jj))];
52
53     %% Construction of Toeplitz matrices of input and output:
54     clear T_u T_yf T_y T_th_i
55     s       = 4;
56
57     for itr = 1:1:(N_batch-s)
58         T_u(itr,:) = flip(inp_u_batch(itr:itr+s-1));
59     end
60
61     for itr = 1:1:(N_batch-s)
62         temp        = flip((yfm_batch(itr:itr+s-1,:))',2);
63         T_yf(itr,:) = temp(:)';
64     end
65
66     for itr = 1:1:(N_batch-s)
67         temp        = flip((y_batch(itr:itr+s-1,:))',2);
68         T_y(itr,:) = temp(:)';
69     end
70
71     for itr = 1:1:(N_batch-s)
72         temp        = flip((theta_k_batch(itr:itr+s-1,:))',2);
73         T_th_i(itr,:) = temp(:)';
74     end
75
76     Yf  = yfm_batch(s+1:end,:);
77     T_th_k      = theta_k_batch(s+1:end,:);
78
79     %% RISRO for solving the rank constrained least squares
80
81     p1      = nz+1;
82     p2      = s+1;
83     A_T     = zeros(p1,p2,(N_batch-s));
84
85     for itr = 1:1:(N_batch-s)
86         temp_th_i       = -flip((theta_k_batch(itr:itr+s-1,:))',2);
87         temp_y          = flip((yf_batch(itr:itr+s-1,:))',2);
88         temp_th         = flip((theta_k_batch(itr+s,:))',2);
89         A_T(:,:,itr)    = [temp_th_i,   temp_th;
90             temp_y,         0];
91     end
92
93     % % % RISRO Initialisation % % %
94     r_app           = 1;
```

```
95      [X0,U0,S0,V0]    = X0_init(r_app,[T_yf -T_th_i T_th_k], yf_batch(s+1:
            end), p1, p2);
96
97      % % % RISRO ALgorithm Call % % %
98      n_iter           = 50;
99      Xt               = zeros(p1,p2,n_iter);
100     dimB             = nu*s;
101     nf_max = nz;
102
103     [TB,Xt1]         = risro_fd_sparse2(n_iter,U0,V0,A_T,Yf,r_app,p1,p2,Xt
            ,T_u,dimB,nf_max);
104     fn_algo_add = '_Sp';
105
106     err_val = zeros(n_iter+1,1);
107     X_str = z;
108     for iter = 1:1:n_iter+1
109         if iter == 1
110             err_val(iter,1) = norm(X0(1:end-1,end)-X_str,'fro')/norm(
                    X_str,'fro');
111         else
112             err_val(iter,1) = norm(Xt1(1:end-1,end,iter-1)-X_str,'fro')/
                    norm(X_str,'fro');
113         end
114     end
115
116     %      semilogy(err_val);
117
118     Xhat = Xt1(:,:,end);
119     K1r = Xhat(nz+1,1);
120     K2r = Xhat(nz+1,2);
121     K3r = Xhat(nz+1,3);
122     K4r = Xhat(nz+1,4);
123     zr  = Xhat(1:nz,end);
124     K1z = Xhat(1:nz,1);
125     K2z = Xhat(1:nz,2);
126     K3z = Xhat(1:nz,3);
127     K4z = Xhat(1:nz,4);
128
129     Zhat = Xt1(1:end-1,end,end);
130
131     for ir=1:1:length(A_T)
132
133         yhat(ir)=T_u(ir,:)*TB+sum(A_T(:,:,ir).*Xhat,'all');
134
135     end
136     z_hat_nt(j_ind,:) = Zhat;
137     [VAF(j_ind,:),~] = VAF_RMSE(yhat,Yf');
138     TBB(:,j_ind) = TB;
139 end
140 %%
141
142 hfig=figure(3);
143 subplot(211);
```

```
plot(t_batch,yf_batch-y_batch,'LineWidth',1.75,'Color','#C71585');hold on;
xlim([14.5 17]);
grid on;
ylabel('Fault Signal [m/s]');

subplot(212);
stem(t_batch(1)+nt*dt,z_hat_nt(:,1),'LineWidth',1.0,'Marker','.',...
    'MarkerSize',10,'Color','#EDB120');hold on;
ylabel('$\widehat{F(z)}$ [-]');
xlabel('Time [s]');
xlim([14.5 17]);
legend('Est using $\theta_1 =$ Bias Signal');
grid on;

hfig = figure(3);
picturewidth = 15;
xlabel('Time [s]');
hw_ratio = 0.65; % height-width ratio
set(findobj(gca,'type','line'),'linew',1.2)
set(findall(hfig,'-property','FontSize'),'FontSize',12);
set(findall(hfig,'-property','Interpreter'),'Interpreter','Latex');
set(findall(hfig,'-property','TickLabelInterpreter'),'TickLabelInterpreter','Latex');
set(hfig,'Units','centimeters','Position',[3 3 picturewidth hw_ratio*picturewidth]);
pos = get(hfig,'Position');
set(hfig,'PaperPositionMode','Auto','PaperUnits','centimeters','PaperSize',[pos(3) pos(4)]);
%%
% % % % % % % % % % Risro Initialisation Function % % % % % % % % % %

% Desc      : Initialisation using truncated spectral initialisation
% r_app     : Rank Approximation
% A         : Subspace of regressor
% y         : Observation signal/ measurement
% n         : Signal length
% p1,p2     : Dimension of X (model order, number of disturbance vectors)
function [X0,U0,S0,V0]  = X0_init(r_app, A, y, p1, p2)
n = length(y);
temp = [y'*A,n];
X0              = reshape(temp,[p1,p2])/n;  % A' for adjoint of A
% X0              = randn(p1,p2);
[U0,S0,V0]      = svd(X0);
U0              = U0(:,1:r_app);
V0              = V0(:,1:r_app);
S0              = S0(1:r_app,1:r_app);
end
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

B-5 Function risro_fd_sparse2.m

```
%% RISRO and Sparsity Function for Blind Fault Identification

function [TB,Xt1] = risro_fd_sparse2(n_iter,U0,V0,A,y,r_app,p1,p2,Xt1,T_u
    ,dimB,nf_fmax)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Returns the estimated X and B from sparse least squares solution
% n_iter        : No. Of RISRO Iterations
% U0,V0         : Initialised SVD of X0
% A             : Regressor ; Known subspace of Signals
% y             : Measured Output
% r_app         : rank constraint
% p1,p2         : Dimensions of X; Model Order, No.of Dist Signals
% T_u           : Toeplitz Matrix of Input Signal
% dimB          : dimensions of B matrix, input dim times model ord
% nf_fmax       : total number of fault (nz)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n   = length(y);
r   = r_app;
Ut  = U0;
Vt  = V0;

for t=1:1:n_iter


    Ut_perp          = null(Ut');
    Vt_perp          = null(Vt');

    tempAB1         = pagemtimes(Ut',A);
    tempAB2         = pagemtimes(tempAB1,Vt);
    AB              = reshape(tempAB2,[r*r,n])';

    tempAD11        = pagemtimes(Ut_perp',A);
    tempAD12        = pagemtimes(tempAD11,Vt);
    AD1             = reshape(tempAD12,[(p1-r)*r,n])';

    tempAD21        = pagemtimes(Ut',A);
    tempAD22        = pagemtimes(tempAD21,Vt_perp);
    AD2             = reshape(tempAD22,[(p2-r)*r,n])';

    A_tot           = [T_u, AB, AD1, AD2];
                                    % Changed to include T_u to get B

    % Solve Dimension Reduced Least square
    opt_var         = (A_tot'*A_tot)\(A_tot'*y);

    TB              = opt_var(1:dimB);
                                        % Changed the access points
       to get B
```

```
47      gamma               = opt_var(dimB+1:end);
48      B                   = gamma(1:r*r);
49      D1                  = gamma((r*r)+1:(r*r)+(p1-r)*r);
50      D2                  = gamma(((r*r)+(p1-r)*r)+1:end);
51
52      Bt1                 = reshape(B,[r,r]);
53      D1t1                = reshape(D1,[p1-r,r]);
54      D2t1                = reshape(D2,[r,p2-r]);
55      D2t1                = D2t1';
56
57      X_ut1               = Ut*Bt1 + Ut_perp*D1t1;
58      X_vt1               = Vt*Bt1' + Vt_perp*D2t1;
59
60      % QR Orthogonalisation to compute U,V
61      [Ut,~]              = qr(X_ut1,0);
62      Ut                  = Ut(:,1:r_app);
63      [Vt,~]              = qr(X_vt1,0);
64      Vt                  = Vt(:,1:r_app);
65
66      % Update step
67      Xt1(:,:,t)          = X_ut1 / Bt1 * X_vt1';
68
69      if t == n_iter
70          break
71      else
72
73          xhat = Xt1(:,:,t);
74          Kobs = xhat(end,1:end-1);
75          z = xhat(1:end-1,end);
76          P = eye(nf_fmax,nf_fmax);
77          R = 10;
78          z_k = zeros(nf_fmax,1);
79          z_k(:,1) = z;
80          n_itr = 1;
81          z_opt = Approx_l0(P,R,z_k,nf_fmax,n_itr);
82
83          xhat_sp = [kron(Kobs,z_opt),z_opt;
84              Kobs,1];
85      end
86      Xt1(:,:,t) = xhat_sp;
87      [Ut,~,Vt] = svd(xhat_sp);
88      Ut = Ut(:,1);
89      Vt = Vt(:,1);
90  end
91
92  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
93  % Choice of sparsity framework - l1, lp and l0 approximation
94  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
95  % Desc : Function to calculate the CSKF-l1 norm
96
97      function z_sparse = CSKF_l1(P,R,z_k,nf_fmax,n_itr)
98          for itr = 1:1:n_itr-1
99
```

```
100             H_bar         = sign(z_k(:,itr))';
101             K             = P*H_bar'*inv(H_bar*P*H_bar' + R);
102             z_k(:,itr+1) = (eye(nf_fmax,nf_fmax)-K*H_bar)*z_k(:,itr);
103             P             = (eye(nf_fmax,nf_fmax)-K*H_bar)*P;
104
105         end
106         z_sparse = z_k(:,end);
107     end
108
109 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111 % Desc : Function to calculate the CSKF-lp norm
112
113     function z_sparse = CSKF_lp(P,R,z_k,nf_fmax,n_itr)
114         p =0.7;
115         for itr = 1:1:n_itr-1
116             z_str   = z_k(:,itr);
117
118             for jtr = 1:1:nf_fmax
119
120                 if z_str(jtr)>0
121                     H_bar(1,jtr) = (sum(abs(z_str).^p)) ^(1/p-1)*(abs(
                            z_str(jtr))^(p-1));
122                 else
123                     H_bar(1,jtr) = -(sum(abs(z_str).^p) ^(1/p-1))*(abs(
                            z_str(jtr))^(p-1));
124                 end
125
126             end
127
128             K = P*H_bar'*inv(H_bar*P*H_bar' + R);
129             z_k(:,itr+1)= z_str - K*norm(z_str,p);
130             P = (eye(nf_fmax,nf_fmax)-K*H_bar)*P;
131
132         end
133         z_sparse = z_k(:,end);
134
135     end
136
137 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
138 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
139 % Desc : Function to calculate the exponential Approximation of L0 norm
140
141     function z_sparse = Approx_l0(P,R,z_k,nf_fmax,n_itr)
142
143         alpha = 1;
144         for itr = 1:1:n_itr-1
145
146             z_str   = z_k(:,itr);
147
148             for jtr = 1:1:nf_fmax
149
150                 if z_str(jtr)>0
```

```
151                     H_bar(1,jtr) = -alpha*exp(-alpha*z_str(jtr));
152                 else
153                     H_bar(1,jtr) = alpha*exp(alpha*z_str(jtr));
154                 end
155
156             end
157
158             K = P*H_bar'*inv(H_bar*P*H_bar' + R);
159             z_k(:,itr+1)= z_str + K*(nf_fmax-sum(exp(-alpha.*abs(z_str)))
                    );
160             P = (eye(nf_fmax,nf_fmax)-K*H_bar)*P;
161         end
162         z_sparse = z_k(:,end);
163     end
164
165 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
166 end
```

Bibliography

[1] Alireza Abbaspour, Sohrab Mokhtari, Arman Sargolzaei, and Kang K. Yen. A Survey on Active Fault-Tolerant Control Systems. *Electronics*, 9(9):1513, September 2020. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[2] K. Abed-Meraim, Wanzhi Qiu, and Yingbo Hua. Blind system identification. *Proceedings of the IEEE*, 85(8):1310–1322, August 1997. Conference Name: Proceedings of the IEEE.

[3] Laurens Bliek, Michel Verhaegen, and Sander Wahls. Online function minimization with convex random relu expansions. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, September 2017.

[4] Avishy Carmi, Pini Gurfil, and Dimitri Kanevsky. Methods for Sparse Signal Recovery Using Kalman Filtering With Embedded Pseudo-Measurement Norms and Quasi-Norms. *IEEE Transactions on Signal Processing*, 58(4):2405–2409, April 2010. Conference Name: IEEE Transactions on Signal Processing.

[5] Bureau d'Enquêtes et d'Analyses pour la Sécurité de l'Aviation Civile. Final Report on the Accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro–Paris. *Tech. Rep. June 2009*, 2012.

[6] EASA. EASA Airworthiniess Directive. Technical 2021-0039R2, European Union Aviation Safety Agency, February 2021. Publisher: easa.eu.

[7] P. Eide and P. Maybeck. An MMAE failure detection system for the F-16. *IEEE Transactions on Aerospace and Electronic Systems*, 32(3):1125–1136, July 1996. Conference Name: IEEE Transactions on Aerospace and Electronic Systems.

[8] Paul Freeman. Robust, Model-based Fault Detection for Commercial Transport Air Data Probes. Master Thesis, The University of Minnesota, 2011.

[9] Paul Freeman, Rohit Pandita, Nisheeth Srivastava, and Gary J. Balas. Model-Based and Data-Driven Fault Detection Performance for a Small UAV. *IEEE/ASME Transactions on Mechatronics*, 18(4):1300–1309, August 2013. Conference Name: IEEE/ASME Transactions on Mechatronics.

[10] Paul Freeman, Peter Seiler, and Gary J. Balas. Air data system fault modeling and detection. *Control Engineering Practice*, 21(10):1290–1301, October 2013.

[11] Nianhong Han, M.A. Siddique, Zichen Zhang, Linchuan Tian, Haiyang Hu, and Hui Hu. A flight-testing campaign to examine inflight icing characteristics and its effects on the flight performance of an Unmanned-Aerial-Vehicle. *Cold Regions Science and Technology*, 207:103775, March 2023.

[12] Anders Hansson, Zhang Liu, and Lieven Vandenberghe. Subspace system identification via weighted nuclear norm optimization. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3439–3444, December 2012. ISSN: 0743-1546.

[13] Haiyang Hu, Faisal Al-Masri, and Hui Hu. An Experimental Study on Ice Accretion and Anti-/De-Icing of a Pitot Tube. In *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, 2022. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2022-1913.

[14] Robert Jäckel, Geydy Gutiérrez-Urueta, and Fidencio Tapia. A review on Pitot tube icing in aeronautics: Research- design and characterization – future trends. *Flow Measurement and Instrumentation*, 81:102033, October 2021.

[15] Zhongzhi Li, Yunmei Zhao, Jinyi Ma, Jianliang Ai, and Yiqun Dong. Fault Detection and Classification of Aerospace Sensors using a VGG16-based Deep Neural Network, July 2022. arXiv:2207.13267 [cs].

[16] P. Lu, L. Van Eykeren, E. van Kampen, and Q. P. Chu. Selective-Reinitialization Multiple-Model Adaptive Estimation for Fault Detection and Diagnosis. *Journal of Guidance, Control, and Dynamics*, 38(8):1409–1424, 2015. Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.G000587.

[17] Peng Lu, Laurens Van Eykeren, Erik-Jan Van Kampen, Coen De Visser, and Qiping Chu. Double-model adaptive fault detection and diagnosis applied to real flight data. *Control Engineering Practice*, 36:39–57, March 2015.

[18] Peng Lu, Erik-Jan van Kampen, Coen de Visser, and Qiping Chu. Air Data Sensor Fault Detection and Diagnosis in the Presence of Atmospheric Turbulence: Theory and Experimental Validation With Real Flight Data. *IEEE Transactions on Control Systems Technology*, 29(5):2255–2263, September 2021. Conference Name: IEEE Transactions on Control Systems Technology.

[19] Peng Lu, Erik-Jan Van Kampen, Coen De Visser, and Q.P. Chu. MATLAB Code for Automatica paper: Framework for state and unknown input estimation of linear time-varying systems, August 2016.

[20] Peng Lu, Erik-Jan van Kampen, Cornelis C. de Visser, and Qiping Chu. Framework for state and unknown input estimation of linear time-varying systems. *Automatica*, 73:145–154, November 2016. arXiv:1606.08090 [cs, math, stat].

[21] Yuetian Luo, Wen Huang, Xudong Li, and Anru R. Zhang. Recursive Importance Sketching for Rank Constrained Least Squares: Algorithms and High-order Convergence, December 2022. arXiv:2011.08360 [cs, math, stat].

[22] Xianglian Lv, Jie Guan, Shengkun Wang, Haiyang Zhang, Shijie Xue, Qi Tang, and Yang He. Pitot Tube-Based Icing Detection: Effect of Ice Blocking on Pressure. *International Journal of Aerospace Engineering*, 2020:e1902053, August 2020. Publisher: Hindawi.

[23] Peter S. Maybeck. Multiple model adaptive algorithms for detecting and compensating sensor and actuator/surface failures in aircraft flight control systems. *International Journal of Robust and Nonlinear Control*, 9(14):1051–1070, 1999. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291099-1239%2819991215%299%3A14%3C1051%3A%3AAID-RNC452%3E3.0.CO%3B2-0.

[24] Yonghao Miao, Boyao Zhang, Jing Lin, Ming Zhao, Hanyang Liu, Zongyang Liu, and Hao Li. A review on the application of blind deconvolution in machinery fault diagnosis. *Mechanical Systems and Signal Processing*, 163:108202, January 2022.

[25] J. Noom, O.A. Soloviev, and M.H.G. Verhaegen. Data-Driven Fault Diagnosis under Sparseness Assumption. *Data-Driven Fault Diagnosis under Sparseness Assumption*, 2022.

[26] MQ Phan and RW Longman. Relationship between state-space and input-output models via observer markov parameters. *WIT Transactions on The Built Environment*, 22, 1970.

[27] Nirupama Sai Ramesh. Blind System Identification using RISRO. Technical, Delft University of Technology, Delft, October 2022.

[28] Nirupama Sai Ramesh. Literature Survey: Survey of Fault Identification Methods for Air Data Sensor. Technical, Delft University of Technology, Delft, March 2023.

[29] Dexter Scobee, Lillian Ratliff, Roy Dong, Henrik Ohlsson, Michel Verhaegen, and S. Shankar Sastry. Nuclear norm minimization for blind subspace identification (N2BSID). In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2127–2132, December 2015.

[30] Filip Sklenář and Jiří Matějů. Indicated airspeed error due to gradual blocking of pitot tube with drain hole. *Aviation*, 26(1):64–71, April 2022. Number: 1.

[31] Tyson Smith. Lecture Notes: Incompressible, Compressible, and Supersonic Flow Fields: Static, Dynamic, and Total Pressure. Lecture Notes, Utah State University, 2018.

[32] Chengpu Yu and Michel Verhaegen. Blind multivariable ARMA subspace identification. *Automatica*, 66:3–14, April 2016.

[33] Chengpu Yu, Lihua Xie, Michel Verhaegen, and Jie Chen. *Blind Identification of Structured Dynamic Systems: A Deterministic Perspective*. Springer Nature, November 2021. Google-Books-ID: D1NQEAAAQBAJ.

[34] Anru R. Zhang, Yuetian Luo, Garvesh Raskutti, and Yuan Ming. ISLET: Fast and Optimal Low-Rank Tensor Regression via Importance Sketching | SIAM Journal on Mathematics of Data Science. *SIAM*, 2(2):444–479, 2020.

[35] Liqing Zhang and Andrzej Cichocki. Blind Deconvolution of Dynamical Systems: A State-Space Approach. *Journal of Signal Processing*, 2000.

[36] Qinghua Zhang. Dynamic System Fault Diagnosis Under Sparseness Assumption. *IEEE Transactions on Signal Processing*, 69:2499–2508, 2021. Conference Name: IEEE Transactions on Signal Processing.

[37] Y. Zhang and X.R. Li. Detection and diagnosis of sensor and actuator failures using IMM estimator. *IEEE Transactions on Aerospace and Electronic Systems*, 34(4):1293–1313, October 1998. Conference Name: IEEE Transactions on Aerospace and Electronic Systems.

[38] Zheng Zhang, Yong Xu, Jian Yang, Xuelong Li, and David Zhang. A Survey of Sparse Representation: Algorithms and Applications. *IEEE Access*, 3:490–530, May 2015.

[39] Yunmei Zhao, Hang Zhao, Jianliang Ai, and Yiqun Dong. Robust Data-Driven Fault Detection: An Application to Aircraft Air Data Sensors. *International Journal of Aerospace Engineering*, 2022:e2918458, March 2022. Publisher: Hindawi.