Autonomous and Resilient Management of All-Source Sensors for Navigation Assurance

Juan D. Jurado

AUTONOMOUS AND RESILIENT MANAGEMENT OF ALL-SOURCE SENSORS FOR NAVIGATION ASSURANCE

DISSERTATION

Juan D. Jurado, Major, USAF

AFIT-ENG-DS-19-S-006

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-DS-19-S-006

AUTONOMOUS AND RESILIENT MANAGEMENT OF ALL-SOURCE SENSORS FOR NAVIGATION ASSURANCE

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy in Electrical Engineering

Juan D. Jurado, B.S.E.E., M.S.E.E, M.S.FTE

Major, USAF

September 2019

AFIT-ENG-DS-19-S-006

AUTONOMOUS AND RESILIENT MANAGEMENT OF ALL-SOURCE SENSORS FOR NAVIGATION ASSURANCE

Juan D. Jurado, B.S.E.E., M.S.E.E, M.S.FTE
Major, USAF

Approved:

[illegible] 3 July 2019

Robert C. Leishman, Ph.D. (Chairman) Date

[illegible] 3 July 2019

John F. Raquet, Ph.D. (Member) Date

[illegible] 3 July 2019

Christine M. Schubert Kabban, Ph.D. (Member) Date

[illegible] 3 July 2019

Donald T. Venable, Ph.D. (Member) Date

Accepted:

[illegible] 9 July 2019

Adedeji B. Badiru Date
Dean, Graduate School of Engineering and Management

AFIT-ENG-DS-19-S-006

Abstract

All-source navigation has become increasingly relevant over the past decade with the development of viable alternative sensor technologies. However, as the number and type of sensors informing a system increases, so does the probability of corrupting the system with sensor modeling errors, signal interference, and undetected faults. Though the latter of these has been extensively researched, the majority of existing approaches have constrained faults to biases, and designed algorithms centered around the assumption of simultaneously redundant, synchronous sensors with valid measurement models, none of which are guaranteed for all-source systems. This research aims to provide all-source multi-sensor resiliency, assurance, and integrity through an autonomous sensor management framework. The proposed framework dynamically places each sensor in an all-source system into one of four modes: monitoring, validation, calibration, and remodeling. Each mode contains specific and novel realtime processes that affect how a navigation system responds to sensor measurements. The monitoring mode is driven by a novel sensor-agnostic fault detection, exclusion, and integrity monitoring method that minimizes the assumptions on the fault type, all-source sensor composition, and the number of faulty sensors. The validation mode provides a novel method for the online validation of sensors which have questionable sensor models, in a fault-agnostic and sensor-agnostic manner, and without compromising the ongoing navigation solution in the process. The remaining two modes, calibration and remodeling, generalize and integrate online calibration and model identification processes to provide autonomous and dynamic estimation of candidate model functions and their parameters, which when paired with the monitoring and validation processes, directly enable resilient, self-correcting, plug-and-play open architecture navigation systems.

Table of Contents

Page

Page

List of Figures

Figure Page

List of Tables

List of Acronyms

Acronym	Definition
ACF	Autocorrelation Function
ARMAS	Autonomous and Resilient Management of All-source Sensors
ARMAV	Autonomous Regression Method for Allan Variance
ADC	Air Data Computer
ADS	Air Data System
AIC	Akaike Information Criterion
AoA	Angle of Attack
AoS	Angle of Sideslip
ASM	Akaike Spline Model
ANT	Autonomy and Navigation Technology
BSEKF	Backwards Smoothing Extended Kalman Filter
DCM	Direction Cosine Matrix
DGPS	Differential Global Positioning System
EKF	Extended Kalman Filter
FOGM	First Order Gauss-Markov
FTLF	F-Test for Lack of Fit
GPS	Global Positioning System
HPL	Horizontal Protection Level
HIL	Horizontal Integrity Limit
IAS	Indicated Airspeed
IEEE	Institute of Electrical and Electronics Engineers
IID	Independent Identically Distributed
IMU	Inertial Measurement Unit

Acronym	Definition
INS	Inertial Navigation System
JMOSS	Jurado-McGehee Online Self-Survey
KF	Kalman Filter
LRT	Likelihood Ratio Test
MLE	Maximum Likelihood Estimation
MMAE	Multiple Model Adaptive Estimation
MMSE	Minimum Mean Squared Error
MSE	Mean Squared Error
MVUE	Minimum Variance Unbiased Estimator
OLS	Ordinary Least Squares
PD	Probability of Detection
PDF	Probability Distribution Function
PF	Probability of False Alarm
PI	Prediction Interval
PSD	Power Spectral Density
APNT	Assured Position Navigation and Timing
RAIM	Receiver Autonomous Integrity Monitoring
RSS	Root Sum Squared
RSR	Resilient Sensor Recovery
ROC	Receiver Operating Characteristic
RVPS	Real-time Validation for Plug-and-play Sensors
SAARM	Sensor-Agnostic All-source Residual Monitoring
SLAM	Simultaneous Localization and Mapping
SPE	Static Position Error
SSE	Sum of Squared Errors

Acronym	Definition
TAS	True Airspeed
TFB	Tower Fly-by
TSE	Taylor Series Expansion
UKF	Unscented Kalman Filter
VINS	Visual-Inertial System
VIF	Variation Inflation Factor
WGN	White Gaussian Noise

AUTONOMOUS AND RESILIENT MANAGEMENT OF ALL-SOURCE SENSORS FOR NAVIGATION ASSURANCE

I. Introduction

Over the past two decades, the United States Air Force has focused on complementing its reliance on the Global Positioning System (GPS) for navigation and timing solutions through the use of alternative navigation sources and sensors. Additionally, senior leaders within the Air Force have recently stated one of the top priorities for the service "to cost-effectively modernize to increase the lethality of the force and drive innovation to secure our future." [104]. With this in mind, the Air Force Institute of Technology's Autonomy and Navigation Technology (ANT) Center has made its vision to provide "defense-focused autonomy and navigation, anywhere, anytime, using anything."

Unlike the well-understood, synchronous, and redundant nature of the GPS multi-sensor constellation, all-source navigation systems tend to be heterogeneous in composition, with each sensor proven only within a well-controlled environment, and not guaranteed to be synchronous or redundant. Additionally, as the number of sensors and measurement domains that are exploited for navigation purposes increases, so does the probability of corrupting the navigation solution with errors in sensor modeling, unexpected signal interference, and or undetected faults. Therefore, in order to fulfill this vision, alternative (non-GPS) all-source navigation technology must be brought up to a level of operational readiness that allows its use in a manner that is resilient, and thus capable of not only detecting when any of the above failure modes are present, but also of assuring navigation integrity in their presence, and self-correcting and recovering from such failures, all in an autonomous, real-time, plug-and-play architecture.

This dissertation details a wide-ranging research effort aimed at solving the "all-source navigation assurance problem," through the development of an online autonomous and resilient sensor management framework. Each component of the framework in focused on solving a portion of the overall problem set, which includes: all-source fault detection and exclusion, integrity monitoring, sensor initialization and validation, online sensor calibration, and online model identification. Existing and related research in each of the problem subsets is briefly summarized below and further expounded in each of its corresponding chapters of this dissertation.

1.1 Summary of Related Research

1.1.1 Sensor Management Frameworks.

As discussed in Chapter 3, sensor management in navigation has been mostly focused on managing sensors from a resources perspective by selecting optimal sensor subsets based on available computational load. Though also important for real-time operational all-source systems, computational load management is not the focus of this research, as it does not address any of the problem subsets identified above. Other research efforts in the area of Visual-Inertial System (VINS) navigation have proposed the use of online statistical tests to trigger camera calibration routines. This line of research addresses a portion of the fault detection and sensor calibration problem subsets, but only focuses on a system with a single sensor, and the statistical tests used are typically custom-tailored for camera systems.

1.1.2 Fault Detection, Exclusion, and Integrity Monitoring.

As shown in Chapter 4, multi-sensor fault detection and exclusion, and integrity monitoring have been extensively researched in the area of GPS navigation. Typically, each satellite in the GPS constellation is regarded as a different (albeit identical in nature and synchronous) sensor in the multi-sensor system, and the "fault" is usually defined as an unmodeled bias that is assumed to only affect one of the sensors (satellites) at any given time, which allows for rigorous computations of system integrity using the probability of a missed (bias) detection. Though adequate for homogeneous and synchronously redundant sensors like GPS, these techniques are inadequate for our research, since the assumptions and conditions necessary for fault detection and system integrity cannot be guaranteed in an all-source environment.

1.1.3 Sensor Initialization and Validation.

Online sensor initialization and validation constitutes a key enabling technology for self-correcting and plug-and-play navigation systems. As discussed in Chapter 5, though the problem is statistically similar to multi-sensor fault detection, the primary objective has traditionally been to provide navigation solution integrity with the assumption that each sensor in the system is equally likely to experience a fault, and that each sensor is properly modeled at the start of the navigation process. These assumptions are invalid for all-source sensors with questionable sensor models that are initialized “on-the-fly”, or sensors that have been previously taken offline and are to be re-initialized after experiencing a fault.

1.1.4 Online Sensor Calibration.

A common technique for improving the resiliency of a particular all-source sensor technology is to provide a means for the online calibration of specific sensor model parameters (e.g., lever arms, rotation matrices, scale factors, etc.). As discussed in Chapter 3, many online calibration methods exist across a variety of sensor technologies. One research area with significant recent advances is online calibration of a VINS. However, as mature as these point-examples may be, they still only tend to focus on a particular sensor, and often do not address the tasks of detecting the need for calibration or independently evaluating the effectiveness of the calibration results.

1.1.5 Online Model Identification.

As described in Chapter 3, another class of existing methods for improving resiliency is to provide a means for the online alteration the sensor model functional form to account for missing parameters or changing environmental conditions (e.g., time-changing biases, stochastic clock errors, temperature effects, etc.). This line of research is typically referred to as multiple-model estimation, or model identification. Current research in this area tends to be divided between continuous estimation of sensor and process noise covariances and model selection from a finite set of competing models. However, most techniques tend to focus on permanent failure modes that require the model identification process to run continuously. Similar to online calibration, these techniques also tend to lack independent validation of model identification results.

1.2 Research Contributions

Having defined the overall all-source resiliency problem set, and the gaps in each of the related problem subsets, we now define the research contributions provided in this dissertation. In general, this research aims to provide all-source multi-sensor resiliency, assurance, and integrity through an autonomous sensor management framework. The proposed framework dynamically places each sensor in an all-source system into one of four modes of operation: monitoring, validation, calibration, and remodeling. Each mode contains specific and novel realtime processes that affect how a navigation system responds to sensor measurements. The monitoring mode is driven by a novel sensor-agnostic fault detection, exclusion, and integrity monitoring method that minimizes the assumptions on the fault type, all-source sensor composition, and the number of faulty sensors. The validation mode provides a novel method for the online validation of sensors which have questionable sensor models, in a fault-agnostic and sensor-agnostic manner, and without compromising the ongoing navigation solution in the process. The remaining two modes (calibration and remodeling) generalize and integrate online calibration and model identification processes to provide autonomous and dynamic estimation of candidate model functions and their parameters. When paired with the monitoring and validation methods, these processes directly enable resilient, self-correcting, plug-and-play open architecture navigation systems. In addition to the overall framework and the novel methods enabling the monitoring and validation modes, this research also contributes two novel online sensor calibration methods for Pitot-static, and inertial measurement sensors, respectively. Both of these calibration methods can be regarded as complementary to the overall research thrust by contributing to the list of available sensors supported in the calibration mode of the framework. The specific research contributions claimed in this dissertation can then be tracked to each problem subset and described as:

(1) **A novel navigation assurance framework that provides:**
   (i) Twelve coherently interlaced resilient sensor management functions,
   (ii) An object-oriented integrity interface for dynamic filter management, and
   (iii) An open system architecture enabling self-correcting navigation systems.

(2) **A novel sensor-agnostic residual monitoring method that provides:**
   (i) Generalized all-source multiple-fault detection and exclusion,
   (ii) Asynchronous and cross-domain detection redundancy, and
   (iii) A robust measure of system integrity without constraining the fault type.

(3) **A novel all-source plug-and-play sensor validation method that enables:**
   (i) Initialization of offline sensors without corrupting the navigation solution,
   (ii) Improved sensor model fault detection over standard residual monitoring, and
   (iii) Protection of system integrity during validation using only a single filter.

(4) **A novel online and complete Air Data System calibration method that:**
   (i) Calibrates four key speed-dependent parameters in a single experiment,
   (ii) Eliminates the need for costly external calibration reference sources,
   (iii) Does not require the need to sustain transonic or supersonic speeds, and
   (iv) Provides a novel smoothing spline model suitable for transonic characterization.

(5) **A novel autonomous, regression-based method for Allan variance analysis that:**
   (i) Meets or exceeds the standard analysis method for usual data lengths,
   (ii) Outperforms the standard analysis method for reduced data lengths, and
   (iii) Eliminates the need for human input across varying sensor types.

1.3 Outline

The remainder of this dissertation is divided into seven additional chapters. Chapter 2 contains the mathematical background and notation used throughout the dissertation as well as a general summary of statistical modeling, model diagnostics, and remedial measures and the methods used for estimating navigation solutions. Given the breadth and depth of topics covered across the aforementioned research contributions, each chapter in Chapters 3 through 7 contains specific background, methodology, results, and conclusions for the subset of topics covered therein. Chapter 3 develops the general framework needed to provide multi-sensor resiliency, assurance, and integrity in an all-source environment. Chapter 4 focuses on all-source fault detection and exclusion and integrity monitoring, which enable the monitoring objective of the overall framework. Meanwhile, Chapter 5 discusses the developments in real-time sensor initialization and validation, which support the framework's validation objective. Chapters 6 and 7 provide the methodology, results, and conclusions for two novel Pitot-static, and inertial sensor calibration algorithms, respectively, which complement the online calibration objective of the overall framework. Finally, Chapter 8 summarizes the entire research effort, major findings and research contributions, and sets the vision for future work in the area of all-source resilient navigation.

II. Background

While subsequent chapters in this dissertation contain specific research background topics needed to understand their contributions, this chapter offers the foundational background needed to understand the concepts of estimation, regression, and detection and how they apply to navigation. Section 2.1 outlines the general notational conventions used throughout this dissertation. Section 2.2 summarizes estimation and detection fundamentals. Section 2.3 describes the expected properties of residual error terms and how they can be used to analyze model performance. Finally, Section 2.4 frames the navigation problem in the context of model regression, while Section 2.5 summarizes how recursive estimation techniques are used to solve the navigation problem.

2.1 Notational Conventions

Scalars: Scalars are represented by either upper or lowercase characters in *italics*, e.g., a or A.

Vectors: Vector quantities are represented by lowercase characters in **bold**, e.g., $\mathbf{a}$. Unless specifically stated otherwise, all vectors should be interpreted as column vectors.

Vector Components: The scalar components of a vector are represented with subscripts indicating their entry, e.g., the k^{th} entry in the vector $\mathbf{a}$ is denoted a_k.

Vector Subscripts: Vectors annotated with subscripts (e.g., $\mathbf{x}_k$) indicate distinct versions of the vector. In the context of recursive estimation, such as the Kalman Filter (KF) algorithm, it indicates a discrete time sample of its argument.

Time Indices: When subscript notation such as $\mathbf{x}_k$ is impractical, a discrete time index such as $\mathbf{x}(t_k)$ may be also be used to indicate a discrete time sample of its preceding vector.

Estimated Variables: Variables that represent an estimate of a particular quantity are represented with the circumflex accent, e.g., $\hat{\mathbf{a}}$.

A Priori and A Posteriori Estimates: When describing the operation of a KF algorithm, it is necessary to distinguish between estimates computed before (*a priori*) or after (*a posteriori*) a measurement update. In such instances, a "minus" character superscript is added to the variable for *a priori* estimates while a "plus" character superscript is added to *a posteriori* estimates, e.g., $\hat{\mathbf{a}}_k^-$ or $\hat{\mathbf{a}}_k^+$ and $\hat{\mathbf{a}}(t_k^-)$ or $\hat{\mathbf{a}}(t_k^+)$.

Matrices: Matrices are represented by uppercase characters in **bold**, e.g., $\mathbf{A}$ or $\boldsymbol{\Psi}$.

Transpose: The superscript $(\cdot)^{\mathrm{T}}$ denotes the transpose operator.

Identity and Zero: The vectors $\mathbf{0}$, $\mathbf{1}$ contain all zeros and all ones, respectively. Additionally, the matrix $\mathbf{I}$ is the identity matrix. When it is not clear from the context, they will be subscripted with their dimensionality.

Reference Frames: If a vector is expressed in a specific reference frame, a superscript letter is used to designate the reference frame, e.g., $\mathbf{p}^a$ is a vector in the a-frame.

Multi-Sensor Parameters In the context of multi-sensor and multi-filter navigation, parameters obtained from a particular sensor or filter source are denoted with a bracketed superscript. For example, the measurement vector $\mathbf{z}^{[i]}$ is obtained from Sensor i, while the estimated measurement vector $\hat{\mathbf{z}}^{[j]}$ is obtained from Filter j.

Direction Cosine Matrices: Direction Cosine Matrices (DCMs) representing a rotation from frame a to frame b are denoted by $\mathbf{C}_a^b$.

Natural Logarithm: The operator $\log(\cdot)$ indicates the natural logarithm unless otherwise noted.

Statistical Notation: A statistical expectation is denoted $E[\cdot]$. The statistical distribution of a random variable is denoted via $\sim$. For example, if a random vector $\mathbf{a}$ at time k follows the multivariate normal distribution with mean vector $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$, it is denoted $\mathbf{a}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

2.2 Estimation and Detection Fundamentals

Model estimation refers to determining the statistical relationship between a set of predictor variables and a response variable through a number of estimated coefficients or model parameters. In classical estimation, the values of a reasonable set of predictor variables are recorded along with the corresponding response variable values in a properly designed experiment. Then, the values of the coefficients relating predictor variables to the response variable are determined using any number of estimation techniques. The type of estimation technique varies, depending on the relationship between predictor variables, unknown coefficients, and response variables. In general, an attempt is always made to utilize an optimal estimator, such as a Minimum Variance Unbiased Estimator (MVUE), which guarantees unbiased coefficient estimates with the lowest variance possible. However, depending on the conditions of the experiment or the relationship among the predictor variables, one may choose to accept biased coefficient estimates in order to further minimize variance as done in the ridge regression technique [68] discussed later in this chapter. Classical model estimation forms the basis for the more applicable online model estimation techniques summarized in later sections. Under classical model estimation, the coefficient estimation technique varies greatly depending on the relationship between the predictor variables and the unknown coefficients, leading to linear and non-linear regression techniques.

2.2.1 Linear Regression.

In linear regression, the relationship between observations of a response variable and the corresponding values of their predictor variables is given by

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{2.1}$$

where $\mathbf{y}$ is a $N \times 1$ vector containing the response variable observations, $\mathbf{X}$ is a $N \times M$ matrix containing N observations of each of the M predictor variables, $\boldsymbol{\beta}$ is a $M \times 1$ vector containing the unknown model coefficients to be estimated, and $\boldsymbol{\epsilon}$ is a $N \times 1$ vector of normal Independent Identically Distributed (IID) error terms with

$$\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right). \tag{2.2}$$

The optimal (MVUE) estimator for this type of problem is referred to as Ordinary Least Squares (OLS) as proven in the Gauss-Markov Theorem [68]. Based on OLS, the optimal estimate of $\boldsymbol{\beta}$ is given by the least-squares projection of the vector $\mathbf{X}\boldsymbol{\beta}$ onto the subspace containing $\mathbf{y}$ and computed using the normal equations

$$\mathbf{X}^{\mathrm{T}}\mathbf{y} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\beta} \tag{2.3}$$

$$\implies \hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}. \tag{2.4}$$

The estimated model coefficients then yield the estimated model response using

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} \tag{2.5}$$

$$= \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2.6}$$

$$= \mathbf{H}\mathbf{y}, \tag{2.7}$$

where the matrix $\mathbf{H}$, referred to as the "hat" matrix, obtained by substituting (2.4) into (2.5), plays a significant role in the model diagnostic techniques. The resulting residual terms are given by

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} \tag{2.8}$$

$$= \mathbf{y} - \mathbf{H}\mathbf{y} \tag{2.9}$$

$$= (\mathbf{I} - \mathbf{H})\,\mathbf{y}. \tag{2.10}$$

Finally, the estimated covariance matrix of the residual terms is given by

$$\mathbf{P} = \frac{\mathbf{e}^{\mathrm{T}}\mathbf{e}}{N - M}(\mathbf{I} - \mathbf{H}) \tag{2.11}$$

$$= s^2(\mathbf{I} - \mathbf{H}), \tag{2.12}$$

where the quantity $\mathbf{e}^{\mathrm{T}}\mathbf{e}$ is referred to as the Sum of Squared Errors (SSE), s^2 is referred to as Mean Squared Error (MSE) and

$$E\left[s^2\right] = \sigma^2. \tag{2.13}$$

2.2.2 Non-linear Regression.

Often, the relationship between predictor variables and model coefficients cannot be described by the linear operation $\mathbf{X}\boldsymbol{\beta}$. In this case, we turn to nonlinear regression techniques [8]. The nonlinear regression model is given by

$$\mathbf{y} = \mathbf{f}[\mathbf{X}, \boldsymbol{\beta}] + \boldsymbol{\epsilon}, \tag{2.14}$$

where $\mathbf{y}$ is a $N \times 1$ vector containing the response variable observations, $\mathbf{f}$ is a $N \times 1$ nonlinear function of predictor variables and model coefficients, $\mathbf{X}$ is a $N \times M$ matrix containing N observations of each of the M predictor variables, $\boldsymbol{\beta}$ is a $P \times 1$ vector containing the unknown model coefficients to be estimated, and $\boldsymbol{\epsilon}$ is a $N \times 1$ vector of IID error terms with the same properties as described in (2.2). It is important to note that in the nonlinear case, the number of predictor variables (M) and the number of model coefficients (P) does not have to match as it did in the linear case. In this case, the optimal estimator is still given by least squares, but additional linear approximations must be applied to the problem in order to find a solution. Using the Gauss-Newton method [8] as one of many available methods, we begin with an initial guess of the model coefficients $\hat{\beta}_0$ and use the first-order Taylor Series Expansion (TSE) of $\mathbf{f}$ in order to iteratively improve the guess until convergence. The first-order TSE of $\mathbf{f}$ about $\boldsymbol{\beta}_0$ is given by

$$\mathbf{f}[\mathbf{X},\boldsymbol{\beta}] \approx \mathbf{f}[\mathbf{X},\hat{\boldsymbol{\beta}}_0] + \left.\frac{\partial \mathbf{f}[\mathbf{X},\boldsymbol{\beta}]}{\partial \boldsymbol{\beta}}\right|_{\hat{\boldsymbol{\beta}}_0} \left(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_0\right) \tag{2.15}$$

$$\implies \boldsymbol{\eta}(\boldsymbol{\beta}) = \boldsymbol{\eta}(\hat{\boldsymbol{\beta}}_0) + \mathbf{V}_0\boldsymbol{\delta}_0, \tag{2.16}$$

where $\boldsymbol{\eta}(\boldsymbol{\beta})$ is the model response given $\mathbf{X}$ and $\boldsymbol{\beta}$, the matrix $\mathbf{V}_0$ represents the Jacobian matrix of $\mathbf{f}$ with respect to the variables $\boldsymbol{\beta}$ evaluated at $\hat{\boldsymbol{\beta}}_0$, and the vector $\boldsymbol{\delta}_0$ is the difference between the unknown true parameter values $\boldsymbol{\beta}$ and the current guess $\hat{\boldsymbol{\beta}}_0$. Next the model residuals given the current guess are given by

$$\mathbf{e}(\boldsymbol{\theta}) = \mathbf{y} - \boldsymbol{\eta}(\boldsymbol{\theta}) \approx \mathbf{y} - \left[\boldsymbol{\eta}(\hat{\boldsymbol{\beta}}_0) + \mathbf{V}_0\boldsymbol{\delta}_0\right] \tag{2.17}$$

$$= \left[\mathbf{y} - \boldsymbol{\eta}(\hat{\boldsymbol{\beta}}_0)\right] - \mathbf{V}_0\boldsymbol{\delta}_0 \tag{2.18}$$

$$= \mathbf{e}_0 - \mathbf{V}_0\boldsymbol{\delta}_0 \tag{2.19}$$

$$\implies \mathbf{e}_0 = \mathbf{V}_0\boldsymbol{\delta}_0. \tag{2.20}$$

Minimizing the approximate residual sum of squares $||\mathbf{e}_0 - \mathbf{V}_0\boldsymbol{\delta}_0||^2$ is equivalent to solving the least squares system given by (2.20), which yields the Gauss Increment for the initial guess, $\boldsymbol{\delta}_0$, using

$$\mathbf{V}_0^{\mathrm{T}}\mathbf{e}_0 = \mathbf{V}_0^{\mathrm{T}}\mathbf{V}_0\boldsymbol{\delta}_0 \tag{2.21}$$

$$\implies (\mathbf{Q}_1\mathbf{R}_1)^{\mathrm{T}}\,\mathbf{e}_0 = (\mathbf{Q}_1\mathbf{R}_1)^{\mathrm{T}}\,\mathbf{Q}_1\mathbf{R}_1\boldsymbol{\delta}_0 \tag{2.22}$$

$$\implies \mathbf{R}_1^{\mathrm{T}}\mathbf{Q}_1^{\mathrm{T}}\mathbf{e}_0 = \mathbf{R}_1^{\mathrm{T}}\mathbf{R}_1\boldsymbol{\delta}_0 \tag{2.23}$$

$$\implies \mathbf{Q}_1^{\mathrm{T}}\mathbf{e}_0 = \mathbf{R}_1\boldsymbol{\delta}_0 \tag{2.24}$$

$$\implies \delta_0 = \left(\mathbf{R}_1^{\mathrm{T}}\mathbf{R}_1\right)^{-1}\mathbf{R}_1^{\mathrm{T}}\mathbf{Q}_1^{\mathrm{T}}\mathbf{e}_0 \tag{2.25}$$

$$= \left(\mathbf{R}_1^{\mathrm{T}}\mathbf{R}_1\right)^{-1}\mathbf{V}_0^{\mathrm{T}}\mathbf{e}_0, \tag{2.26}$$

where $\mathbf{V}_0 = [\mathbf{Q}_1\mathbf{Q}_2][\mathbf{R}_1\mathbf{R}_2]^{\mathrm{T}}$ is the QR decomposition of $\mathbf{V}_0$ utilized to take advantage of the sparseness of $\mathbf{V}_0$ when inverting it. Next, the initial guess, $\hat{\boldsymbol{\beta}}_0$, is updated using

$$\hat{\boldsymbol{\beta}}_1 = \hat{\boldsymbol{\beta}}_0 + \boldsymbol{\delta}_0, \tag{2.27}$$

which allows for the computation of a new Jacobian matrix, $\mathbf{V}_1$, new estimated residuals, $\mathbf{e}_1$, and a new Gauss Increment, $\boldsymbol{\delta}_1$. The process is then repeated until

$$|\boldsymbol{\delta}_{k+1} - \boldsymbol{\delta}_k| < \gamma, \tag{2.28}$$

where γ is a predetermined convergence threshold appropriate for the problem. Given the invertibility of $\mathbf{R}$ is highly dependent on its condition, and the fact that $\mathbf{R}$ may be ill-conditioned if the initial guess, $\hat{\boldsymbol{\beta}}_0$, is significantly far from its optimal value, the "Levenberg-Marquardt" compromise [8][75] calls for the addition of a small bias prior to inverting $\mathbf{R}_1$ using

$$\boldsymbol{\delta}_k = \left(\mathbf{R}_1^{\mathrm{T}}\mathbf{R}_1 + \lambda\mathbf{I}\right)^{-1} \mathbf{V}_k^{\mathrm{T}}\mathbf{e}_k, \tag{2.29}$$

where λ is a small conditioning factor appropriate for the problem that stabilizes the algorithm in a similar fashion to the ridge regression technique [44], which will be discussed in the next section.

Once convergence has been achieved, the final model coefficients, $\boldsymbol{\beta}_f$ can be used to produce the final residual vector $\mathbf{e}_f$, leading to similar computations of MSE and covariance matrix, $\mathbf{P}$ as shown in (2.11) by letting $M = P$. For further information on the Gauss-Newton and other nonlinear regression and optimization techniques (e.g., Newton-Rapson), the reader is directed to the works of Arora [5] and Bates [8].

2.2.3 Ridge Regression.

Ridge regression refers to a class of biased estimators, proposed to solve the problem of multicollinearity, or dependence among predictor variables (i.e., columns in $\mathbf{X}$). Multicollinearity does not affect the reliability of the fitted model as a whole, but does affect the inferences that may be drawn upon individual model coefficients or predictor variables. Since this problem was encountered in the research contained in Chapter 7, its mitigating procedure (ridge regression) is briefly summarized. Although not directly categorized as a residual-based model diagnostic, multicollinearity can be detected via the Variance Inflation Factor diagnostic test [68]. The ridge regression technique [44] introduces a small biasing constant, λ, to the normal equations such that

$$\mathbf{X}^{\mathrm{T}}\mathbf{y} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)\boldsymbol{\beta} \tag{2.30}$$

$$\implies \hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{2.31}$$

which stabilizes the ill-conditioned $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ matrix, making it invertible in a similar fashion as the Levenberg-Marquardt conditioning bias from Section 2.2.2. The result is slightly biased, yet much less variable estimates of the model coefficients $\hat{\boldsymbol{\beta}}$, which are more likely to be close to their true values than their unbiased counterparts. Further information such as the optimal choice of λ can be found in [68].

2.2.4 Weighted Least Squares.

The effects of heteroscedasticity, or unequal error term variances can be mitigated via a weighted least squares technique [68]. Recalling the true error terms are assumed to be distributed as shown in (2.2), we can modify the diagonal covariance matrix $\sigma^2\mathbf{I}$ to account for changing variances. Given a set of error term variances $\sigma_k^2, \ldots, \sigma_N^2$, let the set of weights, w_k be defined as

$$w_k = \frac{1}{\sigma_k^2}, \quad k = 1, \ldots, N, \tag{2.32}$$

then the corresponding weight matrix, $\mathbf{W}$, is given by

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & w_N \end{bmatrix}. \tag{2.33}$$

Next, the normal equations from (2.3) can be modified to produce estimated model coefficients using

$$\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{y} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)\boldsymbol{\beta} \tag{2.34}$$

$$\implies \hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{y}, \tag{2.35}$$

which can also be derived using Maximum Likelihood Estimation (MLE) as shown in the OLS case. The obvious complication with this technique is that true error term variances are seldom known. To alleviate this complication, [68] provides several techniques for estimating the variances needed to build **W**. Additionally, **W** can be used to discount, or de-weight, suspected observations such as outliers or observations where data quality is known to be lower compared to the rest of the observed data. The latter use is exploited in Chapter 6 to de-weight known low-quality Pitot-static sensor data during turns, as later described.

2.2.5 Smoothing Splines.

Often, model transformations are useful in modifying the nature of the regression function in order to satisfy the conditions or assumptions for optimality. A common problem found in classical regression is a lack-of-fit determination by the F-Test for Lack of Fit (FTLF), which can be symptomatic of a mismatch between the true underlying functional form between **y** and **X** (in both the linear and nonlinear cases) and the form specified in the fitted model. In some cases, the functional form may not exist in closed form or apply to the entire domain of the predictor variables. In these cases, it may be useful to use the concept of a smoothing spline [89] as will be shown in the developments of Chapter 6. In the linear context, a second-order smoothing spline follows the form

$$\mathbf{y} = \begin{bmatrix} 1 & x_1 & x_1^2 & (x_1 - s_1)_+^2 & \dots & (x_1 - s_P)_+^2 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_N & x_N^2 & (x_N - s_1)_+^2 & \dots & (x_N - s_P)_+^2 \end{bmatrix} \boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{2.36}$$

where each $s_p, \quad p = 1, \ldots, P$, referred to as a knot, is a preselected inflection point along the domain of x, and the operator $()_+$ denotes negative values of its argument are set to zero, which is equivalent to multiplying by the Heaviside function centered at the knot location. Several works of literature are dedicated at selecting the optimal s_p locations [31][89][90][102]. When splines are used in this research, the knot locations will be selected using data quantiles as shown in [89]. It is important to note if a nonlinear regression technique is used, the knot locations themselves can be set as unknown parameters to be estimated as long as the observed data contains enough independent information for stable estimation. One of the main benefits of using smoothing splines is the ability to provide excellent fit using windowed polynomials even if the observed data is clearly not polynomial in nature. Even further, smoothing splines can be successfully used when the true functional form between $\mathbf{y}$ and $\mathbf{X}$ is unknown. Figure 2.1 illustrates a fitted smoothing spline with eight knots adequately modeling a clearly non-polynomial function using second order polynomials.

2.2.6 Model Selection.

Model selection refers to the class of techniques used to analyze the overall effectiveness of a chosen model. One common criterion is simply the amount of unexplained variation left in the observed data after fitting the model (i.e., SSE or MSE). However, more robust criteria tend to balance error reduction with model complexity (i.e., number of model coefficients in $\boldsymbol{\beta}$). One of the more popular criterion in model selection, Akaike Information Criterion (AIC) [2], is rooted in information theory and is given by

$$\mathrm{AIC}(P) = N \log(s_P^2) - N \log(N) + 2P, \tag{2.37}$$

where P is the number of model coefficients (i.e., the dimension of $\boldsymbol{\beta}$), N is the number of observations (i.e., the dimension of $\mathbf{y}$) and s_P^2 is the SSE of the model with P parameters

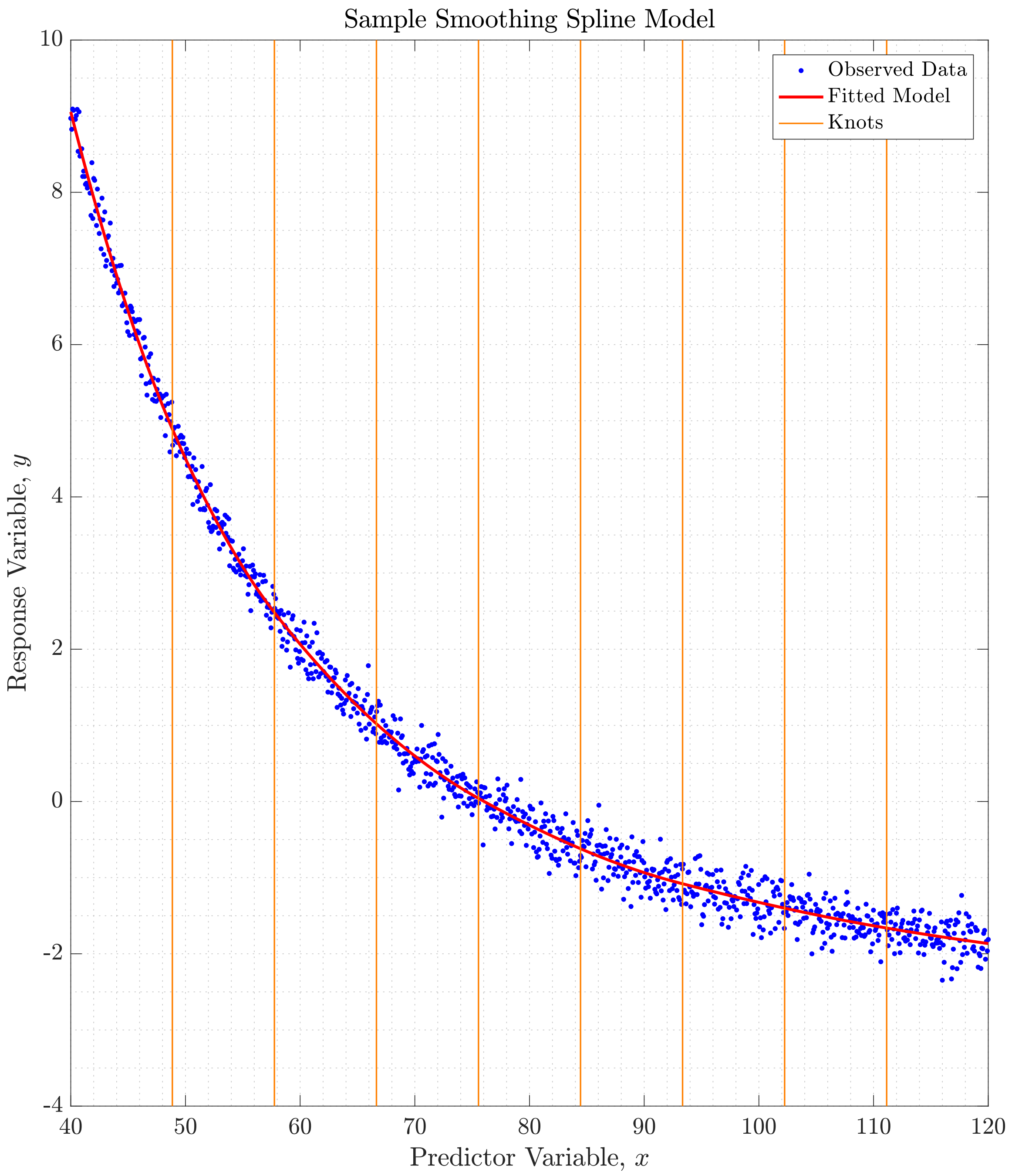


Figure 2.1: Example of a fitted smoothing spline model with eight knots selected using quantiles [89]. The second-order polynomial smoothing spline is able to model an exponential functional form with no prior knowledge.

included. Additionally, for small sample sizes, an adjusted AIC is given by

$$\mathrm{AIC_c}(P) = \mathrm{AIC}(P) + \frac{2P(P+1)}{N-P-1}. \tag{2.38}$$

Although the adjustment given in (2.38) is only necessary for N/P ratios less than 40 [2], the value of the adjustment tends to decrease dramatically as N increases, allowing for the use of AIC$_c$ ubiquitously, regardless of sample size. As later shown in Chapter 3, AIC is used as the model selection criteria in the remodeling mode of the overall framework. Additionally, in Chapter 6, AIC is combined with the concept of smoothing splines in order to autonomously select an appropriate number of knots while balancing error reduction and model complexity.

2.2.7 Maximum Likelihood Estimation.

MLE [62] refers to the concept of estimating model parameters, $\boldsymbol{\beta}$, through the optimization, or maximization, of the associated model likelihood function. The likelihood function is simply the joint Probability Distribution Function (PDF) of the observed data, $\mathbf{y}$, viewed as a function of $\boldsymbol{\beta}$, given $\mathbf{y}$. Given a model of the form shown in (2.1) with error terms given by (2.2), the likelihood function of the model coefficients, $\boldsymbol{\beta}$, given the observed data, $\mathbf{y}$ is given by

$$L(\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) = \frac{1}{\sqrt{|2\pi\sigma^2\mathbf{I}|}} e^{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})}, \tag{2.39}$$

where the linear model $\mathbf{X}\boldsymbol{\beta}$ can be replaced by a nonlinear function $\mathbf{f}[\mathbf{X},\boldsymbol{\beta}]$ in the nonlinear case. Given the exponential form of (2.39), it is often convenient to take its natural logarithm prior to maximizing the function. This leads to the log-likelihood function

$$\log(L) = \mathcal{L} = -\frac{1}{2}\log\left(|2\pi\sigma^2\mathbf{I}|\right) - \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}), \tag{2.40}$$

and the optimization problem given by

$$\underset{\boldsymbol{\beta}}{\arg\max}\,\mathcal{L} = \underset{\boldsymbol{\beta}}{\arg\max}\ (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}), \tag{2.41}$$

which is solved by setting the first-order derivative of $\mathcal{L}$ to zero, yielding

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \mathbf{0} = \frac{\partial}{\partial \boldsymbol{\beta}} \left(\mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\boldsymbol{\beta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \boldsymbol{\beta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\beta} \right) \tag{2.42}$$

$$= \left(\mathbf{0} - 2\mathbf{X}^{\mathrm{T}}\mathbf{y} + 2\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\beta} \right) \tag{2.43}$$

$$\implies \mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2.44}$$

$$\implies \hat{\boldsymbol{\beta}}_{\mathrm{MLE}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} \right)^{-1} \mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{2.45}$$

which is identical to the optimal estimator in the linear model given by (2.4). This same procedure can be applied to the nonlinear function $\mathbf{f}[\mathbf{X}, \boldsymbol{\beta}]$ given it is once-differentiable with respect to $\boldsymbol{\beta}$. MLE estimation is part of a broader set of estimation applications, under which OLS fits, as shown by the above result. Although MLE has additional benefits, such as the ability to handle arbitrary error term distributions or specify known (possibly unequal) error term variances, the majority of the concepts leveraged in the proposed framework are specific to OLS theory and normally distributed error terms. Further information on MLE estimation of linear and nonlinear models is found in [8][62][68].

2.2.8 Binary Detection.

Although not directly used in parameter estimation, detection theory plays a significant role in statistical hypothesis testing as well as many model diagnostic techniques. Classical detection theory is described in [63] and can be generally thought of as classifying a received signal (e.g., a sensor measurement) into one of many plausible originating sources or distributions. In the binary case, a received measurement can be thought of as originating from one of two possible distributions. The detection problem then reduces to choosing (possibly multivariate) thresholds for classification based on desired Probability of Detection (PD) and Probability of False Alarm (PF) rates. In general, a binary detection problem can be defined using

$$H_0 : p(\mathbf{x}|H_0) = p_0(\mathbf{x}) \tag{2.46}$$

$$H_1 : p(\mathbf{x}|H_1) = p_1(\mathbf{x}) \tag{2.47}$$

$$\implies \frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} \underset{H_0}{\overset{H_1}{\gtrless}} \lambda, \tag{2.48}$$

where λ is a likelihood ratio threshold that is proportional to the desired PD and PF. The Likelihood Ratio Test (LRT) described in (2.48) can then be solved for $\mathbf{x}$ such that

$$f(\mathbf{x}) \underset{H_0}{\overset{H_1}{\gtrless}} \gamma, \tag{2.49}$$

where $f(\mathbf{x})$ is a function of the vector $\mathbf{x}$, and γ is a scalar-valued threshold on the function output. In the case where x is one-dimensional, then $f(x)$ reduces to x, and the LRT becomes a simple threshold on the value of x. The corresponding PD and PF rates are then given by

$$p_D = \int_{\Gamma_1} p_1(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{2.50}$$

$$p_F = \int_{\Gamma_1} p_0(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{2.51}$$

where Γ_1 is the set of $\mathbf{x}$ values such that $f(\mathbf{x}) > \gamma$. The values of PD and PF vary from zero to one and constitute unique points along the so-called Receiver Operating Characteristic (ROC) curve. Figure 2.2 illustrates a simple one-dimensional binary detection scenario using univariate Gaussian distributions for H_0 and H_1 along with the corresponding ROC curve. In statistics, p_F can also be described as a hypothesis test's significance level, α. Based on Wilks' Theorem [84], the (possibly multivariate) integral in (2.51) can be related to a corresponding Chi-Square (denoted χ^2) distribution given a desired α, with number of degrees of freedom based on the differences in p_1 and p_0. This relationship gives rise to a multitude of χ^2-based tests such as the ones used in Chapters 4 and 5.

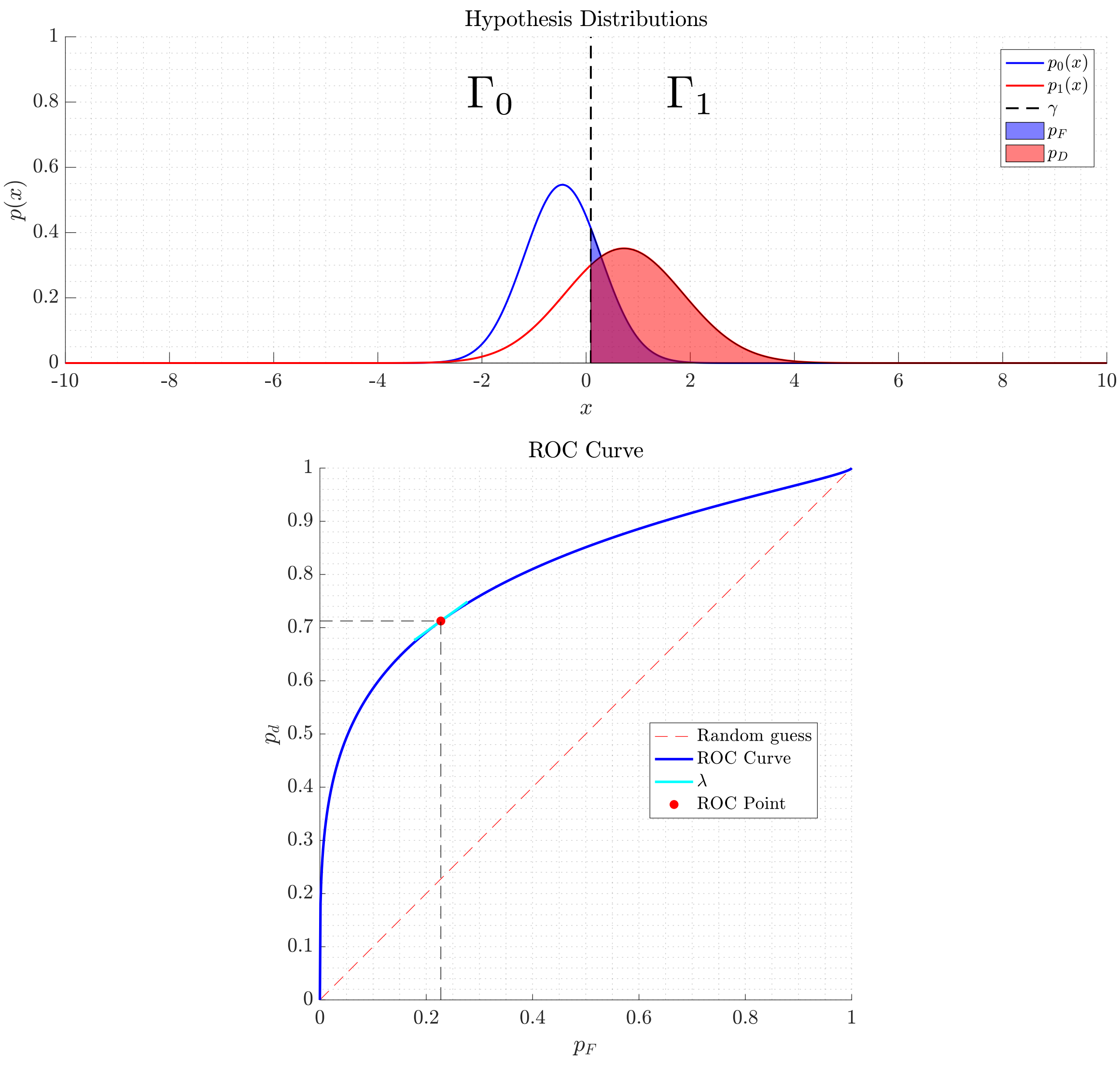


Figure 2.2: Example LRT for binary detection using univariate Gaussian distributions. The Γ_1 region corresponds to $x > \gamma$ and drives PD and PF based on which distribution is integrated. The ROC curve on the bottom graph illustrates the possible combinations of PD and PF for the given problem as well as the operating point based on the chosen γ. The corresponding λ value is equal to the ROC slope at the operating point.

2.3 Residuals and Their Properties

As previously alluded to, and shown in (2.8), model residuals, $\mathbf{e}$, are defined by the difference between the fitted model response, $\hat{\mathbf{y}}$, and the actual observed response, $\mathbf{y}$. The residual vector is thought of as the "observed" error terms, as opposed to the true error terms, which are defined by

$$\boldsymbol{\epsilon} = \mathbf{y} - \hat{\mathbf{y}} \tag{2.52}$$

$$= \mathbf{y} - E\left[\mathbf{y}\right], \tag{2.53}$$

which are assumed to be IID normal random variables with zero mean and constant variance as shown in (2.2). The overarching term given to the techniques to be discussed in this section is "residual analysis," in which the properties of the observed error terms are compared against the expected properties of the true error terms to assess the adequacy of the fitted model and its assumptions. Three residual properties are commonly studied in residual analysis: mean, variance, and independence.

2.3.1 Mean.

The mean of the N residuals for a fitted model is given by

$$\bar{e} = \frac{\sum_{k=1}^{N} e_k}{N}, \tag{2.54}$$

and as enforced by OLS, the sum of residuals always equals zero, which makes the mean of residuals always equal to zero [68]. As such, the mean of residuals cannot be used to assess the mean of the true error terms, since residuals are constructed, by definition, to be zero mean based on the model estimation technique. However, as later discussed, this property can indeed be exploited when analyzing KF residuals [76], since the difference between actual and expected sensor measurements does not necessarily equal zero and can pinpoint the existence of un-modeled sensor biases.

2.3.2 Variance.

The variance of the N residuals for a fitted model is given by (2.11). Additionally, per (2.13), s^2 is an unbiased estimator for the variance of the true error terms σ^2. As previously noted, the variance of the true error terms, $\boldsymbol{\epsilon}$, is assumed to be constant throughout the N observations. Therefore, the variance of the observed error terms, $\mathbf{e}$, can be analyzed across the observations to validate this assumption.

2.3.3 Nonindependence.

In a similar fashion to their mean, residual terms are by definition dependent random variables since they all involve the fitted model $\hat{\mathbf{y}}$. However, serial correlation among the residual terms does point at a number of potential problems with the fitted model including missing predictor variables, and temporal, cyclical, or spatially correlated effects that were not accounted for in the original model.

2.4 The Navigation Problem

Having defined the necessary mechanisms for classical, or offline, model estimation, we now turn our attention to the general motivating problem for this research: navigation. Navigation in the context of this research refers to the real-time estimation of navigation parameters, which describe a vehicle's position, velocity, and attitude (or subset of these) within a defined reference frame.

2.4.1 Relation to Model Estimation.

Navigation is rooted in the estimation of key vehicle properties, usually called states. These states often include the estimated current three-dimensional position, velocity, and attitude of the vehicle as well as their time histories and covariances. As such, the concept of navigation is deeply related to model estimation. The navigation states can be regarded as unknown model coefficients, $\boldsymbol{\beta}$ (later referred to as system states $\mathbf{x}$). Such states are to be estimated recursively and in real-time by combining (via weighted least squares) known vehicle dynamics and external sensor measurements, $\mathbf{y}$ (later referred to as $\mathbf{z}$), which can be mapped to the navigation states via a measurement model $\mathbf{X}$ (later referred to as $\mathbf{H}$). Although the concepts are similar, the requirement for online or real-time estimation makes classical model estimation techniques discussed in previous sections not completely suitable. Fortunately, an adaptation of such, found in the KF algorithm [59] and discussed in a later section, lends itself not only for efficient estimation of kinematic errors, but for the online integration of external measurements. As it will be later shown, although the concept of online model estimation has been well developed thus far in existing literature, the accompanying diagnostic and remedial measures are not as well-defined as those found in classical estimation. Herein lies a key contribution of the proposed framework.

2.4.2 Reference Frames.

Navigation reference frames are fundamentally important when expressing the position, velocity, and orientation of a vehicle. In general, the following major reference frames defined in [95] are used throughout this research.

The true inertial frame (*I*-frame) - a theoretical reference frame in which Newton's laws of motion apply. The frame is defined by a non-accelerating, non-rotating orthonormal basis in $\mathbb{R}^3$. Because of the relative nature of the universe, the true inertial frame has no predefined origin or orientation.

The Earth-centered inertial frame (*i*-frame) - an orthonormal basis in $\mathbb{R}^3$, with its origin at the center of mass of the Earth. The *x* and *y* axes are located on the equatorial plane with the *x*-axis pointing towards Aries. The *z*-axis points towards the North Pole. The *i*-frame is a non-rotating frame, but it does accelerate with respect to the true inertial frame due to the relative rotation between celestial bodies. However, for terrestrial navigation purposes, it can be considered an inertial reference frame. The *i*-frame is illustrated in Figure 2.3.

The Earth-centered Earth-fixed frame (*e*-frame) - an orthonormal basis in $\mathbb{R}^3$, with its origin also at the Earth's center of mass. The *e*-frame is rigidly attached to the Earth, with the *x*-axis on the equatorial plane pointing toward the Greenwich meridian, the *z*-axis aligned with the North Pole, and the *y*-axis on the equatorial plane pointing toward 90 degrees East longitude. Because the *e*-frame is a true Cartesian reference frame, some navigation computations are simplified. The *e*-frame is illustrated in Figure 2.3.

The Earth-fixed navigation frame (*n*-frame) - an orthonormal basis in $\mathbb{R}^3$, with its origin located at a predefined location on the Earth, typically on the surface. The Earth-fixed navigation frame's *x*, *y*, and *z* axes point in the North, East, and down directions relative to the origin, respectively. In this frame, down is defined geometrically as a vector normal to the Earth ellipsoid, which on average, is aligned with the local gravity vector. The Earth-fixed navigation frame remains fixed to the surface of the Earth. While this frame is not useful for very-long distance navigation, it can simplify the navigation kinematic equations for shorter navigation routes. The *n*-frame is illustrated in in Figure 2.3. Though the Earth-fixed *n*-frame was used in Chapter 6, it is important to note long-term navigation problems usually employ a local-level navigation frame, where the origin is moved with the vehicle, as described in [95].

The aircraft body frame (*b*-frame) - an orthonormal basis in $\mathbb{R}^3$, rigidly attached to the aircraft with its origin located on the aircraft's center of mass. The *x*, *y*, and *z* axes point out the nose, right wing, and bottom of the aircraft, respectively. Strap-down inertial sensors are fixed to the *b*-frame, although they may not be located at the origin or aligned with the axes. The *b*-frame is illustrated in Figure 2.4.

The wind frame (*w*-frame) - an orthonormal basis in $\mathbb{R}^3$, rigidly attached to the aircraft with its origin located on the aircraft's center of mass. The x axis is aligned with the direction of the aircraft's velocity vector relative to the airmass, the z axis is perpendicular to the x axis, in the plane of symmetry of the aircraft, and is positive below the aircraft. Finally, the y axis completes the right-handed coordinate system. The w-frame is necessary when using Pitot-static airspeed measurements and is usually rotated about the b-frame via the Angle of Attack (AoA) and Angle of Sideslip (AoS). The w-frame is illustrated in Figure 2.4.

2.4.3 Inertial Navigation.

Inertial navigation is based on the concept that, starting from a known location, attitude, and velocity, a vehicle's current position and attitude can be estimated by integrating measured changes in velocity and rotation. Inertial navigation measurement devices such as an Inertial Measurement Unit (IMU) consist of accelerometers, which measure specific forces, and gyroscopes (commonly referred to as gyros), which measure rotational rates relative to the I-frame. When equipped with a navigation computer capable of integrating measured changes into a position, velocity, and attitude solution, the entire system is referred to as an Inertial Navigation System (INS). In general, there are two types of INSs: platform and strap-down. A platform INS mounts three orthogonal accelerometers onto a gimbaled platform that maintains the vertical accelerometer aligned with local gravity via gyroscopic rigidity. In turn, the gyroscopes used to maintain spatial rigidity are used to read off the vehicle's attitude. In contrast, a strap-down INS consists of a simple three-axis IMU rigidly mounted onto the vehicle with its motion sensors mounted orthogonally and aligned with the vehicle's b-frame. Consequently, the gyroscopes are used to estimate the vehicle's orientation relative to the I-frame, along which the specific forces measured by the accelerometers are integrated by the navigation computer. Although

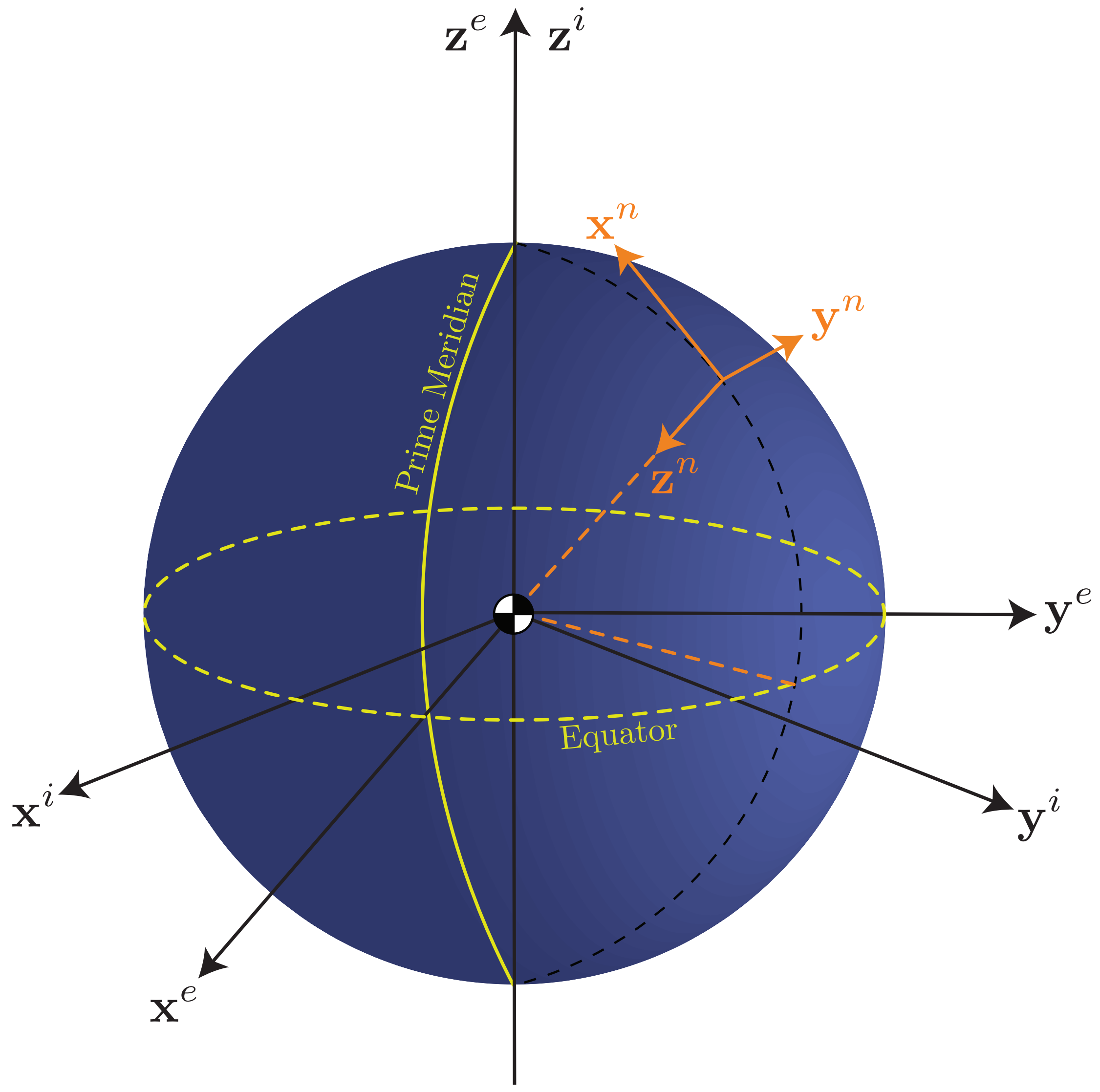


Figure 2.3: Illustration of Earth centered *i*-frame, *e*-frame, and *n*-frame. The *e*- and *i*-frames are rigidly attached to the Earth's center of mass, while the *n*-frame is attached to the surface of the Earth. The *i*-frame is non-rotating with respect to inertial space, while the *e*-frame rotates along with the Earth's rotation. The *n*-frame is aligned with north, east, and local gravity vectors.

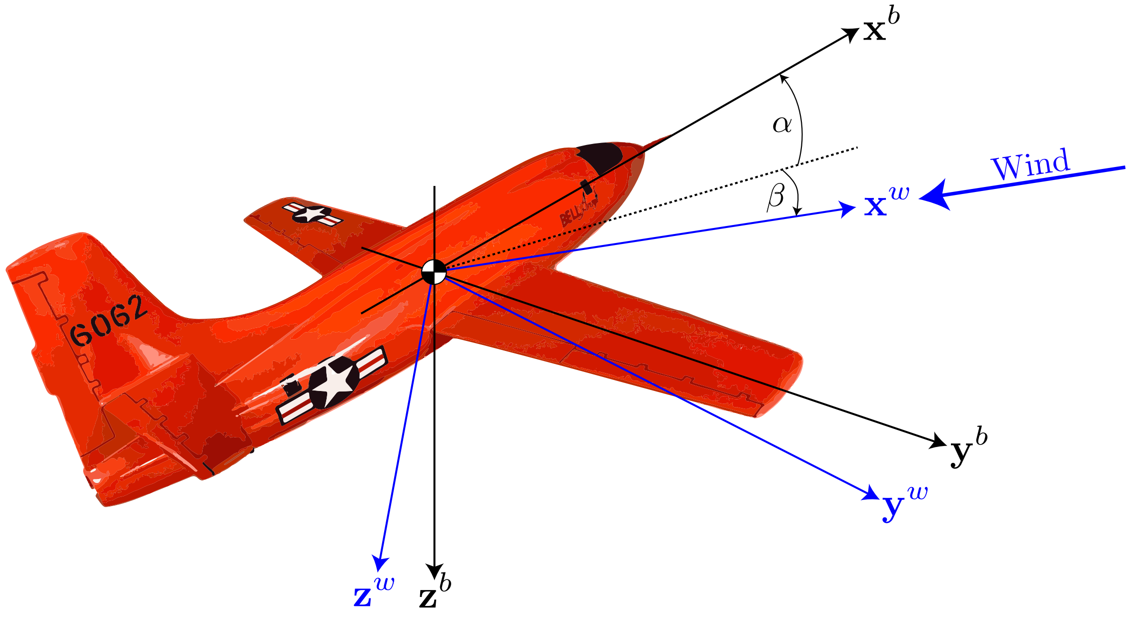


Figure 2.4: Illustration of aircraft *b*-frame and *w*-frame. Both frames are rigidly attached to the aircraft's center of mass. The *b*-frame and *w*-frame are rotated by Angle of Attack, α, and Angle of Sideslip, β, due to the relative orientation between the aircraft body and the wind mass.

more computationally complex, strap-down INSs reduce mechanical complexity, size, and power requirements needed for navigation, especially when aided by an external measurement sensor. This development subsequently enabled the integration of Micro Electro-Mechanical INSs onto small-scale devices such as quad-rotors and mobile phones. The specific mathematical processes governing navigation computers, including the so-called INS mechanization equations are detailed in [95]. In Chapter 7, a novel method for autonomously calibrating accelerometer and gyroscope error models is detailed as a specific online calibration example within the proposed framework.

2.4.4 All-Source Navigation.

The concept of inertial navigation summarized in the previous section can now be expanded into the main motivational thrust for this research, known as all-source navigation. As foreshadowed in Chapter 1, inertial navigation alone is subject to "drift," or a growing error in navigation state estimation due to the buildup of small incremental errors during each recursion of the mechanization equations. Since inertial navigation is a dead-reckoning technique, errors accumulate during each iteration. The problem of drift has been traditionally solved by incorporating a trusted, external measurement update in the KF algorithm. Furthermore, the most common external measurement source has traditionally been GPS position updates. Since GPS provides an absolute position measurement (as opposed to a relative position), using it as a measurement update source tends to reset the errors built-up from the INS-only solution between GPS updates. A considerable amount of literature has been dedicated to researching optimal ways of integrating GPS and INS using differing KF-based feedback and correction loops [18][76][77][95]. However, recent research efforts have shifted focus onto alternative sources of INS aiding in order to combat increasing reliance on GPS. Each major effort in this context has focused on a particular external sensor, such as visual features [97][98], radar signals [60], magnetic fields [22], and Very Low Frequency emissions [27], to name a few. The concept of navigating

”anywhere, anytime, using anything” is the essence of all-source navigation. In each of the aforementioned applications, researchers tend to develop unique optimization techniques in order to use the intended sensor adequately. However, no general approach has been developed to provide a resilient and assured solution in all-source applications.

2.5 Recursive Model Estimation

2.5.1 Kalman Filter.

The Kalman Filter (KF), developed by Rudolf Kalman in 1960 [59] provides a method for the optimal combination of measurements made by multiple sensors. The KF uses Bayesian statistics to combine dynamics and measurements models, which provides a solution estimate with the lowest possible variance or uncertainty. This section outlines the basic principles behind the KF as outlined by Maybeck [76][77]. The physical system dynamics are modeled using the form

$$\dot{\mathbf{x}}(t) = \mathbf{F}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t), \tag{2.55}$$

where $\mathbf{x}$ is a vector containing the system states of interest, $\mathbf{u}$ is a vector containing system control inputs and $\mathbf{w}$ is a vector of white Gaussian noise sources with

$$E\left[\mathbf{w}(t)\right] = \mathbf{0}, \tag{2.56}$$

$$E\left[\mathbf{w}(t)\mathbf{w}^{\mathrm{T}}(t+\tau)\right] = \mathbf{Q}\delta(\tau), \tag{2.57}$$

while the matrices $\mathbf{F}$, $\mathbf{B}$ and $\mathbf{G}$ contain constant coefficients, which specify linear combinations of the vectors they multiply.

In order to implement the KF algorithm in a computer system, the continuous-time model must be discretized to account for system propagation between samples. The discrete process noise strength matrix $\mathbf{Q}_d$ and the discrete control input matrix $\mathbf{B}_d$ are obtained by changing the limits of integration to capture a single time step Δt within the general solution to the system, which is given by Maybeck [77] and VanLoan [96].

Additionally, the discrete state transition matrix, which is used to propagate system states and derived from the system dynamics model, is given by

$$\mathbf{\Phi} = e^{\mathbf{F}\Delta t}. \tag{2.58}$$

Linear discrete measurements from the various sensors are modeled by

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \tag{2.59}$$

where $\mathbf{z}$ is a vector of sensor measurements and $\mathbf{v}$ is a vector of discrete-time, white Gaussian noise sources with

$$E\left[\mathbf{v}_k\right] = \mathbf{0}, \tag{2.60}$$

$$E\left[\mathbf{v}_k\mathbf{v}_l^{\mathrm{T}}\right] = \mathbf{R}\delta_{kl}, \tag{2.61}$$

while the matrix $\mathbf{H}$ contains constant coefficients, which specify linear combinations of the system state vector $\mathbf{x}$. Since the system is linear, the KF algorithm guarantees a Minimum Mean Squared Error (MMSE) optimal solution for estimating the system states.

The quantities of interest estimated by the KF are contained within the random vector $\mathbf{x}$. The KF provides the probability density function for $\mathbf{x}$ at each discrete time step, conditioned on noise corrupted measurements provided by sensors. The KF algorithm begins with initial conditions, which include the initial state estimate vector $\hat{\mathbf{x}}_0$ and its uncertainty, which is contained by the covariance matrix $\mathbf{P}_0$. The initial conditions are propagated from one discrete time step to the next using the discrete state transition matrix such that

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{\Phi}\hat{\mathbf{x}}_k^+ + \mathbf{B}_d\mathbf{u}_k, \tag{2.62}$$

$$\mathbf{P}_{k+1}^- = \mathbf{\Phi}\mathbf{P}_k^+\mathbf{\Phi}^{\mathrm{T}} + \mathbf{Q}_d. \tag{2.63}$$

As linear measurements become available at discrete time intervals, the propagated state estimates and their covariance are optimally combined with the incoming measurements using the Kalman gain matrix **K**, which is given by

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T \left[\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R}\right]^{-1}. \tag{2.64}$$

The state estimates and covariances are updated with the Kalman gain matrix from (2.64) using

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left[\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-\right], \tag{2.65}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\,\mathbf{P}_k^-. \tag{2.66}$$

As shown in Equations (2.65) and (2.66), the Kalman gain matrix serves as an optimal weighting factor that gives adequate preference to either the propagated or measured estimates, given their individual uncertainties, in order to minimize mean squared error.

2.5.2 Extended Kalman Filter.

If a particular system cannot be adequately represented using linear dynamics or measurement models, the linear KF algorithm does not guarantee optimal solutions. However, in certain cases, linear approximations to nonlinear systems can still yield accurate estimates. In such cases, the Extended Kalman Filter (EKF) is used. The basic system dynamics equation for a nonlinear system are given by

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left[\mathbf{x}(t), \mathbf{u}(t), t\right] + \mathbf{G}(t)\mathbf{w}(t), \tag{2.67}$$

where **f** is a vector containing functions which represent the system. In turn, the nonlinear measurement equation is given by

$$\mathbf{z}_k = \mathbf{h}\left[\mathbf{x}_k, t_k\right] + \mathbf{v}_k, \tag{2.68}$$

where **h** is a vector of functions which model the sensor. The main goal is to linearize nonlinear models about their nominal estimates in order to use the conventional linear Kalman update equations. To do so, the states are redefined using the perturbation model given by

$$\delta\mathbf{x}(t) \triangleq \mathbf{x}(t) - \hat{\mathbf{x}}(t), \tag{2.69}$$

where $\delta\mathbf{x}(t)$ represents the difference between the true state vector and its estimate. In order to propagate the system from initial conditions or a previous measurement to the time of the next measurement, the EKF integrates the nonlinear dynamics function over the discrete time difference using

$$\hat{\mathbf{x}}_{k+1}^{-} = \int_{t_k}^{t_{k+1}} \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]\mathrm{d}t + \hat{\mathbf{x}}_k^{+}, \tag{2.70}$$

while the state covariance matrix is propagated using (2.58), (2.63) and a linearized dynamics model matrix $\mathbf{F}$ given by

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^{+}}. \tag{2.71}$$

In order to update the propagated state estimates using incoming (possibly nonlinear) measurements, the measurements must first be predicted by evaluating the measurement model function with the most recent estimate using

$$\hat{\mathbf{z}}_k = \mathbf{h}\left[\hat{\mathbf{x}}_k^{-}, t_k\right], \tag{2.72}$$

$$\delta\mathbf{z}_k = \mathbf{z}_k - \hat{\mathbf{z}}_k, \tag{2.73}$$

where $\delta\mathbf{z}$ is called the pre-update measurement residual, or innovation, and represents the difference between the actual and predicted measurements.

In order to combine the propagated and measured state estimates, the nonlinear measurement function $\mathbf{h}$ is linearized to obtain $\mathbf{H}_k$ in a similar fashion to $\mathbf{F}_k$ using

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^{-}}. \tag{2.74}$$

The linearized matrix $\mathbf{H}$ is then used in (2.64) to obtain a Kalman gain matrix, and the measurement update equation reduces to

$$\delta\hat{\mathbf{x}}_k^{+} = \mathbf{K}_k\delta\mathbf{z}_k, \tag{2.75}$$

due to the use of perturbation state estimates and measurements. The perturbation state $\delta\hat{\mathbf{x}}$, which starts at zero during each filter recursion, is updated using (2.75) and added to the nominal trajectory to produce a nominal estimate. Prior to the next recursion, the perturbation state is reset to zero. Since the EKF does not guarantee optimal solutions, it must often be tuned prior to use by adding process noise and selecting specific initial conditions. Tuning increases filter stability and usually increases solution uncertainty.

III. Autonomous and Resilient Management of All-source Sensors for Navigation

This chapter develops the general strategy for providing all-source multi-sensor resiliency, assurance, and integrity through an autonomous sensor management framework. The proposed framework dynamically places each sensor in an all-source system into one of four modes: monitoring, validation, calibration, and remodeling. Each mode contains specific and novel realtime processes that affect how a navigation system responds to sensor measurements. The framework is developed by first defining the set of desired resilient sensor management objectives, then developing novel approaches for achieving these objectives in an all-source and real-time environment, and finally by interlacing the processes governing each desired objective using a common application programming interface, all in a fault- and sensor-agnostic manner. The benefits of using the proposed framework are demonstrated by comparing all-source navigation performance against conventional filtering using two simulated scenarios including multiple sequential sensor failures and incorrectly modeled sensors. The research developed in this chapter has been published in [52] and [54].

3.1 Introduction

Over the past two decades, a significant portion of navigation research has been devoted to alternative means of precision navigation and timing, through the modeling and testing of non-traditional sensors (e.g., vision [97], radio [27], magnetic [22], etc.). As research matures in each of these sensor areas, multi-sensor alternative navigation is quickly becoming an operational possibility. However, with each additional sensor allowed into a navigation system comes the increased possibility of corrupting the navigation solution due to sensor model misspecification and undetected sensor failures or anomalies. Therefore, a robust method of managing sensors with questionable models is now necessary to ensure navigation solutions are accurate and resilient against errors in sensor modeling, unexpected signal interference, or undetected sensor faults, all in a plug-and-play or online fashion.

Some research has been conducted in the areas of managing navigation sensors for computational requirements [25], managing self-correcting Simultaneous Localization and Mapping (SLAM) sensors [65], and validating ad-hoc sensor networks [82]. However, there are currently no frameworks, to the author's knowledge, that formalize the definitions and interface between the processes of detecting faulty sensors, estimating sensor model parameters, and adapting sensor model functions, all in an online fashion and while protecting the integrity of the navigation solution. There is, however, compartmentalized research in related areas, where common navigation challenges related to multi-sensor navigation and sensor modeling have been solved, albeit in limited scope and usually aimed at specific sensor technologies. These related research areas are summarized below.

The first and arguably most critical step in resilient sensor management is fault detection and exclusion. Multi-sensor fault detection and exclusion has been traditionally accomplished through statistical analysis of redundant snapshot measurements [36][86][93], solution separation vectors [15][16][20][21][66][108][109], or filtered residuals [11][12][13][110]. However, most research has focused on the specific multi-sensor problem posed by the GPS constellation, where each satellite is regarded as a different (albeit identical in nature and synchronous) sensor in the multi-sensor system, and the "fault" is defined as an unmodeled bias that is assumed to only affect one of the sensors (satellites) at any given time; conditions which are not guaranteed in an all-source environment.

Nonetheless, detection and exclusion of a faulty sensor is only the first step of the problem in creating a resilient sensor management system. Since many sensor faults may be caused by model misspecification or temporary anomalies, there also exists a need to recover these failure modes (and continue using the sensor in question) through the autonomous and online modification of the specified sensor model.

One common method for overcoming sensor misspecifications is to estimate variable sensor model parameters that may have changed during navigation (e.g., lever arms, rotation matrices, scale factors, etc.). This type of sensor model modification is often referred to as calibration. There are a large number of online calibration examples across many sensor fields such as magnetometers [10], accelerometers [105], gyroscopes [67], lasers [71], audio sensors [79], Pitot-static sensors [56], to name a few. One such research area with significant recent advances is online calibration of a VINS [29][47][64][74][106]. As mature as these calibration research areas may be, they still only tend to focus on a particular sensor or sensor combination (e.g., visual and inertial or magnetic and inertial), and often do not address the tasks of detecting the need for calibration and independently evaluating the effectiveness of the calibration results. Additionally, they do not adequately address other types of sensor model modifications that may be needed for resiliency.

A second class of methods for overcoming sensor misspecifications is to alter the functional form of the sensor model to account for missing parameters or changing environmental conditions (e.g., time-changing biases, stochastic clock errors, temperature effects, etc.). This type of sensor model modification is often referred to as multiple-model estimation or model identification. Current literature in this area tends to be divided between continuous estimation of sensor and process noise covariances [3][14][99] and multiple model estimation using a finite set of competing models [24][30][43]. However, most techniques tend to focus on permanent failure modes that require the model identification process to run continuously. Similar to online calibration, these techniques also tend to lack independent validation of model identification results.

This chapter proposes a novel framework that contributes both a common language and a set of critical functions and their interactions, that together provide sensor-agnostic, statistically rigorous, and resilient sensor management. The proposed framework combines fault detection and exclusion, sensor model validation, online calibration, and online model identification into four interconnected modes of operation: monitoring, validation, calibration, and remodeling. In doing so, it is able to provide resilient and assured navigation for all-source applications, thereby directly enabling continued navigation operations across a greater range of sensor anomalies. The complete set of resiliency functions or objectives directly enabled through the proposed framework is summarized in Table 3.1. The remainder of this chapter is divided into three sections. Section 3.2 describes related work in the areas of sensor management frameworks, fault detection and identification, multiple-model estimation, and sensor model validation. Section 3.3 illustrates the proposed framework and modes of operation in detail, and provides simulation results for two multi-sensor navigation scenarios. Finally, Section 3.4 summarizes the contributions of the research in this chapter and areas for future work.

3.2 Related Work

Though no framework combining the aforementioned functions from Table 3.1 been found in literature, several works have been found whose contributions were leveraged as part of the framework development process. These works are briefly described below.

Statistical hypothesis testing forms an integral part of the fault detection, model validation, and adaptive estimation tasks. In general, most hypothesis tests can be stated using an LRT [63], by assigning competing statistical distributions to each of the hypotheses in the test. Such LRTs are useful since their distribution tends to be Chi-Square [23], regardless of the distributions of the competing hypotheses, and especially if the competing hypotheses are assumed to be normally distributed. Integrity monitoring and fault detection and exclusion research in the area of GPS and INS integration such as [19][20][37][48][49][78][86][100][103] use LRTs and test statistics to detect faulty

Table 3.1: List of desired resilient sensor management objectives.

Mode	Section	Resilient sensor management objectives
Monitoring	3.3.1.2	Provide sensor-agnostic fault detection and exclusion
		Ensure fault-agnostic system integrity
		Trigger sensor model remedial measures
Validation	3.3.1.3	Initialize offline sensors without compromising integrity
		Validate questionable sensor models in real-time
		Provide independent verification of remedial measures
Calibration	3.3.1.4	Augment state-space with specified model parameters
		Follow a prescribed estimation sequence for observability
		Reduce state-space after specified termination criteria
Remodeling	3.3.1.5	Dynamically spawn multiple-model filter bank
		Select best candidate model using statistical criteria
		Delete filter bank after specified termination criteria

GPS measurements, and more importantly, predict system performance in the presence of undetected faults. Meanwhile, the work in [110] makes use of multiple filters to identify biased satellite measurements and exclude them from affecting the solution, thereby providing a form of system integrity. Leveraging the existing research in the area of GPS, our monitoring mode contains a novel approach for all-source fault detection and exclusion, and integrity monitoring, which is developed and validated in Chapter 4.

While our monitoring test provides the ability to detect and exclude faults across the set of online sensors (i.e., sensors currently informing the navigation solution), there also exists a need to initialize offline sensors by validating their stated measurement models while protecting the navigation solution. This challenge, which we are referring to as the sensor validation problem, is virtually unaddressed (at the time of writing) in current research. The validation problem is further complicated by the fact that many navigation sensors require the estimation of additional states needed in their measurement model (e.g., clock errors, biases, scale factors, etc.), and estimating these additional states would require allowing sensor measurements from the sensor in question to affect the navigation solution. Leveraging the partial update implementation [17] of the Kalman-Schmidt filter [81], our validation mode contains a novel approach for the real-time validation of offline sensors that allows for the estimation of sensor-unique states while protecting the navigation solution and maintaining system integrity. This method is developed and validated in Chapter 5.

Fault detection and exclusion, and independent model validation comprise just two (albeit the most important) objectives in our resilient sensor management framework. The goal, especially in the area of emerging and alternative sensors, is not only to detect and prevent mismodeled sensors from affecting the navigation solution, but also to dynamically modify their stated models in order to enable their continued use. The research in [65] proposes a multi-phased process that accomplishes a subset of tasks from our proposed framework. Namely, the VINS calibration method proposed therein continually estimates camera extrinsic parameters and statistically compares, via a LRT, their short-term and long-term estimates. If the short-term and long-term estimates are statistically different, a three-phased calibration routine is initiated. Finally, the calibration routine is terminated once the covariance of the estimated parameters exhibits desired convergence criteria. In the context of our proposed framework, the research in [65] provides a VINS-centric sensor manager that meets a subset of the objectives contained in our monitoring mode, and most of the objectives contained in our calibration mode. However, it is focused on a single sensor and is specifically designed to work with VINSs. Additionally, it does not provide a method for independent validation of the calibration results.

When perceived sensor faults are not recoverable by re-calibrating the sensor model parameters, we may also consider the possibility of an incorrectly stated measurement function. Since a particular measurement function may be incorrectly stated in an infinite number of ways, the task of finding the most appropriate sensor model function is usually solved by fitting a finite set of models in a multiple model technique such as [30][43]. Alternatively, some may choose to estimate certain stochastic model parameters continuously such as [3][14][99]. All of these methods for overcoming incomplete or incorrectly stated model functions provide general examples of the objectives contained in our remodeling mode. The specific remodeling technique shown in our example scenarios from Section 3.3.2 uses a parallel filter bank with multiple parallel models, much like the aforementioned research. However, it also uses statistical model selection criteria such as AIC [2] to select the most likely model, and additionally independently validates the selection results using our test in the validation mode.

3.3 An Autonomous and Resilient Sensor Manager

We now introduce a novel, autonomous method for resilient sensor management that performs all the previously identified functions by expounding on the building blocks contained in the previous research from Section 3.2, and coherently weaving their functionality into a sensor-agnostic framework. The proposed framework, henceforth referred to as Autonomous and Resilient Management of All-source Sensors (ARMAS), statistically evaluates sensor performance and places each sensor into one of four operating modes: monitoring, validation, calibration, and remodeling. ARMAS then provides resilient sensor management by controlling how a navigation filter responds to measurements from a particular sensor based on that sensor's mode. Table 3.1 (shown previously) summarizes the key functionality provided by each mode of operation.

3.3.1 Framework Implementation.

In general, the ARMAS framework is designed around an online or plug-and-play environment, applies to all navigation sensors, does not require sensor-specific tuning, and can be adapted to any filtering technique. The examples illustrated in Section 3.3.2 were developed in MATLAB® using the SCORPION estimation framework [61] for filter spawning and measurement processing. It is important to note that the proposed framework was designed around the plug-and-play concept of operation, assuming sensors are serially added onto an ongoing navigation process. The only assumption imposed onto the initial ongoing navigation process is that the navigation solution is consistent (i.e., its estimates are unbiased, and the estimated error covariance matches actual performance). This assumption does not preclude a "weak" sensor (i.e., large measurement error covariance) from being the only sensor in the system. It simply dictates that if there is only a single sensor in the system when a new sensor is added, its residual statistics are assumed to be accurate. Figure 3.1 illustrates a proposed state transition diagram that coherently transitions sensors through the various modes of operation. The following sections expound on each of the modes of operation and provide general guidelines for proper implementation.

To facilitate further discussion on ARMAS development, we will adapt the stochastic estimation convention from [76] for multi-sensor applications. Consider a navigation problem of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left[\mathbf{x}(t), \boldsymbol{\epsilon}(t), \mathbf{u}(t), t\right] + \mathbf{G}(t)\mathbf{w}(t), \tag{3.1}$$

where $\mathbf{x}$ is the $N \times 1$ navigation state vector containing the vehicle's core navigation states (position, velocity, attitude, time, etc.), $\boldsymbol{\epsilon}$ is an $M \times 1$ vector containing additional states needed to account for measurement errors, $\mathbf{u}$ is the control input vector, $\mathbf{G}$ is an $(N+M)\times W$ linear operator, and $\mathbf{w}$ is a $W \times 1$ white Gaussian noise process with a $W \times W$ continuous process noise strength matrix $\mathbf{Q}$. The discretized [96] non-linear system is then solved using the EKF algorithm [76][77]. The state estimates are propagated using the (possibly) non-linear state dynamics model, and updated using measurements from available sensors. In a multi-sensor environment with J sensors, the jth sensor provides Z-dimensional discrete measurements, $\mathbf{z}_k^{[j]}$, at time t_k, which are modeled as

$$\mathbf{z}_k^{[j]} = \mathbf{h}^{[j]}\left[\mathbf{x}(t), \boldsymbol{\epsilon}^{[j]}(t), \mathbf{u}(t), t, \mathbf{p}^{[j]}\right] + \mathbf{v}_k^{[j]}, \tag{3.2}$$

where $\mathbf{h}^{[j]}$ is a (possibly) non-linear measurement function for sensor j, $\boldsymbol{\epsilon}^{[j]}$ is a $L \times 1$ ($L \leq M$) subset of $\boldsymbol{\epsilon}$ consisting of the additional states required for processing measurements for sensor j (e.g., a clock error process, constant bias, etc.), $\mathbf{p}^{[j]}$ is a $P \times 1$ vector of observable model parameters for $\mathbf{h}^{[j]}$ selected by the user (e.g. a lever arm, scale factor, etc.), and $\mathbf{v}_k^{[j]}$ is a $Z \times 1$ discrete white Gaussian noise process with covariance matrix $\mathbf{R}_k^{[j]}$. Given the estimated quantities $\hat{\mathbf{x}}_k^-$, $\hat{\boldsymbol{\epsilon}}_k^{[j]^-}$, $\hat{\mathbf{p}}^{[j]}$, the $Z \times 1$ measurement residual, $\mathbf{r}_k^{[j]}$, at time $t = t_k$ for sensor j is given by

$$\mathbf{r}_k^{[j]} = \mathbf{z}_k^{[j]} - \mathbf{h}^{[j]}\left[\hat{\mathbf{x}}_k^-, \hat{\boldsymbol{\epsilon}}_k^{[j]^-}, \mathbf{u}_k, t_k, \hat{\mathbf{p}}_k^{[j]}\right]. \tag{3.3}$$

Additionally, the residual vector from (3.3) is expected to follow the distribution

$$\mathbf{r}_k^{[j]} \sim \mathcal{N}\left(\underset{Z\times 1}{\mathbf{0}}, \mathbf{S}_k^{[j]}\right), \tag{3.4}$$

$$\mathbf{S}_k^{[j]} = \mathbf{H}_k^{[j]}\mathbf{P}_k^-\mathbf{H}_k^{[j]^{\mathrm{T}}} + \mathbf{R}_k^{[j]}, \tag{3.5}$$

where $\mathbf{H}_k^{[j]}$ is the $Z \times (N + M)$ Jacobian of $\mathbf{h}^{[j]}$ about the current estimate, and $\mathbf{P}_k^-$ is the $(N + M) \times (N + M)$ state estimation error covariance matrix at time $t = t_k$. In this context, the goal of ARMAS is to ensure incoming measurements $\mathbf{z}_k^{[j]}$ truly adhere to their stated models by analyzing the statistical distribution of their residuals; and if not, protecting the core navigation solution, $\hat{\mathbf{x}}$, while attempting to modify $\boldsymbol{\epsilon}^{[j]}$ and/or $\mathbf{p}^{[j]}$ to enable continued sensor use.

3.3.1.1 Sensor Initialization.

As previously stated, ARMAS places each sensor into one of four operating modes plus a failed state. In a plug-and-play environment with an ongoing navigation process, each new sensor is initialized into one of two modes: monitoring or validation. This initial placement is based on how confident users may be in the current model available for a particular sensor. For example, sensors that have well understood models and stochastic processes, such as a GPS receiver in a threat-free environment, could be considered "trusted", and placed directly into monitoring mode, while emerging alternative sensors with more questionable error models could be considered "untrusted" and placed into validation mode. To take advantage of this functionality, an ARMAS user can specify the initial trust for each sensor (e.g., trusted or untrusted), or provide a default setting for all new sensors.

3.3.1.2 Monitoring Mode.

Monitoring mode is used to detect, identify, and exclude faulty sensors in a multi-sensor problem. In this context, a faulty sensor refers to a sensor whose stated model is not consistent with its observed performance, which could be caused by temporary or permanent failures, as well as dynamic changes to the sensing environment (e.g., atmosphere, terrain, multi-path, etc.). Sensors in monitoring mode are able to fully affect the navigation solution provided to the user since they are trusted. This poses a challenge to detecting model divergence using a single-filter solution, since the effects of a mismodeled sensor on the navigation solution are not easily attributable to a particular sensor. An example of this problem is a strong (i.e., low measurement noise strength) sensor with an unmodeled bias that "pulls" the filter solution away from truth to absorb the bias. In that case, the faulty sensor's residuals would be statistically valid, while potentially making the residuals from weaker yet not faulty sensors, invalid. Any method that can robustly detect, attribute, and exclude sensor faults can be used in this mode. For our particular all-source implementation, we created a novel multi-filter, sensor-agnostic fault detection and exclusion approach that does not constrain faults to biases, or to single sensors, and can guarantee system performance with a specified false alarm rate, α, and integrity error

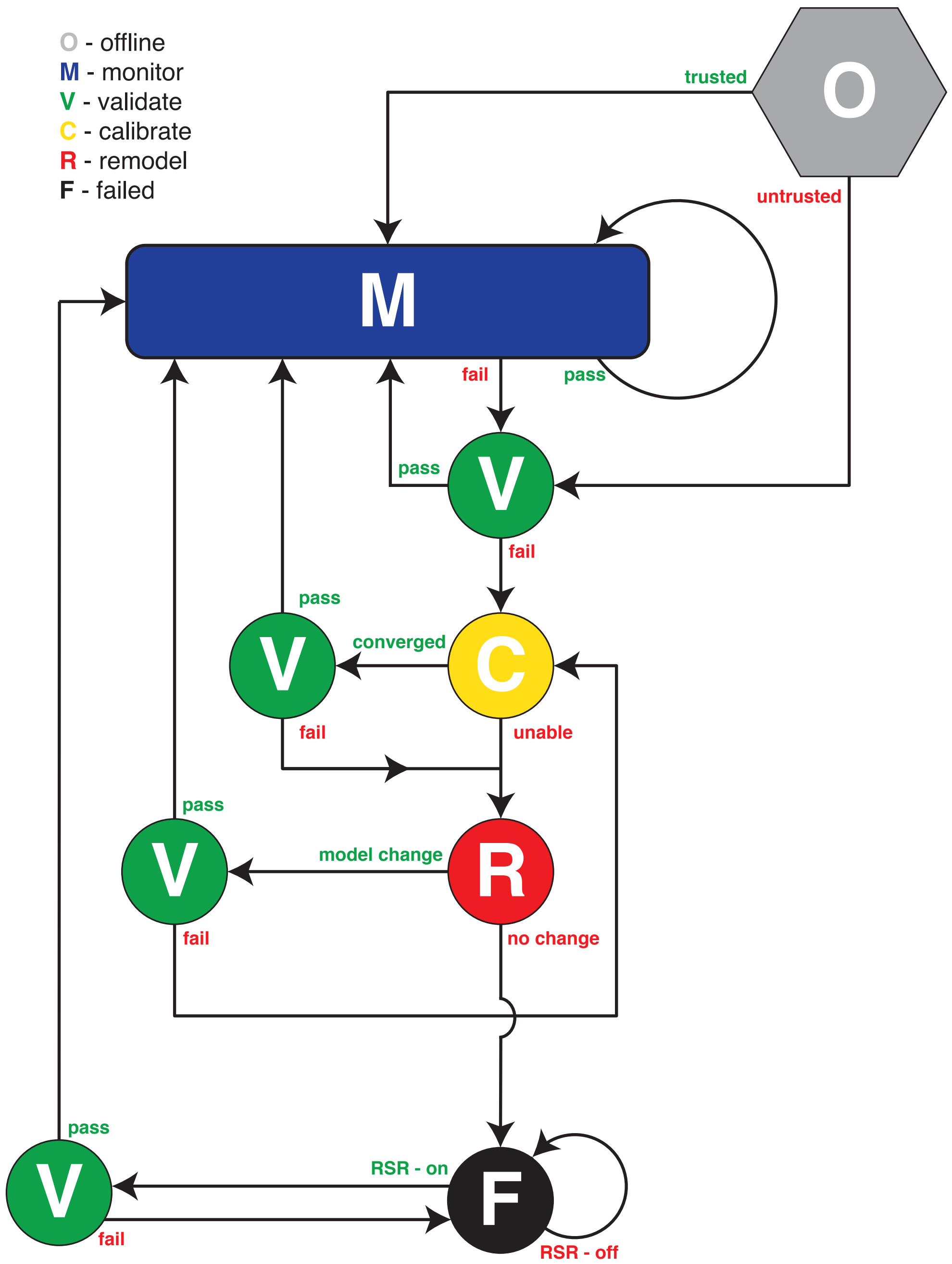


Figure 3.1: Proposed state transition diagram for ARMAS framework. New sensors (offline) begin in either validate or monitor mode. A failed monitoring test starts a calibration and remodeling loop, whose effectiveness is evaluated by the validation test. The loop continues until the validation test is passed or the sensor model selection is unchanged. Failed sensors can be periodically re-validated to recover from temporary anomalies using RSR.

bound, α_I. The specific details of our implementation are found in Chapter 4 and [58]. The monitoring test is run continuously for every sensor in monitoring mode, with a test decision produced for each sensor at a user-defined rate. The test is failed if a fault is detected and attributed to the sensor in question, and passed otherwise. If the test is passed, the sensor in question remains in monitoring mode. Otherwise, the sensor is no longer allowed to affect the solution and is placed into validation mode, where its stated model is independently validated against the other (trusted) sensors. To take advantage of monitoring functionality, an ARMAS user needs to specify the following:

1. The monitoring period (e.g., number of samples or time elapsed between tests).
2. The fault detection false alarm rate, α.
3. The integrity monitoring error bound, α_I.

3.3.1.3 Validation Mode.

Validation mode is used to statistically validate an untrusted sensor model using information from trusted sensors. Sensors in validation mode are only able to affect state estimates for their unique states, $\boldsymbol{\epsilon}^{[j]}$, while the core navigation states, $\mathbf{x}$, are only affected by trusted sensors (i.e., sensors in monitoring mode). This poses a challenge when initializing or processing measurements from an untrusted sensor since we are to consider the stochastic distribution of all filter states while only allowing an untrusted measurement update from sensor j to affect $\boldsymbol{\epsilon}^{[j]}$. An example of this challenge would be estimating a receiver clock error without allowing the receiver to affect the navigation solution, and while preventing the clock error estimate from absorbing any other missing error sources. To solve this challenge in a plug-and-play environment, we developed a novel method for initialization and validation of a sensor with a questionable sensor model that can estimate sensor-unique states, $\boldsymbol{\epsilon}^{[j]}$, while protecting the integrity of the core navigation solution, $\mathbf{x}$, using only the existing main filter. Employing the partial update [17] formulation of the Kalman Schmidt filter [81], measurement updates from a sensor in validation mode are only able to affect its sensor-unique states, $\boldsymbol{\epsilon}^{[j]}$, while measurement updates from sensors in monitoring mode (i.e., trusted) are able to affect all filter states. This not only protects the core navigation solution during the estimation of $\boldsymbol{\epsilon}^{[j]}$, but it also improves fault detection performance since sensor faults are prevented from being absorbed into the core navigation solution. The specific details of our implementation are found in Chapter 5 and [57]. Similar to monitoring mode, a validation test is run continuously for every sensor in validation mode, with a test decision produced at a user-defined rate. The mode transition from validation mode is determined by both the results of the validation test and the previous ARMAS mode of the sensor in question. As illustrated in Figure 3.1, validation mode is used to externally validate sensor models after failing a monitoring test, completing a calibration sequence, and completing a remodeling selection. Additionally, validation mode can be used to periodically re-evaluate permanently failed sensors to check for passage of temporary anomalies that might have caused an insuperable failure. This process is referred to as Resilient Sensor Recovery (RSR). To take advantage of validation functionality, an ARMAS user needs to specify the following:

1. The validation period (e.g., number of samples or time elapsed between tests).

2. The fault detection false alarm rate, α.

3. Which states used in $\mathbf{h}^{[j]}$ are core navigation states (i.e., $\mathbf{x}$).

4. Which states used in $\mathbf{h}^{[j]}$ are sensor-unique states (i.e., $\boldsymbol{\epsilon}^{[j]}$).

5. If RSR is enabled, the RSR period (i.e., how often to attempt post-fail validation).

3.3.1.4 Calibration Mode.

Calibration mode is used to dynamically re-estimate variable model parameters when a sensor model has been identified as faulty. Sensors enter calibration mode after having failed an initial or post-monitoring validation test, and are considered untrusted until they can be validated and placed back into monitoring mode. In calibration mode, the functional form of the sensor model is assumed correct, but certain model parameters, $\mathbf{p}^{[j]}$, inside the measurement model function from (3.2) are to be re-estimated. The parameter estimation may also require specific sequencing (i.e., estimate subsets of $\mathbf{p}^{[j]}$ in a specified order) in order to maintain observability, which in turn requires specifying sequencing criteria, or a method for determining when a step in the sequence is complete. An example of this type of problem would be the camera calibration algorithm described in [106], where the camera extrinsic parameters (lever arm and orientation) are estimated separately (orientation first, then lever arm) in order to maintain observability, and the transition between sequence steps is driven by convergence of the associated state covariance matrix. For our particular implementation, we generalized the calibration process into a set of simple instructions provided by users such that any sensor calibration method that can be expressed in terms of a sequenced state vector augmentation and corresponding sequence transition criteria can be easily and autonomously executed. The parameter estimation is performed using a calibration sub-filter that is initialized based on a copy of the main filter whenever a sensor enters calibration mode. Once all steps in the stated calibration sequence are completed, the sensor in question is placed back into validation mode to externally validate the calibration results against trusted sensors as described above. It is also important to note here that since sensors in calibration mode are considered untrusted, their measurement updates during a calibration sequence are only able to affect estimates for $\boldsymbol{\epsilon}^{[j]}$ and $\mathbf{p}^{[j]}$, which tends to increase observability on $\mathbf{p}^{[j]}$ since estimates for $\mathbf{x}$ are provided by trusted sensors. To take advantage of calibration functionality, an ARMAS user needs to specify the following:

1. Which parameters in $\mathbf{h}^{[j]}$ are to be estimated during calibration (i.e., $\mathbf{p}^{[j]}$).
2. The initial estimate for each parameter (i.e., $\mathbf{p}_0^{[j]}(i),\ i = 1, \ldots, P$).
3. The initial uncertainty for each parameter (i.e., $\boldsymbol{\sigma}_0^{[j]}(i),\ i = 1, \ldots, P$).

4. The sequence group for each parameter (i.e., when to start estimating).

5. Transition criteria for each parameter (i.e., when to stop estimating).

Subsequently, if the sensor in question enters calibration mode, ARMAS automatically augments the filter states using the provided initial estimates and uncertainties for the first group in the sequence, then transitions to the next group once every group member has met its transition criteria, until all groups are completed.

3.3.1.5 Remodeling Mode.

Remodeling mode is used to dynamically modify the functional form of the measurement model when a sensor model has been identified as faulty. Sensors enter remodeling mode after failing a post-calibration validation test or if a calibration routine was not provided. Similar to other modes, sensors in remodeling mode are considered untrusted until they can be validated and placed back into monitoring mode. In remodeling mode, the functional form of the sensor model is now assumed incorrect. This poses a challenge since a measurement function $\mathbf{h}^{[j]}$ could be incorrectly stated in an infinite number of ways. To solve this, we leverage user experience for each sensor application to create a reasonable and finite set of S model options that are dynamically evaluated if the sensor in question enters remodeling mode. Then, we use a statistical test to select the best model from the set. An example of this type of problem would be a VINS user that initially only models radial lens distortion, but includes the option to also model tangential lens distortion if the camera sensor ever enters the remodeling mode. Similarly, a laser range finder user may elect to initially attempt a First Order Gauss-Markov (FOGM) error model, but include the option to try second-order, constant bias, or the combination thereof, if the sensor ever enters remodeling mode. Any method that can evaluate multiple models and statistically select the most appropriate one can be used in this mode. For our particular implementation, we leveraged Multiple Model Adaptive Estimation (MMAE) research such as [24][30][43], but used Akaike Information Criterion (AIC) [2] for the model selection decision since it balances model complexity with error reduction. Additionally, we only evaluate multiple models when the sensor in question enters remodeling mode, spawning S remodeling sub-filters initialized as copies of the main filter, and collecting residuals using (3.3) until all remodeling termination criteria are met. Once a model selection decision is made, the sensor in question along with the selected model are placed into validation mode. Similarly to calibration mode, sensors in remodeling mode are untrusted and only able to affect estimates of $\boldsymbol{\epsilon}^{[j]}$ and $\mathbf{p}^{[j]}$, which strengthens the model selection process since it is heavily influenced by external and trusted sensors. To take advantage of remodeling functionality, an ARMAS user needs to specify the following:

1. A list of candidate measurement models (i.e., $\mathbf{h}_i^{[j]}$ $i = 1, \dots, S$).
2. The initial estimate for all sensor-unique states in each model (i.e., $\boldsymbol{\epsilon}_{i_0}^{[j]}$ $i = 1, \dots, S$).
3. The initial covariance for all sensor-unique states in each model (i.e., $\boldsymbol{\Sigma}_{i_0}^{[j]}$, $i = 1, \dots, S$).
4. The remodeling termination criteria (e.g., number of samples, time period, covariance, etc.).

Subsequently, if the sensor in question enters remodeling mode, ARMAS automatically spawns S parallel filters using the main filter statistics for the core states, $\mathbf{x}$, and the provided initial estimates and covariances for each set of sensor-specific states in each candidate model.

3.3.1.6 Implementation Summary.

As shown in the previous sections, ARMAS provides an autonomous method for implementing various sensor model management functions given a minimal set of additional specifications for each sensor beyond the usual modeling requirements. It is important to note, certain functions including multi-sensor fault detection and identification as well as RSR, are available without any additional sensor-centric specifications, and using only default settings. Table 3.2 summarizes the additional information required to enable ARMAS functionality for each mode of operation.

3.3.2 Example Scenarios.

This section describes two example scenarios that relied on the ARMAS framework for resilient sensor management. It is important to note that these are simply two examples of how the proposed framework could be used. A user of the framework has the flexibility to adopt the framework to the specific problem at hand. In both scenarios, the use of ARMAS enabled continued operations in cases that would have resulted in either significant navigation solution errors or irrecoverable sensor failures. Consider a two-dimensional navigation problem with two vehicles (Aircraft 1 and Aircraft 2) obtaining their navigation solutions from an EKF driven by a 2D kinematic model given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{x}}_p(t) \\ \dot{\mathbf{x}}_v(t) \\ \dot{\mathbf{x}}_a(t) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_v(t) \\ \mathbf{x}_a(t) \\ -\frac{1}{\tau_a}\mathbf{x}_a(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{w}(t) \end{bmatrix}, \tag{3.6}$$

where $\mathbf{x}_p$ is the vehicle's 2D position in [m], $\mathbf{x}_v$ is the 2D velocity in [m/s], $\mathbf{x}_a$ is the 2D acceleration in [m/s^2] and driven by a FOGM process with time constant $\tau_a = 90$ [s], and $\mathbf{w}(t)$ is a 2D white Gaussian noise process with $E\left[\mathbf{w}(t)\mathbf{w}(t+\tau)^{\mathrm{T}}\right] = \mathbf{Q}\delta(\tau)$ and

$$\mathbf{Q} = (1.5 \times 10^{-3})^2 \underset{2\times 2}{\mathbf{I}} \text{ [m}^2\text{/s}^5\text{]}. \tag{3.7}$$

The initial state estimate, $\hat{\mathbf{x}}(0)$, and state error covariance, $\mathbf{P}(0)$, for both vehicles at the beginning of each scenario is given by

$$\hat{\mathbf{x}}(0) = \begin{bmatrix} 0 & 0 & 100 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{3.8}$$

$$\mathbf{P}(0) = \mathrm{diag}\left(\begin{bmatrix} 1 & 1 & 10 & 10 & 1.5 \times 10^{-3} & 1.5 \times 10^{-3} \end{bmatrix}^2\right). \tag{3.9}$$

Each vehicle obtains discrete measurement updates from three sensors (Sensor A, Sensor B, and Sensor C). Sensor A is a two-dimensional position sensor with a model given by

$$\mathbf{z}_k^{[A]} = \mathbf{s} \odot \mathbf{x}_{p_k} + \mathbf{v}_k^{[A]}, \tag{3.10}$$

$$\mathbf{v}_k^{[A]} \sim \mathcal{N}\left(\underset{2\times1}{\mathbf{0}}, 100^2 \underset{2\times2}{\mathbf{I}}\right), \tag{3.11}$$

where $\mathbf{s} = [s_x \;\; s_y]^{\mathrm{T}}$ is a two-dimensional scale factor, $\odot$ denotes the Hadamard product, and $\mathbf{I}$ is an identity matrix. Sensor B is a two-dimensional, eight-satellite pseudorange sensor with a model given by

$$\mathbf{z}_k^{[B]} = \begin{bmatrix} \left\|\mathbf{t}_1 - \mathbf{x}_{p_k}\right\| \\ \vdots \\ \left\|\mathbf{t}_8 - \mathbf{x}_{p_k}\right\| \end{bmatrix} + \begin{bmatrix} b_k \\ \vdots \\ b_k \end{bmatrix} + \mathbf{v}_k^{[B]}, \tag{3.12}$$

$$\mathbf{v}_k^{[B]} \sim \mathcal{N}\left(\underset{8\times1}{\mathbf{0}}, 20^2 \underset{8\times8}{\mathbf{I}}\right), \tag{3.13}$$

where $\mathbf{t}_i$ is the two-dimensional position of satellite i, $\mathbf{x}_{p_k}$ is the two-dimensional position of the vehicle at time t_k, and b_k is a FOGM process simulating a simple receiver clock error with time constant $\tau_B = 3600$ [s] and $\sigma^2 = 8000^2$ [m^2]. Finally, Sensor C is a two-dimensional velocity sensor with the model

$$\mathbf{z}_k^{[C]} = \mathbf{x}_{v_k} + \mathbf{v}_k^{[C]}, \tag{3.14}$$

$$\mathbf{v}_k^{[C]} \sim \mathcal{N}\left(\underset{2\times1}{\mathbf{0}}, 50^2 \underset{2\times2}{\mathbf{I}}\right). \tag{3.15}$$

3.3.2.1 Example 1: Temporary sensor anomaly.

In this scenario, ARMAS was used to detect the presence of a temporary anomaly in Sensor B. Aircraft 1 was equipped with a standard EKF while Aircraft 2 used ARMAS. ARMAS specifications for this scenario were as follows:

- The significance level for monitoring and validation was set to 0.05.
- The monitoring period was defined by time elapsed and set to 20 [s].
- The validation period was defined by time elapsed and set to 60 [s].

- The RSR period was defined by time elapsed and set to 60 [s].
- The core navigation states were defined as all states shown in (3.6) (e.g., $\mathbf{x}_p, \mathbf{x}_v, \mathbf{x}_a$).
- Sensor B included a sensor-unique state, b_k, as defined in (3.12).
- There were no calibration routines or remodeling options specified for any sensor.

At the start of the scenario, all sensors in Aircraft 2 were in monitoring mode. After five minutes, a temporary anomaly defined by an unmodeled bias, which grew from 0 [m] to 1500 [m] over 10 minutes, was applied to the fourth entry in $\mathbf{z}_k^{[B]}$ from (3.12). As shown in Figure 3.2, Aircraft 1 had no means to detect the growing Sensor B bias, causing its solution to drift from the truth. In contrast, Aircraft 2 was equipped with ARMAS and therefore, was able to quickly identify the mismatch between the solution versions across the three sensors, and identify Sensor B as the cause of the divergence. Next, as shown in Figure 3.3, Sensor B failed the monitoring test, and was placed into validation mode, where it also failed the validation test. Since no calibration routine or remodeling options were provided, Sensor B transitioned from validation mode to a failed state, where RSR periodically validated its performance. Aircraft 2 continued navigation using only Sensor A and Sensor C, as indicated by the increase in position covariance in Figure 3.2. After some time, the two aircraft physically transitioned away from the anomaly area, and the solution in Aircraft 1 quickly converged towards the truth. Meanwhile, Aircraft 2 continued to navigate using only Sensor A and Sensor C until RSR led to a passing validation test. Once Sensor B was validated, Aircraft 2 returned to navigation with all sensors in monitoring mode. Table 3.3 compares the solution performance between the two aircraft for this scenario. Note the large difference in Root Sum Squared (RSS) position error between the two aircraft at the point the temporary anomaly ended, as shown in Table 3.3.

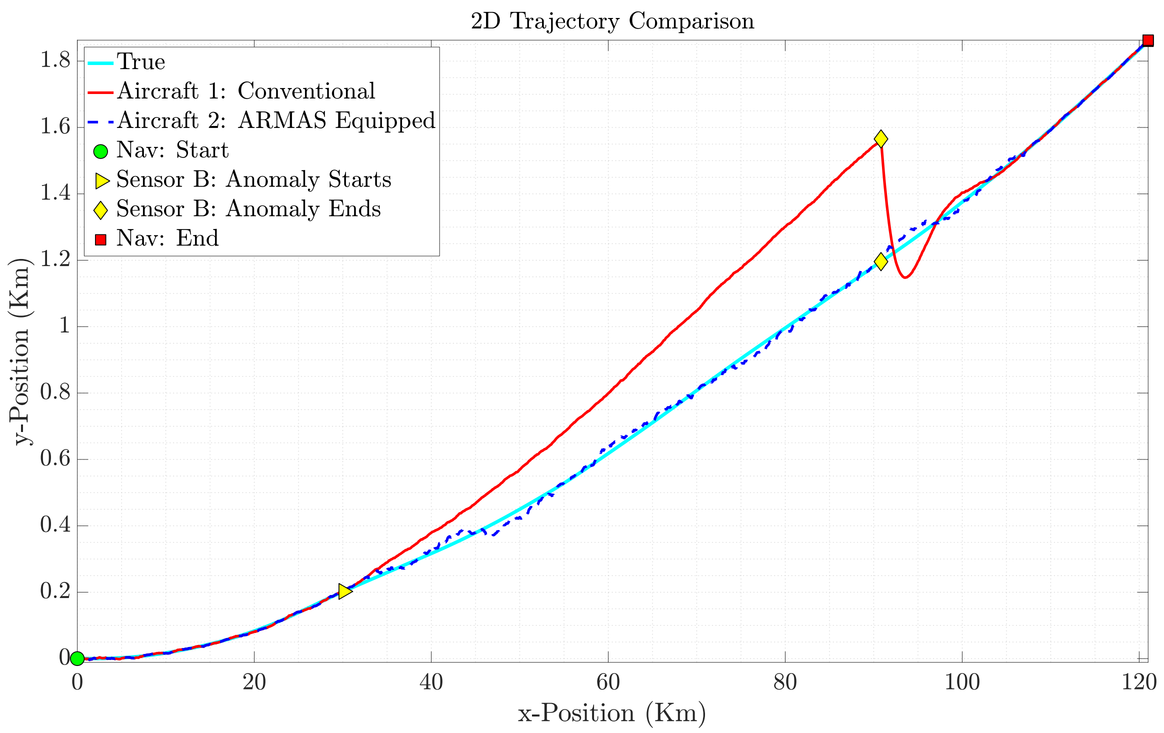


Figure 3.2: Trajectory comparison between Aircraft 1 and Aircraft 2, Example 1.

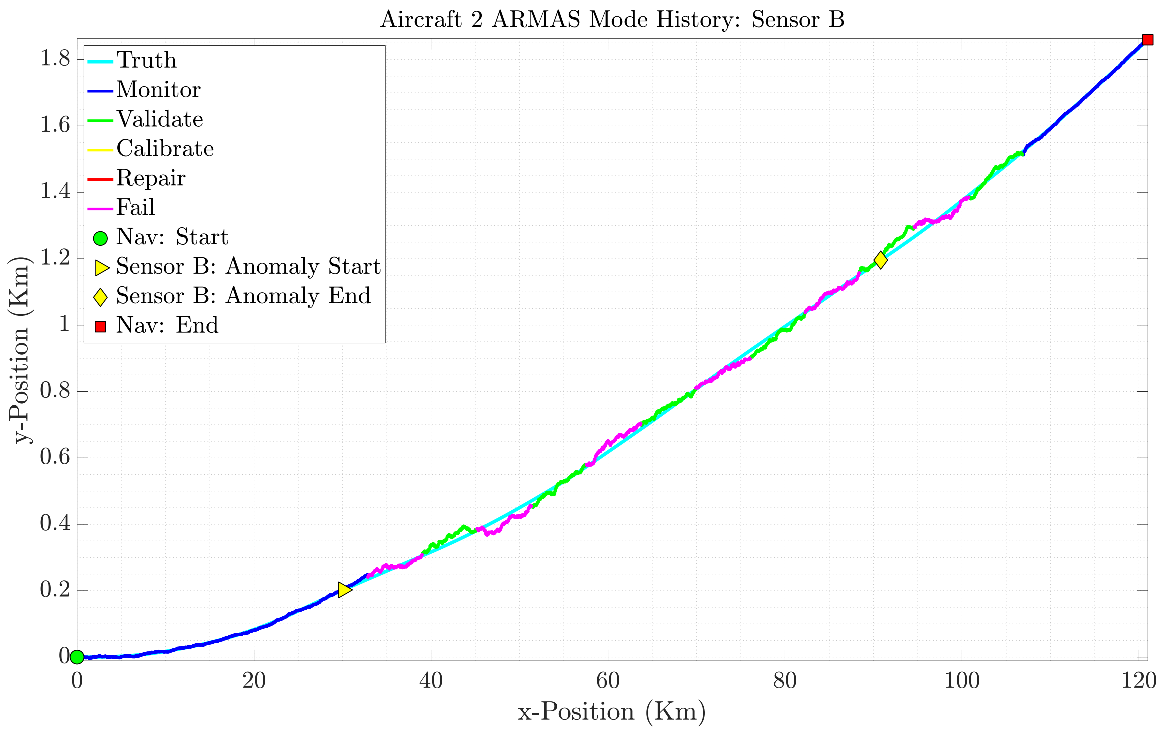


Figure 3.3: ARMAS mode history for Sensor B, Example 1.

Table 3.2: ARMAS example implementation summary.

		Required by[1]				
Parameter name	**Symbol**	**M**	**V**	**C**	**R**	**Example specification**
False alarm rate	α	●	●			0.001
Integrity error bound	α_I	●				0.05
Monitoring period		●				20 [s]
Validation period			●			60 [s]
RSR period			●			30 [s]
Core navigation states	$\mathbf{x}$		●	●	●	EKF States: 1-6
Sensor-unique states	$\boldsymbol{\epsilon}^{[j]}$		●	●	●	EKF States: 7-8
Calibration parameters[2]	$\mathbf{p}^{[j]}$			●		$\mathbf{h} = \left[s_x\mathbf{x}(1) \;\; s_y\mathbf{x}(2)\right]^{\mathrm{T}}$
Parameter values[2]	$\mathbf{p}_0^{[j]}$			●		$\mathbf{p}_0 = [1 \;\; 1]^{\mathrm{T}}$
Parameter uncertainty[2]	$\boldsymbol{\sigma}_0^{[j]}$			●		$\boldsymbol{\sigma}_0 = [10 \;\; 10]^{\mathrm{T}}$
Calibration sequence[2]				●		Group 1: $\{\mathbf{p}(1)\}$
						Group 2: $\{\mathbf{p}(2)\}$
Transition criteria[2]				●		$\mathbf{p}(1)$: 300 [s]
						$\mathbf{p}(2)$: 300 [s]
Candidate models[3]	$\mathbf{h}_i^{[j]}$				●	$\mathbf{h}_1 = [\mathbf{x}(1) + b_1 \;\; \mathbf{x}(2)]^{\mathrm{T}}$
						$\mathbf{h}_2 = [\mathbf{x}(1) + b_1 \;\; \mathbf{x}(2) + b_2]^{\mathrm{T}}$
Initial state estimates[3]	$\boldsymbol{\epsilon}_{i_0}^{[j]}$				●	$\boldsymbol{\epsilon}_{1_0} = 0$
						$\boldsymbol{\epsilon}_{2_0} = [0 \;\; 0]^{\mathrm{T}}$
Initial state covariances[3]	$\boldsymbol{\Sigma}_{i_0}^{[j]}$				●	$\boldsymbol{\Sigma}_{1_0} = 100$
						$\boldsymbol{\Sigma}_{2_0} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$
Termination criteria[3]					●	100 residual samples

[1] M - Monitoring, V - Validation, C - Calibration, R - Remodeling

[2] Example: 2D scale factor, $\mathbf{p} = [s_x \;\; s_y]$, each dimension estimated separately, time-based sequencing between dimensions

[3] Example: Model 1 adds 1D bias, Model 2 adds 2D bias, both models use sample size for termination

Table 3.3: Key events and RSS position error comparison, Example 1.

			RSS Position Error [m]		
Sensor	**Event**	**Time [m]**	**Aircraft 1**	**Aircraft 2**	**% Change**
All	Start	0	0	0	0
B	Anomaly: On	5	2.2	2.2	0
B	Anomaly: Off	15	369.6	10.9	-97.1
All	End	20	4.8	4.8	0

3.3.2.2 Example 2: Multiple sequential faults.

In this scenario, ARMAS was used to validate and remodel an untrusted sensor model for Sensor B, as well as detect the need to calibrate Sensor A. Again, Aircraft 1 was equipped with a standard EKF while Aircraft 2 used ARMAS. ARMAS specifications for this scenario were as follows:

- The significance level for monitoring and validation was set to 0.05.
- The monitoring period was defined by time elapsed and set to 20 [s].
- The validation period was defined by time elapsed and set to 60 [s].
- The RSR period was defined by time elapsed and set to 60 [s].
- The core navigation states were defined as all states shown in (3.6) (e.g., $\mathbf{x}_p, \mathbf{x}_v, \mathbf{x}_a$).
- Sensor B included a sensor-unique state, b_k, as defined in (3.12).
- Sensor A was equipped with a calibration routine defined by

$$\mathbf{p}^{[A]} = \begin{bmatrix} s_x & s_y \end{bmatrix}^{\mathrm{T}}, \tag{3.16}$$

$$\mathbf{h}^{[A]} = \begin{bmatrix} s_x \mathbf{x}_p(1) & s_y \mathbf{x}_p(2) \end{bmatrix}^{\mathrm{T}}, \tag{3.17}$$

$$\mathbf{p}_0^{[A]} = \begin{bmatrix} 1 & 1 \end{bmatrix}^{\mathrm{T}}, \tag{3.18}$$

$$\sigma_0^{[A]} = \begin{bmatrix} 10 & 10 \end{bmatrix}^{\mathrm{T}}, \tag{3.19}$$

where the calibration sequence requires estimation of $\mathbf{p}(1) = s_x$ for 150 [s] first, then $\mathbf{p}(2) = s_y$ for an additional 150 [s].

- Sensor B was equipped with $S = 8$ remodeling candidates defined by

$$\mathbf{h}_1^{[B]} = \mathbf{h}^{[B]} + \begin{bmatrix} c & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.20}$$

$$\mathbf{h}_2^{[B]} = \mathbf{h}^{[B]} + \begin{bmatrix} 0 & c & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.21}$$

$$\vdots \tag{3.22}$$

$$\mathbf{h}_8^{[B]} = \mathbf{h}^{[B]} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & c \end{bmatrix}, \tag{3.23}$$

where $\mathbf{h}^{[B]}$ is the baseline measurement model defined in (3.12), and the constant bias, c, has initial statistics given by

$$c_0 \sim \mathcal{N}\left(0\ [\mathrm{m}], 1000^2\ [\mathrm{m}^2]\right). \tag{3.24}$$

At the start of the scenario, only Sensor A and Sensor C were online in both vehicles. Additionally, both online sensors in Aircraft 2 were in monitoring mode. As shown in Figure 3.4, Sensor B was initialized after five minutes. However, the sensor developers were not confident in the sensor model from (3.12), and provided ARMAS an additional eight model versions, each modeling the addition of a constant bias to a particular satellite. Meanwhile, Aircraft 1 was limited to using (3.12) as given. The actual measurements from Sensor B included a 1500 [m] constant bias added to the fourth entry in $\mathbf{z}_k^{[B]}$ from (3.12). As shown in Figure 3.5, the sensor model provided for Sensor B was incomplete, causing the solution in Aircraft 1 to shift away from the truth. Meanwhile, Aircraft 2 used the validation mode in ARMAS to recognize the model mismatch without compromising its navigation solution. Since there were no calibration routines provided for Sensor B, it transitioned into remodeling mode, where all model options were evaluated in parallel, while continuing to navigate using the other two sensors. The model selection from the remodeling mode was then successfully validated, placing Sensor B into monitoring mode. Five minutes later, an aircraft maneuver changed the variable scale factor on Sensor A from $\mathbf{s} = [1 \quad 1]^{\mathrm{T}}$ to $\mathbf{s} = [1.2 \quad 1.3]^{\mathrm{T}}$. As shown in Figure 3.4, the change did not affect Aircraft 1 since the effect

Table 3.4: Key events and RSS position error comparison, Example 2.

			RSS Position Error [m]		
Sensor	**Event**	**Time [m]**	**Aircraft 1**	**Aircraft 2**	**% Change**
All	Start	0	0	0	0
B	Online	5	15.9	15.9	0
A	Scale factor	10	370.3	5.8	-98.4
B	Offline	20	438.9	2.5	-99.4
All	End	25	2.78×10^4	22.8	-99.9

on the navigation solution from Sensor A was attenuated by the measurement updates from the other two sensors. Conversely, ARMAS in Aircraft 2 detected the growing residuals and identified Sensor A as the source. As shown in Figure 3.6, Sensor A transitioned from monitoring to calibration mode, where the provided scale factor calibration sequence was augmented into the navigation state. The newly calibrated Sensor A then passed the validation test and was placed back into monitoring mode. Finally, ten minutes later, Sensor B was taken offline. At that point, with only Sensor A and Sensor C available to provide additional information, the solution in Aircraft 1 began to exhibit the effects of the un-calibrated Sensor A. Meanwhile, Aircraft 2 continued to operate nominally since it had previously detected the need for and successfully executed the calibration of Sensor A. Table 3.4 compares the solution performance between the two aircraft for this scenario. Note the diverging performance in terms of RSS position error between the two aircraft, especially after the un-calibrated Sensor A becomes the only source of position information in Aircraft 1.

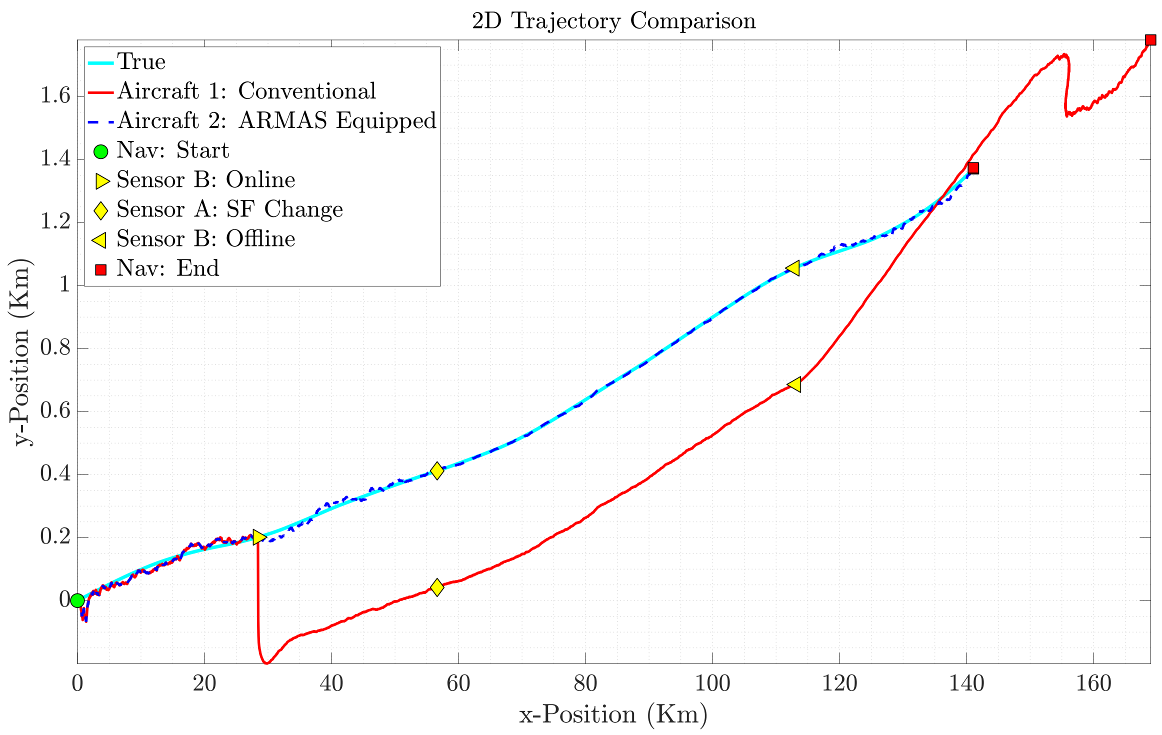


Figure 3.4: Trajectory comparison between Aircraft 1 and Aircraft 2, Example 2.

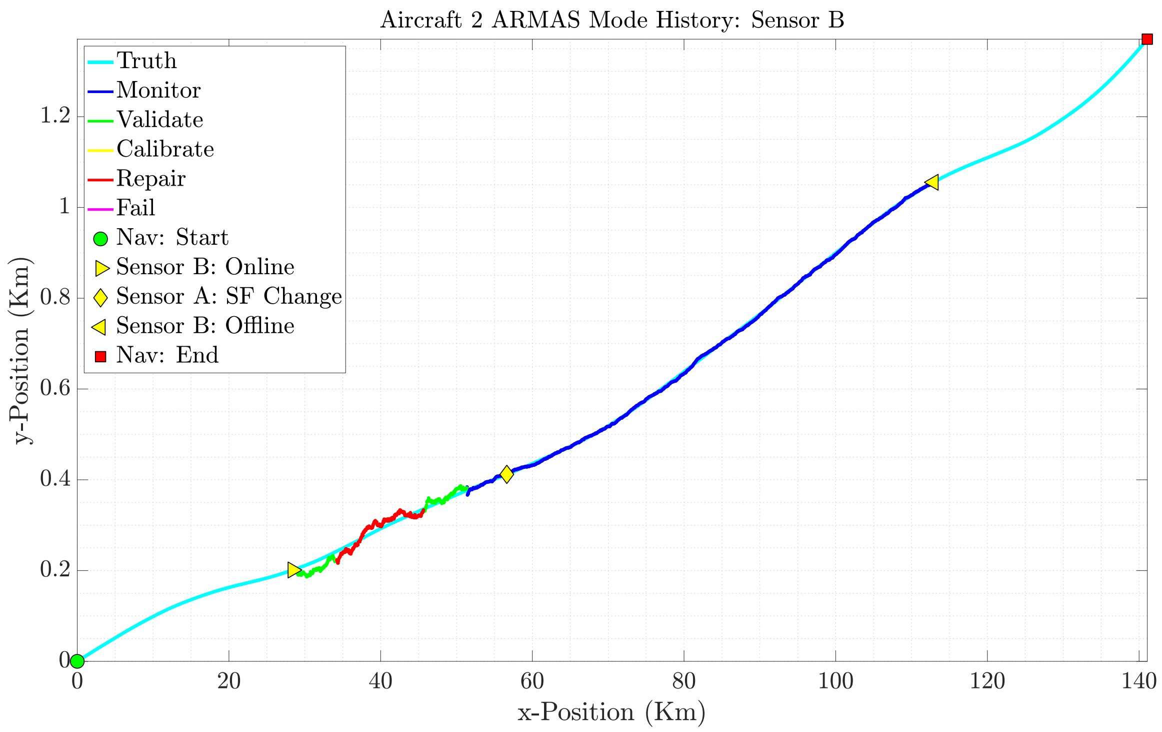


Figure 3.5: ARMAS mode history for Sensor B, Example 2.

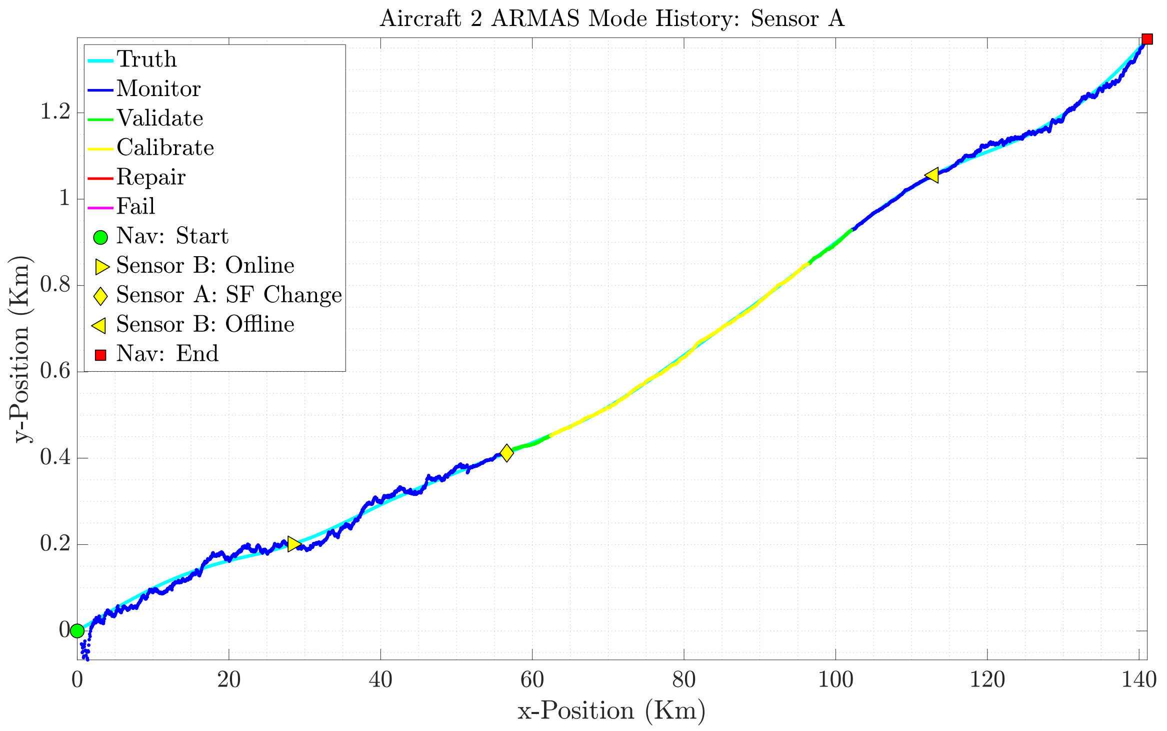


Figure 3.6: ARMAS mode history for Sensor A, Example 2.

3.4 Chapter Summary

This chapter has introduced a novel sensor management framework that provides sensor-agnostic autonomous and resilient sensor management for alternative multi-sensor navigation problems. The proposed framework, named Autonomous and Resilient Management of All-source Sensors (ARMAS), provides a breadth of sensor management functions across four modes of operation: monitoring, calibration, remodeling, and validation. Using a coherent interconnection between these modes, ARMAS was shown to provide resilient and autonomous sensor management across two example multi-sensor navigation scenarios that required a combination of fault detection and identification, online parameter calibration, multiple-model selection, and sensor model validation. In the two examples provided, a vehicle equipped with the ARMAS framework exhibited up to 99.9% less position RSS error during temporary sensor anomalies and multiple sequential sensor failures, when compared to a non-ARMAS equipped vehicle. Future work in this area includes continued development of the novel methods for multi-sensor fault detection and sensor model validation used in the monitoring and validation modes as well as multi-trial Monte Carlo performance analysis using actual and simulated multi-sensor navigation data.

IV. Sensor-Agnostic All-source Residual Monitoring

This chapter focuses on the monitoring objective from the overall ARMAS framework. As previously stated, all-source navigation has become increasingly relevant over the past decade with the rise in alternative sensor technologies. However, as the number and type of sensors informing a system increases, so does the probability of corrupting the system with sensor modeling errors, signal interference, and undetected faults. Though the latter of these has been extensively researched, the majority of existing approaches have constrained faults to single-sensor biases, and designed algorithms centered around the assumption of simultaneously redundant, synchronous sensors, none of which are guaranteed for all-source systems. This chapter develops a novel sensor-agnostic fault detection, exclusion, and integrity monitoring method that minimizes the assumptions on the fault type, all-source sensor composition, and the number of faulty sensors. The proposed method is validated against traditional fault detection techniques, and shown to adequately detect and isolate a variety of faults across several all-source configurations, without the need for scenario-based customization. At the time of this writing, the research developed in this chapter is in review for publication in [58].

4.1 Introduction

All-source navigation and Assured Position Navigation and Timing (APNT) have become increasingly important research areas over the past two decades, especially as alternative navigation sensor technologies (e.g., vision [97], radio [27], magnetic [22], etc.) have been matured and integrated into navigation systems [38]. However, each additional sensor allowed into a navigation system introduces another opportunity for corrupting the navigation solution with errors in sensor modeling, unexpected signal interference, or undetected sensor faults. Of these challenges, the latter has been extensively researched [11][12][13][15][16][20][21][36][66][70][86][93][108][109] as a multi-sensor fault detection problem where each satellite in the GPS constellation is regarded as a different (albeit identical in nature and synchronous) sensor in the multi-sensor system, and the "fault" is defined as an unmodeled bias that is assumed to only affect one of the sensors (satellites) at any given time. As shown in Chapter 3 and [54], our overall research motivation is to create a resilient sensor management system that provides APNT through the online detection and self-correction (i.e., auto-tuning) of sensor models that do not match observed measurements. In support of this overall effort, the specific developments presented in this chapter seek to determine when any of the above sources of corruption are present by detecting any general mismatches between a sensor's stated model (i.e., measurement function, function parameters, and error covariance matrix) and its observed measurements, where an unmodeled bias is simply one specific case of a mismatch. Additionally, our research shifts away from identical and synchronous sensors such as GPS satellites, and focuses on all-source multi-domain (e.g., position, velocity, etc.) and asynchronous sensors. In the following section, we discuss several classes of techniques generally used in GPS fault detection, and highlight the novel developments and adaptations we have made in order to achieve our research objectives.

4.2 Background

4.2.1 Basic Threshold Methods.

One of the most practical methods for detecting unmodeled biases is to place an alarm threshold on an observable quantity that has a predictable range of acceptable values. In [76], the value of the likelihood function for a set of Kalman filter [59] measurement residuals is tracked using a moving window. A fault is then declared if the value of the likelihood function falls below a set threshold, which can happen when either a bias is present or an incorrect measurement error covariance matrix is provided by the sensor model. Similarly, in [70], pseudorange and/or position estimates from each satellite are compared to their predicted values derived from the other satellites in view. In this case, a fault is declared if any of the differences exceeds a threshold defined by satellite geometry, which can happen if one of the satellites is biased. Though simple to implement, both of these types of methods are generally limited in that they are only able to detect a fault (in some cases just a bias), and unable to identify the culprit sensor in a multi-sensor system.

4.2.2 Least Squares Methods.

Another set of bias detection methods such as the parity vector [36][93] and least squares residuals [86] approaches rely on multiple GPS satellites being in view at any given time. In these methods, redundant pseudorange measurements from various satellites are used to form a linear least-squares projection matrix that, combined with assumptions on the distribution of the bias, defines the hypotheses of the fault-free and fault-present cases. Similar to previous methods, these methods also define the fault as a bias in one of the satellites. These so-called snapshot methods are not only effective at detecting faults and identifying the culprit sensor, but also lend themselves to statistically rigorous definitions for system integrity, or the guarantee of system performance under a specific set of assumptions. However, by definition, these methods rely on redundant measurements being available at every time step and often assume linear measurement models, which are good assumptions for GPS, but not the case for all-source multi-domain sensor applications. Additionally, much like the threshold methods, the fault is once again defined as the presence of a bias in a single satellite, which is only part of the sensor model, and excludes the possibility of a wrongly stated measurement error covariance matrix. Finally, although technically possible, none of these methods have been presently adapted to detect faults outside the position domain, as would be needed for velocity-based (or any other domain) all-source sensors.

4.2.3 Filtered Methods.

A third class of bias detection methods relies on testing the statistical distribution of quantities estimated by Kalman filters [18][59][76][77], which is practical given many navigation systems already employ such filters for producing the navigation solution. In [20] and [66], the fault detection test is developed statistically and geometrically, respectively, by analyzing the difference between the a-priori and a-posteriori horizontal position estimates, as well as their corresponding covariances. Meanwhile, in [15][16][21][108][109], a similar statistical test is formed by computing the difference in horizontal position estimates (and associated covariances) between a main filter that is informed by all sensors, and a series of parallel sub-filters, each excluding one sensor. These multi-filter methods are advantageous in that they are not only able to identify the culprit sensor but also produce a navigation solution that is theoretically free from faulty measurements under a single-fault assumption. However, similar to the previously mentioned methods, they have only been presently implemented to detect single-sensor biases in the position domain, which is not sufficient for our desired all-source applications. Additionally, these methods also require the computation of the cross-covariance terms for accurately estimating the expected covariance of the difference in position estimates between the main filter and each of the sub-filters. Finally, in [11][12][13][110], the test statistic is derived not from "solution separation" vectors, but rather from averaging a time sequence of residuals terms from the Kalman update equations. These filter-residual methods are useful in detecting insidious biases with varying growth rates depending on the averaging time used in the test statistic. However, much like the other methods described, they have only presently been implemented to detect GPS pseudorange biases, which is insufficient for all-source applications.

4.2.4 Contributions.

As shown in the previous section, a substantial amount of research has been developed in the area of GPS fault detection, identification, and exclusion, but even the state-of-the-art methods were found inadequate or incomplete to accomplish our research objectives without significant development and adaptation. As such, the method developed in this chapter, refered to henceforth as Sensor-Agnostic All-source Residual Monitoring (SAARM), provides a significant contribution to the state-of-the-art in that it:

- Does not constrain faults to only biases,
- Can easily be scaled for multiple simultaneous sensor faults,
- Detects faults (sensor model mismatches) in and across multiple domains,
- Does not require simultaneously redundant sensors to provide fault detection and identification,
- Provides fault exclusion without the need to compute cross-filter covariances, and
- Provides a robust measure of system integrity without constraining the fault type.

The remainder of this chapter is divided into three additional sections. Section 4.3 develops the necessary multi-filter multi-sensor notation, the residual test statistic, the fault detection and exclusion process, and the system integrity assumptions and guarantees. In Section 4.4, the detection performance of the proposed method is compared against existing snapshot methods in a simulated GPS navigation problem and several simulated all-source navigation problems. Finally, Section 4.5 summarizes the research contributions from this chapter and provides ideas for future work.

4.3 Methodology

4.3.1 Multi-Sensor Multi-Filter Notation.

This section expands the conventional Kalman filter [59] notation from [76][77] to include estimates from multiple filters as well as measurements from multiple non-identical sensors. The notation and underlying considerations will be crucial in the later development of the residual-space test statistic and the resulting fault identification process. Consider a (possibly) non-linear dynamic system of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left[\mathbf{x}(t), \mathbf{u}(t), t\right] + \mathbf{G}(t)\mathbf{w}(t), \tag{4.1}$$

where $\mathbf{x}$ is the $N \times 1$ navigation state vector containing the system states, $\mathbf{u}$ is the control input vector, $\mathbf{G}$ is an $N \times W$ linear operator, and $\mathbf{w}$ is a $W \times 1$ white Gaussian noise process with a $W \times W$ continuous process noise strength matrix $\mathbf{Q}$. Suppose the discretized [96] system states are estimated by J separate filters. Then at time $t = t_k$, the system state estimate vector and corresponding state estimation error covariance matrix from filters $j = 1, \ldots, J$ are given by $\hat{\mathbf{x}}^{[j]}(t_k)$ and $\mathbf{P}^{[j]}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}(t_k)$, respectively. Next, each of the J filters can be informed by any, all, or a subset of I sensors. At time t_k, the $i = 1, \ldots, I$ sensor provides (possibly) multidimensional $Z_i \times 1$ measurements of the form

$$\mathbf{z}^{[i]}(t_k) = \mathbf{h}^{[i]}\left[\mathbf{x}(t_k), \mathbf{u}(t_k), t_k\right] + \mathbf{v}^{[i]}(t_k), \tag{4.2}$$

where $\mathbf{h}^{[i]}$ is a (possibly) nonlinear measurement function, and $\mathbf{v}^{[i]}(t_k)$ is a $Z_i \times 1$ discrete white Gaussian noise process with covariance matrix $\mathbf{R}^{[i]}(t_k)$. Immediately prior to a measurement update, the estimated measurement for sensor i from filter j, $\hat{\mathbf{z}}^{[i,j]}$, is generated using

$$\hat{\mathbf{z}}^{[i,j]}(t_k^-) = \mathbf{h}^{[i]}\left[\hat{\mathbf{x}}^{[j]}(t_k^-), \mathbf{u}(t_k), t_k\right], \tag{4.3}$$

while its estimated covariance matrix, $\mathbf{P}^{[i,j]}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}(t_k^-)$, is generated based on the type of filtering algorithm. For example, in a linearized filter (such as an Extended Kalman Filter) it can be computed using

$$\mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}^{[i,j]} = \mathbf{H}^{[i]}\mathbf{P}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}^{[j]}\mathbf{H}^{[i]^{\mathrm{T}}}, \tag{4.4}$$

where the time index (t_k^-) is omitted for simplicity, and $\mathbf{H}^{[i]}$ represents the Jacobian of $\mathbf{h}^{[i]}$ about the point $\hat{\mathbf{x}}^{[j]}(t_k^-)$. For information on generating $\mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}$ in an Unscented Kalman Filter (UKF), the reader is referred to [101]. Finally, the (so-called) pre-update residual vector computed between sensor i and filter j, $\mathbf{r}^{[i,j]}$, and associated covariance matrix, $\mathbf{P}_{\mathbf{rr}}^{[i,j]}$, is given by

$$\mathbf{r}^{[i,j]}(t_k) = \mathbf{z}^{[i]}(t_k) - \hat{\mathbf{z}}^{[i,j]}(t_k^-) \tag{4.5}$$

$$\mathbf{P}_{\mathbf{rr}}^{[i,j]}(t_k) = \mathbf{R}^{[i]}(t_k) + \mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}^{[i,j]}(t_k^-). \tag{4.6}$$

4.3.2 Fault Detection Test Statistic.

Having derived the residual vector, $\mathbf{r}^{[i,j]}(t_k)$, and its associated covariance matrix, $\mathbf{P}_{rr}^{[i,j]}(t_k)$, in (4.5) and (4.6), we now define a residual-space test statistic to determine if a set of observed residuals between a specific sensor-filter pair are adhering to their expected distribution. Since our goal is to limit the assumptions on the type of fault (i.e., the fault could be a bias, an incorrectly stated noise covariance matrix, or incorrect calibration of measurement function parameters), we did not model two competing distributions as would be needed to employ a LRT [63]. Instead, we focused on the single likelihood function of the residuals, based on measuring their Mahalanobis distances [28] due to their simplicity and "standardizing" properties, which were found useful in time-changing processes such as navigation.

Given a Z_i-dimensional Gaussian distribution with mean $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma}$, the squared Mahalanobis distance, d^2, between an observation $\mathbf{y}$, and the centroid of the distribution is then given by

$$d^2 = (\mathbf{y} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \boldsymbol{\mu}). \tag{4.7}$$

Additionally, d^2 is known [23][28] to follow a Chi-Square distribution with Z_i degrees of freedom. Moreover, the sum of M independent d^2 distances is also known to follow a Chi-Square distribution with $M \times Z_i$ degrees of freedom. Since Kalman filter pre-update residuals are assumed to be a zero-mean white sequence [76], we let $\mathbf{y} = \mathbf{r}^{[i,j]}(t_k)$ from (4.5), $\mathbf{\Sigma} = \mathbf{P}_{\mathbf{rr}}^{[i,j]}(t_k)$ from (4.6), and $\boldsymbol{\mu} = \mathbf{0}$. Subsequently, we can develop a fault detection test, given set of M, Z_i-dimensional residual vectors collected between $t = t_k$ and $t = t_{k+M}$ using

$$\begin{aligned} H_0 : \chi^*_{[i,j]} &> \chi^2(\alpha/2, M \times Z_i) \text{ and} \qquad (4.8) \\ \chi^*_{[i,j]} &< \chi^2(1-\alpha/2, M \times Z_i) \\ H_1 : \chi^*_{[i,j]} &< \chi^2(\alpha/2, M \times Z_i) \text{ or} \qquad (4.9) \\ \chi^*_{[i,j]} &> \chi^2(1-\alpha/2, M \times Z_i), \end{aligned}$$

where M defines the number of averaging samples in the test, α is the significance level of the test (i.e., probability of false alarm, P_{FA}), H_0 is defined as a a fault *not* present in filter j, H_1 is defined as a fault present in filter j, and

$$\chi^*_{[i,j]} = \sum_{s=k}^{k+M} d^2_{[i,j]}(t_s), \qquad (4.10)$$

$$d^2_{[i,j]}(t_k) = \mathbf{r}^{[i,j]^{\mathrm{T}}}(t_k) \left[\mathbf{P}_{rr}^{[i,j]}(t_k) \right]^{-1} \mathbf{r}^{[i,j]}(t_k). \qquad (4.11)$$

It is important to note a few key points about the above test statistic. First, it is designed to detect mismatches (in any domain such as position, velocity, etc.) between sensor measurements and their stated models, in both the upper and lower ends of the resulting Chi-Square distribution. This was purposely done so that our method could not only detect unlikely large or variable residuals resulting from biases, problems in the measurement function, its parameters, or under-stating the measurement error covariance matrix, but additionally, unlikely small residuals resulting from over-stating the measurement error covariance matrix. Next, similarly to [94], we experimentally found the thresholds in (4.8) and (4.9) needed to be adjusted (by approximately ±5% during our simulations) to account for differences between their theoretical and empirical values (such as linearization errors) in order to mitigate unnecessary false alarms. Finally, the test can only determine if *any* of the I sensors providing measurement updates to filter j has a fault. In order to determine which sensor(s) within filter j are faulty, additional assumptions and computations must be made, as shown in the next section.

4.3.3 Fault Identification Process.

Up to this point, we've defined how a time-sequence of residual vectors from a specific sensor-filter combination may be tested for likelihood, without making assumptions on the domain of the sensor measurement, or the type of fault. Here it is important to emphasize that a H_1 result derived from a set of residual vectors from a particular sensor-filter pair (i, j) does not imply that sensor i is faulty. It is only an indication that one of the sensors informing filter j is faulty. In other words, low-likelihood residuals can then either be caused by faulty measurements, $\mathbf{z}^{[i]}$, or faulty estimated measurements, $\hat{\mathbf{z}}^{[i,j]}$, the latter of which is influenced by all sensors informing filter j whose state-space overlaps with sensor i. To solve this challenge, we developed a "fault consensus" process that associates the presence of a sensor with the presence of a fault in order to determine the most probable sensor associated with H_1 (faulty) results. Though the proposed method is not limited to just single sensor faults, it is best to begin our discussion with this case before scaling to the generalized multiple simultaneous fault cases. The next two sections develop the single and multiple fault cases, respectively.

4.3.3.1 Single Serial Faults.

As described in Section 4.2, a commonly assumed fault scenario is a single sensor fault per testing epoch (i.e., during a single M-sample test window in our case). Multiple faults are still considered, but restricted to occur serially. In this case, we set up our fault identification process by creating $J = I$ filters, each informed by a unique set of

$I - 1$ sensors. In other words, each filter excludes one of the I sensors. Here, it is important to note two points. First, since we expect all-source sensors to be non-identical, some states may become unobservable within a particular filter if the only sensor that has observability over them is excluded from that filter. To prevent potential numerical issues with the covariance of unobservable states, we can perform a stochastic observability test [7] on each filter and/or design the set of sensors such that all states are observable in all filters. Second, as with other parallel-filter methods, a "main filter" that is informed by all sensors is also created, but in this method we do not use its information for "solution separation" comparisons, thereby eliminating the need for computing the cross-covariance terms between it and all other filters. Having designed the set of filters using this method guarantees, *under the assumption that, at most, one sensor can fail simultaneously*, at least one of the filters will be completely unaffected by faulty measurements. As shown below, we can then use this axiom in conjunction with the full set of (i, j) residual test results to determine the culprit sensor.

We begin the fault identification process by populating the $I \times J$ (which becomes $I \times I$ in this single-fault case) test results matrix, $\mathbf{T}$, using

$$\mathbf{T}(i,j) = \begin{cases} 0, & \text{Sensor } i \text{ does not inform filter } j, \\ 0, & \chi^*_{[i,j]} \text{ yields } H_0 \text{ (no fault detected)}, \\ 1, & \chi^*_{[i,j]} \text{ yields } H_1 \text{ (fault detected)}. \end{cases} \tag{4.12}$$

Figure 4.1 illustrates the information from each sensor-filter pair needed to populate $\mathbf{T}$ in the case where the jth filter excludes the jth sensor. In the figure, each of the $i = 1, \ldots, I$ rows corresponds to the measurement, $\mathbf{z}^{[i]}$, and its associated error covariance matrix, $\mathbf{R}^{[i]}$, obtained from the ith sensor. These two parameters define the modeled distribution of the sensor measurement, and make up the first half of (4.5) and (4.6), respectively. Next, each of the $j = 1, \ldots, J$ ($J = I$ in this single fault case) columns corresponds to the estimated measurement, $\hat{\mathbf{z}}^{[i,j]}$, and its associated error covariance matrix, $\mathbf{P}^{[i,j]}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}$. These two parameters make up the remainder of (4.5) and (4.6) and define the modeled distribution of the estimated sensor measurement. As shown in the figure, these last two parameters are influenced by all sensors informing the filter in the jth column, which corresponds to all sensors except the jth sensor.

Once populated, a fault is declared when $\mathbf{T}$ contains any non-zero entries. This means a fault is declared when any of the $I^2 - I$ residual test results from (4.8) and (4.9) that are contained in $\mathbf{T}$ result in H_1, which theoretically increases the probability of false alarm up to a maximum of $P_{FA} \leq \alpha(I^2 - I)$ for the single sensor fault assumption. If a fault is declared, the culprit sensor may be identified if a consensus is reached. That is, since each sensor is excluded from one filter, we can identify a faulty sensor if only a single filter (presumably the filter that excluded it) remains fault free. Mathematically, we first compute the fault scores vector, $\mathbf{s}$, whose dimension is equal to the number of filters, J, using

$$\mathbf{s}(j) = \sum_{i=1}^{I} \mathbf{T}(i, j), \tag{4.13}$$

which produces a sum across the rows (sensors) for each column (filter) in $\mathbf{T}$. Once computed, we have four possible scenarios:

1. If $\mathbf{s}$ contains all zeros, then no fault has been detected,

2. If $\mathbf{s}$ contains at least one *non-zero* entry, but more than one *zero* entry, then a fault is declared and the culprit is not identified,

3. If $\mathbf{s}(j)$ is the only *zero* entry in $\mathbf{s}$, then a fault is declared and the culprit sensor is the sensor that was excluded from the j^{th} column in $\mathbf{T}$, or the j^{th} filter, if constructed according to Figure 4.1.

4. Finally, if $\mathbf{s}$ contains no *zero* entries, then more than one sensor is faulty, and the assumptions of the test have been violated.

Each of these "states" can be used in conjunction with the system integrity computations presented in Section 4.3.4 in order to continuously inform users of their APNT protection status. Depending on the type and dynamics of the fault, as well as the set and type of sensors in the system, the results in T may continue to change during every epoch and eventually lead to a culprit. If and when a culprit is determined, the corresponding fault-free filter is used as the new main filter, and a new set of $I - 1$ filters is initialized using its states. The process can then be repeated sequentially for multiple serial faults with the assumption that a second fault does not occur during the first M samples after having re-spawned the filter set, which will be addressed in the next section.

4.3.3.2 Simultaneous Faults.

The serial-fault methodology described above can be easily scaled to enable detection of a secondary fault occurring during the first M samples after an initial fault, as well as multiple simultaneous faults. To do so, we first re-define the number of filters required, J, the structure of the associated test results matrix, $\mathbf{T}$, and the dimension of the faults score vector, $\mathbf{s}$, as a functions of the assumed maximum number of simultaneous faults, which we define as a "layer." In general, the number of additional filters required, J, for each layer, N, is given by

$$J_N = \begin{pmatrix} I \\ I - N \end{pmatrix} = \frac{I!}{N!(I - N)!}. \tag{4.14}$$

As shown in Section 4.3.3.1, in layer one, we assumed $N = 1$ simultaneous fault was possible, and created

$$J_1 = \begin{pmatrix} I \\ I - 1 \end{pmatrix} = \frac{I!}{1!(I - 1)!} = I \tag{4.15}$$

filters each excluding one sensor, which were then used to populate $\mathbf{T}_1 \in \mathbb{R}^{I \times J_1}$, detect the fault, and identify the single culprit using $\mathbf{s}_1$. If we now assume $N = 2$ simultaneous faults

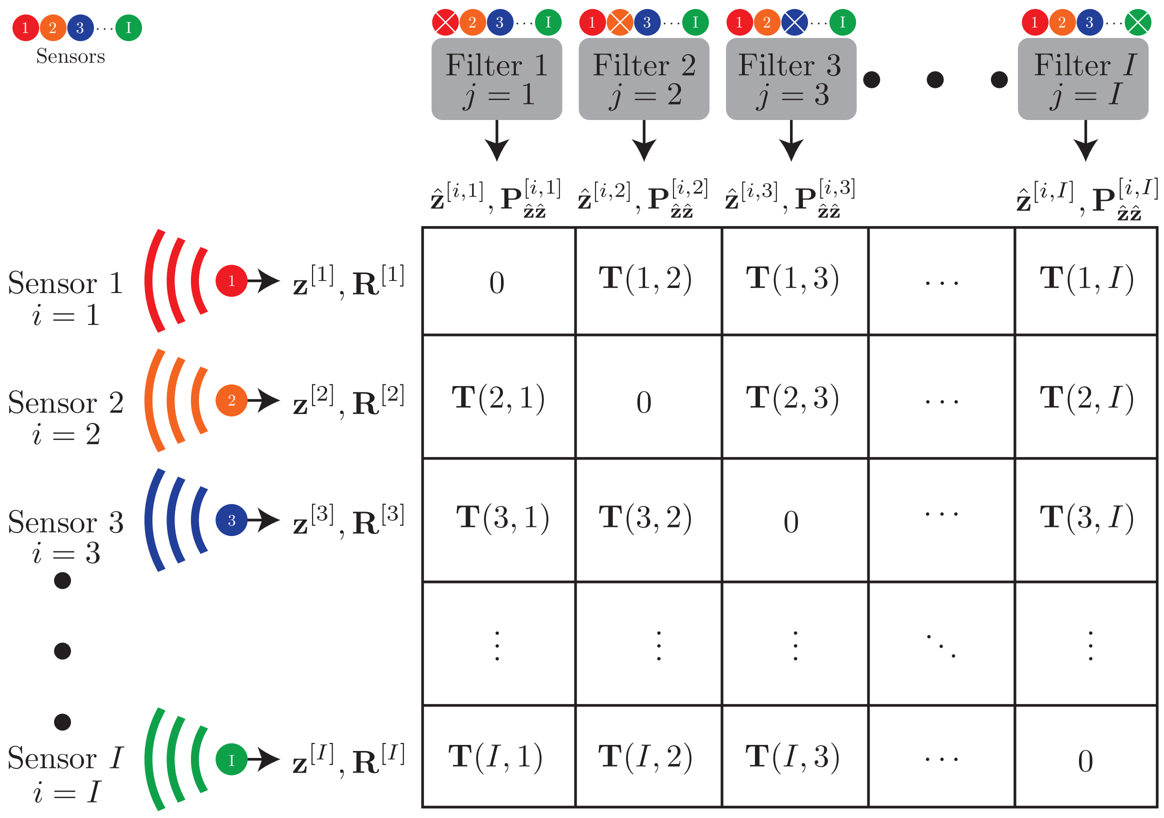

Sensors
Filter 1 $j = 1$
Filter 2 $j = 2$
Filter 3 $j = 3$
Filter I $j = I$
$\hat{\mathbf{z}}^{[i,1]}, \mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}^{[i,1]}$
$\hat{\mathbf{z}}^{[i,2]}, \mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}^{[i,2]}$
$\hat{\mathbf{z}}^{[i,3]}, \mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}^{[i,3]}$
$\hat{\mathbf{z}}^{[i,I]}, \mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}^{[i,I]}$
Sensor 1 $i = 1$
$\mathbf{z}^{[1]}, \mathbf{R}^{[1]}$
Sensor 2 $i = 2$
$\mathbf{z}^{[2]}, \mathbf{R}^{[2]}$
Sensor 3 $i = 3$
$\mathbf{z}^{[3]}, \mathbf{R}^{[3]}$
Sensor I $i = I$
$\mathbf{z}^{[I]}, \mathbf{R}^{[I]}$
0 $\mathbf{T}(1,2)$ $\mathbf{T}(1,3)$ $\cdots$ $\mathbf{T}(1,I)$
$\mathbf{T}(2,1)$ 0 $\mathbf{T}(2,3)$ $\cdots$ $\mathbf{T}(2,I)$
$\mathbf{T}(3,1)$ $\mathbf{T}(3,2)$ 0 $\cdots$ $\mathbf{T}(3,I)$
$\vdots$ $\vdots$ $\vdots$ $\ddots$ $\vdots$
$\mathbf{T}(I,1)$ $\mathbf{T}(I,2)$ $\mathbf{T}(I,3)$ $\cdots$ 0

Figure 4.1: Illustration of the multi-sensor multi-filter test statistic matrix, $\mathbf{T}$.

are possible, we require an additional

$$J_2 = \binom{I}{I-2} = \frac{I!}{2!(I-2)!} = \frac{I^2 - I}{2} \tag{4.16}$$

filters each excluding two sensors, which are then used to populate $\mathbf{T}_2 \in \mathbb{R}^{I \times J_2}$. Using this two-layer configuration, the culprit sensor in a single fault scenario can continue to be identified as previously described, using $\mathbf{T}_1$ and $\mathbf{s}_1$. In the case of a simultaneous fault, $\mathbf{s}_1$ indicates the single fault assumption has been violated (no non-zero entries remain), which prompts the system to use $\mathbf{T}_2$ and $\mathbf{s}_2$ to identify the two culprits. In the case of a secondary fault during the first M samples after an initial fault, the the subset of filters in the J_2 layer that excluded the first culprit corresponds exactly to the new J_1 layer of filters needed after re-spawning, which enables uninterrupted fault detection.

For example, consider a system with $I = 5$ sensors where up to two simultaneous sensor faults are assumed. The first layer consists of $J_1 = 5$ filters, and each filter excludes one of the sensors, as shown in Table 4.1. Using (4.16), the second layer consists of $J_2 = 10$ filters, and each filter excludes two of the sensors, as shown in Table 4.2. Suppose Sensor 3 experiences a fault. In this case, Filter 3 (shaded gray in Table 4.1) is uncorrupted by any faulty measurements, and its corresponding column in $\mathbf{T}_1 \in \mathbb{R}^{5\times 5}$ uniquely contains all zeros. After determining Sensor 3 is the culprit via (4.13), Sensor 3 is taken offline and a new set of $J_1 = I - 1 = 4$ filters, each excluding one of the remaining four sensors, is spawned. Without a J_2 layer, this would mean the system could not detect a subsequent fault while it repopulates the new $\mathbf{T}_1 \in \mathbb{R}^{4\times 4}$. However, having the J_2 layer already running, we can see the new J_1 layer of filters is actually equivalent to the subset of J_2 filters that had also excluded Sensor 3 (shaded gray in Table 4.2), which guarantees uninterrupted fault detection after detecting an initial fault. Finally, suppose both Sensor 3 and Sensor 5 experience a simultaneous fault. In this case, every single filter in the J_1 layer (i.e., every column in Table 4.1) would be corrupted and no column in $\mathbf{T}_1$ would contain all zeros. However, the J_2 filter that excluded both Sensor 3 and Sensor 5 (shaded dark

Table 4.1: Sensor-filter configuration for layer J_1, $I = 5$ sensors

	Included in filter				
Sensor	1	2	3[a]	4	5
1		●	●	●	●
2	●		●	●	●
3	●	●		●	●
4	●	●	●		●
5	●	●	●	●	

[a]This filter is uncorrupted by faulty Sensor 3 measurements

gray in Table 4.2) would be guaranteed to be uncorrupted by faulty measurements, and its corresponding column in $\mathbf{T}_2 \in \mathbb{R}^{5\times 10}$ would uniquely contain all zeros. In principle, this process can be scaled up to any number of layers, corresponding to any number of simultaneous faults. It is important, however, to consider the trade-off in computational power required to support the growing number of required filters for each additional layer.

4.3.4 Integrity Assumptions and Guarantees.

Having defined an all-source fault detection, identification, and exclusion process, we now turn our attention to the resulting system integrity computation. In general, we define system integrity as a guarantee of system performance under a particular set of assumptions. This definition fits the established methods in the previously mentioned GPS integrity methods. Namely, the guarantees of performance for GPS-based methods are commonly given as a Horizontal Protection Level (HPL) or Horizontal Integrity Limit (HIL), which are essentially constant-probability ellipses in the horizontal plane that are guaranteed to contain the true position while meeting the assumptions of the fault

Table 4.2: Sensor-filter configuration for layer J_2, $I = 5$ sensors

	Included in filter									
Sensor	1	2[a]	3	4	5[a]	6	7	8[a]	9[b]	10
1					●	●	●	●	●	●
2		●	●	●				●	●	●
3	●		●	●		●	●			●
4	●	●		●	●		●		●	
5	●	●	●		●	●		●		

[a]These filters are uncorrupted by faulty Sensor 3 measurements

[b]This filter is uncorrupted by faulty Sensor 3 and Sensor 5 measurements

detection test (e.g., bias distribution, single-sensor fault, probability of missed detection, etc.). Similarly in SAARM, we aim to provide an HPL guaranteed to contain the true position under the assumptions of our method. Our lack of assumptions in terms of the type, number, and domain of the fault(s), or the type and number of sensors, arguably complicated our fault detection process. At the same time however, it allows us to simplify our fault identification assumption into a single axiom: *assuming at least one of the filters is informed entirely by properly modeled, uncorrupted sensors, then at least one filter contains consistent state estimation error statistics*. In other words, since it is assumed that one of the filters is fault-free (based on properly designing the set of filters), then the estimated error statistics from one of the filters truly describe actual errors committed by the filter. Defining α_I as the acceptable error bound, we can then derive an accurate $100(1-\alpha_I)\%$ error ellipse on the horizontal position using the uncorrupted filter's horizontal position estimate and its associated error covariance matrix. Given the uncorrupted filter is not identifiable prior to determining a culprit, we simply union the $100(1-\alpha_I)\%$ horizontal position error ellipses from all of the filters, thereby guaranteeing the true horizontal position is contained within the union with *at least* a $100(1 - \alpha_I)\%$ probability, since the union can only grow the resulting ellipse. This guarantee is valid regardless of the status of the underlying fault detection and identification process.

To illustrate our HPL, consider a 2D navigation problem using $I = 5$ sensors. The sensor suite is composed of three 2D position sensors: POS1, POS2, POS3, and two 2D velocity sensors: VEL1, VEL2. The system dynamics are propagated using a 2D kinematics model driven by 2D FOGM acceleration. In order to best visualize the effects of faults on HPL, the fault has been defined as a growing bias starting from 0 [m] at $t_k = 60$ [s], growing at a rate of 2 [m/s], and applied to the x-dimension measurements from the POS2 sensor. Figures 4.2 through 4.5 illustrate a time sequence of events along a sample instantiation of the simulation. As shown in the figures, an error bound of $\alpha_I = 0.05$ was used to produce 95% error ellipses. The HPL derived from the union of the 95% horizontal position error ellipses from all filters is guaranteed to contain the true location at least 95% of the time, regardless of the presence of a fault, ability to detect, or ability to determine a culprit. In all examples, the main filter is informed by all sensors but its states and their error covariances are not used in the detection of faults or the computation of HPL. Finally, though these sample illustrations were limited to a single fault and two dimensions, the underlying axiom and assumptions are still valid for multiple faults and the error ellipses can be scaled to 3D error ellipsoids, if desired.

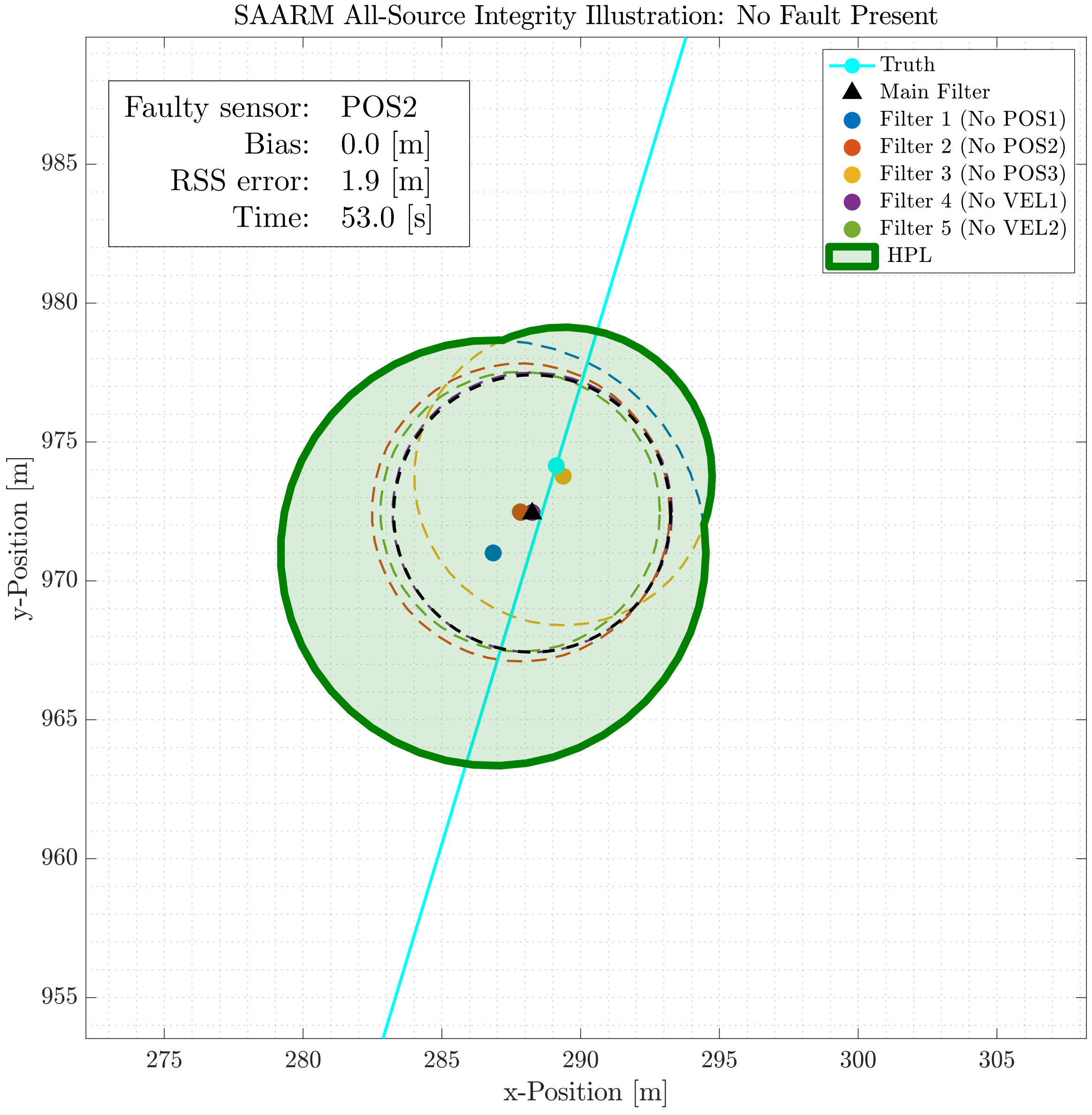


Figure 4.2: Example SAARM HPL: No fault present. In this example, there is no fault induced into any of the five sensors, and no fault has been detected (all entries in **s** are zero), which is shown to the user as a green HPL. All filters are uncorrupted. The HPL is comprised of the union of the 95% position error ellipses from all filters, and contains the true position at least 95% of the time.

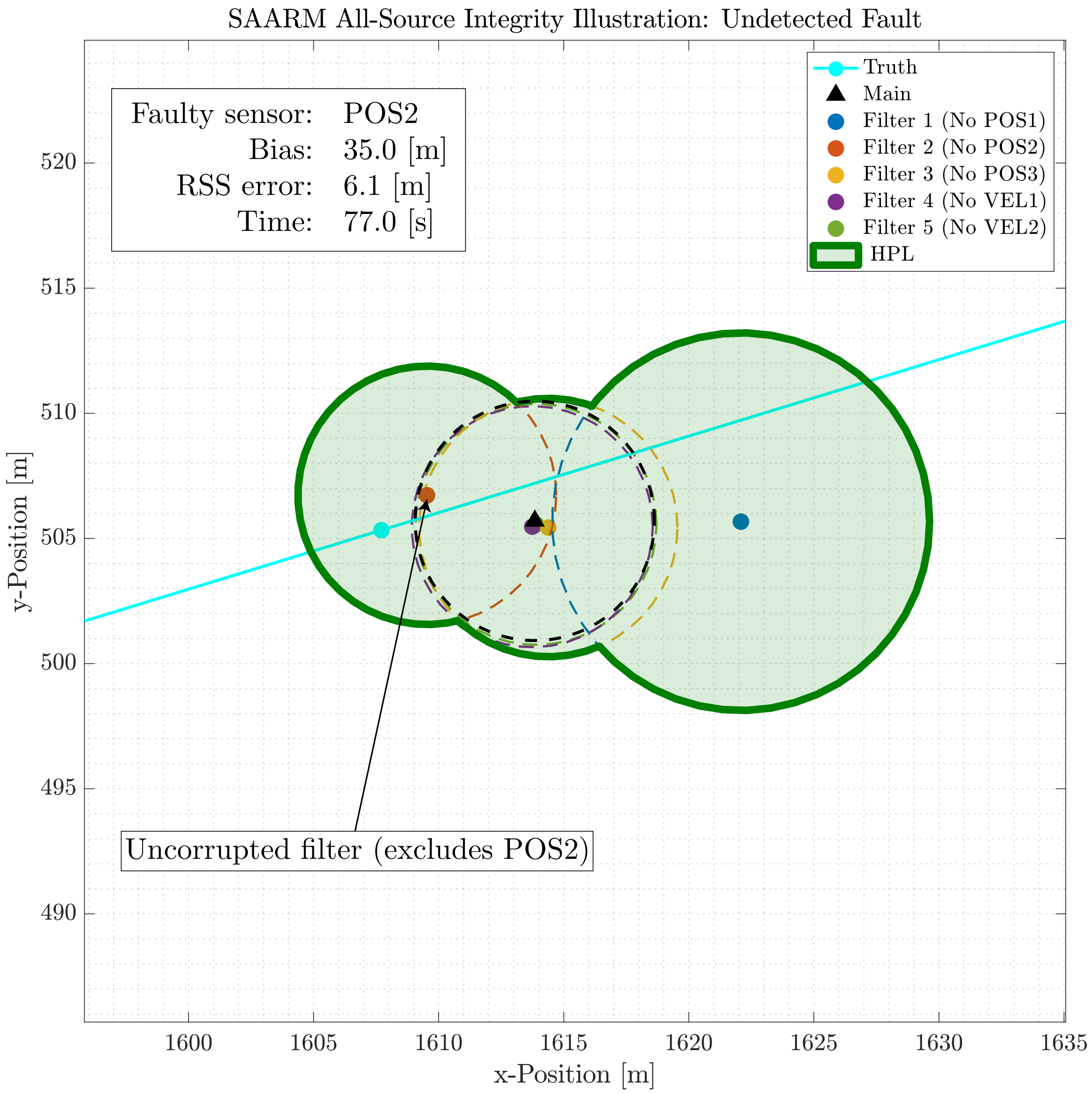


Figure 4.3: Example SAARM HPL: Undetected fault. In this example, a 30 [m] bias is affecting the POS2 sensor, but no fault has been detected yet (all entries in **s** are still zero), which is shown to the user as a green HPL. All filters except Filter 2 are corrupted. The HPL is comprised of the union of the 95% position error ellipses from all filters, and it is guaranteed to contain the true position at least 95% of the time since one of the filters is guaranteed to be uncorrupted.

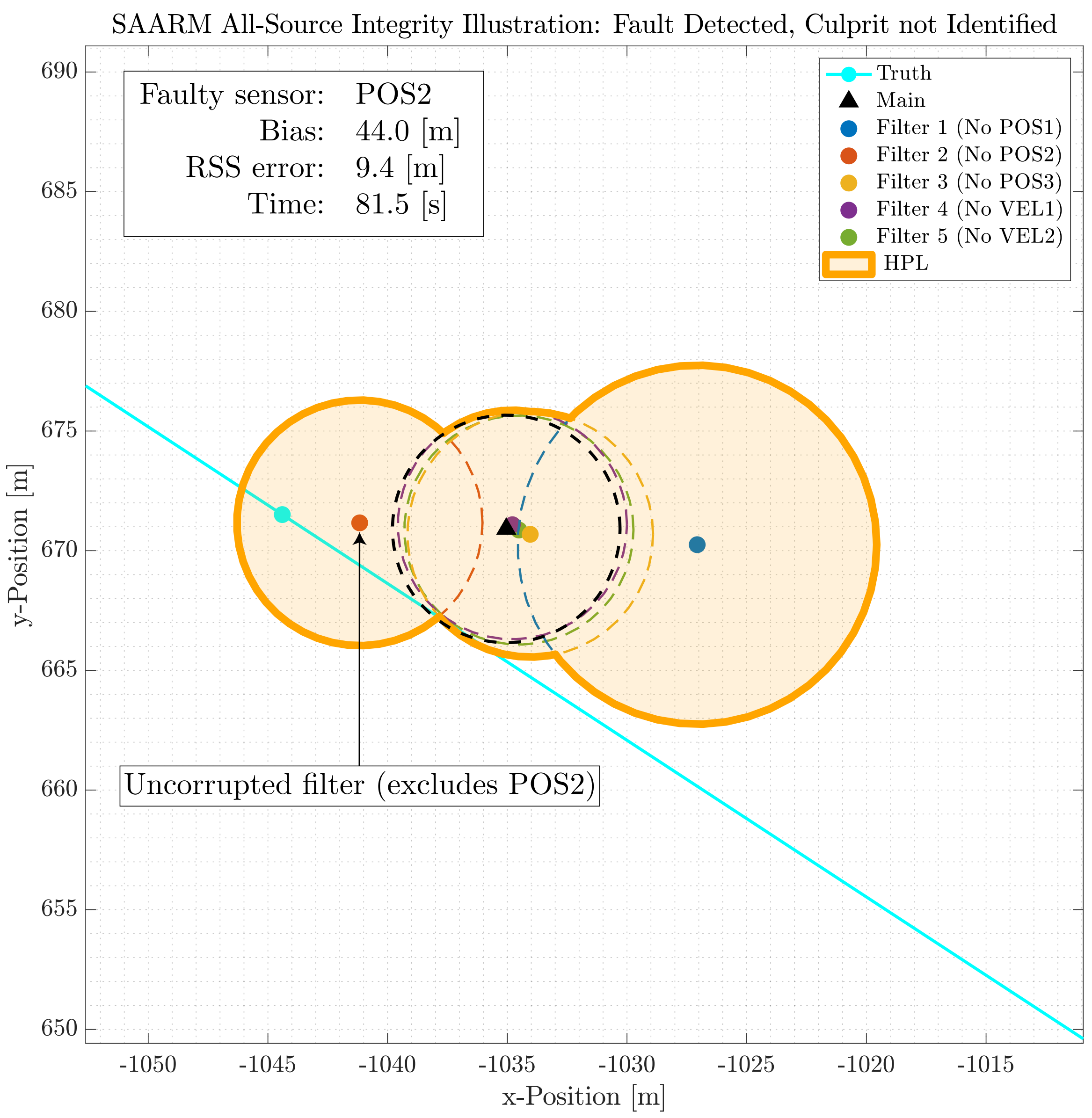


Figure 4.4: Example SAARM HPL: Unidentified culprit. In this example, a 44 [m] bias is affecting the POS2 sensor, and a fault has been detected (at least one entry in **s** is non-zero), but no culprit has been identified (there is more than one zero entry in **s**), which is shown to the user as an orange HPL. All filters except Filter 2 are corrupted. The HPL is comprised of the union of the 95% position error ellipses from all filters, and it is guaranteed to contain the true position at least 95% of the time since one of the filters is guaranteed to be uncorrupted.

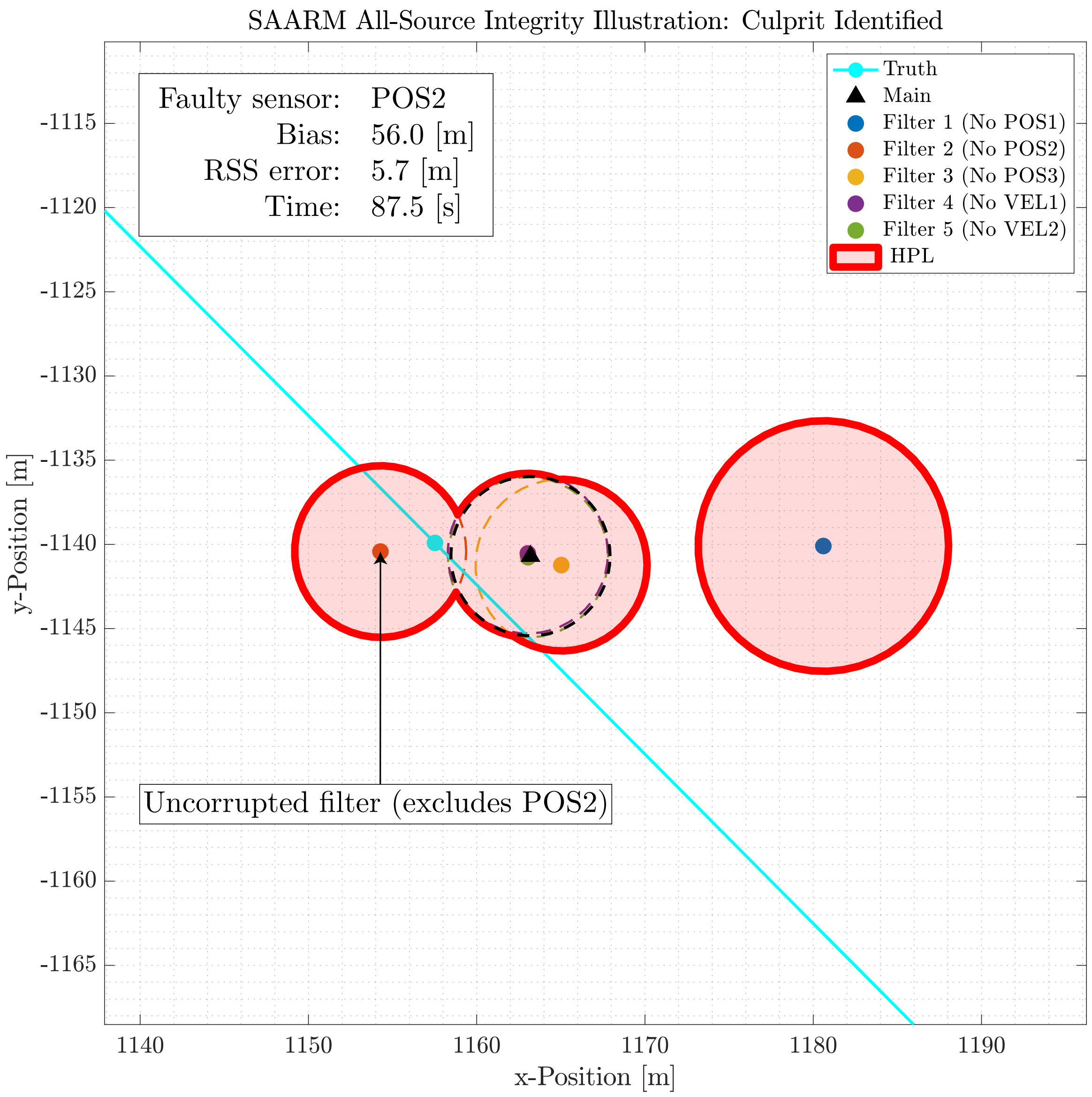


Figure 4.5: Example SAARM HPL: Culprit identified. In this example, a 56 [m] bias is affecting the POS2 sensor, a fault has been detected and the culprit has been identified (there is a single zero-entry in **s**), which is shown to the user as a red HPL. All filters except Filter 2 are corrupted. The HPL is comprised of the union of the 95% position error ellipses from all filters, and it is guaranteed to contain the true position at least 95% of the time since one of the filters is guaranteed to be uncorrupted. Immediately after this time step, the POS2 sensor is taken offline, and a new set of filters is re-spawned from Filter 2.

4.4 Simulation Results

The proposed method was evaluated using Monte Carlo simulations across a variety of navigation problems. For all simulations, the true system dynamics were driven by a 2D kinematic model given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{x}}_p(t) \\ \dot{\mathbf{x}}_v(t) \\ \dot{\mathbf{x}}_a(t) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_v(t) \\ \mathbf{x}_a(t) \\ -\frac{1}{\tau_a}\mathbf{x}_a(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{w}(t) \end{bmatrix}, \tag{4.17}$$

where $\mathbf{x}_p$ is the vehicle's 2D position in [m], $\mathbf{x}_v$ is the 2D velocity in [m/s], $\mathbf{x}_a$ is the 2D acceleration in [m/s^2] and driven by a FOGM process with time constant $\tau_a = 90$ [s], and $\mathbf{w}(t)$ is a 2D white Gaussian noise process with $E\left[\mathbf{w}(t)\mathbf{w}(t+\tau)^{\mathrm{T}}\right] = \mathbf{Q}\delta(\tau)$ and

$$\mathbf{Q} = (1.5 \times 10^{-3})^2 \underset{2\times 2}{\mathbf{I}} \text{ [m}^2/\text{s}^5]. \tag{4.18}$$

For each level or fault type in each simulation, 3000 trials were conducted. For each trial in each simulation, the initial true position and initial position state estimates were set to zero. The initial true velocity was randomly drawn from a $\mathcal{N}\left(0, 10^2\right)$ [m^2/s^2] distribution for both the x and y axes, while the initial velocity state estimates were set to zero. The initial true acceleration and acceleration state estimates were set to zero. The initial state estimation error covariance was set to 10^2 [m^2] in position, 10^2 [m^2/s^2] in velocity, and $(1.5 \times 10^{-3})^2$ [m^2/s^4] in acceleration. Each trial was propagated using $\Delta t_k = 0.5$ [s], starting at $t_k = 0$ [s] with a prescribed fault occurring at $t_k = 22$ [s], and ending at $t_k = 60$ [s]. Across all simulations, the SAARM test significance level was set to $\alpha = 1/15000 = 6.67 \times 10^{-5}$, while the test epoch was set to $M = 40$ samples, which was equivalent to 20 [s].

4.4.1 RAIM Comparisons.

In the first set of simulations, SAARM was applied to the common pseudorange fault detection and identification problem from GPS, and compared to the results from the least-squares Receiver Autonomous Integrity Monitoring (RAIM) method developed in [86], using $r_D = 9.5$ [m] for detection, and $r_I = 11$ [m] for isolation. The RAIM detection and isolation thresholds values were chosen such that the algorithm would generally match the performance shown in [86, Table 4] for a 25 [m] bias. For these simulations, $I = 8$ 2D pseudorange sensors were used, each with a nonlinear measurement model given by

$$\mathbf{z}_k^{[i]} = \left\|\mathbf{t}_i - \mathbf{x}_{p_k}\right\| + b_k + \mathbf{v}_k^{[i]}, \tag{4.19}$$

$$\mathbf{v}_k^{[i]} \sim \mathcal{N}\left(0, 4^2\right) \text{ [m}^2\text{]}, \tag{4.20}$$

where $\mathbf{t}_i$ is the 2D position of the i^{th} satellite, $\mathbf{x}_{p_k}$ is the 2D position of the vehicle at time t_k, and b_k is a FOGM process simulating a receiver clock error with time constant $\tau_b = 3600$ [s] and $\sigma^2 = 8000^2$ [m^2]. For each trial, the true initial clock error was drawn from a $\mathcal{N}\left(0, 8000^2\right)$ [m^2] distribution, while its corresponding initial state estimate was set to zero.

The first four rows in Table 4.3 summarize the results across varying bias levels from 10 [m] up to 100 [m]. In the last row, the fault was defined by a sudden change in observed pseudorange measurement noise (i.e., 10x scaling of $\mathbf{R}$). The columns of Table 4.3 summarize the probability (between 0 and 100%) of false alarm, missed detection, fault detection and isolation, and fault detection without isolation, respectively. As intended, RAIM performance matched its expected performance for a 25 [m] bias since its thresholds were tuned as such. As shown, SAARM generally outperformed RAIM for biases under 50 [m], and performed just as well for larger biases. Here, it is important to note the objective of the simulations in Table 4.3 was not to show how SAARM would perform against RAIM in a real GPS environment, but rather demonstrate that it performs no worse than an established RAIM method, given a simple bias fault, and using a reasonable set of significance and threshold levels. It is important, however, to point out that SAARM clearly outperformed RAIM in detecting a fault other than a simple bias. And although different r_D and r_I thresholds could be derived for RAIM for such cases, the generality

Table 4.3: Fault detection and identification comparison, simulated pseudoranges

	Probability (%)							
	False alarm		**Missed**		**Isolated**		**Detect only**	
Fault	SAARM	RAIM	SAARM	RAIM	SAARM	RAIM	SAARM	RAIM
10 [m] bias	0.00	0.00	5.25	100.0	57.15	0.00	37.60	0.00
25 [m] bias	0.00	0.00	0.00	18.48	100.0	7.31	0.00	74.20
50 [m] bias	0.00	0.00	0.00	0.00	100.0	100.0	0.00	0.00
100 [m] bias	0.00	0.00	0.00	0.00	100.0	100.0	0.00	0.00
10x **R** scale	0.00	0.00	7.23	91.27	78.05	0.65	14.72	8.08

- SAARM: $M = 40$ samples, $\alpha = 6.67 \times 10^{-5}$, RAIM: $r_D = 9.5$ [m], $r_I = 11$ [m]

- Pseudorange measurement noise modeled as $\mathcal{N}(0, \mathbf{R})$, $\mathbf{R} = 4^2$ [m^2]

of the SAARM method allows such performance with no additional modification to the underlying algorithm, which will be further showcased in the all-source simulations below.

4.4.2 All-source Performance.

Having established a benchmark for performance in a simulated but well understood GPS pseudorange problem, the next set of simulations were focused on establishing SAARM performance across a wide variety of fault types and domains, without the need to modify any portion of the fault detection and isolation algorithm. To do so, five all-source navigation simulations were conducted, each using three out of four sensors, which are described in Table 4.4. The fault detection results from each of the five scenarios is summarized in Table 4.5. In the first two scenarios, a simple position bias was injected into the x-axis of the POS1 position sensor, with the other two sensors chosen as velocity sensors. In these scenarios, SAARM detected and isolated the majority of 30 [m] and

Table 4.4: Sensor configuration for all-source scenarios

Scenario	Fault	Affected sensor	Other sensors
1	30 [m] bias	POS1	VEL1, VEL2
2	40 [m] bias	POS1	VEL1, VEL2
3	2 [m/s] bias	VEL1	POS1, POS2
4	5 [m/s] bias	VEL1	POS1, POS2
5	0.1x **R** scale	POS2	POS1, VEL1

POS1: 2D position sensor, $\mathbf{R} = \text{diag}\big([10^2, 10^2]\big)$ [m^2]

POS2: 2D position sensor, $\mathbf{R} = \text{diag}\big([20^2, 20^2]\big)$ [m^2]

VEL1: 2D velocity sensor, $\mathbf{R} = \text{diag}\big([1^2, 1^2]\big)$ [m^2/s^2]

VEL2: 2D velocity sensor, $\mathbf{R} = \text{diag}\big([1^2, 1^2]\big)$ [m^2/s^2]

all of the 40 [m] position biases using velocity sensors for redundancy, which could not be done using least-squares methods or position "solution separation" methods without specific customization. Similarly, in the second set of simulations (scenarios 3 and 4), a simple velocity bias was injected into the x-axis of the VEL1 velocity sensor, with the other two sensors chosen as position sensors. In this case, SAARM detected and isolated the majority of 2 [m/s] and all of the 5 [m/s] velocity biases using position sensors for redundancy. Finally, in the last scenario, SAARM detected virtually all faults defined by a sudden change of the measurement noise variance in the POS2 sensor using a position and a velocity sensor for redundancy. It is important to note here, the change in noise variance in this scenario caused the observed measurements to become *less* noisy than prior to the fault. This type of detection is not critical for safety of flight or protection, but would be necessary in a system that self-corrects or auto-tunes sensor models, as stated in our overall research objective.

Table 4.5: Fault detection and identification performance, all-source simulations

	Probability (%)			
Scenario	**False alarm**	**Missed**	**Isolated**	**Detected only**
1	0.00	13.19	86.75	0.07
2	0.00	0.00	100.0	0.00
3	0.20	6.55	64.15	29.30
4	0.07	0.00	100.0	0.00
5	0.00	0.00	99.93	0.07

SAARM: $M = 40$ samples, $\alpha = 6.67 \times 10^{-5}$

4.5 Chapter Summary

This chapter has proposed a novel method for fault detection and isolation in all-source navigation systems. The proposed method, referred to as SAARM, did not constrain faults to only biases, as has been done in the majority of existing research, and additionally provided a mechanism for detection of multiple simultaneous faults. Driven by a sensor- and fault-agnostic residual monitoring algorithm, the proposed method was not only shown to perform comparably to existing RAIM techniques in the case of a single-satellite bias, but more importantly, shown to detect and isolate various types sensor model mismatches in and across multiple sensing domains such as position and velocity, without the need for synchronous or simultaneous sensor redundancy. Finally, the proposed method was shown to provide a robust measure of system integrity under minimal assumptions, guaranteed to contain the true vehicle horizontal position (within a specified error bound) throughout all phases of the fault detection and identification process. The research in this chapter directly enables self-correcting plug-and-play open architecture navigation systems as well as APNT by providing a generalized and robust method for system integrity in the challenging application of all-source multi-domain navigation.

V. Real-time Validation for Plug-and-play Sensors

As part of the overall all-source resilient navigation objective in our ARMAS framework, this chapter contributes a key component - validation of sensors which have questionable sensor models, in a fault- and sensor-agnostic manner, and without compromising the ongoing navigation solution in the process. The proposed algorithm combines a residual test statistic with the partial update formulation of the Kalman-Schmidt filter to provide a reliable method for sensor model validation that protects the integrity of the navigation solution during the validation process, all using a single existing filter. The proposed method is validated via Monte Carlo simulations against conventional residual monitoring and shown to outperform the standard approach in both fault detection and errors incurred during the validation process. At the time of this writing, the research developed in this chapter is in review for publication in [57].

5.1 Introduction

All-source navigation and APNT have become increasingly relevant over the past two decades, as the research community continues to mature sensor technologies (e.g., vision [97], radio [27], magnetic [22], etc.) and integrate them into navigation systems [38]. However, each additional sensor allowed into a navigation system creates an opportunity for corrupting the navigation solution with errors in sensor modeling, unexpected signal interference, or undetected sensor faults. Though the latter of these challenge areas has been extensively researched [11][12][13][15][16][20][21][36][66][70][86][93][108][109], the primary objective has traditionally been to provide navigation solution integrity (via fault detection and exclusion) in an ongoing multi-sensor navigation process, with the assumption that each sensor in the system is equally likely to experience a fault, and that sensors are properly modeled at the start of the navigation process. Additionally, the research in this challenge area has focused almost exclusively on simultaneously redundant and synchronous multi-sensor systems such as the satellites in the GPS.

Our overall research motivation was to address the APNT challenge for all-source navigation by creating a general sensor-agnostic resiliency framework, which is described in Chapter 3 and [54]. The ARMAS framework provides APNT through the online or real-time detection and self-correction (i.e., auto-tuning) of sensor models that do not match observed measurements, where a biased sensor is simply one possible model mismatch. In support of this overall objective, two specific all-source research areas were investigated for the aforementioned framework: (1) the ability to monitor online sensors (i.e., sensors currently informing the navigation solution) for fault detection and exclusion, and integrity computations, and (2) the ability to initialize offline sensors by validating their stated measurement models while protecting the navigation solution. Of these two processes, the former is partly addressed in the previously mentioned research for systems like GPS (though faults are limited to single-sensor biases), and in Chapter 4 and [58] for all-source sensors, multiple-sensor faults, and faults beyond biases. The latter, which we are referring to as the sensor validation problem, is virtually unaddressed (at the time of writing) in current research and constitutes the focus of this chapter.

5.2 Background

In the context of our research, sensor validation refers to the process of initializing an offline plug-and-play sensor into an ongoing navigation system that is already being informed by a set of (previously initialized) online sensors, and determining whether or not its measurements are statistically adhering to their stated measurement model. The set of online sensors are presumably being monitored for fault detection and exclusion using one of the many integrity monitoring methods previously discussed, such as [58]. In this sense, the offline sensor is initialized separately due to the presumption of possible errors in sensor modeling or the presence of sensor faults. This concept also encompasses the possibility of re-initializing a sensor that was previously found "faulty" and placed offline by a multi-sensor integrity monitoring process (e.g., due to temporary interference or model changes due to environmental variables). Therefore, the challenge in sensor validation is not focused around continuous integrity monitoring for the duration of the navigation sequence, but rather during a fixed "initialization" period, after which the sensor is either determined valid and placed into the integrity monitoring pool with all other online sensors, or deemed invalid and either placed back into offline status or placed into sensor model remedial procedures as described in [54]. The primary challenge in validating an offline sensor is ensuring its (presumed) faulty measurements do not corrupt the ongoing navigation solution, while simultaneously ensuring the chosen statistical test is capable of detecting nuanced differences between the observed measurements and their stated measurement model [52]. Technically, this could be accomplished via two existing methods, neither requiring additional development. The first method would be to simply collect a series of pre-update residuals between the offline (untrusted) sensor and the (trusted) navigation solution, without actually applying the measurement update, and testing the observed statistical distribution [20][28][76]. However, this method would preclude any sensors that require additional filter state estimates (i.e., not currently being estimated by the filter) as part of the measurement model such as a GPS receiver clock error state, a sensor run-to-run variable scale factor, or variable camera intrinsic and extrinsic parameters, to name a few, since they are only observable by the untrusted sensor. The second method would be to add the untrusted sensor to the pool of trusted online sensors already being monitored by a fault detection or integrity monitoring method. However, this would not only require an additional sub-filter in the multi-filter fault exclusion architecture, but more importantly, it would unnecessarily degrade the system integrity computations by allowing an untrusted sensor to affect the various sub-filter solutions involved in the fault exclusion process. As later shown, our proposed method leverages the concept of a "partial update" [17], derived from the Kalman-Schmidt [81] filter formulation in order to provide sensitive fault detection for sensor model validation while protecting the ongoing navigation solution, using a single-filter architecture. As such, the method developed in this chapter, refered to henceforth as Real-time Validation for Plug-and-play Sensors (RVPS), provides a significant contribution to the state-of-the-art in that it:

- Provides a general method for all-source plug-and-play sensor model validation,
- Estimates additional sensor states without compromising the navigation solution, and
- Protects system integrity computations during validation using a single existing filter.

The remainder of this chapter is divided into three additional sections. Section 5.3 develops the necessary notation, the partial update formulation, and the residual test statistic used for validation. In Section 5.4, the performance of the proposed method is compared (in terms of probability of detection and navigation solution corruption) against full-update residual methods in a simulated all-source navigation problem. Finally, Section 5.5 summarizes the research contributions and provides ideas for future work.

5.3 Methodology

Adapting the Kalman filter [59] notation from [76][77], consider an ongoing (possibly) non-linear dynamic system of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left[\mathbf{x}(t), \mathbf{u}(t), t\right] + \mathbf{G}(t)\mathbf{w}(t), \tag{5.1}$$

where $\mathbf{x}$ is the $N \times 1$ navigation state vector containing the system states, $\mathbf{u}$ is the control input vector, $\mathbf{G}$ is an $N \times W$ linear operator, and $\mathbf{w}$ is a $W \times 1$ white Gaussian noise process with a $W \times W$ continuous process noise strength matrix $\mathbf{Q}$. At time $t = t_k$, the state estimate vector and corresponding state estimation error covariance matrix are given by $\hat{\mathbf{x}}(t_k)$ and $\mathbf{P}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}(t_k)$, respectively, and produced by the system's "main filer," which is informed by a set of online (trusted) sensors that are presumably being monitored for fault detection and exclusion. Next, starting at a given initialization time, we wish to begin validating an offline and untrusted plug-and-play sensor that provides (possibly) multidimensional $Z \times 1$ measurements of the form

$$\mathbf{z}(t_k) = \mathbf{h}\left[\mathbf{x}(t_k), \boldsymbol{\epsilon}(t_k), \mathbf{u}(t_k), t_k\right] + \mathbf{v}_k, \tag{5.2}$$

where $\mathbf{h}$ is a (possibly) nonlinear measurement function, $\boldsymbol{\epsilon}$ is a $U \times 1$ vector of sensor-unique states needed for measurement processing but not currently estimated by the filter, and $\mathbf{v}_k$ is a $Z \times 1$ discrete white Gaussian noise process with covariance matrix $\mathbf{R}(t_k)$. In order to initialize the sensor and process measurements, the sensor-unique states, $\boldsymbol{\epsilon}$, must be augmented into $\mathbf{x}$ using their initial estimate $\hat{\boldsymbol{\epsilon}}(t_i^-)$ and corresponding initial estimation error covariance matrix, $\mathbf{P}_{\hat{\epsilon}\hat{\epsilon}}(t_i^-)$, where t_i is the initialization time. After augmenting the state-space, we wish to begin collecting "pre-update" residuals in order to evaluate their likelihood given the stated measurement model in (5.2). In order to prevent the initial uncertainty in $\boldsymbol{\epsilon}$, $\mathbf{P}_{\hat{\epsilon}\hat{\epsilon}}(t_i^-)$, from dominating the uncertainty around the residuals (and thus masking any potential sensor model faults), we must estimate $\boldsymbol{\epsilon}$ by accepting measurement updates. However, since the sensor is untrusted, we wish to protect the ongoing navigation solution, $\mathbf{x}$, while doing so. Employing the partial update [17] formulation of the Kalman-Schmidt filter [81] allows us to accept measurement updates while designating a subset of the state-space variables as "consider" states whose statistical distribution is considered during the measurement update, but whose distribution (e.g., the state estimate and error covariance matrix) is unaffected by the measurement update. Though traditionally the states designated as "consider" states have been primarily unobservable biases, in our research, we designate the "core" navigation solution states (e.g., position, velocity, acceleration, etc.) as "consider" in order to protect their estimates from corruption during the estimation of any sensor-unique states. To formulate the partial update, we initialize the augmented state-space as

$$\hat{\mathbf{y}}(t_i^-) = \begin{bmatrix} \hat{\mathbf{x}}(t_i^-) & \hat{\boldsymbol{\epsilon}}(t_i^-) \end{bmatrix}^{\mathrm{T}}, \tag{5.3}$$

$$\mathbf{P}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}(t_i^-) = \begin{bmatrix} \mathbf{P}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}(t_i^-) & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{\hat{\boldsymbol{\epsilon}}\hat{\boldsymbol{\epsilon}}}(t_i^-) \end{bmatrix}, \tag{5.4}$$

where $\hat{\mathbf{y}}$ is the $(N+U) \times 1$ augmented state estimate, $\mathbf{P}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}$ is the corresponding $(N+U) \times (N+U)$ augmented state estimation error covariance matrix, and keeping in mind the state dynamics associated with the propagation of $\boldsymbol{\epsilon}$ must also be augmented into $\mathbf{G}$, $\mathbf{w}$, and $\mathbf{Q}$. Next, we define a partial update vector, $\boldsymbol{\beta}$, using

$$\underset{(N+U)\times 1}{\boldsymbol{\beta}} = \begin{bmatrix} \underset{1\times N}{\mathbf{0}} & \underset{1\times U}{\mathbf{1}} \end{bmatrix}^{\mathrm{T}}, \tag{5.5}$$

where the entries in $\boldsymbol{\beta}$ corresponding to $\mathbf{x}$ are set to zero, and the entries corresponding to $\boldsymbol{\epsilon}$ are set to one. Once $\boldsymbol{\beta}$ is defined, we execute a standard (full) measurement update to obtain the post-update augmented state estimate, $\hat{\mathbf{y}}(t_i^+)$, and its corresponding error covariance matrix, $\mathbf{P}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}(t_i^+)$. Then, as shown in [17], we essentially "roll back" the distribution of the protected states using

$$\hat{\mathbf{y}}(t_i^{++}) = \boldsymbol{\beta} \odot \hat{\mathbf{y}}(t_i^+) + \boldsymbol{\gamma} \odot \hat{\mathbf{y}}(t_i^-), \tag{5.6}$$

$$\mathbf{P}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}(t_i^{++}) = \mathbf{B} \odot \mathbf{P}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}(t_i^+) + \boldsymbol{\Gamma} \odot \mathbf{P}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}(t_i^-), \tag{5.7}$$

where $\odot$ is the Hadamard or point-wise product, and

$$\boldsymbol{\gamma} = \mathbf{1} - \boldsymbol{\beta}, \tag{5.8}$$

$$\boldsymbol{\Gamma} = \boldsymbol{\gamma}\boldsymbol{\gamma}^{\mathrm{T}}, \tag{5.9}$$

$$\mathbf{B} = \mathbf{1} - \boldsymbol{\Gamma}. \tag{5.10}$$

In essence, the partial updates prevent the untrusted sensor measurements from corrupting the navigation solution to account for a bad model or a fault, and increase fault observability in the residuals (i.e., prevent residuals from becoming zero-mean), as later shown.

Therefore, the application of partial updates becomes critical when initializing an untrusted sensor that has a low measurement noise error covariance matrix compared to the online sensors since it would otherwise greatly influence the ongoing navigation solution. The partial update process described above can be then repeated after t_i until steady-state is reached, or for a fixed number of samples, as set by the user. Once this first initialization phase is complete, we can then continue applying partial updates and begin collecting residuals to be tested for likelihood. Immediately prior to a partial measurement update, the estimated measurement, $\hat{\mathbf{z}}$, is generated using

$$\hat{\mathbf{z}}(t_k^-) = \mathbf{h}\left[\hat{\mathbf{y}}(t_k^-), \mathbf{u}(t_k), t_k\right], \tag{5.11}$$

where $\hat{\mathbf{x}}(t_k^-)$ and $\hat{\boldsymbol{\epsilon}}(t_k^-)$ are contained in the augmented state vector $\hat{\mathbf{y}}(t_k^-)$. Meanwhile, the error covariance matrix for the estimated measurement, $\mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}(t_k^-)$, is generated based on the type of filtering algorithm. For example, in a linearized filter (such as an Extended Kalman Filter) it can be computed using

$$\mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}} = \mathbf{H}\mathbf{P}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}\mathbf{H}^{\mathrm{T}}, \tag{5.12}$$

where the time index (t_k^-) is omitted for simplicity, and $\mathbf{H}$ represents the Jacobian of $\mathbf{h}$ about the point $\hat{\mathbf{y}}(t_k^-)$. For information on generating $\mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}$ in an UKF, the reader is referred to [101]. Finally, the pre-update residual vector, $\mathbf{r}$, and its associated covariance matrix, $\mathbf{P}_{\mathbf{rr}}$, is given by

$$\mathbf{r}(t_k) = \mathbf{z}(t_k) - \hat{\mathbf{z}}(t_k^-) \tag{5.13}$$

$$\mathbf{P}_{\mathbf{rr}}(t_k) = \mathbf{R}(t_k) + \mathbf{P}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}(t_k^-). \tag{5.14}$$

Having derived the residual vector, $\mathbf{r}(t_k)$, and its associated covariance matrix, $\mathbf{P}_{rr}(t_k)$, in (5.13) and (5.14), we now define a residual-space test statistic to determine if a set of observed residuals are adhering to their expected distribution. Since our overall research objective is to limit the assumptions on the type of fault (i.e., the fault could be a bias, an incorrectly stated noise covariance matrix, or incorrect calibration of measurement function parameters), we did not model two competing distributions as would be needed to employ a LRT [63]. Instead, we focused on the single likelihood function of the residuals, based on measuring their Mahalanobis distances [28] due to their simplicity and "standardizing" properties, which were found useful in time-changing processes such as navigation.

Given a Z-dimensional Gaussian distribution with mean $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma}$, the squared Mahalanobis distance, d^2, between an observation $\mathbf{g}$, and the centroid of the distribution is then given by

$$d^2 = (\mathbf{g} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{g} - \boldsymbol{\mu})\,. \tag{5.15}$$

Additionally, d^2 is known [23][28] to follow a Chi-Square distribution with Z degrees of freedom. Moreover, the sum of M independent d^2 distances is also known to follow a Chi-Square distribution with $M \times Z$ degrees of freedom. Since Kalman filter pre-update residuals are assumed to be a zero-mean white sequence [76], we let $\mathbf{g} = \mathbf{r}(t_k)$ from (5.13), $\boldsymbol{\Sigma} = \mathbf{P}_{\mathbf{rr}}(t_k)$ from (5.14), and $\boldsymbol{\mu} = \mathbf{0}$. Subsequently, we can develop a fault detection test, given set of M, Z-dimensional residual vectors collected between $t = t_k$ and $t = t_{k+M}$ using

$$H_0 : \chi^* > \chi^2(\alpha/2, M \times Z) \text{ and } \chi^* < \chi^2(1 - \alpha/2, M \times Z) \tag{5.16}$$

$$H_1 : \chi^* < \chi^2(\alpha/2, M \times Z) \text{ or } \chi^* > \chi^2(1 - \alpha/2, M \times Z), \tag{5.17}$$

where M defines the number of averaging samples in the test (and consequently the sensor validation period), α is the significance level of the test (i.e., probability of false alarm, P_f), H_0 is defined as the sensor model is valid (fault not present), H_1 is defined as the sensor model is invalid (fault present), and

$$\chi^* = \sum_{s=k}^{k+M} d^2(t_s), \tag{5.18}$$

$$d^2(t_k) = \mathbf{r}^{\mathrm{T}}(t_k)\,[\mathbf{P}_{rr}(t_k)]^{-1}\,\mathbf{r}(t_k). \tag{5.19}$$

It is important to note a few key points about the above test statistic. First, it is designed to detect mismatches (in any domain such as position, velocity, etc.) between sensor measurements and their stated models, in both the upper and lower ends of the resulting Chi-Square distribution. This was purposely done so that our method could not only detect unlikely large or variable residuals resulting from biases, problems in the measurement function, its parameters, or under-stating the measurement error covariance matrix, but additionally, unlikely small residuals resulting from over-stating the measurement error covariance matrix.

In summary, this section has developed the formulation needed to:

- Estimate a set of sensor-unique states for an untrusted offline sensor while protecting the ongoing and trusted navigation solution,
- Prevent the untrusted sensor from "masking" an invalid sensor model or fault by corrupting the navigation solution, and
- Test a collection of residuals for likelihood with minimal assumptions on the fault type.

5.4 Simulation Results

The proposed method was evaluated via a series of Monte Carlo simulations using two vehicles informed by an online trusted 2D position sensor (Sensor A) and an offline untrusted 2D velocity sensor (Sensor B). In the first vehicle (Aircraft 1), RVPS was used to initialize and validate Sensor B based on the method described in this chapter, namely using partial updates during the collection of the residual test statistic. In the second vehicle (Aircraft 2), the residuals from Sensor B were monitored using the same residual statistic but without using partial updates, thereby allowing its measurements to influence the trusted solution. For all simulations, the true system dynamics were driven by a 2D kinematic model given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{x}}_p(t) \\ \dot{\mathbf{x}}_v(t) \\ \dot{\mathbf{x}}_a(t) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_v(t) \\ \mathbf{x}_a(t) \\ -\frac{1}{\tau_a}\mathbf{x}_a(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{w}(t) \end{bmatrix}, \tag{5.20}$$

where $\mathbf{x}_p$ is the vehicle's 2D position in [m], $\mathbf{x}_v$ is the 2D velocity in [m/s], $\mathbf{x}_a$ is the 2D acceleration in [m/s^2] and propagated by a FOGM process with time constant $\tau_a = 90$ [s], and $\mathbf{w}(t)$ is a 2D white Gaussian noise process with $E\left[\mathbf{w}(t)\mathbf{w}(t+\tau)^{\mathrm{T}}\right] = \mathbf{Q}\delta(\tau)$ and

$$\mathbf{Q} = (1.5 \times 10^{-3})^2 \underset{2\times2}{\mathbf{I}} \ [\mathrm{m}^2/\mathrm{s}^5]. \tag{5.21}$$

Sensor A measurements were modeled using

$$\mathbf{z}^{[A]}(t_k) = \mathbf{x}_p(t_k) + \mathbf{v}_k^{[A]}, \tag{5.22}$$

$$E\left[\mathbf{v}_k^{[A]}\mathbf{v}_k^{[A]^{\mathrm{T}}}\right] = \mathbf{R}^{[A]}(t_k) = \begin{bmatrix} 20^2 & 0 \\ 0 & 20^2 \end{bmatrix} [\mathrm{m}^2], \tag{5.23}$$

and its simulated measurements were drawn from the modeled distribution. Meanwhile, Sensor B measurements were modeled as

$$\mathbf{z}^{[B]}(t_k) = \boldsymbol{\epsilon} \odot \mathbf{x}_v(t_k) + \mathbf{v}_k^{[B]}, \tag{5.24}$$

$$E\left[\mathbf{v}_k^{[B]}\mathbf{v}_k^{[B]^{\mathrm{T}}}\right] = \mathbf{R}^{[B]}(t_k) = \begin{bmatrix} 1^2 & 0 \\ 0 & 1^2 \end{bmatrix} [\mathrm{m}^2/\mathrm{s}^2], \tag{5.25}$$

where $\boldsymbol{\epsilon}$ is a constant but unknown 2D scale factor. Additionally, Sensor B measurements were corrupted by adding a constant x-velocity bias. The simulations consisted of seven different velocity biases, and one scaling of $\mathbf{R}^{[B]}$. For each simulation, 3000 trials were conducted. For each trial in each simulation, the initial true position and initial position state estimates were set to zero. The initial true velocity was randomly drawn from a $\mathcal{N}\left(0, 10^2\right)$ [m^2/s^2] distribution for both the x and y axes, while the initial velocity state estimates were set to zero. The initial true acceleration and acceleration state estimates were set to zero. The initial state estimation error covariance was set to 10^2 [m^2] in position, 10^2 [m^2/s^2] in velocity, and $(1.5 \times 10^{-3})^2$ [m^2/s^4] in acceleration. Each trial was propagated using $\Delta t_k = 0.5$ [s], starting at $t_k = 0$ [s] with an offline sensor initialization at $t_i = 60$ [s], and terminating at $t_k = 180$ [s]. At t_i, the constant Sensor B scale factor was drawn from a $\mathcal{N}\left(1, 0.1^2\right)$ distribution for both the x and y axes, their initial estimate was set to 1 for both axes, and their initial uncertainty was set to 0.1^2. Across all simulations, the RVPS false alarm rate was set to $\alpha = 1/15000 = 6.67 \times 10^{-5}$, while the sensor validation period was set to 120 samples, which was equivalent to 60 [s]. The first half of the validation period (60 samples) was used to allow the estimation of the Sensor B scale factor to reach steady-state, and the second half was used in formulating the residual test statistic shown in (5.16) and (5.17), making the testing epoch $M = 60$ samples.

From a partial update perspective and using (5.8) through (5.10), at the time of Sensor B initialization in Aircraft 1, the 6×1 trusted navigation solution, $\mathbf{x}$, was augmented with the 2×1 scale-factor states, $\boldsymbol{\epsilon}$, to form the 8×1 augmented state vector, $\mathbf{y}$, resulting in

$$\boldsymbol{\beta} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \tag{5.26}$$

$$\boldsymbol{\gamma} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \tag{5.27}$$

$$\underset{8\times 8}{\boldsymbol{\Gamma}} = \begin{bmatrix} \underset{6\times 6}{\mathbf{1}} & \vdots \\ \dots & \mathbf{0} \end{bmatrix}, \tag{5.28}$$

$$\underset{8\times 8}{\mathbf{B}} = \begin{bmatrix} \underset{6\times 6}{\mathbf{0}} & \vdots \\ \dots & \mathbf{1} \end{bmatrix}. \tag{5.29}$$

Figure 5.1 illustrates a sample trajectory comparison from one of the biased Monte Carlo trials, while Figure 5.2 illustrates the corresponding d^2 observations used in the computation of the test statistic from (5.18). As shown, the Aircraft 1 (using RVPS) navigation solution was unaffected while the system estimated the scale factor, $\boldsymbol{\epsilon}$, resulting in higher than expected Mahalanobis distances due to the unmodeled bias, and ultimately leading to a correct H_1 decision (sensor model invalid). In contrast, the navigation solution from Aircraft 2 (using only the residual monitoring test from (5.16) and (5.17) without partial updates) was corrupted due to the filter's ability to absorb the Sensor B unmodeled bias into observable states, causing the corresponding d^2 observations to rapidly fall to statistically expected values, and ultimately leading to an incorrect H_0 (sensor model valid) decision.

Table 5.1 summarizes the Monte Carlo detection performance for each of the seven bias levels and the simulation where $\mathbf{R}^{[B]}$ was scaled by 10x from its stated model value. As shown, using the same residual test statistic and P_f, the RVPS method (using partial updates) clearly outperformed conventional residual monitoring in its ability to detect an unmodeled sensor bias while simultaneously estimating the unknown scale factor, $\boldsymbol{\epsilon}$. In the case of an incorrectly stated measurement noise covariance matrix, RVPS performed as well as conventional residual monitoring in terms of detection, but still provided significant benefits in terms of solution integrity during detection as shown below.

Figure 5.3 illustrates the distribution of the mean (across 3000 trials) position RSS errors (between each aircraft's navigation solution and the true trajectory) committed during the 120 [s] validation period. As shown, using RVPS prevented an improperly modeled sensor from corrupting the ongoing navigation solution during the validation period, resulting in significantly lower position RSS errors across all fault types.

Finally, Figure 5.4 illustrates the fault detection performance for a fixed 30 [m/s] bias based on 30 detection threshold levels, each with 3000 trials. Once again, this demonstrates using RVPS clearly improves the system's ability to detect an invalid sensor model using the proposed residual test statistic.

Table 5.1: Sensor validation fault detection comparison, 2D velocity sensor

	Probability (%)					
	False alarm		**Missed**		**Detected**	
Fault type	RVPS	RM	RVPS	RM	RVPS	RM
1 [m/s] bias	0.00	0.00	99.15	100.0	0.85	0.00
5 [m/s] bias	0.00	0.00	74.30	99.35	25.70	0.65
10 [m/s] bias	0.00	0.00	56.57	100.0	43.43	0.00
20 [m/s] bias	0.00	0.00	32.61	91.85	67.39	8.15
30 [m/s] bias	0.00	0.00	17.49	85.50	82.51	14.50
50 [m/s] bias	0.00	0.00	4.76	73.11	95.24	26.89
100 [m/s] bias	0.00	0.00	0.48	61.16	99.52	38.84
10x **R** scale	0.00	0.00	0.00	0.00	100.0	100.0

*χ^2 test for both RVPS and residual monitoring (RM) was set up using:

$M = 60$ samples, $\alpha = 6.67 \times 10^{-5}$

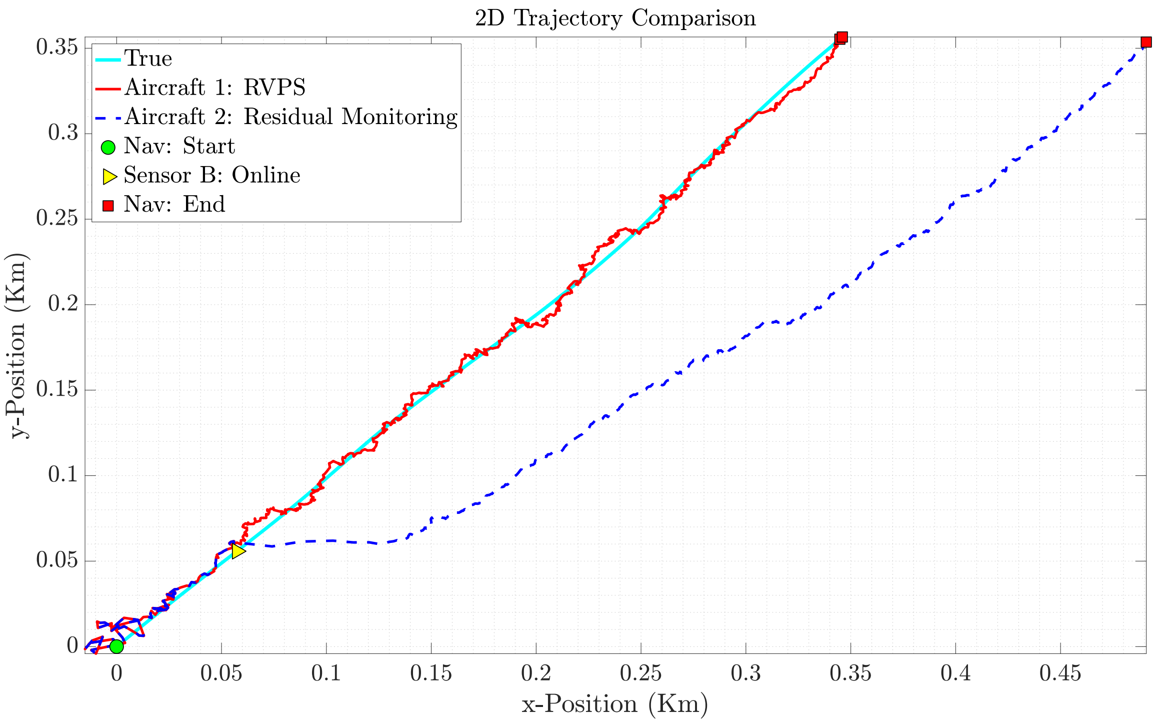


Figure 5.1: Example 2D trajectory comparison, 2D velocity sensor, bias = 20 [m/s]

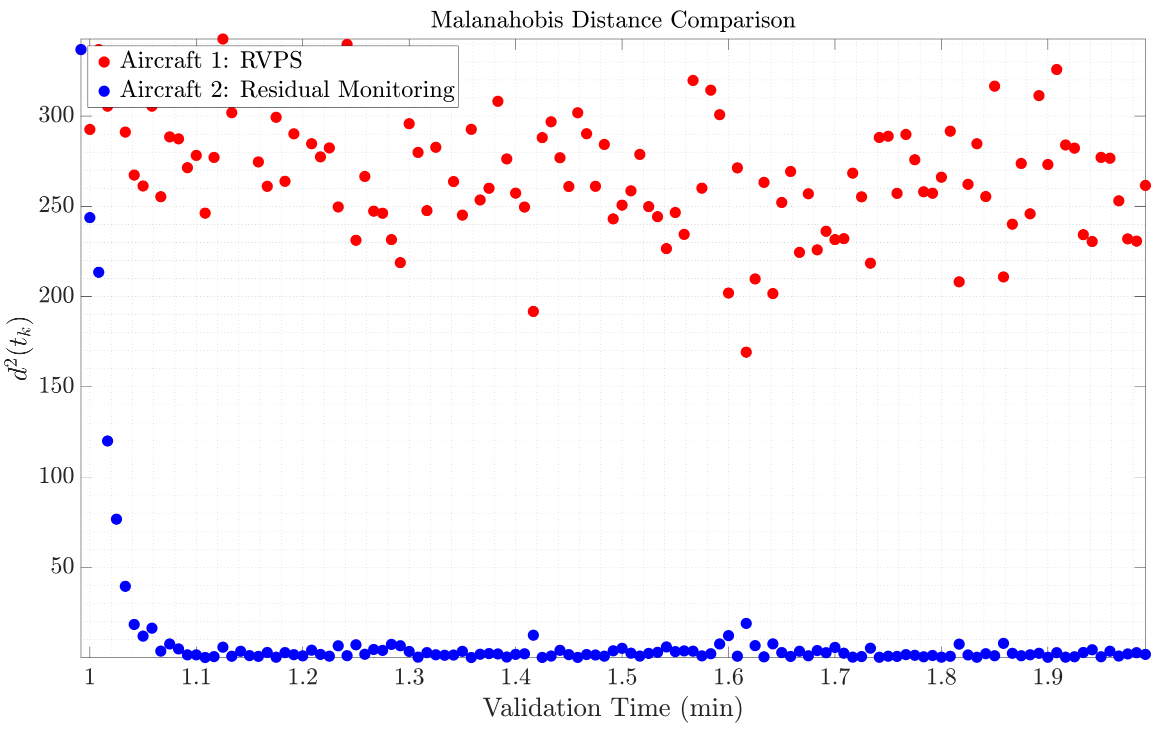


Figure 5.2: Example residual d^2 comparision, 2D velocity sensor, bias = 20 [m/s]

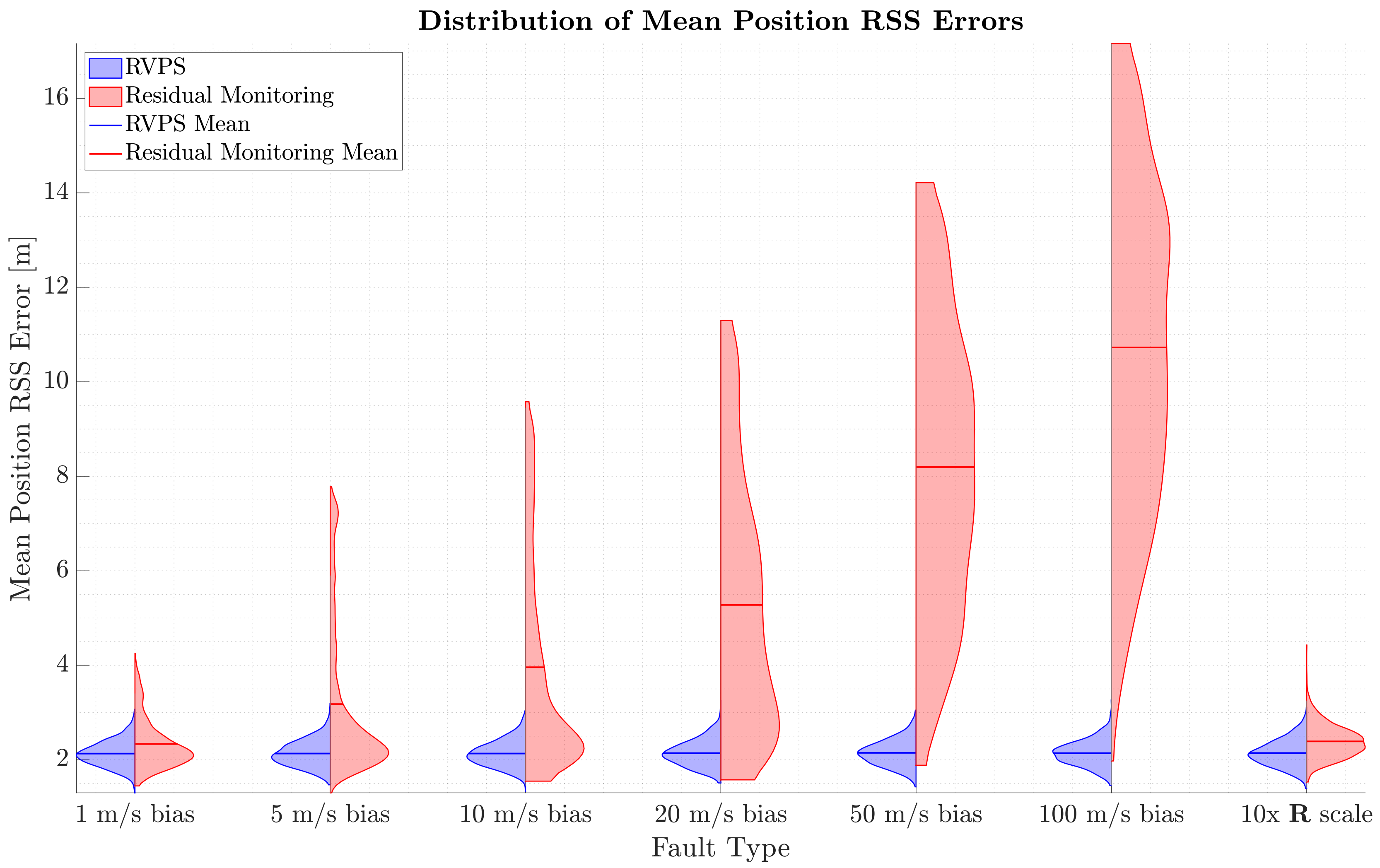


Figure 5.3: Comparison of mean position RSS errors, RVPS vs. residual monitoring.

5.5 Chapter Summary

This chapter has proposed a novel method for real-time model validation for plug-and-play sensors, specifically aimed at all-source navigation systems. The proposed method, referred to as RVPS, enabled the estimation of sensor-unique states without compromising the navigation solution, thereby protecting the system integrity computations during the validation period, all using a single existing filter. A series of Monte Carlo simulations demonstrated the method's ability to not only detect invalid sensor models more reliably, but additionally prevent the detection process from corrupting the navigation solution. This method complements previous developments in all-source APNT integrity monitoring, such as the ones described in Chapter 4 and [58], and directly enables self-correcting plug-and-play open architecture navigation systems such as the one described in Chapter 3 and [54].

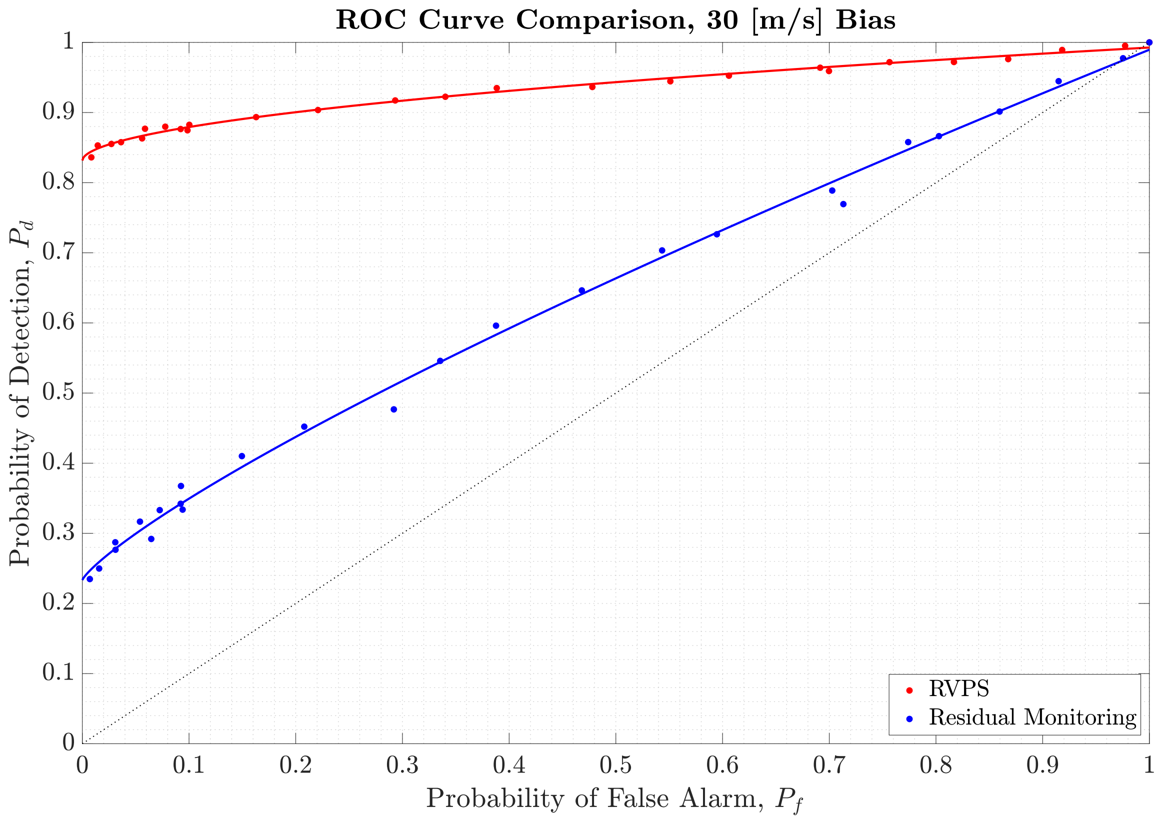


Figure 5.4: Fault detection ROC curve comparison, bias = 30 [m/s]

VI. A Complete Online Algorithm for Air Data System Calibration

This chapter provides one of two sensor calibration methods that complement the overall resilient navigation research thrust. Specifically, this chapter details a novel Pitot-static online calibration algorithm suitable for inclusion in the calibration mode of the ARMAS framework. Air Data Systems require costly calibration of their static pressure sensors to characterize errors caused by the act of flying. Altitude-based methods for measuring these so-called static position errors, such as the Tower Fly-by, produce accurate results but require an elaborate fly-by site, multiple experiments to capture the relationship between error and airspeed, and are limited to subsonic airspeeds due to inherent hazards to land-based and aircraft structures from low-altitude supersonic flight. Airspeed-based methods using GPS are generally easier to execute, but tend to yield less precise results, and still require multiple experiments. Additionally, they require temperature probe calibration from external sources. This chapter proposes a self-contained, online method for complete air data calibration. The proposed method uses a Kalman Smoother to fuse GPS altitude and airspeed measurements, aircraft attitude, and air data, to produce the full static position error curve as a function of Mach number in a single experiment, with no need for external temperature calibration, and with no supersonic limitations. The proposed method is validated using T-38C flight data, and is shown to reduce cost by 88% while modeling a 42% larger domain when compared to current methods. The research developed in this chapter has been published in [56].

6.1 Introduction

Production aircraft are typically equipped with a Pitot-static sensor system, sometimes referred to as an Air Data System (ADS). The ADS is composed of a Pitot-tube, which measures total air pressure, a static port, which measures static air pressure, and an Air

Data Computer (ADC), which combines the sensor readings into various airspeed and altitude instrument readings. The ADC uses Pitot-static relationships to convert differences between total and static pressure into airspeed readings, and static pressure measurements into altitude readings. Since airspeed and altitude are directly derived from pressure, they are intrinsically linked to lift and drag, which in turn, are linked to key performance parameters such as climb rate, climb angle, specific range, and endurance. Unfortunately, the act of flying through an air mass inherently corrupts the static port's ability to measure ambient pressure, or the true static pressure in the undisturbed atmosphere, and creates an error called Static Position Error (SPE) [42].

SPE, or ΔP_p, is defined as the difference between static pressure, P_s, and ambient pressure, P_a, and is often normalized by P_s when comparing readings from various flight conditions, using

$$\frac{\Delta P_p}{P_s} = \frac{P_s - P_a}{P_s}. \tag{6.1}$$

Since it affects static pressure readings, SPE is responsible for errors in both airspeed and altitude. Such errors are not only unique for each type of aircraft, but also tend to change as a function of Mach number and AoA. Many offline algorithms have been used to estimate SPE via altitude or airspeed measurements [33][42][51][72][73][80][83]. However, even the most advanced techniques tend to either: require a large logistical footprint, result in biased estimates, or use assumptions that only apply to a small subset of airframes. This chapter proposes a novel algorithm for determining SPE that is significantly more accurate than state-of-the-art methods, and can be executed in an online fashion for any aircraft without the need for multiple controlled experiments.

6.2 Background

A considerable amount of research has been devoted to solving the problem of SPE. Most notably, the flight test community has developed numerous experiments designed to characterize SPE for each type of aircraft across its entire Mach number domain. In general, three types of techniques have been found in literature: altitude methods, airspeed methods, and pressure methods. Since SPE affects both altitude and airspeed, such techniques are aimed at determining airspeed and altitude error, respectively, as a function of airspeed, using external truth sources. Meanwhile, pressure techniques directly measure static pressure errors using ambient pressure readings from weather balloons.

6.2.1 Altitude Methods.

The most widely used altitude method for SPE calibration is called the Tower Fly-by (TFB) [33][42]. A general TFB diagram is shown in Figure 6.1. The TFB technique is easy to execute from a flying perspective and also produces data that is easy to process. The TFB aims to determine an altitude error correction, ΔH_{pc}, by comparing the indicated altitude in the aircraft's altimeter, H_{ic}, to an externally measured reference altitude, H_c, which is derived from a theodolite measurement at a ground-based observation tower. The aircraft flies at a constant altitude and airspeed as it passes by the observation tower, where the "truth" altitude, H_c, is recorded. At the same time, the aircraft records its altitude, H_{ic}. The error correction relationship is then given by

$$\Delta H_{pc} = H_c - H_{ic}. \tag{6.2}$$

The error correction given by (6.2) is then used to sample SPE across the entire Mach number domain for a given aircraft by repeating the TFB at various Mach number conditions. The resulting altitude error can be converted to a corresponding pressure error using (6.1) with

$$P_s = P_{SL}\left(1 - 6.875\,59 \times 10^{-6} H_{ic}\right)^{5.2559}, \tag{6.3}$$

$$P_a = P_{SL}\left(1 - 6.875\,59 \times 10^{-6} H_{c}\right)^{5.2559}, \tag{6.4}$$

where P_{SL} is the atmospheric pressure at sea level on a standard day [6]. The computed pressure error can then be used to infer airspeed errors at similar conditions. Even though it is simple and accurate, the TFB method is limited in that it requires multiple (time-consuming) fly-bys to sample the underlying $\Delta H_{pc}(M)$ curve, a team of individuals at the tower site to perform manual theodolite readings, and most importantly, an established TFB site with known geometric conditions. Additionally, obtaining TFB data for transonic and supersonic conditions proves to be problematic due to sonic boom concerns for nearby structures and personnel, as well as structural concerns for the test aircraft due to the high dynamic pressure experienced at supersonic speeds and low altitude.

6.2.2 Airspeed Methods.

Airspeed methods for characterizing SPE have seen the most innovation in recent literature due to the emergence and proliferation of Differential Global Positioning System (DGPS) in military and commercial aircraft. Several state-of-the art airspeed techniques [51][72][73][80][83] rely on a simplified two-dimensional transformation from the body-frame (*b*-frame) to the navigation frame (*n*-frame), usually referred to as the "wind triangle", and shown in Figure 6.2.

In [51], the true airspeed error, ΔV_T, which is caused by SPE, is estimated using DGPS by assuming a constant and unknown wind vector. The aircraft flies a 360-degree turn at a constant indicated altitude and indicated airspeed, which allows the unknown parameters ($\mathbf{v}_W$ and $\mathbf{v}_{T_i}$) to become observable to a linear model. The model is developed using two-dimensional vector geometry from the wind triangle via

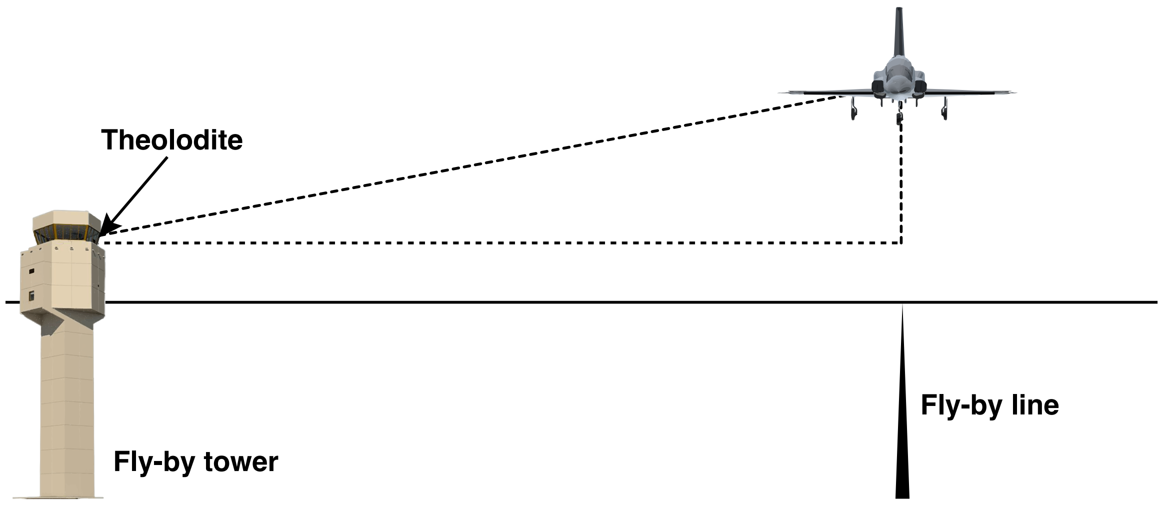


Figure 6.1: Illustration of the Tower Fly-by.

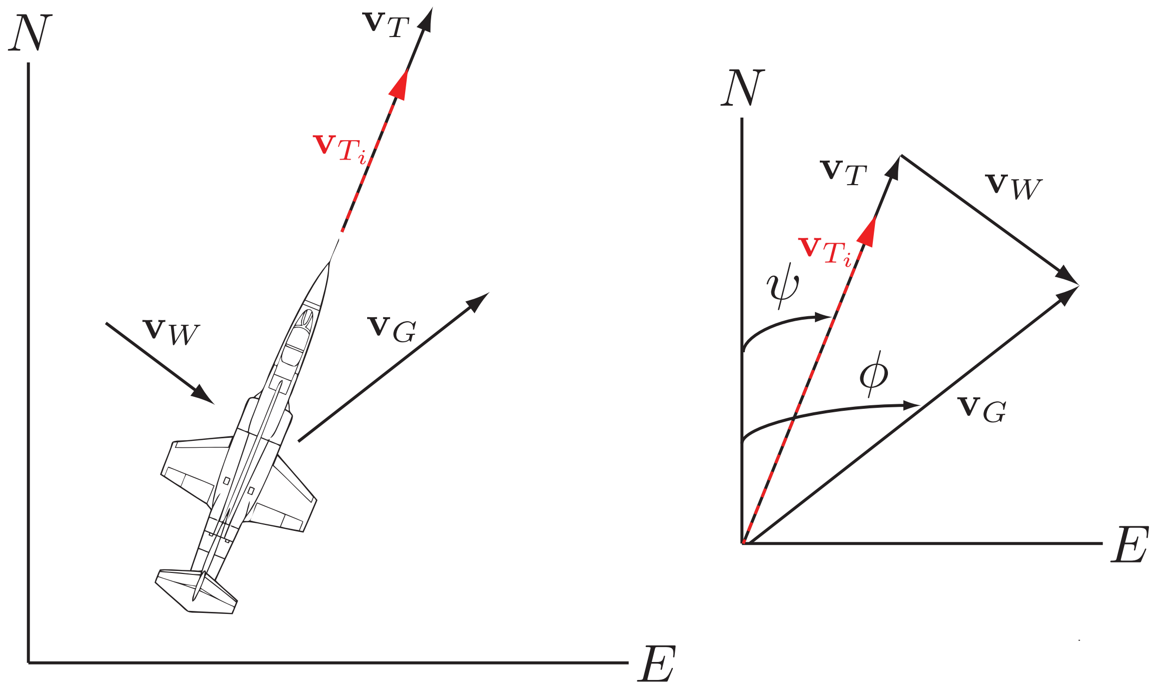


Figure 6.2: Illustration of the wind triangle.

$$\mathbf{v}_T + \mathbf{v}_W = \mathbf{v}_G \tag{6.5}$$

$$\implies (\mathbf{v}_{T_i} + \mathbf{\Delta V}_T) + \mathbf{v}_W = \mathbf{v}_G \tag{6.6}$$

$$\implies \mathbf{\Delta V}_T + \mathbf{v}_W = \mathbf{v}_G - \mathbf{v}_{T_i} \tag{6.7}$$

$$\implies \Delta V_T \cos(\psi) + v_{W_N} = v_G \cos(\phi) - v_{T_i} \cos(\psi), \tag{6.8}$$

$$\Delta V_T \sin(\psi) + v_{W_E} = v_G \sin(\phi) - v_{T_i} \sin(\psi) \tag{6.9}$$

$$\implies \underbrace{\begin{bmatrix} v_{G_1} \cos(\phi_1) - v_{T_{i_1}} \cos(\psi_1) \\ \vdots \\ v_{G_M} \cos(\phi_M) - v_{T_{i_M}} \cos(\psi_M) \\ v_{G_1} \sin(\phi_1) - v_{T_{i_1}} \sin(\psi_1) \\ \vdots \\ v_{G_M} \sin(\phi_M) - v_{T_{i_M}} \sin(\psi_M) \end{bmatrix}}_{2S\times 1} = \underbrace{\begin{bmatrix} \cos(\psi_1) & 1 & 0 \\ \vdots & \vdots & \vdots \\ \cos(\psi_M) & 1 & 0 \\ \sin(\psi_1) & 0 & 1 \\ \vdots & \vdots & \vdots \\ \sin(\psi_M) & 0 & 1 \end{bmatrix}}_{2S\times 3} \begin{bmatrix} \Delta V_T \\ v_{W_N} \\ v_{W_E} \end{bmatrix}, \tag{6.10}$$

where S is the number of data samples collected during the turn, v_{T_i} is the aircraft's measured (or SPE corrupted) True Airspeed (TAS), ψ is true heading, v_G and ϕ are ground speed and ground track respectively, as measured by DGPS, ΔV_T is the unknown TAS error, and v_{W_N} and v_{W_E} are the unknown constant wind vector components. In contrast to similar methods such as the Cloverleaf [72] (where the aircraft is flown at three distinct headings instead of around a full circle), this method produces a statistical model for its estimated variables that takes advantage of modern-day in-flight data recording systems. However, it is limited in the fact that, much like the TFB, it relies on multiple experiments to collect the necessary point samples of the underlying $\Delta V_{pc}(M)$ function, and some aircraft may not be able to sustain a constant-speed turn at supersonic conditions. Additionally, v_{T_i} is difficult to measure since it must be derived from Indicated Airspeed (IAS), V_{ic}, and ambient temperature, T_a, which is given by

$$T_a = \frac{T_{ic}}{1 + 0.2 K_t M_{pc}^2}, \tag{6.11}$$

where T_{ic} is the measured temperature (known as Total Temperature), M_{pc} is SPE-corrected Mach number, and K_t is a temperature calibration parameter that must be derived from other experiments such as the TFB, or external sources such as weather balloons. As shown, (6.11) requires knowledge of SPE to correct T_{ic}, however, determining SPE requires K_t. Therefore, the problem is usually solved by determining K_t prior to SPE and approximating M_{pc} with indicated Mach number, M_{ic}, when deriving v_{T_i}, and iterating until convergence is achieved. Having obtained ΔV_T, the subsonic airspeed error can be converted to a pressure error using

$$\frac{\Delta P_p}{P_s} = \left(\frac{1}{\frac{q_{c_{ic}}}{P_s} + 1} - \frac{1}{\frac{q_c}{P_a} + 1} \right) \frac{P_T}{P_s}, \tag{6.12}$$

where

$$\frac{q_c}{P_a} = \left(1 + 0.2 \left(\frac{V_{T_i} + \Delta V_T}{a} \right)^2 \right)^{7/2} - 1, \tag{6.13}$$

$$\frac{q_{c_{ic}}}{P_s} = \left(1 + 0.2 \left(\frac{V_{T_i}}{a} \right)^2 \right)^{7/2} - 1, \tag{6.14}$$

$$a = a_{SL} \sqrt{\frac{T_a}{T_{SL}}}, \tag{6.15}$$

a_{SL} is the speed of sound at sea level on a standard day [6], and T_{SL} is the standard temperature at sea level. It is important to note (6.13) and (6.14) take different forms for supersonic v_T and v_{T_i}, which can be found in [33].

Another class of calibration methods [26][41][50][80], use recursive estimation techniques such as the KF [59] in order to converge onto calibration parameters of interest. In [26], a so-called scale factor, γ_v, which is assumed to be constant for the entire airspeed (or Mach number) domain such that

$$V_T = \gamma_v V_{T_i}, \tag{6.16}$$

is estimated using DGPS measurements, and often estimating AoA and AoS simultaneously. These recursive methods along with the angle of sideslip estimation simulations in [69] provided the baseline foundation for the proposed solution due to their use of the KF and non-linear regression [8][75]. However, they were limited in the fact that the constant scale factor assumption is only valid for small aircraft (namely unmanned vehicles) with a limited Mach number domain, as shown in Section 6.4. Additionally, their use of a KF was limited to static airspeed conditions similar to the methods in [51][72][80], requiring once again the need to repeat the experiment at multiple Mach number conditions in order to sample the underlying function.

6.2.3 External Reference Methods.

One of the most accurate methods for estimating SPE is the pressure survey method [42]. In this technique, a weather balloon capable of measuring T_a, H_c, and P_a is launched into the local airmass. The balloon measurements of P_a can then directly be used to compute SPE using (6.1). Obviously, this method provides the most accurate results since it directly measures the desired error. However, it is rarely used unless experimental budgets are amenable due to its cost and associated logistical footprint. Besides the financial and logistical complications, the survey method is also limited by the assumed constant atmospheric properties between the balloon launch site and the area where the experimental aircraft collects its data. This assumption also limits the ability to perform this technique in an online fashion since the truth data needed for calibration is only available and/or valid for a limited time and geographical region. Similarly, in the Pacer method [33], an aircraft that has been previously calibrated can also be used as an external reference when flown alongside the uncalibrated aircraft. The benefits of such a method include the ability to compare both altitude and airspeed simultaneously, model a large portion of the Mach number domain in a single experiment (if a level acceleration is performed), and model supersonic airspeeds at safe altitudes. However, much like the pressure survey method, the

Pacer method suffers similar logistical footprint issues since it may be difficult to schedule a calibrated aircraft with a similar performance envelope as the aircraft to be calibrated. Additionally, any errors incurred during the calibration of the pacer aircraft will be directly transferred into the calibration of the aircraft in question.

6.2.4 Contributions.

Having explored the underlying characteristics of the SPE problem and the state-of-the-art solutions, we now turn to the proposed algorithm, henceforth referred to as Jurado-McGehee Online Self-Survey (JMOSS), and its specific contributions. The JMOSS algorithm provides a drastic improvement over all other methods in that it:

(a) Utilizes a hybrid pressure-airspeed-altitude algorithm inside a Backwards Smoothing Extended Kalman Filter (BSEKF) framework to estimate ΔP_p and K_t in an online fashion, without the need for multiple experiments or external truth sources.

(b) Develops an autonomous information-theory-based spline smoothing process, referred to as the Akaike Spline Model (ASM), which balances model complexity with error reduction, and captures transonic and supersonic effects with no prior knowledge of the $\Delta P_p(M_{ic})$ functional form.

(c) Enables full Mach number domain characterization including transonic and supersonic effects using a single experiment, without the need to sustain supersonic speeds.

6.2.5 Outline.

The remainder of this chapter is organized into three additional sections. Section 6.3 develops the flying and data processing algorithms that enable the research advancements proposed herein. Section 6.4 presents results from a T-38C flight test program comparing the proposed algorithm against state-of-the-art airspeed, altitude, and pressure methods, using weather balloon pressure survey data as the reference truth. Finally, Section 6.5 summarizes the research effort, and presents conclusions and future work.

6.3 Methodology

This section describes the flight and data processing algorithms developed during this research, which enable the specific contributions previously outlined. Figure 6.3 illustrates the information flow from required input data to BSEKF output and subsequent ASM estimation. The specific methods used are described in the following sections.

6.3.1 Flight Technique.

The flight technique needed to meet observability requirements is based on [51]. However, it was found that SPE estimates tend to become noisy during non-level flight, most likely due to dynamic changes in AoA and AoS during turns. As such, the flight technique for the proposed algorithm was modified to meet the observability requirements for wind estimation and Mach number dependency by separating them into three distinct phases:

(i) A constant-altitude deceleration from M_{max} to M_{turn},

(ii) A constant altitude, constant airspeed, 360-degree turn at M_{turn},

(iii) A constant-altitude deceleration from M_{turn} to M_{min},

where M_{max} is the aircraft's maximum Mach number, M_{min} is the minimum Mach number, and M_{turn} is an arbitrary constant turning speed within that domain. The three-phase design of the flight technique provided three main efficiencies in the context of data collection. First, it enabled collection of flight test data across the entire Mach number domain in a single experiment, without the need for sustained supersonic conditions. Next, the constant-Mach turn phase allowed for wind observability. Finally, the decoupling of turning and deceleration allowed for collection of non-corrupted SPE data while still meeting wind observability requirements. Several experiments were conducted in order to establish repeatability and algorithm stability. Table 6.1 summarizes the actual values for the above conditions that were used for experimental data collection. Additionally, the experiments

Table 6.1: Summary of flight conditions for JMOSS experiments.

	Conditions				
Experiment	H_{ic} [ft. PA]	M_{max}	M_{turn}	M_{min}	**Duration [min]**
1	18.27K	1.06	0.54	0.52	7.40
2	19.95K	1.05	0.64	0.54	6.81
3	18.82K	1.06	0.73	0.53	8.95
4	21.23K	1.05	0.92	0.54	8.32

Table 6.2: Summary of flight conditions for comparison methods.

Method	**Points**	**Domain [M]**	**Duration**
Level Turn	10	0.53-0.92	53.57
Cloverleaf	9	0.52-0.94	56.69
Tower Fly-by	10	0.54-0.90	42.00

summarized in Table 6.2 were performed in order to collect data for later comparison against the proposed method.

6.3.2 Required Data.

One of the key enabling technologies provided in this chapter is the development of a hybrid pressure-airspeed-altitude method using a BSEKF. As such, pressure readings (P_T and P_s) from the ADS are used directly, instead of indirectly via H_{ic} or V_{ic}. As previously discussed, T_{ic} is required in order to estimate T_a, which is done inside the algorithm, eliminating the need for an external temperature calibration. Next, AoA and AoS from the ADS, along with aircraft body angles from the IMU are required to compute the necessary

Table 6.3: Required data parameters for JMOSS algorithm.

Name	Symbol	Source
Static pressure	P_s	ADS
Total pressure	P_T	
Total temperature	T_{ic}	
Indicated AoA	α_i	
Indicated AoS	β_i	
Roll angle	Φ	IMU
Pitch angle	Θ	
Yaw angle	Ψ	
North ground speed	v_N	GPS
East ground speed	v_E	
Down ground speed	v_D	
Geometric altitude	h_g	

DCMs to transform vectors from the wind frame (w-frame) to the n-frame. Finally, GPS velocity and altitude measurements are required to compute flight path angle (a parameter needed in AoA and AoS correction), and provide measurement updates to the BSEKF.

6.3.3 AoA and AoS Corrections.

Begin by correcting indicated AoA, α_i, for upwash errors, which are a function of Mach number [107], using

$$\alpha_c = \alpha_i + \Delta\alpha(M_{ic}), \tag{6.17}$$

$$\Delta\alpha(M_{ic}) = \hat{\beta}_0 + \hat{\beta}_1 M_{ic} + \hat{\beta}_2 M_{ic}^2, \tag{6.18}$$

where M_{ic} is derived from P_s and P_T using standard Pitot-static equations [33], α_c is corrected AoA, α_i is indicated AoA, and the function $\Delta\alpha(M_{ic})$ is given by the second-order polynomial model

$$\begin{bmatrix} \Theta_1 - \gamma_1 - \alpha_{i_1} \\ \vdots \\ \Theta_S - \gamma_S - \alpha_{i_S} \end{bmatrix} = \begin{bmatrix} 1 & M_{ic_1} & M_{ic_1}^2 \\ \vdots & \vdots & \vdots \\ 1 & M_{ic_S} & M_{ic_S}^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}, \tag{6.19}$$

where Θ is pitch angle, S is the number of measurements, γ is the flight path angle given by

$$\gamma = \arcsin\left(\frac{-v_D}{v_T}\right), \tag{6.20}$$

which can be approximated using the ground velocity vector, $\mathbf{v}_g$, by

$$\gamma \approx \arcsin\left(\frac{-v_D}{\left\|\mathbf{v}_g\right\|}\right), \tag{6.21}$$

assuming V_T is much larger than the wind speed. Finally, AoS is corrected by projecting β_i onto the corrected w-frame using

$$\beta_c = \arctan\left(\cos(\alpha_c)\tan(\beta_i)\right). \tag{6.22}$$

6.3.4 Ambient Temperature Optimization.

Prior to processing the data using the BSEKF, an estimate of ambient temperature is obtained by minimizing the least-squares [75] cost function given by

$$\min_{b_1,b_2,b_3} \quad C(b_1,b_2,b_3) = \sum_{s=1}^{S} \left[T_{ic_s} - \hat{T}_{ic_s}\right]^2, \tag{6.23}$$

where

$$\hat{T}_{ic_s} = \hat{T}_{a_s}\left(1 + 0.2\hat{K}_{t_s} M_{ic_s}^2\right), \tag{6.24}$$

$$\hat{T}_{a_s} = T_{SL} f_{\theta_H}(h_{g_s}) + b_1, \tag{6.25}$$

$$\hat{K}_{t_s} = b_2 + b_3 M_{ic_s}^2, \tag{6.26}$$

and the function f_{θ_H} is given in [33]. Essentially, minimizing (6.23) leads to optimal coefficients b_1, b_2, b_3 that best approximate the actual T_{ic} measurements while constraining $\hat{T}_a$ to follow the standard temperature profile given by h_g, plus a constant bias, and K_t to depend on M_{ic}^2. Once converged, the resulting optimal estimates of $\hat{T}_a$ are used as control inputs in the BSEKF, where a better estimate of K_t, based on M_{pc} is produced alongside the other variables of interest.

6.3.5 BSEKF Implementation.

Using the notations described in [76], the main estimation engine of the algorithm is driven by a six-state BSEKF [59][76][77][87] with system dynamics defined by

$$\dot{\mathbf{x}}(t) = \mathbf{G}(t)\mathbf{w}(t), \tag{6.27}$$

$$\mathbf{x} = \begin{bmatrix} \Delta P_p & v_{W_N} & v_{W_E} & v_{W_D} & K_t & P_0 \end{bmatrix}^{\mathrm{T}}, \tag{6.28}$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{6.29}$$

where ΔP_p is SPE, v_{W_N}, v_{W_E}, and v_{W_D} are the wind components, K_t is the temperature recovery factor, P_0 is an ambient pressure about which the relationship between geometric altitude and pressure altitude is linearized, and $\mathbf{w}(t)$ is a bivariate Gaussian white-noise process with

$$E\left[\mathbf{w}(t)\right] = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{6.30}$$

$$E\left[\mathbf{w}(t)\mathbf{w}(t+\tau)^{\mathrm{T}}\right] = 0.1 \begin{bmatrix} \delta(\tau) & 0 \\ 0 & \delta(\tau) \end{bmatrix}. \tag{6.31}$$

The BSEKF discrete measurement model at time k is defined by

$$\mathbf{z}_k = \mathbf{h}[\mathbf{x}_k, \mathbf{u}_k] + \mathbf{v}_k, \tag{6.32}$$

$$\mathbf{u} = \begin{bmatrix} P_s & P_T & \alpha_c & \beta_c & \Phi & \Theta & \Psi & \bar{h}_g & \hat{T}_a \end{bmatrix}^{\mathrm{T}}, \tag{6.33}$$

where $\bar{h}_g$ is the mean geometric altitude, and the vector $\mathbf{v}_k$ is composed of five independent Gaussian white-noise processes with

$$E\left[\mathbf{v}_k\right] = \underset{5\times1}{\mathbf{0}}, \tag{6.34}$$

$$E\left[\mathbf{v}_k\mathbf{v}_l^{\mathrm{T}}\right] = \underset{5\times5}{\mathbf{I}}\,\delta_{kl}. \tag{6.35}$$

The nonlinear measurement function, $\mathbf{h}$, in (6.32) estimates incoming GPS groundspeed and altitude measurements, as well as total temperature measurements, to form the vector

$$\hat{\mathbf{z}}_k = \begin{bmatrix} \hat{\mathbf{v}}_T^n + \hat{\mathbf{v}}_W & \hat{h}_g & \hat{T}_{ic} \end{bmatrix}^{\mathrm{T}} \tag{6.36}$$

$$= \begin{bmatrix} \hat{v}_N & \hat{v}_E & \hat{v}_D & \hat{h}_g & \hat{T}_{ic} \end{bmatrix}^{\mathrm{T}}, \tag{6.37}$$

and is constructed from standard Pitot-static equations [33] using

$$\hat{P}_a = P_s - \hat{\Delta}P_p, \tag{6.38}$$

$$\hat{M}_{pc} = f(\hat{P}_a, P_T), \tag{6.39}$$

$$\hat{T}_{ic} = \hat{T}_a\left(1 + 0.2\hat{K}_t\hat{M}_{pc}^2\right), \tag{6.40}$$

$$\hat{a} = a_{SL}\sqrt{\frac{\hat{T}_a}{T_{SL}}}, \tag{6.41}$$

$$\hat{v}_T = \hat{M}_{pc}\hat{a}, \tag{6.42}$$

$$\hat{\mathbf{v}}_T^w = \begin{bmatrix} \hat{v}_T & 0 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{6.43}$$

$$\hat{\mathbf{v}}_T^n = \mathbf{C}_b^n\mathbf{C}_w^b\hat{\mathbf{v}}_T^w, \tag{6.44}$$

where the function in 6.39 is given in [33], the DCMs $\mathbf{C}_b^n$, and $\mathbf{C}_w^b$ are created using the frame transformations in [35] given by

$$\mathbf{C}_w^b = \begin{bmatrix} \cos(\alpha_c)\cos(\beta_c) & -\cos(\alpha_c)\sin(\beta_c) & -\sin(\alpha_c) \\ \sin(\beta_c) & \cos(\beta_c) & 0 \\ \sin(\alpha_c)\cos(\beta_c) & -\sin(\alpha_c)\sin(\beta_c) & \cos(\alpha_c) \end{bmatrix}, \tag{6.45}$$

$$\mathbf{C}_b^n = \begin{bmatrix} \cos(\Theta)\cos(\Psi) & \cos(\Psi)\sin(\Theta)\sin(\Phi) - \cos(\Phi)\sin(\Psi) & \sin(\Phi)\sin(\Psi) + \cos(\Phi)\cos(\Psi)\sin(\Theta) \\ \cos(\Theta)\sin(\Psi) & \cos(\Phi)\cos(\Psi) + \sin(\Theta)\sin(\Phi)\sin(\Psi) & \cos(\Phi)\sin(\Theta)\sin(\Psi) - \cos(\Psi)\sin(\Phi) \\ -\sin(\Theta) & \cos(\Theta)\sin(\Phi) & \cos(\Theta)\cos(\Phi) \end{bmatrix}, \tag{6.46}$$

the estimated wind vector, $\hat{\mathbf{v}}_W$, is given by

$$\hat{\mathbf{v}}_W = \begin{bmatrix} \hat{v}_{W_N} & \hat{v}_{W_E} & \hat{v}_{W_D} \end{bmatrix}^{\mathrm{T}}, \tag{6.47}$$

the estimated geometric altitude measurement, $\hat{h}_g$, is given by

$$\hat{\delta} = \frac{\hat{P}_a}{P_{SL}}, \quad \hat{\delta}_0 = \frac{\hat{P}_0}{P_{SL}}, \tag{6.48}$$

$$\hat{H}_c = f_{H_\delta}(\hat{\delta}), \quad \hat{H}_{c_0} = f_{H_\delta}(\hat{\delta}_0), \tag{6.49}$$

$$\hat{\theta} = f_{\theta_H}(\hat{H}_c), \tag{6.50}$$

$$T_{std} = T_{SL}\hat{\theta}, \tag{6.51}$$

$$\hat{h}_g = \bar{h}_g + \frac{\hat{T}_a}{T_{std}}\left(\hat{H}_c - \hat{H}_{c_0}\right), \tag{6.52}$$

P_{SL} is sea-level standard pressure, and the functions f_{H_δ} and f_{θ_H} are given in [33].

In order to enable online estimation, the BSEKF is initialized with no prior knowledge of the system states. As such, all initial estimates are set to zero, with the exception of $\hat{K}_t$, which is set to 1. Additionally, the initial state estimation covariance matrix is set to a 6×6 identity matrix. Since the turn data ($M_{ic} = M_{turn}$) is not necessarily collected at $t_k = 0$, the BSEKF is processed "forward" from $t_k = 0$ to $t_k = (M - 1)\Delta t$, where Δt is the sampling period, in order to converge onto accurate estimates of the wind states and corresponding

$\hat{\Delta}P_p$, K_t, and P_0. Next, the resulting final state estimates from the forward run are used as initial estimates for the BSEKF smoothing run from $t_k = (M-1)\Delta t$ to $t_k = 0$ in order to smooth any biased $\hat{\Delta}P_p$, K_t, and δP_0 estimates that occurred on the forward run prior to $M_{ic} = M_{turn}$.

6.3.6 Akaike Spline Model.

Having obtained the estimates from the BSEKF, one may choose to fit a model to the SPE observations with respect to Mach number in a number of ways. In this research, the resulting BSEKF estimates of SPE, are modeled as a function of M_{ic} using a novel linear smoothing spline model referred to as ASM. The ASM algorithm is crucial in smoothing BSEKF output with no prior knowledge of the functional relationship between $\Delta P_p/P_s$ and M_{ic} for the ADS being calibrated. Additionally, it allows for the accurate modeling of unknown changes to the functional form in the transonic and supersonic regions as shown in Section 6.4.

Algorithm 6.1 illustrates a pseudocode implementation of the ASM process. ASM smoothing begins with a simple second-order model of the form

$$\underbrace{\begin{bmatrix} \frac{\hat{\Delta}P_{p_1}}{P_{s_1}} \\ \vdots \\ \frac{\hat{\Delta}P_{p_S}}{P_{s_S}} \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} 1 & M_{ic_1} & M_{ic_1}^2 \\ \vdots & \vdots & \vdots \\ 1 & M_{ic_S} & M_{ic_S}^2 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}}_{\boldsymbol{\beta}}, \tag{6.53}$$

where S is the number of measurements in the experiment. Next, a simple optimization routine is executed to sequentially add smoothing spline knots using

$$\underbrace{\begin{bmatrix} \frac{\hat{\Delta}P_{p_1}}{P_{s_1}} \\ \vdots \\ \frac{\hat{\Delta}P_{p_S}}{P_{s_S}} \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} 1 & M_{ic_1} & M_{ic_1}^2 & (M_{ic_1}-s_1)_+^2 & \dots & (M_{ic_1}-s_P)_+^2 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & M_{ic_S} & M_{ic_S}^2 & (M_{ic_S}-s_1)_+^2 & \dots & (M_{ic_S}-s_P)_+^2 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{P+3} \end{bmatrix}}_{\boldsymbol{\beta}}, \tag{6.54}$$

where each s_p, $p = 1, \dots, P$, referred to as a knot, is a preselected inflection point along the Mach number domain, and the operator $()_+$ denotes negative values of its argument are set to zero, which is equivalent to multiplying by the Heaviside function centered at the knot location. The optimization is based on minimizing the resulting AIC$_c$ value [2], which balances error reduction with model complexity. At each increment, a single spline knot is added to the linear model (6.53), at a location within the M_{ic} domain based on statistical quantiles [89]. Then, the resulting AIC$_c$ model criterion is compared to its previous value to verify that at least a one-percent decrease in AIC$_c$ was achieved by the additional knot. If at any point this criterion is not met, the optimization is considered complete and the spline model is finalized. Finally, if the particular experiment contained supersonic data (i.e., $M_{ic} > 1$), an additional seven knots are automatically added evenly between $M_{ic} = 0.93$ and $M_{ic} = 1.00$ in order to capture any potential drastic changes to the functional relation in the transonic and supersonic regions. Once completed, the resulting model inferences such as Prediction Interval (PI) were computed using [68].

6.4 Results

Figures 6.4 and 6.5 illustrate the state estimation histories of the forward and backward BSEKF passes for a single JMOSS experiment, respectively. As shown in Fig. 6.4, the BSEKF states are unobservable (and inaccurate) during forward pass from $M_{ic} = M_{max}$ until the turn is executed, and begin to converge after $M_{ic} <= M_{turn}$ as the measurements are processed from M_{max} to M_{min}. Using the final estimates of the forward pass (i.e., when $M_{ic} = M_{min}$) as initial estimates for the backward smoothing pass produced stable and accurate estimates of all six states as shown in Fig. 6.5.

Figures 6.6 and 6.7 illustrate the results from a single JMOSS experiment, and all JMOSS experiments combined, respectively. As shown in Fig. 6.6, a single JMOSS experiment yielded accurate results across the entire Mach number domain, with no need for external sources or prior knowledge, while simultaneously calibrating the temperature recovery factor, K_t. As shown in Fig. 6.7, combining the BSEKF results from all four

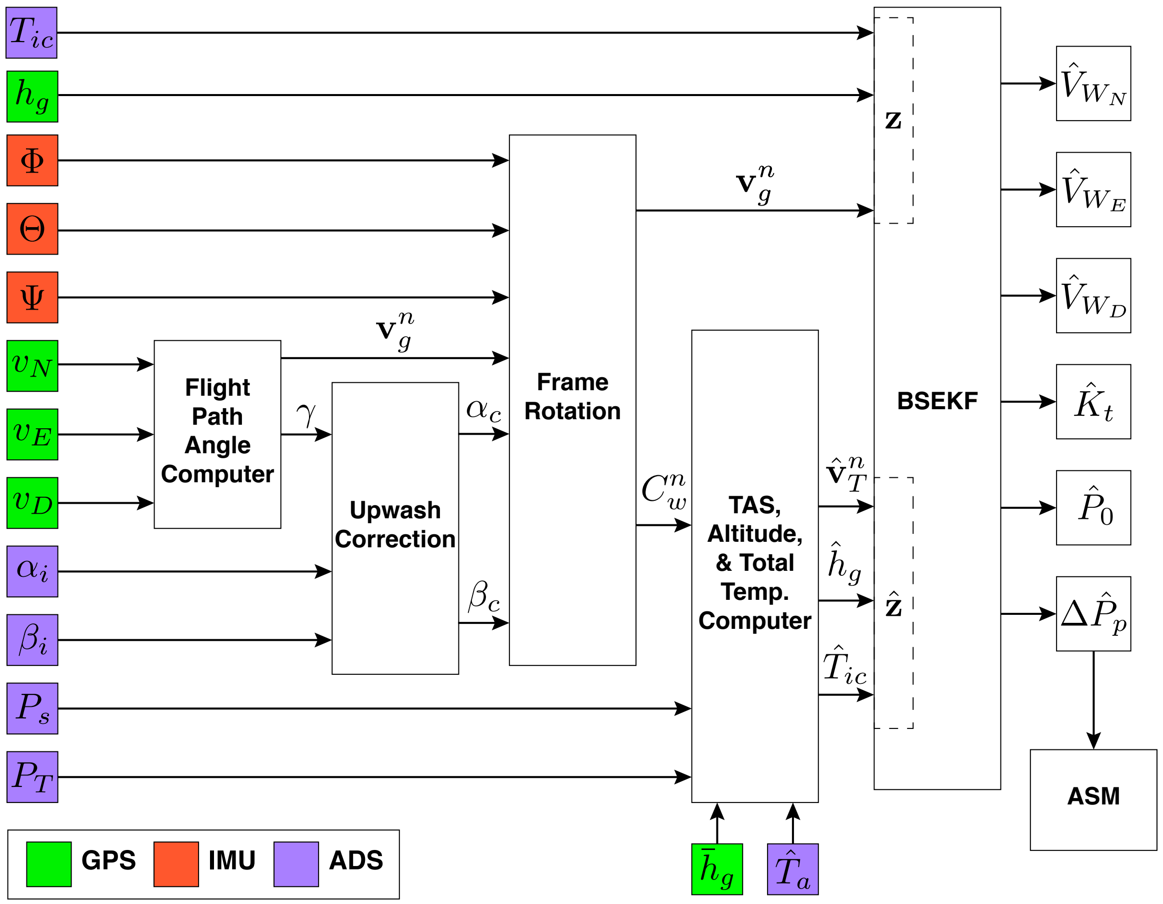


Figure 6.3: Data processing flow for JMOSS algorithm.

Algorithm 6.1 ASM = fitASM($\mathbf{m}_{ic}$, $\hat{\Delta}\mathbf{P}_p/\mathbf{P}_s$)

```
Input: $\hat{\Delta}\mathbf{P}_p/\mathbf{P}_s, \mathbf{m}_{ic}$     ▶ Inputs are BSEKF output for SPE, and computed $M_{ic}$
 1: $\mathbf{y} \leftarrow \hat{\Delta}\mathbf{P}_P/\mathbf{P}_s$     ▶ Create observation vector
 2: if max($\mathbf{m}_{ic}$) > 1 then     ▶ If supersonic data present, add supersonic knots
    $M_{ic} \in (0.93, 1)$
 3:     superSonic←true
 4: end if
 5: $P \leftarrow 0$, go←true     ▶ Initialize loop with $P = 0$ knots
 6: while go do
 7:     $\mathbf{X} \leftarrow$ createSplineRegressor($P$, $\mathbf{m}_{ic}$,superSonic)     ▶ Use Eq. (6.54) to create $\mathbf{X}$
    based on $P$
 8:     $\hat{\boldsymbol{\beta}}$, AIC$_c(P+3) \leftarrow \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$     ▶ Compute $P+3$ total coefficients and AIC$_c$
 9:     if $P = 0$ then
10:         $\mathbf{X}_{prev}, \hat{\boldsymbol{\beta}}_{prev}$, AIC$_{c_{prev}} \leftarrow \mathbf{X}, \hat{\boldsymbol{\beta}}$, AIC$_c$     ▶ First time in the loop, add a knot
11:         $P \leftarrow P + 1$
12:     else
13:         if AIC$_c$ < 1.01×AIC$_{c_{prev}}$ then     ▶ If AIC$_c$ is decreased by 1%, add a knot
14:             $\mathbf{X}_{prev}, \hat{\boldsymbol{\beta}}_{prev}$, AIC$_{c_{prev}} \leftarrow \mathbf{X}, \hat{\boldsymbol{\beta}}$, AIC$_c$
15:             $P \leftarrow P + 1$
16:         else
17:             $\mathbf{X}, \hat{\boldsymbol{\beta}} \leftarrow \mathbf{X}_{prev}, \hat{\boldsymbol{\beta}}_{prev}$     ▶ Otherwise, stop the loop
18:             go←false
19:         end if
20:     end if
21: end while

Output: ASM
    ASM.Model← $\hat{\boldsymbol{\beta}}$
    ASM.deltaPp_Ps← $\mathbf{X}\hat{\boldsymbol{\beta}}$
    ASM.machIC← $\mathbf{m}_{ic}$
```

experiments slightly increased the associated model PI, due to the variation in BSEKF estimates across experiments, but increased accuracy when compared to the survey truth data.

Figures 6.8 through 6.10 illustrate the calibration results from Level Turn, Cloverleaf, and TFB techniques. As previously mentioned, these methods required varying levels of logistical footprints, were limited in their Mach number domain, and/or required prior knowledge of K_t. As shown, all methods yielded results that closely followed survey truth data, with varying levels of bias and PI widths. It is important to note the data reduction for the Level Turn and Cloverleaf methods included the enhancements that were developed as part of the JMOSS algorithm (i.e., AoA and AoS corrections, direct computations of airspeeds using pressures, and three-dimensional reference frame rotations), which may have contributed to their accuracy.

Finally, Table 6.4 compares effort metrics across all methods tested during this research. To highlight the true potential in efficiency from the JMOSS algorithm, only the results from a single experiment were considered. The time figures were computed by summing all flight time dedicated to collecting data for each experiment. The cost figures are directly proportional to T-38C flight time at a representative flight test rate of \$11.3K/hr. Meanwhile, Δ Mach captures the difference between minimum and maximum Mach number modeled by each experiment. Finally, the mean bias was taken as the average difference between each method's results and survey truth data, contained within the bounds of each method's Mach number domain, and normalized by that width (i.e., divided by Δ Mach). As shown, the JMOSS algorithm was able to produce accurate results with as much as 90% fewer test points, 88% less time/cost, 83% less bias, and 78% less uncertainty, all while modeling 42% more Mach number domain.

Table 6.4: Metric comparison for ADS calibration algorithms.

	JMOSS	**Level Turn**	**Cloverleaf**	**TFB**
Points	1	10	9	10
Time [min]	6.81	53.57	56.69	42.00
Cost [USD]	1.3K	10.1K	10.7K	7.9K
Min Mach	0.54	0.53	0.52	0.54
Max Mach	1.05	0.92	0.94	0.90
Δ Mach	0.51	0.39	0.42	0.36
Mean Bias	-7.75×10^{-4}	1.89×10^{-3}	4.61×10^{-3}	-8.85×10^{-4}
95% PI	$\pm 1.59 \times 10^{-3}$	$\pm 7.23 \times 10^{-3}$	$\pm 1.33 \times 10^{-3}$	$\pm 3.82 \times 10^{-4}$

6.5 Chapter Summary and Future Work

This chapter has introduced a fully self-contained, pressure-airspeed-altitude hybrid BSEKF-based ADS calibration algorithm with an accompanying autonomous smoothing spline process rooted in information theory. As shown in the previous sections, the proposed algorithm models a larger portion of the Mach number domain while drastically reducing the cost, flight time, mean error, and maximum width of the 95% prediction intervals around the resulting model. Additionally, when experimental data from multiple dates and across varying atmosphere conditions were used, the model proved to have stable, repeatable results. The JMOSS algorithm introduced herein provides a fully automated and self-contained means of establishing an accurate SPE correction curve for any aircraft with no prior knowledge and minimal maneuver requirements for observability. The proposed method sets the course for an emerging class of online calibration and dynamic performance modeling algorithms that not only take advantage of modern data collection capabilities, but also make full use of sensor fusion technology in order to relax the required experimental conditions for such modeling. Future work in this area includes developing more robust post-BSEKF smoothing techniques beyond ASM (e.g., neural networks), identifying potential additional sources of information for sensor fusion, and expanding the concept of sensor-fusion-based online calibration to aircraft performance and flying qualities.

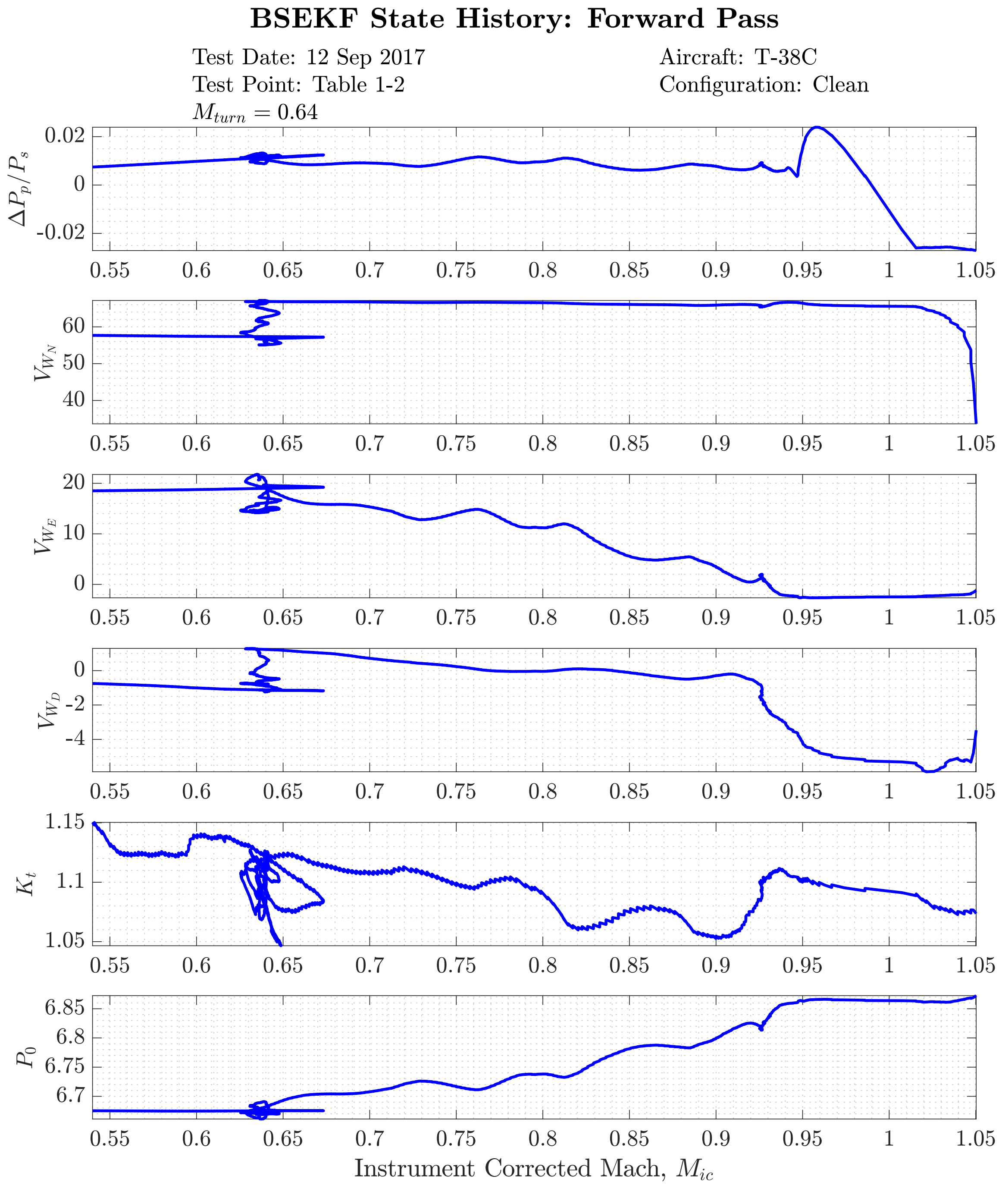


Figure 6.4: Illustration of JMOSS BSEKF output on forward pass.

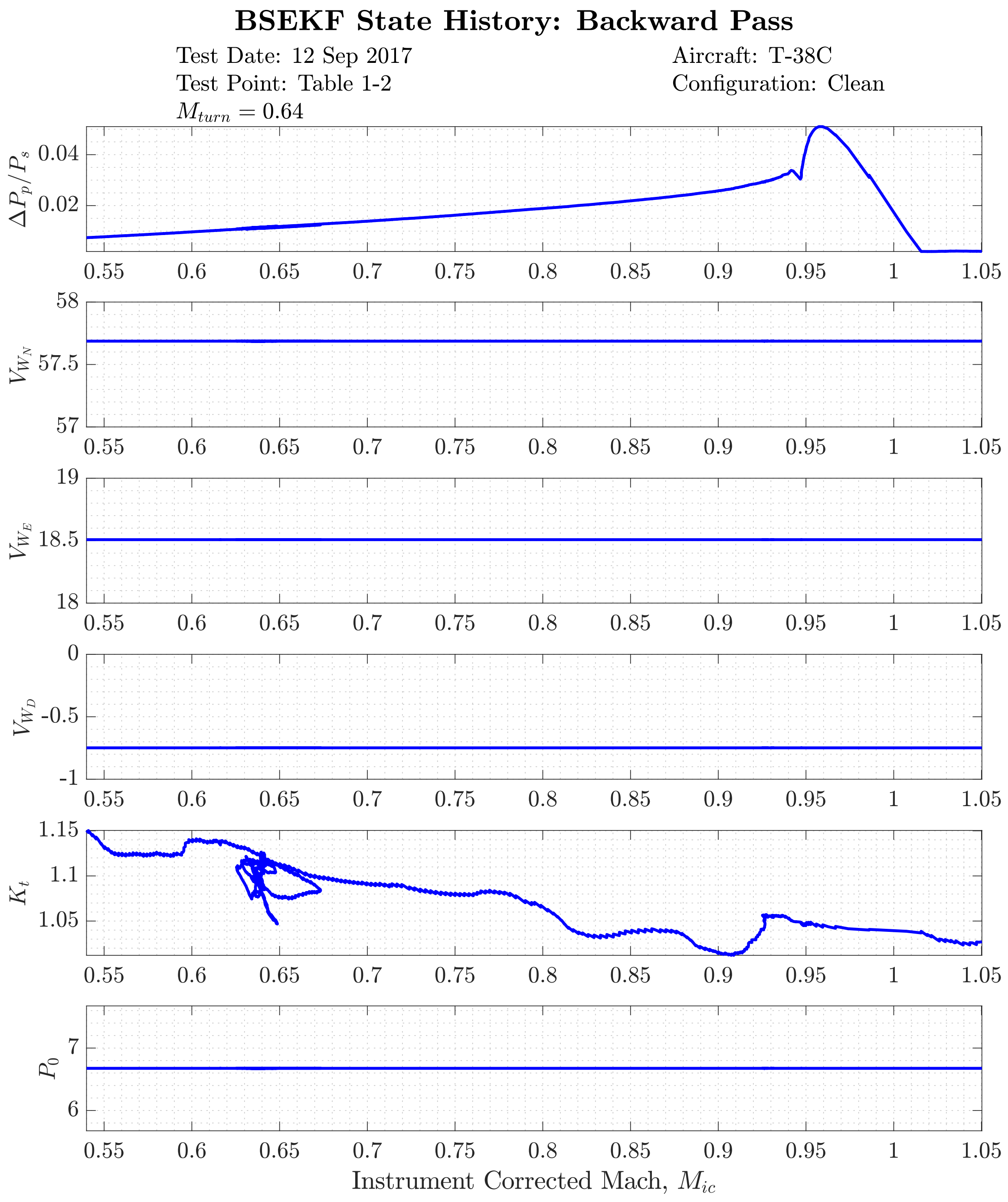
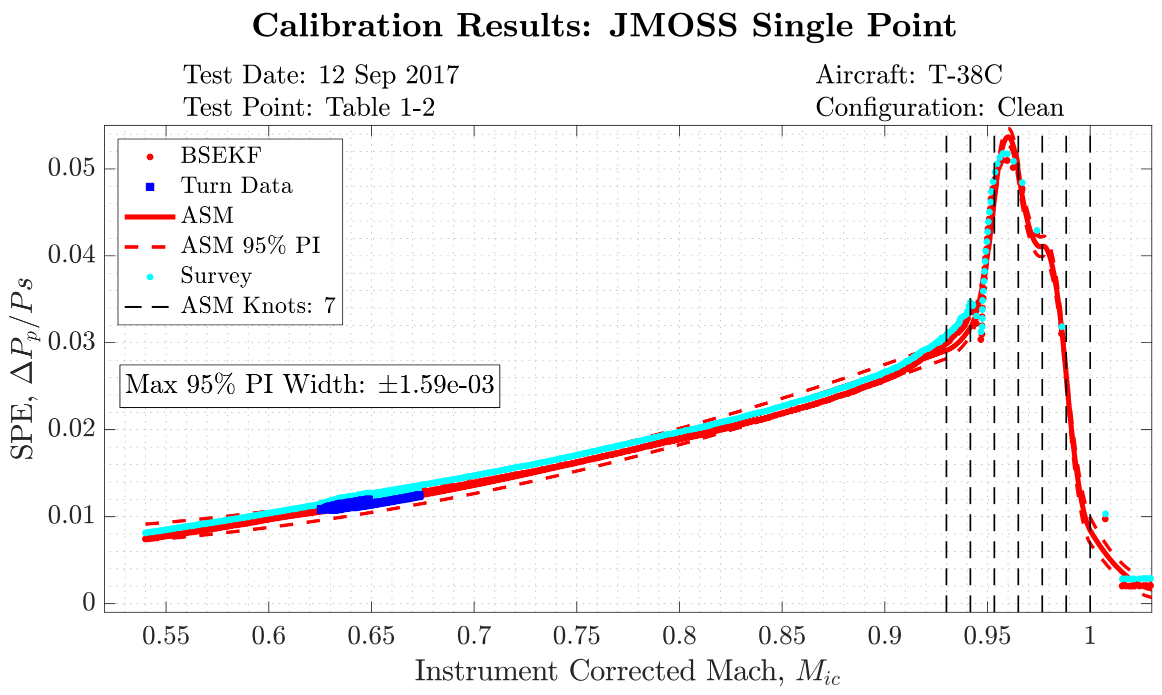


Figure 6.5: Illustration of JMOSS BSEKF output on backward pass.



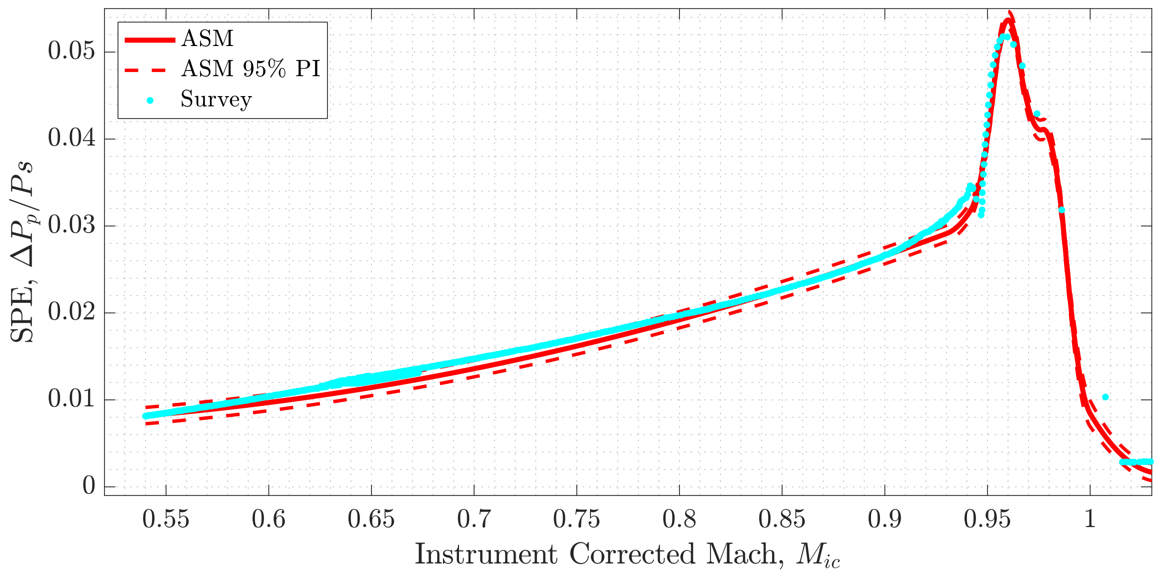
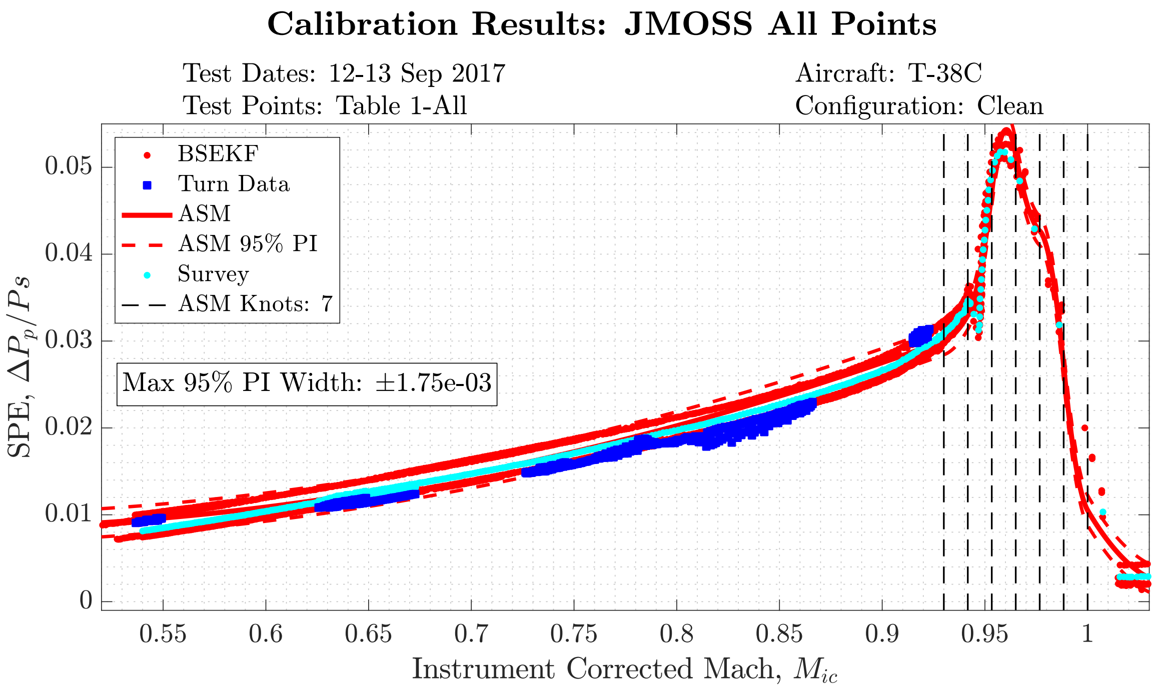


Figure 6.6: JMOSS results for a single test point.



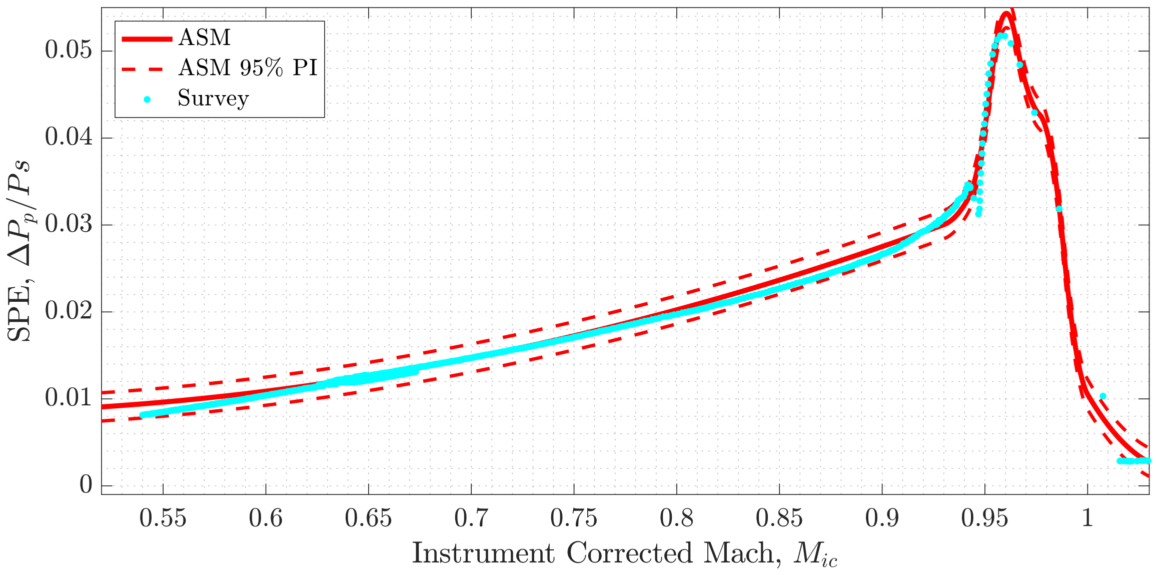
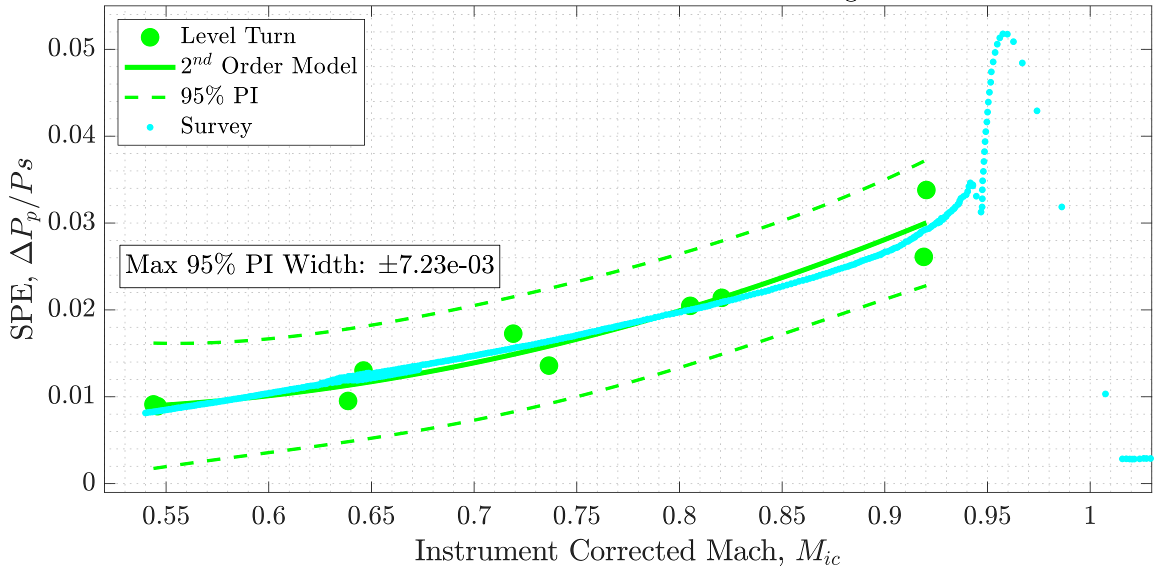


Figure 6.7: JMOSS results when combining all test points.

Calibration Results: Level Turn Data

Test Dates: 6-13 Sep 2017 Aircraft: T-38C
Test Points: Table 2 Configuration: Clean



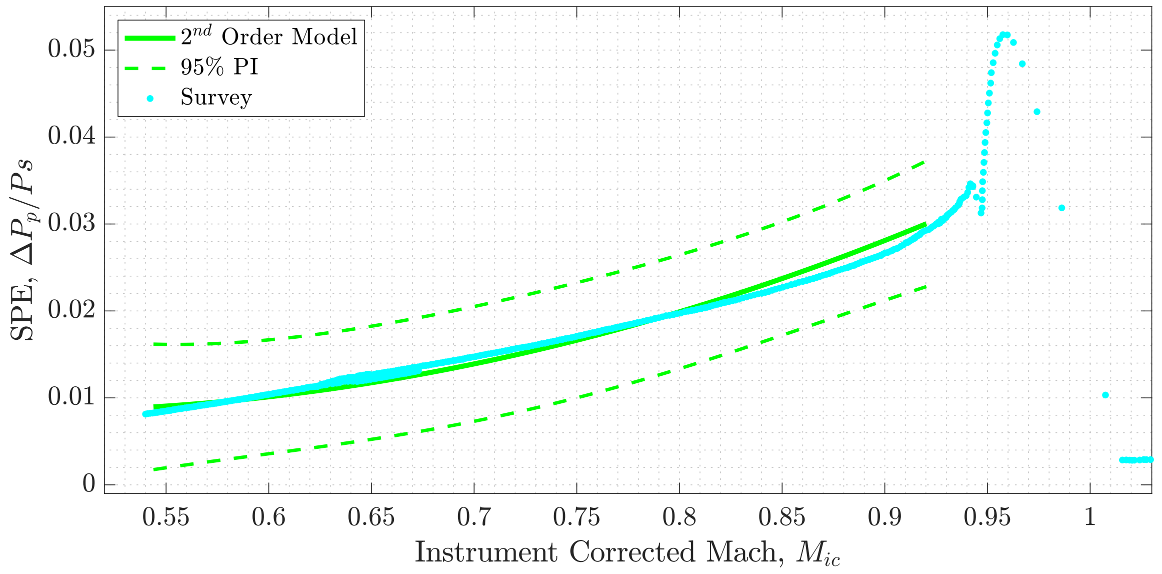
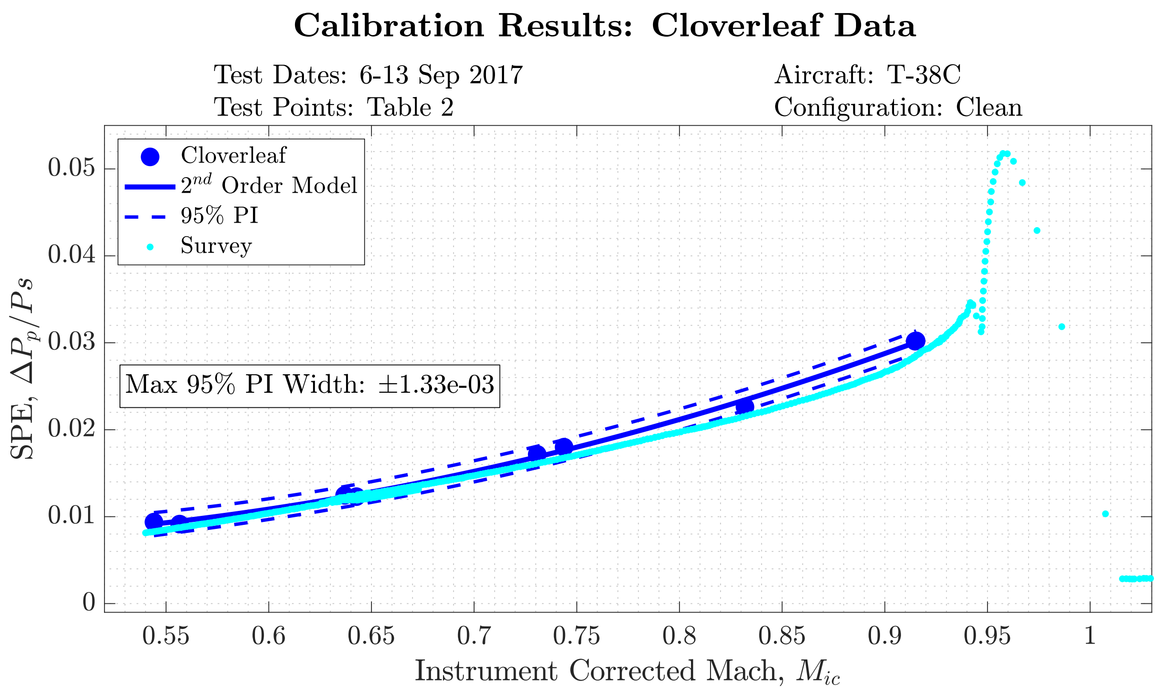


Figure 6.8: Results from Level Turn test points.



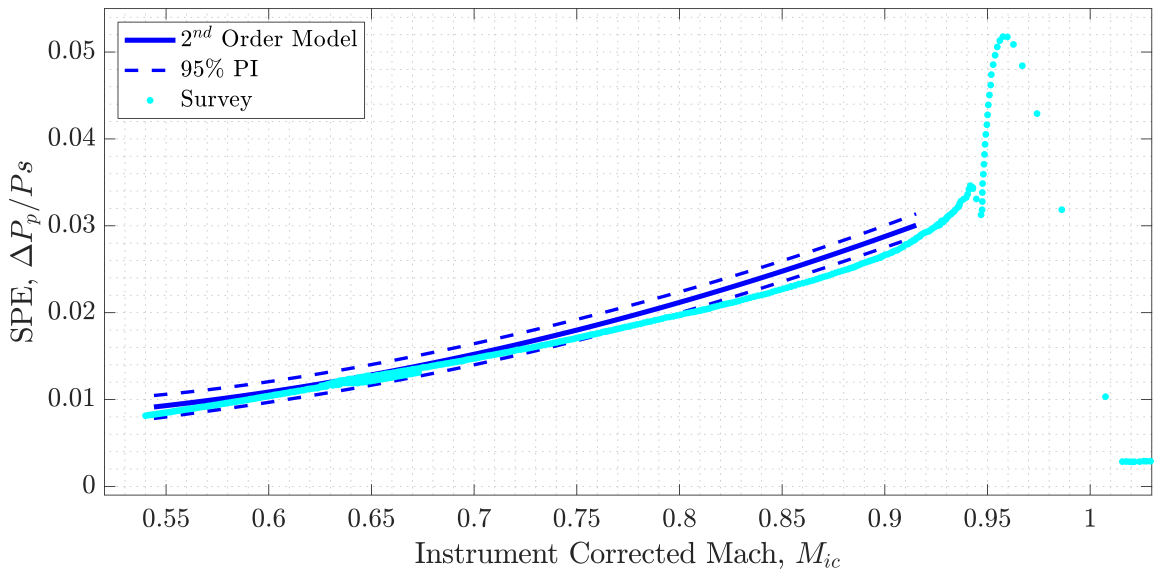


Figure 6.9: Results from Cloverleaf test points.

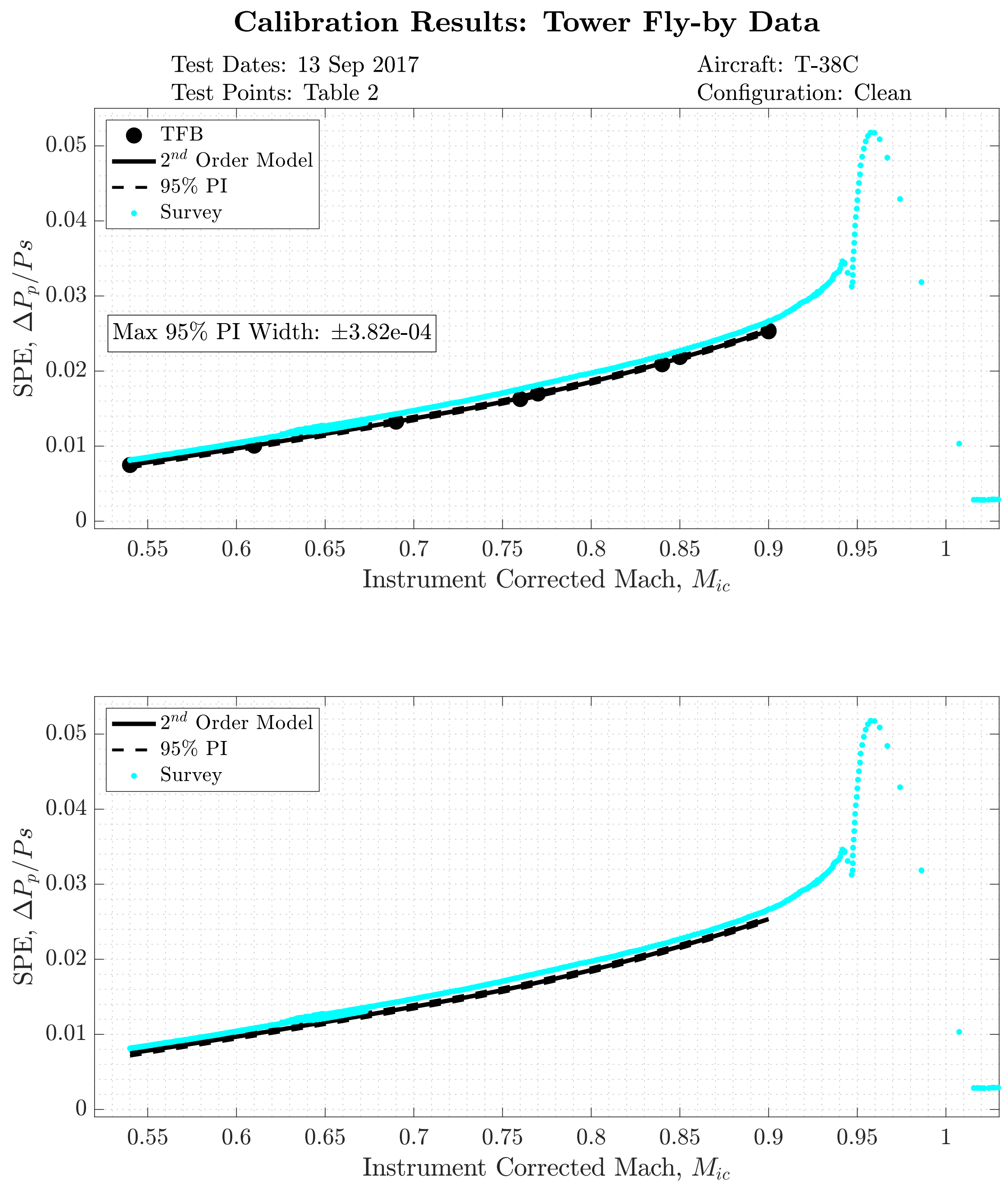
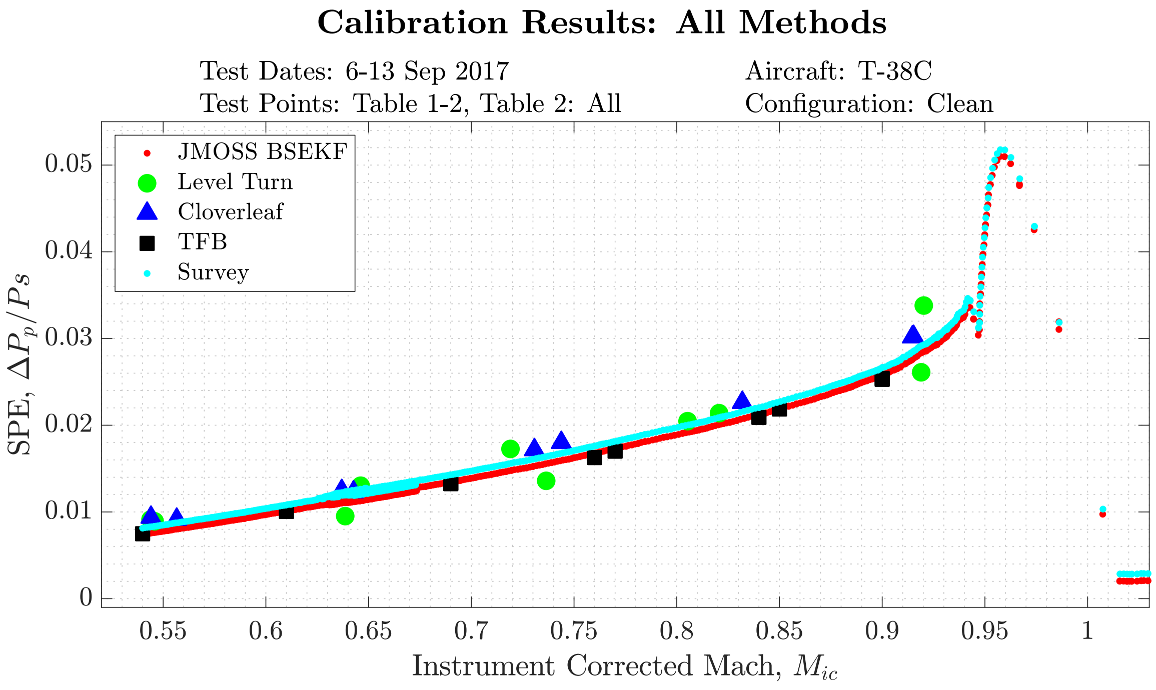


Figure 6.10: Results from Tower Fly-by test points.



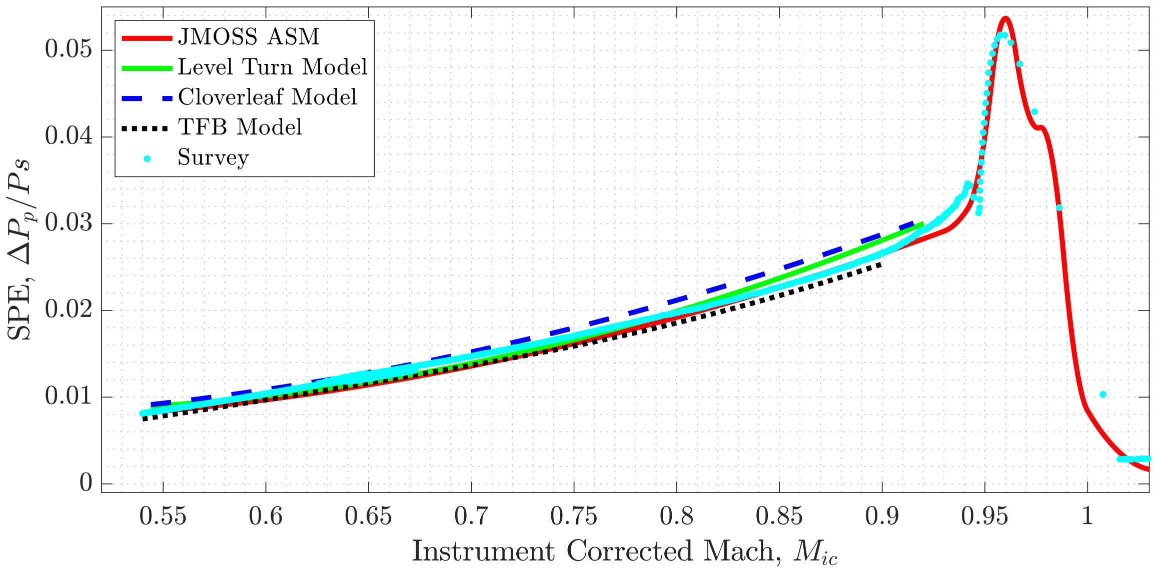


Figure 6.11: Results comparison across all methods.

VII. A Regression-Based Methodology to Improve Estimation of Inertial Sensor Errors Using Allan Variance Data

This chapter proposes a novel, autonomous, regression-based methodology for Allan variance analysis of Inertial Measurement Unit (IMU) sensors, which much like the research provided in Chapter 6, also complements the overall resilient navigation research thrust by contributing a novel autonomous sensor calibration technique suitable for the calibration objective in the ARMAS framework. Current methods for Allan variance analysis have been rooted in the human-based interpretation of linear trends, referred to as the slope method. The slope method is so prolific, it is referenced among electrical and electronics engineering standards for IMU error analysis [46]. However, the graphical nature and visual-inspection based use of the method limits its ability to be programmed as a generalized algorithm, and lacks the autonomy desired in modern-day navigation computations. Using nonlinear regression with a ridge-regression initial guess, the proposed method is shown to produce comparable results as the gold standard slope method when using standard-length data collections, and outperforms the slope method when the amount of available data is limited. This development directly enables accurate navigation solutions for all vehicles in land, air, sea, and space operations. The research developed in this chapter has been published in [53] and [55]. Additionally, at the time of this writing, a collaborative effort based on this research and similar wavelet-based methods is currently in review for publication in [39].

7.1 Introduction

Inertial navigation systems are used to track the location and velocity of an object and are relied upon commonly by many vehicles as a means of establishing orientation in open spaces such as ships in the ocean and airplanes in the sky. However, the availability of an accurate inertial navigation solution depends on the proper calibration of the deterministic and stochastic errors associated with accelerometers and gyroscopes, which compose the IMU. Without proper quantification of their deterministic and stochastic errors, the solutions rendered by the IMU based upon accelerometer and gyroscope measurements are subject to drift and are, at best, erroneous and at worst, provide fatal navigation information. As such, a considerable amount of time and energy has been invested in the understanding and modeling of the various sources of noise that affect the components of the IMU. Adequate modeling of inertial sensor errors begins with an understanding of the physical processes from which deterministic and stochastic errors arise. In general, any given sensor output signal can be written in the form

$$\mathbf{y}_k = \mathbf{M}\mathbf{x}_k + \boldsymbol{\epsilon}_k, \tag{7.1}$$

where $\mathbf{y}_k$ is the measured output signal, $\mathbf{x}_k$ is the true signal, $\mathbf{M}$ is a linear operator on $\mathbf{x}_k$ and $\boldsymbol{\epsilon}_k$ is an additive, possibly non-linear signal composed of a combination of stochastic and deterministic errors, which vary with sensor type. For inertial sensors, the majority of existing research adapts a version of (7.1) to both accelerometers and gyroscopes by providing specific forms of $\mathbf{M}$ and further refining the deterministic components and stochastic processes governing $\boldsymbol{\epsilon}_k$.

In [95], Titterton provides general error models for gyroscopes and accelerometers in order to describe a wide array of deterministic and stochastic errors. For gyroscopes, the relationship between true (ω_x) and measured ($\tilde{\omega}_x$) angular rate for a single axis x is given by

$$\tilde{\omega}_x = (1 + S_x)\omega_x + M_y\omega_y + M_z\omega_z + B_{fx} + B_{gx}a_x + B_{gz}a_z + B_{axz}a_xa_z + \eta_x, \tag{7.2}$$

where S_x is the x-axis scale factor, M_y and M_z are cross coupling coefficients, B_{fx} is a constant x-axis bias (non g-sensitive), B_{gx} and B_{gz} are g-sensitive bias coefficients along the input and spin axes, B_{axz} is the anisoelastic bias coefficient, and η_x is zero-mean additive white Gaussian noise. It is important to note the previosuly discussed terms may be deterministic or stochastic in nature. Most often, such terms are modeled as correlated stochastic processes, which eventually motivates the need for a reliable stochastic characterization method such as Allan variance [4]. The expanded form in (7.2) can be applied to the remaining two axes and expressed in terms of (7.1) by letting

$$\mathbf{y} = \begin{bmatrix} \tilde{\omega}_x \\ \tilde{\omega}_y \\ \tilde{\omega}_z \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{7.3}$$

$$\mathbf{M} = \begin{bmatrix} 1 + S_x & M_y & M_z \\ M_x & 1 + S_y & M_z \\ M_x & M_y & 1 + S_z \end{bmatrix}, \tag{7.4}$$

$$\boldsymbol{\epsilon} = \begin{bmatrix} B_{fx} + B_{gx}a_x + B_{gz}a_z + B_{gxz}a_{xz} + \eta_x \\ B_{fy} + B_{gy}a_y + B_{gx}a_x + B_{gyx}a_{yx} + \eta_y \\ B_{fz} + B_{gz}a_z + B_{gy}a_y + B_{gzy}a_{zy} + \eta_z \end{bmatrix}. \tag{7.5}$$

Similarly, [95] also describes a general error model for accelerometers in terms of the relation between true (a_x) and measured ($\tilde{a}_x$) acceleration for a single axis x as

$$\tilde{a}_x = (1 + S_x)a_x + M_y a_y + M_z a_z + B_f + B_v a_x a_y + \eta_x, \tag{7.6}$$

where S_x is the x-axis scale factor, M_y and M_z are cross coupling coefficients, B_f is a constant measurement bias, B_v is the vibro-pendulus error coefficient, and η_x is zero-mean additive white Gaussian noise. Again, (7.6) can be expressed using the form in (7.1) by letting

$$\mathbf{y} = \begin{bmatrix} \tilde{a}_x \\ \tilde{a}_y \\ \tilde{a}_z \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tag{7.7}$$

$$\mathbf{M} = \begin{bmatrix} 1 + S_x & M_y & M_z \\ M_x & 1 + S_y & M_z \\ M_x & M_y & 1 + S_z \end{bmatrix}, \tag{7.8}$$

$$\boldsymbol{\epsilon} = \begin{bmatrix} B_f + B_v a_x a_y + \eta_x \\ B_f + B_v a_y a_z + \eta_y \\ B_f + B_v a_z a_y + \eta_z \end{bmatrix}. \tag{7.9}$$

Subtle differences in $\mathbf{M}$ and $\boldsymbol{\epsilon}$ are found throughout literature based on the technology used in sensor development (i.e. mechanical, ring laser, etc...). Although such differences affect the specific set of parameters found in $\mathbf{M}$, in general, all models for gyroscopes and accelerometers can be expressed as an adaptation of (7.1), with $\boldsymbol{\epsilon}$ composed of a common mixture of deterministic and stochastic terms. Focusing on such terms, [67] uses a form similar to (7.1) and describes three types of stochastic gyroscopic errors in $\boldsymbol{\epsilon}$ as: "constant bias, uncorrelated white noise, and $1/f$ (flicker) noise." Similar terms appear along with additional sources of error in [45], where a common set of five error sources are modeled using the "Allan variance slope method" [4][32][34]. Although many stochastic modeling and calibration methods have been developed, the "Allan variance slope method" is commonly used in the navigation community, and listed as the method of choice in IMU error analysis standards [46]. As such, this research focuses on improving the mathematical methods for autonomously analyzing Allan variance data in the context of IMU calibration.

7.2 Allan Variance

Having established the importance of properly modeling the sources of noise in $\boldsymbol{\epsilon}$, we now turn to the most commonly used method for doing so found in literature, Allan variance [4]. It is important to note that although very common, Allan variance is not the only existing method for IMU characterization. Other methods based on Power Spectral Density (PSD) and Autocorrelation Function (ACF) [32][45], as well as modern and robust wavelet-variance methods [40][88][92] are often used when analysis of complex signals is inadequate with Allan variance (e.g., when the signal is composed of more than one latent correlated noise process). Though wavelet-variance methods such as the ones referenced have been shown to produce more optimal IMU characterization results than Allan variance, Allan variance remains the standard method of choice [46], and therefore motivates this research. Nevertheless, Allan variance was originally developed for the analysis of error sources in atomic clocks. Later, it was found useful for identifying error sources in accelerometers and gyroscopes using a "slope method" for analyzing Allan variance measurements. Such use of Allan variance in IMU modeling is so prolific across literature that it was compiled into an Institute of Electrical and Electronics Engineers (IEEE) standard [46]. In general, the Allan variance, $\sigma_a^2(\tau)$, of a continuous time signal, $\Omega(t)$, is a function of a quantity called averaging time, τ, and is given by

$$\sigma_a^2(\tau) = \frac{1}{2(N-2n)} \sum_{k=1}^{N-2n} \left[\bar{\Omega}_{k+1}(\tau) - \bar{\Omega}_k(\tau)\right]^2, \tag{7.10}$$

$$n = \frac{\tau}{\Delta t}, \tag{7.11}$$

where N is the total number of samples in the discretized signal, Δt is the sampling period, and

$$\bar{\Omega}_k(\tau) = \frac{1}{\tau}\int_{t_k}^{t_k+\tau} \Omega(t)\mathrm{d}t, \qquad \Delta t \leq \tau \leq N\Delta t/2. \tag{7.12}$$

Essentially, (7.10) divides the sampled signal into clusters, $\bar{\Omega}_k(\tau)$, which are averaged over a duration, τ, and computes the variance among groups as a function of varying τ. It is important to note the form of (7.10) is referred to as “non-overlapping,” since the clusters $\bar{\Omega}_k(\tau)$ do not overlap across time. Additionally, since each Allan variance point is computed from a finite set of samples per cluster, a percent error was derived in [85] and is given by

$$\delta_a = \frac{1}{\sqrt{2\left(\frac{N}{n} - 1\right)}}. \tag{7.13}$$

Allan variance can then be equated to the PSD of the input signal using

$$\sigma^2(\tau) = 4\int_0^\infty S_\Omega(f)\frac{\sin^4(\pi f\tau)}{(\pi f\tau)^2}\mathrm{d}f, \tag{7.14}$$

where f is frequency and $S_\Omega(f)$ is the PSD of $\Omega(t)$. The relationship illustrated in (7.14) is then used in [45] and [32] to exploit the properties of five key error sources: quantization, velocity/angle random walk, bias instability, acceleration/angular rate random walk, and rate ramp. Each of the five sources of error are systematically identified from an Allan variance plot of sensor (accelerometer or gyroscope) data using the slopes of the relationship between the PSD of each error source and its corresponding Allan variance formula. This relationship is explored in the following sections in order to develop an understanding of the slope method.

7.2.1 Slope Method.

This section describes the prolific slope method of identifying the five aforementioned sources of accelerometer and gyroscope error using Allan variance analysis. As shown below, this method exploits the relationship between an error source’s PSD and its corresponding Allan variance formula in a graphical context, whereby the slope of the Allan variance vs. τ graph is visually analyzed in order to extract the necessary information to estimate error. Although this method is simple and generally accurate, it suffers from two main limitations. First, it is difficult to automate since it is rooted in human visual inspection of an Allan variance vs. τ graph. As such, it requires complex logical programming or human intervention in the presence of nonstandard conditions (e.g., missing sources of noise). Additionally, when the length of sensor data is incomplete (i.e., not long enough to capture the underlying noise processes), the resulting Allan variance curve tends to become much more variable across data collections as τ increases. Such variability results in Allan variance slope behavior that is difficult to predict, making automated slope detection unreliable. Table 7.1 summarizes the key components of the slope method while Figures 7.1 and 7.2 illustrate the process.

7.2.1.1 Quantization Error.

Quantization is defined as the act of sampling an analog signal into discrete levels of size Δ during the analog-to-digital conversion process. The errors (differences between the analog signal and the digitize signal) caused by such quantization can be characterized as additive noise [9][46], which is uniformly distributed between $-\Delta/2$ and $\Delta/2$ [46]. Analyzing the relationship between the PSD function and the Allan deviation, $\sigma_a(\tau)$, of a signal composed only of quantization noise [45] gives

$$\sigma_a(\tau) = \frac{\sigma_q\sqrt{3}}{\tau} = \sigma_q\sqrt{3}\tau^{-1}. \tag{7.15}$$

Next, taking the common logarithm of both sides in (7.15) yields

$$\log_{10}(\sigma_a(\tau)) = \log_{10}(\sigma_q\sqrt{3}\tau^{-1}) \tag{7.16}$$

$$= -\log_{10}(\tau) + \log_{10}(\sigma_q) + \log_{10}(\sqrt{3}), \tag{7.17}$$

which implies σ_q can be identified in an Allan deviation curve by finding a -1 slope when plotting $\log_{10}(\sigma_a(\tau))$ against $\log_{10}(\tau)$. Additionally, letting $\tau = \sqrt{3}$ in (7.17) solves the equation for σ_q, which means if the -1 slope line is projected to $\tau = \sqrt{3}$, the value of $\sigma_a(\tau)$ at that point will equal σ_q. This process is illustrated in Figure 7.1 and summarized in Table 7.1.

7.2.1.2 Angle/Velocity Random Walk.

As indicated by its name, angle or velocity random walk is a random walk process observed in the angle or velocity signal output of an inertial sensor. In terms of (7.2) or (7.6), angle/velocity random walk arises from integrating η_x in $\tilde{\omega}_x$ or $\tilde{a}_x$. The relationship between the Allan deviation and the PSD for a signal of this type is given by

$$\sigma_a(\tau) = \frac{\sigma_{rw}}{\sqrt{\tau}} = \sigma_{rw}\tau^{-1/2}. \tag{7.18}$$

Repeating the process followed for quantization noise yields

$$\begin{aligned}\log_{10}(\sigma_a(\tau)) &= \log_{10}(\sigma_{rw}\tau^{-1/2}) &(7.19)\\ &= -\frac{1}{2}\log_{10}(\tau) + \log_{10}(\sigma_{rw}), &(7.20)\end{aligned}$$

which implies σ_{rw} can be identified in an Allan deviation curve by finding a $-1/2$ slope when plotting $\log_{10}(\sigma_a(\tau))$ against $\log_{10}(\tau)$. Letting $\tau = 1$ in (7.20) solves the equation for σ_{rw}, which means if the $-1/2$ slope line is projected to $\tau = 1$, the value of $\sigma_a(\tau)$ at that point will equal σ_{rw}.

7.2.1.3 Bias Instability.

Bias instability, sometimes referred to ironically as bias stability, refers to the tendency of an inertial sensor's constant bias (B_f in (7.2) or (7.6)) to change or drift during use. The most accurate description of the stochastic process behind this drift is flicker (or $1/f$) noise as shown by [67]. However, due to complications in the modeling of flicker noise in common navigation estimation algorithms, such as a Kalman filter [59], this process is often approximated by a first order Gauss-Markov process [46, Fig. C.6][76] . Following the slope method process yields

$$\begin{aligned}\sigma_a(\tau) &= \sigma_b\sqrt{\frac{2\log(2)}{\pi}} = \sigma_b\sqrt{\frac{2\log(2)}{\pi}}\tau^0, &(7.21)\\ \log_{10}(\sigma_a(\tau)) &= 0\log_{10}(\tau) + \log_{10}(\sigma_b) + \log_{10}\left(\sqrt{\frac{2\log(2)}{\pi}}\right), &(7.22)\end{aligned}$$

which indicates there is no relation to τ in (7.21). That is, the flicker noise coefficient can be identified in an Allan deviation curve by finding a 0 slope when plotting $\log_{10}(\sigma_a(\tau))$ against $\log_{10}(\tau)$. Additionally, (7.22) implies the value of $\sigma_a(\tau)$ at that point should be scaled by $\sqrt{2\log(2)/\pi}$ to solve for σ_b.

7.2.1.4 Acceleration/Angular Rate Random Walk.

In contrast to angle/velocity random walk, rate random walk refers to a random walk process observed in the inertial sensor's rate signal (acceleration or angular rate). In terms of (7.2) or (7.6), rate random walk arises from integrating white noise found in $\dot{\tilde{\omega}}_x$ or $\dot{\tilde{a}}_x$. Again, the relationship between the Allan deviation and the PSD for a signal of this type yields

$$\sigma_a(\tau) = \sigma_{rrw}\sqrt{\frac{\tau}{3}} = \sigma_{rrw}\frac{1}{\sqrt{3}}\tau^{1/2}, \tag{7.23}$$

$$\log_{10}(\sigma_a(\tau)) = \frac{1}{2}\log_{10}(\tau) + \log_{10}(\sigma_{rrw}) - \frac{1}{2}\log_{10}(3), \tag{7.24}$$

which implies σ_{rrw} can be identified in an Allan deviation curve by finding a $+1/2$ slope when plotting $\log_{10}(\sigma_a(\tau))$ against $\log_{10}(\tau)$. Letting $\tau = 3$ in (7.24) solves the equation for σ_{rrw}, which means if the $+1/2$ slope line is projected to $\tau = 3$, the value of $\sigma_a(\tau)$ at that point will equal σ_{rrw}.

7.2.1.5 Rate Ramp.

Finally, rate ramp refers to the deterministic, linear and usually long-term increase of the inertial sensor's rate signal output. In terms of (7.2) or (7.6), rate random walk arises when B_{fx} or B_f linearly changes over time at a deterministic (e.g., non stochastic but unknown) rate. The slope method then yields

$$\sigma_a(\tau) = \sigma_{rr}\frac{\tau^1}{\sqrt{2}}, \tag{7.25}$$

$$\log_{10}(\sigma(\tau)) = \log_{10}(\tau) + \log_{10}(\sigma_{rr}) - \log_{10}(\sqrt{2}), \tag{7.26}$$

which implies σ_{rr} can be identified in an Allan deviation curve by finding a $+1$ slope when plotting $\log_{10}(\sigma_a(\tau))$ against $\log_{10}(\tau)$. Letting $\tau = \sqrt{2}$ in (7.26) solves the equation for σ_{rr}, which means if the $+1$ slope line is projected to $\tau = \sqrt{2}$, the value of $\sigma_a(\tau)$ at that point will equal σ_{rr}.

Although the five sources of error are well defined mathematically along the Allan deviation curve via the use of the slope method, methods for solution are based upon visual inspection of the graph. That is, for a specific sensor and application, lines with the specific slope(s) of interest are created and estimates for each parameter are back solved by hand or through human-visual inspection. With current autonomous systems, this tedious process hampers efficient calibration of IMUs, especially when the available sensor data is not long enough to ensure a stable Allan variance curve. Although the length of available data required varies with each source of error and its true underlying value, general rules of thumb [46][45] suggest several hours of data are usually required for the slope method to provide accurate estimates of all sources, especially for those prevalent in the latter regions of the τ domain (e.g., σ_{rrw} and σ_{rr}) since their effects are only visible after several hours of continuous IMU operation.

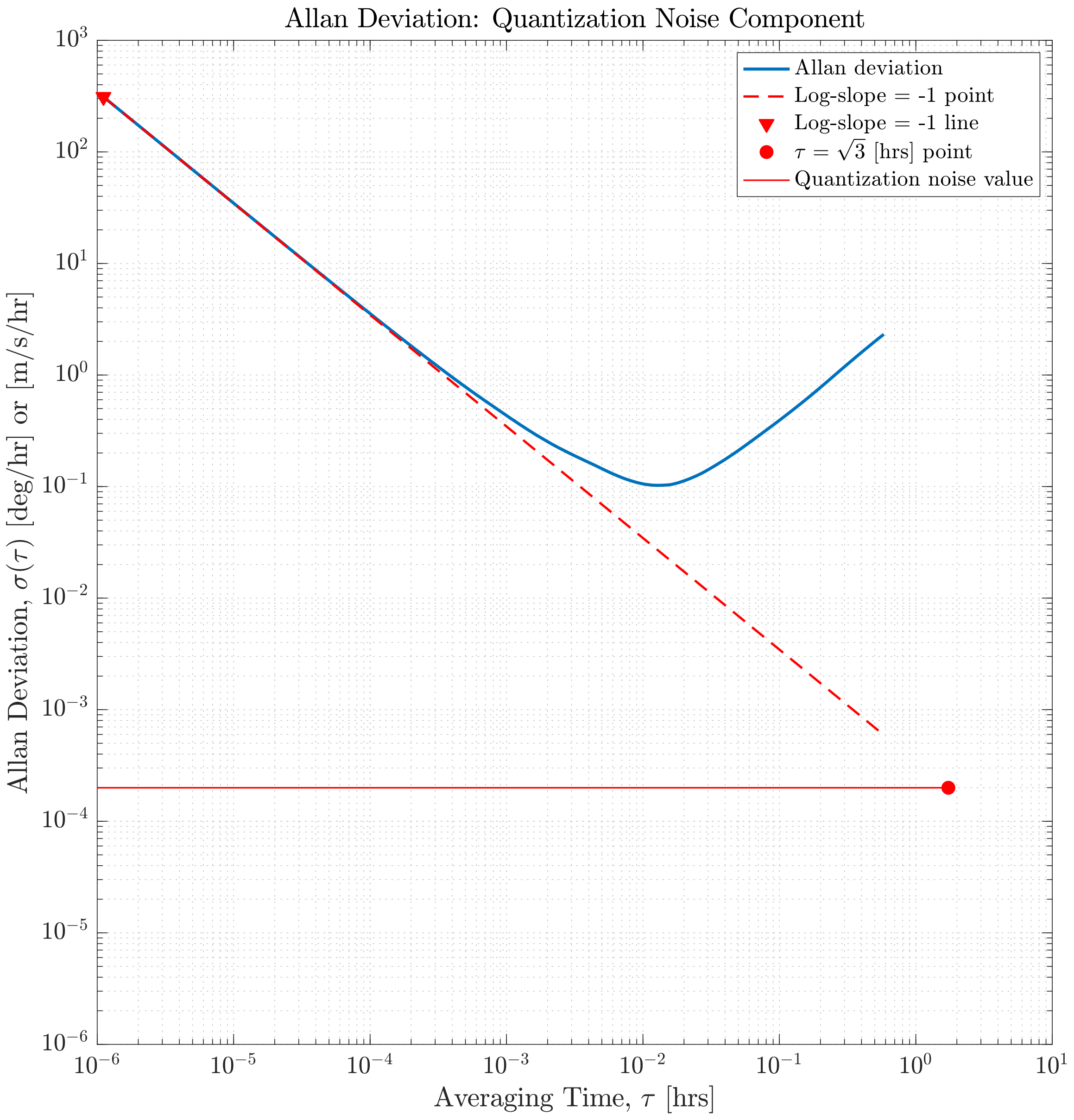


Figure 7.1: Illustration of Allan variance slope method for quantization noise.

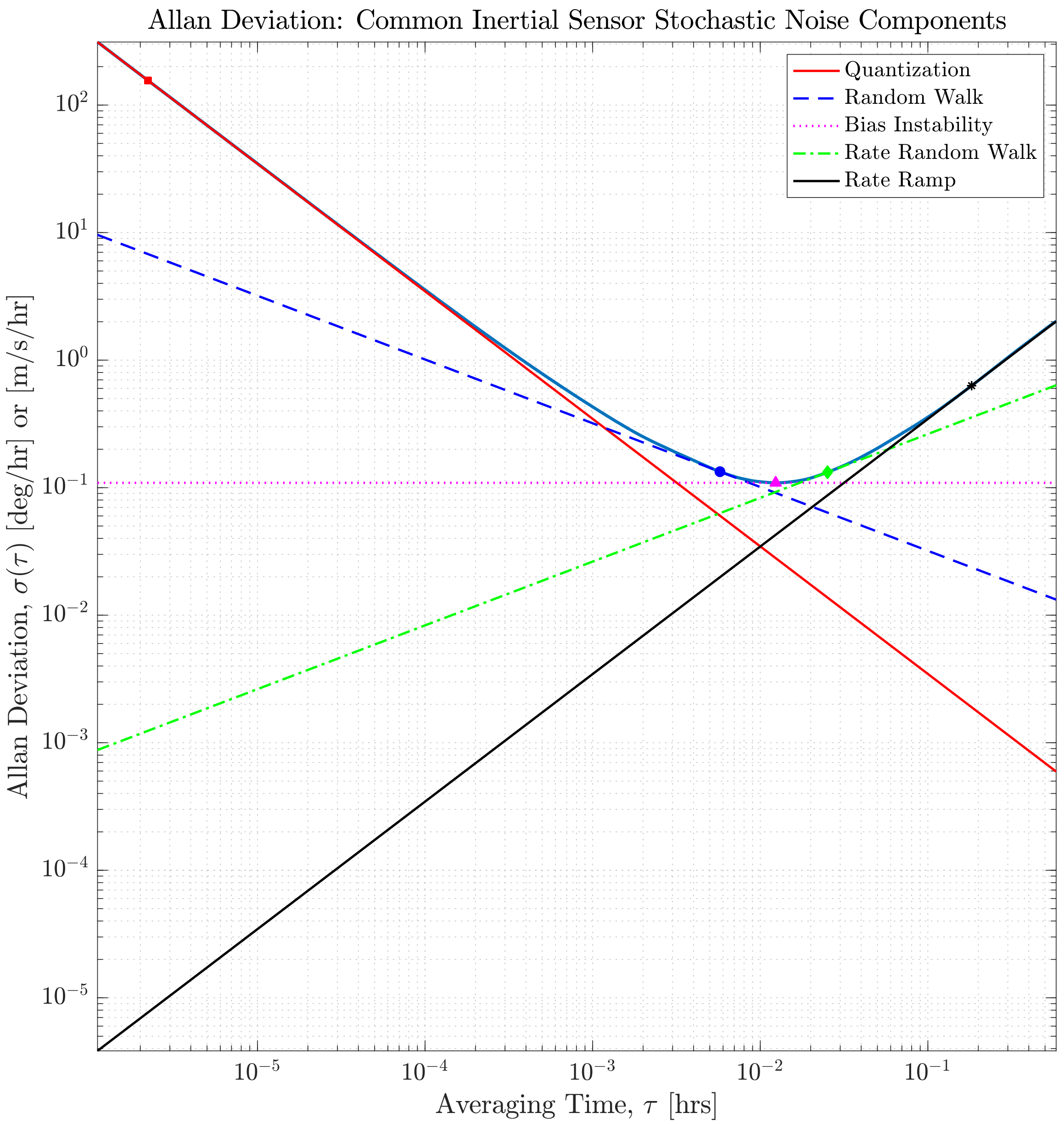


Figure 7.2: Illustration of Allan variance slope method for common stochastic noise processes.

Table 7.1: Summary of Allan deviation slopes for common IMU noise processes.

			Graphical ID		
Noise source	Symbol	**Relation to PSD**	**Slope**	**τ at desired σ**	**Coefficient units***
Quantization	σ_q	$\sigma_a(\tau) = \sigma_q \sqrt{3}\tau^{-1}$	-1	$\sqrt{3}$	[deg] or [m/s]
Random walk	σ_{rw}	$\sigma_a(\tau) = \sigma_{rw}\tau^{-1/2}$	$-1/2$	1	[deg/ $\sqrt{\text{hr}}$] or [m/s/ $\sqrt{\text{hr}}$]
Bias instability	σ_b	$\sigma_a(\tau) = \sigma_b \sqrt{\frac{2\log(2)}{\pi}}\tau^0$	0	–	[deg/hr] or [m/s/hr]
Rate random walk	σ_{rrw}	$\sigma_a(\tau) = \sigma_{rrw}\frac{1}{\sqrt{3}}\tau^{1/2}$	$1/2$	3	[deg/hr/ $\sqrt{\text{hr}}$] or [m/s/hr/ $\sqrt{\text{hr}}$]
Rate ramp	σ_{rr}	$\sigma_a(\tau) = \sigma_{rr}\frac{1}{\sqrt{2}}\tau^1$	1	$\sqrt{2}$	[deg/hr/hr] or [m/s/hr/hr]

*Units result from $\sigma(\tau)$ measured in [deg/hr] or [m/s/hr] and τ measured in [hrs]

7.3 An autonomous method for estimating noise strength

The proposed method, referred to as Autonomous Regression Method for Allan Variance (ARMAV) from hereon, differs from the slope method in that it combines linear ridge regression [44] and nonlinear model estimation [8] in order to yield accurate and stable estimates for the five common noise sources in IMUs instead of using visual inspection of graphical methods, which are hard to automate. As designed, ARMAV not only performs comparably in terms of estimation accuracy, but is also completely autonomous, stable under limited data conditions, and suitable for online IMU characterization.

The key components of the slope method, which are summarized in Table 7.1, provide the fundamental relationships between observed data (Allan variance) and its predictor variable, τ. However, the slope method identifies each noise strength coefficient individually by restricting the graphical search to the areas of the τ domain where each noise source is dominant. Using [46], and the assumption of independence among the sources of noise, the combined relationship between total Allan variance, σ_a, and the contributions from each of the sources is given by

$$\log_{10}\left(\sigma_a^2\right) = \log_{10}\left(\sigma_{a_q}^2 + \sigma_{a_{rw}}^2 + \sigma_{a_b}^2 + \sigma_{a_{rrw}}^2 + \sigma_{a_{rr}}^2\right). \tag{7.27}$$

Next, substituting the relationships from Table 7.1 yields

$$\log_{10}\left(\sigma_a^2\right) = \log_{10}\Bigg[\left(\sigma_q\sqrt{3}\tau^{-1}\right)^2 + \left(\sigma_{rw}\tau_1^{-1/2}\right)^2 + \left(\sigma_b\sqrt{\frac{2\log(2)}{\pi}}\right)^2 + \left(\sigma_{rrw}\frac{1}{\sqrt{3}}\tau^{1/2}\right)^2 + + \left(\sigma_{rr}\frac{1}{\sqrt{2}}\tau\right)^2\Bigg], \tag{7.28}$$

from which a nonlinear regression problem with N observations of the form

$$\log_{10}(\mathbf{y}^2) = \log_{10}\left[(\mathbf{X}\boldsymbol{\beta})^2\right] + \boldsymbol{\epsilon}, \tag{7.29}$$

where

$$\mathbf{y} = \begin{bmatrix} \sigma_{a_1} \\ \vdots \\ \sigma_{a_N} \end{bmatrix}, \tag{7.30}$$

$$\mathbf{X} = \begin{bmatrix} \sqrt{3}\tau_1^{-1} & \tau_1^{-1/2} & 1 & \frac{1}{\sqrt{3}}\tau_1^{1/2} & \frac{1}{\sqrt{2}}\tau_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sqrt{3}\tau_N^{-1} & \tau_N^{-1/2} & 1 & \frac{1}{\sqrt{3}}\tau_N^{1/2} & \frac{1}{\sqrt{2}}\tau_N \end{bmatrix}, \tag{7.31}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \sigma_q \\ \sigma_{rw} \\ \sigma_b^* \\ \sigma_{rrw} \\ \sigma_{rr} \end{bmatrix}, \tag{7.32}$$

and

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \tag{7.33}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} (\sigma_{a_1}\delta_{a_1})^2 & 0 & \dots & 0 \\ 0 & (\sigma_{a_2}\delta_{a_2})^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & (\sigma_{a_N}\delta_{a_N})^2 \end{bmatrix}, \tag{7.34}$$

can be constructed and solved using any weighted least-squares nonlinear regression algorithm such as Gauss-Newton [8] or Levenberg-Marquardt [75]. However, since nonlinear regression problems often require an accurate initial guess, $\boldsymbol{\beta}_0$, to converge onto the global minimum, we first use a linear approximation of (7.29) to solve the linear model

$$\underbrace{\begin{bmatrix} \sigma_{a_1} \\ \vdots \\ \vdots \\ \sigma_{a_N} \end{bmatrix}}_{\mathbf{y}\in\mathbb{R}^{N\times 1}} = \underbrace{\begin{bmatrix} \sqrt{3}\tau_1^{-1} & \tau_1^{-1/2} & 1 & \frac{1}{\sqrt{3}}\tau_1^{1/2} & \frac{1}{\sqrt{2}}\tau_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sqrt{3}\tau_N^{-1} & \tau_N^{-1/2} & 1 & \frac{1}{\sqrt{3}}\tau_N^{1/2} & \frac{1}{\sqrt{2}}\tau_N \end{bmatrix}}_{\mathbf{X}\in\mathbb{R}^{N\times 5}} \underbrace{\begin{bmatrix} \sigma_{q_0} \\ \sigma_{rw_0} \\ \sigma_{b_0}^* \\ \sigma_{rrw_0} \\ \sigma_{rr_0} \end{bmatrix}}_{\boldsymbol{\beta}_0\in\mathbb{R}^{5\times 1}} + \boldsymbol{\epsilon}. \tag{7.35}$$

The model described in (7.35), however, presents a significant multicollinearity problem since almost every column in $\mathbf{X}$ is dependent on τ. Although multicollinearity is usually not a problem when evaluating the model's ability to predict the observed data, it is extremely problematic here since the desired inference (i.e. the initial guess) is based on the individual coefficient values in $\boldsymbol{\beta}$. Therefore, ridge regression [44] is used to solve (7.35) using

$$\hat{\boldsymbol{\beta}}_0 = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{7.36}$$

where λ is a tunable, small biasing constant. Using the initial guess $\hat{\boldsymbol{\beta}}_0$, the nonlinear model (7.29) is then solved to produce $\hat{\boldsymbol{\beta}}$. Finally, it is important to realize the desired estimate of σ_b is not directly given by $\hat{\sigma}_b^*$ since it is simply an estimate of the model's intercept. To obtain the desired $\hat{\sigma}_b$, the fitted model in (7.29) is used along with (7.22) to yield

$$\hat{\sigma}_b = \sqrt{\frac{\pi}{2\log(2)}}\min\left(\mathbf{X}\hat{\boldsymbol{\beta}}\right). \tag{7.37}$$

This process is summarized in Algorithm 7.1. The ARMAV method was validated using a series of Monte-Carlo simulations along with real-world sensor data from a STIM-300 (tactical grade) IMU; the results of which are discussed in the following sections.

7.4 Simulation

A 3000-trial Monte-Carlo simulation was executed across 30 unique levels where the length of available sensor data was incrementally decreased from 6 hours (5.4 million samples) to 6 minutes (90 thousand samples). Using simulated IMU data, both the slope method and ARMAV (Algorithm 7.1) were used to estimate the five known simulated noise strength coefficients. The true coefficients were fixed for the entire simulation and are summarized in Table 7.2. Simulated IMU data were generated using the numerical methods described by [53], and are summarized in the following paragraph for completeness.

In general, simulated IMU data were generated as rate signals, with units of [deg/s] or [m/s/s], by applying the appropriate arithmetic operation to the underlying random process for each source of noise, and with the corresponding standard deviation from Table 7.2.

Algorithm 7.1 Autonomous Regression Method for Allan Variance

```
Input: $\boldsymbol{\sigma}_a, \boldsymbol{\tau}$    ▶ Allan deviation data in [hrs]
1: $\mathbf{X}_1 \leftarrow \left[ \begin{array}{ccccc} \frac{\sqrt{3}}{\tau} & \frac{1}{\sqrt{\tau}} & \mathbf{1} & \frac{\sqrt{\tau}}{\sqrt{3}} & \frac{\tau}{\sqrt{2}} \end{array} \right]$    ▶ Construct linear regressor matrix
2: $\boldsymbol{\beta}_0 \leftarrow \left(\mathbf{X}_1^{\mathrm{T}}\mathbf{X}_1 + \lambda\mathbf{I}\right)^{-1} \mathbf{X}_1^{\mathrm{T}}\boldsymbol{\sigma}_a$    ▶ Ridge regression for initial guess
3: $\mathbf{X}_2 \leftarrow \left[ \begin{array}{ccccc} \frac{3}{\tau^2} & \frac{1}{\tau} & \mathbf{1} & \frac{\tau}{3} & \frac{\tau^2}{2} \end{array} \right]$    ▶ Construct nonlinear regressor matrix
4: $f(\mathbf{X},\boldsymbol{\beta}) \leftarrow \log_{10}(\mathbf{X}\boldsymbol{\beta}^2)$    ▶ Build nonlinear regression function
5: $w_i \leftarrow \frac{1}{(\sigma_{a_i}\delta_{a_i})^2} \quad i = 1, \ldots, N$    ▶ Create weight vector from percent error formula
6: $\boldsymbol{\beta}_1 \leftarrow$ fitnlm($\mathbf{X}_2, \log_{10}(\mathbf{a}^2), f, \boldsymbol{\beta}_0,$ 'weights', $\mathbf{w}$)    ▶ Use nonlinear solver to estimate

Output: $\sigma_q, \sigma_{rw}, \sigma_b, \sigma_{rrw}, \sigma_{rr}$
    $\sigma_q \leftarrow \boldsymbol{\beta}_1(1)$    ▶ Extract quantization noise coefficient
    $\sigma_{rw} \leftarrow \boldsymbol{\beta}_1(2)$    ▶ Extract random walk noise coefficient
    $\sigma_b \leftarrow \sqrt{\frac{\pi}{2\log(2)}} \min(\mathbf{X}_1\boldsymbol{\beta}_1)$    ▶ Extract bias instability noise coefficient
    $\sigma_{rrw} \leftarrow \boldsymbol{\beta}_1(4)$    ▶ Extract rate random walk noise coefficient
    $\sigma_{rr} \leftarrow \boldsymbol{\beta}_1(5)$    ▶ Extract rate ramp noise coefficient
```

For example, Angle/Velocity Random Walk (σ_{rw}) data were generated directly as zero-mean White Gaussian Noise (WGN) since a random walk process in the integrated signal, with units [deg] or [m/s], arises from the integration of WGN in the rate signal, which has units [deg/s] or [m/s/s]. Meanwhile, Rate Random Walk (σ_{rrw}) data were generated by numerically integrating a WGN sequence, with units [deg/s/s] or [m/s/s/s], since the desired random walk was to be found in the rate signal and not its integral.

Next, the ARMAV method was programmed as shown in Algorithm 7.1, with a $\lambda = 5 \times 10^{-3}$ value. The particular λ value was found experimentally by monitoring Variation Inflation Factor (VIF) values [68]. It is important to note, the specific value of λ did not have a significant effect on the final coefficient estimates since it only affected the initial guess used in nonlinear regression.

Finally, the slope method was programmed for comparison to ARMAV also using the methods described by [53]. As a brief summary, the slope method was programmed to calculate the slope of the observed Allan deviation data and find the closest point (Euclidean distance) on the slope curve to each of the five slopes of interest. Then, the Allan deviation value at the particular τ of interest was found by using a point-slope formula for the desired line.

For each of the 30 distinct levels of available sensor data, 3000 trials were conducted, and the resulting mean relative bias and associated 95% basic percentile confidence intervals were estimated by bootstrapping. Figure 7.3 illustrates the percent relative mean bias and associated 95% basic percentile confidence interval for each level of the simulation and for each of the five noise sources, all relative to their respective true values from Table 7.2. Additionally, Table 7.3 summarizes percent relative bias results from the simulation at the 1-hour, 3-hour, and 6-hour levels.

Overall, both the figure and the table illustrate more stable and generally more accurate estimates when using ARMAV, especially when the length of available data is greater than 1 hour. As shown in both Figure 7.3 and Table 7.3, ARMAV produced substantially more stable results (in terms of variance) as indicated by the width of the confidence interval, especially as the length of available data decreased below 2 hours. With the exception of σ_{rrw}, percent relative bias was smaller with less variability when ARMAV was used to estimate the errors than when the slope method was used. In the case of σ_{rrw}, ARMAV resulted in a lower percent relative bias until the length of available data fell below 0.5 hours. It is also important to note, the slope method resulted in several instances of large variance for particular lengths of available data, generally less than two hours, across every noise source.

Comparisons of resulting estimation of ARMAV to the slope method were also conducted for applications and settings in which the Allan deviation curve is essentially incomplete, that is, in scenarios where the simulated stochastic processes did not include one of the five common noise components. Namely, where either quantization or rate ramp components were not included in the simulated IMU data. In these comparisons, another set of 3000-trial Monte-Carlo simulations were conducted separately for each source of error and in the same manner as previously presented, with the exception that the true σ_q and σ_{rr} were set to zero in each simulation, respectively. For comparison, results from the slope method were also computed using two techniques: in autonomous mode, the slope method was allowed to run as previously described [53], with no additional human intervention, while in manual mode, the slope method was re-programmed to skip the estimation of the particular noise coefficient that was known to be zero. Tables 7.4 and 7.5 provide, respectively, the results of these simulations for the cases when quantization and rate ramp components were missing. Results are presented in terms of bias rather than percent relative bias for better comparisons.

As shown in both tables, the non-normalized biases from ARMAV were up to four orders of magnitude closer to the truth (zero) when compared to the autonomous slope method. This is due to the fact the slope method, when programmed, looks for all parameters (i.e., finds the closest answer matching the graphical method for each noise coefficient). In contrast, the manual slope method was re-programmed to assume the missing noise coefficient was zero. As expected, its results for all other coefficients were an exact match to the autonomous-mode slope method. Here, it is important to emphasize these scenarios had to be specially programmed from an initial visual inspection of the Allan variance data for the slope method to produce good results. In contrast, for data applications in which these sources of noise are not estimable from the data, ARMAV autonomously and reliably provided reasonable estimates with no changes to the programmed algorithm.

7.5 Application to STIM-300 IMU Analysis

The ARMAV method was applied to real-world sensor data from a STIM-300 (tactical grade) IMU. The manufacturer of this sensor provides specifications for the values of σ_{rw} and σ_b [91], which are reproduced along with the analysis results in Table 7.6. A a single

6-hour data collection was performed at static conditions and room temperature for the x-axis accelerometer and gyroscope at a sampling rate of 250 Hz. It is important to note that the purpose of this data collect was to simply demonstrate the ability of ARMAV to match manufacturer specifications, which were only specified for the random walk and bias instability components. Therefore the internal temperature of the IMU was not tightly controlled. The MATLAB code and inertial dataset used in this research are provided as supplementary materials via [1]. Plots of the fitted Allan deviance curve resulting from the application of the regression method for both the accelerometer and gyroscope are provided in Figures 7.4 and 7.5. As shown, ARMAV is able to accurately model the observed data from both devices with no need for human intervention and directly enables the accurate estimation of the necessary noise strength coefficients. The resulting 95% prediction bands generated in each figure cover the possible range of future observations simultaneously and provides a measure of the model's quality. Finally, Table 7.6 provides the results from ARMAV on the STIM-300 IMU data and compares these estimates to the slope method. As shown, the particular sensor tested did not exhibit quantization noise (i.e $\sigma_q = 0$), yet ARMAV was able to accurately estimate all noise coefficients, and in the case of σ_q, its estimates were up to five orders of magnitude closer to zero when compared to the autonomous slope method. Additionally, in the case of σ_{rw} and σ_b, where the manufacturer provided specifications [91], ARMAV was closer to specifications in the majority of cases, and always at least as accurate as the slope method.

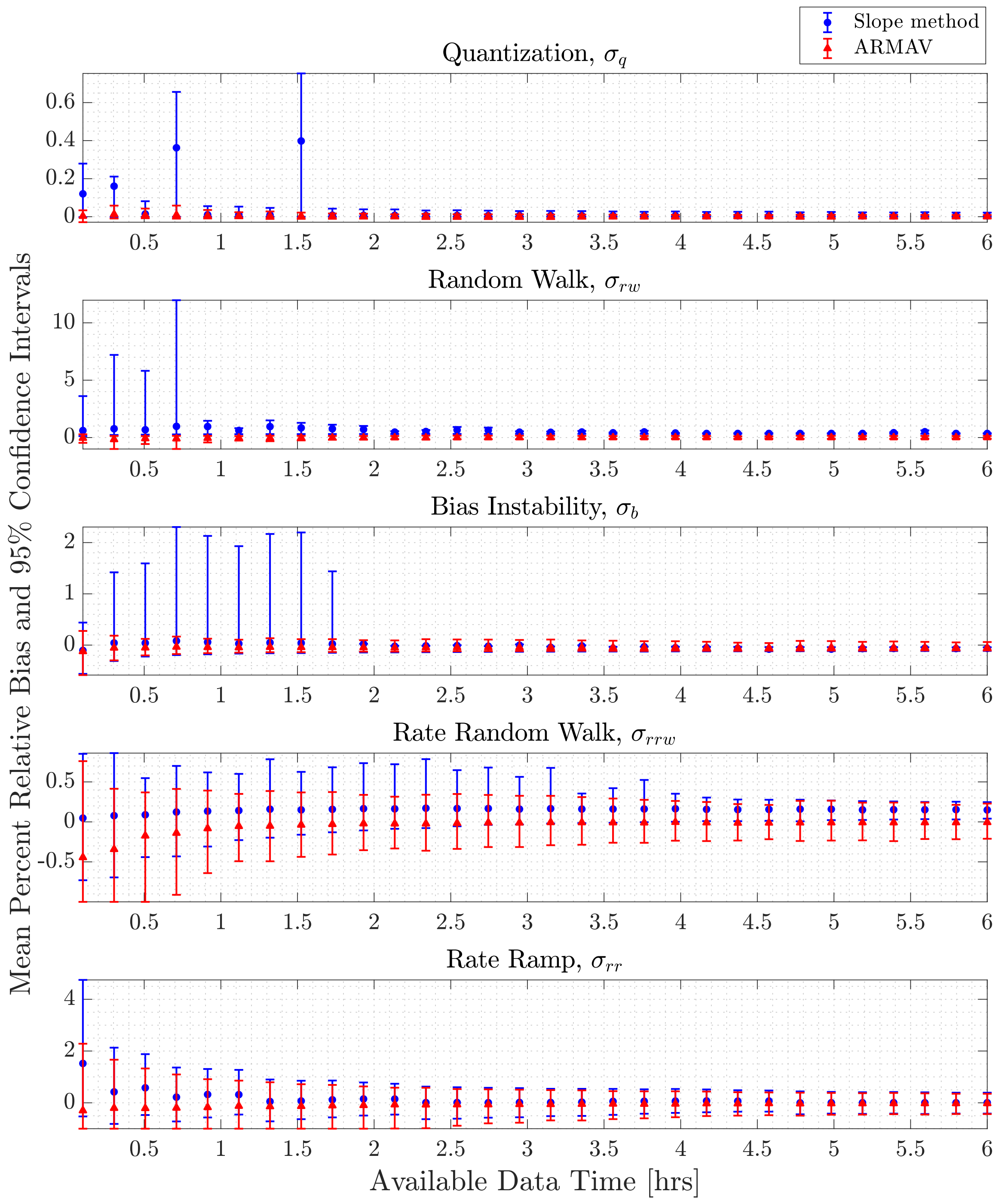


Figure 7.3: Monte-Carlo comparison between slope and ARMAV methods. The ARMAV method produced significantly more accurate, and stable results (in terms of variance) as indicated by the confidence intervals, especially as the length of available data decreased below 2 hours.

Table 7.2: Summary of true noise coefficients for Monte-Carlo simulation

Noise source	**Value**	**Units**
σ_q	2×10^{-4}	[deg] or [m/s]
σ_{rw}	8×10^{-3}	[deg/ $\sqrt{\text{hr}}$] or [m/s/ $\sqrt{\text{hr}}$]
σ_b	1×10^{-1}	[deg/hr] or [m/s/hr]
σ_{rrw}	2.00	[deg/hr/ $\sqrt{\text{hr}}$] or [m/s/hr/ $\sqrt{\text{hr}}$]
σ_{rr}	5.00	[deg/hr/hr] or [m/s/hr/hr]

Table 7.3: Mean percent relative bias comparison, slope vs. ARMAV.

		Slope method				**ARMAV**			
Time	$\hat{\beta}$	**Mean**	**Std. dev.**	**95% LCL**[1]	**95% UCL**[2]	**Mean**	**Std. dev.**	**95% LCL**[1]	**95% UCL**[2]
	$\hat{\sigma}_q$	1.03×10^{-2}	1.48×10^{-2}	-3.02×10^{-4}	5.31×10^{-2}	3.65×10^{-3}	8.26×10^{-3}	-8.47×10^{-3}	2.37×10^{-2}
	$\hat{\sigma}_{rw}$	6.15×10^{-1}	2.87	2.93×10^{-1}	4.16×10^{-1}	-2.98×10^{-2}	1.11×10^{-1}	-2.67×10^{-1}	1.26×10^{-1}
1 hr	$\hat{\sigma}_b$	2.80×10^{-2}	5.71×10^{-1}	-1.65×10^{-1}	1.93	-4.79×10^{-2}	6.18×10^{-2}	-1.48×10^{-1}	1.05×10^{-1}
	$\hat{\sigma}_{rrw}$	1.42×10^{-1}	2.04×10^{-1}	-2.28×10^{-1}	6.00×10^{-1}	-5.42×10^{-2}	2.12×10^{-1}	-4.92×10^{-1}	3.50×10^{-1}
	$\hat{\sigma}_{rr}$	3.16×10^{-1}	4.34×10^{-1}	-4.54×10^{-1}	1.27	-1.26×10^{-1}	5.25×10^{-1}	-1.00	8.57×10^{-1}
	$\hat{\sigma}_q$	6.38×10^{-3}	8.56×10^{-3}	2.27×10^{-6}	3.04×10^{-2}	9.81×10^{-4}	5.77×10^{-3}	-1.24×10^{-2}	1.17×10^{-2}
	$\hat{\sigma}_{rw}$	4.55×10^{-1}	3.26	3.17×10^{-1}	3.87×10^{-1}	1.78×10^{-3}	7.10×10^{-2}	-1.30×10^{-1}	1.61×10^{-1}
3 hrs	$\hat{\sigma}_b$	-4.31×10^{-2}	5.81×10^{-1}	-1.29×10^{-1}	-2.88×10^{-2}	-4.64×10^{-2}	5.37×10^{-2}	-1.15×10^{-1}	1.08×10^{-1}
	$\hat{\sigma}_{rrw}$	1.66×10^{-1}	1.86×10^{-1}	-2.02×10^{-2}	6.76×10^{-1}	-6.57×10^{-3}	1.51×10^{-1}	-2.91×10^{-1}	3.26×10^{-1}
	$\hat{\sigma}_{rr}$	1.32×10^{-2}	2.71×10^{-1}	-5.15×10^{-1}	5.41×10^{-1}	-5.72×10^{-2}	2.93×10^{-1}	-6.80×10^{-1}	4.91×10^{-1}
	$\hat{\sigma}_q$	4.75×10^{-3}	6.09×10^{-3}	1.03×10^{-4}	2.17×10^{-2}	8.32×10^{-4}	4.23×10^{-3}	-9.06×10^{-3}	8.82×10^{-3}
	$\hat{\sigma}_{rw}$	3.53×10^{-1}	1.30×10^{-2}	3.27×10^{-1}	3.78×10^{-1}	4.26×10^{-3}	5.32×10^{-2}	-9.73×10^{-2}	1.37×10^{-1}
6 hrs	$\hat{\sigma}_b$	-6.32×10^{-2}	5.05×10^{-1}	-1.14×10^{-1}	-4.25×10^{-2}	-5.50×10^{-2}	3.92×10^{-2}	-1.05×10^{-1}	5.96×10^{-2}
	$\hat{\sigma}_{rrw}$	1.50×10^{-1}	1.14×10^{-1}	4.06×10^{-2}	2.47×10^{-1}	-5.18×10^{-3}	1.09×10^{-1}	-2.11×10^{-1}	2.30×10^{-1}
	$\hat{\sigma}_{rr}$	2.45×10^{-3}	2.02×10^{-1}	-4.16×10^{-1}	3.92×10^{-1}	-2.29×10^{-2}	1.95×10^{-1}	-4.35×10^{-1}	3.50×10^{-1}

[1] LCL - Lower confidence level from basic percentile

[2] UCL - Upper confidence level from basic percentile

Table 7.4: Actual estimation bias comparison, slope vs. ARMAV, no quantization.

		Slope method: Auto		**Slope method: Manual**		**ARMAV**	
Time	$\hat{\beta}$	**Mean**	**Std. dev.**	**Mean**	**Std. dev**	**Mean**	**Std. dev.**
	$\hat{\sigma}_q$	4.25×10^{-3}	1.69×10^{-2}	–	–	6.81×10^{-7}	7.73×10^{-7}
	$\hat{\sigma}_{rw}$	5.60×10^{-5}	1.26×10^{-3}	5.60×10^{-5}	1.26×10^{-3}	1.83×10^{-5}	1.14×10^{-4}
1 hr	$\hat{\sigma}_b$	1.39×10^{-1}	1.44×10^{-1}	1.39×10^{-1}	1.44×10^{-1}	1.14×10^{-1}	1.17×10^{-2}
	$\hat{\sigma}_{rrw}$	2.56×10^{-1}	4.38×10^{-1}	2.56×10^{-1}	4.38×10^{-1}	-1.20×10^{-1}	3.77×10^{-1}
	$\hat{\sigma}_{rr}$	1.57	2.33	1.57	2.33	-7.38×10^{-1}	2.80
	$\hat{\sigma}_q$	1.40×10^{-3}	3.13×10^{-2}	–	–	3.37×10^{-7}	5.07×10^{-7}
	$\hat{\sigma}_{rw}$	2.45×10^{-5}	4.64×10^{-5}	2.45×10^{-5}	4.64×10^{-5}	4.56×10^{-5}	6.08×10^{-5}
3 hrs	$\hat{\sigma}_b$	1.19×10^{-1}	1.49×10^{-1}	1.19×10^{-1}	1.49×10^{-1}	1.13×10^{-1}	6.60×10^{-3}
	$\hat{\sigma}_{rrw}$	3.14×10^{-1}	3.30×10^{-1}	3.14×10^{-1}	3.30×10^{-1}	-2.41×10^{-2}	2.17×10^{-1}
	$\hat{\sigma}_{rr}$	2.66×10^{-1}	1.27	2.66×10^{-1}	1.27	-1.96×10^{-1}	1.34
	$\hat{\sigma}_q$	7.71×10^{-5}	4.07×10^{-5}	–	–	2.40×10^{-7}	4.08×10^{-7}
	$\hat{\sigma}_{rw}$	2.27×10^{-5}	4.25×10^{-5}	2.27×10^{-5}	4.25×10^{-5}	4.53×10^{-5}	4.89×10^{-5}
6 hrs	$\hat{\sigma}_b$	1.09×10^{-1}	4.08×10^{-3}	1.09×10^{-1}	4.08×10^{-3}	1.12×10^{-1}	5.04×10^{-3}
	$\hat{\sigma}_{rrw}$	2.91×10^{-1}	2.10×10^{-1}	2.91×10^{-1}	2.10×10^{-1}	-2.09×10^{-2}	1.68×10^{-1}
	$\hat{\sigma}_{rr}$	3.94×10^{-2}	1.02	3.94×10^{-2}	1.02	-6.97×10^{-2}	9.73×10^{-1}

Table 7.5: Actual estimation bias comparisons, slope vs. ARMAV, no rate ramp.

		Slope method: Auto		**Slope method: Manual**		**ARMAV**	
Time	$\hat{\beta}$	**Mean**	**Std. dev.**	**Mean**	**Std. dev**	**Mean**	**Std. dev.**
1 hr	$\hat{\sigma}_q$	5.04×10^{-5}	1.56×10^{-3}	5.04×10^{-5}	1.56×10^{-3}	2.94×10^{-6}	3.46×10^{-6}
	$\hat{\sigma}_{rw}$	1.28×10^{-2}	4.32×10^{-2}	1.28×10^{-2}	4.32×10^{-2}	-1.43×10^{-3}	2.06×10^{-3}
	$\hat{\sigma}_b$	1.83×10^{-1}	1.69×10^{-1}	1.83×10^{-1}	1.69×10^{-1}	1.28×10^{-1}	1.78×10^{-2}
	$\hat{\sigma}_{rrw}$	-9.21×10^{-2}	4.76×10^{-1}	-9.21×10^{-2}	4.76×10^{-1}	-3.77×10^{-1}	4.90×10^{-1}
	$\hat{\sigma}_{rr}$	5.28	2.85	–	–	1.17	1.66
3 hrs	$\hat{\sigma}_q$	5.42×10^{-4}	1.73×10^{-2}	5.42×10^{-4}	1.73×10^{-2}	2.22×10^{-6}	2.86×10^{-6}
	$\hat{\sigma}_{rw}$	3.70×10^{-2}	1.57×10^{-1}	3.70×10^{-2}	1.57×10^{-1}	-9.57×10^{-4}	1.47×10^{-3}
	$\hat{\sigma}_b$	2.60×10^{-1}	3.62×10^{-1}	2.60×10^{-1}	3.62×10^{-1}	1.28×10^{-1}	1.30×10^{-2}
	$\hat{\sigma}_{rrw}$	-8.31×10^{-2}	3.69×10^{-1}	-8.31×10^{-2}	3.69×10^{-1}	-1.90×10^{-1}	3.06×10^{-1}
	$\hat{\sigma}_{rr}$	3.04	2.08	–	–	5.43×10^{-1}	8.31×10^{-1}
6 hrs	$\hat{\sigma}_q$	9.36×10^{-4}	2.87×10^{-2}	9.36×10^{-4}	2.87×10^{-2}	2.01×10^{-6}	2.60×10^{-6}
	$\hat{\sigma}_{rw}$	6.83×10^{-2}	3.06×10^{-1}	6.83×10^{-2}	3.06×10^{-1}	-7.79×10^{-4}	1.23×10^{-3}
	$\hat{\sigma}_b$	3.25×10^{-1}	5.28×10^{-1}	3.25×10^{-1}	5.28×10^{-1}	1.29×10^{-1}	1.23×10^{-2}
	$\hat{\sigma}_{rrw}$	-6.79×10^{-2}	3.53×10^{-1}	-6.79×10^{-2}	3.53×10^{-1}	-1.53×10^{-1}	2.69×10^{-1}
	$\hat{\sigma}_{rr}$	2.17	1.78	–	–	4.02×10^{-1}	6.32×10^{-1}

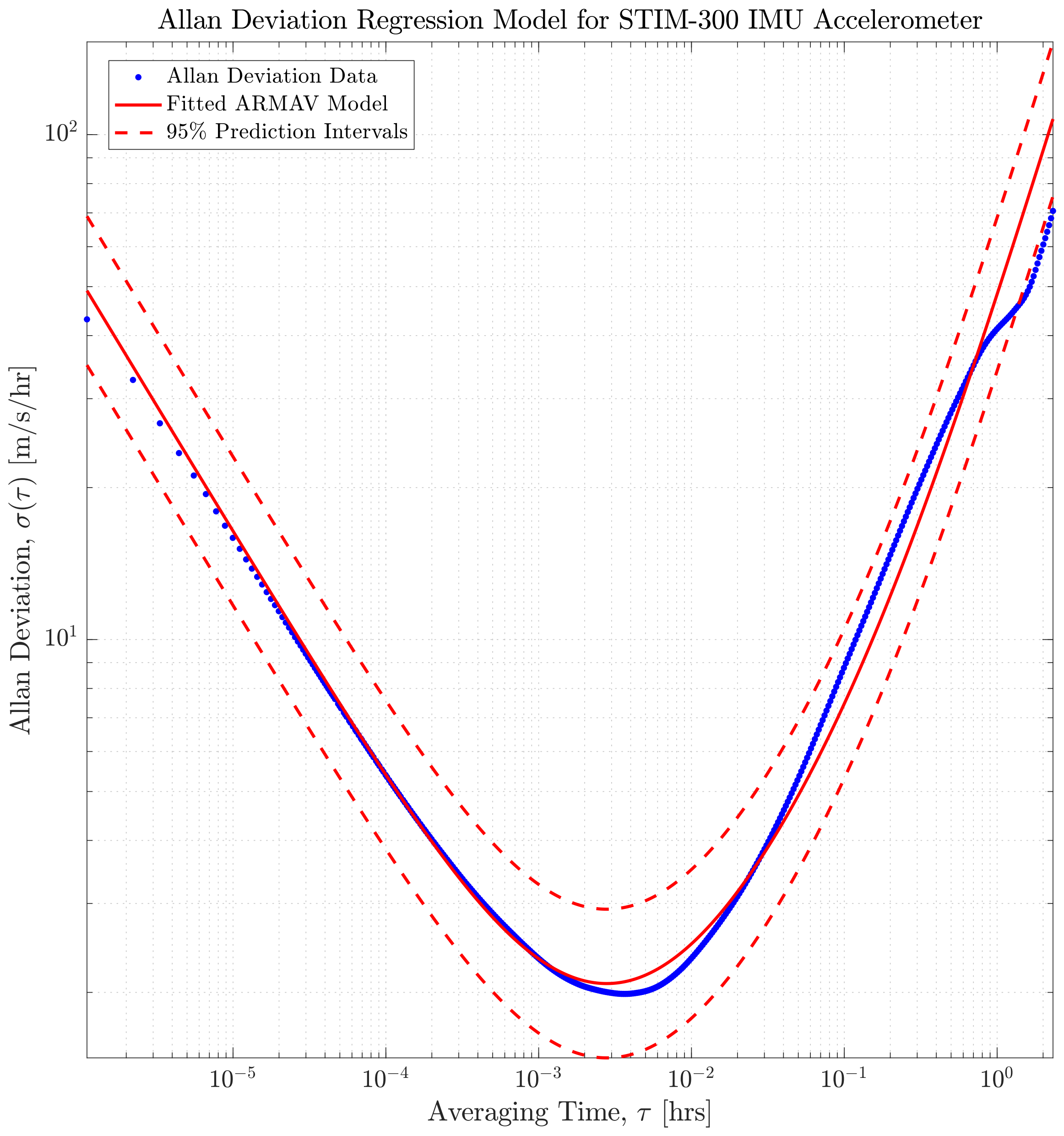


Figure 7.4: Illustration of ARMAV model on accelerometer Allan deviation measurements. As shown, ARMAV is able to accurately model the observed data with no need for human intervention, which directly enables the accurate estimation of the necessary noise strength coefficients. The resulting 95% prediction intervals cover a possible range of Allan deviation observations for each τ.

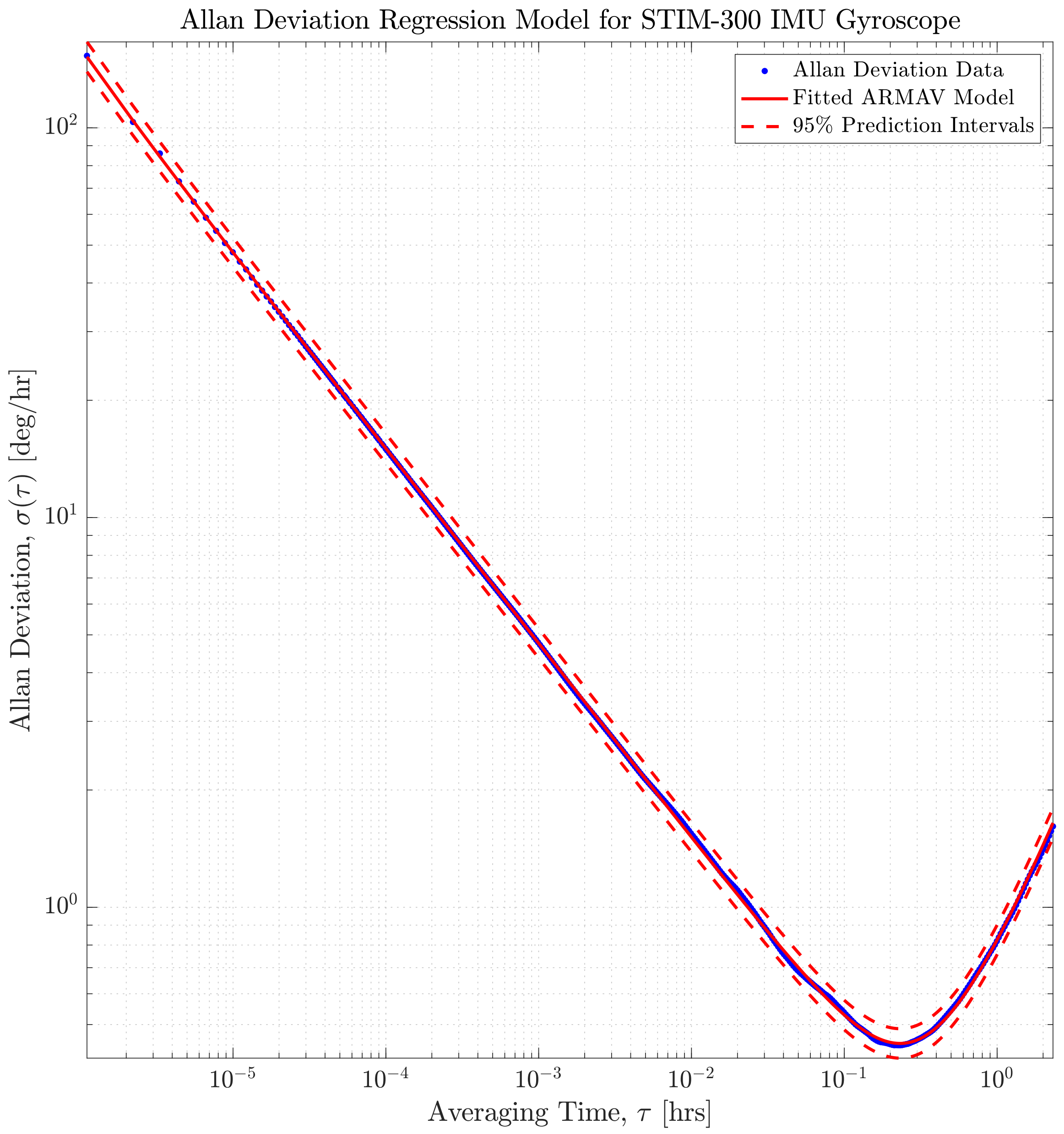


Figure 7.5: Illustration of ARMAV model on gyroscope Allan deviation measurements. As shown, ARMAV is able to accurately model the observed data with no need for human intervention, which directly enables the accurate estimation of the necessary noise strength coefficients. The resulting 95% prediction intervals cover a possible range of Allan deviation observations for each τ.

Table 7.6: Allan variance analysis results for STIM-300 IMU.

	Accelerometer				**Gyroscope**			
$\hat{\beta}$	**Slope: Auto**	**Slope: Manual**	**ARMAV**	**Spec**	**Slope: Auto**	**Slope: Manual**	**ARMAV**	**Spec**
$\hat{\sigma}_q$	7.44×10^{-5}	–	7.28×10^{-10}	–	1.61×10^{-2}	–	3.28×10^{-5}	–
$\hat{\sigma}_{rw}$	5.08×10^{-2}	5.08×10^{-2}	5.41×10^{-2}	6.00×10^{-2}	1.51×10^{-1}	1.51×10^{-1}	1.51×10^{-1}	1.50×10^{-1}
$\hat{\sigma}_b$	2.99	2.99	2.99	2.66	6.66×10^{-1}	6.66×10^{-1}	6.75×10^{-1}	5.00×10^{-1}
$\hat{\sigma}_{rrw}$	3.79×10^{1}	3.79×10^{1}	3.84×10^{1}	–	1.33	1.33	9.00×10^{-1}	–
$\hat{\sigma}_{rr}$	4.28×10^{1}	4.28×10^{1}	4.73×10^{1}	–	9.85×10^{-1}	9.85×10^{-1}	8.45×10^{-1}	–

7.6 Chapter Summary and Future Work

This chapter has proposed a novel autonomous, regression-based method for Allan variance analysis, in the context of IMU sensor calibration. The ARMAV method was shown to be generally more accurate and stable than the state-of-the-art slope method, especially when the length of available sensor data was greater than 1 hour. Additionally, ARMAV was shown to be completely autonomous and simple to program, requiring no further human input even when particular sources of noise were not present in the observed data. These findings provide significant advances in the calibration of inertial sensors as the state-of-the art method (the slope method) requires human input either via visual inspection of Allan deviance curves, or specific coding that is not transferable to other observed or updated data sets. As such, this method directly enables online or autonomous IMU sensor calibration using Allan variance, with no prior knowledge on the specific sources of noise affecting an inertial sensor of interest. Future work in this area involves the integration of the ARMAV method into an online navigation framework, where a navigation computer is able to constantly update its internal IMU model based on Allan variance analysis of observed IMU output. The implications of this integration is a continually updated, safe, accurate awareness of vehicle position, velocity, and orientation at all times.

VIII. Summary and Conclusions

As evidenced by the variety and maturity of emerging alternative navigation sensors, all-source navigation is not only becoming an imminent reality, but more importantly, it is uniquely poised to provide precision navigation systems with much-needed redundancy in GPS-challenged environments. The goal of this dissertation was to develop a means to provide navigation assurance and resiliency in the emerging all-source environment. To do so, this research detailed the development of an online autonomous and resilient sensor management framework aimed at solving a multi-faceted problem set including: all-source fault detection and exclusion, integrity monitoring, sensor initialization and validation, online sensor calibration, and online model identification.

In Chapter 3, the proposed framework, named Autonomous and Resilient Management of All-source Sensors (ARMAS), was shown to provide a breadth of sensor management functions across four modes of operation: monitoring, validation, calibration, and remodeling. Using a coherent interconnection between these modes, ARMAS was successfully shown to provide resilient and autonomous sensor management across two example multi-sensor navigation scenarios that required a combination of fault detection, sensor model validation, online calibration, and model identification, for continued operations in challenging environments.

In Chapter 4, a novel method for fault detection and isolation in all-source navigation systems was developed as the key enabler of the framework's monitoring mode. This monitoring method, referred to as Sensor-Agnostic All-source Residual Monitoring (SAARM), did not constrain faults to only biases, and provided a mechanism for detection of multiple simultaneous faults. Driven by a sensor-agnostic and fault-agnostic residual monitoring algorithm, SAARM was not only shown to perform comparably to existing RAIM techniques in the case of a single-satellite bias, but more importantly, shown to detect and isolate various types sensor model mismatches in and across multiple sensing domains such as position and velocity, without the need for synchronous or simultaneous sensor redundancy. Finally, SAARM was shown to provide a robust measure of system integrity under minimal assumptions, guaranteed to contain the true vehicle horizontal position (within a specified error bound) throughout all phases of the fault detection and identification process.

In Chapter 5, a novel method for real-time model validation for plug-and-play sensors, specifically aimed at all-source navigation applications, was developed as the key enabler of the framework's validation mode. The proposed method, referred to as Real-time Validation for Plug-and-play Sensors (RVPS), enabled the estimation of sensor-unique states without compromising the navigation solution, thereby protecting the system integrity computations during the validation period, all using a single existing filter. Equipped with the partial update implementation of the Kalman Schmidt filter, RVPS was shown not only to detect invalid sensor models more reliably than conventional residual monitoring, but additionally, prevent the detection process from corrupting the navigation solution.

In Chapter 6, a fully self-contained, pressure-airspeed-altitude ADS calibration algorithm was developed along with an accompanying autonomous smoothing spline process rooted in information theory. Driven by a BSEKF, the Jurado-McGehee Online Self-Survey (JMOSS) algorithm was shown, using real-world supersonic flight data, to model a larger portion of the Mach number domain while drastically reducing the cost, flight time, mean error, and uncertainty around the resulting model.

Finally, in Chapter 7, a novel autonomous, regression-based method for Allan variance analysis was developed. The Autonomous Regression Method for Allan Variance (ARMAV) method was shown to be generally more accurate and stable than the standard slope method, especially as the length of available data was reduced. Additionally, ARMAV was shown to be completely autonomous and simple to program, requiring no further human input even when particular sources of noise were not present in the observed data.

Bibliography

[1] "Autonomous Regression Method for Allan Variance". https://www.mathworks.com/matlabcentral/fileexchange/66462-autonomous-regression-method-for-allan-variance, July 2018.

[2] Akaike, H. "A new look at the statistical model identification". *IEEE Transactions on Automatic Control*, 19(6):716–723, dec 1974. URL https://doi.org/10.1109%2Ftac.1974.1100705.

[3] Akhlaghi, Shahrokh, Ning Zhou, and Zhenyu Huang. "Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation". *Power & Energy Society General Meeting, 2017 IEEE*, 1–5. IEEE, 2017.

[4] Allan, D. W. "Statistics of atomic frequency standards". *Proceedings of the IEEE*, 54(2):221–230, Feb 1966. ISSN 0018-9219.

[5] Arora, Jasbir. *Introduction to optimum design*. Academic Press, 2004.

[6] Atmosphere, US Standard. *US standard atmosphere*. National Oceanic and Atmospheric Administration, 1976.

[7] Bageshwar, Vibhor L, Demoz Gebre-Egziabher, William L Garrard, and Tryphon T Georgiou. "Stochastic observability test for discrete-time Kalman filters". *Journal of Guidance, Control, and Dynamics*, 32(4):1356–1370, 2009.

[8] Bates, Douglas M. and Donald G. Watts. *Nonlinear regression analysis and its applications*. Wiley series in probability and mathematical statistics. Applied probability and statistics. J. Wiley, New York, Chichester, 1988. ISBN 0-471-81643-4. Includes indexes.

[9] Bennett, William Ralph. "Spectra of quantized signals". *Bell System Technical Journal*, 27(3):446–472, 1948.

[10] Beravs, Tadej, Samo Beguš, Janez Podobnik, and Marko Munih. "Magnetometer calibration using Kalman filter covariance matrix for online estimation of magnetic field orientation". *IEEE Transactions on Instrumentation and Measurement*, 63(8):2013–2020, 2014.

[11] Bhatti, Umar I, Washington Y Ochieng, and Shaojun Feng. "Integrity of an integrated GPS/INS system in the presence of slowly growing errors. Part I: A critical review". *Gps Solutions*, 11(3):173–181, 2007.

[12] Bhatti, Umar I, Washington Y Ochieng, and Shaojun Feng. "Integrity of an integrated GPS/INS system in the presence of slowly growing errors. Part II: analysis". *GPS Solutions*, 11(3):183–192, 2007.

[13] Bhatti, Umar Iqbal. “An improved sensor level integrity algorithm for GPS/INS integrated system”. *Procceding of ION GNSS 19th International Technical Meeting*, 3012–3023. 2006.

[14] Bishop, Craig H, Brian J Etherton, and Sharanya J Majumdar. “Adaptive sampling with the ensemble transform Kalman filter. Part I: Theoretical aspects”. *Monthly weather review*, 129(3):420–436, 2001.

[15] Brenner, Mats. “Implementation of a RAIM Monitor in a GPS Receiver and an Integrated GPS/IRS”. *ION GPS-90*, 397–406, 1990.

[16] Brenner, Mats. “Integrated GPS/inertial fault detection availability”. *Navigation*, 43(2):111–130, 1996.

[17] Brink, Kevin M. “Partial-Update Schmidt–Kalman Filter”. *Journal of Guidance, Control, and Dynamics*, 1–15, 2017.

[18] Britting, Kenneth R. *Inertial navigation systems analysis.* Artech House, 1971.

[19] Brown, R Grover and Paul McBurney. “Self-Contained GPS Integrity Check Using Maximum Solution Separation”. *Navigation*, 35(1):41–53, 1988.

[20] Brumback, B and M Srinath. “A chi-square test for fault-detection in Kalman filters”. *IEEE Transactions on Automatic Control*, 32(6):552–554, 1987.

[21] Call, Curt, Mike Ibis, Jim McDonald, and Kevin Vanderwerf. “Performance of Honeywell’s inertial/GPS hybrid (high) for RNP operations”. *2006 IEEE/ION Position, Location, And Navigation Symposium*, 244. IEEE, 2006.

[22] Canciani, Aaron and John Raquet. “Absolute positioning using the Earth’s magnetic anomaly field”. *Navigation*, 63(2):111–126, 2016.

[23] Casella, George and Roger L Berger. *Statistical Inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[24] Chang, Chaw-Bing and Michael Athans. *Hypothesis testing and state estimation for discrete systems with finite-valued switching parameters.* Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ELECTRONIC SYSTEMS LAB, 1977.

[25] Chiu, Han-Pang, Xun S. Zhou, Luca Carlone, Frank Dellaert, Supun Samarasekera, and Rakesh Kumar. “Constrained optimal selection for multi-sensor robot navigation using plug-and-play factor graphs”. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 663–670, 2014. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6906925.

[26] Cho, Am, Young-shin Kang, Bum-jin Park, Chang-sun Yoo, and Sam-ok Koo. "Air data System Calibration Using GPS Velocity information". *2012 International Conference on Control, Automation and Systems International Conference on Control, Automation and Systems*, 433–436, 2012. ISSN 15987833.

[27] Curro, Joseph and John Raquet. "Navigation using VLF environmental features". *Position, Location and Navigation Symposium (PLANS), 2016 IEEE/ION*, 373–379. IEEE, 2016.

[28] De Maesschalck, Roy, Delphine Jouan-Rimbaud, and Désiré L Massart. "The mahalanobis distance". *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.

[29] Diverdi, Stephen and Jonathan T. Barron. "Geometric calibration for mobile, stereo, autofocus cameras". *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 2016.

[30] Eide, Peter and P Maybeck. "An MMAE failure detection system for the F-16". *IEEE Transactions on Aerospace and Electronic systems*, 32(3):1125–1136, 1996.

[31] Eilers, Paul HC and Brian D Marx. "Flexible smoothing with B-splines and penalties". *Statistical science*, 89–102, 1996.

[32] El-Sheimy, Naser, Haiying Hou, and Xiaoji Niu. "Analysis and modeling of inertial sensors using Allan variance". *IEEE Transactions on instrumentation and measurement*, 57(1):140–149, 2008.

[33] Erb, Russell E. *Pitot-statics Textbook*. Technical report, USAF Test Pilot School, 2015.

[34] Freescale Semiconductor, Inc. "Allan Variance: Noise Analysis for Gyroscopes". http://cache.freescale.com/files/sensors/doc/app_note/AN5087.pdf, 2015.

[35] Gainer, Thomas G and Sherwood Hoffman. *Summary of transformation equations and equations of motion used in free flight and wind tunnel data reduction and analysis*. Technical Report NASA-SP-3070, NASA, 1972.

[36] van Graas, Frank and James L Farrell. "Baseline fault detection and exclusion algorithm". *Proceedings of the 49th annual meeting of the institute of navigation*, 413–420. 1993.

[37] Gratton, Livio, Mathieu Joerger, and Boris Pervan. "Carrier phase relative RAIM algorithms and protection level derivation". *The Journal of Navigation*, 63(2):215–231, 2010.

[38] Grejner-Brzezinska, Dorota A, Charles K Toth, Terry Moore, John F Raquet, Mikel M Miller, and Allison Kealy. "Multisensor navigation systems: A remedy for GNSS vulnerabilities?" *Proceedings of the IEEE*, 104(6):1339–1353, 2016.

[39] Guerrier, Stephane, Juan Jurado, Mehran Khaghani, Gaetan Bakalli, Mucyo Karemera, Roberto Molinari, Samuel Orso, John Raquet, Christine M. Schubert Kabban, Jan Skaloud, and Yuming Zhang. "Optimal Moment Matching Techniques for Inertial Sensor Calibration (draft)". *IEEE Sensors Journal*, 2019.

[40] Guerrier, Stéphane, Jan Skaloud, Yannick Stebler, and Maria-Pia Victoria-Feser. "Wavelet-variance-based estimation for composite stochastic processes". *Journal of the American Statistical Association*, 108(503):1021–1030, 2013.

[41] Haering Jr, Edward A. "Airdata calibration techniques for measuring atmospheric wind profiles". *Journal of Aircraft*, 29(4):632–639, jul 1992. URL https://doi.org/10.2514%2F3.46212.

[42] Haering Jr, Edward A. *Airdata measurement and calibration*. Technical Memorandum 104316, NASA, 1995.

[43] Hanlon, Peter D and Peter S Maybeck. "Multiple-model adaptive estimation using a residual correlation Kalman filter bank". *IEEE Trans. Aerosp. Electron. Syst.*, 36(2):393–406, 2000. ISSN 0018-9251.

[44] Hoerl, Arthur E and Robert W Kennard. "Ridge regression: Biased estimation for nonorthogonal problems". *Technometrics*, 12(1):55–67, 1970.

[45] Hou, Haiying. *Modeling Inertial Sensor Errors Using Allan Variance*. Master's thesis, University of Calgary, September 2004.

[46] IEEE. "IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros". *IEEE Std 952-1997*, 1998.

[47] Jia, Chao and Brian L. Evans. "Online calibration and synchronization of cellphone camera and gyroscope". *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, 731–734, 2013.

[48] Joerger, Mathieu, Jason Neale, and Boris Pervan. "Iridium/GPS carrier phase positioning and fault detection over wide areas". *Proceedings of the 22nd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2009)*, 1371–1385. 2009.

[49] Joerger, Mathieu and Boris Pervan. "Kalman filter-based integrity monitoring against sensor faults". *Journal of Guidance, Control, and Dynamics*, 36(2):349–361, 2013.

[50] Johansen, Tor A., Andrea Cristofaro, Kim Sorensen, Jakob M. Hansen, and Thor I. Fossen. "On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors". *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2015. URL https://doi.org/10.1109%2Ficuas.2015.7152330.

[51] Jorris, Timothy R, Mildred M Ramos, Russell E Erb, and Reagan K Woolf. "Statistical Pitot-static Calibration Technique using Turns and Self-Survey Method". *42nd International SFTE Symposium*. Seattle, Washington, August 2011.

[52] Jurado, J. D. and J. F. Raquet. "Towards an online sensor model validation and estimation framework". *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 1319–1325. April 2018. ISSN 2153-3598.

[53] Jurado, Juan and John Raquet. "Keynote: A Common Framework for Inertial Sensor Error Modeling". *Proceedings of the ION 2017 Pacific PNT Meeting*, 725–740. Honolulu, Hawaii, May 2017.

[54] Jurado, Juan, John Raquet, and Christine M. Schubert Kabban. "Autonomous and Resilient Management of All-Source Sensors for Navigation". *Proceedings of the ION 2019 Pacific PNT Meeting*. Honolulu, Hawaii, April 2019.

[55] Jurado, Juan, Christine M. Schubert Kabban, and John Raquet. "A regression-based methodology to improve estimation of inertial sensor errors using Allan variance data". *Navigation*, 66(1):251–263, January 2019. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/navi.278.

[56] Jurado, Juan D. and Clark C. McGehee. "Complete Online Algorithm for Air Data System Calibration". *Journal of Aircraft*, 1–12, 2018/10/31 2018. URL https://doi.org/10.2514/1.C034964.

[57] Jurado, Juan D., John F. Raquet, and Christine M. Schubert Kabban. "Real-time Validation for Plug-and-play Sensors (Draft)". *IEEE Transactions on Aerospace and Electronic Systems*, 2019.

[58] Jurado, Juan D., John F. Raquet, Christine M. Schubert Kabban, and Jonathon. Gipson. "Sensor-Agnostic All-source Residual Monitoring (Draft)". *Navigation*, 2019.

[59] Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems". *Journal of Basic Engineering*, 82(1):35, 1960. URL https://doi.org/10.1115%2F1.3662552.

[60] Kauffman, Kyle J. *Radar based navigation in unknown terrain*. Ph.D. thesis, Air Force Institute of Technology, 2012.

[61] Kauffman, Kyle J. "SCORPION". https://www.afit.edu/docs/Scorpion.pdf, November 2018.

[62] Kay, Steven M. *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*. Prentice Hall, 1993.

[63] Kay, Steven M. *Fundamentals of Statistical Signal Processing, Vol. II: Detection Theory*. Prentice Hall, 1998.

[64] Keivan, N and G Sibley. "Constant-time monocular self-calibration". *International Conference on Robotics and Biomimetics*, 1590–1595, 2014.

[65] Keivan, Nima and Gabe Sibley. "Online SLAM with Any-time Self-calibration and Automatic Change Detection". *2015 International Conference on Robotics and Automation (ICRA)*, 1–8, 2015.

[66] Kerr, T. "Statistical analysis of a two-ellipsoid overlap test for real-time failure detection". *IEEE Transactions on Automatic Control*, 25(4):762–773, 1980.

[67] Kirkko-Jaakkola, M., J. Collin, and J. Takala. "Bias Prediction for MEMS Gyroscopes". *IEEE Sensors Journal*, 12(6):2157–2163, June 2012. ISSN 1530-437X.

[68] Kutner, Michael H, Chris Nachtsheim, and John Neter. *Applied Linear Regression Models*, volume 4. McGraw-Hill/Irwin, 2004.

[69] Lando, Marco, Manuela Battipede, and Piero A. Gili. "Neuro-Fuzzy Techniques for the Air-Data Sensor Calibration". *Journal of Aircraft*, 44(3):945–953, may 2007. URL https://doi.org/10.2514%2F1.26030.

[70] Lee, Young C et al. "Analysis of range and position comparison methods as a means to provide GPS integrity in the user receiver". *Proceedings of the 42nd Annual Meeting of the Institute of Navigation*, 1–4. 1986.

[71] Levinson, Jesse and Sebastian Thrun. "Automatic Online Calibration of Cameras and Lasers." *Robotics: Science and Systems*, volume 2. 2013.

[72] Lewis, Gregory V. "A Flight Test Technique Using GPS for Position Error Correction Testing". *COCKPIT, Society of Experimental Test Pilots Quaterly Publication*. March 1997.

[73] Lewis, Gregory V. *Using GPS to Determine Pitot-Static Errors*. Technical report, National Test Pilot School, 2003.

[74] Ling, Yonggen and Shaojie Shen. "High-precision online markerless stereo extrinsic calibration". *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1771–1778. IEEE, oct 2016. ISBN 978-1-5090-3762-9. URL http://ieeexplore.ieee.org/document/7759283/.

[75] Marquardt, Donald W. "An algorithm for least-squares estimation of nonlinear parameters". *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[76] Maybeck, Peter S. *Stochastic Models, Estimation, and Control Volume 1*. Navtech, Virginia, 1982.

[77] Maybeck, Peter S. *Stochastic Models, Estimation, and Control Volume 2*. Navtech, Virginia, 1984.

[78] Michalson, William R. "Ensuring GPS navigation integrity using receiver autonomous integrity monitoring". *IEEE Aerospace and Electronic Systems Magazine*, 10(10):31–34, 1995.

[79] Miura, Hiroaki, Takami Yoshida, Keisuke Nakamura, and Kazuhiro Nakadai. "SLAM-based online calibration of asynchronous microphone array for robot audition". *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 524–529. IEEE, 2011.

[80] Niewoehner, Robert Jay. "Refining Satellite Methods for Pitot-Static Calibration". *Journal of Aircraft*, 43(3):846–849, may 2006. URL https://doi.org/10.2514%2F1.18976.

[81] Novoselov, Roman Y., Shawn M. Herman, Sabino M. Gadaleta, and Aubrey B. Poore. "Mitigating the effects of residual biases with Schmidt-Kalman filtering". *2005 7th International Conference on Information Fusion, FUSION*. 2005. ISBN 0780392868.

[82] Oleshchuk, Vladimir A. "Ad-hoc sensor networks: modeling, specification and verification". *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings of the Second IEEE International Workshop on*, 76–79. IEEE, 2003.

[83] Olson, Wayne M. "Pitot-Static Calibrations Using a GPS Multi-Track Method". *28th SFTE Symposium*. September 1998.

[84] Owen, Art B. *Empirical likelihood*. Wiley Online Library, 2001.

[85] Papoulis, A and SU Pillai. *Probability, Random Variables, and Stochastic Processes*. McGaw-Hill, 1991.

[86] Parkinson, Bradford W and Penina Axelrad. "Autonomous GPS integrity monitoring using the pseudorange residual". *Navigation*, 35(2):255–274, 1988.

[87] Psiaki, Mark L. "Backward-Smoothing Extended Kalman Filter". *Journal of Guidance, Control, and Dynamics*, 28(5):885–894, sep 2005. URL https://doi.org/10.2514%2F1.12108.

[88] Radi, A., G. Bakalli, N. El-Sheimy, S. Guerrier, and R. Molinari. "An automatic calibration approach for the stochastic parameters of inertial sensors". *Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017)*, 3053–3060. 2017.

[89] Ruppert, David. "Selecting the number of knots for penalized splines". *Journal of computational and graphical statistics*, 11(4):735–757, 2002.

[90] Ruppert, David and Raymond J Carroll. "Theory & Methods: Spatially-adaptive Penalties for Spline Fitting". *Australian & New Zealand Journal of Statistics*, 42(2):205–223, 2000.

[91] Sensonor AS. "STIM 300 Inertial Measurement Unit Datasheet". http://www.sensonor.com/media/91313/ts1524.r8%20datasheet%20stim300.pdf, April 2013.

[92] Stebler, Yannick, Stephane Guerrier, Jan Skaloud, and Maria-Pia Victoria-Feser. "Generalized method of wavelet moments for inertial navigation filter design". *IEEE Transactions on Aerospace and Electronic Systems*, 50(3):2269–2283, 2014.

[93] Sturza, Mark A. "Navigation system integrity monitoring using redundant measurements". *Navigation*, 35(4):483–501, 1988.

[94] Taylor, Clark N and Shane Lubold. "Verifying the predicted uncertainty of Bayesian estimators". *Geospatial Informatics, Motion Imagery, and Network Analytics VIII*, volume 10645, 106450E. International Society for Optics and Photonics, 2018.

[95] Titterton, D. and J. Weston. *Strapdown Inertial Navigation Technology, Second Edition*, volume 207. AIAA, 2005. ISBN 1563476932.

[96] Van Loan, Charles F. "Computing Integrals Involving the Matrix Exponential". *IEEE Transactions on Automatic Control*, volume 23:3, 395–404. June 1978.

[97] Venable, Donald T. *Improving Real World Performance of Vision Aided Navigation in a Flight Environment*. Technical report, Air Force Institute of Technology WPAFB, 2016.

[98] Veth, Michael J. *Fusion of Imaging and Inertial Sensors for Navigation*. Ph.D. thesis, Air Force Institute of Technology, 2006.

[99] Waller, Joanne A, Sarah L Dance, Amos S Lawless, and Nancy K Nichols. "Estimating correlated observation error statistics using an ensemble transform Kalman filter". *Tellus A: Dynamic Meteorology and Oceanography*, 66(1):23294, 2014.

[100] Walter, Todd and Per Enge. "Weighted RAIM for precision approach". *PROCEEDINGS OF ION GPS*, volume 8, 1995–2004. Institute of Navigation, 1995.

[101] Wan, Eric A and Rudolph Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, 153–158. Ieee, 2000.

[102] Wand, Matt P. "Smoothing and mixed models". *Computational statistics*, 18(2):223–249, 2003.

[103] Wang, Ershen, Ming Cai, and Tao Pang. "A simple and effective GPS receiver autonomous integrity monitoring and fault isolation approach". *Control Engineering and Communication Technology (ICCECT), 2012 International Conference on*, 657–660. IEEE, 2012.

[104] Wilson, Heather, David Goldfein, and Kaleth Wright. "Air Force Priorities". http://www.af.mil/Portals/1/documents/SECAF/2017_Air_Force_Priorities.pdf, July 2017.

[105] Wu, ZC, ZF Wang, and Y Ge. "Gravity based online calibration for monolithic triaxial accelerometers' gain and offset drift". *Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on*, volume 3, 2171–2175. IEEE, 2002.

[106] Yang, Zhenfei and Shaojie Shen. "Monocular visual-inertial fusion with online initialization and camera-IMU calibration". *SSRR 2015 - 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2016. ISSN 15455955.

[107] Yechout, Thomas R and Keith B Braman. *Development and Evaluation of a Performance Modeling Flight Test Approach Based on Quasi Steady-state Maneuvers*. Contractor Report 170414, NASA, 1984.

[108] Young, Ryan SY and Gary A Mcgraw. "Fault detection and exclusion using normalized solution separation and residual monitoring methods". *Navigation*, 50(3):151–169, 2003.

[109] Young, Ryan SY, Gary A McGraw, and Brian T Driscoll. "Investigation and comparison of horizontal protection level and horizontal uncertainty level in FDE algorithms". *ION GPS-96*, 1607–1614. 1996.

[110] Young, Shih-Yih R and Gary A McGraw. "Method and system for fault detection and exclusion for multi-sensor navigation systems", May 15 2007. US Patent 7,219,013.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 12–09–2019	**2. REPORT TYPE** Doctoral Dissertation	**3. DATES COVERED** *(From — To)* Aug 2016–Sep 2019

4. TITLE AND SUBTITLE Autonomous and Resilient Management of All-Source Sensors for Navigation Assurance	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Jurado, Juan D., Major, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB, OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-DS-19-S-006
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally left blank.	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
All-source navigation has become increasingly relevant over the past decade with the development of viable alternative sensor technologies. However, as the number and type of sensors informing a system increases, so does the probability of corrupting the system with sensor modeling errors, signal interference, and undetected faults. Though the latter of these has been extensively researched, the majority of existing approaches have constrained faults to biases, and designed algorithms centered around the assumption of simultaneously redundant, synchronous sensors with valid measurement models, none of which are guaranteed for all-source systems. This research aims to provide all-source multi-sensor resiliency, assurance, and integrity through an autonomous sensor management framework. The proposed framework dynamically places each sensor in an all-source system into one of four modes: monitoring, validation, calibration, and remodeling. Each mode contains specific and novel realtime processes that affect how a navigation system responds to sensor measurements. The monitoring mode is driven by a novel sensor-agnostic fault detection, exclusion, and integrity monitoring method that minimizes the assumptions on the fault type, all-source sensor composition, and the number of faulty sensors. The validation mode provides a novel method for the online validation of sensors which have questionable sensor models, in a fault-agnostic and sensor-agnostic manner, and without compromising the ongoing navigation solution in the process. The remaining two modes, calibration and remodeling, generalize and integrate online calibration and model identification processes to provide autonomous and dynamic estimation of candidate model functions and their parameters, which when paired with the monitoring and validation processes, directly enable resilient, self-correcting, plug-and-play open architecture navigation systems.

15. SUBJECT TERMS
navigation, assurance, integrity, all-source, Pitot-static, Allan variance, residual monitoring, fault detection, fault exclusion, sensor model, validation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Robert C. Leishman (ENG)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	193	(937) 255-3636 x4755 Robert.Leishman@afit.edu

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18