POLITECNICO DI TORINO
UNIVERSIDAD POLITÉCNICA DE MADRID
E.T.S.I. AERONÁUTICA Y DEL ESPACIO

M.Sc. in Aerospace and Astronautical Engineering - Space
M.Sc. in Aeronautical Engineering (M.U.I.A.) - Space Vehicles

Master's Degree Thesis

Designing a Preliminary FDIR of a Hybrid IMU/GNSS Navigation Unit for Launchers

Academic Supervisors:
prof. Paolo Maggiore
prof. Matteo Davide Lorenzo Dalla Vedova
prof. Javier Cubas Cano

Professional Supervisor:
Sergio Ramírez Navidad

Author:
Massimiliano Carcia

March 2020

THIS PROJECT HAS BEEN DEVELOPED UNDER THE SUPERVISION OF SENER AEROESPACIAL SA IN COLLABORATION WITH TECHNICAL UNIVERSITY OF MADRID - ETSIAE

Ai miei genitori,
a mio fratello,
alla mia meravigliosa famiglia,
ai cari amici che ho avuto la fortuna
di trovare nel mio cammino,
per avermi aiutato ad essere
la persona che sono oggi.

"It matters not how strait the gate,
How charged with punishments the scroll,
I am the master of my fate:
I am the captain of my soul."

W. E. Henley, Invictus

Abstract

Conventional inertial-based navigation for launch solution is not error-bounded but tends to drift due to error accumulation, posing important limitations to launch duration and injection accuracy, so very expensive IMUs are needed to accomplish the required accuracy.

A promising alternative to lower navigation unit cost is to integrate inertial sensors with GNSS receiver using a Kalman filter, inside a Hybrid Navigation Unit. This allows using a lower-grade, cheaper IMU because GNSS measurements are used to correct inertial solution and reduce the errors.

The complexity of this new system and the criticality of the application generate the necessity of a Fault Detection, Isolation and Recovery system for the hybrid navigation unit. The FDIR system is an important tool to limit the effect of faults that can affect the navigation unit, making the system robust against them. The system is usually designed in a hierarchical way and it represents the last line of defence of the system. Of course, not every fault can be recovered on board, for this reason, this study has been performed with a particular software insight.

After an intense literature search, state of the art review and previous analyses, the FDIR system has been designed for a hybrid navigation unit which architecture was provided by SENER Aeroespacial. The study, design and implementation of the FDIR systems involve the complete design of the system logic and the selection and implementation of preliminary and advanced algorithms for outlier detection and recovery actions, focusing on sensors data acquisition at software level, addressing especially the faults that can be reproduced in SENER Aeroespacial simulator developed in Simulink.

The system has been integrated in SENER Aeroespacial simulator together with a Fault Injection Model designed to introduce the identified relevant faults in the navigation unit. Finally, a validation and verification campaign has been executed to properly validate and verify the algorithms implemented. The results obtained in Validation campaign certify the good performances of the methods implemented regarding GNSS part of FDIR, while the impossibility to access IMU's measurement residuals poses important challenges for outlier detection in IMU's data.

Sommario

La navigazione convenzionale per i lanciatori, basata su sistemi di tipo inerziale, tende a divergere a causa di accumulazione di errori, imponendo importanti limitazioni alla durata del lancio e alla precisione dell'iniezione in orbita, rendendo necessarie IMU molto costose per ottenere l'accuratezza richiesta.

Un'alternativa promettente per abbassare i costi dell'unità di navigazione arriva dall'integrazione di un sistema inerziale con una soluzione ottenuta dal GNSS, attraverso un filtro di Kalman, ottenendo così una unità di navigazione ibrida, che permette di utilizzare sensori inerziali di grado inferiore e ridurre gli errori attraverso l'integrazione tra sensori.

La complessità di questo nuovo sistema e la criticità dell'applicazione per cui é pensata rendono necessaria l'introduzione di un FDIR, che é un importante strumento per ridurre l'effetto di eventuali errori e rendere il sistema robusto. Il sistema FDIR é strutturato su più livelli in maniera gerarchica, e costituisce l'ultima linea di difesa dell'unità di navigazione. Naturalmente, non tutti gli errori possono essere recuperati attraverso azioni correttive a bordo del lanciatore, perciò quest'analisi é stata eseguita con particolare attenzione agli aspetti software.

Dopo una lunga ricerca in letteratura riguardo lo stato dell'arte e alcune analisi previe, é stato progettato un sistema FDIR per una unità di navigazione ibrida per lanciatori basata su un architettura proposta da SENER Aeroespacial. Lo studio effettuato include la progettazione completa della logica e la selezione ed implementazione di algoritmi semplici ed avanzati per il rilevamento di outliers e per il recupero del sistema, focalizzando l'attenzione sull'acquisizione dei dati provenienti dai sensori e sugli eventuali errori che é possibile riprodurre nel simulatore sviluppato in SENER Aeroespacial.

Il sistema é stato integrato con successo nel simulatore, insieme ad un modello di Fault Injection progettato per introdurre nell'unità di navigazione gli errori rilevanti per questa analisi. Infine, una campagna di validazione degli algoritmi é stata eseguita. I risultati di queste simulazioni mostrano le ottime prestazioni della parte relativa al GNSS, mentre l'Impossibilità di accedere ai residui dei dati della IMU ha causato non pochi problemi riguardo il rilevamento di outliers in detti sensori.

Acknowledgements

It's not always easy to make a balance of your experiences at the end of a long road and take some time to thank properly, also because I would need another chapter of this Master Thesis just to thank a huge number of people. But someone told me I have to include some technical stuff in this Thesis, so I will try to be short and synthetic without forgetting about anyone.

First, I would like to thank my parents: for all their sacrifices that allowed me to study at Politecnico di Torino and abroad, at UPM; thank you for believing in me and for supporting me throughout all my academic career, in good and bad moments, but most of all, thank you for teaching me important virtues and values and for to face all the experiences giving always my best. You are extraordinary people and I hope I payed you back, at least in part, with my academic results.

A special thank to my brother, constant role model and encouragement to aim for the greatest and to be the best version of myself, but mostly a great friend. Thanks to his excellent academic results, he gave me the desire to always go beyond and out of my way to overcome him, he supported my choices and he has always been my guiding light.

A huge THANKS to my grandparents. They are the best in the world, and I am very lucky to have them in my life. They have been a very important part of my life since I was a child. They taught me a lot of things and they were like parents to me. No matter the distance and the age, I will always be your boy and I hope I could pay you back for all your love and all the Nutella you bought for me because you said it's good for the brain.

Also, thanks to my uncles Domenico, Pinuccio, Mary Rosa and to my little cousin Vincenzo, wishing him to have a wonderful life: thank you for your advises, your support and for being always there for me. Thanks to Michela, Alfonso, Zia Giovanna and Flavia and Ludovica, for keeping me in your thoughts.

Thanks are due to my professional tutor Sergio, for his encouragement and support to give my best in writing this Master Thesis, but mostly for being a good friend. Thanks also to Francesco, Antonio, Mayte, Salva, Jorge, Santi, Davíd, Javi and everybody else at SENER Aeroespacial, for making my experience amazing and very interesting. Working in a leader company in Space sector like SENER Aeroespacial has been a very useful experience to understand the inner flows and processes of a real project and it has been

very challenging to deal with strict deadlines and requirements. The experience gained during this 8-months period and the possibility to be part of an important reality represent the perfect culmination of a wonderful educational path and I am very satisfied and proud of it.

I would like to thank Federica, for supporting me and encouraging me to always believe in myself and in my dreams, even when I did not. You are really special and the time we shared made me a better person.

Last but not least, a heartful thank you to all my friends, near and far; particularly thanks to Federica, Simone, Laura, Dario, Francesco Pio, Luca, Francesca, Manuel, Giuseppe, Patricia, Marita, Pablo, José, Vincenzo, my teammates of Fatine Volanti, my "flatmates" at Collegio Einaudi, my wonderful friends from high school and every single person who stayed by my side. Friends are the family you choose, and I feel very lucky to have such fantastic people in my life, which make me better giving me the best gift of all: their friendship (and also something more). I always keep you in my mind.

P.S. A special thanks to my friend and collegue Vincenzo, who drove me every day on the adventurous highways near Madrid, towards Tres Cantos. Thank you for our journeys, good musical choices and for helping me moving my stuff.

Ringraziamenti

Non sempre è facile, alla fine di un lungo percorso, tirare le somme e fare dei (dovuti) ringraziamenti, anche solo perché le persone da ringraziare sono davvero tante e un capitolo intero di questa tesi sarebbe necessario solo per farlo. Ma siccome bisogna lasciare qualche pagina di contenuti tecnici (o almeno così mi è stato detto), proverò ad essere sintetico e a non dimenticare nessuno.

Vorrei ringraziare in primis i miei genitori: per tutti i sacrifici che hanno fatto per "mandarmi" a studiare prima al Politecnico di Torino e poi alla Universidad Politécnica de Madrid; per aver sempre creduto in me e per avermi sostenuto, supportato e spronato per tutta la mia carriera accademica, nei momenti belli (per fortuna, tanti) e in quelli più bui (sempre per fortuna, pochi); ma soprattutto per avermi trasmesso valori importanti ed avermi insegnato ad affrontare la vita dando sempre il massimo. Sono delle persone straordinarie e spero di averli ripagati, almeno in parte.

Un grazie speciale va poi a mio fratello, costante esempio da seguire e sprone a puntare sempre al massimo e ad essere la migliore versione di me stesso, ma anche e soprattutto un grande amico. Grazie ai suoi eccellenti risultati accademici mi ha dato la possibilità di voler andare sempre oltre e a dare qualcosa in più, mi ha sostenuto nelle mie scelte ed è stata la mia guida durante questo percorso.

Ai miei nonni va un enorme, sconfinato GRAZIE. Sono i nonni migliori del mondo, e mi sento molto fortunato ad averli conosciuti e ad essere uno dei loro *"civiloni"*. Sono stati una parte importantissima della mia vita, sin dall'infanzia. Mi hanno insegnato moltissime cose e sono stati dei secondi genitori per me. Nonostante la distanza e l'età, sarò sempre il vostro *"criaturo"*, e spero di ripagarvi, un giorno, di tutto il vostro amore, di tutti i pranzi della domenica, di tutti i regali e anche di tutta la Nutella che mi avete comprato, perché "fa bene al cervello".

Grazie a Zio Domenico, Zio Pinuccio e Zia Mary Rosa e a Vincenzino, a cui auguro una vita felice e piena di soddisfazioni; grazie per il vostro supporto, i vostri consigli e per esserci sempre. Grazie a Michela, Alfonso, Zia Giovanna e le mie cuginette Flavia e Ludovica per avermi tenuto sempre nei vostri pensieri.

Un ringraziamento doveroso va al mio tutor aziendale Sergio, per avermi spronato a dare il massimo e per avermi aiutato con io suoi consigli durante la scrittura di questa tesi, ma

soprattutto per essere stato un buon amico. Grazie anche a Francesco, Antonio, Mayte, Salva, Jorge, Santi, Davíd, Javi e a tutte le altre persone che hanno reso la mia esperienza in SENER Aeroespacial davvero fantastica, divertente e interessante. Avere l'opportunità di lavorare in un'azienda leader nel settore spaziale come SENER Aeroespacial ha rappresentato per me un'esperienza davvero utile per comprendere i processi e lo sviluppo di un progetto reale ed è stato molto stimolante dover affrontare tempi di consegna e requisiti molto stretti. L'esperienza che ho guadagnato durante questi 8 mesi e la possibilità di essere parte di un'importante realtà costituisce il culmine di un fantastico percorso di studi di cui sono molto soddisfatto ed orgoglioso.

Vorrei ringraziare Federica, per avermi sempre supportato e incoraggiato a credere in me stesso e nei miei sogni, anche quando ci credevo poco io stesso. Sei davvero speciale e il tempo trascorso insieme mi ha reso una persona migliore.

Infine, ma non per questo meno importante, un grazie di cuore a tutti i miei amici, lontani e vicini, in particolare a Simone, Laura, Dario, Francesco Pio, Luca, Francesca, Manuel, Giuseppe, Patricia, Marita, Pablo, José, Vincenzo, i miei compagni di squadra delle Fatine Volanti, i miei "coinquilini" del Collegio Einaudi di Torino, i miei splendidi amici e compagni del liceo e tutte le altre persone che mi sono state vicine. Gli amici sono la famiglia che ti scegli e io sono stato fortunato a trovare sulla mia strada delle persone fantastiche, che mi hanno accettato con i miei difetti e mi hanno reso una persona migliore facendomi il dono più grande: la loro amicizia (e anche molto di più). Vi porto sempre con me.

P.S. Un ringraziamento speciale al mio amico e collega Vincenzo, che mi ha scarrozzato in giro per le avventurose autostrade di Madrid e dintorni in direzione Tres Cantos, compagno di mille viaggi, buone scelte musicali e traslochi improbabili.

Contents

List of Figures

List of Tables

List of Acronyms

ACC	Accelerometers
BIT	Built-In Tests
CBIT	Continuous Built-In Tests
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CUSUM	Cumulative Sum
DKE	Dynamics, Kinematics and Environment
EDAC	Error Detection and Correction
eEKF	error-state Extended Kalman Filter
EKF	Extended Kalman Filter
FA	False Alarm
FDI	Fault Detection and Isolation
FDIR	Fault Detection, Isolation and Recovery
FIM	Fault Injection Model
FMEA	Failure Modes Effect Analysis
FMECA	Failure Modes Effect and Criticality Analysis
FTA	Fault Tree Analysis
FW	Firmware
GDOP	Geometric Diluition of Precision
GLR	Generalize Likelihood Ratio
GNSS	Global Navigation Satellite System
GPS	Global Positioning System

GYR	Gyroscopes
HSE	High Severity Error
I/F	Interface
IBIT	Initialization Built-In Test
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KDE	Kernel Density Estimator
KF	Kalman Filter
k-NN	Nearest Neighbour
LEOP	Launch and Early Orbit Phase
LSE	Low Severity Error
MC	Montecarlo
MD	Misdetection
MGMT	Management
NU	Navigation Unit
OBC	On-Board Computer
OBSW	On-Board Software
ODE	Ordinary Differential Equation
ODR	Outlier Detection and Removal
OTR	Out of Range
PAM	Partition Around Medoids
PBSW	Processing Board Software
PC	Predictor-Corrector algorithm
PVT	Position-Velocity-Time
RAMS	Reliability, Availability, Maintanability and Safety
RF	Radiofrequency
RTOS	Real Time Operating System
SW	Software
TM	Telemetry

TC Telecommands

V&V Validation and Verification

WCET Worst Case Execution Time

Chapter 1

Introduction

Since the beginning of space flight, the first and greatest problem to face has been the development of a safe, reliable and affordable way to reach space to deliver every kind of payload in the right orbit. With our society relying more and more on several space services such as navigation and meteorology (public services), telecommunication (commercial services), Earth observation and technology demonstration (military and scientific services), there is a constantly growing need towards economically feasible launch systems, also because of the growing market of small satellites. In fact, according to SpaceWorks forecast for 2018-2022 (see [1]), in the next few years satellite launch market will experience a significant grow regarding launches of micro-nano satellites in the range 1-50 kg (around 2600 new launch opportunities). For this reason, space agencies and organization from all around the globe are trying to develop dedicated small launch vehicles to provide easy, reliable and affordable access to space. This can be obtained with the help of modern and out-of-the-box solutions, especially in structure production and flight systems and avionics fields. Focusing on this last, navigation units are very expensive systems due to the high precision needed to properly deliver the payload. Therefore, the need to obtain a cheaper navigation unit, maintaining performances level and accuracy to high standards, arises to lower launch recurrent cost and make launchers affordable. In this frame, the development of a hybrid navigation unit is a fundamental step towards the achievement of this purpose.

1.1 Why do we need a hybrid navigation unit?

Currently, during ascent to space, navigation estimation of the state vector (position, velocity and attitude) is provided by a traditional inertial-only navigation system (INS). The INS is a complete 3D dead-reckoning navigation system based on an Inertial Measurement Navigation (IMU), that is a set of inertial sensors, and a navigation processor. IMU essentially comprises gyroscopes and accelerometers that measure angular rates and specific force accelerations. Starting from initial attitude, position and velocity, angular rate is

numerically integrated by the navigation processor to obtain attitude and acceleration is numerically integrated twice to obtain position and velocity, as shown in 1.1.

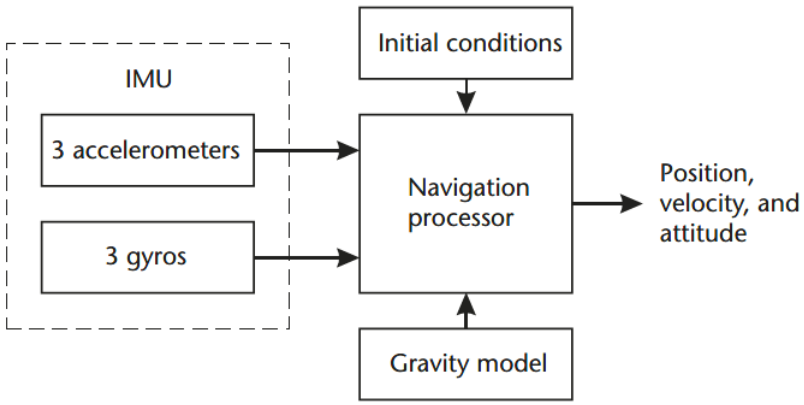


Figure 1.1: Inertial Navigation System (INS) basic scheme [2]

Thanks to the fact that IMU's sensors do not depend on an external inertial reference, they provide a good solution against interference and jamming. Nevertheless, this kind of navigation system imposes severe limitations to launch execution, because navigation solutions continuously drift due to error accumulation (during integration) that is not corrected, setting limitations on launch duration and orbit injection accuracy. In return, this can cause an increase in LEOP duration and cost due to poor orbit injection and reduce mission life of months or years. For these reasons, very accurate and precise (and, of course, very expensive!) IMUs are needed to achieve strict mission positioning requirements, without considering that return and landing of non-expendable launchers is practically unfeasible.

A clear way to correct the error due to navigation solution drift is to update it integrating IMUs measurements with another sensor's measurements, to reduce the error, allowing to use a lower-grade IMU, reducing the cost of the navigation unit while maintaining performances. The most suitable way to update the navigation estimate is using Global Navigation Satellite Systems (GNSS) measurements, developing a Hybrid IMU/GNSS navigation unit and exploiting the strengths of both sensors to counter their drawbacks. Sensors measurement coupling allows to reduce the effect of errors produced by each of the sensor and to improve error estimation and accuracy, providing robustness and redundancy. The main tool to integrate several sensors is the Kalman filter.

Several hybrid navigation units, involving not only IMU and GNSS measurements, have already been used in land, maritime, aeronautical and space applications (see [2]). For example, ships use INS and GNSS coupling with a magnetic compass for heading calibration, helicopters use Doppler radar (like trains) as well as INS coupled with GNSS and, of course, commercial and most military aircraft use INS and GNSS navigation systems combined with several other sensors. Regarding spacecraft applications, position and velocity

are determined by force models occasionally updated with GNSS measurement (only in Low Earth Orbit) or tracking from Earth ground stations. This is more accurate than using INS/GNSS navigation system, although several units that used this combination have already been flown in orbit, like German spacecraft BIRD and Swedish demonstrator PRISMA (see [3]).
European space launch operators are currently focusing on GNSS applications for accurate estimation of position and velocity of launch vehicles, especially after promising experiments carried out on European launch vehicles Vega and Ariane. Vega launch vehicle currently uses a COTS GPS receiver as part of its Autonomous Localization and Telemetry Subsystem for localization purposes, while OCAM-G experiment flew on Ariane 5, and both demonstrated that GNSS can provide very accurate estimations of position and velocity during most of the flight, showing its potential as future part of a hybrid IMU/GNSS navigation system for launchers (see [4]).
Most recently, several concepts of hybrid IMU/GNSS navigation systems have been proposed and tested (see [5]), e.g. DLR has successfully designed and flight-tested the Hybrid Navigation System HNS (see [3]) on board of SHEFEX-2 experiment.

Given this background, it is clear that hybrid navigation represents the future of navigation system for launchers, to lower recurrent cost and improve accuracy. Nevertheless, hybridisation and coupling of several sensors present many aspects to which attention must be paid:

- Proper modelling of the system
- Coupling algorithm complexity
- Tuning and Verification campaigns
- Dealing with multiple sensors and their errors

1.2 Importance of FDIR system

A failure in navigation system, and in particular INS-related failure can result in catastrophic or major consequences, as shown by past events. In particular, the most emblematic episodes are:

- The failure of the INS of Chinese CZ-3B launcher in 1996, causing the loss of the payload, huge damage to properties and several deaths.
- The failure of Ariane 5 maiden flight in 1996 due to an INS-related failure, causing the loss of the mission.
- Failed landing of ESA's Schiaparelli EDM (Entry, Descend and Landing Demonstrator Module), part of 2016 ExoMars mission due to IMU saturation and "inadequate handling of IMU saturation by GNC, insufficient FDIR approach and design robustness".

Thus, it is clear that navigation unit is a critical unit for the achievement of mission goals. For this reason, several modern and bigger launch vehicles include redundant IMU's or internal redundancy at sensors level.

Even more critical is the application of hybrid navigation, which implies, of course, that more complicated algorithms need to be implemented, increasing system complexity and fault probability also due to the use of several sensors. To compensate for this increase in fault probability and provide a reliable navigation solution, it is necessary to introduce a robust system able to detect and identify failures and to recovery/reconfigure the navigation system, reducing the probability of failure. This system, that is the object of this Master Thesis, is the Fault Detection, Isolation and Recovery (FDIR) system.
Being the navigation system a flight-critical application, Reliability, Availability Maintainability and Safety (RAMS) recommendations assume an important role in developing FDIR system and performing integrity monitoring to improve navigation unit performances; furthermore, it is a must in parameter estimation and it represents an unsolved, difficult problem, so there is a lot of literature about this active field of research. Nevertheless there are no common procedures or standards to follow for FDIR activities planning and implementation. For this reason, there is a plurality of FDIR techniques, but two main approaches can be identified, depending on mission characteristics: to prevent and contain faults before they occur or to manage them after they occur through detection, identification and response activities.

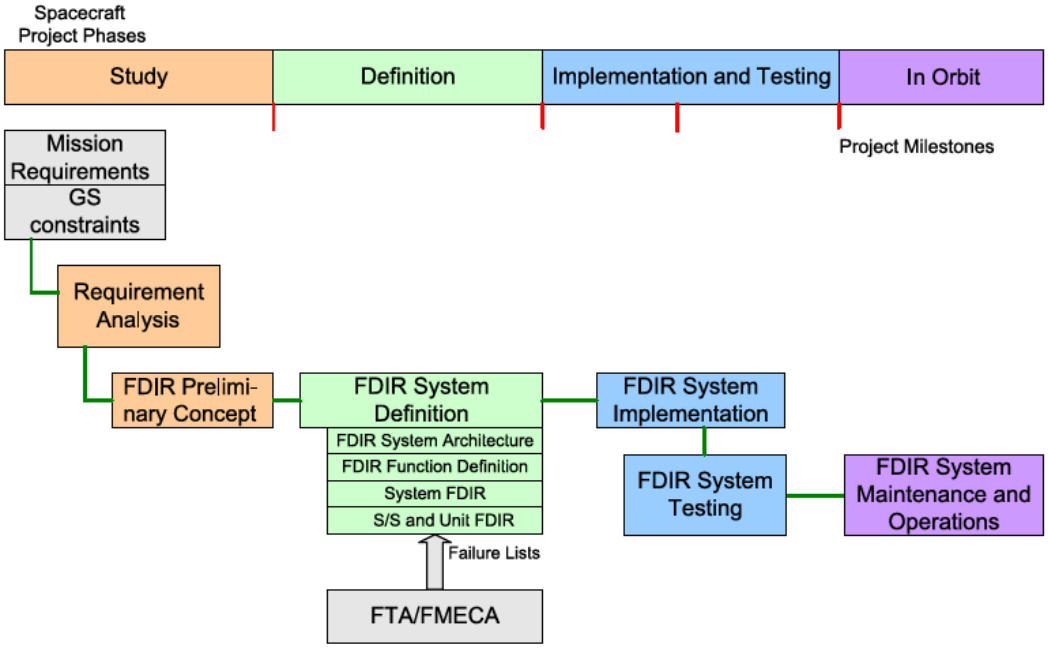


Figure 1.2: FDIR development phases vs project phases [6]

In general, the development of an FDIR system is an activity that should be carried out since the very beginning of the project, starting from preliminary information at high

level and extending then to lower levels when information is available, all the way to final concept and design (as shown in Figure 1.2 from [6]), because a late or restricted implementation drastically reduces its potential in improving performances.

1.3 Purpose of the Thesis

The main purpose of this Master Thesis is the design and implementation of Fault Detection, Isolation and Recovery algorithms for a hybrid navigation unit for launchers. For the particular case analyzed, the FDIR system must deal with multiple sensors (IMU and GNSS), their integration and the faults that could happen to measurements provided by all of them. Furthermore, FDIR system must deal with complexity of Kalman filter algorithms, that are used to integrate the measurement of the different sensors, by increasing or decreasing filter parameters in case of a sensor fault or detecting and removing faulty measurements from the computation. Among FDIR tasks in a navigation unit, there is also the management of redundant components, e.g. switching from a faulty IMU to a redundant one in case of fault detection. Also, the redundant units can be used as independent solution to detect and isolate faults in sensors, or they can be used as navigation safeguard.

Of course, not every fault is addressed to on-board detection or recovery, for this reason, it is essential to perform some previous analysis to identify the failures subject to detection and isolation, for example, Failure Mode Effect Analysis and/or Fault Tree Analysis. For this particular case, only software-related aspects, ranging from previous analysis to approach, design and implementation of a preliminary FDIR algorithm, are covered by this thesis work.

1.4 Scope of the Thesis

In this section, objectives and limitations of the project are described. Original requirements evolved during the project due to further investigation and consultations with experts of the field and according to results obtained.

The following goals and limitations of this work must be remarked:

- The work developed focus on integrity monitoring of navigation solution and on the actions that can be undertaken to provide a reliable and accurate navigation solution.
- The focus is centered on a software analysis, not dealing with hardware failures that can't be reproduced in the simulator. One of the objectives of the previous analyses performed is to identify the errors that can be reproduced in the hybrid navigation simulator developed by SENER Aeroespacial and that can be recovered on-board.

- An intensive study of the system has been carried out, through several analysis, with the aim of deeply understanding the features of the unit, highlighting the critical items and the possible weak spots to which attention must be paid.
- The designed system is tailored for a navigation unit whose design and characteristics have been defined and fixed by SENER Aerospacial. For this reason, some limitations had to be taken into account: first, the necessity of integration of FDIR system inside an already developed architecture; second, the lack of sensor redundancy, neither internal nor external, limiting the implementation of some of the solutions found after a wide and intensive literature search.
- For the development of the system, Matlab/Simulink has been used as main tool, with the aim of integrating the FDIR system in the hybrid navigation simulator developed by Sener Aeroespacial SA.

1.5 Outline of the Thesis

The outline of the Master Thesis is the following:

Chapter 1 contains introduction and motivation information for this work.

Chapter 2 provides background information about hybrid navigation unit, with the aims of:

- outlining types, strengths and weaknesses of the different types of sensors (IMU and GNSS) involved in the hybrid navigation units, and outlining advantages and drawbacks of several hybrid architectures;
- introducing the algorithms that allow the integration of the two systems;
- defining FDIR principal characteristics and functions.

Chapter 3 contains all the previous analysis necessary to properly develop a functioning FDIR algorithm, in particular, state of the art review and fault-related analysis to identify the faults which the FDIR system is addressed to.

Chapter 4 contains a detailed description of the consideration made during design and implementation phases, with particular attention to the FDIR system algorithm description, FDIR logic and architecture design, and model interfaces description.

Chapter 5 discusses validation campaign that FDIR system underwent and the system simulator used, including the design and verification of a fault injector model developed to introduce the relevant faults in the simulator provided by SENER Aeroespacial.

Chapter 6 represents the final part of this work, drawing the conclusions reached, the lesson learnt and possible future works.

Chapter 2

Background

This Chapter provides useful background information about how a hybrid navigation unit works, an its use on board launch vehicles, in order to describe system functionalities in view of FDIR system design. In particular, in Section 2.1 the different architectures of the current navigation equipment are discussed, with an insight on Inertial Navigation and Satellite Navigation, regarding their strengths and weaknesses, navigation equations and sensors' errors. In Section 2.2 the main instrument to perform IMU/GNSS integration is presented: the Kalman filter is described in this Section, outlining the features of this estimation method and the various types of filter that can be implemented. Finally, Section 2.3 gives an overview of FDIR system, its functions and tasks, focusing on its use within a navigation system for a launch vehicle.

2.1 IMU/GNSS Hybrid Navigation

Inertial navigation and satellite-based navigation systems provide different navigation solutions, each with its advantages and drawbacks but they are particularly suitable for hybridisation and integration because their benefits and weaknesses are complementary. This will be better outlined in Section 2.1.3, but in order to understand hybrid navigation's added value, it is necessary to introduce the features of INS and GNSS.

2.1.1 Inertial Navigation features

Inertial navigation systems are based upon inertial sensors, which comprises accelerometers and gyroscopes, that measure specific force acceleration and angular rate respectively, both without an external reference. To obtain a valid three-dimensional independent navigation solution, at least three orthogonal gyroscopes and three orthogonal accelerometers are needed. Typically, they are integrated with a processor inside an IMU. There are two basic types of inertial systems: stable platform systems and strapdown systems.

Stable platform systems consists in accelerometers and gyroscopes mounted on a platform isolated from rotation, using gimbals; any kind of rotation is detected by gyroscopes and then this signal is used to rotate the gimbals (through some motors) to cancel such rotations and align platform frame and inertial frame.
On the other side, in strapdown systems, accelerometers and gyroscopes are fixed and have the advantage of reducing complexity and dimensions of the system, for this reason, these systems represent the most used type of INS. Angular rate measured by gyroscopes are integrated and used to rotate specific force acceleration into inertial frame, then velocity and position are obtained through double integration after correcting for local gravity acceleration, as shown in Figure 2.1

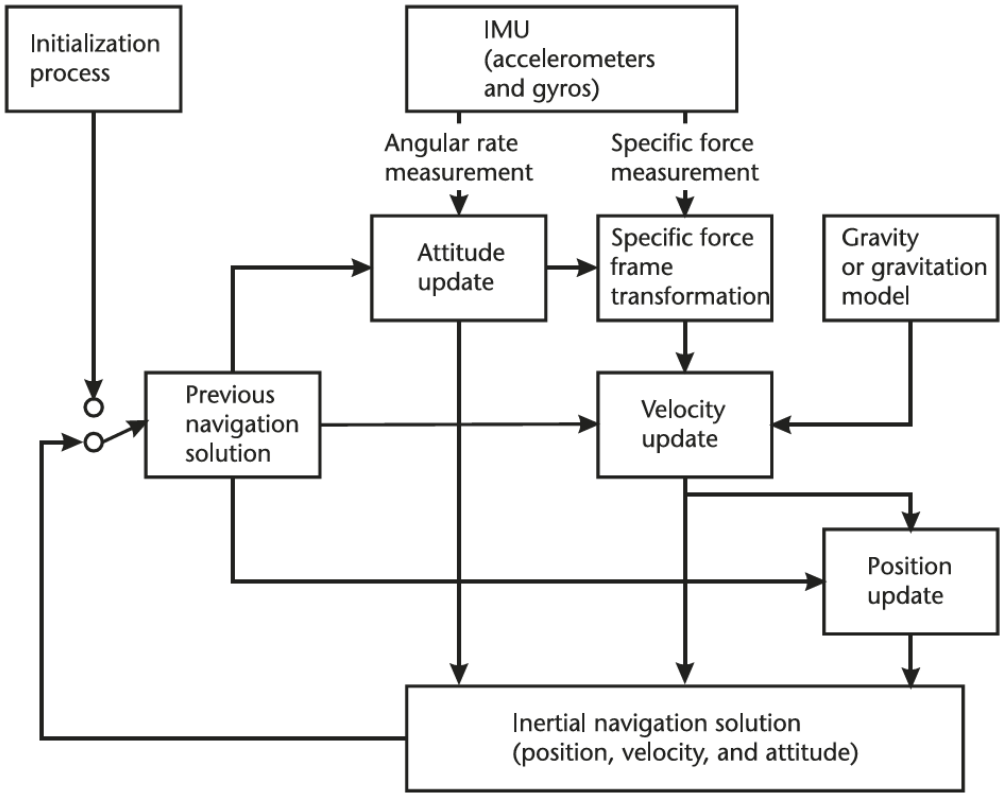


Figure 2.1: Inertial Navigation System operation [2]

Navigation equations are presented in (2.1). They are expressed in an inertial reference frame but could be expressed in other reference frames:

$$\begin{aligned} \dot{q}_{I\leftarrow B} &= \tfrac{1}{2} q_{I\leftarrow B} \odot \begin{pmatrix} \omega_B \\ 0 \end{pmatrix} \\ \dot{\vec{r}}_I &= \vec{v}_I \\ \dot{\vec{v}}_I &= \mathbf{R}_{I\leftarrow B}\, \vec{a}_B + \vec{g}_I(\vec{r}_I) \end{aligned} \tag{2.1}$$

where the subscript "B" is referred to magnitudes in Body reference frame, i.e. magni-

tudes measured by IMU's accelerometers and gyroscopes and the operator $\odot$ represents quaternion multiplication.

- The vector $\vec{g}_I(\vec{r}_I)$ represents gravitation acceleration obtained using a gravity model depending on the position.
- $q_{I \leftarrow B}$ is the quaternion that expresses the rotation from Body to Inertial frame.
- $\mathbf{R}_{I \leftarrow B}$ represents the rotation matrix from Body to Inertial frame.

Regarding the propagation algorithm, it is important to make some considerations. First, angular rates are provided by gyroscopes in the form of angular increments, so they have to be intended as a mean integral of instantaneous angular rate over the time interval between two gyroscope readings. The same applies to accelerometers: non-gravitational accelerations are provided as velocity increments and have to be intended as mean integral of continuous acceleration over the time interval between two accelerometer readings.

Second, both gyroscopes and accelerometers present several types of error, in particular biases, scale factors, misalignment errors and random noises. Each of these has several contributions, in particular: a fixed contribution, a temperature-dependent contribution, a run-to-run contribution and an in-run contribution. Fixed contribution, temperature and run-to-run effect can be corrected by calibration and initial alignment. In-run effect can only be estimated during navigation using additional sensors, but usually, they are difficult to estimate. Also, gyroscopes may be sensitive to accelerations. A brief description of every type of error is presented in Table 2.1

Table 2.1: Error characteristics of IMU's sensors

Error	**Type**	**Features**
Biases $\vec{b}$	Constant error presented by accelerometers and gyros	Static bias: fixed+run-to-run Dynamic bias: in-run (10% static)
Scale Factors **s**	Proportional errors	Due to the fact that input-output gradient of the sensor is not 1
Misalignment **M**	Cross coupling errors due to misalignment of the sensitive axes with respect to body frame	Due to manufacturing limitations. Represented as out-of-diagonal terms of a matrix which presents scale factors on the diagonal
g_sensitivity $\mathbf{a_g}$	Biases presented by gyroscopes	Sensitivity to non-gravitational acceleration due to mass unbalance
Random noise η	Approximately white noise	Due to several causes, cannot be calibrated or corrected.

So, the error characteristics of each sensor's measurement can be expressed as shown in Equation (2.2) and (2.3), respectively for accelerometers and gyroscopes.

$$\tilde{a}_B = [\mathbf{I} + \mathbf{diag}\{\mathbf{s_a}\} + \mathbf{M_a}]\,\vec{a}_B + \vec{b}_a + \eta_a \tag{2.2}$$

$$\tilde{\omega}_B = [\mathbf{I} + \mathbf{diag}\{\mathbf{s_g}\} + \mathbf{M_g}]\,\vec{\omega}_B + \vec{b}_g + \mathbf{diag}\{\mathbf{a_g}\}\vec{a}_B + \eta_g \tag{2.3}$$

In both previous equations, the left term represents effectively sensed magnitudes while the right term represents the error added to real magnitudes during measurement.
Further error can be included in the analysis like random walk errors or vibration effects. Because of these errors, INS solution drifts from correct solution.

2.1.2 GNSS features

Global Navigation Satellite Systems are systems based on radio ranging signals from orbiting satellite that provide user with positioning service. Several of these systems are currently available, the most famous of which is undoubtedly the Navigation by Satellite Timing and Ranging Global Positioning System (NAVSTAR GPS), operated by U.S. Government, followed by the Russian GLONASS and the European Galileo. Nevertheless, the basic architecture of the system is the same for each one of them. The main components of this architecture are:

- **Space segment:** it comprises the constellation of satellites, properly positioned in space to maximize global coverage, which broadcast radio signals to users and ground segment.
- **Ground segment:** it comprises monitor stations, which are meant to calibrate satellite clocks and report ranging measurement to control stations, which, in turn, generates navigation data messages to be sent to satellites and performs eventual manoeuvres.
- **User's equipment:** it comprises an antenna that receives the signal and convert it to electrical signals, a receiver that demodulates signals and attach a time-tag to the measurement, a ranging processor that calculates pseudo-ranges (which is the measured range in presence of clock errors), pseudo-range rates (or Doppler shifts) and accumulated delta range, and finally a navigation processor that outputs navigation solution in the form of a PVT (Position Velocity Time) solution.

Each satellite broadcasts a signal that includes ranging codes, that allows the receiver to know the instant of the transmission and navigation messages, which gives important information about timing parameters and satellite orbit.

It is known that at least 4 ranging signals from 4 different satellites are needed to provide positioning service. This is due to the fact that using only one ranging signal the locus of

user's positions is a sphere of radius ρ, centred on the satellite; using two signals, it reduces to the circle produced by the intersection of two spheres and using three signals, it reduces to two separate points. Then another ranging signal is needed to remove ambiguity. This is also evident because there are 4 unknowns in the problem: three spatial coordinates representing the user's position and user's receiver instant of reception. In fact, satellite position is known from navigation messages information about satellite orbit (ephemeris) and the instant of transmission is also known. Therefore, at least 4 equations are needed to determine a user's position.

GNSS also provide velocity measurements obtained from pseudo-range rate (by measuring Doppler shift in radio frequency carrier and knowing satellite velocity from ephemeris) through several algorithms.

There are several possible sources of error in GNSS measurements, for instance, clock errors and inaccuracy in broadcast ephemeris, delays caused by ionosphere and troposphere, noises, external interference and jamming and multipath. Also, several other factors could prevent the user from receiving the 4 signals needed. Nevertheless, GNSS can provide positioning service with an accuracy up to a few meters and very accurate velocity measurements.

As said before, GPS is the most famous of these kind of systems. It comprises a nominal constellation of 24 satellites up to a maximum of 36 satellites (all of them are active and working at the same time) at a radius of 26600 km, in six orbital planes inclined approximately at 55°with 60°of separation in longitude. The satellites are not equally spaced in the orbital planes, because the current disposition minimize the effect of a satellite outage and to guarantee that at least 5-6 satellites are visible in any instant of time, assuming a clear line of sight.
GPS provides different services depending on the level of precision required, using 10 different types of signals.

GLONASS was developed by URSS and then by Russia as a military navigation system. It comprises 24 satellites and 3 active spares at a radius of 25600 km, equally spaced in 3 orbital planes separated by 120°of longitude.

Galileo positioning system is a civil navigation system developed by the European Union. It comprises 30 satellites, of which 3 are spares, at an orbital radius of 29994 km placed in 3 orbital planes nominally inclined at 56°.

2.1.3 Integrated System Architectures

In summary, the inertial navigation solution has the advantage of being continuously available, it does not rely on external reference frames, and it is provided at high output rates. Furthermore, inertial navigation systems provide a valid estimation of attitude, position and velocity, which accuracy depends on sensors' grade and cost; nevertheless, this solution presents a continuous drift due to error integration.

On the other side, GNSS solution is very accurate over the long-term but it is provided at a lower rate than inertial one; furthermore, it can be subject to jamming or interference. These issues, in addition to the fact that attitude cannot be estimated by standard GNSS equipment, make this solution unreliable for continuous navigation purposes, although the necessary equipment is very cheap. By integrating the two sensors' measurement it is possible to combine both technologies' benefits, obtaining a continuous and accurate navigation solution. As explained in Chapter 1, it is possible to exploit almost uninterrupted access to GNSS to lower the grade of the inertial sensors and obtain a low-cost but still accurate navigation system. Advantages and disadvantages of the three systems are resumed in the following Table 2.2.

Table 2.2: Comparison of navigation systems

	Advantages	**Disadvantages**
INS	High data rate Translational and rotational data Does not depend on external references	Growing error Knowledge of gravity needed Expensive if high accuracy is required
GNSS	Error is bounded Cheap	Low data rate Susceptible to external interference Outage periods Cannot estimate attitude
IMU/GNSS	High data rate Cheaper and more accurate than INS only navigation	Knowledge of gravity needed Algorithm complexity Filter tuning campaign needed

The basic architecture of an integrated IMU/GNSS systems obtained by coupling different sensors is represented in Figure 2.2.

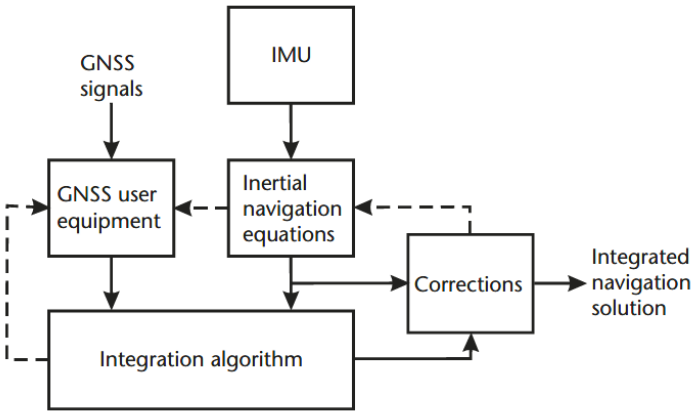


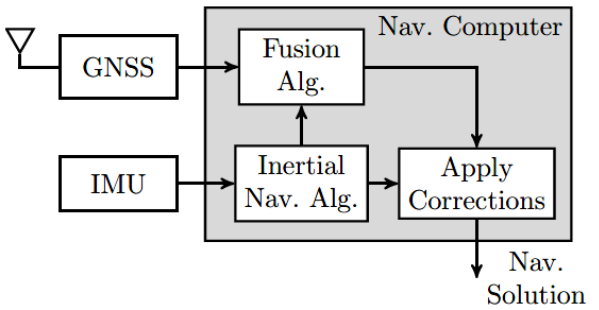
Figure 2.2: Basic integrated navigation system architecture [2]

IMU is used as dead-reckoning system while GNSS measurements represents position and velocity correction system. Dead reckoning solution is always available while estimation algorithm, represented usually by the Kalman filter described in next Section, uses position fixing measurements to update and correct the propagated state vector and obtain the integrated navigation solution.

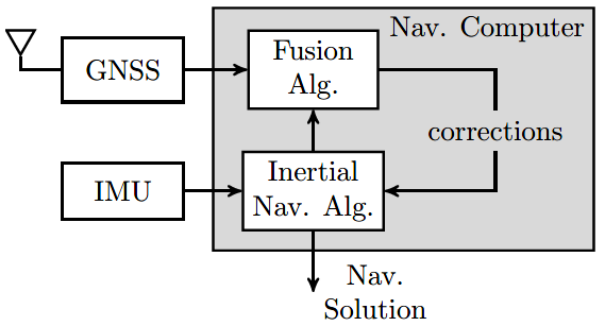
Two important approaches can be identified when choosing the architecture of a hybrid IMU/GNSS system, and they are practically independent:

- Corrections loop
- Coupling depth

The first one refers to how the corrections calculated by integrating algorithm are applied to the IMU solution. In this classification, there are two types of possible architectures:



(a) Open-loop



(b) Closed-loop

Figure 2.3: Open-loop and closed-loop architectures in hybrid IMU/GNSS system [7]

- **Open-loop**: in this configuration, the hybridization algorithm estimates correction to apply to inertial navigation solution (i.e. position, velocity and attitude), but corrections are not fed up to INS. This allows to have a raw inertial solution for monitoring but this solution continuously drifts and so the corrections may grow indefinitely.

- **Closed-loop**: in this architecture, the corrections applied to inertial navigation solution are fed to INS and are used to periodically reset the inertial navigation solution; in this way, it is possible to reduce numerical and linearizations errors, as the filter always stays close to the origin. The main drawback is that this configuration doesn't allow to have an independent inertial solution for integrity monitoring purposes.

The choice between these two types of architecture depends mainly from IMU grade and algorithm quality: if low-grade sensors are used, the only possible configuration is closed-loop, to avoid great corrections and because redundant independent solution may not be helpful. On the contrary, open-loop architecture is preferred if IMU's grade is higher and algorithm quality is lower, to perform integrity monitoring of the solution.
Figure 2.3 shows a schematic view of open-loop and closed-loop architectures.

Regarding coupling depth, the four main strategies are presented below:

- **Uncoupled systems**: this architecture allows to integrate benefits of INS and GNSS in the simplest possible way and still, providing system redundancy and independent operations of the two systems. It consists in resetting the INS using position and velocity estimates from GNSS. In this way, the inertial systems error are kept bounded, but in the absence of GNSS signal, the solution drifts as always. For this reason, although it provides redundancy, simplicity and minimum impact on software and hardware, this architecture is not common, also because attitude is not corrected, thus it drifts.

- **Loosely coupled systems**: this configuration is mainly suited for retro-fit application because it can be used with any INS and GNSS equipment and consists in GNSS autonomously operating and updating inertial system. GNSS raw measurements are used to calculate GNSS solution (PVT) in a first Kalman filter step, then they are used as measurement input to update INS/GNSS filter and correct inertial solution. Two main strengths of this system are simplicity and redundancy, because a parallel GNSS navigation solution is also available for integrity monitoring (if open-loop configuration is used, there is also an independent INS solution) and that a navigation solution is always available if GNSS is lost. Furthermore, the inertial solution can be used by GNSS receiver as tracking aid, to improve receiver performance in a noisy environment. The main weakness is that the system includes cascaded Kalman filters, lacking of the hypothesis that Kalman filter measurement errors should be uncorrelated. Finally, at least four satellite signals are needed to obtain a valid PVT solution, although it is possible to have a degraded solution for a limited time by using only three satellites signals. Figure 2.4 shows a schematic view of Loosely coupled architecture.

- **Tightly coupled systems**: in this architecture, the GNSS Kalman filter is now included in the IMU/GNSS Kalman filter, which takes as inputs raw measurements (pseudo-ranges and pseudo-range rates) from GNSS. The integration filter then calculates the corrections to apply to the INS solution. Either corrected or raw INS

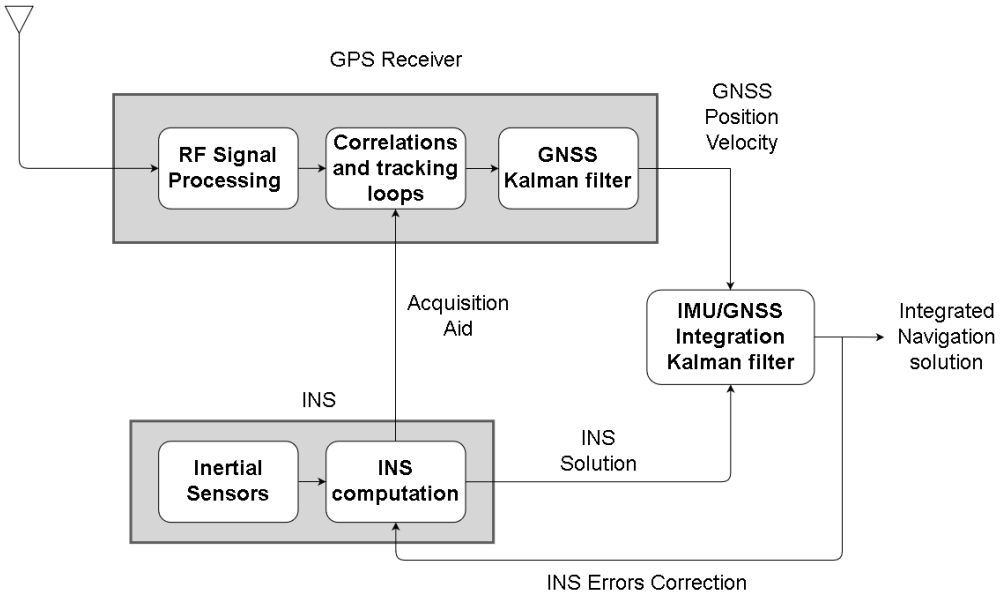


Figure 2.4: Loosely coupled architecture schematic view from [8]

solution can be used as acquisition aid for GNSS. There are several benefits given by this configuration: first, the problem of the correlation between measurement errors is eliminated because cascaded filters are not present; second, the system does not require a full GNSS solution, and consequently can work with less than 4 satellites signals (the greater the number of signals, the better the performances); this feature makes this method particularly suited for navigation in so-called *urban canyons*, where GNSS signal may be blocked by buildings. According to most of literature works (like [8], [2] and others), this approach makes the system more robust against GNSS outages and offers better performances in terms of accuracy, due to a deeper level of integration with respect to loosely coupled systems. Nevertheless, in other works, like [9], it is stated that loosely coupled architectures give a better answer to integrity monitoring challenge. Figure 2.5 shows a schematic view of tightly coupled architectures.

- **Deeply integrated or Ultra-tightly coupled systems**: this configuration (in Figure 2.6) is based on joining tracking and IMU/GNSS integration algorithm in a single filter. These methods are currently under development and theoretically should be more effective against interference, jamming, noisy environments and multipath errors but at the cost of highly increased complexity and computational load. It consists on directly input the accumulated correlator outputs in the Kalman filter, estimating INS and GNSS error to obtain corrected navigation solution, improving signal to noise ratio. Its implementation requires internal modification of GNSS and INS equipment, i.e. an access to a very deep level is required. Furthermore, GNSS and INS solutions are not independent anymore.

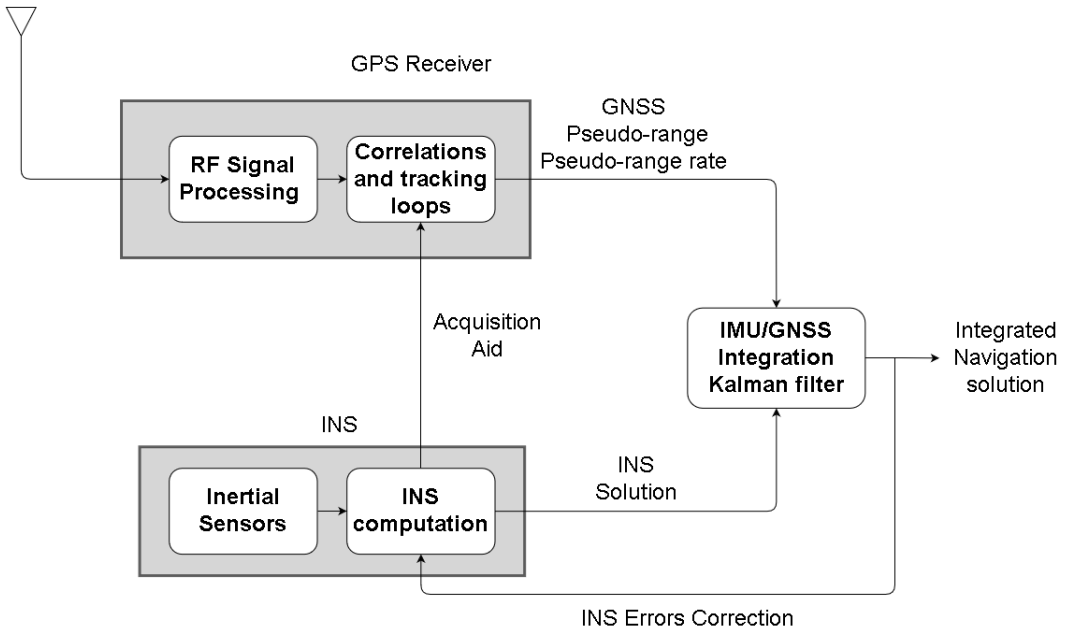


Figure 2.5: Tightly coupled architecture schematic view from [8]

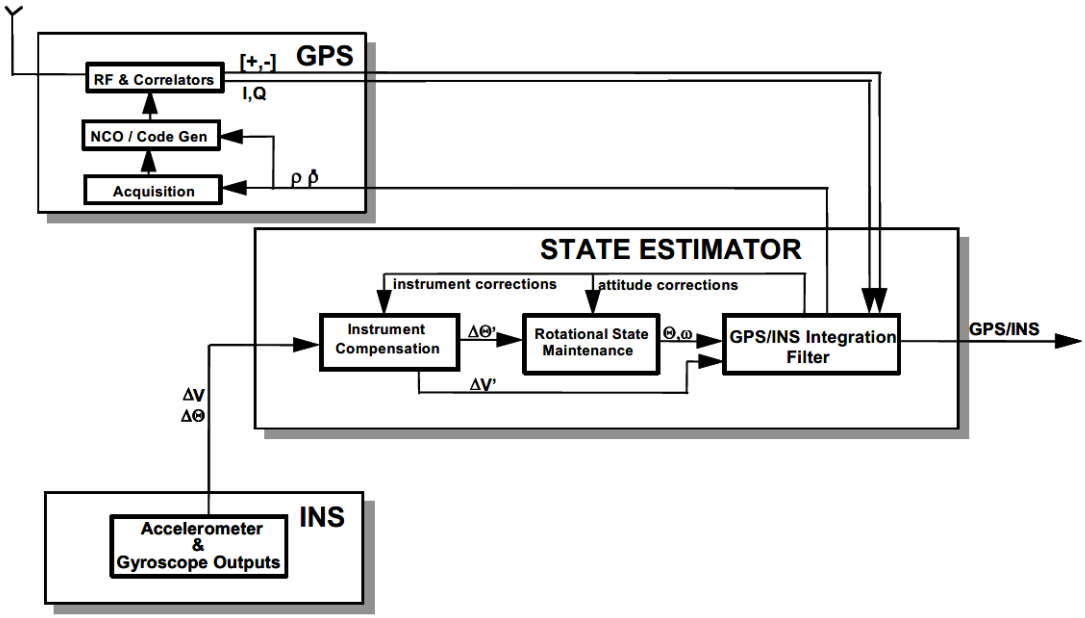


Figure 2.6: Deeply integrated systems schematic view [10]

In summary, uncoupled systems represent undoubtedly the less invasive manner to integrate GNSS and INS, but their use is not widespread because of the advantages that a tighter integration provides. In particular, loosely coupled architectures improve performances and provide a good grade of redundancy; furthermore, are especially suited in case of already existing equipment. On the other hand, tightly coupled architectures provide even better performances and eliminates the need of 4 satellites signals to perform a filter update, but may not be able to provide a fail-operational system. Finally, ultra-tightly coupled schemes could introduce an improvement in performances, especially in case of jam to signal ratio, at cost of system complexity and no parallel stand-alone solutions, and appear to be more suited for receiver designers because a very deep access to sensors functioning is needed.

There is not an absolute answer to system architecture choice. Indeed, the scenario and the application that requires a hybrid navigation system always drive the choice. Anyway, in case of GNSS loss or long outages, the quality of the inertial systems is the key towards good navigation performances.

2.2 Kalman Filter

The Kalman filter represented a breakthrough in modern estimation and control theory since its publication in 1960 by R. E. Kalman and its use in Apollo program (see [11]). It played and still plays, with all its developments and extensions, a major role in the aerospace industry when it comes to optimal estimation and finds application in a huge number of activities, like INS calibration, initial alignment, GNSS estimates of position and velocity, GNSS augmentation and, of course, multiple sensors integration.

The Kalman filter is an estimation algorithm that allows to obtain the best estimation of the navigation solutions using available measurements. In particular, it is the optimal recursive linear estimator: *optimal* because it minimizes the error covariance, *recursive* because it only uses the best previous step estimate and a new measurement to generate the best estimate of state at current step, reducing computational load for the processor, *linear* because it is based on linear process and measurement models and the state update is based on a linear weighted combination of state prediction and measurement. The weights of this linear combination are called Kalman Gains.

Figure 2.7 shows the main components of a Kalman filter:

- *State vector and covariance*: normally indicated by $\mathbf{x}$ and $\mathbf{P}$ respectively, they are the parameters estimated by the Kalman filter. State vector or states are the parameters that describe the system, in this case, position, velocity, attitude and all IMU's calibration parameters like biases, scale factors and so on. State covariance represents the error covariance matrix, i.e. a measure of the uncertainty of the estimate provided by the Kalman filter and of the correlation of the errors.
- *Measurement vector and covariance*: commonly indicated by $\mathbf{z}$ and $\mathbf{R}$ respectively.

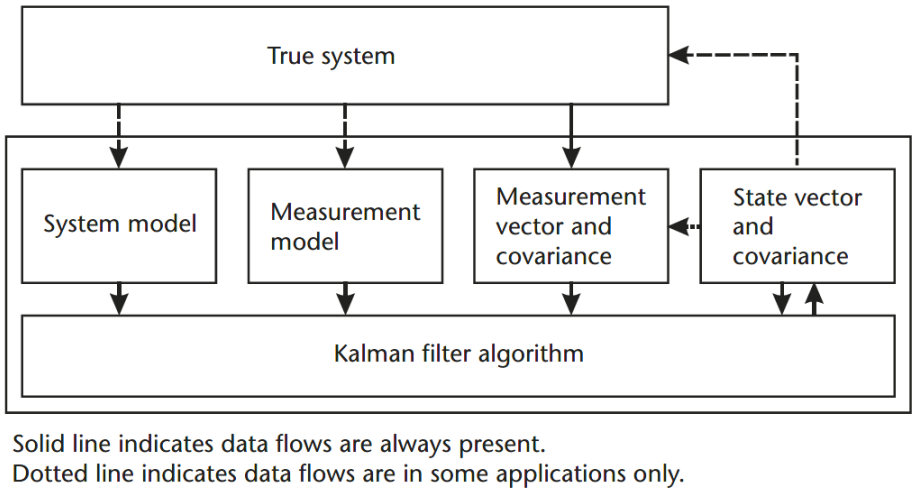


Figure 2.7: Kalman Filter main components [2]

Measurement vector is a set of measurement of available parameters that describe the system and are a function of the state vector. In this case, for example, GNSS measurement like position and velocity are used to update the Kalman filter. The measurement noise covariance matrix describes the characteristics of the noise that is present in the measurement.

- *System model*: commonly known as process model, is the model used to propagate in time the states and its covariance between two consecutive Kalman filter's updates. For this reason, it is usually represented by a model that takes into account previous step best estimation, an input or control vector and some noises to obtain current step updated estimate.
- *Measurement model*: this model relates measurements at the current time step with current step states. It is a deterministic model, like the previous one, that depends only on system properties and characteristics.
- *Kalman filter algorithm*: it is a series of steps to be performed to combine current step estimates and new measurements to obtain a new, corrected, estimate of the states at the current step of time.

General Kalman filter algorithm is essentially made up of two steps: prediction step and update step. For sake of simplicity and to better understand the functioning of this algorithm, the Linear Discrete-Time Kalman Filter is described in the following sections; then its formulation will be extended to other types of filters that find a more interesting application in this particular case of study. For a thorough and wider discussion on Kalman Filters, see [12], [13] and [14].

It is assumed that the system could be represented using a linear process, usually in the

form:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \nu_k \tag{2.4}$$

where $\mathbf{F}_k$ is the state transition matrix that relates previous step state to current step state, $\mathbf{B}_k$ is the input transition matrix, $\mathbf{u}_k$ is the control or input vector and finally ν_k is system noise. The measurement model is usually expressed in the following form:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \eta_k \tag{2.5}$$

where $\mathbf{z}_k$ is the measurement or observation vector, $\mathbf{H}_k$ is observation matrix that shows the dependence of measurement from states, and η_k is measurement noise. It is also assumed that the noises are gaussian, uncorrelated and with zero mean, and their covariance is indicated by $\mathbf{Q}$ for process noise and by $\mathbf{R}$ for measurement noise.

Starting from this assumptions, Kalman filter recursive algorithm is now described in detail as follows:

1. Calculate state transition matrix $\mathbf{F}_k$
2. Calculate process noise covariance matrix $\mathbf{Q}_k$
3. Propagate the state vector estimates from $\hat{\mathbf{x}}_{k-1}^{+}$ to $\hat{\mathbf{x}}_k^{-}$

$$\hat{\mathbf{x}}_k^{-} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1}^{+} + \mathbf{B}_k \mathbf{u}_k \tag{2.6}$$

4. Propagate the error covariance matrix from $\mathbf{P}_{k-1}^{+}$ to $\mathbf{P}_k^{-}$

$$\mathbf{P}_k^{-} = \mathbf{F}_k \mathbf{P}_{k-1}^{-} \mathbf{F}_k^T + \mathbf{Q}_k \tag{2.7}$$

5. Calculate the measurement or observation matrix $\mathbf{H}_k$
6. Calculate the measurement noise covariance matrix $\mathbf{R}_k$
7. Calculate the predicted measurement $\hat{\mathbf{z}}_k$

$$\hat{\mathbf{z}}_k = \mathbf{H}_k \hat{\mathbf{x}}_k^{-} \tag{2.8}$$

8. Calculate Kalman filter gains

$$\mathbf{K}_k = \mathbf{P}_k^{-} \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^{-} \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1} \tag{2.9}$$

9. Update states estimate and obtain $\hat{\mathbf{x}}_k^{+}$

$$\hat{\mathbf{x}}_k^{+} = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k \right) \tag{2.10}$$

10. Update covariance matrix estimate and obtain $\mathbf{P}_k^{+}$

$$\mathbf{P}_k^{+} = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_k^{-} \tag{2.11}$$

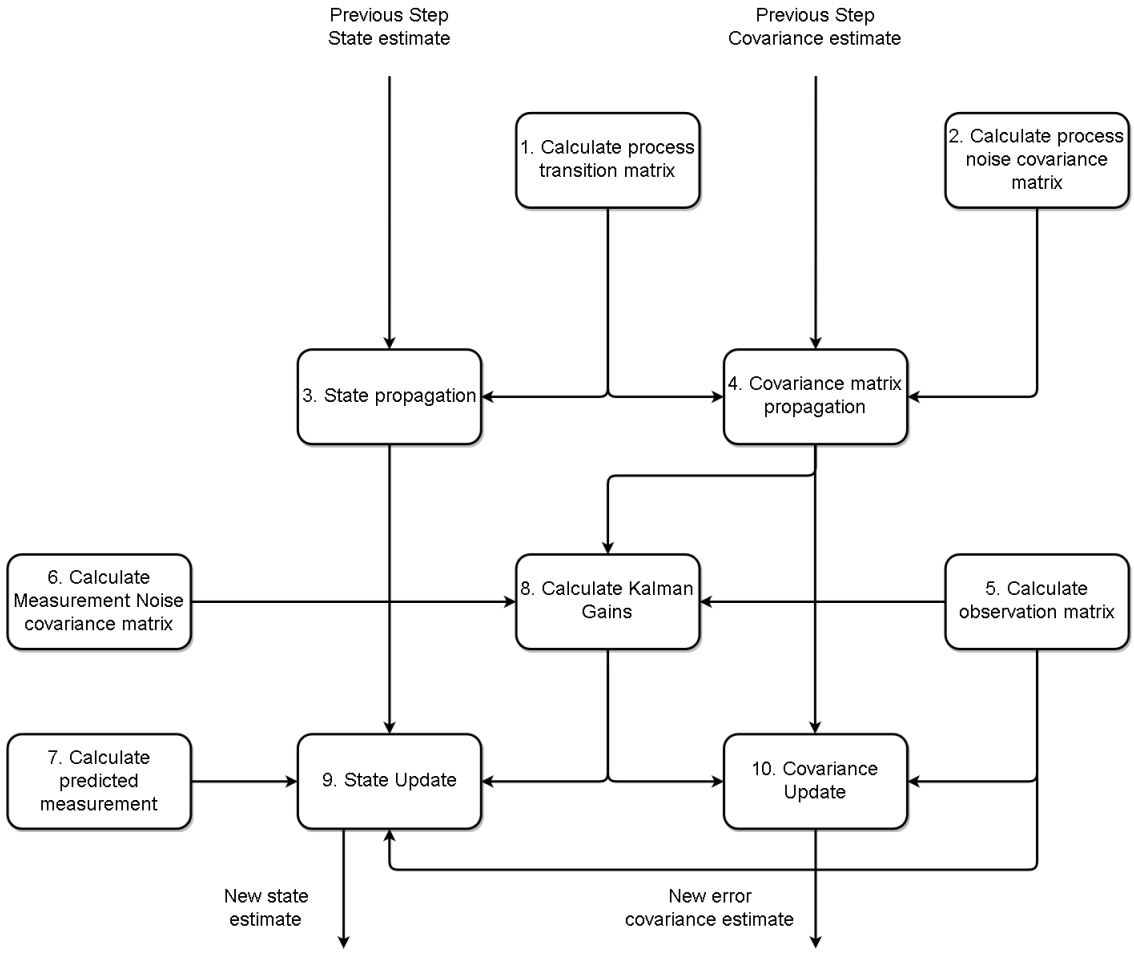


Figure 2.8: Schematic view of Kalman Filter Algorithm from [2]

In previous Equations, $\hat{\mathbf{x}}_k$ refers to estimated quantities at time step k, and superscripts "+" and "−" refer to corrected or updated state and predicted state respectively. Being a recursive algorithm, it is necessary to provide an initial estimate for state vector and error covariance. Steps from 1 to 7 belongs to the prediction phase, while steps from 7 to 10 belongs to update phase. A schematic view of the Kalman filter algorithm is presented in Figure 2.8.

In Equation (2.10), an important parameter appears. The difference between the actual measurement and the predicted measurement obtained using a measurement model is called *innovation*. The innovation contains the new information about the state vector and represents an important measure of how good is filter estimation. In fact, innovation should be zero-mean and white with covariance $\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k$ and if it's not as expected, then something is going wrong in the filter. It can be used for tuning, or validate

measurement before their inclusion in the update process, so it is an important attribute in relation to FDIR algorithms.

Although the Kalman filter algorithm appears to be simple, it has to be noticed that implementing a functioning Kalman filter in order to comply with some accuracy requirements is not usually trivial. This is due to the fact that its implementation is strongly influenced by specific features of the particular problem. The proper tuning of the filter is often very time-consuming but also the consistency of process and measurement models is a key to obtain good performances. So it requires a deep understanding of process and sensors and great ability in modelling them. Tuning of the Kalman filters is here intended as selecting the proper values for process and measurement noises covariance matrices and the initial estimate of state covariance matrix, that strongly influence the estimation: in fact, if these values are too small, the system is under-estimating the actual errors, while if these values are too large, then the systems over-estimates the actual errors with respect to true states. In particular, a key parameter is the ratio between state error covariance matrix and measurement noise covariance matrix because it is directly involved in Kalman gains computation, and affects the weight given to measurement with respect to prediction step.

Another issue to take into account is the dimension of the state vector. It strongly affects computational and processing load, in particular in the covariance propagation phase and measurement update phase. There are several solution than could lower the impact of this issue on processor load, like using a sparse matrix or taking advantage of matrix symmetry to reduce the number of operations.

2.2.1 Extended Kalman Filter

As said before, most of the systems cannot be modelled by linear processes because the states will have a non-linear behaviour in the majority of data-fusion problems. So, it seems necessary to develop another tool that allows to obtain a good estimation of the state vector for the studied system. This tool is a further development of the Kalman filter, i.e. the extended Kalman filter. This is the non-linear version of the previously described filter and has a similar recursive form with respect to linear Kalman filter. It is obtained by linearization of process and measurement models about the current best estimate of the state vector and its covariance. Of course, the linearization process introduces error in the estimation algorithm that makes this filter a sub-optimal estimator and, furthermore, very hard to tune. Nevertheless, it provides very good performances and it is the most used of IMU/GNSS integration problems, although it could diverge if the linearized system does not adequately reproduce the true system or if initial conditions are not accurate.

The algorithm of the Extended Kalman filter is very similar to the linear's one. It is now assumed that the process and measurement models are expressed by the following Equations respectively:

$$\mathbf{x}_k = \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, k\right) + \nu_k \tag{2.12}$$

$$\mathbf{z}_k = \mathbf{h}\left(\mathbf{x}_k\right) + \eta_k \tag{2.13}$$

In Equations (2.12) and (2.13) $\mathbf{f}$ and $\mathbf{h}$ represents non-linear state transition function and a non-linear measurement model respectively. The algorithm can still be divided in a prediction phase and an update phase, and the steps and equations to be resolved are basically the same as before, with the only difference that state transition matrix and observation matrix are now linearized about the current best estimate and can be expressed as the Jacobian matrix of partial derivative of $\mathbf{f}$ and $\mathbf{h}$ with respect to state vector:

$$\mathbf{F}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1}^+} \qquad \mathbf{H}_k = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_k^-} \tag{2.14}$$

The equations of the Extended Kalman filter are resumed in Table 2.3, where they comes divided in prediction step and update step.

Table 2.3: Summary of discrete-time EKF equations

State propagation	$\hat{\mathbf{x}}_k^- = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_k\right)$
Covariance propagation	$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1}^+ \mathbf{F}_k^T + \mathbf{Q}_k$
Kalman Gain calculation	$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}$
State update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}\left(\hat{\mathbf{x}}_k^-\right)\right)$
Covariance Update	$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k^-$

The same implementation issues discussed for linear Kalman filters apply even more to the EKF.
It is common, in navigation problems, to use an error-state Extended Kalman Filter (eEKF), that is estimating the error state, i.e. the difference between true state and estimated state, instead of the full state. This approach has risen to prominence because the error state dynamics could be linear, and this is a condition for optimal state estimation using Kalman filter. As said before, this filter is the same as EKF with the only difference of using error state $\delta\mathbf{x}$ instead of the full state. The eEKF propagates both full state and error state, then uses the measurement to update error state prediction, which is then used to update the full state. The only new operation in this type of filter is the reset of the filter by resetting error state to 0. Assuming the process model can be represented by:

$$\mathbf{x}_k = \mathbf{f}\left(\mathbf{x}_{k-1}\right) + \nu_k \tag{2.15}$$

where the control vector has not been considered for sake of simplicity, then the error state process is:

$$\delta\mathbf{x}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1} + \delta\mathbf{x}_{k-1}\right) - \mathbf{f}\left(\mathbf{x}_{k-1}\right) + \nu_k \tag{2.16}$$

and the equations of eEKF can be summed up in Table 2.4.

Two further improvements can be implemented to obtain better performances, dealing with numerical issues, and to reduce the processing load. The first one is State Scaling.

Table 2.4: Summary of discrete-time eEKF equations

State propagation	$\hat{\mathbf{x}}_k^- = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1}^+\right)$
Error State propagation	$\delta\mathbf{x}_k^- = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1}^+ + \delta\mathbf{x}_{k-1}^+\right) - \mathbf{f}\left(\mathbf{x}_{k-1}^+\right)$
Covariance propagation	$\mathbf{P}_k^- = \mathbf{F}_k\mathbf{P}_{k-1}^+\mathbf{F}_k^T + \mathbf{Q}_k$
Kalman Gain calculation	$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}$
Error state update	$\delta\hat{\mathbf{x}}_k^+ = \delta\hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{h}\left(\hat{\mathbf{x}}_k^- + \delta\mathbf{x}_k^-\right) + \mathbf{h}\left(\mathbf{x}_k^-\right)\right)$
State update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \delta\hat{\mathbf{x}}_k^+$; Filter reset
Covariance Update	$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^-$

This method is used to deal with numerical approximation due to limited memory used to represent a number. Every operation introduces a round-off error that could become problematic because of the great number of operations needed. Furthermore, it may result in error covariance conditioning problem. The proposed solution is to scale the state vector to reduce conditioning number of covariance matrix using a diagonal matrix called scaling matrix.
The second method is the sequential processing of measurement. It is a method that reduces the processing load and it is very useful in real-time applications. It consists in consider the measurement vector as a series of scalar measurement, so the update process could be reduced to scalar operations.

2.2.2 Schmidt-Kalman Filter

Kalman filter main assumption is that measurement noises are uncorrelated in time. This is not the case of all navigation systems, like in a loosely coupled system where, for example, input measurements of Kalman filter comes from GNSS dedicated Kalman filter. This could affect the performances of the filter. There are three way to account for time correlation of measurement noises:

- Down-weight Kalman gains or increase measurement noise covariance matrix. This method is simple but increase convergence time of the filter and also provides a larger uncertainty of the estimates.
- Include measurement noises as states to be estimated. This approach is very computationally expensive and may not work due to observability problems.
- Use a Schmidt-Kalman filter or *consider Kalman filter*.

The latter solution is the most common also because it can help when process noises are time-correlated and it also reduces processor workload. It consists in decomposing the states in estimated and considered parameters. Considered parameters are not estimated during the update step, so this solution greatly reduces the number of operations in the estimation process. In fact, only estimated parameters and their covariance will be affected

by the acquisition of new measurements, while considered states will only be propagated and their covariance will increase following noise characteristics.

2.3 Fault Detection, Identification and Recovery System

As already explained in Chapter 1, FDIR is a fundamental system when it comes to define and comply with safety, reliability and availability requirements, because it is a key component in system protection from any failure that may lead to mission loss.

FDIR functions definition and implementation can represent very challenging activities for several different reasons. First, failure analysis in a complex system like a launch vehicle could reach very sophisticated depth levels: it is really difficult to identify the cause of the fault and single faults generating in one component can quickly propagate to other system's components. Furthermore, failure analysis is tightly connected to systems' operational modes and FDIR should include a wide range of recovery actions. Also, FDIR testing and validation is not an immediate process, in fact, it requires a great amount of time, especially in test definition, for the difficulty in replication all the possible faults and their combination and also for problems of observability of the processes. That's why, until now, the most common approach in FDIR definition and design has been to consider FDIR as "*band-aids to already existing designs*" (from [6]), instead of considering it as a deeply integrated activity throughout the whole system study, definition and implementation.

The main contributions to FDIR design and implementation come from geodetic and navigation communities. Geodetic approach consists in applying multiple outlier detection theories developed for GNSS measurement integrity monitoring and multi-sensor integration, and are essentially based on statistics and residuals. Navigation community have focused on real-time application features and its approach is commonly based on single outlier theories assumptions.

To the present day, there is no common practice in FDIR development or definition, there are, instead, many concept and FDIR related methods in technical literature according to the specific discipline to which they are applied, like Fault Management, Fault Detection and Exclusion, Fault Protection and Hazard analysis. For this reason, when it comes to the FDIR study, it is important to give some definitions.

First of all, FDIR main functions are here described and depicted in Figure 2.9:

- **Fault Detection**: it refers to the systems knowing that there are some set of wrong measurements.
- **Identification** or **Isolation**: detection provided, it is essential to know where the fault happened and which measurements are wrong. In fact, detection does not provide identification because faults could propagate to other measurements. Isolation refers to avoiding the propagation of faulty measurements. The combination of these two functions is also commonly called *Fault Diagnosis*.

- **Recovery**: identification provided, the system should be able to choose the right action to perform given the detected fault, from the removal of wrong measurements to reconfiguration of the whole unit.

A *fault* is here considered as undesired deviation of a system parameter from its nominal value, and *failure* is considered as unexpected working or loss of a system function.

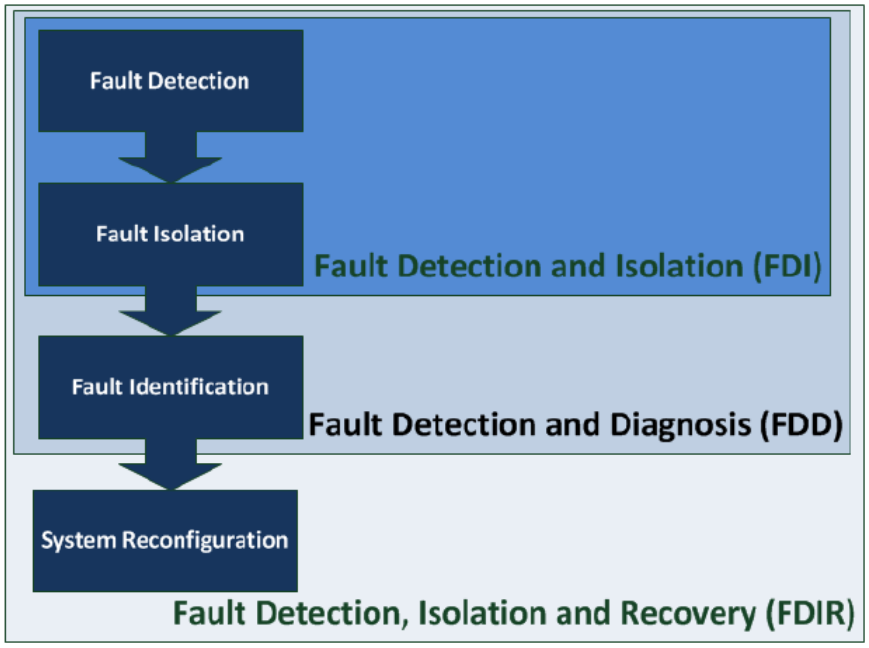


Figure 2.9: Schematic view of Fault Detection, Isolation and Recovery Functions [15]

For this particular case of study, given the functions of the hybrid navigation unit and the software-oriented insight of this analysis, it is more correct to refer to FDIR system as an Outliers Detection and Removal system (ODR), where *outlier* refers to a huge deviation from nominal behaviour of a system parameter, such as to be identified as generated by a different mechanism. This is especially true for GNSS measurements, while for IMU's measurements also some corrective actions are needed as part of the recovery.

ODR is generally based on redundancy. There are two different types of redundancy:

- Measurement redundancy: it refers to having more measurement than necessary. This is the case of using more satellite signals than necessary or using redundant IMUs, but the latter solution is usually unfeasible for launchers that aims to lower navigation unit cost, also because this means an increase in cost and mass. Redundancy is here intended as a way to make an eventual catastrophic fault as the result of multiple failures, avoiding single-point failures. Redundancy can be *cold*, if redundant unit is powered-off when not used (typically for systems that can deal with temporaries outages); clearly this type of redundancy is not the most suitable for a navigation application, due to its critical nature. The alternative is represented

by *hot* redundancy, where the redundant unit is always working and there is an immediate correction or switch to the second unit when a failure is detected.
For this application, measurement redundancy can be achieved by increasing sensors number (for example, using tan IMU with 4 accelerometeres and 4 gyroscopes in a skewed configuration), i.e. at sensor level; or it can be achieved by using a totally redundant navigation unit that works in parallel with the main one. In both case, there are several procedures that allow fault detection and isolation and provide a back-up solution to continue providing an accurate navigation solution.

- Knowledge redundancy: it refers to exploit *a priori* restrictions, for example using known models to represent the vehicle's motion.

It is also important to notice that algorithms cannot replace redundancy: existing redundancy could greatly improve performances through algorithms but they cannot be a remedy for its absence.

Finally, it is important to make a brief statement about errors type:

- *False Alarm* (FA), it refers to the rejection of correct measurement, and so this correct measurement is not used in the estimation process.
- *Misdetection* (MD), it refers to accepting an incorrect measurement, and so this incorrect measurements participates in the estimation process.

After a general view of the FDIR system, it is important to deal with some aspects for the particular case of a hybrid navigation unit for launchers:

- **Failures:** as explained in Section 2.1, Inertial Measurement Unit is the main sensor used in any navigation unit, due to its higher data rate that provides a bridge between GNSS measurements. Thus, an IMU-related failure is likely to have catastrophic consequences because GNSS navigation is not able to provide an accurate solution with a high-enough rate. On the other side, GNSS failures, being partial or total, can be considered acceptable even though they need to be flagged, in order to remove faulty measurements from computation. In fact, inertial system can provide a full navigation solution, but for a limited amount of time, as it tends to drift. If a failure of the navigation unit happens, the only way to perform recovery is to switch to a redundant unit. For this reason, it is important, for the FDIR system, to be able to detect and isolate the fault. Also, FDIR reports could help in posterior failure analysis and investigation, and in the development of corrective actions.
- **Environment:** launch vehicle environment is very demanding from many points of view. It is important to notice that vibrational and shock environment is very harsh. All these effects must be taken into account to comply with the high accuracy required for orbital injection. Missile phase is very representative of the conditions that launch vehicle and navigation unit have to bear. Failures that happen in navigation unit in this phase can result in catastrophic consequences if not recovered, because of the high dynamic of the system. A faulty measurement incorporated in the computational process may cause a quick divergence of the filter, the loss of the

system and of the mission and may lead to property damage and casualties. Furthermore, due to several burns and stage separation that occurs in this phase, it is very likely to have a saturation of the accelerometers, that results in frozen measurements being incorporated in the navigation solution. Also, GNSS vulnerabilities are an important issue to deal with: the very functioning of Satellite Navigation implies several drawbacks like jamming, external signal disturbance or ionospheric scintillation that may cause signal loss. Shocks suffered by launch vehicle and vehicle high dynamics can cause problems to satellite tracking and can cause outages, making GNSS solution unavailable.

- **FDIR vs Robust Navigation:** in navigation, a system could be defined *robust* if it is able to absorb wrong measurements, i.e. to continue working well in presence of outliers. This can be seen as a fault-tolerant philosophy, i.e. the system is specifically designed to fulfill its tasks even in presence of faults, possibly maintaining performances or at a degraded level. Thus, robust navigation refers to the ability of the integrated system to provide an accurate navigation solution in presence of outliers, i.e. the system absorbs the faults and the navigation solution may be considered accurate after a short transient. FDIR objective is to detect when a fault happens, isolate and identify the type of fault and its location, and undertake compensating provisions in order to remove the outliers and avoid the incorporation of faulty measurement that could lead to filter divergence.

Chapter 3

Approach to Hybrid Navigation FDIR

In this Chapter, a review of State of the Art ODR methodologies and techniques and FDIR architectures is presented in Section 3.1, introducing the difficulties related to outliers detection in the navigation field. Then, some analyses are carried out to identify the possible faults to deal with during the design of the FDIR system, like a Functional Analysis in Section, a Failure Mode Effect Analysis and a Fault Tree Analysis presented in Section 3.2, 3.3 and 3.4 respectively. To do so, a brief overview of the system is also provided.

3.1 State-of-Art Review

Despite a wide literature about FDIR and ODR methodologies being available, the design of an FDIR system for a hybrid navigation unit is a brand new investigation area. For this reason, there are not well established architectural recommendations. In fact, there are no recommended common practice for the general case neither: usually the design of a FDIR system is made *ad-hoc* for the specific application. Nevertheless, a brief description of space systems' FDIR architecture will be given in Section 3.1.1, followed by a general overview of outliers detection methods in Section 3.1.2. Finally, a specific insight on ODR methods for hybrid navigation systems will be given in Section 3.1.3.

3.1.1 FDIR Architectures

In space systems and space missions, the state-of-the-art for FDIR architecture is based on a hierarchical approach, for technical and programmatic reasons. FDIR systems are organized according to a set of hierarchical levels, characterized by a clear structure, well-defined interfaces, different tasks, different failure/fault to deal with and different recovery

actions, allowing a gradual intervention on the system (each failure/fault is recovered at the lowest possible level, to reduce the impact on the mission). The highest level is in charge of the vehicle's vital functions and integrity, while lower levels operate on subsystems or units. Lower levels trigger higher levels when they are not capable of recovery after the failure happens. So, FDIR higher levels control lower levels functions and execution. To make this possible, a great level of integration is needed: every subsystem of the space system must be a part of FDIR architecture, which makes the FDIR a complex system. A representation of a spacecraft FDIR hierarchical structure is shown in Figure 3.1 as an example.

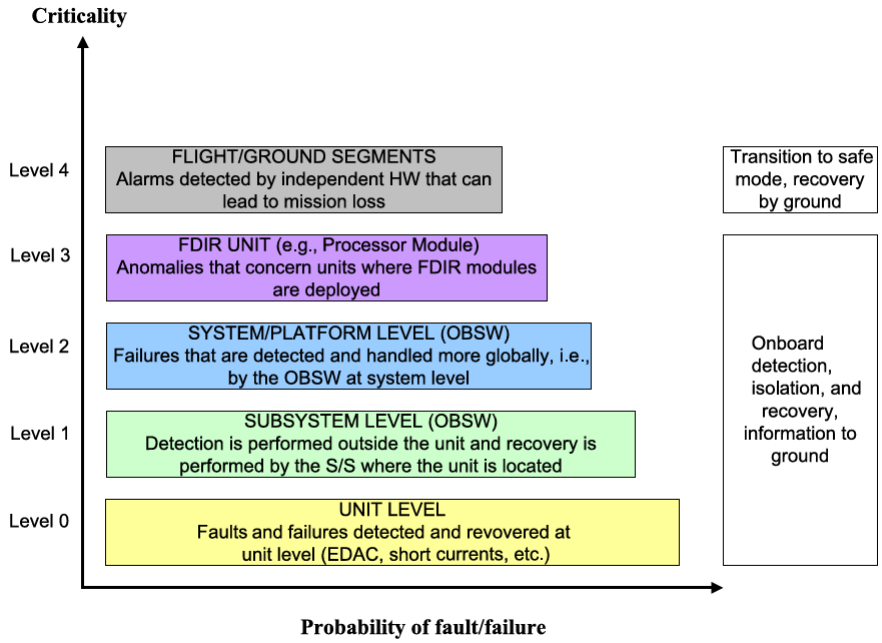


Figure 3.1: Example of spacecraft FDIR hierarchical structure

Generally, FDIR systems are organized in 4 or 5 levels, that have a different reaction time and are activated gradually depending on failure criticality (see [6] and [16]):

1. Level 0: it deals with failures and faults that can be recovered locally at unit level (e.g., EDAC, data bus failures, etc.) and that does not impact on system's performances. Detection is performed internally in the unit and the recovery is autonomous.

2. Level 1: it deals with failures and faults handled at subsystem level (e.g., unit failure, I/F failure), which typically require switching to a redundant unit and can degrade subsystem performances temporarily without any effect on mission goals (if a sensor fails, the system can use last available measurements to propagate). Detection is performed outside of the units and recovery is done at subsystem level.

3. Level 2: it deals with failures or faults handled by On-Board SW at system level

(e.g., failure of a system function, caused by undetected failures in lower levels). These failures can cause loss of system/function and performances degradation.

4. Level 3: it deals with failures related to On-Board Computer, which executes On Board SW and manage FDIR lower levels. It can be mixed with Level 2. Possible recovery actions are reconfiguration checks, OBC reset or a redundant processor module.

5. Level 4: the highest level deals with failures that can't be autonomously recovered by FDIR. For a spacecraft, this level handles all the critical failures that lead the vehicle into safe mode transition.

The higher the level, the higher the criticality but the lower the probability of failure.

The FDIR system for a hybrid navigation unit can be organized in a similar hierarchical way, including some FDIR functions at sensors level, at interfaces level and at general processing software level. The basic architecture is slightly simpler due to the reduced number of elements in the system and due to the particular insight given to the system. Furthermore, it can be divided in two separated levels relative to fault detection/isolation and recovery.

3.1.2 Outliers Detection methodologies

The aim of this section is to present an overview of outliers detection methods grouped in categories according to their approach to the problem (see [17], [18],[19] for more information).

There are several constraints concerning the design and the implementation of outliers detection algorithms for streaming data, the most important of which are, undoubtedly, real-time application, memory limitation and online processing. So, the choice between the several different methods presented below is driven by the geometry of the problem and available information. There is a huge literature about outliers detection over streaming data and lots of methods have been studied and implemented but, generally speaking, each of them can fit in one of these categories:

- ***Distance-based methods***: these methods detect outliers by examining the distance (usually Euclidean or Mahalanobis distance) to their nearest neighbours. In particular, if one data point has less than k points within a defined distance, then it classified as an outlier. There are several versions of distance-based methods, also called *k-NN* (Nearest Neighbour) methods[17], however almost all of them are affected by computational growth as they need to calculate distance to every data point, for this reason they are often combined with sliding windows models or their performances are enhanced by prototyping (comparing new data points with data prototypes, to reduce memory storage and computations). Distance-based methods have strong theoretical foundations and are efficient but attention must be paid because the neighbour set of each data point is constantly varying over time as the

window slides.

- ***Clustering-based methods***: these methods are based on grouping similar data in clusters. When a cluster has a size that is smaller than a defined threshold, then the whole cluster is considered as outlier, because they assume that inlier belongs to large and dense clusters. Clustering-based methods are unsupervised, i.e. they do not need previous knowledge of the data. *k-means* methods [17] are the most representative of this category: data points are compared to prototype cluster vectors that are represented by the mean of each cluster data and some boundaries. These statistics are stored over time as candidate outliers are identified and then further compared with following data. Other interesting members of this group are *Partition Around Medoids (PAM)* method [17], that uses an actual cluster point instead of the mean as center data of the set, and so is more robust to outliers, and *CLARANS* method [19]. These techniques are very suited for autonomous outlier detection and are very flexible and easy to implement to different problems, but may generate false alarms due to clustering algorithms.

- ***Density-based methods***: these methods are very similar to the ones described above because they consider one data point as outlier if it is in a region of very low density. Some of these methods are based on adaptive probability density functions, some others are based on modifications or further developments of nearest neighbour methods. They require low prior knowledge of data and can detect outliers close to inliers but can be very time-expensive.

- ***Parametric methods***: these methods allow to easily and rapidly incorporate new data for evaluation and are computationally efficient as the model grows with problem complexity and not with data size. They are statistical methods based on the pre-selection of a distribution model to fit the data and, for this reason, their application is limited. An interesting method is based on *Gaussian Mixture* distribution [18], where each data point is assigned with a score. If the score is higher than a threshold, the point is considered as outlier. These methods are easy to implement but limited to specific problems. Furthermore, the choice of the threshold is always difficult (fixed threshold are more robust but harder to calculate, adaptive threshold are less reliable).

- ***Non-parametric methods***: all of the techniques described above need some *a priori* knowledge or parameters to perform outlier detection. Non-parametric methods, on the other hand, does not require such expensive information or assumptions. They do not assume a statistical distribution of the data, but try to learn the distribution from the data, for this reason they are usually more expensive.

- ***Kernel-based or Semi-parametric methods***: these methods aim to merge the strength of parametric and non-parametric methods: in particular, the flexibility of the latter and the speed and efficiency of the former. Representative of this approach are Kernel Density Estimator methods([17] and [18]) that estimate probability density function of random variables.

- ***Sliding-Windows based methods***: in these methods, only a small portion of data contained inside a "window" of a certain size (here intended as the number of data points that the window contains) is stored in memory and considered to perform outlier detection. These methods are often combined with some of the previously described ones, and include both non-overlapping and overlapping sliding windows.

Distance-based, Clustering-based and Density-based methods can be included in the bigger category of *Proximity-based methods*.

The categories described above belong to the huge group of statistical models. They are designed for quantitative data and were the first ones to be used in outlier detection problems. Nevertheless, they are affected by some important problems when problem dimensionality increases, causing increase in computational time and distorting data distribution. This is called *"the Curse of Dimensionality"*[17]. This problem could be mitigated using some brand new techniques belonging to *Neural Networks* field.

Neural Networks are generally non-parametric and designed for the specific model. After proper training, they are able to recognize unforeseen patterns and learn complex boundaries. However training and tuning are very difficult and long to perform.

Other mitigating provisions can be machine learning algorithms and hybrid models that exploit statistical models features using neural network or machine learning approach; however, these methods can be difficult to implement in a launch vehicle navigation unit, where computational and memory resources are limited. Also, these methods are not considered robust enough for critical applications like launch vehicles navigation.

3.1.3 Possible FDIR approaches

As explained in previous Sections, Fault Detection, Isolation and Recovery or, in this case, Outlier Detection and Removal power is not limited by algorithms but by redundancy and problem geometry. With sufficient redundancy, outliers could be detected by a standard Least Square Estimator, by looking at normalized residuals. Unfortunately, because of problem geometry and of budget issues, redundancy is often limited. This is the case of small launchers, where the reduced cost is the design driver for every component. Nevertheless, there are several procedures that can be applied. The possible FDIR approaches can be classified according to the following categories:

- **Instantaneous (single-epoch) ODR** vs **Interval (multiple-epoch) ODR**.

  Instantaneous or single-epoch ODR refers to the search for outliers in the current navigation time step, assuming that the trajectory does not have outliers until the current time step. It is a simple and reliable method but does not account for correlation between current time measurements and previous measurements.
  Interval or multiple-epoch ODR refers to the use of a sliding window that creates an interval in which outlier detection is implemented. This procedure allows to re-evaluate false alarms while they are still inside the window and, furthermore,

increases redundancy but at the cost of increasing computational load (increase of window length). One of the drawbacks of this method is the correct sizing of the window length.

- **Single ODR** vs **Multiple ODR**.

  Single ODR is one of the simplest methods of outlier detection, and is the most widespread. It is based on the generation of residuals using a least square estimator or of the innovation using a Kalman filter, that are used to calculate a statistic. The statistic is then compared against a threshold to detect outliers. This is done taking advantage of the properties of the residuals and of the innovation, that generally follow a specific distribution. For example, the basic way of detecting outliers in GNSS measurements and removing the faulty measurement is to take advantage of the fact that innovation must be zero-mean and white.
  Multiple ODR is a generalization of single ODR, that generates a statistic using a full set of measurements instead of a single measurement, leading to the removal of the entire set once an outlier is detected.

- **Regression diagnostic-based ODR** vs **Robust estimators-based ODR**

  Regression diagnostic-based ODR is essentially based on the use of statistics test for the detection of groups of possible outliers. It is opposed to Robust estimators-based ODR that refers to the use of robust estimators, i.e. estimators that are minimally affected by outliers, to detect and remove outliers. There are robust estimators of location (central tendency of a set of measurements) and of scale (dispersion of the measurement set). They usually guarantee a good behaviour in presence of a small subset of outliers, i.e. they provide an accurate solution comparable to classic estimators without outliers, at the cost of a lower efficiency when outliers are not present. The robustness of the estimator is given by its breakdown point that is the proportion of incorrect measurements that the estimator can handle before giving an incorrect result. Examples of robust estimator that may be suitable for outlier detection are M-estimators (an alternative of least squares estimators that is based on minimizing another function of residuals, different from squared residuals, and is suitable in case of outliers) and W-estimators (an alternative form of M-estimators that uses weights to represent the importance of a sample in a dataset).

- **Only update-step measurement included** vs **All measurements included**

  Usually in outlier detection and removal procedures, only measurements that participates in the update step are used for outlier detection. This is due to the fact that access to all measurements residuals is not often guaranteed, especially in loosely coupled approach. All measurement included ODR refers to outliers detection in every measuremen and is a technique that has not been used so far.

3.2 System Overview and Functional Analysis

In this section, an overview of the hybrid navigation unit is provided together with its functional breakdown. The system developed by SENER AEROESPACIAL is a sensing/processing unit that aims to provide an accurate and reliable navigation solution throughout all mission phases, to be integrated in launch vehicles' Guidance Navigation and Control system. The core of the system is the sensors integration software, also called Navigation software, whose main function is to process and integrate data from an Inertial Measurement Unit (IMU) and GNSS equipment. It has a closed-loop, loosely coupled architecture and uses an Iterative Schmidt-Kalman error-state filter approach, with the implementation of state scaling and sequential processing of measurements. A functional scheme of the hybrid navigation unit is presented in Figure 3.2.

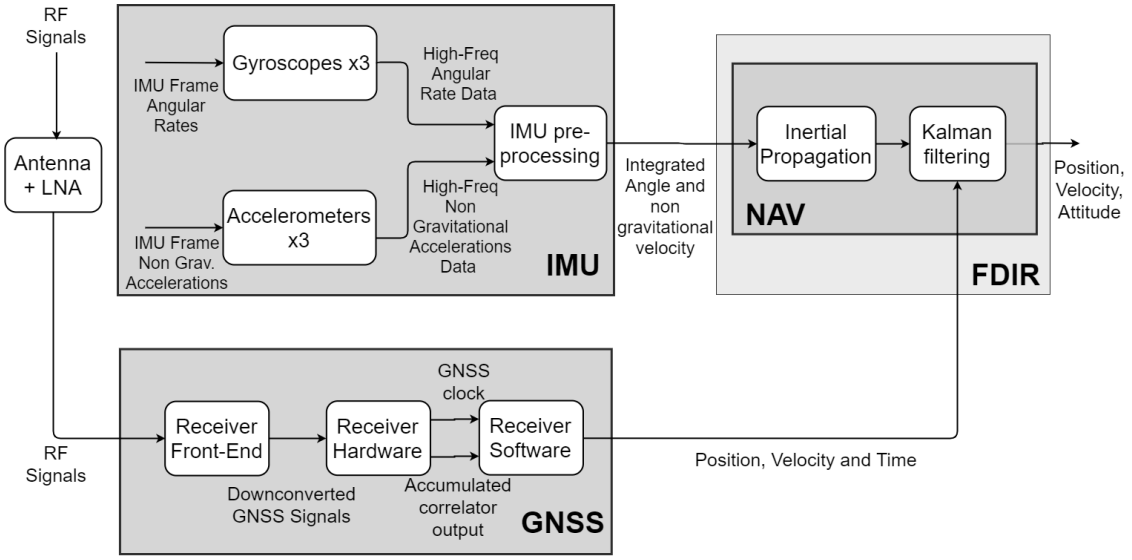


Figure 3.2: Hybrid Navigation Unit functional scheme

The IMU is responsible for acquisition and pre-processing, through temperature compensation and coning/sculling algorithm, of high frequencies inertial measurements, resulting in integrated angle and non-gravitational velocity that enter Navigation block. It also has its own time management unit and sensors management functions. GNSS equipment is responsible for signal acquisition and processing to obtain PVT solution, necessary in Kalman filter update step. Navigation block includes the necessary algorithm to provide a reliable navigation solution, including inertial state and covariance propagation and the implementation of the previously mentioned Kalman Filter. FDIR's main function is detection and identification of possible outliers in sensors' measurements and their removal from the computation of the navigation solution. Besides, power conditioning and distribution function is provided by a power unit, time management and synchronization function and memory management function are executed by OBSW.

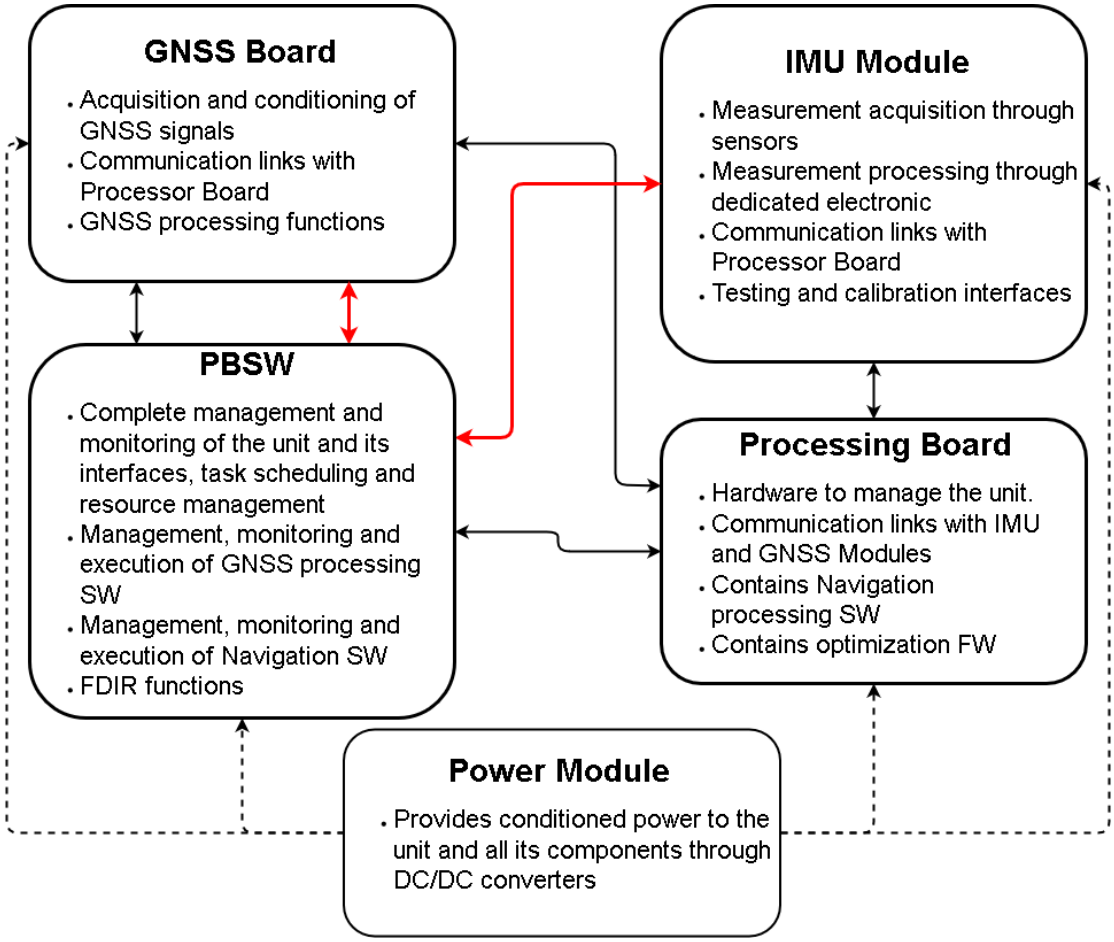


Figure 3.3: Hybrid Navigation Unit architecture and functions

A more detailed architectural representation is shown in Figure 3.3, where each function is assigned to its respective physical counterpart and interfaces. In particular, dotted lines represent power transfer, while black lines represent management data flow and red lines represent navigation data flow.

Processing Board is the core of Navigation Unit and, from a functional point of view, its functions can be divided into Software Functions and Firmware functions. In particular, software functions, including GNSS and core software and respective boards/modules, are:

- Initialization Built-In Tests (IBITs) execution.
- Mode management for GNSS module and Processing Board Software.
- External communication management.
- Implement FDIR functions.
- GNSS processing functions.

- Navigation algorithms.
- Memory management.
- Navigation unit time management.
- Sensors data acquisition and time-stamping.
- Discrete signals management.
- Synchronization and task management.
- Housekeeping data management.
- IMU pre-processing.

Regarding Firmware functions, they are listed below:

- IMU Acquisition and timestamping.
- Additional communication lines management.
- Acquisition and reset of navigation unit time counter.
- General management functions.
- TC managements, SW-FW interface management.

The software functions are contained in the Processing Board that also contains all the physical communication links with GNSS and IMU modules. GNSS board contains GNSS receiver elements for measurement acquisition and the GNSS part of the software that is controlled by the main software. IMU module contains all the inertial sensors to perform acquisition of measurements and its internal pre-processing with its dedicated electronics and the necessary interfaces in order to allow communication with processing board software. All the elements are powered through the power module that provides conditioned power to all the elements. The IMU has its own dedicated power module.

3.2.1 FDIR functions

In general terms, the FDIR system monitors the nominal operation of the navigation unit through the sensors' data and unit status check. In case of an undesired event, where a monitored parameter is detected to be out of limit, a report is generated and a recovery action is executed if available. These functions are implemented via periodic checking of navigation unit parameters, in particular:

- Analog sensors acquired (voltage and temperature);
- IMU measurements reading
- GNSS data acquisition
- Navigation algorithm monitoring

- Timing/synchronization functions
- Internal memory checks (memory scrubbing)

Detected errors are classified as Low Severity Errors (LSE) or High Severity Errors (HSE), the latter being catastrophic errors and the former being minor errors. When an LSE is detected, the FDIR is required to execute recovery action, generate an event report and continue with normal operation in the same mode. When an HSE is detected the FDIR is required to generate a death report and enter Failure Mode.

3.3 FMEA

FMEA definitions and procedures are referred to European Cooperation for Space Standardization ECSS-Q-ST-30-02C standard. Failure Mode Effect and Analysis is an inductive bottom-up method to identify potential failures in products and processes and evaluate the effect of failure causes on subsystem or the whole system, allowing the definitions of countermeasures and corrective actions. It represents a fundamental tool in the design process and it is necessary to implement it from the very beginning of the design phase. FMEA is essentially a reliability analysis, but it also gives information about safety, maintainability and FDIR system.

FMEA classifies each potential failure according to its severity, i.e. according to the effect that each failure has on subsystems and the global system. It is used to identify systems weak spots and critical areas (critical items) and define corrective actions to limit, reduce or eliminate potential risk for the system. A further development of FMEA is Failure Mode Effects and Criticality Analysis, FMECA, which classifies the failures according to their Criticality, that is the combination of a failure severity and the probability of its occurrence.

FMEA is used to define:

- product architecture and design justification;
- compensating provisions (redundancy, FDIR etc.);
- test procedures;
- maintenance operations;
- operational procedures.

FMEA steps are here described:

1. Description of the product/process under analysis, in particular of its functions, interfaces and interdependencies of its constitutive elements, but also its operational modes in every mission phase.
2. Identify all possible failure modes for each item and their effect on the product.

3. Assume that the considered failure mode is the only failure present in the product analyzed. This makes this kind of analysis not suitable for multiple failure analysis.
4. Evaluate potential failures and assign a severity category.
5. Identify failure detection mode
6. Identify corrective actions (existing or to be implemented) for each failure mode
7. Documentation and summary of the analysis
8. Identification of all the critical items and of corrective design actions to be implemented.

The definition of *critical item* will be given after the description of severity levels. Severity categories must be assigned without considering the existence of any compensating provisions, to give a general idea of the effect of the failure over the system in the worst possible condition. Severity categories are represented in Table 3.1.

Table 3.1: Severity categories of FMEA analysis [20]

Severity Category and Level	**Description of failure effects**	
	Dependability effects	**Safety effects**
Catastrophic (1)	Failure Propagation	-Loss of life, life-threatening or permanently disabling injury or occupational illness. -Loss of an interfacing manned flight system. -Severe detrimental environmental effects. -Loss of launch site facilities. -Loss of system.
Critical (2)	Loss of mission	-Temporarily disabling not life-threatening injury, or temporary occupational illness. -Major detrimental environmental effects. -Major damage to public/private properties. -Major damage to interfacing flight systems -Major damage to ground facilities.
Major (3)	Major mission degradation	
Minor or negligible (4)	Minor mission degradation or any other effect	

The number describing the severity of the failure is followed by a letter describing the type of failure: it can be *R*, meaning that the failure occurs when all redundant systems are failing, *SH* meaning the failure represents a Safety Hazard, *SP* to indicate single-point failures, i.e. the failure can cause directly the consequences expressed by the respective severity level.

A *critical item* is defined as an item which has a failure mode that is classified as a single-point failure and Catastrophic, Critical or Major consequences or simply has consequences listed as Catastrophic.

FMEA can be performed at different system levels and with different levels of profundity. In any case, FMEAs of lower levels have to be integrated in the higher level FMEA and its failure mode should be inserted in higher-level FMEA as failure causes of the end effect detected.

For this case of study, a Failure Mode and Effect Analysis has been conducted at IMU and GNSS levels and then at navigation unit level, including all failure modes that relate to FDIR functions. It has to be noticed that in this case Loss of Mission refers to the fact that the system is not able to deliver a constant, reliable and accurate navigation solution throughout all mission phases. This could have particularly rough effects during Missile Phase, including failures classified as Catastrophic.

As it can be seen in FMEA worksheets included in the following pages, the results of this analysis can be resumed as follows:

- IMU is a critical component for this system. IMU-related failures that involve accelerometers and gyroscopes measurements, and also undetected outliers often lead to loss of mission or catastrophic consequences, especially if the failure happens in Missile Phase.
- GNSS failures may cause worse, but still working, navigation solution. In fact, even in case of long GNSS outages, the navigation unit is still able to provide a full navigation solution, although this solution, being only inertial, is bound to drift.
- At navigation unit level, attention must be paid on interfaces between FW and SW and during sensors' data acquisition and timestamping. Also, an important function is performed by memory and mode management.
- A key function and requirement of FDIR is to properly detect failure mode and undertake the correct compensating action with right timing, allowing the unit to continue working in the right way or, in the worst case event where no recovery action is possible, to detect and report the error.

Failure Mode and Effect Analysis (FMEA)

System: Navigation Unit			Subsystem: Navigation Unit PBSW & I/F								
Identification Number	Item/Block	Function	Mission phase / Operational Mode	Failure Mode	Failure Cause	Failure Effect	Severity classification	Detection method	Compensating provisions	Recommendations	Remarks
NU - 01	PBSW - IBIT	Initialization and Built-In tests	GROUND	Initialization not performed	Processor SW is not able to perform the required functionality	a. Local effects: NU SW is not able to start b. End effects: Launch postponed/aborted	4	Initialization NOK	FDIR		HSE
NU - 02				Initialization performed wrongly	At least one IBIT failed	a. Local effects: Initialization NOK b. End effects: Launch postponed/aborted	4	Initialization NOK	FDIR		HSE
NU - 03				Initialization and BIT performed with wrong timing	Processor SW took more time to initialize than the expected	a. Local effects: None b. End effects: Negligible	4	None	None		LSE
NU - 04	PBSW - MM	Mode Management	TRANSITION TO FLIGHT MODE	Expected mode transition (commanded or autonomous) not performed	Wrong SW configuration for the mode change	a. Local effects: NU SW commandability is lost b. End effects: Mission Equipment Loss	1	Unexpected mode transition	SW criticality classification 'A'		HSE
NU - 05			TRANSITION TO FLIGHT MODE	Mode transitions performed wrongly, e.g. the mode conditions or the mode stability are compromised	SW malfuncion due to a non expected situation	a. Local effects: NU malfunction b. End effects: Mission Equipment Loss	1	Unexpected NU behavour	SW criticality classification 'A'		HSE
NU - 06			FLIGHT MODE	Mode transitions performed with wrong timing	Processor SW took more time than the expected one to perform the right mode transitions	a. Local effects: Real-time navigation performance degraded in transitions b. End effects: Mission Data Degradation	3	Degraded performances in mode transitions	FDIR	Depending on the moment when the failure happens and on its duration, severity level may increase to 2	LSE
NU - 07	PBSW - BUS	Bus communication link	FLIGHT MODE	Bus Communication not possible	Processor communication services are not performing as expected	a. Local effects: Total NU loss b. End effects: Mission Equipment loss	2*	NU not responding	SW criticality classification 'A'		HSE
NU - 08				Bus Communication data is corrupted	Memory buffers are corrupted, or data is wrongly coded/decoded	a. Local effects: Total NU loss b. End effects: Mission Equipment loss	2*	NU wrongly responding	SW criticality classification 'A'		HSE
NU - 09				Bus Communication data with wrong timing	Processor SW took more time than the expected one to perform the communications	a. Local effects: Real-time navigation performance degraded b. End effects: Mission Data Degradation	3	Degraded performances	FDIR	Depending on the moment when the failure happens and on its duration, severity level may increase to 2	LSE
NU - 10	GNSS-PBSW I/F	Serial communication I/F between GNSS SW and PBSW	FLIGHT MODE	GNSS Communication not possible	Processor communication services are not performing as expected	a. Local effects: GNSS total loss b. End effects: Mission Data Degradation	3	No GNSS data	FDIR	Depending on the moment when the failure happens and on its duration, severity level may increase to 2	LSE / HSE
NU - 11				GNSS Communication data is corrupted	Memory buffers are corrupted, or data is wrongly coded/decoded	a. Local effects: GNSS data not valid b. End effects: Mission Data Degradation	3	GNSS data not valid for navigation data	FDIR	Depending on the moment when the failure happens and on its duration, severity level may increase to 2	LSE / HSE
NU - 12				GNSS Communication data with wrong timing	Processor SW took more time than the expected one to perform the communications	a. Local effects: GNSS data not valid b. End effects: Mission Data Degradation	3	GNSS data not valid for navigation data	FDIR	Depending on the moment when the failure happens and on its duration, severity level may increase to 2	LSE / HSE

NU - 13	NU - FDIR	Failure Detection, Isolation and Recovery	FLIGHT MODE	Failure mode is not detected or is not identified	Failure detection mechanism fails, or the failure mode is not identified as a FDIR entry	a. Local effects: NU malfunction b. End effects: Mission Equipment Loss	2*	Unexpected NU performances	SW criticality classification 'A'	HSE
NU - 14			FLIGHT MODE	Recovery action for the detected failure is not executed	The recovery procedure does not success, or the failure mode is persistent	a. Local effects: Unrecoverable NU malfunction b. End effects: Mission Equipment Loss	2*	Recovery action not performed	SW criticality classification 'A'	HSE
NU - 15			FLIGHT MODE	FDIR performed wrongly	Failure mode is detected, but wrongly identified or recovered	a. Local effects: Unrecoverable NU malfunction b. End effects: Mission Equipment Loss	2*	Recovery action not performed	SW criticality classification 'A'	HSE
NU - 16			FLIGHT MODE	FDIR performed with wrong timing	The FDIR procedure takes more time than the expected one to solve a detected failure	a. Local effects: Real-time navigation performance degraded b. End effects: Mission Data Degradation	3	Degraded performances	FDIR	LSE
NU - 17	PBSW - NAV	Navigation algorithms	FLIGHT MODE	Navigation Algorithm not executed	SW malfuncion due to a non expected situation	a. Local effects: Navigation Data not provided b. End effects: Mission Equipment Loss	2*	No navigation data	SW criticality classification 'A'	HSE
NU - 18			FLIGHT MODE	Navigation algorithms wrongly executed	SW malfuncion due to a non expected situation	a. Local effects: Wrong Navigation Data is provided b. End effects: Mission Equipment Loss	2*	Unexpected NU performances	SW criticality classification 'A'	HSE
NU - 19			FLIGHT MODE	Navigation algorithms executed with wrong timing	SW malfuncion due to a non expected situation	a. Local effects: Wrong Navigation Data is provided b. End effects: Mission Equipment Loss	2*	Unexpected NU performances	SW criticality classification 'A'	HSE
NU - 20	NU - MEMORY	Memory Management	GROUND	Memory IBITs not performed	Memory configuration error	a. Local effects: NU unrecoverable error b. End effects: Launch postponed/aborted	4	Initialization NOK	FDIR	HSE
NU - 21			FLIGHT MODE	Memory dump functionality not performed	Unrecoverable memory read error	a. Local effects: NU unrecoverable error b. End effects: Mission Equipment Loss	2*	NU TM	FDIR	HSE
NU - 22			FLIGHT MODE	Memory buffers for tasks data exchange not managed	Memory resources problems, memory read/write errors	a. Local effects: NU unrecoverable error b. End effects: Mission Equipment Loss	2*	NU TM	FDIR	HSE
NU - 23			FLIGHT MODE	Memory integrity functionality not performed	EDAC mechanism is not enabled, checksum verification is not working	a. Local effects: NU unrecoverable error b. End effects: Mission Equipment Loss	2*	NU TM	FDIR	HSE
NU - 24			GROUND	Memory IBITs performed, but the memory contents or initialization are not the expected ones	Corrupted memory	a. Local effects: NU unrecoverable error b. End effects: Launch postponed/aborted	4	Initialization NOK	FDIR	HSE
NU - 25			FLIGHT MODE	Wrong memory dump functionality	SW malfuncion due to a non expected situation	a. Local effects: Wrong NU TM delivered b. End effects: Mission TLM Degradation	3	Unexpected NU TM	None	LSE
NU - 26			FLIGHT MODE	Memory buffers for tasks data exchange are wrongly managed, incoherence of the data between tasks	SW malfuncion due to a non expected situation	a. Local effects: NU unrecoverable error b. End effects: Mission Equipment Loss	2*	NU TM	FDIR	LSE

NU - 27			FLIGHT MODE	The memory integrity checks are compromised	Memory integrity functionality performed wrongly	a. Local effects: NU unrecoverable error b. End effects: Mission Equipment Loss	2*	NU TM	FDIR			HSE
NU - 28			GROUND	Memory IBITs performed with wrong timing	Processor SW took more time to initialize and check than the expected	a. Local effects: None b. End effects: Negligible	4	None	None			LSE
NU - 29			FLIGHT MODE	The memory dump function is performed late	SW malfuncion due to a non expected situation	a. Local effects: Wrong NU TM delivered b. End effects: Mission TLM Degradation	3	NU TM not in time	None			LSE
NU - 30			FLIGHT MODE	Task data exchange are managed with wrong timing	SW malfuncion due to a non expected situation	a. Local effects: Wrong Navigation Data is provided b. End effects: Mission Equipment Loss	2*	Unexpected NU performances	SW criticality classification 'A'			HSE
NU - 31			FLIGHT MODE	The memory integrity checks are performed late	Memory integrity functionality performed wrongly	a. Local effects: Wrong Navigation Data is provided b. End effects: Mission Equipment Loss	2*	Unexpected NU performances	SW criticality classification 'A'			HSE
NU - 32		Time Management	FLIGHT MODE	NU Time not managed	SW malfunction affecting the NU Time expected functionality	a. Local effects: Timestamp is missing in the TLM b. End effects: Mission TLM Degradation	3	Frozen timestamp	None			LSE
NU - 33	NU - TIME MGMT			NU Time managed wrongly	SW malfunction affecting the NU Time expected functionality	a. Local effects: Timestamp is missing in the TLM b. End effects: Mission TLM Degradation	3	Erroneous timestamp	None			LSE
NU - 34				NU Time managed with wrong timing	SW malfunction affecting the NU Time expected functionality	a. Local effects: Timestamp is missing in the TLM b. End effects: Mission TLM Degradation	3	Erroneous timestamp	None			LSE
NU - 35		Discrete signal Management	FLIGHT MODE	Discrete signals not managed	SW errors preventing to receive that signals	a. Local effects:The NU user is not aware about certain special events/errors b. End effects: Mission TLM Degradation	3	None	None			LSE
NU - 36	NU - DS MGMT			Discrete signals management performed wrongly	Discrete signals malfunction	a. Local effects: The NU user is not aware about certain special events/errors b. End effects: Mission TLM Degradation	3	None	None			LSE
NU - 37				Discrete signals managed with wrong timing	SW malfuncion due to a non expected situation	a. Local effects: The NU user is not aware about certain special events/errors b. End effects: Mission TLM Degradation	3	None	None			LSE
NU - 38		IMU Communication Management	FLIGHT MODE	IMU communications not performed	PB FW Design failure PB IMU comm. Failure	a. Local effects: IMU data is not available b. End effects: Mission Equipment Loss	2*	No IMU data	FDIR			HSE
NU - 39	NU - COMM			Wrong IMU communications	PB FW Design failure PB IMU comm. Failure	a. Local effects: IMU data not valid if persistent b. End effects: Mission Equipment Loss	2*	Erroneous measurements	FDIR			HSE
NU - 40				IMU communications performed with wrong timing when it is needed	PB FW Design failure PB IMU comm. Failure	a. Local effects: IMU data is not available b. End effects: Mission Equipment Loss	2*	Erroneous measurements	FDIR			HSE

NU - 41	NU - SEN	Acquisition and time-stamping of sensors data	FLIGHT MODE	Sensor data not acquired	SW malfunction affecting the FW/processor interface	a. Local effects: IMU data is not available b. End effects: Mission Equipment Loss	2*	IMU error data	FDIR			HSE
NU - 42				Sensor data not timestamped	SW malfunction affecting the FW/processor interface	a. Local effects: Timestamp is missing in the TLM b. End effects: Mission TLM Degradation	3	Frozen timestamp	None			LSE
NU - 43				Sensor data wrongly acquired	SW malfunction affecting the FW/processor interface	a. Local effects: IMU data is not available b. End effects: Mission Equipment Loss	2*	IMU error data	FDIR			HSE
NU - 44				Sensor data wrongly timestamped	SW malfunction affecting the FW/processor interface	a. Local effects: Timestamp is missing in the TLM b. End effects: Mission TLM Degradation	3	Frozen timestamp	None			LSE
NU - 45				Sensor data acquired with wrong timing	SW malfunction affecting the FW/processor interface	a. Local effects: IMU data is not available b. End effects: Mission Equipment Loss	2*	IMU error data	FDIR			HSE
NU - 46				Sensor data timestamped with wrong timing	SW malfunction affecting the FW/processor interface	a. Local effects: Timestamp is missing in the TLM b. End effects: Mission TLM Degradation	3	Frozen timestamp	None			LSE
NU - 47	NU - SYNC	Synchronization and Tasks Management	FLIGHT MODE	Synchronization not performed	Use case where the CPU load is bigger than the obtained during the WCET analysis	a. Local effects: The navigation data are not valid for real-time navigation b. End effects: Mission Equipment Loss	2*	CPU load above threshold	FDIR			HSE
NU - 48				Synchronization performed wrongly	Use case where the CPU load is bigger than the obtained during the WCET analysis	a. Local effects: The navigation data are not valid for real-time navigation b. End effects: Mission Equipment Loss	2*	CPU load above threshold	FDIR			HSE
NU - 49				Synchronization performed with wrong timing	Use case where the CPU load is bigger than the obtained during the WCET analysis	a. Local effects: The navigation data are not valid for real-time navigation b. End effects: Mission Equipment Loss	2*	CPU load above threshold	FDIR			HSE
NU - 50	NU - HK MGMT	Housekeeping Data Management	FLIGHT MODE	Housekeeping data management not performed	SW malfuncion due to a non expected situation	a. Local effects: The navigation data are not valid for real-time navigation b. End effects: Mission Equipment Loss	2*	Unexpected NU performances	FDIR			HSE
NU - 51				Housekeeping data management performed wrongly	SW malfuncion due to a non expected situation	a. Local effects: The navigation data are not valid for real-time navigation b. End effects: Mission Equipment Loss	2*	Unexpected NU performances	FDIR			HSE
NU - 52				Housekeeping data managed with wrong timing	SW malfuncion due to a non expected situation	a. Local effects: The navigation data are not valid for real-time navigation b. End effects: Mission Equipment Loss	2*	Unexpected NU performances	FDIR			HSE
NU - 53	PBSW - FW I/F	Interface between Processing Board SW and FW	FLIGHT MODE	NU SW/FW communications not performed	PB FW Design failure PB SW/FW IF failure	a. Local effects: IMU data is not available b. End effects: Mission Equipment Loss	2*	Unexpected FW status/behavour	FDIR			HSE
NU - 54				NU SW/FW communications wrongly performed	PB FW Design failure PB SW/FW IF failure	a. Local effects: IMU data not valid if persistent b. End effects: Mission Equipment Loss	2*	Unexpected FW status/behavour	FDIR			HSE

NU - 55			NU SW/FW communications performed later than when they are needed	PB FW Design failure	a. Local effects: IMU data not valid if persistent b. End effects: Mission Equipment Loss	2*	Unexpected FW status/behavour	FDIR		HSE

*Severity level depends on mission phase: Catastrophic (1) if it happens in Missile Phase or at trasition into Flight Mode; Critical (2) in every other case

Failure Mode and Effect Analysis (FMEA)

System: Navigation Unit		Subsystem: IMU Module								
Identification Number	Item/Block	Function	Mission phase / Operational Mode	Failure Mode	Failure Cause	Failure Effect	Severity classification	Detection method	Compensating provisions	Recommendations	Remarks
IMU - 01	GYROSCOPES	GYROSCOPES - (POW) Power supply/voltage supervisor	ALL	Gyroscopes power reset circuit failure	/	a. Local effects: degradation of gyroscope performances (all axes) b. End effects: degradation of navigation performances	3	PBIT, CBIT	FDIR		LSE
IMU - 02	GYROSCOPES	GYROSCOPES - (ELEC) Programmable electronics to gather and process synchronized data	ALL	Failure in gate switching signal	/	a. Local effects: degradation of gyroscope performances (one axis) b. End effects: degradation of navigation performances	3	N/A	FDIR		LSE
IMU - 03	GYROSCOPES	GYROSCOPES - (ELEC) Programmable electronics to gather and process synchronized data	FLIGHT MODE	Failure in gyroscopes process and control electronics	/	a. Local effects: incorrectness of gyroscope data (all axes) b. End effects: Mission Equipment Loss	2*	PBIT, CBIT	FDIR		HSE
IMU - 04	GYROSCOPES	GYROSCOPES - (SAT) Control of gyroscope saturation	ALL	Gyroscopes output above the threshold	/	a. Local effects: degradation of gyroscope performances b. End effects: degradation of navigation performances	3	PBIT, CBIT	FDIR	Propagate with previous measurements	LSE
IMU - 05	ACCELEROMETER	ACCELEROMETER - (SAT) Control of accelerometer saturation	ALL	Accelerometer output above the threshold	/	a. Local effects: degradation of accelerometer performances b. End effects: degradation of navigation performances	3	PBIT, CBIT	FDIR	Propagate with previous measurements	LSE
IMU - 06	ACCELEROMETER	ACCELEROMETER - (ELEC) Programmable electronics to gather and process synchronized data	FLIGHT MODE	Failure in accelerometer process and control electronics	/	a. Local effects: incorrectness of accelerometer data (all axes) b. End effects: Mission Equipment Loss	2*	PBIT, CBIT	FDIR		HSE
IMU -07	IMU - FDIR	FDIR Management	FLIGHT MODE	Undetected errors in gyroscopes output data	/	a. Local effects: incorrectness of gyroscope data (all axes) b. End effects: Mission Equipment Loss	2*	N/A			HSE
IMU - 08	IMU - FDIR	FDIR Management	FLIGHT MODE	Undetected errors in accelerometers output data	/	a. Local effects: incorrectness of accelerometer data (all axes) b. End effects: Mission Equipment Loss	2*	N/A			HSE
IMU - 09	IMU - FDIR	FDIR Management	ALL	False Alarm	/	a. Local effects: degradation of output data b. End effects: Negligible	4	PBIT/CBIT	FDIR		LSE
IMU - 10	IMU - FDIR	FDIR Management	FLIGHT MODE	Undetected errors in board temperature data	/	a. Local effects: incorrect board temperature data b. End effects: incorrectness of output data, Mission Equipment Loss	2*	N/A			HSE
IMU - 11	IMU - FDIR	FDIR Management	START-UP	IMU IBIT failure	/	a. Local effects: Loss of IMU output data b. End effects: Launch aborted/postponed	4	FDIR			HSE
IMU - 12			FLIGHT MODE	Failure in Voltage regulator circuits	/	a. Local effects: Loss of output data b. End effects: Mission Equipment Loss	2*	S		Should be detected by other systems	HSE

	POW	Voltage generator		Failure in Voltage reference circuits	/	a. Local effects: incorrectness of gyroscopes and accelerometers output data b. End effects: Mission Equipment Loss	2*	PBIT/CBIT	FDIR		HSE
IMU - 13	MEMORY	Provides configuration parameters to the system	FLIGHT MODE	Failure in Flash Memory	/	a. Local effects: loss of configuration parameters, incorrectness of gyroscopes and accelerometers output data in case of restart during operative phase. b. End effects: Mission Equipment Loss	2*	PBIT			HSE
IMU - 14	BOARD TEMPERATURE	Allow to monitor electronic board temperature	FLIGHT MODE	Loss of temperature data	/	a. Local effecets: loss of temperature board data b. End effects: incorrectness of output data, Mission Equipment Loss	2*	PBIT, CBIT			HSE
IMU - 15			FLIGHT MODE	Failure in temperature sensor output data	/	a. Local effecets: incorrectness of temperature board data b. End effects: incorrectness of output data, Mission Equipment Loss	2*	PBIT, CBIT			HSE
IMU - 16	SYSTEM CLOCK GENERATOR	Generate an external, more accurate, system clock signal	FLIGHT MODE	Failure in Clock generator circuit	/	a. Local effects: Loss of SYS_CLK signal b. End effects: loss of telemetry data, Mission Equipment Loss	2*	N/A			HSE
IMU - 17	TELEMETRY TRANSCEIVER	Provides inertial data of the IMU to Navigation Unit	FLIGHT MODE	Impossibility to exchange data between IMU and Navigation Unit	/	a. Local effects: Loss of telemetry data b. End effects: Mission Equipment Loss	2*	S		Should be detected by other systems	HSE
IMU - 18	SYNC. TRANSCEIVER	Provide synchronization pulse train signal for clock drift evaluation	FLIGHT MODE	Impossibility to provide synchronization pulse train	/	a. Local effects: Loss of synchronization b. End effects: Mission Equipment loss	2*	S		Should be detected by other systems	HSE
IMU - 19											

* This failure severity level depends on mission phase: it is considered as Catastrophic (1) if it happens in Missile Phase, and Minor (4) if it happens in Initial alignment phase (Launch Aborted/Postponed). In every other case, its severity level is 2

Failure Mode and Effect Analysis (FMEA)

System: Navigation Unit			Subsystem: GNSS Module								
Identification Number	Item/Block	Function	Mission phase / Operational Mode	Failure Mode	Failure Cause	Failure Effect	Severity classification	Detection method	Compensating provisions	Recommendations	Remarks
GNSS - 01	GNSS Antenna	RF signal reception	FLIGHT MODE	RF signal not delivered, wrongly conditioned or delivered late	RF chain failure Corrupted RF chain configuration	a. Local effects: loss of RF signal, loss of GNSS b. End effects: degradation of navigation performances	3	NO GNSS data available	FDIR & IMU data		LSE
GNSS - 02			FLIGHT MODE	GNSS data not delivered, wrongly delivered or delivered late	TMTC serial line failure Memory EDAC mechanism failure	a. Local effects: GNSS signal degradation, degraded PVT accuracy b. End effects: degradation of navigation performances	3	Wrong GNSS data available	FDIR & IMU data		LSE
GNSS - 03	Memory Controller	Memory Management	ALL	Recoverable memory error	/	a. Local effects: Fixed error b. End effects: Negligible	4	Memory scrubbing detects a recoverable error	Memory scrubbing will be performed on all on-board memories equipped with EDAC	Send event	LSE
GNSS - 04			ALL	Double EDAC memory error	/	a. Local effects: GNSS FAIL mode, loss of GNSS b. End effects: Degradation of navigation performances	3	Memory scrubbing detects a recoverable error	Memory scrubbing will be performed on all on-board memories equipped with EDAC	Send event and transition to GNSS FAIL mode	LSE
GNSS - 05	RTOS	RTOS Management	ALL	Worst Case Execution Time violation / deathline reached	/	a. Local effects: GNSS FAIL mode, loss of GNSS b. End effects: Degradation of navigation performances	3	A task response time exceeds its deadline	Software watchdog monitors the tasks response time	Send event and transition to GNSS FAIL mode	LSE
GNSS - 06			ALL	Processor unexpected exception	/	a. Local effects: GNSS FAIL mode, loss of GNSS b. End effects: Degradation of navigation performances	3	The processor produces an unexpected exception that interrupts the ASW execution	N/A	Send event and transition to GNSS FAIL mode	LSE
GNSS - 07	Processor	Watchdog	ALL	Watchdog violation	/	a. Local effects: GNSS SW reboot b. End effects: degradation of navigation performances	3	The Hardware watchdog is not rearmed on time	N/A	Reset GNSS Board. Identify anomalous reboot cause, send event and transition to FAIL mode	LSE
GNSS - 08	GNSS SW - FDIR	FDIR Management	GNSS START-UP	IBIT Failure	/	a. Local effects: GNSS FAIL mode, loss of GNSS b. End effects: Launch postponed/aborted	4	Boot report shows a NOK in IBIT result	Check Boot report at start-up	Send Event and command transition to FAIL mode.	LSE
GNSS - 09			ALL	CPU high lload	/	a. Local effects: CPU above threshold but operational b. End effects: Negligible	4	CPU load is above allowed threshold	Monitor CPU load and check if it rises above threshold	Issue event	LSE
GNSS - 10	GNSS SW - CFGM	GNSS ASW Configuration Management (CFGM)	ALL	GNSS parameters with illegal values	/	a. Local effects: Default configuration applied b. End effects: Negligible	4	Illegal values in ASW configuration parameters	Check if ASW configuration values (received via telecomand) are not compliant with validity intervals/criteria.	Issue event and apply default configuration (depending on operation mode)	LSE
GNSS - 11	GNSS SW - MEAS	GNSS Measurements Generation (MEAS)	FLIGHT MODE	GNSS measurement spike	/	a. Local effects: Measurement marked as invalid b. End effects: Degradation of navigation performances	3	Measurement outlier detected	Check if difference between current measurement and set of previous measurements is higher than threshold (defined taking into account mission envelope).	Issue event and mark measurement unusable (measurement to be excluded from navigation solution.	LSE
GNSS - 12			FLIGHT MODE	GNSS measurement frozen	/	a. Local effects: Measurement marked as invalid b. End effects: Degradation of navigation performances	3	Repeated measurement detected (same exact value during consecutive measurement epochs)	Check if difference between current measurement and previous measurement is zero.	Issue event and mark measurement unusable (measurement to be excluded from navigation solution.	LSE

GNSS - 13			FLIGHT MODE	GNSS invalid measurement	/	a. Local effects: Measurement marked as invalid b. End effects: Degradation of navigation performances	3	Measurement outside mission envelope or without physical meaning (e.g. negative PR)	Check if measurement is outside mission envelope or without physical meaning (e.g. negative PR).	Issue event and mark measurement unusable (measurement to be excluded from navigation solution.	LSE
GNSS - 14	GNSS SW - GSV + MDEC	GNSS Satellites Management (GSV) + Navigation Message Decoding (MDEC)	FLIGHT MODE	GPS/Galileo health status	/	a. Local effects: Measurement marked as invalid b. End effects: Degradation of navigation performances	3	Unhealthy satellite being tracked	Check if health flag of satellite being tracked is set to "unhealthy".	Issue event and mark satellite and corresponding measurements unusable (measurements to be excluded from navigation solution). Release tracking channel to be used for another satellite.	LSE
GNSS - 15	GNSS SW - SSYN	Symbol Synchronization (SSYN)	FLIGHT MODE	Loss of bit/secondary code synchronization	/	a. Local effects: Measurement marked as invalid b. End effects: Degradation of navigation performances	3	Bit/secondary code synchronization check fails	Before symbol or secondary code synchronization is achieved: check if there is more than one candidate bit transition or secondary code start location. After symbol or secondary code synchronization was achieved: check if there is more than one candidate bit transition or secondary code start location or if the current candidate is different from the previous one.	Issue event and exclude measurements from corresponding satellite from navigation solution.	LSE
GNSS - 16	GNSS SW - MDEC	Navigation Message Decoding (MDEC)	FLIGHT MODE	Data bit decoding errors	/	a. Local effects: Navigation Message not decoded b. End effects: Negligible	4	Error checking in data decoding task fails	Check outputs of bit decoding algorithm and see if bit errors have been detected.	Issue event and continue attempting message decoding.	LSE
GNSS - 17			FLIGHT MODE	Frame start pattern not found	/	a. Local effects: Frame start pattern is not found b. End effects: Negligible	4	Search for frame synchronization pattern fails	Check outputs of frame synchronization task.	Issue event and continue attempting frame synch.	LSE
GNSS - 18	GNSS SW - TRK	Tracking (TRK)	FLIGHT MODE	Total loss of GNSS signal/data <= 5s (TBD)	/	a. Local effects: Temporal loss of the GNSS b. End effects: Negligible	4	Loss of code lock is detected for all channels	Monitor code lock indicator and check if it crosses loss-of-lock threshold.	Issue event and continue processing (possible re-lock after propagation time-out or re-acquisition will naurally follow).	LSE
GNSS - 19			FLIGHT MODE	Total loss of GNSS signal/data >= 5s (TBD)	/	a. GNSS FAIL mode, loss of GNSS b. Degradation of navigation performances	3	Loss of code lock is detected for all channels	Monitor code lock indicator and check if it crosses loss-of-lock threshold.	Send Event and command transition to FAIL mode.	LSE
GNSS - 20	GNSS SW - NAV	Navigation (NAV)	FLIGHT MODE	GDOP above the limit (TBC)	/	a. Local effects: GDOP above the limit b. End effects: Negligible	4	GDOP is higher than TBD threshold	Monitor GDOP and check if it rises above threshold.	Issue event	LSE
GNSS - 21			FLIGHT MODE	Least-Squares Residuals over threshold	/	a. Local effects: GDOP above the limit b. End effects: Negligible	4	LSQ residuals higher than TBD threshold	Monitor LSQ residuals and check if they rise above threshold.	Issue event	LSE

3.4 Fault Tree Analysis

FTA definitions and procedures are referred to European Cooperation for Space Standardization ECSS-Q-ST-40-12C and International Electrotechnical Commission IEC 61025. Fault Tree Analysis is a deductive top-down method to identify faults, conditions, events and factors that may contribute to the occurrence of a defined top event, that usually leads to degradation of system performances or system's loss.

FTA is a graphic, organized analysis that represents the contribution of basic events or faults to a top event, being able of detecting multiple faults effects. It aims to identify factors/faults or combinations of faults that could affect the top event (from a qualitative approach), factor affecting system's reliability when FTA is used to evaluate the probability of occurrence of a failure (quantitative approach) and help in redundancy design. It also helps demonstrate the results obtained in other analyses, it helps improving system reliability and preventing or mitigating potential causes of the top event.

This analysis is particularly suited to be combined with FMEA for several reasons:

- the two techniques follow two opposite approaches, allowing to reach the same conclusions starting from different points and providing a complete analysis and a consistency check;
- FMEA is a single failure analysis, while FTA is both a single and a multiple failure analysis, and both are required by safety standards;
- FMEA helps in the identification of basic events while FTA helps with the identification of causes of potential hazards.

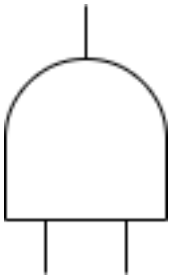
FTA main components are: all the relevant events, from the lowest level event to the defined top event, the logical gates (static or dynamic) and connection lines between events and gates. As for the FMEA, to successfully perform an FTA, it is necessary to have a detailed knowledge of the system.

The first step in the construction of a fault tree is a clear definition of the undesired top event and the boundaries of the system to be analysed. Then immediate causes are identified as inputs of the top event and these are further analysed systematically from top to bottom, using appropriate gates, to reach primary or basic events, depending from the chosen level of analysis. Then the fault tree is evaluated to identify and mitigate possible events that could lead to a system failure.

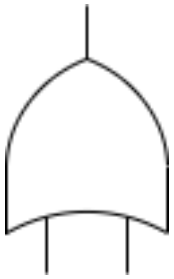
To clearly understand the fault tree developed during this analysis, it is necessary to introduce the symbols used, shown in Figure 3.4.

Furthermore, for this analysis, the top event is the Loss of Mission as specified during Failure Mode and Effect Analysis in Section 3.3.

The Fault Tree Analysis performed for the navigation unit system is depicted in Figure 3.5. In this Fault Tree, the failures have been divided between navigation unit SW related

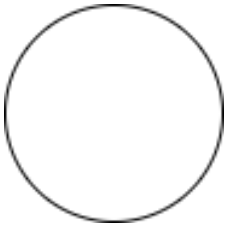


(a) AND gate: the output event only occurs if both inputs event occur.



(b) OR gate: the output event occurs if one of the inputs occurs.

(c) Top Event or Intermediate Event



(d) Basic Event, the lowest level event

Figure 3.4: Symbols used for the Fault Tree Analysis [21]

failures, Interfaces failures and Sensors failures, according to their level.

This analysis is very important because it allows to detect the types of errors that can be reproduced in the navigation unit simulator, thus it allows to identify the errors that are subjected to FDIR tasks and are involved in FDIR development and implementation. Furthermore, it allows to identify the points in which the FDIR systems must act, for detection and recovery purposes. In particular, the FDIR must perform an integrity check on sensors measurements, before and after measurement pre-processing, and can exploit Kalman filter outputs, as innovation and innovation covariance, to determine whether a GNSS measurement is faulty or not and whether the filter is convergent or not.

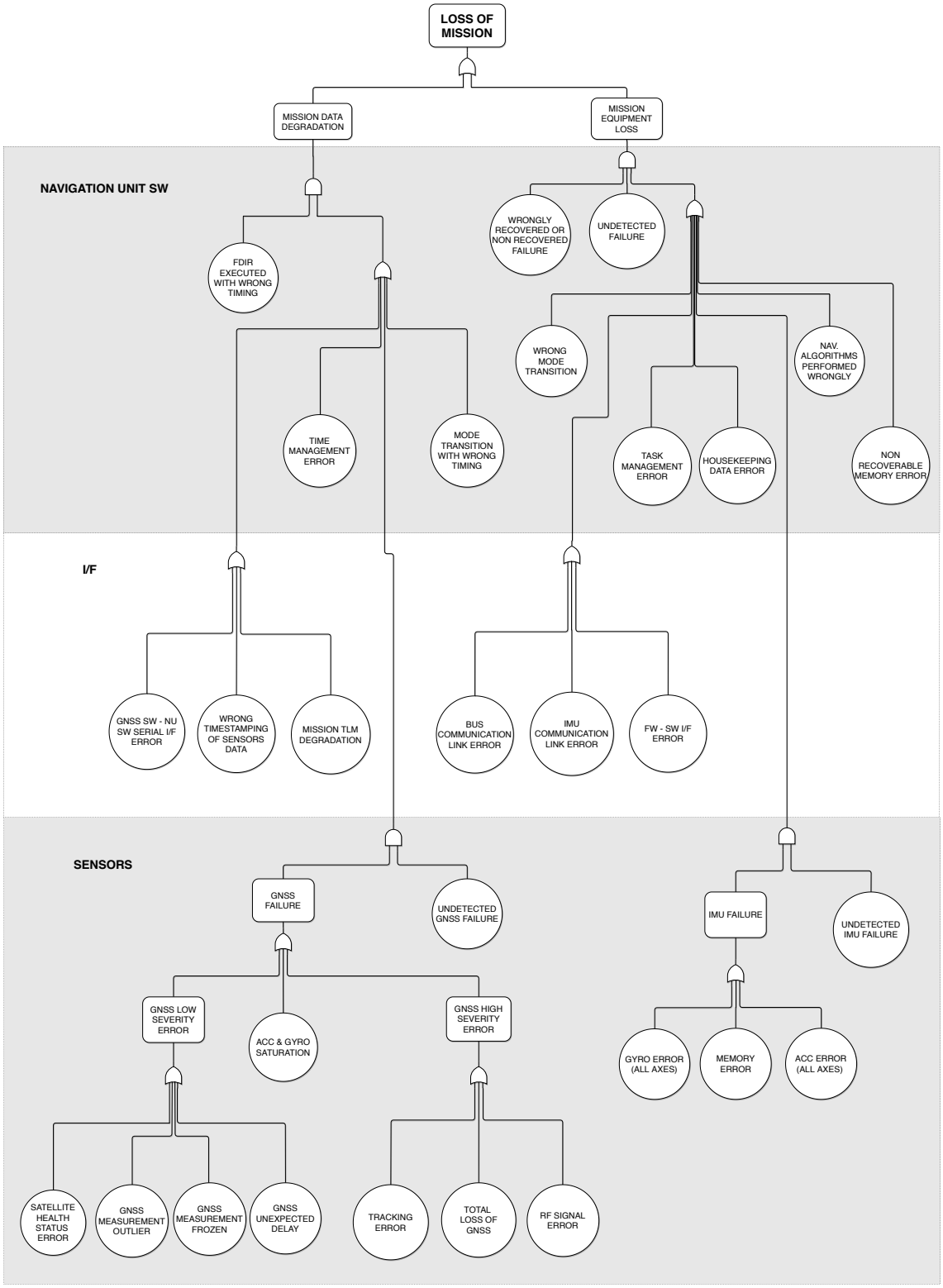


Figure 3.5: Fault Tree

3.5 Results of Preliminary Analyses

As result of these two analyses, a list of errors that can be reproduced in the simulator has been identified. The following list of errors is a compromise between the faults detected through FMEA and FTA, the focus of the analysis and the requirements implied in introducing faults in SENER simulator, to validate the FDIR system. In fact, the current study focuses on the detection of outliers in the measurements and on handling sensors flag in order to properly recover the unit, as memory and management software errors cannot be reproduced in the simulator. The faults that can be reproduced in the simulator are here listed:

- GNSS measurement spike
- GNSS frozen measurement (fixed or transient)
- GNSS invalid measurement undetected by GNSS (fixed or transient)
- GNSS measurement with unexpected delay
- Total loss of GNSS for a period of time lower than 5s (TBD[1]).
- Total loss of GNSS for a period of time bigger than 5s (TBD).
- Gyroscope measurement spike in one or more axes
- Gyroscope frozen measurement in one or more axes (fixed or transient)
- Gyroscope invalid measurement undetected by IMU in one or more axes (fixed or transient)
- Accelerometer measurement spike in one or more axes
- Accelerometer frozen measurement in one or more axes (fixed or transient)
- Accelerometer invalid measurement undetected by IMU in one or more axes (fixed or transient)
- Total loss of IMU output (fixed or transient)

The previous list of errors has been used to develop requirements and specifications of the Fault Injection Model described in Section 5.2, designed to validate the system.

[1]The value of 5 s is not definitive and a more realistic value has been determined during Validation campaign in Chapter 5.

Chapter 4

Design and Implementation

This Chapter deals with the description of FDIR design, logic and algorithms. First, an intensive search of suitable outlier detection algorithms for IMU and GNSS data has been carried out, leading to the selection of the implemented algorithms. Then, the logic of the FDIR system has been designed and modeled. The measurement checks and tests implemented for outlier detection on IMU and GNSS measurements are presented in this section, divided in two categories: basic FDIR checks, that includes preliminary consistency checks on measurements, and advanced FDIR methods, that involves more complicated algorithms, taking advantage of Kalman filter properties and motion equations. It is important to remember that IMU and GNSS already provide a validity signal for their measurements and some flags coming from internal Continuous Built-In Tests, regarding sensor's current state. The detection methods implemented are not aimed to substitute these indicators, but represent the last line of defense of the system against failures in data transfer.

The FDIR model including flag handling, system logic, outlier detection algorithms and recovery actions, has been developed and implemented in Simulink and finally it has been integrated in the hybrid navigation unit architecture model, inside the simulator developed by SENER Aeroespacial, in view of its validation, described in Chapter 5.

The Chapter is structured as follows: in the first Section, the basic design of the FDIR system is presented, focusing on the preliminary measurements check that have been implemented. Section 4.2 deals with advanced outlier detection methods for GNSS and IMU measurements. Both of these Sections focus on *Detection* and *Isolation* methods. Finally, Section 4.3 presents the final design of the FDIR system, including its logic and its interfaces, and some proposed Recovery actions.

4.1 Basic FDIR Design

In this Section, the basic design of the FDIR system is described. In particular, some preliminary detection checks on IMU and GNSS measurements are here presented.

4.1.1 Preliminary Checks

Preliminary checks implemented in FDIR design involves basically some consistency check of the measurements before their processing in the navigation. These checks includes frozen measurement checks and out of range measurement checks.

Regarding IMU measurements preliminary checks:

- **IMU frozen measurement check**: as IMU's data is output at a very high frequency, the implemented detection methods consists in raising a flag if a user-defined number of measurements are equals between them.
- **IMU out of range measurement check**: a simple and reliable way of detecting outliers in IMU measurements is to compare them against fixed thresholds and flag them as outliers if bigger than the thresholds. In this case, the thresholds are derived from IMU's design specifications, where a maximum value of non-gravitational acceleration and angular rate are specified. Knowing IMU's data output frequency, it's possible to obtain the maximum and minimum ΔV and $\Delta\theta$ that can be sensed by IMU, as shown in Equation (4.1) and (4.2).

$$a_{min}\Delta t < \Delta V < a_{max}\Delta t \tag{4.1}$$

$$\omega_{min}\Delta t < \Delta\theta < \omega_{max}\Delta t \tag{4.2}$$

Regarding GNSS, implemented preliminary checks are here listed:

- **GNSS frozen measurement check:** as GNSS measurements are output at low frequencies, the flag of frozen measurement is raised whenever two measurements are equals between them.
- **GNSS out of range measurement check**: for GNSS case, there are not well defined fixed threshold to implement an out of range measurement, because GNSS measurement may have variable frequency during some flight phases due to GNSS outages. For this reason, another type of detection method has been implemented, based on a variable threshold. In fact, taking advantage of Kalman filter outputs, i.e. position and velocity in inertial frame and their covariances, it is possible to design a control method based on the difference between the GNSS measurement $x^{(k)}_{GNSS}$ and $v^{(k)}_{GNSS}$ and previous navigation estimates of position $x^{(k)}_{NAV}$ and velocity $v^{(k)}_{NAV}$, using an adaptive threshold calculated as a user-defined number h of standard deviations σ, as shown in Equations (4.3) and (4.4). In this way, after a GNSS outage, when GNSS measurement and previous estimated state may present the bigger difference,

also position and velocity covariance has grown, producing a higher threshold. To implement this method, it's necessary to compare new GNSS measurements with the estimated state at GNSS acquisition time, so several previous states have to be stored. Furthermore, this check is executed at navigation frequency, because previous state and covariance are needed.

$$x_{GNSS}^{(k)} - x_{NAV}^{(k)} < h \cdot \sigma_x^{(k)} \tag{4.3}$$

$$v_{GNSS}^{(k)} - v_{NAV}^{(k)} < h \cdot \sigma_v^{(k)} \tag{4.4}$$

4.2 Advanced Robust FDIR Design

This Section describes GNSS and IMU outlier detection methods, focusing on the studied solution and their implementation in FDIR system.

4.2.1 GNSS outliers detection

GNSS receiver is the new element of this navigation system with respect to conventional ones, and its presence greatly improves navigation performances, even though it is not a critical component, as the navigation unit can keep working even in case of GNSS loss. Nevertheless, GNSS outlier detection is a necessary task, because a faulty measurement that is incorporated in navigation solution computation may lead to filter divergence. The proposed outliers detection methods are based on innovation analysis, because the filter is designed in a two-rate processes that allow to optimize the Kalman filtering, by separating its computation from the state and covariance update. The proposed methods are here listed:

- CUSUM test
- Generalized Likelihood Ratio test
- Parity Space method

Of these methods, only the first has been implemented, because the other two presents some problems that do not allow their implementation for this application. These two latter methods are here briefly described, and an overview of the problems that prevent their application are listed:

- **GLR test** [22]: this method uses the likelihood approach to calculate likelihood ratio and evaluate the probability of outlier. It is based on expressing the additive change as linear regression with the innovation as measurements. The nominal Kalman filter is applied, under the hypothesis of no change, then the linear regression quantities are calculated, which in turn allows to calculate the likelihood ratio. The logarithm of this likelihood ratio is used as test statistics and compared against a threshold,

to detect outliers. The method presents several problems: first, the GLR test is the optimal test statistics when the change magnitude is known and only one change is assumed. In order to adapt the GLR for the application considered, it is necessary to calculate the test statistics for each change time, leading to a huge computational load, or for a restricted number of change times, using a sliding window, allowing to detect the outlier but may also introduce delays in the detection, preventing the removal of the faulty measurement. Furthermore, decision threshold are not easy to tune.

- **Parity Space method** [23]: this method is based on the generation of residuals using parity relations to pass from measurement domain to parity (or residual) space. In this application, as no redundancy is provided, the temporal redundancy approach has been considered. This approach is based on using measurements over a time interval, as illustrated in Figure 4.1.

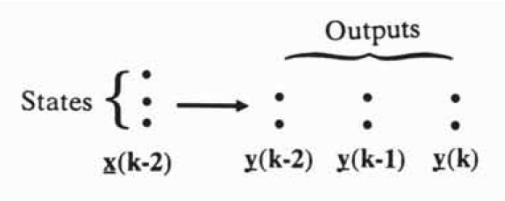


Figure 4.1: Temporal analytical redundancy from [23]

The parity relations are expressed as shown in Equation (4.5)

$$\underbrace{\begin{Bmatrix} y(k-s) \\ y(k-s+1) \\ . \\ . \\ y(k) \end{Bmatrix}}_{\mathbf{Y}(k)} = \underbrace{\begin{Bmatrix} \mathbf{H} \\ \mathbf{HF} \\ . \\ . \\ \mathbf{HF^s} \end{Bmatrix}}_{\mathbf{Q}} \tag{4.5}$$

where $\mathbf{H}$ and $\mathbf{F}$ are observation transition matrix and state transition matrix respectively. The method consists in finding the matrix $\mathbf{V}$ that allows to generate residuals through Equation (4.6). The generation of the residuals is then followed by the generation of a test statistics that is compared against a threshold to detect outliers.

$$r(k) = \mathbf{V}\mathbf{Y}_k \tag{4.6}$$

The matrix $\mathbf{V}$ is called *residual generator* and must satisfy the condition that the residuals that it generates are independent of the system operating state, leading to

Equation (4.7).

$$\mathbf{VQ} = \mathbf{0} \tag{4.7}$$

Equation (4.7) leads to one of the problems of the method: the computational load needed to solve this equation at every iteration is huge. Furthermore, this method has to be implemented in a sliding window fashion, leading to possible delays in outlier detection, preventing the removal of the faulty measurement.

4.2.1.1 CUSUM test

The simplest form of outlier detection in Kalman filter is to use a distance measure with a stopping rule test on normalized innovation, using Kalman filter known properties to evaluate filter performances. For this particular case, this method can be used for outlier detection because the filter designed by SENER Aeroespacial is a two-rate filter, meaning that the effective state update is delayed with respect to error state filter update computation. For this reason, it is possible to use error state filter innovation and innovation covariance to perform outlier detection on GNSS measurements and to exclude a faulty measurement from state update when an outlier is detected.

The method is based on a whiteness innovation test. In fact, in an ideal Kalman filter, the innovation is supposed to be white and zero-mean. The whiteness test used is the Cumulative Sum (CUSUM) test (see [22]). A schematic view of the method is depicted in Figure 4.2.

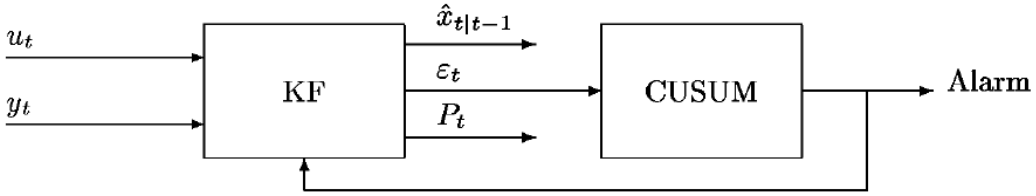


Figure 4.2: Whiteness innovation test for Kalman filter, where ε_t is the innovation [22]

The distance measure used for the method is based on normalized innovation for vector measurements shown in Equation (4.8):

$$s_t = \frac{1}{n_y}\mathbf{1}_{n_y}\mathbf{S}_t^{-1/2}\nu_t \tag{4.8}$$

where n_y is the dimension of the measurements, $\mathbf{S}_t$ is innovation covariance and ν_t is the innovation. As the filter is implemented with sequential measurement, innovation covariance is a vector instead of a matrix; this helps a lot with computational load reduction.

Furthermore, normalized innovation should be $N(0,1)$ distributed under the assumption of zero outliers.

Once a distance measure has been obtained, the CUSUM test is applied. The CUSUM test generates a test statistics using a stopping rule, then the statistics is thresholded to detect outliers. The test statistics consists in summing up the input s_t, until the threshold is reached, generating an alarm. With a white noise input, the test statistics is doomed to drift away, for this reason, another parameter is introduced: a small drift term d is subtracted at each step. Furthermore, the test statistics is reset to 0 every time that it has a negative value, to prevent negative drift which would increase the time to detection after a change. The CUSUM algorithm is presented in Equation (4.9).

$$\begin{aligned} g_t &= g_{t-1} + s_t - d \\ g_t &= 0 \quad \text{if } g_t < 0 \\ g_t &= 0 \quad \text{if } g_t > h \text{ and alarm outlier detection} \end{aligned} \tag{4.9}$$

The design parameters are the threshold h and the drift d. These two parameters need to be tuned in order to have a properly working outlier detection method. Usually, decreasing the drift gives a very fast change detection, while increasing it reduces the number of false alarms. The threshold is then tuned consequently. This method is simple and reliable, and provides a good measure of filter divergence. For the validation campaign described in Chapter 5, the following values of drift and threshold have been selected:

- Drift $d = 3$.
- Threshold $h = 10$.

4.2.2 IMU outliers detection

As resulted from analyses performed and described in Chapter 3, Inertial Measurement Unit is a critical component of the navigation system. For this reason, it is necessary to perform outlier detection in IMU measurements to avoid faulty measurement being incorporated with high frequency in the navigation solution. The following methods have been identified as viable solutions for IMU measurements outliers detection:

- Kernel Density Estimator with Binned Summary
- Predictor-Corrector Algorithm

Both methods are described in following sections.

4.2.2.1 Kernel Density Estimator with Binned Summary

This method is based on an online algorithm for real-time outlier detection over streaming data described in [24]. The proposed method is based on a sliding window and expired points are mined into bins: they contribute to the prediction of outlier for new data through bins statistics that is maintained during simulation.

In order to estimate the probability density function, a Kernel Density Estimator is used. This estimator allows to dynamically estimate probability density function as new data arrives, and increases the density for each point by a probability $p(x_i)$, distributing the rest $1-p(x_i)$ to the point's neighbours according to the Kernel function used. For this application, a Gaussian Kernel is used, in order to have a smooth estimation, as it distributes probability of occurrence to all data points from $-\infty$ to $+\infty$.

To avoid storing all observed data points, a non-overlapping sliding window is implemented. This allows to reduce computational burden, as the kernel density function is calculated over the points contained in the window. The dimension of the window is defined as the frequency of IMU's data divided by the frequency of navigation data. The Gaussian kernel density function applied is described in Equation (4.10).

$$f_{window}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{(2\pi)^{D/2}H}\exp -\frac{1}{2}\left(\frac{x-x_i}{H}\right)^2 \tag{4.10}$$

In Equation (4.10), n is the length of the window expressed as the number of points contained in it, D is the dimension of data points and H is the bandwidth of Kernel density function: the bandwidth is the parameter used to control the extent to which probability is distributed over data point other than x_i. The higher the bandwidth, the more the probability is distributed over x_i neighbours. The implemented probability density function is exclusively based on the data points contained in the window, ignoring historical data. Therefore, the results of outlier detection based on the probability density function calculated using only the data inside the window might be inaccurate, because previous data are not taken into account.

As said before, in order to maintain statistics of previously observed and expired data, a binned summary is implemented. The binned summary consists in dividing each dimension of the data points in bins and then mining expired points into these bins. Then, only a bin statistics is maintained, allowing to keep track of historical points. For each bin, the number of data points and the mean value vector of each bin is maintained and updated through the formulas explained in the following paragraph.

Assuming that $n-1$ windows of data have been processed, if C_i^{n-1} and M_i^{n-1} indicate respectively the number of points in bin i and the mean value of the points in bin i up to the $n^{th}-1$ window, to update the number of points and their mean value in the bin after the processing of the n^th window, Equations (4.11) and (4.12) are used:

$$M_i^n = \frac{c_i^n \mu_i^n + C_i^{n-1} M_i^{n-1}}{c_i^n + C_i^{n-1}} \tag{4.11}$$

$$C_i^n = c_i^n + C_i^{n-1} \tag{4.12}$$

where c_i^n and μ_i^n are the number of points in bin i and their mean value at current window n.

Before the expiration of the points from the window, the bin statistics is calculated and maintained. Before this step, though, a slightly modified kernel density function is applied to the bins, in order to consider historical data in outlier detection. This kernel density function is described in Equation (4.13).

$$f_{bin}(x) = \frac{1}{C} \sum_{i=1}^{m} \frac{C_i}{(2\pi)^(D/2)H} \exp -\frac{1}{2} \left(\frac{x - M_i}{H} \right)^2 \tag{4.13}$$

where C is the sum of all expired points and m is the number of bins. Finally, the total probability density function is calculated as described in Equation (4.14).

$$f(x) = f_{window}(x) + f_{bin}(x) \tag{4.14}$$

To detect an outlier, the outlier factor is defined for every data point as the inverse of probability density function. This outlier factor is then compared against a threshold that is updated dynamically as it is calculated with the average density p_{avg} of all points in $f(x)$ and a tuning parameter $0 < \xi < 1$, as described by Equation (4.15).

$$f_0 = \frac{1}{f(x)} > \theta = \frac{1}{p_{avg}\xi} \tag{4.15}$$

4.2.2.2 Predictor-Corrector Algorithm

This algorithm is based on a multistep ODE solver, used by some advanced navigation system. It consists in a predictor step and a corrector step that solve the differential equations that govern launcher motion and the comparison of the two solutions, to detect outliers in IMU measurements. For this particular case, a 4th order predictor-corrector solver has been implemented, using Adams-Bashforth formula (4.16) for the predictor step and Adams-Moulton formula (4.17) for the corrector step. A schematic view of the method can be seen in Figure 4.3.

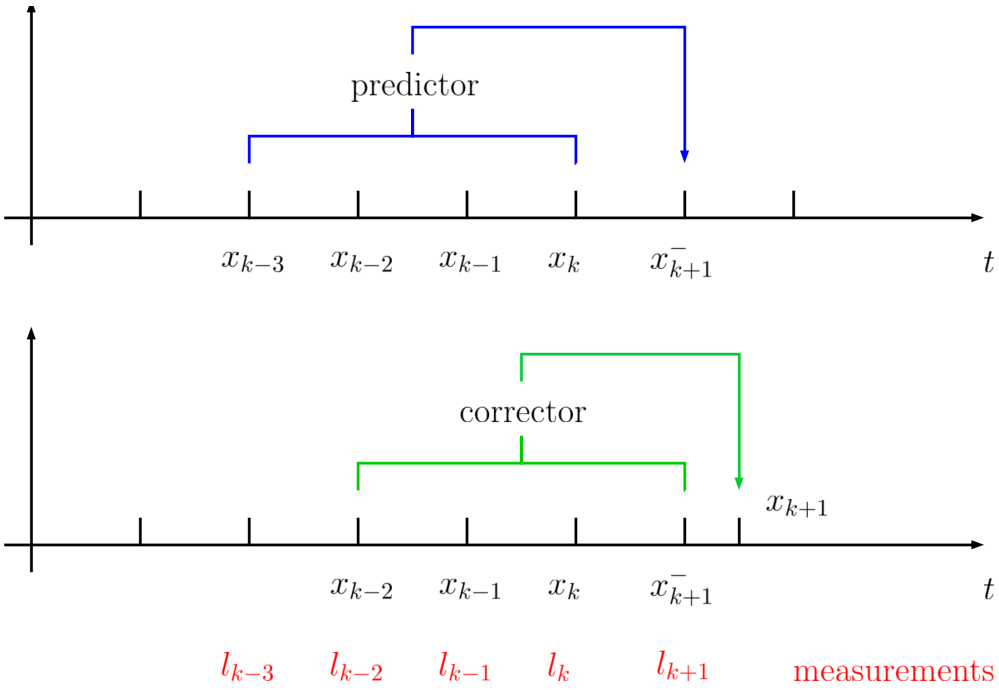


Figure 4.3: 4th order Adams predictor-corrector ODE solver

In Figure 4.3, x_k represents the state at time k, while l_k represents the measurement at time k. In the prediction step, it is possible to compute the intermediate solution x^-_{k+1} through Adams-Bashforth formula described in Equation (4.16):

$$x^-_{k+1} = x_k + \frac{h}{24}\left(55f(x_k, l_k) - 59f(x_{k-1}, l_{k-1}) + 37f(x_{k-2}, l_{k-2}) - 9f(x_{k-3}, l_{k-3})\right) \quad (4.16)$$

and then it is possible to compute the final solution in the corrector step using Adams-Moulton formula described in Equation (4.17):

$$x_{k+1} = x_k + \frac{h}{24}\left(9f(x^-_{k+1}, l_{k+1}) + 19f(x_k, l_k) - 5f(x_{k-1}, l_{k-1}) + f(x_{k-2}, l_{k-2})\right) \quad (4.17)$$

In Equations (4.16) and (4.17), h refers to the time-step, while $f(x_k, l_k)$ indicates the governing equations (INS-mechanization equations), described by Equation (2.1) in Section 2.1. For known state from time $k-3$ up to time k, determined by the Kalman filter, and for previously cleaned measurement from time $k-3$ up to time k, the difference between predicted and corrected solution only depends on the new measurement l_{k+1} at time $k+1$. Therefore, if an outlier is present in measurement l_{k+1}, the difference between predictor and corrector solution will be bigger than a threshold, showing the presence of the outlier.

In order to compute the difference between predictor solution x_{k+1}^- and corrector solution x_{k+1}, the Mahalanobis distance has been used, as shown in Equation (4.18):

$$||x_{k+1} - x_{k+1}^-|| = d_M = \sqrt{\left(x_{k+1} - x_{k+1}^-\right)^T \mathbf{C}^-1 \left(x_{k+1} - x_{k+1}^-\right)} > \sqrt{k} \tag{4.18}$$

where $\mathbf{C} = 2\mathbf{S}$ and $\mathbf{S}$ is state covariance matrix. The threshold k in Equation (4.18) should be tuned properly as the critical value of the correspondent distribution. If the difference between corrector and predictor follows a multivariate normal distribution, the squared Mahalanobis distance follows a χ^2 distribution, and so:

$$k = \chi^2_{d,1-\alpha_0} \tag{4.19}$$

where $d = dim(x_k)$ are the dimensions of the state vector and α_0 is the so-called test significance level, i.e. the probability of having a false alarm.

4.3 FDIR design description

This Section describes final FDIR design description, focusing on FDIR logic and hierarchy, and its interfaces with other elements of the simulator, describing model parameters, inputs and outputs data.

4.3.1 FDIR logic

In order to design a proper functioning FDIR system, it's very important to pay attention to FDIR's logic and to define a hierarchy as established in Section 3.1. For this case, FDIR's operations can be divided in two level:

- **IMU and GNSS FDI**: the first level regards flag handling and outliers detection and isolation in sensor's measurement. In fact, several flags regarding IMU's and GNSS's state are available for FDIR system, in particular a measurement validity flag. They all are handled by the system, to obtain information about sensors' state and are followed by some preliminary checks in order to identify and isolate faulty measurements. Once the cause of the fault has been identified, a series of flags is transferred to the upper level.
- **Navigation FDIR**: this level takes as input the flags from lower level and undertakes corrective actions. Furthermore, this level deals with navigation solution monitoring, through CUSUM test on innovations (also used as GNSS outlier detection method). The basic hierarchy of the FDIR system is depicted in Figure 4.4.

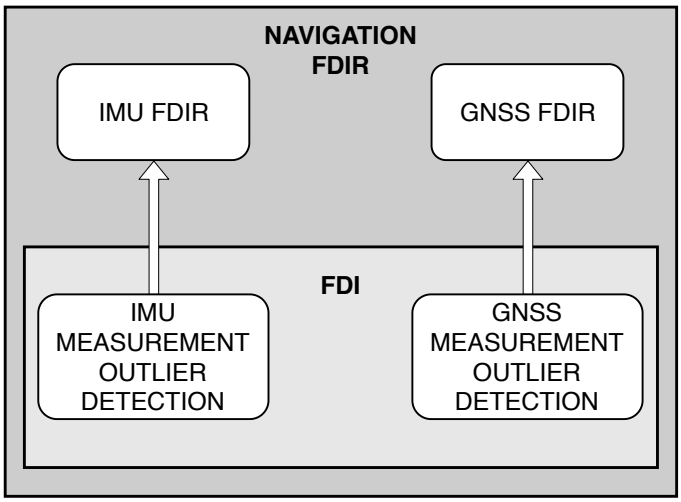


Figure 4.4: Basic FDIR hierarchy and levels

GNSS and IMU operations/decisions flow in each level of FDIR structure is described in the following Sections. In order to better understand following Figures, where FDIR's logic is depicted, a legend that describes all the blocks used is provided in Figure 4.5.

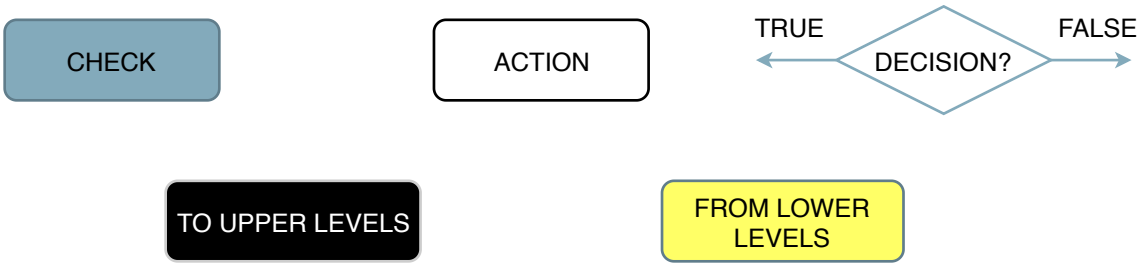


Figure 4.5: FDIR's logic diagrams legend

4.3.1.1 GNSS FDI

For the first level of FDIR structure, the GNSS operation flow is depicted in Figure 4.6. Every time that a new GNSS measurement is available, `gnss_validity` is checked and then Preliminary checks are executed on GNSS measurements. The parameter passed to higher level is `gnss_validity`. The logical steps of this level are here listed:

1. `gnss_validity` check: if the measurement is not valid, GNSS Severity error is flagged, according to fault duration.
2. If the measurement is valid, GNSS frozen measurement check is executed. If the measurement is frozen, `gnss_validity` is set to 0.

3. If the measurement is not frozen, GNSS Out of Range measurement check is executed. If an outlier is detected `gnss_validity` is set to 0, otherwise `gnss_validity` remains valid, meaning that the first level has not detected any outlier.

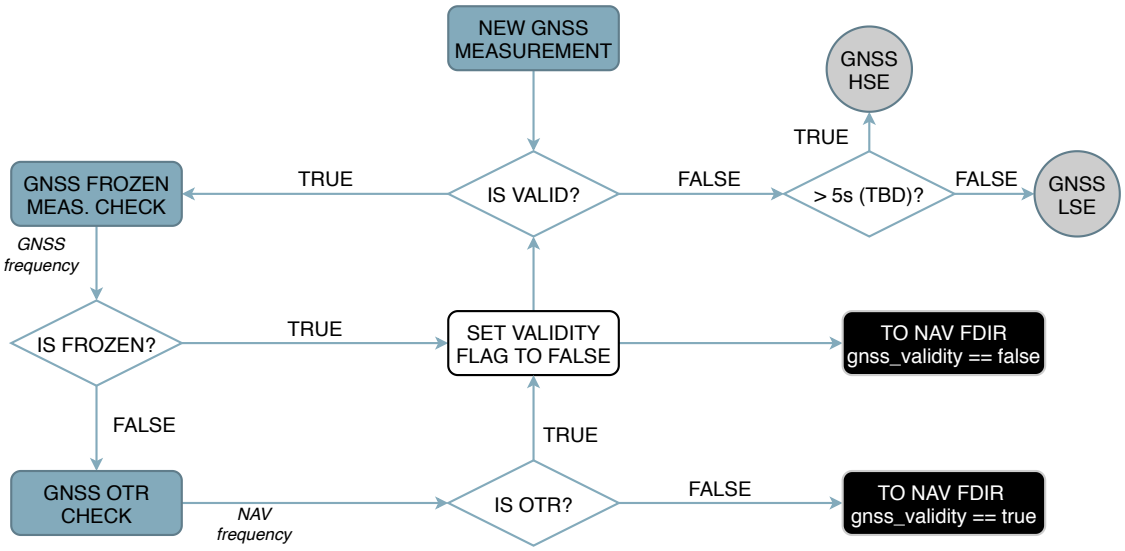


Figure 4.6: GNSS FDI (1st level) logic

4.3.1.2 NAVIGATION FDIR: GNSS

In the second level, proposed detection and recovery actions are executed, as depicted in Figure 4.7. In particular, if GNSS signal coming from 1st level is not valid, the system transfer this information to navigation functions that are designed to discard an invalid GNSS measurement. If the measurement is valid, a final check is performed on Kalman filter innovations using CUSUM test that flags outliers in GNSS measurements and avoid the update of state and covariance. The steps followed by the system during this level are here listed:

1. `gnss_validity` check: if the measurement is not valid, then the navigation functions discard it and do not compute the update of state and covariance.

2. If the measurement is valid, navigation functions compute the update of state and covariance, without executing the update.

3. CUSUM test is executed using innovation calculated after the computation of KF update. If an outlier is detected, `no_update_flag` is set to 1. This parameter is passed to navigation functions that discard the update just computed in the error state Kalman filter. If no outlier is detected, the measurement is considered valid and state update is performed, incorporating GNSS measurement in navigation solution.

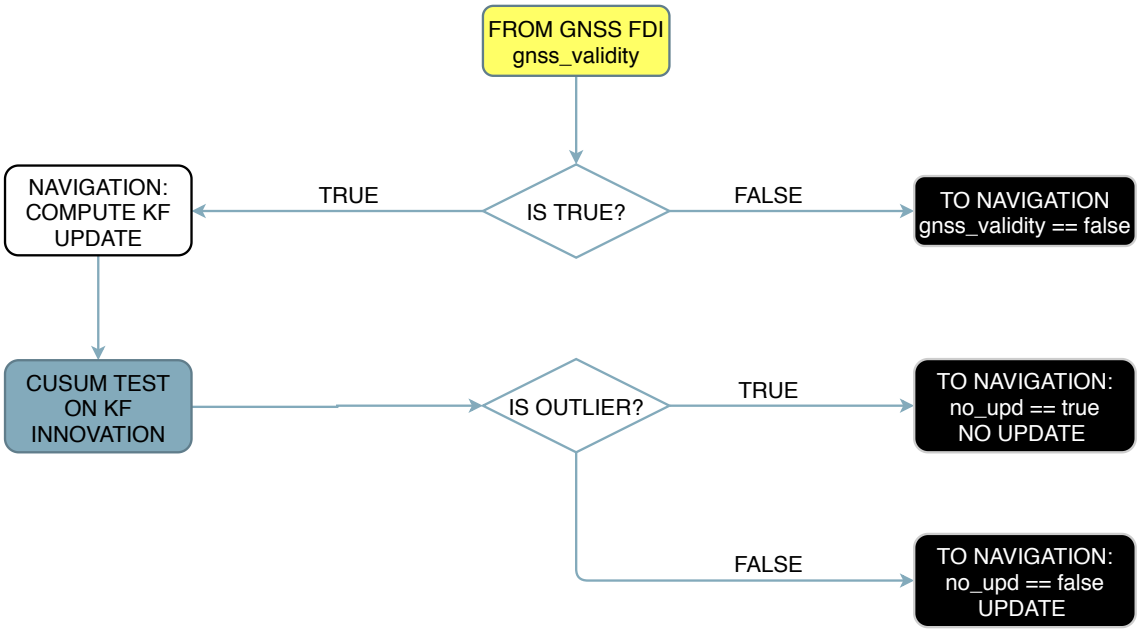


Figure 4.7: NAV_FDIR: GNSS (2nd level) logic

4.3.1.3 IMU FDI

Regarding IMU FDI, the first level is a little more complicated than GNSS's one due to the fact that IMU is a critical component in navigation system, while GNSS measurement can be discarded with the unit still functioning (though with degraded performances). The logic followed in this level is depicted in Figure 4.8. Every new IMU measurement comes with a series of flags that describe the internal state of the IMU, obtained internally by the IMU through Continuous Built-In Tests (CBIT). In particular, IMU provides information about temperature and its derivative, about gyroscopes and accelerometers saturation and a $\Delta V/\Delta\theta$ validity flag. The first task executed by IMU FDI is flag handling. Then preliminary checks are executed, with proper recovery actions triggered by upper level. Finally, advanced outlier detection is performed, again with proper recovery action triggered by upper level. Naturally when an outlier or a frozen measurement is detected by the described methods, IMU's measurement is flagged as invalid. The parameters passed to the upper level are `imu_validity`, `T_flag`, `ASflag`, `GSflag`, `imu_frozen`, `imu_OTR`, `imu_kde`, `imu_pc`. The steps followed by the system are listed below:

1. CBIT decodification and flag handling: if the measurement is not valid, the CBIT parameters are evaluated to isolate the fault and send to upper level the proper information (temperature out of range, accelerometer of gyroscope saturation and invalid measurement through the respective parameters `T_flag`, `GSflag` and `ASflag` indicating which accelerometer or gyroscope is saturated, and `imu_validity`);

2. If the measurement is valid, IMU frozen measurement check is performed: if an

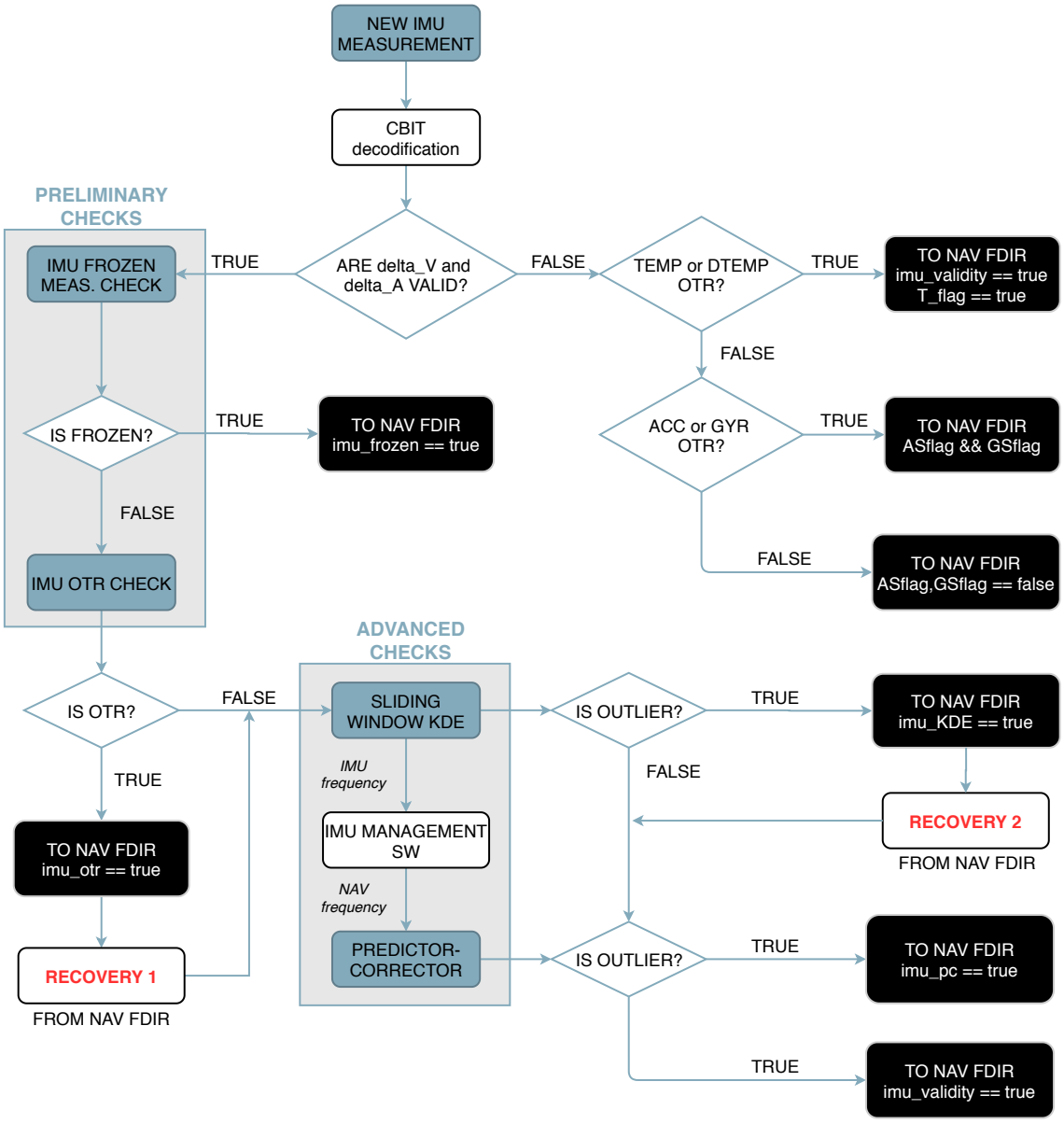


Figure 4.8: IMU FDI (1st level) logic

outlier is detected, the flag `imu_frozen` is used to pass the information to the upper level regarding which sensor is giving frozen measurement.

3. If the measurement is not frozen, OTR check is executed: if an outlier is detected, the information is passed to the upper level through the flag `imu_otr` that indicates which sensor is giving measurement out of range.

4. A first recovery action is executed by the upper level in case an outlier is detected

by Preliminary Checks, as indicated by the event **RECOVERY 1**.

5. If there are no outliers, Sliding Window KDE is applied: the detection of outlier by this algorithm is passed to upper level through `imu_kde` flag, indicating in which sensor an outlier has been detected.

6. A second recovery happens, triggered by the upper level in case of outlier detection by KDE, as indicated by the event **RECOVERY 2**.

7. Management of IMU's signals is executed.

8. Predictor-Corrector algorithm is implemented and `imu_pc` flag, indicating that the whole measurement is faulty, is used to pass the information to the upper level. This algorithm is already implemented including a sort of recovery: in order to detect outliers in the new measurement, the previous ones should have been already "cleaned", that is, faulty measurements are substituted with frozen ones, using the previous correct measurement.

As can be seen, the two levels are more interconnected for IMU than for GNSS, also due to the fact that the different checks are executed on data with different frequencies.

4.3.1.4 NAVIGATION FDIR: IMU

The second level of the FDIR part dedicated to IMU recovery is also a little bit more complicated with respect to GNSS's one. First, it has to deal with data available at different frequency (*IMU frequency* and *Navigation frequency*); furthermore, recovery algorithms are implemented at different stages, according to the frequency and the information coming from the lower level. The logic implemented in this level is depicted in Figure 4.9.

The information regarding IMU's flag handling and FDI preliminary checks, coming from lower level, is processed and the flags `acc_rec` and `gyr_rec` are generated, indicating in which sensor the fault has been detected. If a fault has been detected, the systems performs the **RECOVERY 1**. In particular, this recovery is performed according to the following logic:

a) Recovery applied to the whole measurement (accelerometers and gyroscopes) if IMU's measurement is not valid.

b) Recovery applied to the corresponding sensors when a sensor's saturation is detected, when a sensor is giving frozen measurement or when a sensor is giving out of range measurements.

At the moment, two different types of actions have been implemented for **RECOVERY 1**:

- Frozen Recovery: using the previous correct measurement as constant value.

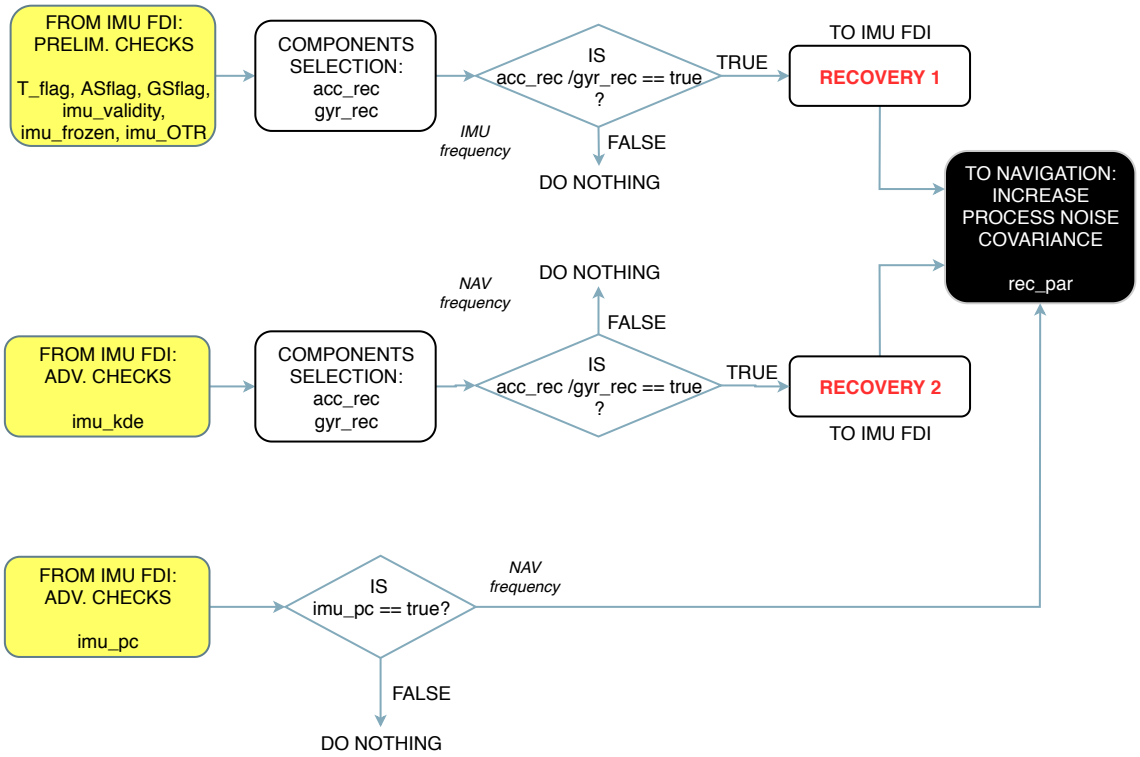


Figure 4.9: NAV_FDIR: IMU (2nd level) logic

- Prediction Recovery: trying to interpolate data profile to have an estimation of the new measurement. The prediction is performed using a spline or a 4-step backward derivative.

The information regarding the result of Sliding Window Kernel Density Estimator is used to generate the flags `acc_rec` and `gyr_rec`, which indicates in which sensor the fault has been detected. If a fault has been detected by IMU FDI (1st level), then the **RECOVERY 2** is performed: in this case, the system takes the previous correct measurement and uses it as a constant value. This is due to the fact that KDE algorithm only analyzes the stream of data, without considering launcher motion equation, so it can only detect short outliers and the difference between using Frozen or Prediction Recovery for short outliers is insignificant. For this reason, the Frozen recovery has been selected, as its simpler, more reliable and lighter in terms of computational load. The Frozen recovery is executed according to the following logic:

a) Recovery is applied to the accelerometer output if an outlier is detected into these sensors.

b) Recovery is applied to the whole measurement if an outlier is detected into gyroscopes, because the KDE algorithms works with batches of data at IMU frequency and gives the output flags at Navigation frequency, after Management SW, which

involves a rotation of the measurements, calculated using gyroscopes values. For this reason, if a gyroscope measurement is corrupted, after processing in the Management SW, the whole measurement will be corrupted.

Finally, information from Predictor-Corrector algorithm coming from 1st level, together with the flags `acc_rec` and `gyr_rec` generated after Sliding Window KDE and preliminary checks, is used to generate some parameters that are passed to Navigation. These parameters, `rec_par`, indicate if an outlier has been detected and corrected and in which sensor. This is used inside Navigation software to increase process noise covariance matrix **Q** by a user-defined factor, to take into account the error introduced in the measurement and increase state uncertainty during propagation, improving filter performances.

4.3.2 FDIR model design and interfaces

In this section, the architecture of the FDIR model developed for the integration in the hybrid navigation simulator is described, including the description of input and output signals and of model parameters. The general view of the system developed in Simulink is depicted in Figure 4.10.

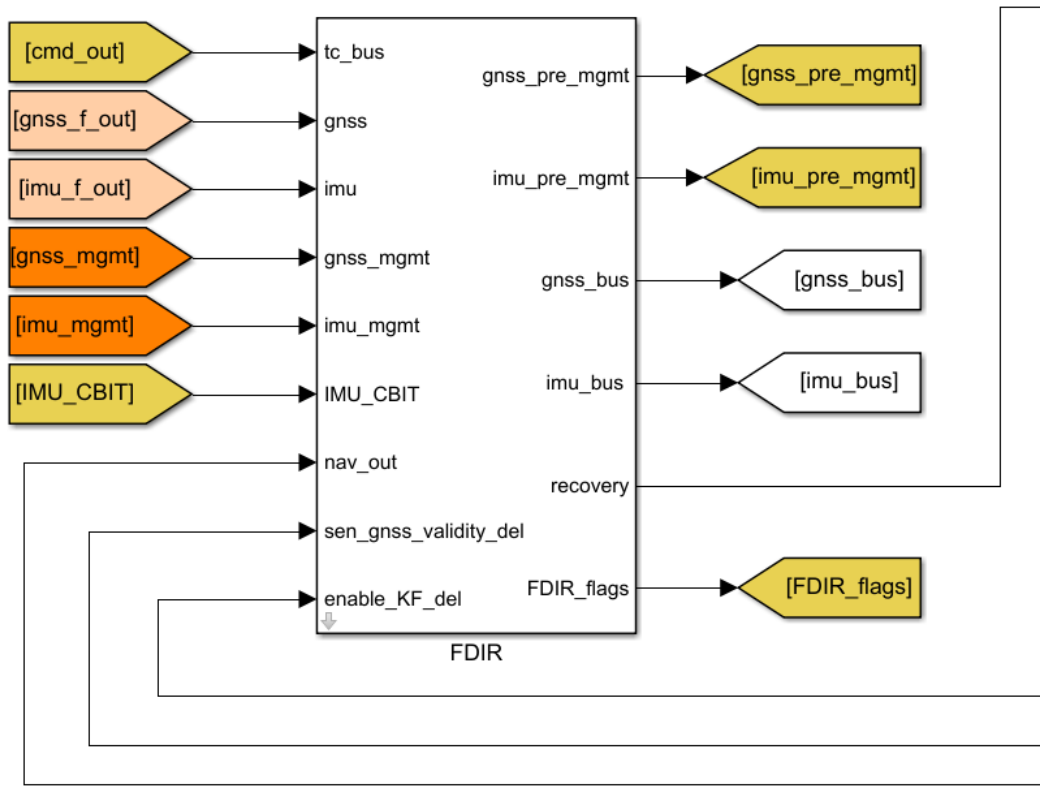


Figure 4.10: FDIR model integrated in SENER Aeroespacial hybrid navigation simulator

In Figure 4.10, the input signals `nav_out`, `sen_gnss_validity_del` and `enable_KF_del` come from Navigation block as feedback, while the output signal `recovery` is an input of the Navigation block. The output parameter `FDIR_flags` is a signal that allows to evaluate the proper functioning of the system, especially for detection and isolation.

The input signals are described in Table 4.1.

Table 4.1: FDIR model inputs - size = [signals(total_size)]

Inputs	Size	Units	Description
`cmd_out`	[(2)]	N/A	Bus containing the commands used to command mode transition.
`gnss_f_out`	[4(8)]	N/A	Bus containing GNSS output data after fault injection. It contains 4 signals (GNSS position, velocity, timetag and validity)
`imu_f_out`	[4(8)]	N/A	Bus containing IMU output data after fault injection. It contains 4 signals (IMU non-gravitational acceleration, angular rate, timetag and validity)
`gnss_mgmt`	[4(8)]	N/A	Bus containing GNSS output data feedback from management software, that is at Navigation frequency. It contains 4 signals (GNSS position, velocity, timetag and validity)
`imu_mgmt`	[4(8)]	N/A	Bus containing IMU output data feedback from management software, that is at Navigation frequency. It contains 4 signals (IMU non-gravitational acceleration, angular rate, timetag and validity)
`nav_out`	[-]	N/A	Bus containing the estimated navigation solution in terms of position, velocity and attitude (and their covariance) in different reference frames, the estimation of sensor's errors and useful information to monitor the integrity of the filter, like innovation and innovation covariance.
`IMU_CBIT`	[1,1]	N/A	Integer that represents the flag resulting from IMU's CBIT, indicating IMU's state. It contains information about imu validity, temperature and/or sensors validity and saturation, when converted to a binary number.
`gnss_valid`	[1,1]	N/A	Signal that indicates GNSS validity as processed by Kalman Filter.
`enable_KF`	[1,1]	N/A	Signal that indicates when the update is computed, that is when the filter is executed.

The output signals are described in Table 4.2.

Table 4.2: FDIR model outputs - size = [signals(total_size)]

Outputs	Size	Units	Description
`gnss_pre_mgmt`	[4(8)]	N/A	Bus containing GNSS output data after GNSS preliminary checks (after GNSS FDI) and before management software. It contains 4 signals (GNSS position, velocity, timetag and validity). It transmits data at GNSS frequency.
`imu_pre_mgmt`	[4(8)]	N/A	Bus containing IMU output data after preliminary checks and KDE and before management software. It contains 4 signals (IMU non-gravitational acceleration, angular rate, timetag and validity). It transmits data at IMU frequency.
`gnss_bus`	[4(8)]	N/A	Bus containing GNSS output data after FDIR tasks. It contains 4 signals (GNSS position, velocity, timetag and validity). This bus is an input for Navigation block, i.e. it transmits data at Navigation frequency.
`imu_bus`	[4(8)]	N/A	Bus containing IMU output data after management software, that is at navigation frequency. It contains 4 signals (IMU non-gravitational acceleration, angular rate, timetag and validity). This bus is an input for Navigation block, i.e. it transmits data at Navigation frequency.
`recovery`	[3(8)]	N/A	Bus containing the flags needed to perform recovery action inside navigation block. It contains `no_update_flag` as result of the GNSS CUSUM test, to avoid the state and covariance update, and the flags `rec_par`, indicating when to increase the process noise covariance.
`FDIR_flags`	[38]	N/A	Bus containing all the flags generated and handled by the FDIR model. These parameters are used to verify the proper functioning of the system and its performances in fault detection and isolation.

It is important to notice how FDIR system is a highly integrated system, because it handles signals from different sources at different frequency and also exploits feedback from Navigation block, to which it sends commands to execute recovery actions. For the model, it is necessary to handle signals taken from different points because detection algorithm and recovery actions implemented must be executed at different frequencies, after or before management software, making the internal structure of the system a lot more complicated. In the following Chapter 5, where the simulator is described, the data

flow involving FDIR system is clearly depicted in Figure 5.1.

Finally, FDIR model parameters are listed in Table 4.3.

Table 4.3: FDIR model parameters (N = Sliding window length - B = number of bins)

Parameters	**Size**	**Units**	**Description**
`gnss_froz`	[1,1]	N/A	Number of GNSS frozen measurement allowed before flag raising.
`gnss_OTR`	[1,1]	N/A	Number of σ that GNSS measurement cannot exceed, with respect to state covariance.
`CUSUM_drift`	[1,1]	N/A	Drift used in CUSUM test to calculate test statistics.
`CUSUM_thrs`	[1,1]	N/A	Outliers threshold for CUSUM test statistics.
`imu_froz`	[1,1]	N/A	Number of IMU frozen measurement allowed before flag raising.
`imu_dv_max`	[1,1]	[m/s]	Maximum ΔV allowed for IMU Out of range check.
`imu_da_max`	[1,1]	[deg]	Maximum $\Delta\theta$ allowed for IMU Out of range check.
`imu_dv_min`	[1,1]	[m/s]	Minimum ΔV allowed for IMU Out of range check.
`imu_da_min`	[1,1]	[deg]	Minimum $\Delta\theta$ allowed for IMU Out of range check.
`imu_pc_thrs`	[1,1]	[N/A]	Predictor-Corrector threshold.
`scal_mat`	[36,36]	[N/A]	Covariance scaling matrix.
`SW_length`	[1,1]	[N/A]	KDE sliding window length.
`bandwidth`	[N,1]	[N/A]	KDE bandwidth
`C_i_0`	[B,1]	[N/A]	Counting vector initialization.
`M_i_0`	[B,1]	[N/A]	Mean vector initialization
`bins`	[B+1,1]	[N/A]	Bins initialization. One for each sensor.
`KDE_thrs`	[1,1]	[N/A]	KDE function threshold. One for each sensor.

A detailed description of the flags generated and handled by FDIR is shown in the following list:

- `IMU_flags`: 11 flags containing information about IMU's state. In particular, the flags handled by FDIR system are those relative to signal validity, accelerometer saturation (x3) and gyroscope saturation (x3), temperature and ΔT out of range.
- `GNSS_frozen_flag`: flag that indicates if GNSS measurements are frozen.
- `GNSS_OTR`: flag that indicates if GNSS measurements are out of range.
- `GNSS_Sev`: flag indicating GNSS severity (1 == LSE, 2 == HSE).
- `no_update_flag`: flag indicating that a faulty measurement has been detected by CUSUM test using information on the KF innovation, and indicating that the state and covariance update must not be performed.
- `IMU_frozen_flag`: 6 flags indicating if a sensor's measurement are frozen.

- `IMU_OTR`: 6 flags indicating if a sensor's measurements are out of range.
- `IMU_kde`: 6 flags indicating if Sliding Window KDE has detected outliers in any of IMU's sensors, separated in two signals of three flags each, for accelerometers and gyroscopes.
- `IMU_pc`: flag indicating that Predictor-Corrector has detected an outlier in IMU's measurement.
- `rec_flag`: 6 flags indicating if an increase in process noise covariance must be executed by Navigation.

The following sections will briefly describe the models developed to simulate FDIR internal structure, focusing in particular on GNSS and IMU levels as described in Section 4.3.1.

4.3.2.1 GNSS FDIR levels

As described before, FDIR structure for the GNSS part is divided into two levels, depicted in Figure 4.11.

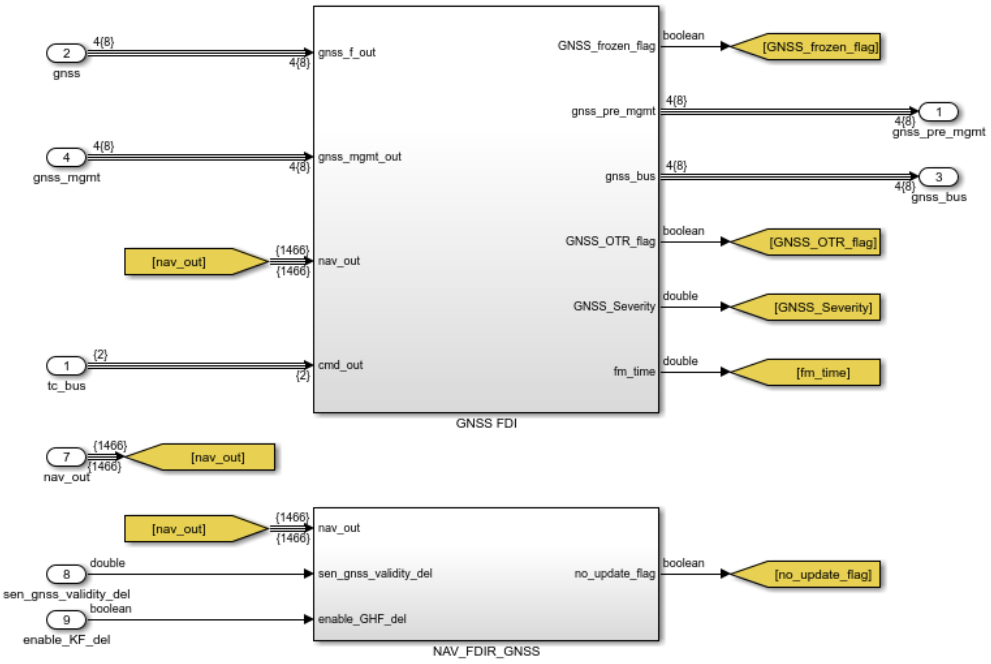


Figure 4.11: GNSS FDIR levels

The first level, GNSS FDI, contains preliminary checks, that is GNSS frozen measurement and Out of Range measurement checks, as depicted in Figure 4.12. It performs frozen measurement check on GNSS output data after fault injection and sends data to management software. Frozen measurement check is performed using a counter to count

the number of equals measurement and a flag is raised when the count is bigger than a user-defined threshold. Then OTR check is performed after the management software, at Navigation frequency, using previous state estimation. First, previous state information is compared to new GNSS measurement to find the values of position and velocity at GNSS acquisition time, taking into account the possible delays introduced in the measurement acquisition, then the difference between these two measurements is compared against an adaptive threshold based on estimated signals' σ, multiplied by a user-defined factor.

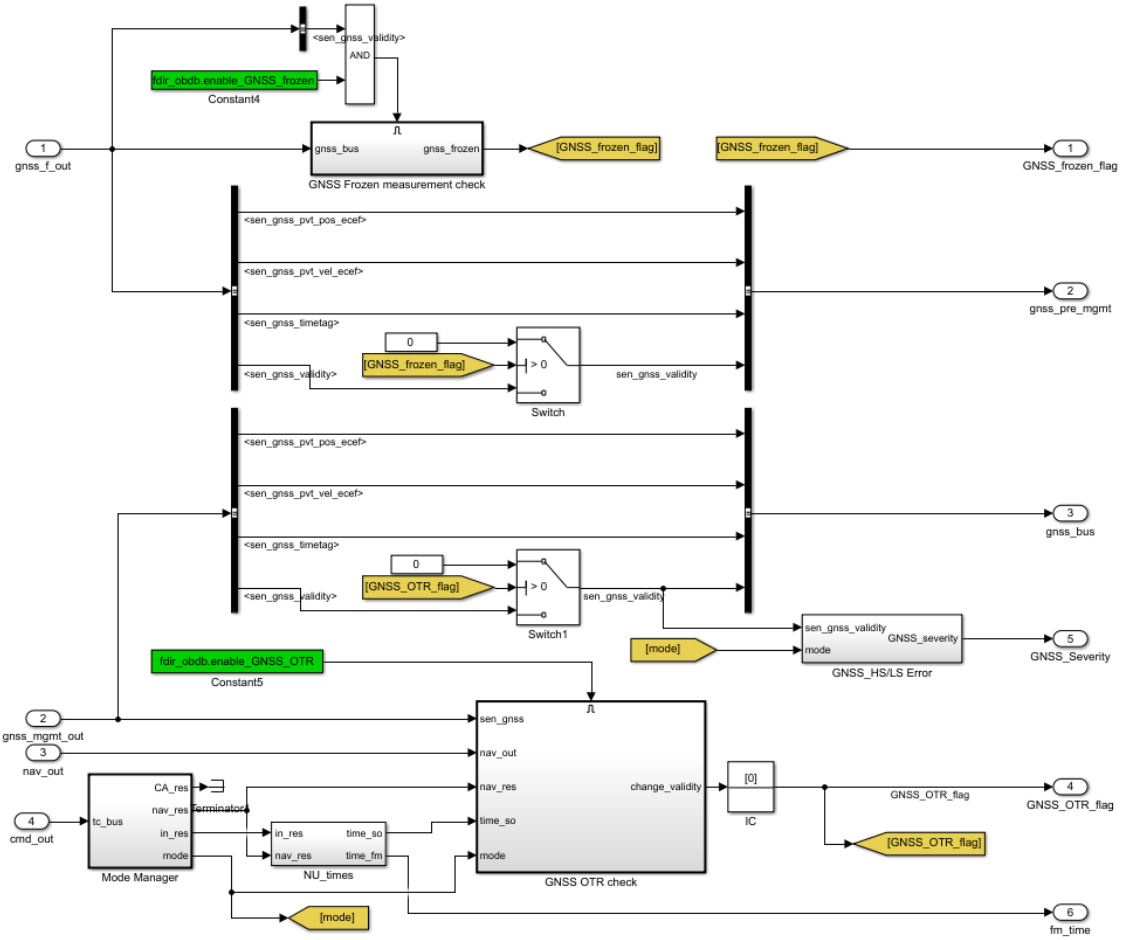


Figure 4.12: GNSS FDIR 1st level model

Command bus is used to generate mode transition flag and navigation unit times (time from start up and time from transition to flight mode). The time from startup is needed in order to obtain the state estimation at the time when a new GNSS measurement is available, allowing to compare them in the OTR check. It is important to notice how, if a GNSS measurement is not valid, none of the checks is executed because the measurement is already discarded automatically by Navigation. The time from transition to flight mode is an output of this level, because it is needed in some IMU outlier detection algorithm to

enable a change of threshold values.

The second level of GNSS FDIR contains the CUSUM test to detect GNSS outliers and monitor filter performances. The model developed is depicted in Figure 4.13. Here some of the flags used in Navigation are passed to the CUSUM test block as feedback, to calculate the test statistics based on filter innovation and detect outliers.

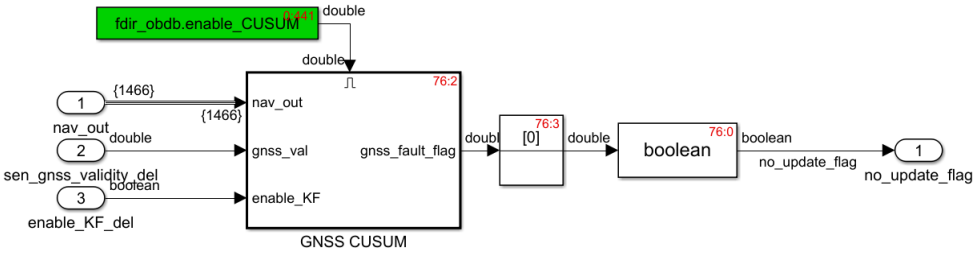


Figure 4.13: GNSS FDIR 2nd level model

4.3.2.2 IMU FDIR levels

FDIR structure for IMU outlier detection, isolation and recovery is divided into two levels, depicted in Figure 4.14. The two levels are more integrated than for GNSS part, so the complexity of this part of the system is bigger, in fact the two levels exchange more information between them, at different levels of navigation unit data flow and at different frequencies.

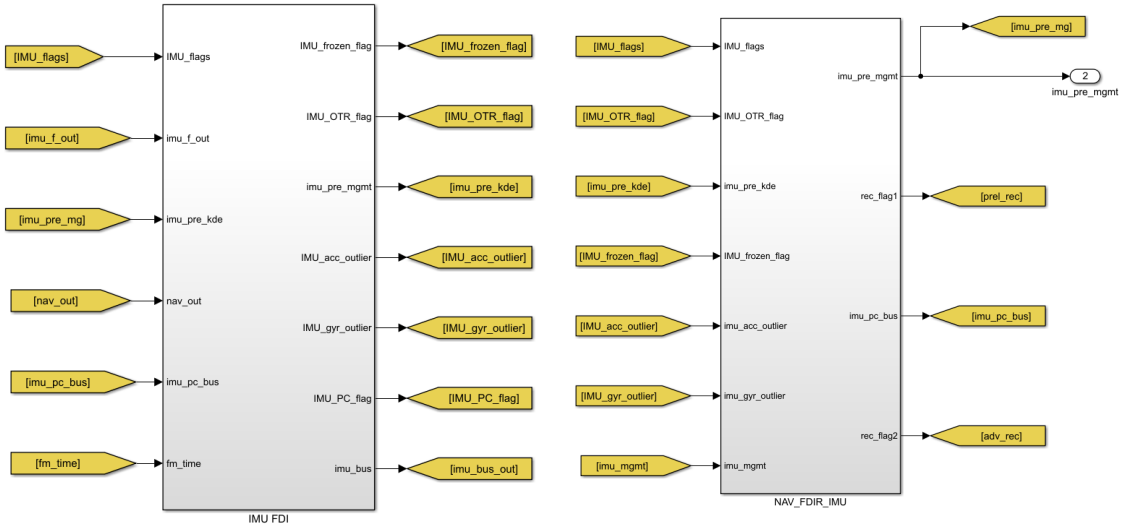


Figure 4.14: IMU FDIR levels.

The first level is divided in Preliminary Checks (Figure 4.15), and Advanced Checks (Fig-

ure 4.16) and both of them follow and implement the logic explained in Section 4.3.1. Preliminary Checks include Frozen measurement check (that is similar to GNSS frozen measurement check) and Out of range check, that consists in comparing IMU's measurement against fixed threshold coming from sensors specifications.

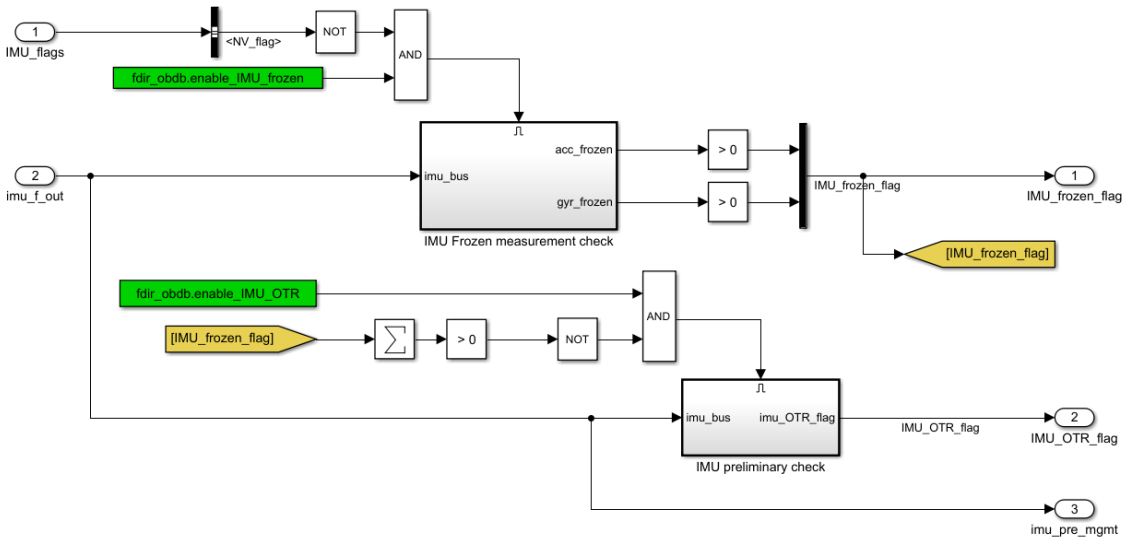


Figure 4.15: IMU FDI Preliminary Checks model

Advanced checks include Sliding Window Kernel Density Estimator and Predictor-Corrector Algorithm: both of them make use of buffers to keep memory of previous data (from IMU's measurement or from navigation solution). For Sliding Window KDE, bins are passed as external parameter to the model, and the Kernel Density Estimator is applied to every component of accelerometers and gyroscopes output. This allows to detect and isolate the faulty measurement. This algorithm can provide outlier detection only at Navigation frequency, because it works using batches of IMU's measurements. For this reason, recovery action can only be implemented after Management software and Predictor-corrector execution, when IMU's signal has been downsampled and converted to Navigation Frequency. Isolation is not possible for Predictor-Corrector algorithm, it can only detect an outlier in the whole measurement, when the difference between prediction and correction is too high.

The second level contains the recovery system, and is divided in **RECOVERY 1** and **RECOVERY 2**, according to the description given in Section 4.3.1. In the first recovery block, that runs at IMU frequency, both Frozen and Prediction Recovery are implemented, while for the second block, that runs at Navigation frequency, only Frozen recovery has been modeled. The model for the second level is shown in Figure 4.17.

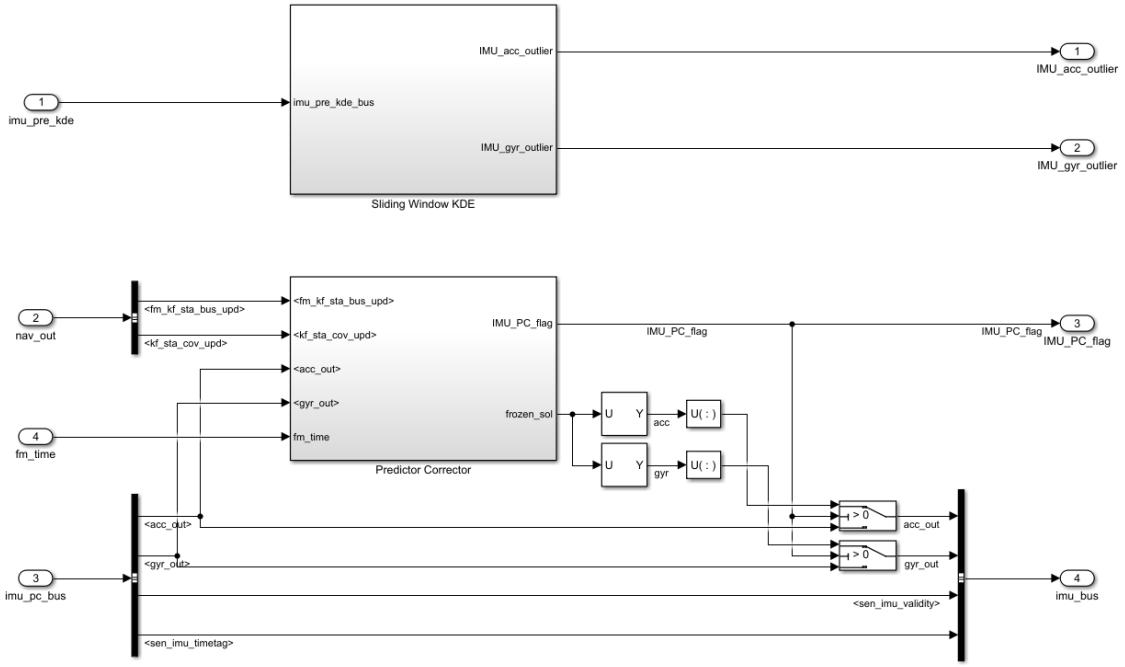


Figure 4.16: IMU FDI Advanced Checks model

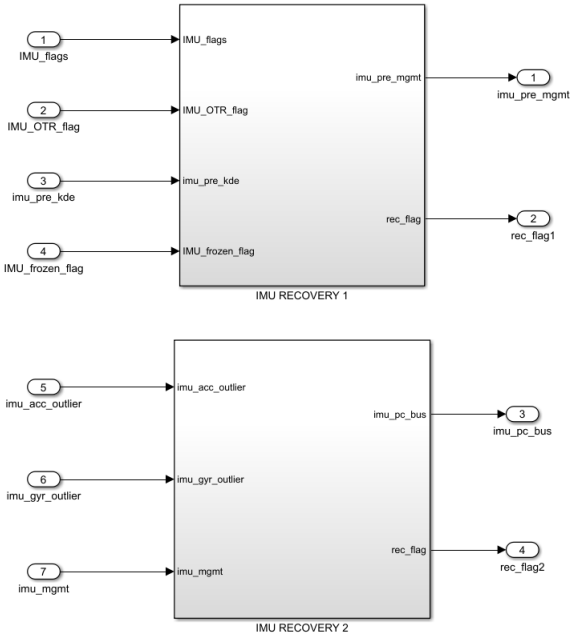


Figure 4.17: IMU FDIR 2nd level model

Beside these two levels, IMU FDIR also contains a flag handling block, whose task is to

decode IMU CBIT and generate useful flags indicating IMU's state. Although in Section 4.3.1 CBIT decodification and flag handling was included in IMU FDI (1st level), this model has been implemented outside 1st level for sake of simplicity, and is represented in Figure 4.18.

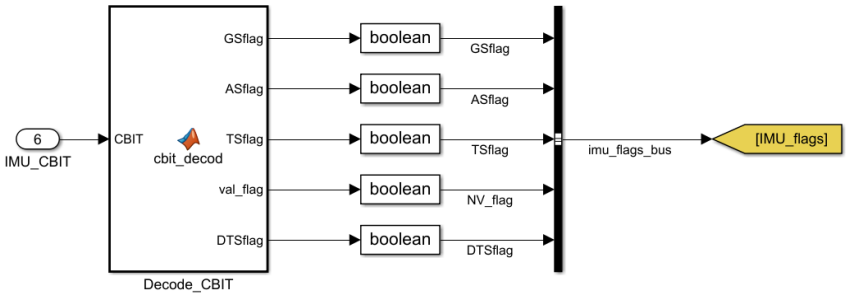


Figure 4.18: IMU's flags handling model

4.3.2.3 FDIR output flags

Finally, all the flags generated by FDIR model and described in Section 4.3.2 are grouped in buses and form part of the system's output signals, as depicted in Figure 4.19.

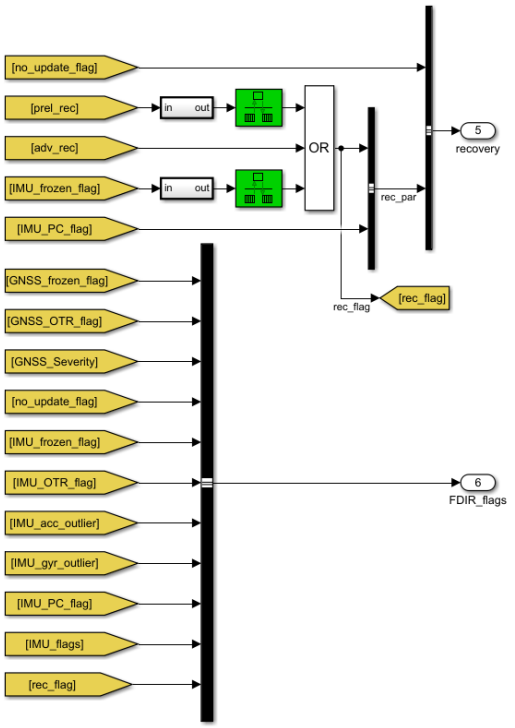


Figure 4.19: FDIR output flags buses

The first bus `recovery`, contains the parameters needed by Navigation block to execute the recovery actions, in particular, the flags that prevent from updating state and covariance when a GNSS outlier is detected and the flags that generate the increase in process noise covariance when an outlier is detected in IMU's measurement.

The bus `FDIR_flags` group all the detection and recovery flags generated by the FDIR model, to provide feedback and control information and detection/isolation performances of the implemented algorithms.

Chapter 5

Validation

The aim of this chapter is to present the results obtained during the verification and validation campaign, where the system has been subjected to a large number of simulations. Section 5.1 provides a brief description of the SENER Aeroespacial hybrid navigation simulator. In order to validate the FDIR functions, a Fault Injection Model has been implemented. This model is described and verified in Section 5.2. Finally, Section 5.3 contains the verification and validation campaign applied to the FDIR design implemented.

5.1 System Concept Simulator Description

The simulator developed by SENER Aeroespacial using Matlab/Simulink software consists of several fundamental blocks:

- **Dynamics, Kinematics and Environment**: this block simulates the real world environment and reproduces launcher trajectory, in particular acceleration, velocity and attitude. This block is made up of a trajectory generator block that outputs non-gravitational acceleration and angular rate to the integration block, where these two signal are processed according to the differential equations that govern launcher motion, in several reference frames. Furthermore, this block generates vibrations (both on launch-pad, caused by wind, and during flight, by mechanical vibrations) and shocks caused by stages' separation or motor burns. The differential equations that govern launcher motion are shown in Equation (5.1), expressed in an inertial reference frame:

$$\begin{aligned} \dot{q}_{B\leftarrow I} &= \tfrac{1}{2} q_{B\leftarrow I} \odot \begin{pmatrix} \omega_B \\ 0 \end{pmatrix} \\ \dot{r}_I &= v_I \\ \dot{v}_I &= \bar{q}_{B\leftarrow I} \odot a_{B,tot} \odot q_{B\leftarrow I} + g_I(r_I) \\ a_{B,tot} &= a_B + a_{shock} + a_{vibr} \end{aligned} \tag{5.1}$$

It is possible to observe that the equations used for the propagation of the trajectory are almost the same as the ones presented in Section 2.1: first, attitude quaternion from Body to Inertial frame is calculated and then it is used to rotate total non gravitational acceleration from Body frame to Inertial frame. Total non-gravitational acceleration in Body frame is obtained by adding Shock and Vibration acceleration profiles to non-gravitational acceleration from trajectory data (the only difference with respect to the equations in Section 2.1). Local gravity acceleration in Inertial frame is obtained from launcher position (using a central + J6 model, with the possibility to reduce the harmonics including from J2 to J6 contribution) and added to obtain the total acceleration in Inertial frame, which is, then, integrated two times to obtain velocity and position. Shock acceleration is obtained using the superposition of damped sinusoids to model the acceleration profile caused by the shock, while wind vibrations on the launch pad are modeled by a 2D harmonic oscillator and mechanical vibration during flight are simulated as white noise. Finally, this block provides also a conversion of position and velocity to orbital elements, to keep track of the latter ones during orbital phase.

- **Sensors**: this block contains both IMU and GNSS models, that include sensors clock and an accurate model that reproduces all the errors described in Chapter 2 for accelerometers and gyroscopes.
  Regarding GNSS model, it is designed to work for different configurations of available satellite, both from GPS and Galileo constellations. It includes several GNSS receiver outages due to stages separations, that makes GNSS unavailable (GNSS validity signal is set to 0). This block represents a Performance model and provides a PVT (position-velocity-time) solution as measured by GNSS receiver, taking into account the position of the antenna and the presence of a position bias.
  IMU model provides a measurement of non-gravitational acceleration (in the form of delta-velocities) and angular rate (in the form of delta-angles) as sensed by IMU (in launcher's Body reference frame) and it also includes a validity signal and a clock signal. Gyroscopes model includes some non-idealities like: Angle Random Walk noise, biases (switch-on bias, in-run bias, bias drif and instability, bias Rate Random Walk, vibration bias), scale factor errors, mounting errors, linear acceleration sensitivity, quantization and deadband. Accelerometer model includes non-idealities like: Velocity Random Walk noise, biases (switch-on bias, in-run bias, bias drift, bias instability, Acceleration Random Walk bias, vibration bias), scale factor errors, monting errors, quantization and deadband and saturation.

- **Management SW**: this block contains the modeling of the necessary software operations in order to obtain IMU and GNSS data at the same frequency of the Navigation block. In fact, IMU typically runs at very high frequencies, while GNSS typically runs at lower frequencies: coning-sculling, frame rotation and downsampling are performed by this block. The resulting outputs are IMU and GNSS signals (including validity signal and clock signal) at the same frequency, that is the Navigation frequency.

- **Navigation**: this block is the core of the system as it presents the whole navigation software and includes the necessary functions to perform hybrid navigation, that is inertial propagation, covariance propagation, filter update computation, state update, leading to the generation of a navigation solution that is the final output of the system. The block also provides interesting parameters that allow to monitor filter performances if compared to sensors raw data.

In this frame, **FDIR** block has been designed to provide navigation unit monitoring through flag handling, outlier detection and error reporting. The block takes as input several signals as described in Chapter 4, and perform detection, isolation and recovery (if possible) of the unit. Also, a Fault Injection model has been designed to introduce faults in the simulator. This model is described in the following Section. A basic scheme of the blocks and the relative data flows of the simulator developed by SENER Aeroespacial is depicted in Figure 5.1.

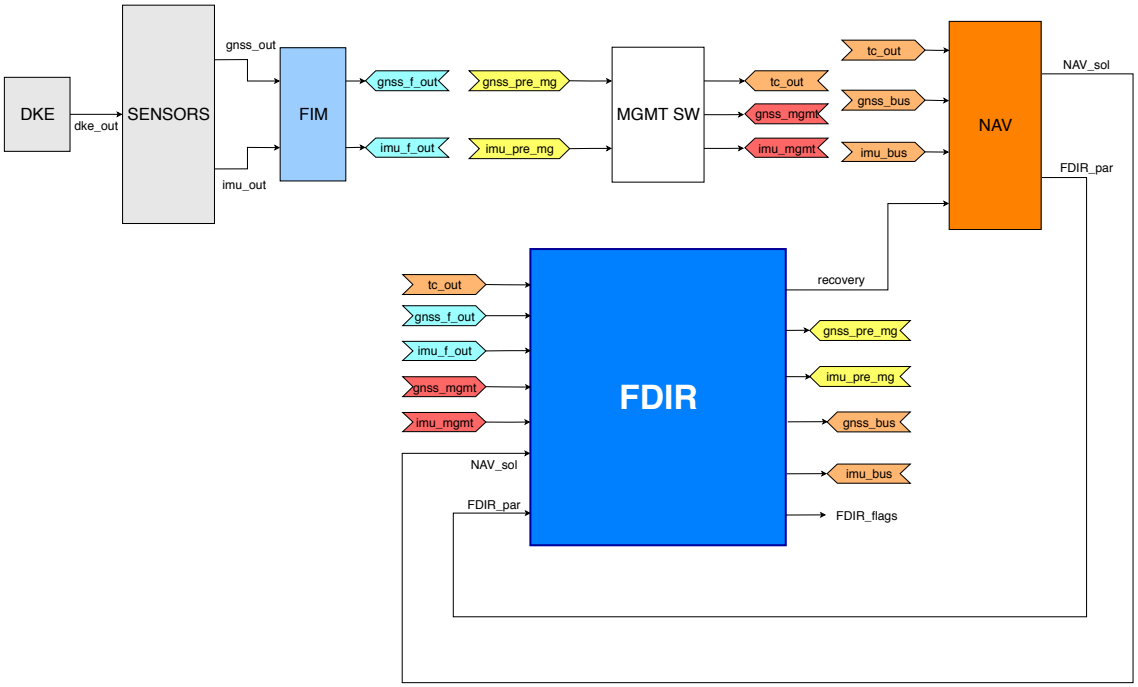


Figure 5.1: Scheme of SENER Aeroespacial simulator

5.2 Fault Injection Model

In order to reproduce in the simulator the faults described in Chapter 3, after the analyses performed, a Fault Injection Model has been designed and tested.

5.2.1 Requirements and Specifications

First, the following specifications and requirements have been identified:

[SIM - FI - 01] - Type of errors
Fault Injector Model shall be able to introduce the following errors:

a. Frozen measurement: changing current measurement with last valid measurement before injection instant.

b. Constant measurement: changing current signal value with user-defined constant.

c. Function of time: changing current signal values with user-defined function of time $(a \cdot t + b)$.

d. User-defined signal: changing current signal with another user-defined signal.

[SIM - FI - 02] - Addition or replacement of faulty value
Fault Injector Model shall be able to introduce errors relative to **[SIM − FI − 01]** requirement by changing current signal or by summing values to current signal.

[SIM - FI - 03] - Delays
Fault Injector Model shall be able to introduce delays on a signal line.

[SIM - FI - 04] - Injection time and duration
Injection instant of time and fault duration shall be configurable by user.

[SIM - FI - 05] - Adaptability to input signal size and frequency
Fault Injection model shall be able to adapt to input signal frequency and size.

[SIM - FI - 06] - Signals
Fault Injection model shall be applied to all input/output signal.

[SIM - FI - 07] - Signals components
Fault Injection model shall be able to introduce different faults in every signal component or in only one component.

[SIM - FI - 08] - Ability to introduce several errors in the same signal
Fault Injection model shall be able to introduce several type of errors in different instants of time and with different duration in the same signal throughout the whole simulation time.

The previous requirements have been defined to fulfill fault introduction specifications. The faults described in Chapter 3 have been investigated to identify a way of introducing them in the simulator. Table 5.1 shows how these faults can be reproduced in the simulator and helps keeping track of the requirements during Fault Injection Model design.

Table 5.1: Simulation of faults through Fault Injection Model

Fault	Simulation	Requirement
GNSS measurement spike	Introducing a user-defined constant value, function of time or position-velocity profile (added to the real signal or instead of it).	**[SIM - FI - 01]** **[SIM - FI - 02]**
GNSS frozen measurement (fixed or transient)	Introducing the last value before injection time instead of the real signal, for a limited amount of time or for the whole simulation.	**[SIM - FI - 01]** **[SIM - FI - 04]**
GNSS invalid measurement undetected by GNSS (fixed or transient)	Fixing GNSS validity signal to 1 when the measurement is not valid.	**[SIM - FI - 01]** **[SIM - FI - 04]**
GNSS measurement with unexpected delay	Introducing a delay after GNSS reading.	**[SIM - FI - 03]** **[SIM - FI - 04]**
Total loss of GNSS	Fixing GNSS validity to 0 for a period of time lower or bigger than 5 s (TBD).	**[SIM - FI - 01]** **[SIM - FI - 04]**
GYR/ACC measurement spike in one or more axes	Introducing a user-defined constant value, function of time or delta-angle/delta-velocity profile (added to the real signal or instead of it).	**[SIM - FI - 01]** **[SIM - FI - 02]** **[SIM - FI - 07]**
GYR/ACC frozen measurement in one or more axes (fixed or transient)	Introducing the last value before injection time instead of the real signal for a limited amount of time or for the whole simulation time.	**[SIM - FI - 01]** **[SIM - FI - 04]** **[SIM - FI - 07]**
GYR/ACC invalid measurement undetected by IMU in one or more axes (fixed or transient)	Fixing IMU validity signal to 1 when the measurement is not valid.	**[SIM - FI - 01]** **[SIM - FI - 04]** **[SIM - FI - 07]**
Total loss of IMU output (fixed or transient)	Fixing IMU validity signal to 0 for a limited amount of time or for the whole simulation time.	**[SIM - FI - 01]** **[SIM - FI - 04]**

5.2.2 Fault Injection Model design description

This section describes the Fault Injection Model developed to introduce the selected faults in the simulator, in compliance with the requirements listed above, in previous Section.

The model is able to introduce several types of faults in a specific signal, whose size is variable. The types of error that the model can introduce are:

1. Constant value fault
2. Function of time fault
3. User-defined profile fault
4. Frozen signal fault
5. Delay fault

These types of errors can be introduced in one or in all of signal's components, and, in the latter case, the same value can be introduced for each component, or several different values can be introduced in every component.

The model can introduce multiple errors in the same signal throughout the whole simulation time, it can also reproduce the same type of error several times, with different values.

The model can introduce errors by summing the fault to the current value of the signal or by replacing the real signal with the new user-defined value. The user can, also, configure injection time and duration of the faults. Finally, the model can handle signals with different frequency and size.

Model interfaces, i.e. inputs and outputs signals, are listed below:

- **INPUTS**: input signal in which faults must be introduced and clock signal;
- **OUTPUTS**: output signal in which faults have been introduced as specified by user.

The parameters that needs to be introduced in the model by user in order to configure fault injection are presented in Table 5.2.

Table 5.2: FIM Parameters (F == number of faults, C == number of components)

Parameters	Size	Units	Description
`fault_mode`	[1,F]	N/A	Array containing the faults to be inserted into the signal, sorted by instant of time: 0. no faults 1. constant fault 2. f(t) fault ($a \cdot t + b$) 3. profile fault 4. frozen fault 5. delay fault
`fault_start`	[1,F]	[s]	Array containing the injection instants of the faults, sorted by instant of time.
`fault_duration`	[1,F]	[s]	Array containing the duration of the faults to be injected, sorted by instant of time.
`fault_comp`	[1,F]	N/A	Array containing the components of the signal in which the faults are injected. Input 0 to inject the fault in all components.
`fault_sum`	[1,F]	N/A	Array containing the flags that enable sum of the fault value with current signal value, if it is 1. It concerns only faults 1, 2 and 3 but this parameter must have the same size as fault_mode.
`const_fault`	[C,F]	N/A	Matrix which columns represents the constant values to be injected in the signal. Even if there is only one constant fault in the signal, this parameter must be specified as a 2D matrix, the second column will be ignored. If the signal has only one component, this parameter can be introduced as an array of dimensions [1,F]
`funct_fault`	[C,2,F]	N/A	3D Matrix. For every fault, the rows of the submatrix contain the parameters a and b to be applied in every component to calculate the function. Even if there is only one fault of this type, this parameter must be defined as a 3D matrix, the second submatrix will be discarded.
`delay_fault`	[F,1]	N/A	Array containing the number of samples of delay to be introduced in the whole signal or in one component only.
`prof_fault`	TS	N/A	Name of the signal to be loaded from workspace, defined as a timeseries (TS) containing time and a column of data for each component.
`prof_tsamp`	[1,1]	[s]	Sample time of user-defined profile.

The model is composed by three main blocks that are presented below:

- **Fault Selection Block**: This block select the fault to be introduced in the signal according to input parameters as fault type, fault starting time, fault duration, fault sum flag and component in which the fault is applied. The block does not handle the signal, it just takes a clock input, to compare it with input parameters and raise flags when the system must introduce a fault. The outputs of this block are represented by a fault mode flag, that indicates the type of fault when there is a fault to be introduced and equals 0 otherwise; a flag that indicates whether to sum the fault to current signal value or to replace the latter with the new one; and the component in which the signal is injected. The clock is compared with fault starting time and duration to detect when a fault must be introduced and to generate an index that changes fault type, duration, component and sum flag.
- **Fault Injection Block**: this block is where the fault is injected in the signal. Fault type flag is used as condition of a switch-case block that select and inject the fault in input signal. Inputs of the block are the current input signal, clock signal, fault type flag and sum flag. The only output is the output signal after the introduction of the faults. It has to be noticed that in this block, fault is injected in all the components. The selection of the right component happens in Concatenation block. A series of counters has been implemented to detect if there are several errors of the same type to be injected in the same signal. Sum flag is used to indicate whether to replace current signal value with new value or to add the latter to the current signal value.
- **Concatenation Block**: this block consists in a Matlab function that takes as inputs the original input signal, the modified signal from Fault Injection Block, the component in which the fault must be introduced and the size of the signal. The function correctly select the component in which the fault must be introduced as configured by user.

The model is depicted in Figure 5.2.

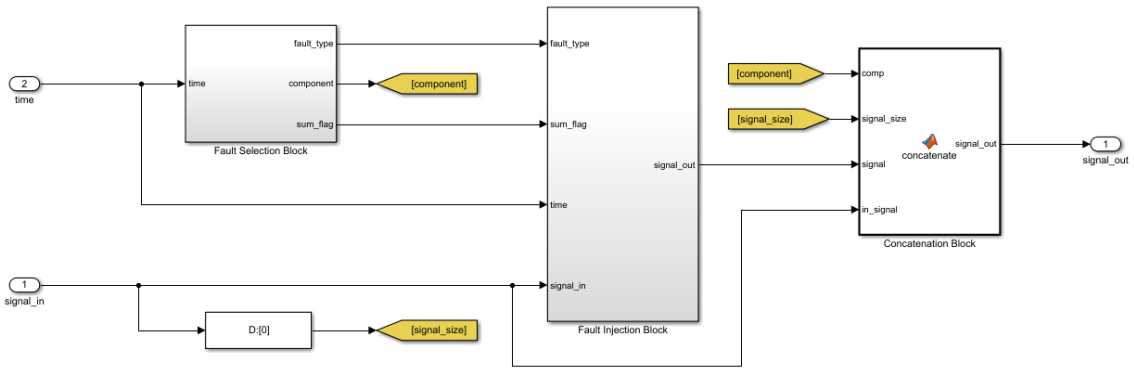


Figure 5.2: Fault Injection Model

5.2.3 Fault Injection Model verification

Before integration in the simulator, the Fault Injection model has been subjected to a verification campaign, to verify the compliance with all the requirements listed above. To perform model verification, several tests have been executed, using linear signals with different slopes, during a simulation of 5000 s.

Table 5.3 summarizes the test that have been performed for the Fault Injection Model. The following sub-sections contain the detailed description and results for each test.

Table 5.3: Fault Injection Model verification tests

ID	TEST	RESULT
FI-01	Constant Fault Injection	Test Passed
FI-02	Function Fault Injection	Test Passed
FI-03	Profile Fault Injection	Test Passed
FI-04	Frozen Fault Injection	Test Passed
FI-05	Delay Fault Injection	Test Passed

Table 5.4 shows the compliance of the designed model with requirements and how it has been verified.

Table 5.4: Fault Injection Model verification results summary

Requirement	Compliance	Verification (TEST)
[SIM - FI - 01]	YES	Frozen fault injection (FI-04) Constant fault injection (FI-01) Function fault injection (FI-02) Profile fault injection (FI-03)
[SIM - FI - 02]	YES	Sum of fault or replacement of current value (FI-01, FI-02, FI-03)
[SIM - FI - 03]	YES	Delay injection (FI-05)
[SIM - FI - 04]	YES	Configurable injection time and duration (FI-01, FI-02, FI-03, FI-04, FI-05)
[SIM - FI - 05]	YES	Verification on signals with different size and frequency (FI-04, FI-05)
[SIM - FI - 06]	YES	Verification through model inspection
[SIM - FI - 07]	YES	Faults injected in different components (FI-01, FI-02,FI-03)
[SIM - FI - 08]	YES	Multiple faults injected on the same signal/component (FI-01, FI-02, FI-04, FI-05)

5.2.3.1 TEST FI-01 - Constant Fault Injection

Test Description:

The purposes of this test have been:

- to demonstrate compliance with requirement [**SIM - FI - 01**] **b)** by introducing constant faults;
- to demonstrate compliance with requirement [**SIM - FI - 02**] by summing the second constant fault to the current signal value;
- to demonstrate compliance with requirement [**SIM - FI - 04**] by introducing the faults in user-defined time instant and with user-defined duration;
- to demonstrate compliance with requirement [**SIM - FI - 07**] by introducing different constant values in each component for the first fault, the same values in all components for the second fault and by modifying just the third component in the third fault.
- to demonstrate compliance with requirement [**SIM - FI - 08**] by introducing three constant faults;

Test Configuration:

The test have been configured with following parameters:

- **Input signal**: signal with three linear components with a slope of 1, -1 and 0.5 respectively.
- **Faults**: three constant faults with the following parameters:

$$\text{const\_fault} = \begin{matrix} 1000 & 4000 & 0 \\ 2000 & 4000 & 0 \\ 3000 & 4000 & 1000 \end{matrix}$$

- **Faults injection time**: 500 s, 1500 s and 2500 s respectively.
- **Faults duration**: 100 s, 250 s and 500 s respectively.
- **Faults component**: the first two faults in all components, the third fault in the third component only.
- **Faults sum_flag**: the fault is added to current signal value only for the second fault.

Pass/Fail Criteria:

In order to pass the tests, some criteria have been defined. They are shown in Table 5.5.

Test Results:

Table 5.5: TEST FI-01 - pass/fail criteria

	Fault 1		**Fault 2**		**Fault 3**	
TIME [s]	500	600	1500	1750	2500	3000
Component 1	1000	1000	5500	5750	2500	3000
Component 2	2000	2000	2500	2250	-2500	-3000
Component 3	3000	3000	4750	4825	1000	1000

The compliance with the previous criteria can be seen in Figure 5.3, where some values of the signals in the crucial points are highlighted. It is important to notice that in the third fault, the first two components of the signal do not show any change and follow as linear signals with slope 1 and -1 respectively, as the fault has been introduced in the third component only.

The test has been passed successfully.

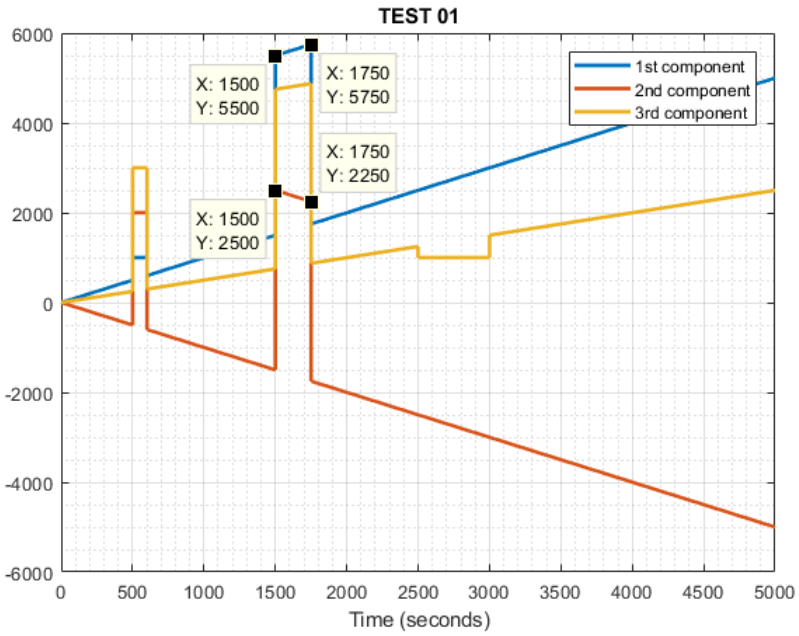


Figure 5.3: Test FI-01: results

5.2.3.2 TEST FI-02 - Function Fault Injection

Test Description:

The purposes of this test have been:

- to demonstrate compliance with requirement [**SIM - FI - 01**] **c) b**y introducing function of time faults;
- to demonstrate compliance with requirement [**SIM - FI - 02**] by summing the second fault to the current signal value;
- to demonstrate compliance with requirement [**SIM - FI - 04**] by introducing the faults in user-defined time instant and with user-defined duration;
- to demonstrate compliance with requirement [**SIM - FI - 07**] by introducing different function values in each component for the first fault and by modifying just the second component in the second fault.
- to demonstrate compliance with requirement [**SIM - FI - 08**] by introducing two function faults;

Test Configuration:

The test have been configured with following parameters:

- **Input signal**: signal with three linear components with a slope of 1, -1 and 0.5 respectively.
- **Faults**: two function faults with the following parameters:

$$\text{funct\_fault}(:,:,1) = \begin{matrix} 100 & 0.1 \\ 200 & 0.2 \\ 300 & 0.3 \end{matrix} \qquad \text{funct\_fault}(:,:,2) = \begin{matrix} 0 & 0 \\ 500 & 0.1 \\ 0 & 0 \end{matrix}$$

- **Faults injection time**: 1000 s and 2000 s respectively.
- **Faults duration**: 100 s and 200 s respectively.
- **Faults component**: the first fault in all components, the second fault in the second component only.
- **Faults sum_flag**: the fault is added to current signal value in the second fault.

Pass/Fail Criteria:

In order to pass the tests, some criteria have been defined. They are listed in Table 5.6:

Table 5.6: TEST FI-02 - pass/fail criteria

	Fault 1		**Fault 2**	
TIME [s]	1000	1100	2000	2200
Component 1	200	210	2000	2200
Component 2	400	420	-1300	-1480
Component 3	600	630	1000	1100

Test Results:

The compliance with the previous criteria can be seen in Figure 5.4, where some values of the signals in the crucial points are highlighted.

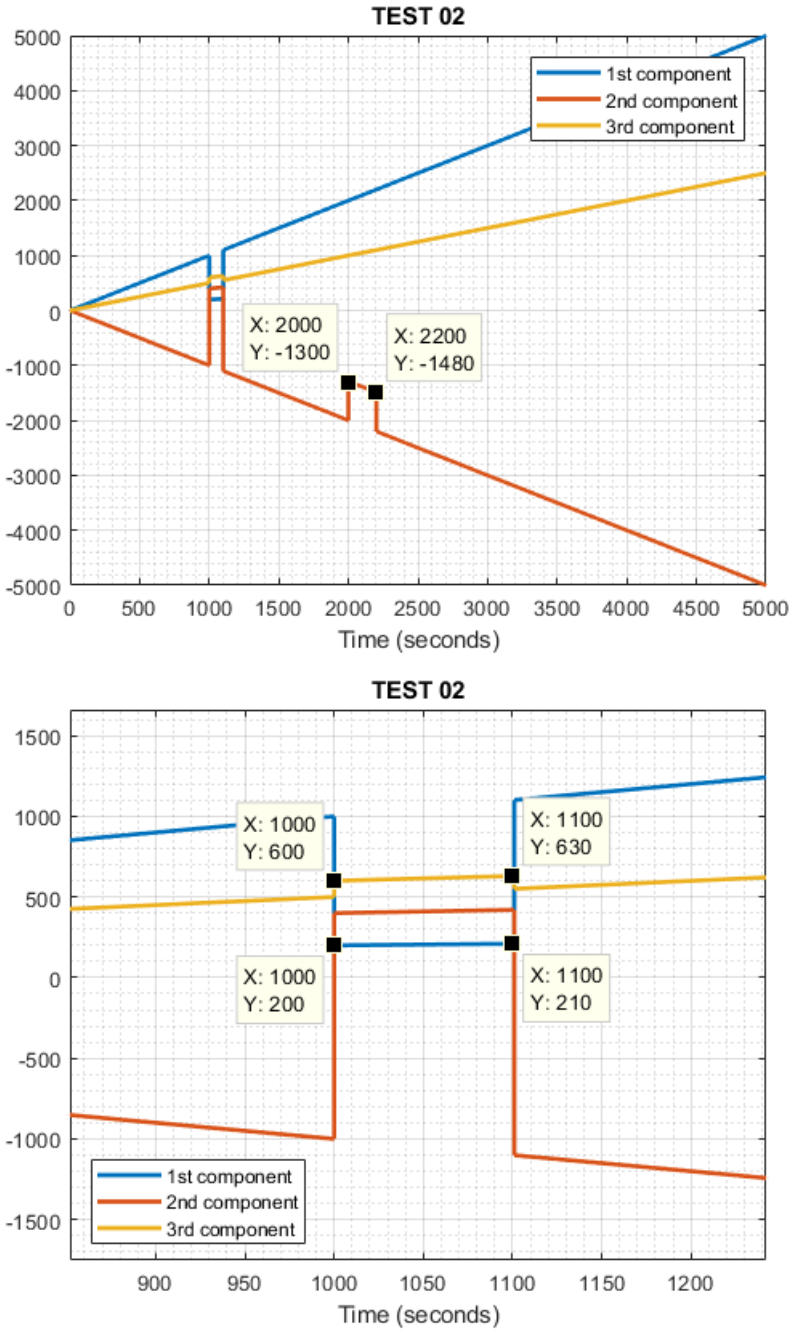


Figure 5.4: Test FI-02: results

It is important to notice that in the second fault, the first and the third component of

the signal do not show any change and follow as linear signals with slope of 1 and 0.5 respectively, as the fault is introduced in the second component only.

The test has been passed successfully.

5.2.3.3 TEST FI-03 - Profile Fault Injection

Test Description:

The purposes of this test have been:

- to demonstrate compliance with requirement **[SIM - FI - 01] d)** by introducing profile faults;
- to demonstrate compliance with requirement **[SIM - FI - 02]** by summing the fault to the current signal value;
- to demonstrate compliance with requirement **[SIM - FI - 04]** by introducing the faults in user-defined time instant and with user-defined duration;
- to demonstrate compliance with requirement **[SIM - FI - 07]** by introducing by modifying just the second component in the second fault.

Test Configuration:

The test have been configured with following parameters :

- **Input signal**: signal with three linear components with a slope of 1, -1 and 0.5 respectively.
- **Faults**: one profile fault consisting in a linear signal with slope 1.
- **Faults injection time**: 2000 s..
- **Faults duration**: 100 s.
- **Faults component**: the fault is introduced in the second component only.
- **Faults sum_flag**: the fault is added to current signal value.

Pass/Fail Criteria:

In order to pass the tests, some criteria have been defined. They are listed in Table 5.7:

Table 5.7: TEST FI-03 - pass/fail criteria

	Fault 1	
TIME [s]	2000	2100
Component 1	2000	2100
Component 2	0	0
Component 3	1000	1050

Test Results:

The compliance with the previous criteria can be seen in Figure 5.5 where some values of the signals in the crucial points are highlighted. It is important to notice that first and third component of the signal do not show any change and follow as linear signals with slope of 1 and 0.5 respectively, as the fault is introduced in the second component only.

The test has been passed successfully.

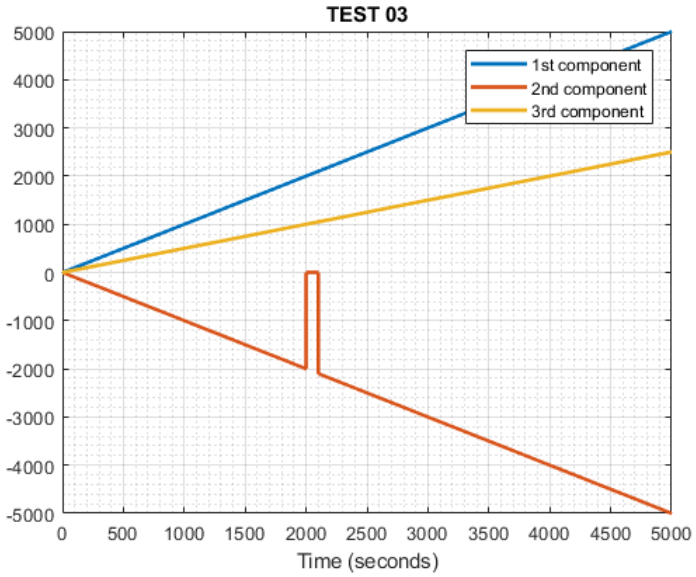


Figure 5.5: Test FI-03: results

5.2.3.4 TEST FI-04 - Frozen Fault Injection

Test Description:

The purposes of this test have been:

- to demonstrate compliance with requirement [**SIM - FI - 01**] **a)** by introducing frozen faults;
- to demonstrate compliance with requirement [**SIM - FI - 04**] by introducing the faults in user-defined time instant and with user-defined duration;
- to demonstrate compliance with requirement [**SIM - FI - 05**] by introducing the faults in a signal with different size from the previous one;
- to demonstrate compliance with requirement [**SIM - FI - 08**] by introducing three frozen faults.

Test Configuration:

The test have been configured with following parameters:

- **Input signal**: signal with one linear component with a slope of 1.
- **Faults**: three frozen faults.
- **Faults injection time**: 1000 s, 2000 s and 3000 s respectively.
- **Faults duration**: 100 s, 200 s and 300 s respectively.
- **Faults component**: the faults are introduced in all components.
- **Faults sum_flag**: N/A.

Pass/Fail Criteria:

In order to pass the tests, some criteria have been defined. They are shown in Table 5.8.

Table 5.8: TEST FI-04 - pass/fail criteria

	Fault 1		**Fault 2**		**Fault 3**	
TIME [s]	1000	1100	2000	2200	3000	3300
Component 1	1000	1000	2000	2000	3000	3000

Test Results:

The compliance with the previous criteria can be seen in Figure 5.6, where some values of the signals in the crucial points are highlighted.

The test has been passed successfully.

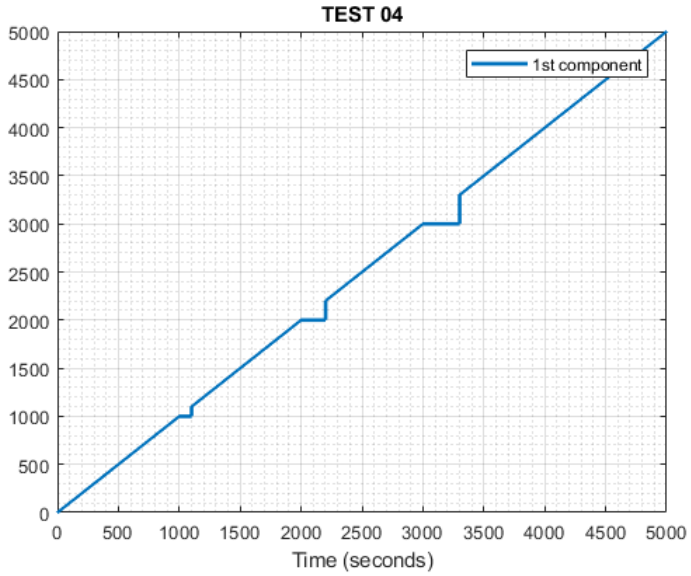


Figure 5.6: Test FI-04: results

5.2.3.5 TEST FI-05 - Delay Fault Injection

Test Description:

The purposes of this test have been:

- to demonstrate compliance with requirement [**SIM - FI − 03**] by introducing delays;
- to demonstrate compliance with requirement [**SIM - FI - 04**] by introducing the faults in user-defined time instant and with user-defined duration;
- to demonstrate compliance with requirement [**SIM - FI - 05**] by introducing the faults in a signal with different size from the previous one.
- to demonstrate compliance with requirement [**SIM - FI - 04**] by introducing the faults in user-defined time instant and with user-defined duration;

Test Configuration:

The test have been configured with following parameters:

- **Input signal**: signal with two linear components with a slope of 1 and -1 respectively.
- **Faults**: two delay faults with 100 and 50 samples of delay respectively.
- **Faults injection time**: 2000 s and 4000 s respectively.
- **Faults duration**: 200 s and 100 s respectively.
- **Faults component**: the faults are applied in all components.
- **Faults sum_flag**: N/A.

Pass/Fail Criteria:

In order to pass the tests, some criteria have been defined. They are listed in Table 5.9:

Table 5.9: TEST FI-05 - pass/fail criteria

	Fault 1		**Fault 2**	
TIME [s]	2000	2200	4000	4100
Component 1	1900	2100	3950	4050
Component 2	-1900	-2100	-3950	-4050

Test Results:

The compliance with the previous criteria can be seen in Figure 5.7, where some values of the signals in the crucial points are highlighted.

The test has been passed successfully.

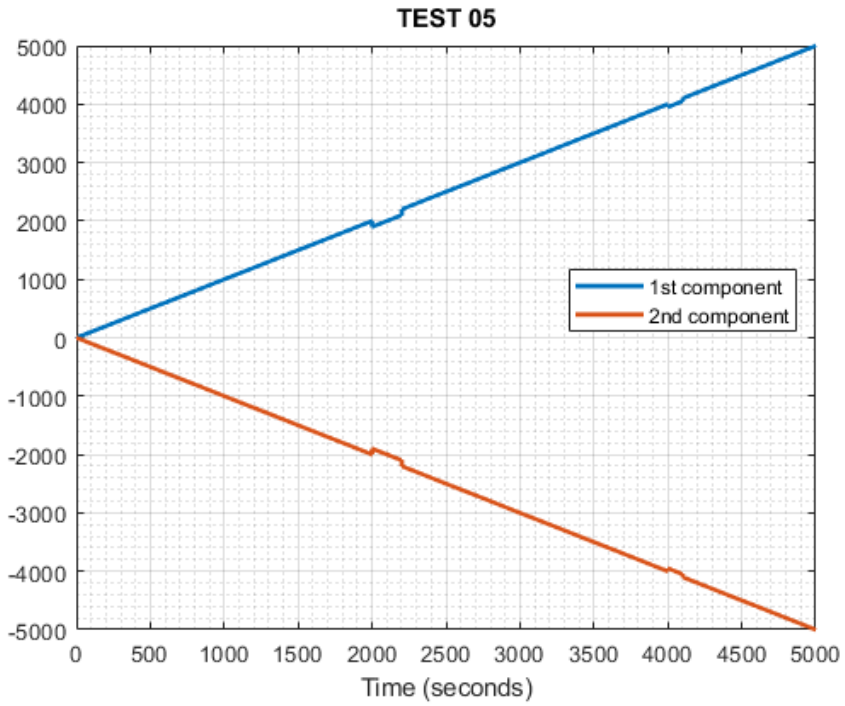


Figure 5.7: Test FI-05: results

5.3 Verification & Validation Campaign

FDIR V&V (Verification and Validation) is a very complicated and time consuming task, especially for software validation, due to the critical importance this subsystem has. The testing procedures represent an interesting challenge because it is not feasible, and sometimes neither possible, to reproduce all the possible faults and their combination as explained in Chapter 3. Furthermore, testing and validation campaigns are influenced by other factors such as partial observability of the system and the particular nature of the processes to be observed.

Verification refers to testing a subsystem, to verify that its response to defined input values and commands is the expected one. In other words, it answer to the question: *"Is the system built in the right way?"*. For this particular case, the Verification part is executed to confirm that the implemented algorithms work properly. This is not always possible, since some of the advanced algorithms are not easy to verify, due to the complexity of the algorithm that has been specifically designed for this application. In this context, Verification is performed over simple algorithms like preliminary checks, in particular the ones for the detection of IMU and GNSS frozen measurements and for IMU Out of Range measurements.

On the other side, Validation refers to the evaluation of the final product/system to verify

the compliance with some validation requirements and that the system meets the needs of the user. In other words, it answers to the question: *"Was the right system built?"*. For this case, validation refers to extensive testing campaign for the FDIR system, including all the algorithms implemented and the general functioning of the system.

5.3.1 Interpreting FDIR results

Regarding Validation campaign results, it is important to describe the method used to obtain and represent such results. Validation campaign has been performed through several Montecarlo simulations. Montecarlo analysis represents a statistical analysis based on random sampling used to study the response of a model to randomly generated inputs. The simulation aim is that the magnitudes of interest statistically converge to their true value, if the developed model does not contain errors.

5.3.1.1 Simulations configuration

The simulations performed during Validation campaign have been executed introducing faults (with variable intensity, duration and injection time) on a nominal trajectory during a simulation time of 5000 s. The nominal trajectory includes 900 s of initial alignment phase, where the inertial sensors are calibrated after the navigation unit is initialized and before entering flight mode. After the beginning of flight mode, the launcher will experience harsh environmental conditions, because of mechanical vibrations due to atmospheric flight that have been taken into account in nominal conditions. Shock accelerations due to stages' separation and burns are not included in nominal trajectory. This phase is usually called *missile phase*. Finally, the launcher enters *orbital phase*, and payload deployment begins at programmed time. The fault have been introduced after sensors' readings, before the data enter Management Software. In every MC shot, the same navigation and FDIR parameters have been maintained, varying fault injection time, duration and intensity and also varying sensors' errors.

5.3.1.2 Representation method

The following steps have been executed to obtain the results presented in next sections:

1. Several Montecarlo simulations have been executed, introducing different types of faults, with different magnitude, duration and injection time.
2. For each shot of each Montecarlo simulation, the estimated values of position, velocity and attitude are compared against a reference simulation, performed under nominal conditions and without errors. In particular, position, velocity and attitude estimated by the navigation system are compared against the respective 3-σ value obtained from the nominal simulation.

3. When the estimated value is bigger than 3-σ nominal value, the difference between the two curves is calculated and integrated over simulation time to obtain a measure of the distance between the performances of the system under failures hypothesis and under the action of FDIR system with respect to nominal performances.

4. A quadratic sum is used to compute the effect of failures and FDIR actions on the three components of position velocity and attitude.

5. Finally, the result of this operation is represented, for each shot of each Montecarlo simulation, in a scatter plot with Fault Injection time on the X axis and Fault duration on the Y axis, to accurately reproduce the difference between the current and the nominal performances for each data point.

This procedure allows to evaluate how the fault introduced and the respective corrective action influence the performances of the system. Four levels of performances are defined:

- **Not Relevant**: identified by the green circle, it indicates that the performances obtained in the MC shot are perfectly inside the 3-σ boundaries defined by nominal trajectory.

- **Acceptable**: identified by the orange square, it indicates that the performances obtained in the MC shot are slightly outside nominal 3-σ boundaries, but the integrated error is lower than the area bounded by nominal 5-σ and 3-σ. As the integrated error is calculated with a quadratic sum between the three components of position, velocity or attitude, simulation performances can fall into this category even if only one component is slightly over the limit.

- **Not Acceptable**: identified by the red diamond, it indicates that the intensity, duration and injection time of faults introduced in the models have an important, non-negligible effect on the performances of the navigation unit. This happens when the integrated error is lower than bounded by nominal 20-σ and 5-σ. This information, compared with FDIR algorithm detection performances, can give a hint about the performances and the limitations of FDIR model.

- **Catastrophic**: identified by the blue star, it indicates that in the corresponding MC shot the hybrid navigation unit has much worse performances with respect to nominal trajectory, in particular when the integrated error is bigger than the area bounded by nominal 20-σ. Typically it happens when FDIR system is not functioning properly or a very long fault is introduced.

A schematic representation of the procedure implemented to interpret FDIR simulations results is shown in Figure 5.8, where the steps executed for a random signal component are presented. The data profiles shown do not represent the real data in any way and are used only for sake of simplicity, to allow a better understanding of the method used.

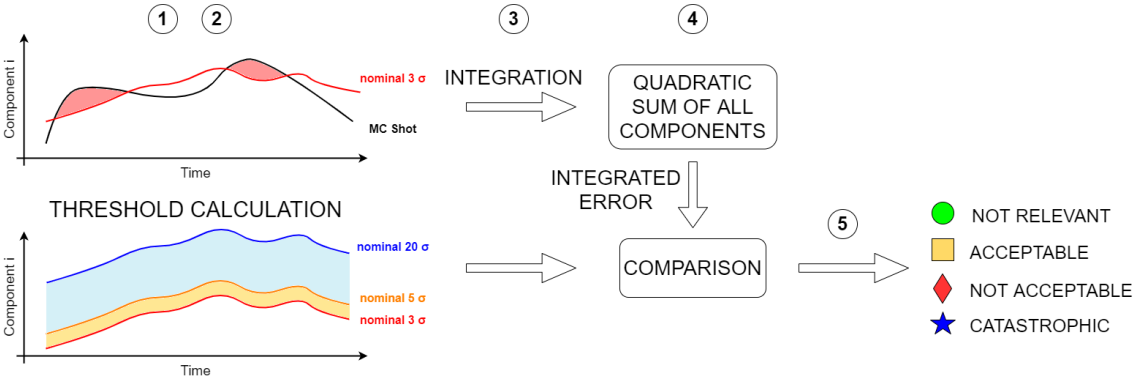


Figure 5.8: Procedure used to interpret FDIR results

5.3.2 FDIR Verification

Verification campaign for FDIR system is executed over a small group of subsystems/algorithms that are very easy to verify. In particular, as explained before, this procedure is applied to IMU and GNSS frozen measurement checks and to IMU Out of Range measurement check. It is important to notice that the input signals used for the verification campaign are generic signals, not related in any way with the real GNSS and IMU signals. The list of simulations that have been executed and their results are listed in Table 5.10 and presented shortly in the following sections.

Table 5.10: FDIR Verification tests

ID	TEST	RESULT
FD-01	GNSS frozen measurement check	Test Passed
FD-02	IMU frozen measurement check	Test Passed
FD-03	IMU OTR measurement check	Test Passed

5.3.2.1 TEST FD-01 - GNSS Frozen Measurement Check Verification

Test Description:

The purpose of this test is to verify the proper functioning of GNSS frozen measurement detection algorithm inserted in the GNSS part of FDIR system, as represented in Figure 4.12. This algorithm detects when a user-defined number of measurement are equals between them and raises a flag.

Test Configuration:

To perform the verification of this block, sinusoidal signals have been used in all three components of position and velocity signals, with different frequencies and phases. The

simulation time has been set to 5000 s. Furthermore, the following faults have been introduced:

- Position: a frozen measurement of 19 s has been injected after 1901 s of simulation, in the second component.
- Velocity: a frozen measurement of 9 s has been injected after 2301 s of simulation, in the first component.

Pass/Fail Criteria:

The algorithm does not perform identification, because when an outlier is detected in GNSS measurements, the whole measurement is discarded. For this reason, the test is considered passed if the algorithm detects the two frozen faults independently from their injection in position or velocity signals, i.e. the only pass/fail criteria is represented by the successful raising of the flag between 1901 and 1920 s and between 2301 and 2310 s.

Test Results:

The results of the test are shown in Figure 5.9, where some important values are highlighted.

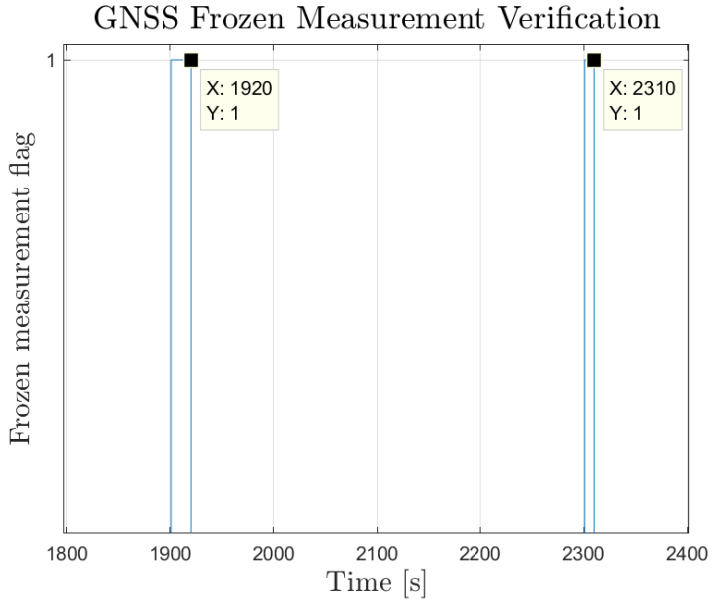


Figure 5.9: Test FD-01: results

It can be observed that the results of the test are compliant to pass/fail criteria, as the algorithm successfully detected the two injected faults.

The test has been passed successfully.

5.3.2.2 TEST FD-02 - IMU Frozen Measurement Check Verification

Test Description:

The purpose of this test is to verify the proper functioning of IMU frozen measurement detection algorithm inserted in the IMU part of FDIR system included in the Preliminary Checks, as represented in Figure 4.15. This algorithm detects when a user-defined number of measurement are equals between them and raises a flag.

Test Configuration:

To perform the verification of this block, sinusoidal signals have been used in all three components of accelerometers and gyroscopes signals, with different frequencies and phases. The simulation time has been set to 5000 s. Furthermore, the following faults have been introduced:

- Accelerometers: a frozen measurement of 49 s has been injected after 1501 s of simulation in the second component and after 1701 s of simulation in the third component.
- Gyroscopes: a frozen measurement of 29 s has been injected after 2001 s of simulation in the first component and after 2201 s in the third component.

Pass/Fail Criteria:

The algorithm does perform identification, because it is important to know where an outlier is detected to undertake the proper recovery action. For this reason, the test is considered passed if the results of the rest respects the following pass/fail criteria, listed in Table 5.11, where the components of the signal represents the expected outcome in terms of flags raised by the system.

Table 5.11: TEST FD-02 - pass/fail criteria

	ACC		**GYR**	
TIME [s]	1501-1550	1701-1750	2001-2030	2201-2230
Component 1	0	0	1	0
Component 2	1	0	0	0
Component 3	0	1	0	1

Test Results:

The results of the test are shown in Figure 5.10, where some important values are highlighted.

It can be observed that the results of the test are compliant to pass/fail criteria shown in Table 5.11, as the algorithm successfully detected all the faults injected, being able to identify the component in which the fault had been injected (Component 2 and 3 for accelerometers and Component 1 and 3 for gyroscopes).

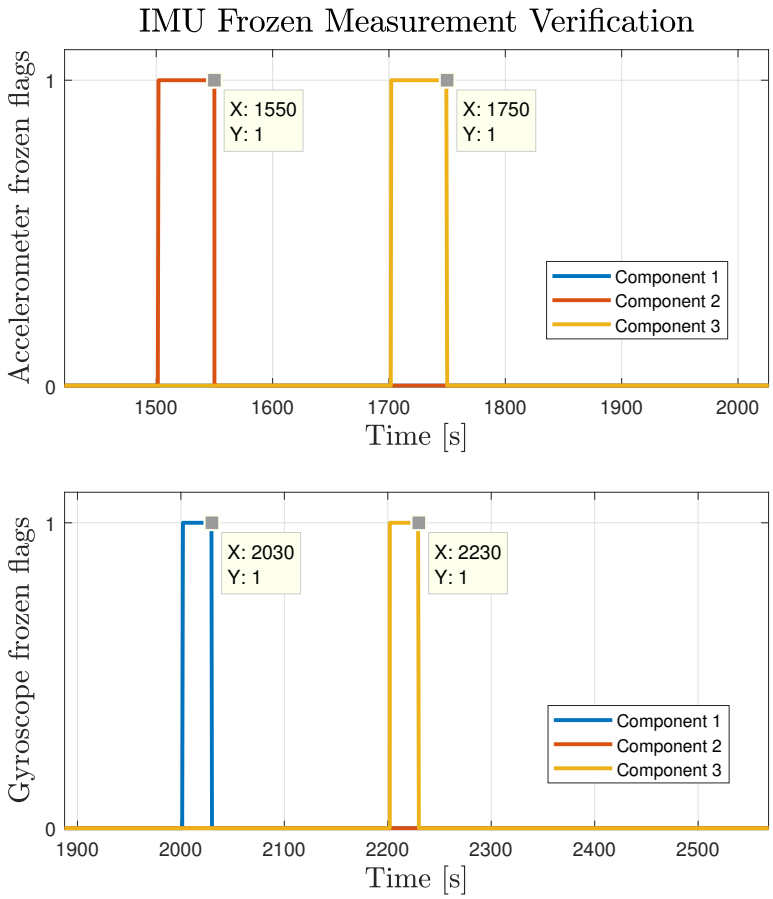


Figure 5.10: Test FD-02: results

The test has been passed successfully.

5.3.2.3 TEST FD-03 - IMU OTR Measurement Check Verification

Test Description:

The purpose of this test is to verify the proper functioning of IMU Out of range measurement detection algorithm inserted in the IMU part of FDIR system included in the Preliminary Checks, as represented in Figure 4.15. This algorithm detects when a measurement overcomes a user-defined threshold and raises a flag.

Test Configuration:

To perform the verification of this block, linear signals have been used in all three components of accelerometers and gyroscopes signals, with different slopes. The simulation time has been set to 5000 s. Furthermore, the following faults have been introduced:

- Accelerometers: a constant fault is added to current measurement during 50 s and after 1000 s of simulation in the first component and the same negative fault is introduced after 1300 s of simulation in the third component.
- Gyroscopes: a constant fault is added to current measurement during 100 s and after 1500 s of simulation in the second component and the same negative fault is introduced after 1800 s of simulation in the first component.

The constant fault intensity is calibrated in order to overcome the user-defined threshold during fault injection.

Pass/Fail Criteria:

The algorithm does perform identification, because it is important to know where an outlier is detected to undertake the proper recovery action. For this reason, the test is considered passed if the results of the rest respects the following pass/fail criteria, listed in Table 5.12, where the components of the signal represents the expected outcome in terms of flags raised by the system.

Table 5.12: TEST FD-03 - pass/fail criteria

	ACC		**GYR**	
TIME [s]	1000-1050	1300-1350	1500-1600	1800-1900
Component 1	1	0	0	1
Component 2	0	0	1	0
Component 3	0	1	0	0

Test Results:

The results of the test are shown in Figure 5.11, where some important values are highlighted.

It can be observed that the results of the test are compliant to pass/fail criteria shown in Table 5.12, as the algorithm successfully detected all the faults injected, being able to identify the component in which the fault had been injected (Component 1 and 3 for accelerometers and Component 2 and 1 for gyroscopes).

The test has been passed successfully.

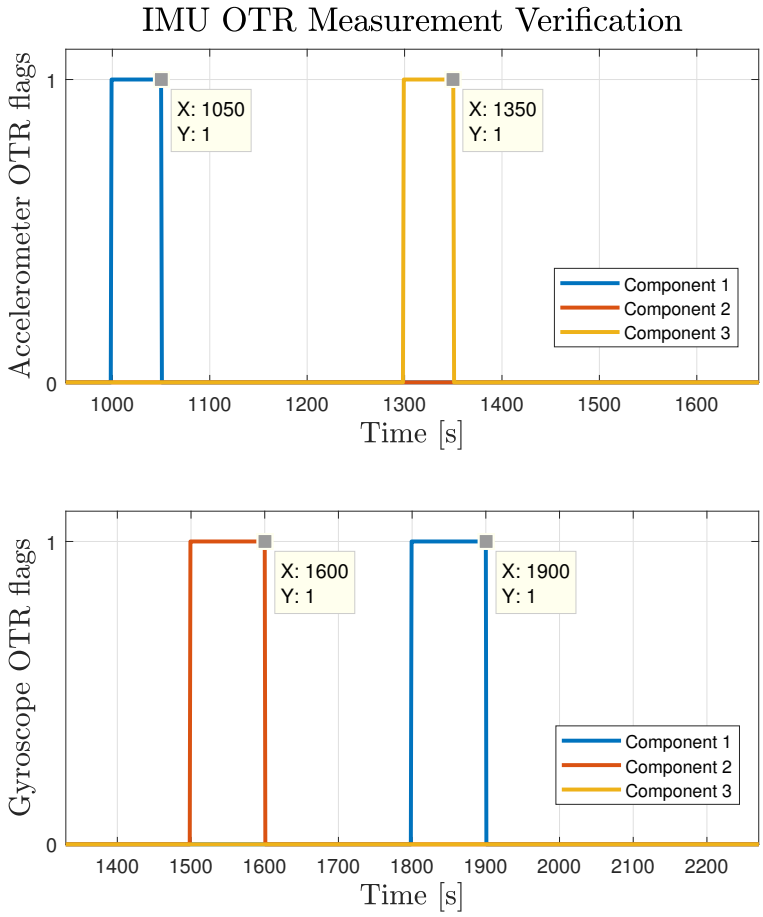


Figure 5.11: Test FD-03: results

5.3.3 FDIR Validation

5.3.3.1 GNSS FDIR Validation

The simulations listed in Table 5.13 have been executed to validate GNSS part of FDIR.

Table 5.13: Simulations executed for GNSS FDIR validation

ID	Fault Duration [s]	Description
FD-04	1 - 480	GNSS OTR and CUSUM test activated
FD-05	1 - 120	GNSS OTR only (increased threshold)
FD-06	1 - 120	GNSS CUSUM test only

In particular, the simulations have been executed introducing a constant fault with vari-

able intensity, duration and injection time in the whole GNSS measurement. The error introduced is summed to the current signal value in all the components with a different value. A MC analysis with 500 shots has been executed, each with a simulation time of 5000 s, of which the first 900 s represent the initial alignment phase. The purpose of this simulation is to validate GNSS FDIR algorithms and to evaluate the amount of seconds that the system can bear without using GNSS measurements, discarded when an outlier is detected.

Simulation FD-04 - GNSS OTR & CUSUM TEST

The results of Simulation FD-04, in terms of hybrid navigation unit performances, are represented in Figure 5.12, 5.13 and 5.14. In this simulation, both GNSS OTR check and CUSUM test are performed in cascade.

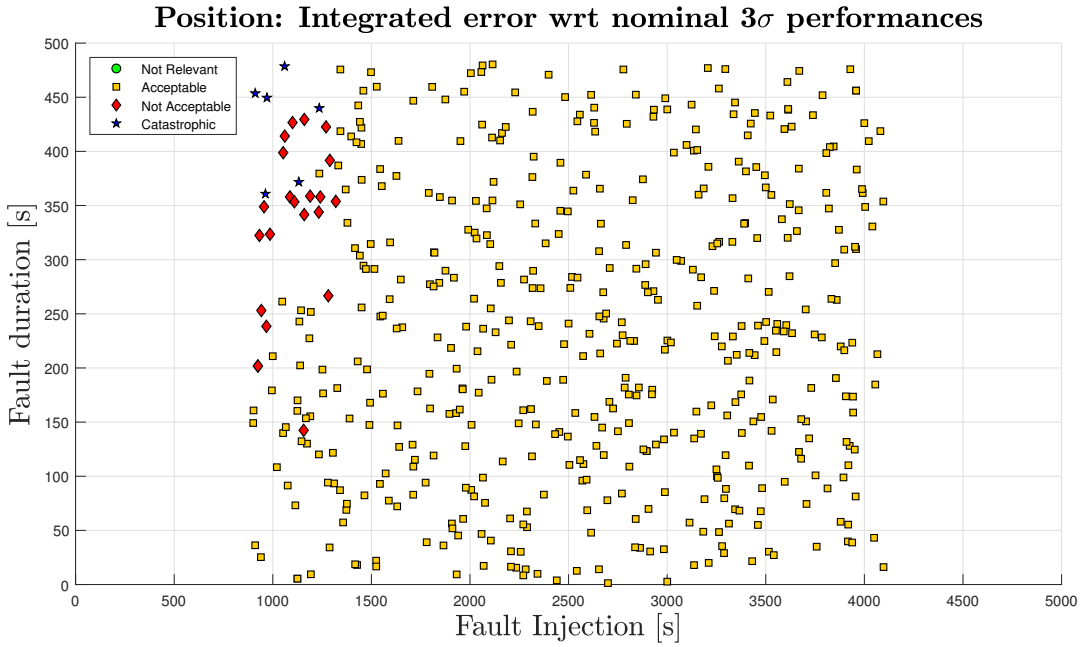


Figure 5.12: GNSS FDIR Validation: position performances (Simulation FD-04)

As it can be observed, position performances are acceptable in around 94% of the cases (Figure 5.12), and velocity performances are acceptable in around 92% of the simulations (Figure 5.13). As expected, the introduction of a fault with long duration during missile phase may result in catastrophic consequences, due to the fact that this phase presents a high dynamic. It can be observed as after around 120 s, unacceptable performances or catastrophic ones begins to appear for position and velocity, establishing an upper limit for transition to GNSS HSE.

Attitude performances are slightly better, with less than 1% of unacceptable performances, always in the missile phase and for faults with long duration, as shown in Figure 5.14.

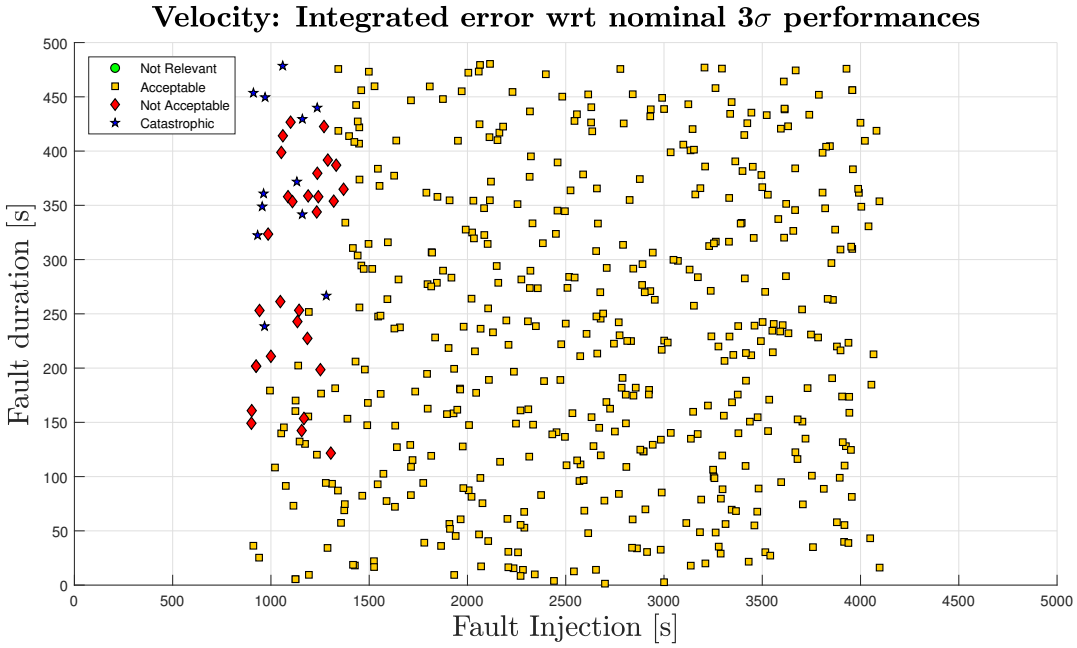


Figure 5.13: GNSS FDIR Validation: velocity performances (Simulation FD-04)

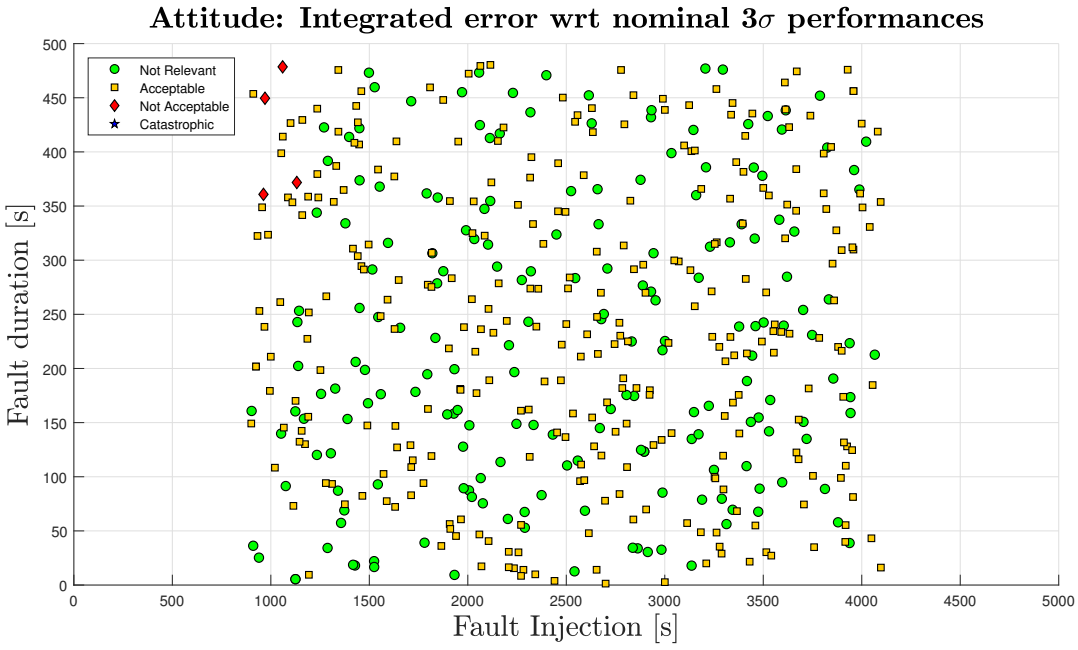


Figure 5.14: GNSS FDIR Validation: attitude performances (Simulation FD-04)

FDIR algorithms performances can be observed in Figure 5.15. In particular, this Figure represents the fault detected by GNSS Out of Range check and by CUSUM test. It can be noticed from that the faults introduced have been detected in the 100% of the cases, validating the two algorithm, i.e. no faulty measurement have been wrongly included in the computation of the navigation solution. It is also important to notice that only in a few simulations the fault has been detected by CUSUM test: this can be explained considering that when a fault is detected by GNSS OTR check, the flag `gnss_validity` is set to 0, and the CUSUM test is not executed. So, in this simulation, the CUSUM test only detects the faults that are considered inside the allowable range, in particular it can be observed that some faults are detected by both CUSUM and GNSS OTR: this is possible because during the fault injection, the state covariance grows and as the adaptive threshold of OTR check depends from covariance, this may result in accepting a wrong measurement, i.e. a mis-detection, so the CUSUM test is activated to remove it.

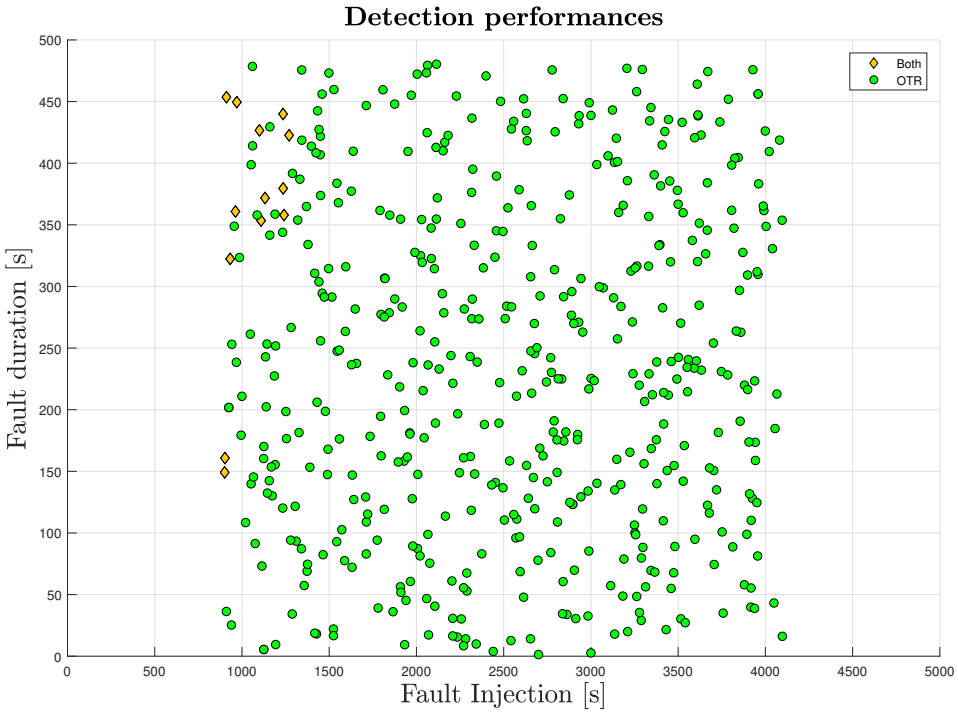


Figure 5.15: GNSS FDIR Validation: detection performances (Simulation FD-04)

Also, it is interesting to see another performance statistics regarding FDIR algorithms, regarding the false alarms. While for CUSUM test the false alarm rate is very low (false alarms have been detected in 2% of the simulations), false alarms flagged by GNSS OTR can be detected in more than 95 % of simulations. The causes of these false alarms may be due to poor tuning of the thresholds involved, GNSS OTR check in particular, since it has an adaptive threshold which is difficult to tune. False alarms are also due to

the fact that when a long fault is introduced, the covariance of the state grows because GNSS measurement is discarded and cannot correct the estimation, that drifts. For this reason, when the fault injection ends, the difference between GNSS and Inertial solutions may be greater than the threshold, leading to the wrongly rejection of correct GNSS measurements. For these reasons, two more simulations (listed in Table 5.13) have been executed to validate CUSUM test and GNSS OTR check independently. Also, these two simulations can help comparing the two algorithms, that have the same function. In the two additional simulations, the fault duration has been reduced to 120 s. Also, OTR Check has been executed with a higher threshold, to reduce false alarm rate.

Simulation FD-05 - GNSS OTR only

The results of Simulation FD-05, in terms of hybrid navigation unit performances, are represented in Figure 5.16, 5.17 and 5.18. In this simulation, only GNSS OTR is enabled.

Regarding position performances (Figure 5.16), as expected the level is acceptable in the 99% of the cases, with 1% of catastrophic outcomes in missile phase.

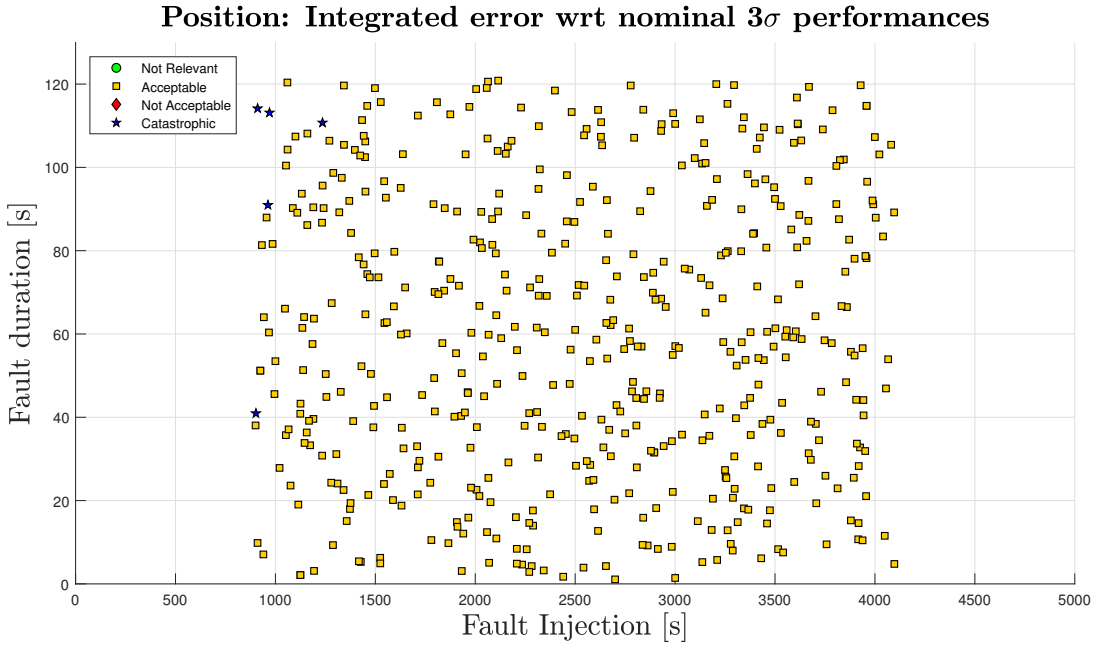


Figure 5.16: GNSS FDIR Validation: position performances (Simulation FD-05)

For the case of velocity performances in Figure 5.17, the situation is very similar to position's one. The level of performances is acceptable in more than 98 % of the cases with a 1% of catastrophic performances and less than 1% of unacceptable performances.

Finally, in Figure 5.18, attitude performances are represented. Here, the effect of the faults introduced is not relevant or acceptable in the 99% of the cases, with less than 1% of catastrophic and unacceptable performances.

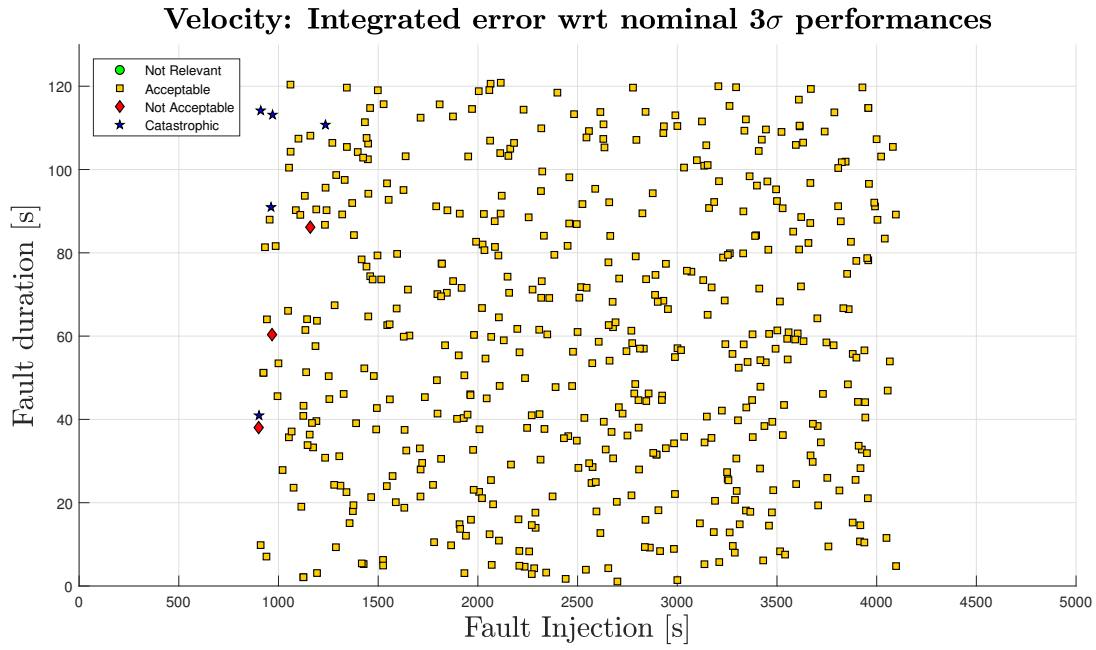


Figure 5.17: GNSS FDIR Validation: velocity performances (Simulation FD-05)

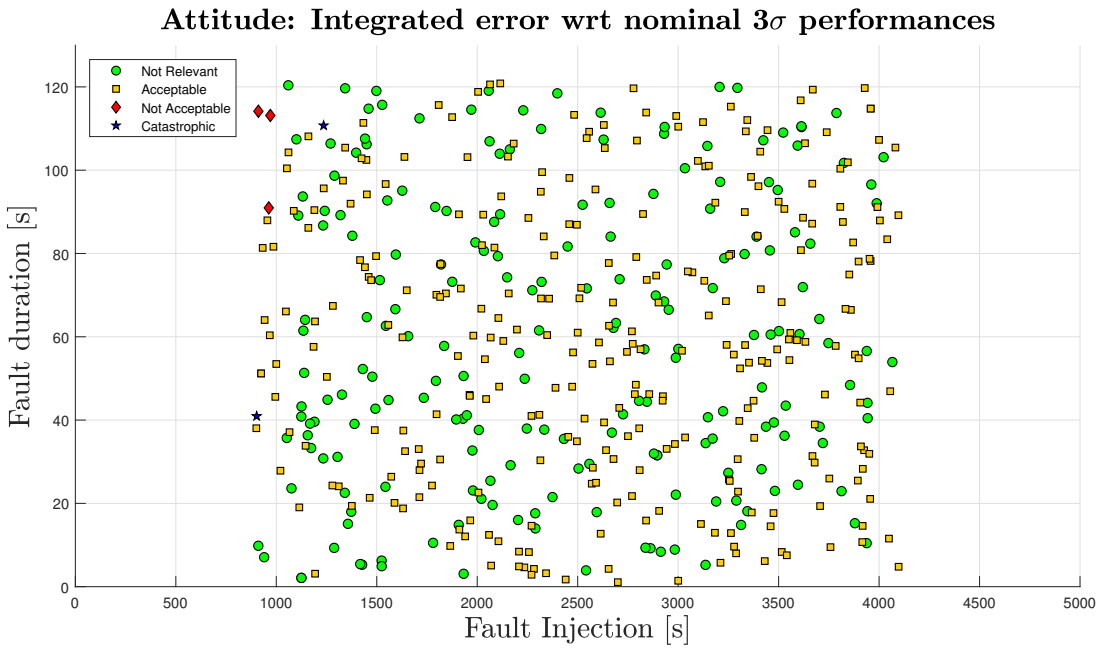


Figure 5.18: GNSS FDIR Validation: attitude performances (Simulation FD-05)

Every fault introduced has been detected by OTR algorithm, but, despite a higher threshold, the false alarm rate remain practically unchanged, probably due to the adaptive nature of the threshold of this algorithm. The few catastrophic/unacceptable performances can be explained as for the previous simulation: after the detection of the fault, the state covariance keep growing, raising the threshold until the faulty measurement is accepted.

Simulation FD-06 - GNSS CUSUM TEST only

The results of the last simulation concerning the validation of GNSS part of FDIR system, Simulation FD-06, are represented in Figure 5.19, 5.20 and 5.21. In this simulation, CUSUM test only has been enabled.

Regarding position performances (Figure 5.19), it can be noticed that there is a deterioration with respect to Simulation FD-05 results, as the performances are acceptable in the 88% of the cases and present unacceptable performances in 1% of the cases and catastrophic performances in 11% of the cases.

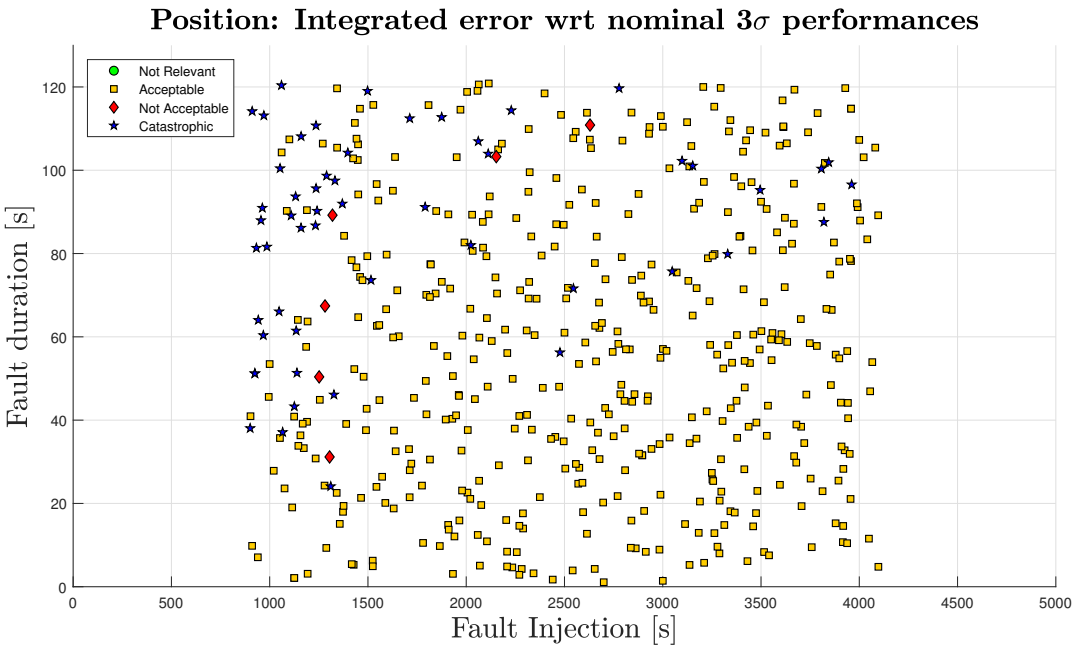


Figure 5.19: GNSS FDIR Validation: position performances (Simulation FD-06)

The situation is almost the same for velocity performances in Figure 5.20, with around 12% of catastrophic performances.

Finally, Figure 5.21 shows attitude performances, where a worsening can be seen with respect to Simulation FD-05 outcome, with 5% of catastrophic and unacceptable performances, while the effect of the faults introduced is not relevant or acceptable in 90% of the cases. As it was expected, the bigger concentration of catastrophic performances can be found when the fault is introduced during missile phase.

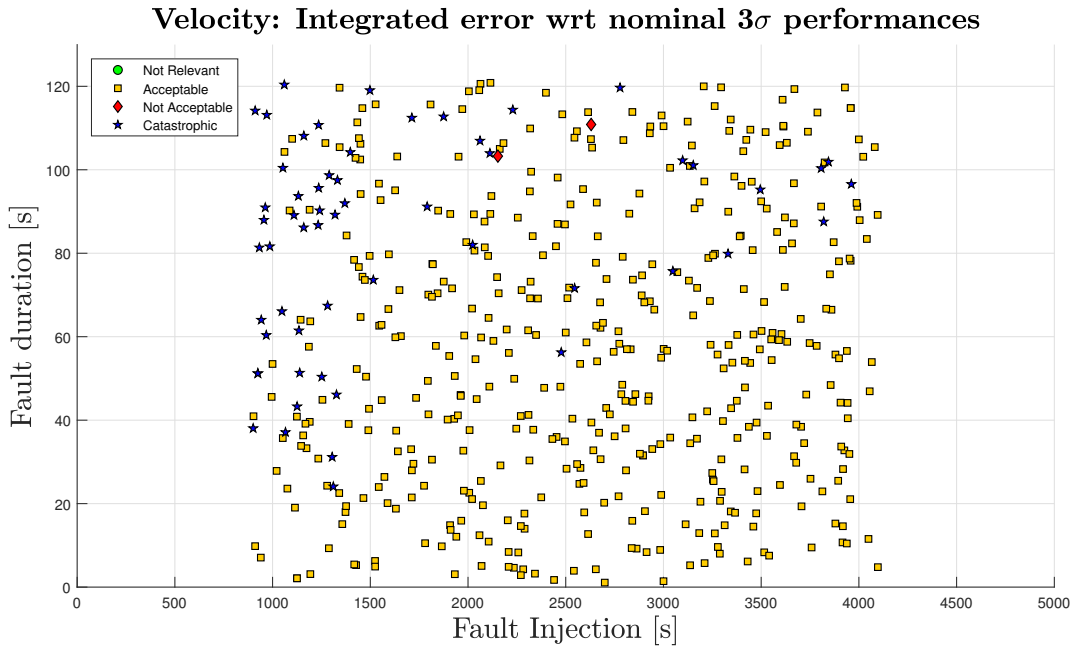


Figure 5.20: GNSS FDIR Validation: velocity performances (Simulation FD-06)

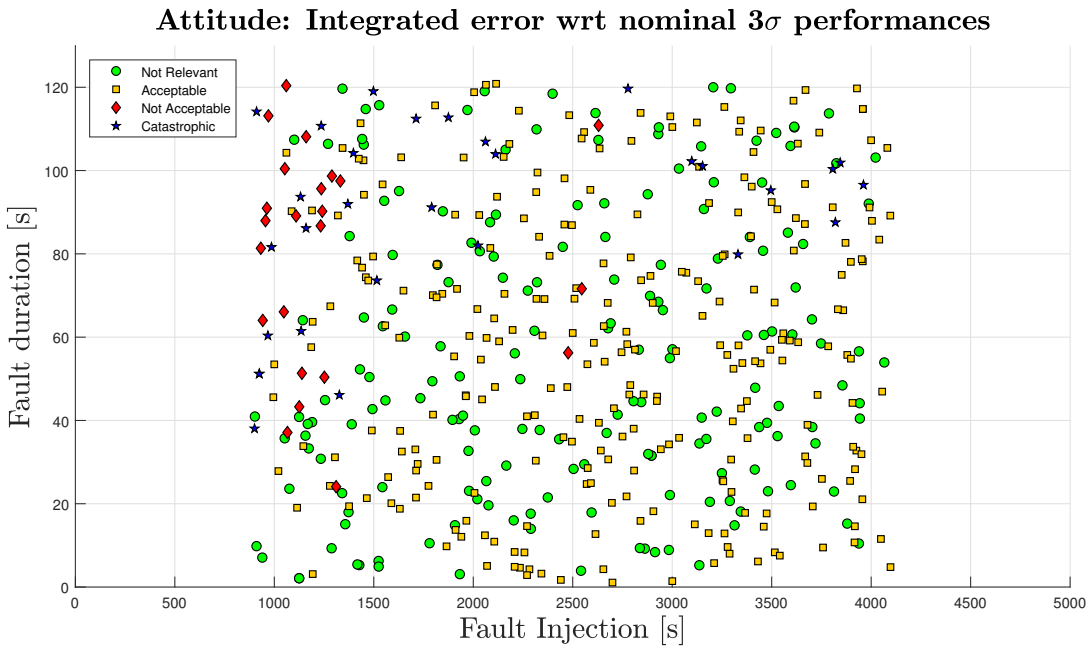


Figure 5.21: GNSS FDIR Validation: attitude performances (Simulation FD-06)

The algorithm has detected the fault introduced in 100% of the simulations, nevertheless in some cases the detection has been partial, meaning that for faults with low intensity, the algorithm may incorporate the first faulty measurement in the computation because the test statistic has not reached the threshold yet. Lowering the threshold may help with this inconvenient, but may also cause a growth of false alarm rate, that is already high.

The OTR check algorithm has very good performances, despite a very high false alarm rate and a required computational load that is slightly higher than CUSUM's one. Nevertheless, tuning the threshold is not immediate. CUSUM test shows good performances, that can be improved by a fine tuning of threshold and drift parameters, it can be considered the last defense of the system against the incorporation of faulty GNSS measurements, and also, it has a very low computational load. The best performances of GNSS FDIR, though, can be obtained by using the two algorithms together.

5.3.3.2 FDIR IMU Validation

Before describing Verification and Validation procedures and results, some previous studies about IMU's recovery, regarding the effect of frozen IMU measurements on hybrid navigation unit performances, that is how the system react to frozen recovery, and the effect of increasing process noise measurement in case of saturation due to shocks are described in the following Sections.

Effect of frozen recovery for IMU measurements

The aim of this study is to evaluate the effect of a frozen IMU measurement on hybrid navigation performances, that is to evaluate system response to frozen recovery. To perform this study, six Montecarlo simulations of 100 shots each have been executed. The simulations have been executed under nominal flight conditions, with a simulation time of 5000 s of which the first 900 s represent the initial alignment phase; the frozen recovery has been simulated introducing Frozen measurement faults in accelerometers and gyroscopes output during flight mode, with random injection time. The simulations executed are listed in Table 5.14.

Table 5.14: Simulations executed for the parametric study on frozen recovery

ID	Fault duration [s]	Description
FD-07	5	Frozen measurement in all GYR
FD-08	10	Frozen measurement in all GYR
FD-09	20	Frozen measurement in all GYR
FD-10	5	Frozen measurement in all GYR and ACC
FD-11	10	Frozen measurement in all GYR and ACC
FD-12	20	Frozen measurement in all GYR and ACC

As it can be observed from Table 5.14, the fault duration differs between the simulations in order to evaluate the time after which the navigation unit becomes unrecoverable in

case of a fault that implies frozen recovery. Also, the fault has been introduced only in gyroscopes at first, and then in the whole IMU's measurement. This is due to the fact that these two conditions implies that the whole IMU's measurement is not correct. In fact, Management SW algorithms involve a frame rotation, so even if the fault is introduced only in gyroscopes, the rotation transmits the error to accelerometers measurements too.

The result of the simulations in which the fault is introduced in gyroscopes only are presented in Figure 5.22, 5.23 and 5.24 for position, velocity and attitude respectively.

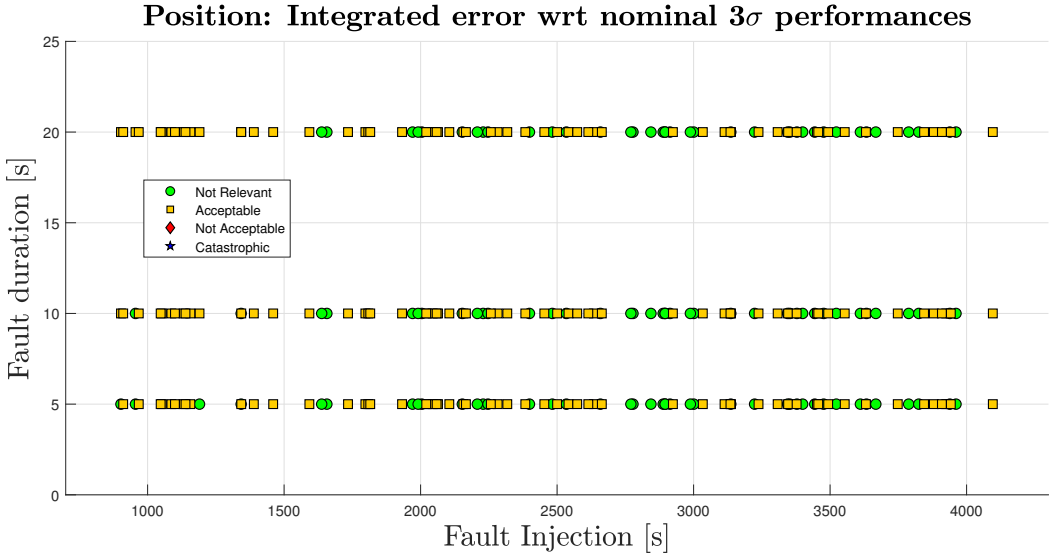


Figure 5.22: Hybrid navigation unit position performances for frozen gyroscopes measurements (Simulations FD-07, FD-08 and FD-09)

As it can be observed from Figure 5.22 the hybrid navigation unit gives a pretty good estimation of position, considering that introducing a 20 seconds long frozen measurement, the effect on the estimation, compared to the nominal condition, is not relevant in the 48% of the shots and introduce an acceptable error in the rest of the cases.

From Figure 5.23, it is possible to observe that velocity estimation gives slightly worse results than position estimation. For a 10 seconds long and a 20 seconds long frozen measurement, non-acceptable results and catastrophic errors respectively appear.

Finally, attitude performances can be evaluated from Figure 5.24. As expected, attitude estimation present the worst performances, and some non acceptable results are already present in case of introduction of a 5 seconds long frozen measurement (despite the fact that the effect of the fault is not relevant in 14% of the cases and the error is acceptable 76% of the cases).

As expected, the most critical zone for fault injection is the missile phase, that includes the powered flight phase through atmosphere and a harsh vibrational environment and high dynamic of the launcher in this phase.

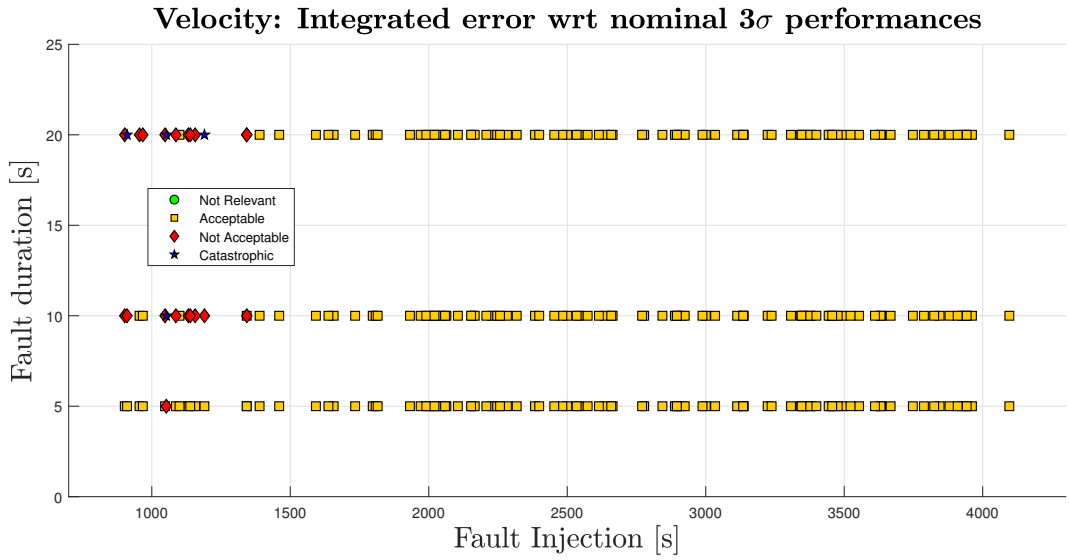


Figure 5.23: Hybrid navigation unit velocity performances for frozen gyroscopes measurements (Simulations FD-07, FD-08 and FD-09)

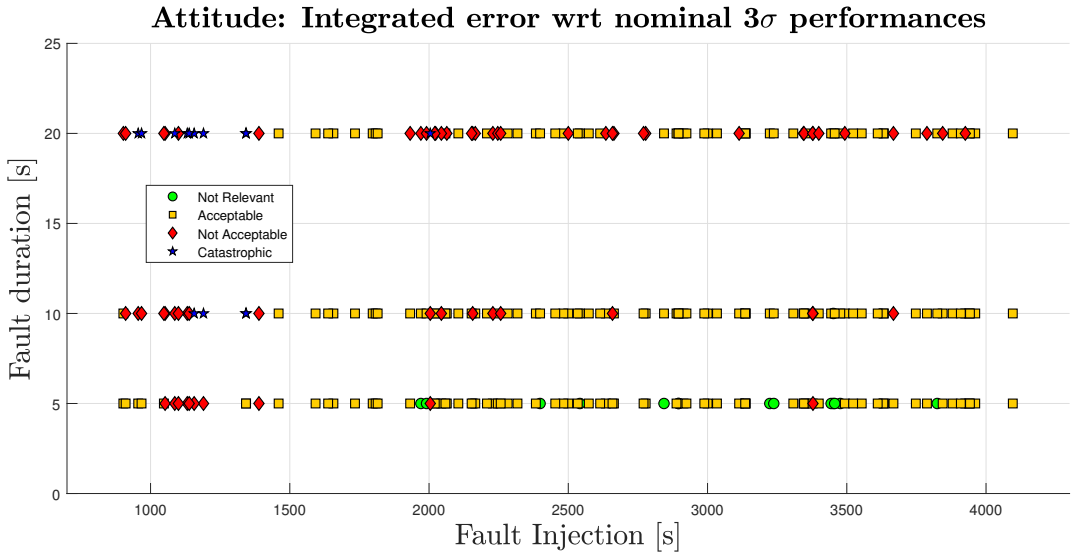


Figure 5.24: Hybrid navigation unit attitude performances for frozen gyroscopes measurements (Simulations FD-07, FD-08 and FD-09)

The result of the simulations in which the fault is introduced in gyroscopes and accelerometers are presented in Figure 5.25, 5.26 and 5.27 for position, velocity and attitude respectively.

For position, unacceptable results can be observed already for a 5 seconds long frozen measurement, as shown in Figure 5.25. Nevertheless, outside of missile phase, the frozen

measurement introduced have not relevant or acceptable consequences on hybrid navigation performances.

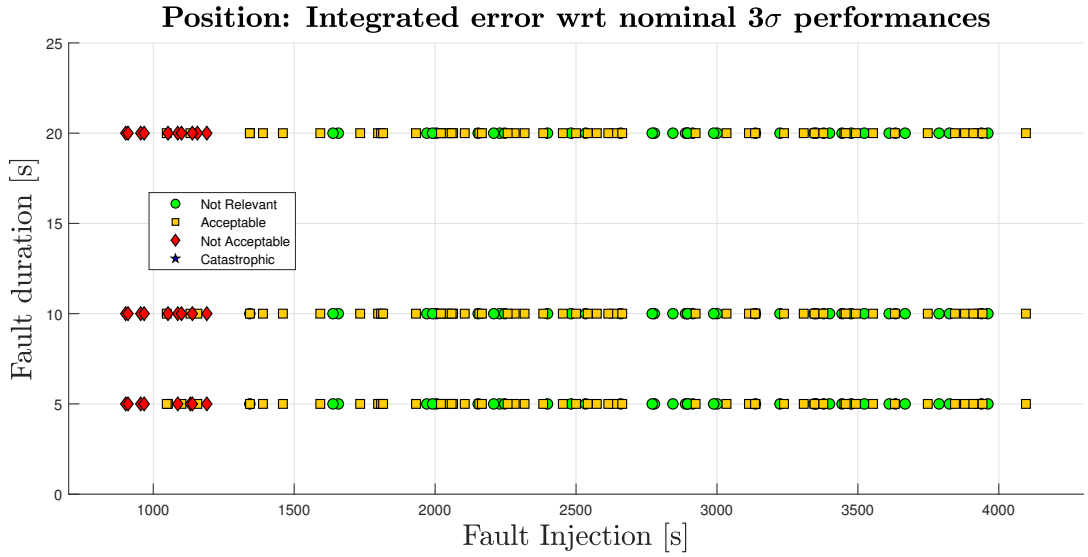


Figure 5.25: Hybrid navigation unit position performances for frozen IMU measurements (Simulations FD-10, FD-11 and FD-12)

Regarding velocity performances, almost all the frozen measurements introduced in missile phase during flight vibration give a catastrophic result, for the three different fault durations, as it can be observed in Figure 5.26. Nevertheless, the frozen measurements introduced after this phase generates acceptable performances except in one case, for a 20 seconds long frozen measurement.

As for position and velocity, there is a general worsening in hybrid navigation unit performances for attitude too. Catastrophic performances begin to appear already for a 5 seconds long frozen measurement introduced during missile phase.

As expected, the introduction of a frozen measurement in accelerometers too worsen the performances of the hybrid navigation unit for position, velocity and attitude. Again, missile phase is confirmed to be the critical zone for frozen measurement introduction, i.e. for frozen recovery, for long periods of time.

In general, it can be established that a frozen measurement with a duration shorter than 5 seconds can generate not acceptable or catastrophic consequences regarding hybrid navigation performances only if it is introduced in all the gyroscopes at least, and during missile phase. It is also important to consider that a 5 seconds long IMU fault is really unlikely, even more if it happens in the whole measurement. For these reasons, for IMU FDIR validation a maximum duration of 5 seconds for fault injection has been assumed.

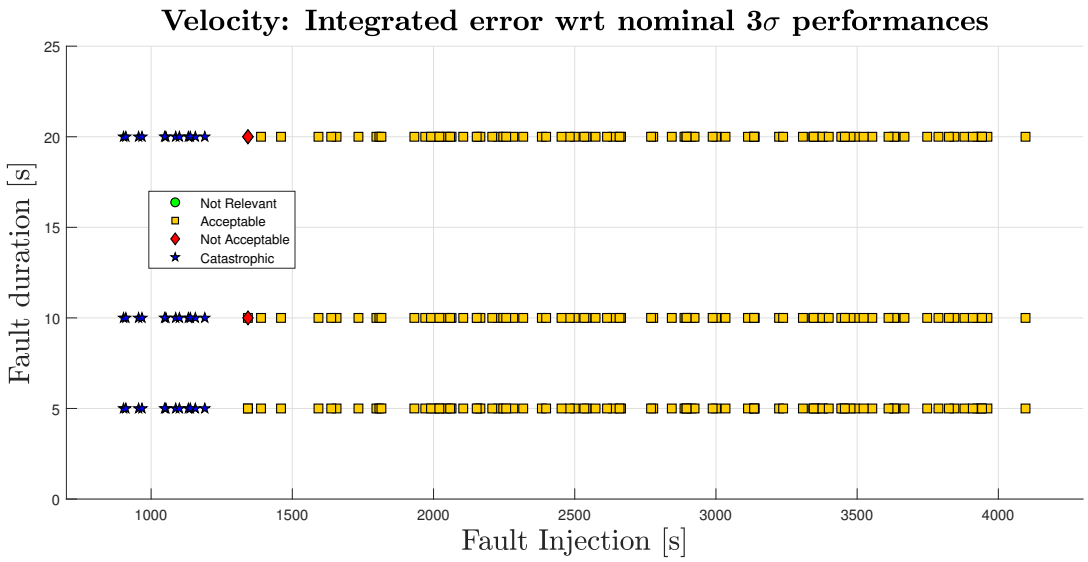


Figure 5.26: Hybrid navigation unit velocity performances for frozen IMU measurements (Simulations FD-10, FD-11 and FD-12)

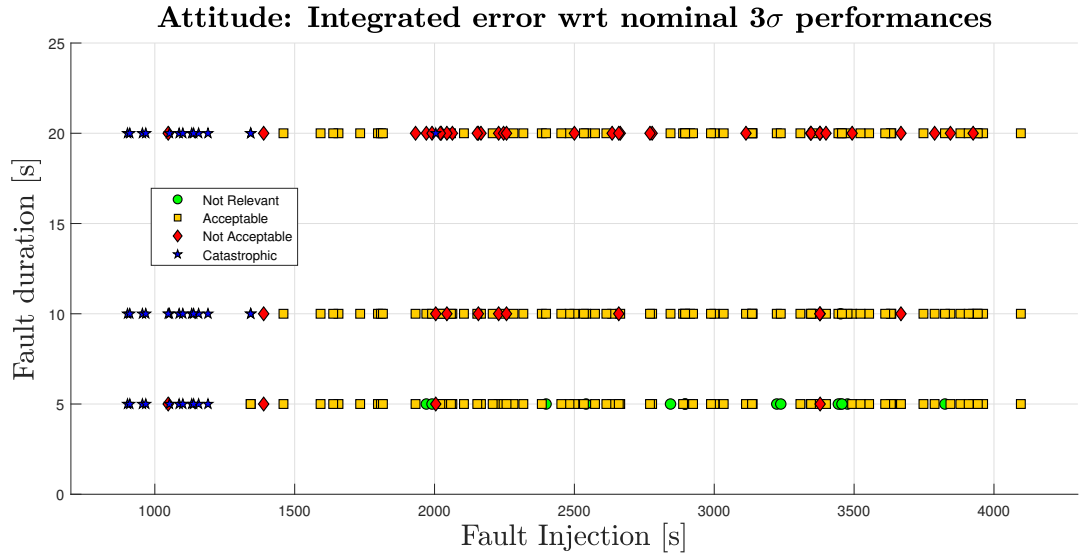


Figure 5.27: Hybrid navigation unit attitude performances for frozen IMU measurements (Simulations FD-10, FD-11 and FD-12)

Effect of increasing process noise covariance

The aim of this study is to evaluate the advantage that increasing process noise covariance when a faulty measurement is detected can bring to the hybrid navigation unit performances. For this purpose, two MC simulations of 100 shots each have been executed. The simulations are executed under nominal conditions, including flight vibrations during missile phase but also include accelerations generated by shocks due to stages burn and separation. In fact, when shock acceleration history is added to nominal acceleration, some of the sensors may provide out of range measurements. In this cases, the previous correct measurement is maintained until the IMU's CBIT stop flagging saturation. In addition to this recovery, one of the simulation has been executed increasing process noise covariance in Navigation block, when saturation is detected. The list of simulation executed is presented in Table 5.15.

Table 5.15: Simulations executed for the study on process noise covariance

ID	Shock enabled	Recoveries
FD-13	YES	Frozen measurement and increase of process noise in case of saturation
FD-14	YES	Frozen measurement only in case of saturation

The results of the simulations are not represented with the method described in previous sections because the difference between the performances of the hybrid navigation unit in the two cases is not appreciable. Nevertheless, the simulations shows that increasing the process noise covariance in presence of sensors' saturation slightly improves estimation of position, velocity and attitude, providing a quicker (few seconds) convergence of the filter to acceptable values.

IMU outlier detection methods validation

For the validation of the IMU part of FDIR system, the simulations listed in Table 5.16 have been executed:

Table 5.16: Simulations executed for IMU FDIR validation

ID	Fault Duration [s]	Description
FD-15	0 - 0.04	KDE only (fault in ACC only)
FD-16	0 - 0.04	KDE only (fault in GYR only)
FD-17	0.5 - 5	PC only (fault in ACC and GYR)
FD-18	0 - 5	PC and KDE (fault in ACC and GYR)

In particular, the simulations have been executed introducing a constant fault with variable intensity, duration and injection time in accelerometers, gyroscopes or both. The intensity of faults introduced has been set to represent faults that are not detected by IMU Out of Range check, i.e. faults intensity is lower than the maximum allowed value of ΔV and $\Delta\theta$. The error introduced is summed to the current signal value in all the components with a

different value. A MC analysis with 500 shots has been executed, each with a simulation time of 5000 s, of which the first 900 s represent the initial alignment phase.

As explained in Chapter 4, Sliding Window Kernel Density Estimator algorithm is an algorithm that only works with the data stream, without taking into account the governing equations of the phenomenon studied. For this reason, as it analyzes data at IMU frequency, that is very high, it is able to detect only short faults introduced. So, shorter faults have been injected in Simulation FD-15 and FD-16.

Furthermore, the simulations relative to IMU FDIR validation have been executed without introducing flight vibrations during missile phase, due to the fact that the KDE and the Predictor Corrector work very poorly in presence of vibrations.

Only frozen recovery has been used as recovery method, because prediction recovery solution tends to grow unbounded even for short faults introduced.

Simulation FD-15 and FD-16 - KDE Validation

The results of Simulation FD-15 and FD-16, about IMU Sliding Window KDE validation, are represented in Figure 5.29, 5.30, 5.31 and 5.32. In both simulations, position performances are omitted because they are not affected in a particular way by the fault injection and the recovery, showing a 100% of acceptable performances.

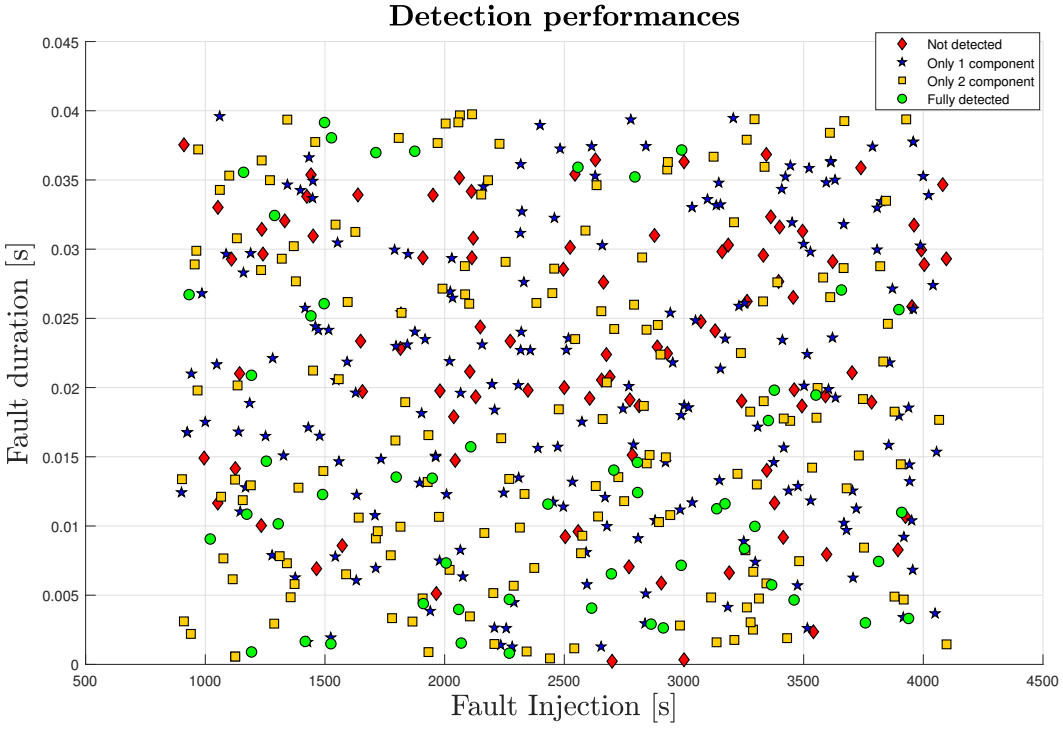


Figure 5.28: IMU FDIR Validation: detection performances (Simulation FD-15)

Also, it has to be noticed that KDE algorithm worked much better for the detection of faults introduced in accelerometers than for the detection of the ones introduced in gyroscopes. In fact, no faults were detected during Simulation FD-16. KDE Detection performances for Simulation FD-15 are shown in Figure 5.28: 18.4% of undetected faults, 70.8% of partially detected faults (only one or two components) and 10.8% of fully detected faults can be observed. Also, no false alarms were flagged.

Velocity performances are shown in Figure 5.29 and 5.30 for faults introduced in accelerometers only and gyroscopes only, respectively.

As it can be noticed in Figure 5.29, velocity performances are not acceptable in around 7% of the simulations, due to the partial detection of the fault in accelerometers, acceptable for the rest of the simulations. Much worse performances are shown in Figure 5.30 due to the propagation of faults injected in gyroscopes and not detected.

Attitude performances are shown in Figure 5.31 and 5.32 for Simulation FD-15 and FD-16 respectively. It can be observed that, during Simulation FD-15, where the fault is injected in accelerometers only, catastrophic performances begin to appear in around 12% of the cases, together with a 13% of not acceptable performances, due to the partial detection of some faults. Also 75% of the simulations show that the effect of the fault introduced is not relevant or leads to acceptable performances. Much worse performances are obtained from Simulation FD-16, shown in Figure 5.32. As none of the faults injected in gyroscopes has been detected by KDE, their effects are catastrophic in 94% of the simulations, not acceptable in around 5% of the simulations.

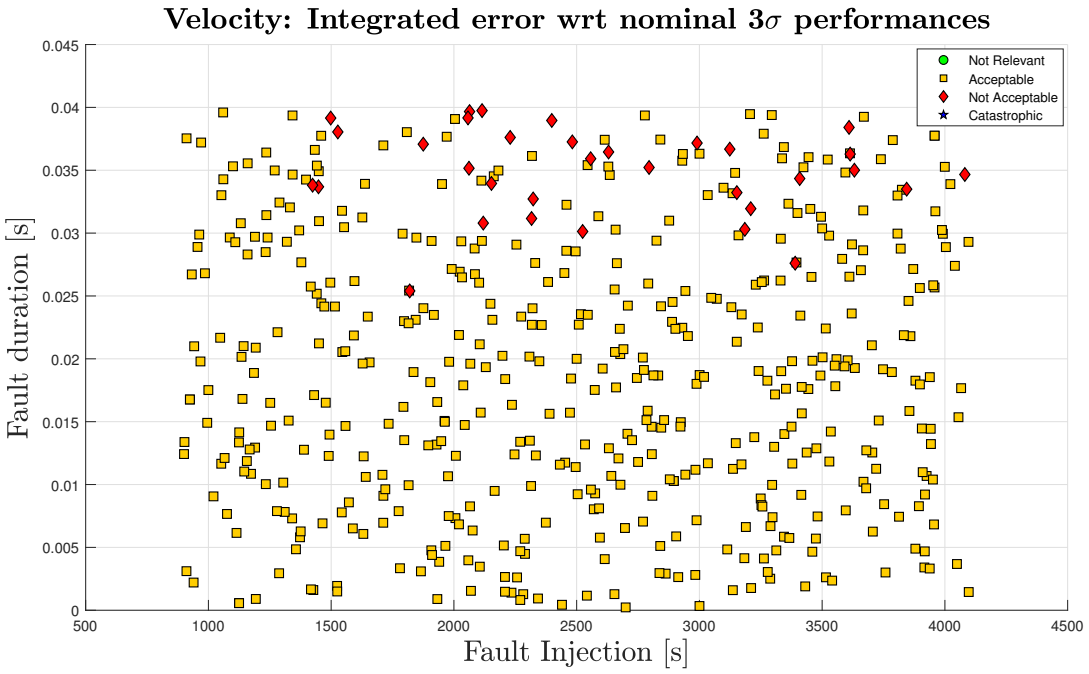


Figure 5.29: IMU FDIR Validation: velocity performances (Simulation FD-15)

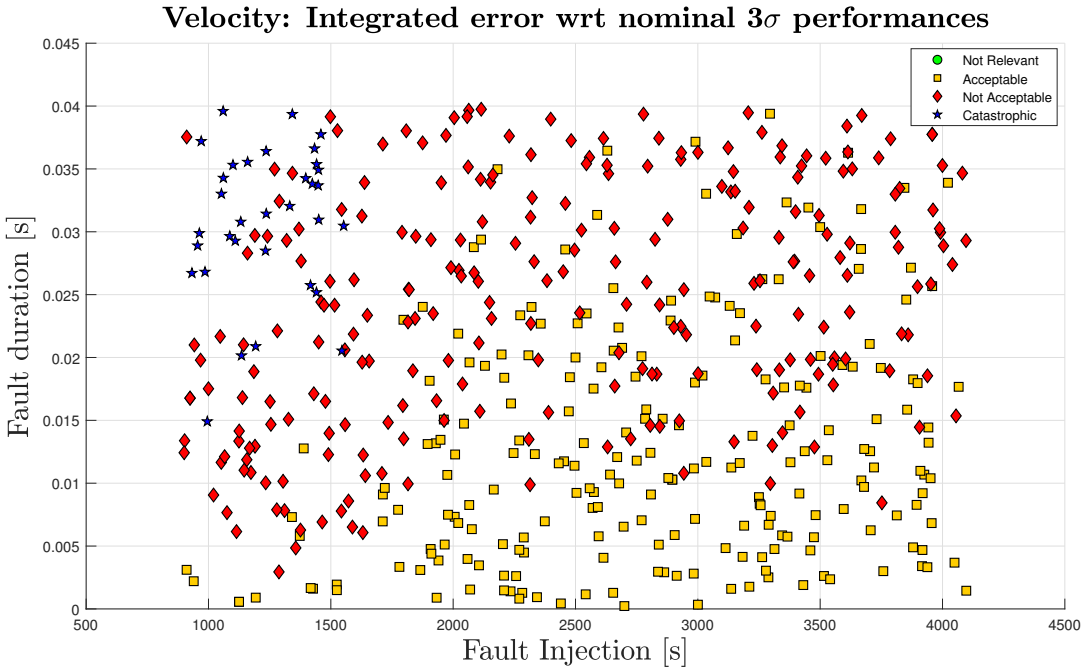


Figure 5.30: IMU FDIR Validation: velocity performances (Simulation FD-16)

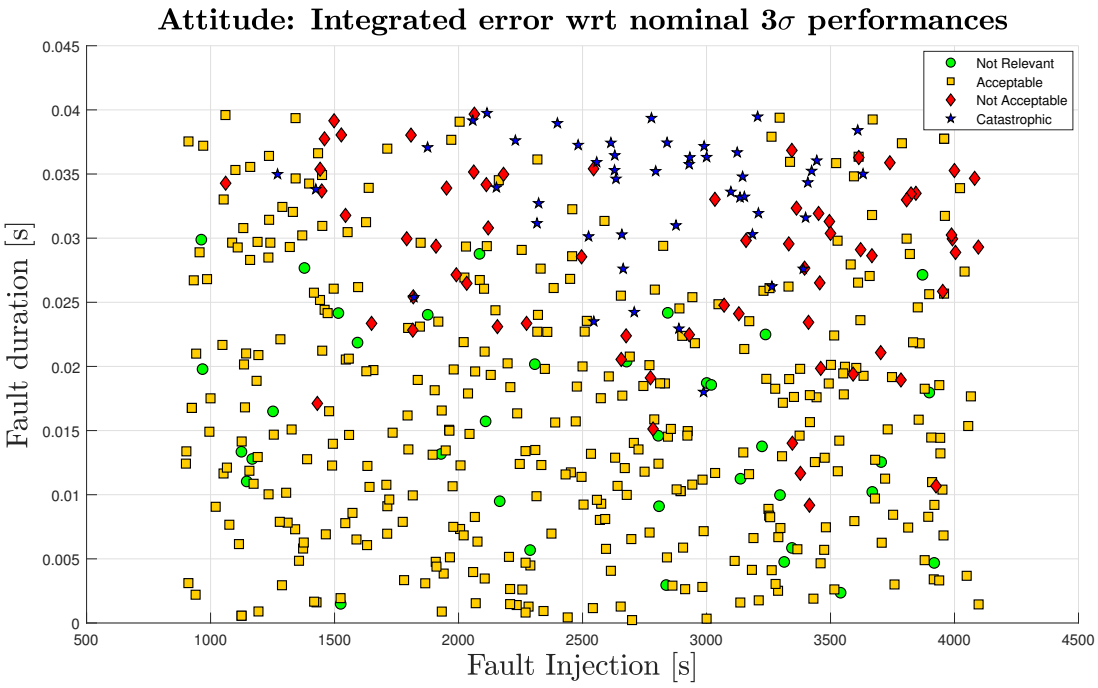


Figure 5.31: IMU FDIR Validation: attitude performances (Simulation FD-15)

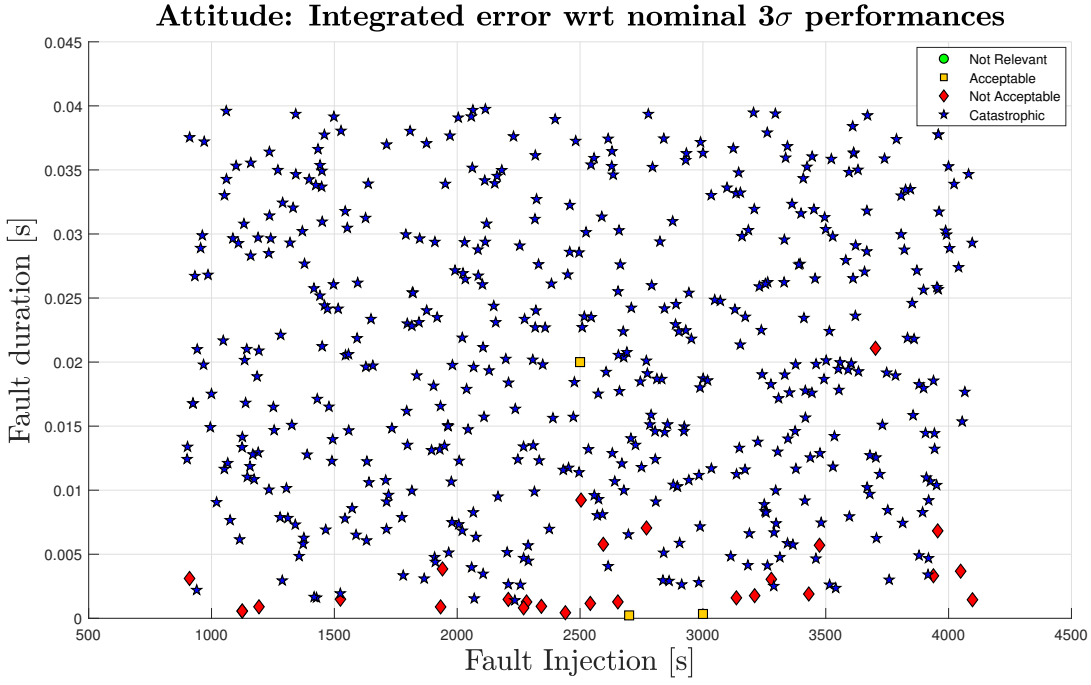


Figure 5.32: IMU FDIR Validation: attitude performances (Simulation FD-16)

Sliding Windows Kernel Density Estimator has proven to work much better for outlier detection in accelerometers, even though in around 70% of the cases the detection was only partial. Nevertheless, position and velocity performances can be considered acceptable, given the outcome of Simulation FD-15. On the other side, the algorithm works very poorly when it comes to outlier detection in gyroscopes, in fact the detection rate is 0%. None of the fault injected was detected, causing very poor performances both in attitude and velocity. This is probably due to the difference between accelerometers and gyroscopes data streams. In fact, the algorithm is based only on statistics concepts and does not take into account the governing equations of the studied phenomenon.

Simulation FD-17 - Predictor-Corrector validation

The outcome of Simulation FD-17, regarding the validation of Predictor-Corrector algorithm, is shown in Figure 5.33, 5.34 and 5.35. The algorithm presents a perfect detection score, with 100% of detected faults, even though the false alarm rate is high too, with 99% of simulations presenting false alarms.

Position performances are shown in Figure 5.33. It can be observed that catastrophic (around 15%) and not acceptable (4%) performances appear, mostly for faults introduced during missile phase. The performances are acceptable for the rest of the simulations.

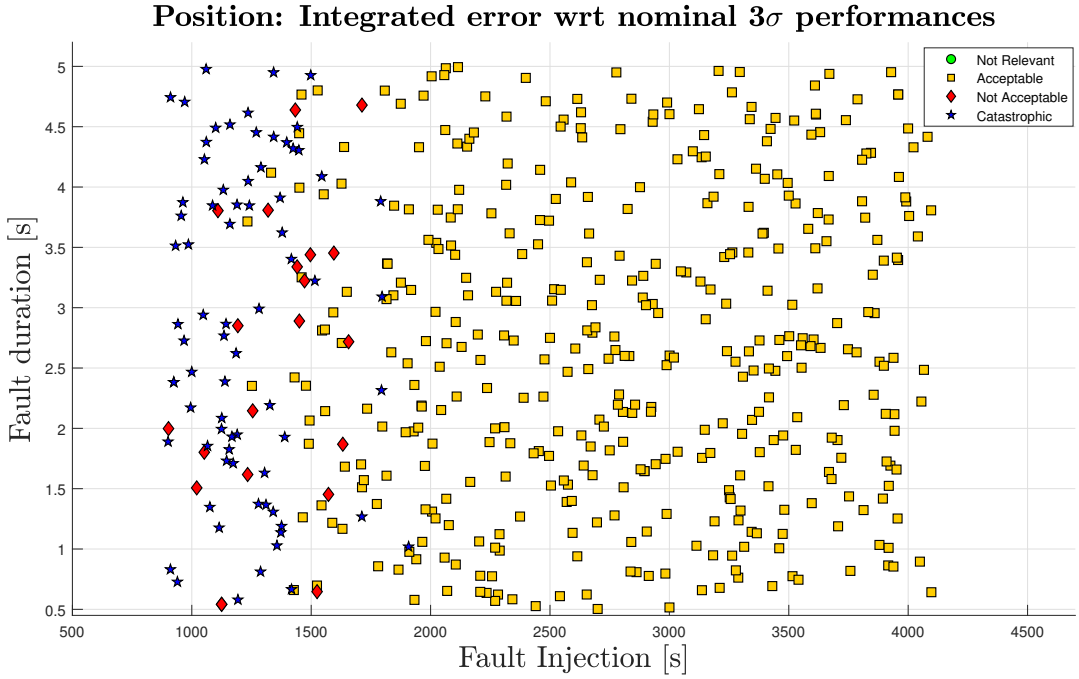


Figure 5.33: IMU FDIR Validation: position performances (Simulation FD-17)

Similar performances can be observed for velocity in Figure 4.3, where the effect of the faults introduced is catastrophic in the 17% of the cases and not acceptable in around 5%

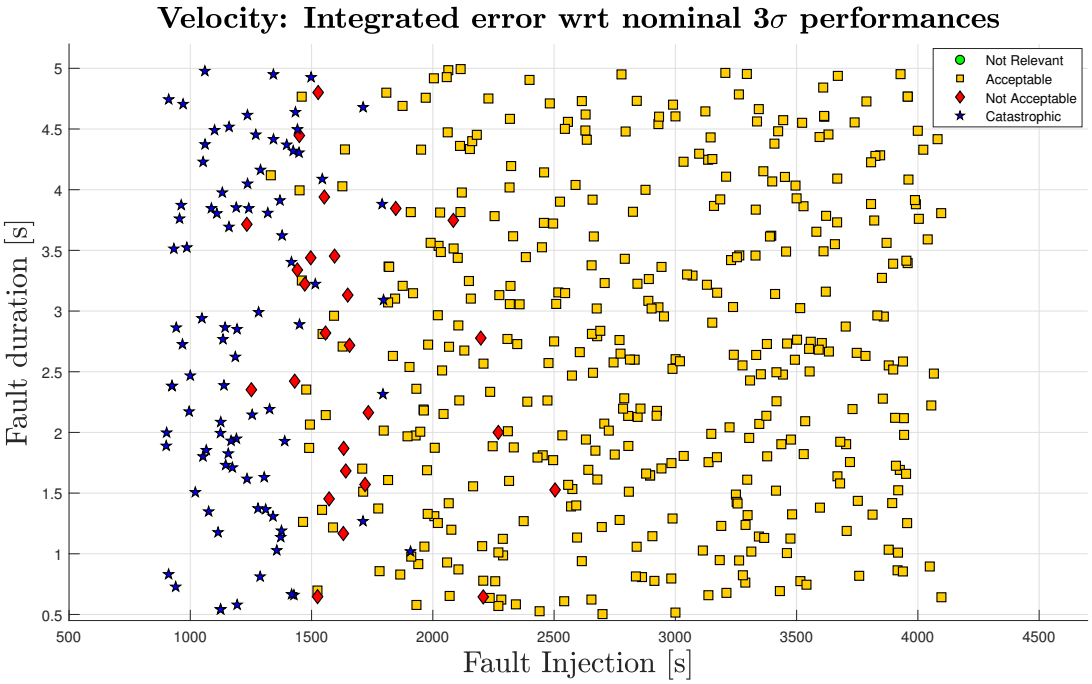


Figure 5.34: IMU FDIR Validation: velocity performances (Simulation FD-17)

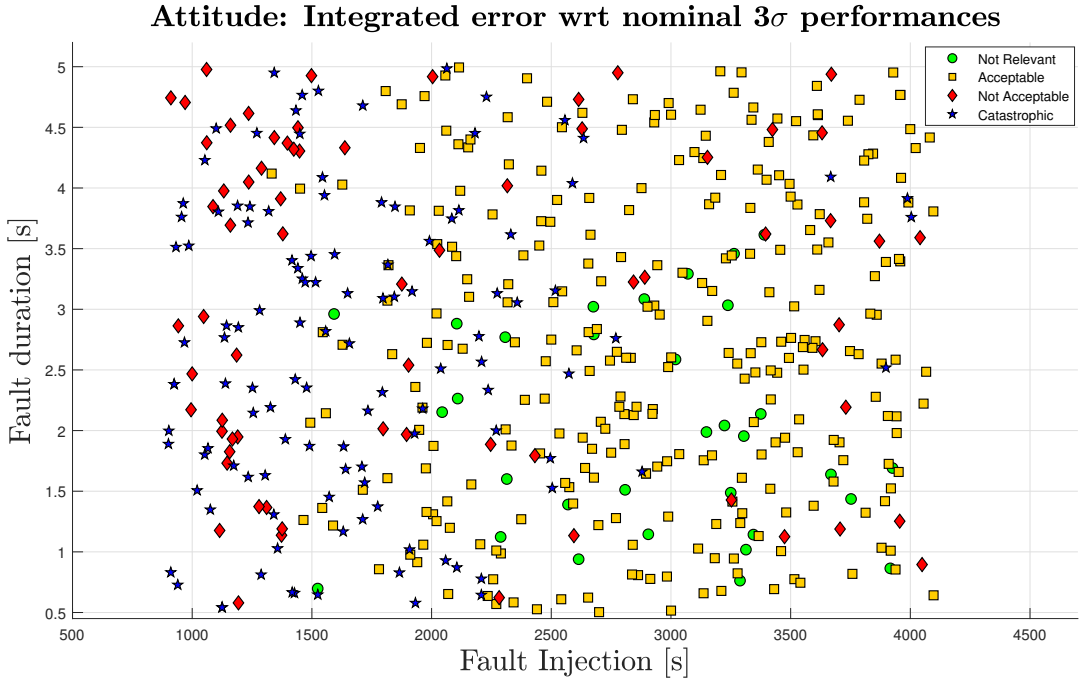


Figure 5.35: IMU FDIR Validation: attitude performances (Simulation FD-17)

of the simulations. This is especially true for faults introduced during missile phase.

Attitude performances are shown in Figure 5.35 and are slightly worse than velocity and position ones. In fact, the effect of the faults introduced in accelerometers and gyroscopes is catastrophic in 23% of the simulations, not acceptable in 13.8% of the cases, acceptable in 56.4% of the simulations and not relevant in the remaining 6.4%. Even though most of the catastrophic and not acceptable performances can be found during missile phases, some faults introduced during orbital phase had important consequences on attitude performances.

After an analysis of the worst performances obtained during Simulation FD-17, the cause of the high rate of catastrophic and not acceptable performances, even for short faults, can be identified in a small delay in outlier detection by Predictor-Corrector algorithm, that causes the incorporation of some faulty measurements in the computation. Furthermore, the false alarms flagged by the detection method may have caused the rejection of many inliers, causing a worsening of the performances.

Simulation FD-18: KDE & PC

The results of simulation FD-18 are represented in Figure 5.36, 5.37 and 5.38. The Figures basically confirm the results of Simulation FD-17, showing the same performances regarding position velocity and attitude.

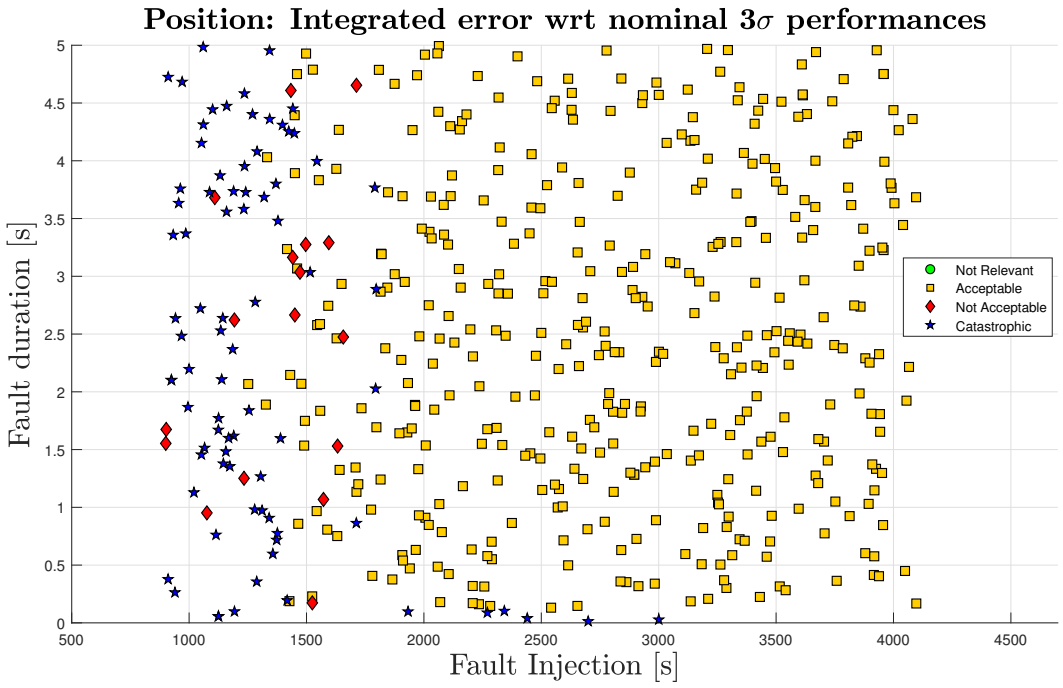


Figure 5.36: IMU FDIR Validation: position performances (Simulation FD-18)

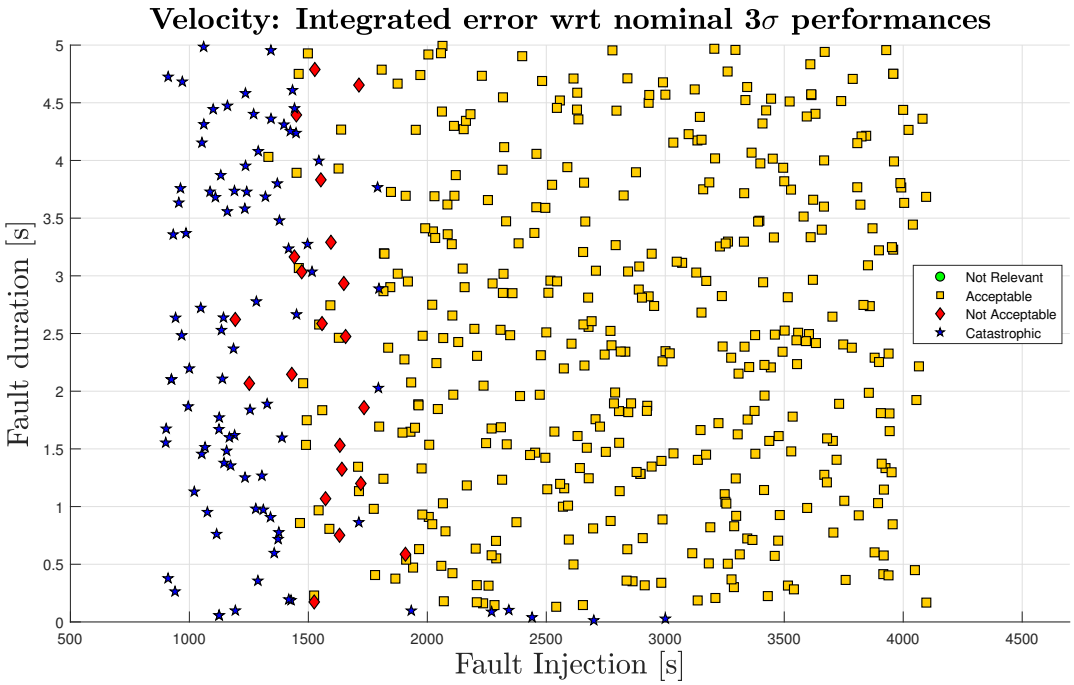


Figure 5.37: IMU FDIR Validation: velocity performances (Simulation FD-18)

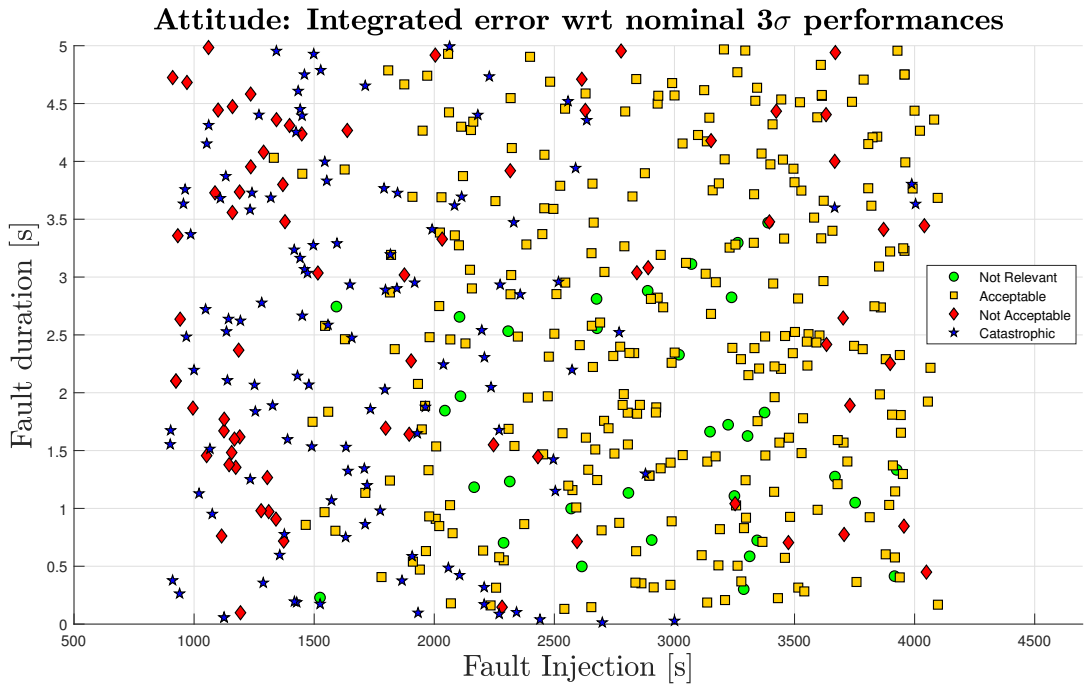


Figure 5.38: IMU FDIR Validation: attitude performances (Simulation FD-18)

As said before, the results of previous simulation FD-17 are confirmed, with the same percentage of catastrophic, not acceptable and acceptable performances regarding position, velocity and attitude.

Also, it is interesting to notice that the Predictor-Corrector algorithm detects every fault introduced in the simulation, despite a high false alarm rate, flagging false alarms in every simulation. On the other side, KDE is not able to detect long faults but it only gives a partial detection for some of the introduced faults. This can be observed in Figure 5.39, where detection performances of the two algorithms are shown. This partial detection has to be intended as partial in the number of components in which the fault has been detected (only 1 or 2 components out of 6, in fact the faults have been injected in the whole IMU measurement), and partial in duration (fault detected only during some instants of time and not during its whole duration). This happens because the points that expires from the window and are not detected as outliers contribute to make some faults harder to detect.

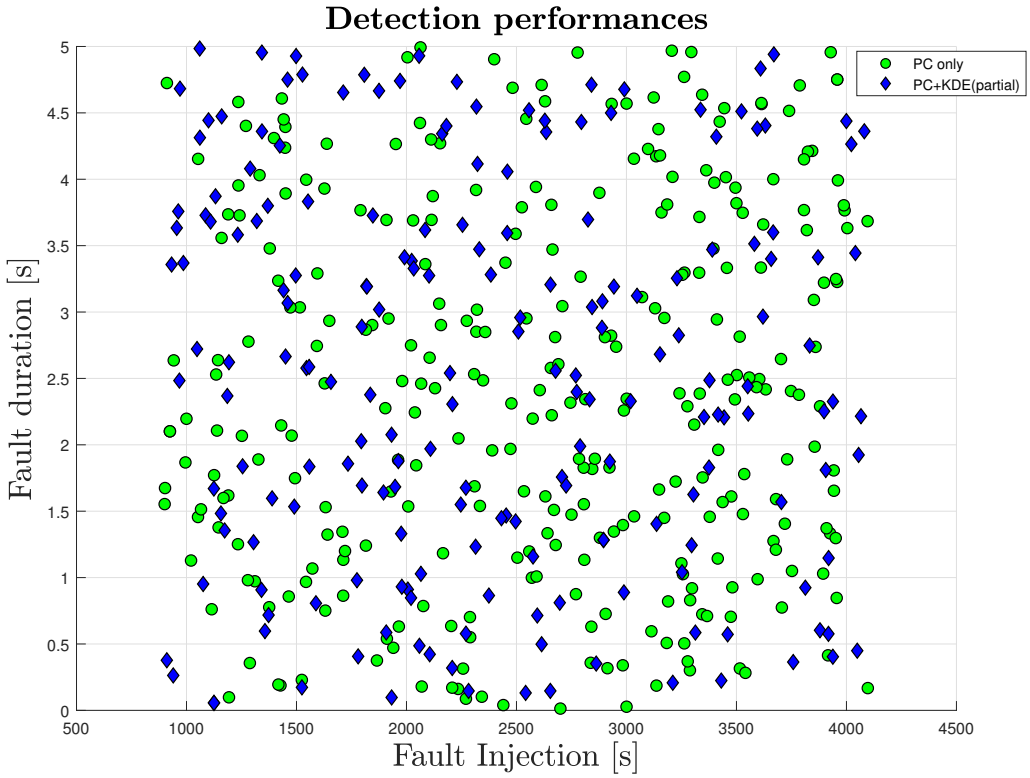


Figure 5.39: IMU FDIR Validation: detection performances (Simulation FD-18)

5.3.4 Simulation list

A list of all the simulations executed during the V&V campaign is shown in Table 5.17, including the simulation time, the number of MC shots and a short description.

Table 5.17: Simulations executed during Verification and Validation campaign

ID	MC shots	Time [s]	Description
FI-01	1	5000	FIM verification: constant fault
FI-02	1	5000	FIM verification: function fault
FI-03	1	5000	FIM verification: profile fault
FI-04	1	5000	FIM verification: frozen fault
FI-05	1	5000	FIM verification: delay fault
FD-01	1	5000	FDIR verification: GNSS frozen measurements
FD-02	1	5000	FDIR verification: IMU frozen measurements
FD-03	1	5000	FDIR verification: IMU OTR measurements
FD-04	500	5000	GNSS FDIR Validation (OTR + CUSUM)
FD-05	500	5000	GNSS FDIR Validation (OTR)
FD-06	500	5000	GNSS FDIR Validation (CUSUM)
FD-07	100	5000	Study on frozen recovery (5 s GYR only)
FD-08	100	5000	Study on frozen recovery (10 s GYR only)
FD-09	100	5000	Study on frozen recovery (20 s GYR only)
FD-10	100	5000	Study on frozen recovery (5 s GYR & ACC)
FD-11	100	5000	Study on frozen recovery (10 s GYR & ACC)
FD-12	100	5000	Study on frozen recovery (20 s GYR & ACC)
FD-13	100	5000	Study on increasing process noise covariance
FD-14	100	5000	Reference simulation for process noise study
FD-15	500	5000	IMU FDIR Validation (KDE only, fault in ACC)
FD-16	500	5000	IMU FDIR Validation (KDE only, fault in GYR)
FD-17	500	5000	IMU FDIR Validation (PC only, fault in ACC and GYR)
FD-18	500	5000	IMU FDIR Validation (KDE+PC, fault in ACC and GYR)

Chapter 6

Conclusions

The aim of this section is to draw the conclusions of the work done during the development, the design, the implementation and the verification and validation of a preliminary FDIR system of a hybrid IMU/GNSS navigation unit for launchers, focusing on the results obtained. The lessons learnt and the future works that can be developed starting from this study.

After a State of the Art review of the FDIR methodologies and procedures, and given the focus of the study addressed towards the detection and recovery of software faults that can be reproduced in SENER Aeroespacial simulator, some previous analyses have been carried out in order to identify these faults and the critical items of the navigation unit, and a Fault Injection Model has been designed to introduce the faults in the simulator.

Then, the FDIR system has been designed to prevent these fault from affecting the performances of the navigation unit, through the selection of proper outlier detection algorithms applied to sensor's signals. The appropriate recovery actions have been selected, taking into account the consolidated and well defined system architecture provided by SENER Aeroespacial, and the whole system's logic has been designed and implemented in Simulink. The system has been designed *ad-hoc* for the studied application, as no common standards are available for FDIR design in general.

Finally, the integration of Fault Injection Model and FDIR system in SENER Aeroespacial simulator has been performed, so that the V&V campaign could be carried out through several Montecarlo analysis.

The results obtained from Validation & Verification campaign and a comparison between the implemented algorithms of FDIR system are shown in Table 6.1, as final overview of the performances of the system developed. The last column of Table 6.1 indicates the suitability of the method for future implementation in flight software, according to the conclusions drawn at the end of V&V campaign that are presented below.

Table 6.1: Comparison between FDIR algorithms

Algorithm	Application	CPU	Recovery	Impl.
GNSS Frozen	GNSS FDI	Low	Remove measurement	YES
GNSS OTR	GNSS FDI	Low	Remove measurement	YES
GNSS CUSUM	GNSS FDIR	Low	No update	YES
IMU Frozen	IMU FDI	Low	Increase of process noise covariance	YES
IMU OTR	IMU FDI	Low	Increase of process noise covariance or frozen recovery	YES
KDE	IMU FDI	High	Increase of process noise covariance or frozen recovery	NO
Predictor-Corrector	IMU FDI	High	Increase of process noise covariance or frozen recovery	NO
Frozen recovery	Recovery	Low		YES
Prediction recovery	Recovery	Med		NO
Increase process noise	Recovery	Low		YES

The final conclusions drawn after the results of Validation & Verification campaign are here presented. In particular, these results regard FDIR implemented algorithms performances can be summed up as follows:

GNSS FDIR algorithms:

- Regarding GNSS FDIR algorithms, CUSUM test is lighter, speaking of computational load, and has a perfect detection score but a fine tuning of threshold and drift is needed to prevent some catastrophic performances due to the incorporation of small constant faults that the algorithm is not able to detect immediately, as shown by Simulation FD-06 results. The tuning for detecting these small errors will surely lead to an increase in the false alarms, thus an extensive tuning campaign should be necessary to trade-off misdetection and false alarms ratio.
- On the other side, GNSS OTR check has better performances, as shown by Simulation FD-05 results, and it also has a perfect detection score despite its high false alarm rate, but is slightly heavier computationally speaking.
- Nevertheless, the best performances are obtained when the two algorithms works together, being able to withstand very long faults, even if the limit of allowable time without GNSS measurement has been set to 120 s. The GNSS FDIR part has a perfect detection score and a very high percentage of acceptable performances as shown by Simulation FD-04 outcome. Both algorithms can be considered suitable for implementation on a flight prototype of the hybrid navigation unit.

IMU FDIR algorithms:

- The impossibility to access IMU's measurement residuals limits the applicability of several outliers detection methods for IMU's measurements. In fact innovations and residuals are only available for GNSS measurements, and they have been used in CUSUM test. Nevertheless, some outlier detection algorithms have been found and studied in literature, relatively to IMU's data outlier detection, even if they does not exploit Kalman Filter properties and features.

- Regarding IMU FDIR algorithm validation, KDE shows much better performances in outlier detection for accelerometers than for gyroscopes, with around 82% of partially or fully detected faults for accelerometers and 0% for gyroscopes. An improvement in tuning for gyroscopes part is needed for this algorithm, which is not simple because of the many parameters involved in the tuning (bandwidth, thresholds, bin size and number).

- Predictor Corrector algorithm presents a perfect detection score. Nevertheless, it also presents an high false alarm rate and a small delay in fault detection that are the causes of catastrophic and not acceptable performances observed in Simulation FD-18, mostly during missile phase.

- Apart from their performances, IMU's detection algorithms work very poorly in presence of flight vibrations, for this reason their possible future implementation in a flight prototype must be studied more carefully. Furthermore, the implementation of Sliding Windows KDE is discouraged as it can only detect short errors for its own nature. Predictor Corrector algorithm presents more interesting features. Nevertheless, both algorithms are very heavy computationally speaking, and this drastically reduces their application when the available CPU is low. IMU's preliminary checks (frozen measurement and OTR) have the most chances to be implemented in future.

- During Validation campaign, it has been noticed that Prediction Recovery, that was implemented as possible recovery action for IMU's measurements after the execution Preliminary Checks, was not able to simulate properly new IMU's measurements because the recovered measurements tend to grow unbounded even when interpolating during short faults, causing the incorporation of IMU's data with huge values and leading to catastrophic effects on hybrid navigation unit performances. For this reason, this type of recovery has not been used during the Validation of IMU's part of FDIR system.

Finally, after the evaluation of FDIR performances, some additional conclusions can be drawn at hybrid navigation unit level. In fact, the results obtained from V&V campaign highlighted several FDIR-related design drivers that can be applied to the whole hybrid navigation unit, which are listed below:

- After observing results from Simulation FD-04, a limit of 120 s without GNSS measurement has been set as threshold for flagging a GNSS High Severity Error. Nevertheless, GNSS HSE are not critical for the navigation unit, leading to a degradation

of performances but not to the loss of mission. On the other side, from Simulation FD-07 to FD-12 outcomes, it has been observed that the navigation unit can barely withstand an IMU-related fault with a maximum duration of 5 s, especially if injected in the whole IMU's measurement (i.e. in accelerometers and gyroscopes).

- IMU-related faults are not considered acceptable in any case, being the IMU a critical component of the hybrid navigation unit. Furthermore, faults in IMU are difficult to detect and IMU's recovery is not guaranteed anyway, as shown by the outcomes of Simulation FD-15, FD-16, FD-17 and FD-18. A redundant inertial unit could help to avoid or reduce the risk caused by IMU-related faults.

6.1 Lesson learnt

During the study, design and implementation of FDIR system, several difficulties have been encountered. First, the lack of common standards in FDIR design in general and for the specific application of navigation unit for launchers in literature caused the choice of an *ad hoc* design, that is difficult to validate. In fact, as FDIR is becoming an interesting research topic for its importance in critical application, there is a huge literature currently developing and spreading, but it involves aeronautic or spacecraft systems mostly. So, there are no common practical rules to follow during the design and implementation of the FDIR system for the specific application.

Navigation units are critical application for launch vehicles, whose failures may lead to catastrophic consequences. For this reason, thorough analyses have to be carried, paying attention to the smallest detail, in order to be sure to consider all the relevant aspects. This attitude was used during previous analyses, especially during FMEA and FTA, that were completed after a long iterative procedure to take into account all the details relative to a complicated system as a hybrid navigation unit can be. This process requires experience in systems engineering and a wide and deep knowledge of the system to be studied, and it would have been a lot harder without the help of experts in RAMS from SENER Aeroespacial.

The design and implementation process of a complicated and highly integrated system as FDIR has been a challenging process, for the necessity to adapt to a well consolidated and functioning architecture without influencing unit performances. Also, the proper modelling in Simulink of the implemented system has posed many challenges and has been a very intense process.

The Validation and Verification campaign is also one of the most challenging processes of this study. The definition of a Validation plan and specification in order to verify the fulfillment of all user requirements is not an easy task. Furthermore, the high number of simulations executed and the complexity of the system made this process very time-consuming.

6.2 Future Works

Regarding future works, the following paths have been identified as possible future developments of this study:

- **Implementation and tuning of GNSS FDIR in flight prototype software**: GNSS FDIR algorithms, both outlier detection and recovery/removal ones, have been proven to have great performances in detecting faults in GNSS measurements, which, if incorporated in navigation solution computation, i.e. in the Kalman filter, will cause a complete integrity loss of the navigation data. The suitability of these algorithms against jamming and spoofing may be object of study, too.
- **Further investigation and improvement of IMU FDIR** :
  IMU FDIR selected algorithms present some difficulties in outlier detection. Also, effective IMU recovery still represents an open issue. Others detecting and recovery methods could be investigated in a future work focused only in the IMU FDIR. An example of these type of methods could be including the IMU measurements in a Kalman filter to have complete access to IMU's measurement residuals and innovations, at the cost of a re-design of filter architecture, especially observation-related algorithms and matrices.
- **Assessment and implementation of IMU hot redundancy**:
  During this work, it has been observed the criticality of the IMU in a hybrid navigation, within a launcher environment. This, again with the difficulty of detecting the IMU errors, strongly suggests the use of a redundant IMU (with an equal or lower IMU grade). Redundant IMU can give access to physical redundancy, i.e. redundant measurements, that could be used as a backup integrity monitoring solution or that could improve navigation solution estimation. Also, a whole new range of detection algorithms can be implemented taking advantage of IMU redundancy.

Bibliography

[1] SpaceWorks. "Nano-Microsatellite Market Forecast, 8th Edition". In: (2018).

[2] P.D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition*. GNSS/GPS. Artech House, 2013. ISBN: 9781608070053. URL: https://books.google.es/books?id=t94fAgAAQBAJ.

[3] Stephen Steffes. "Development and Analysis of SHEFEX-2 Hybrid Navigation System Experiment". In: (Apr. 2013).

[4] Benjamin Braun, Markus Markgraf, and Oliver Montenbruck. "Performance analysis of IMU-augmented GNSS tracking systems for space launch vehicles". In: *CEAS Space Journal* 8.2 (June 2016), pp. 117–133. ISSN: 1868-2510. DOI: 10.1007/s12567-016-0113-9. URL: https://doi.org/10.1007/s12567-016-0113-9.

[5] Guilherme F. Trigo et al. "Robust Tightly Coupled Hybrid Navigation for Space Transportation". In: *Journal of Spacecraft and Rockets* 56.2 (2019), pp. 596–609. DOI: 10.2514/1.A34232. eprint: https://doi.org/10.2514/1.A34232. URL: https://doi.org/10.2514/1.A34232.

[6] Massimo Tipaldi and Bernhard Bruenjes. "Survey on Fault Detection, Isolation, and Recovery Strategies in the Space Domain". In: *Journal of Aerospace Information Systems* 12.2 (2015), pp. 235–256. DOI: 10.2514/1.I010307. eprint: https://doi.org/10.2514/1.I010307. URL: https://doi.org/10.2514/1.I010307.

[7] Guilherme F. Trigo and Stephan Theil. "Architectural elements of hybrid navigation systems for future space transportation". In: *CEAS Space Journal* 10.2 (June 2018), pp. 231–250. ISSN: 1868-2510. DOI: 10.1007/s12567-017-0187-z. URL: https://doi.org/10.1007/s12567-017-0187-z.

[8] D.H. Titterton, J.L. Weston, and J.L. Weston. *Strapdown Inertial Navigation Technology*. IEE radar, sonar, navigation and avionics series. American Institute of Aeronautics and Astronautics, 2004. ISBN: 9781563476938. URL: https://books.google.es/books?id=UibbAQAACAAJ.

[9] Antonio Angrisano. "GNSS/INS Integration Methods". PhD thesis. Jan. 2010.

[10] George T. Schmidt and Richard E. Phillips. "INS/GPS Integration Architectures". In: 2010.

[11] Mohinder Grewal and Angus Andrews. "Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives". In: *Control Systems, IEEE* 30 (July 2010), pp. 69–78. DOI: 10.1109/MCS.2010.936465.

[12] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006. ISBN: 9780470045336. URL: https://books.google.es/books?id=UiMVoP%5C_7TZkC.

[13] Hugh Durrant-Whyte. *Introduction to Estimation and the Kalman Filter*. Tech. rep. Australian Centre and Field Robotics, 2001.

[14] Greg Welch and Gary Bishop. "An Introduction to the Kalman Filter". In: *Proc. Siggraph Course* 8 (Jan. 2006).

[15] Alexandra Wander and Roger Forstner. "Innovative Fault Detection, Isolation and Recovery Strategies On-Board Spacecraft: State of the Art and Research Challenges". In: 2013.

[16] Xavier Olive. "FDIR for Satellites". In: *Int. J. Appl. Math. Comput. Sci.* 22.1 (Mar. 2012), pp. 99–107. ISSN: 1641-876X. DOI: 10.2478/v10006-012-0007-8. URL: https://doi.org/10.2478/v10006-012-0007-8.

[17] Victoria Hodge. " Survey of Outlier Detection Methodologies". In: *Artificial Intelligence Review* 22 (Oct. 2004), pp. 85–126. DOI: 10.1023/B:AIRE.0000045502.10941.a9.

[18] Kangqing Yu, Wei Shi, and Xiangyu Ma. "Real-time Outlier Detection over Streaming Data". In: July 2019.

[19] Liangchen Chen, Shu Gao, and Cao Xiufeng. "Research on real-time outlier detection over big data streams". In: *International Journal of Computers and Applications* (Oct. 2017), pp. 1–9. DOI: 10.1080/1206212X.2017.1397388.

[20] European Cooperation for Space Standardization. *ECSS-Q-ST-30-02C - Failure mode effects (and criticality) analysis (FMEA/FMECA)*. Mar. 2009.

[21] International Electrotechnical Commission. *IEC 61025 - Fault Tree Analysis*. 2006.

[22] Fredrik Gustafsson. "Adaptive Filtering and Change Detection." In: 2001.

[23] R.J. Patton and J. Chen. "A Review of Parity Space Approaches to Fault Diagnosis". In: *IFAC Proceedings Volumes* 24.6 (1991). IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS'91), Baden-Baden, Germany, 10-13 September 1991, pp. 65–81. ISSN: 1474-6670. DOI: https://doi.org/10.1016/S1474-6670(17)51124-6. URL: http://www.sciencedirect.com/science/article/pii/S1474667017511246.

[24] Kangqing Yu, Wei Shi, and Xiangyu Ma. "Real-time Outlier Detection over Streaming Data". In: July 2019.

