Improving Efficiency and Accuracy of Safety Related Algorithms for Unmanned Aircraft Systems

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy
by

Chinmaya Mishra

December 2016

Abstract

This thesis examines the problem of large computational loads generated by safety related algorithms for Unmanned Aircraft Systems (UAS). Efficient and accurate methods for multiple sensor fault detection and Sense-And-Avoid systems for UAS are proposed.

A novel sensor fault detection method is proposed and tested by simulation. The method detects multiple sensor faults by evaluating normal and faulty hypotheses for each sensor sequentially using measurements obtained from sensors on-board the aircraft. A Six-Degrees-of-Freedom flight model for a Navion aircraft is used to simulate faulty sensor data to test the fault detection method. The proposed sequential fault detection method detects faulty sensors, the update process is fast and maintains a more accurate state-estimate than the parallel fault detection method.

For Sense-And-Avoid systems, an efficient method for estimating the probability of conflict between traffic in a non-cooperative environment is proposed. Estimating low probabilities of conflict using 'naive' Direct Monte Carlo method generates a significant computational load. The proposed method uses a technique called Subset Simulation where small failure probabilities are computed as a product of larger conditional failure probabilities – reducing the computational load whilst improving the accuracy of the probability estimates. The utility of the approach is demonstrated by modelling a series of conflicting and potentially conflicting scenarios based on the standard Rules of the Air specified by the International Civil Aviation Organization.

Contents

List of Figures

List of Tables

Acronyms

UAS	Unmanned Air Systems
UA	Unmanned Aircraft
SAA	Sense-And-Avoid
FDD	Fault Detection and Diagnosis
FDI	Fault Detection and Isolation
SWAP-C	Size Weight and Power - Cost
SixDoF	Six-Degrees-of-Freedom
GPB-1	Generalized Pseudo Bayes-1
NOTAM	NOtice To AirMen
CAP	Civil Aviation Publication
CAA	Civil Aviation Authority
UK	United Kingdom
FAA	Federal Aviation Authority
USA	United States of America
ICAO	International Civil Aviation Organization
VFR	Visual Flight Rules
VMC	Visual Meteorological Conditions
IFR	Instrument Flight Rules
AMSL	Above Mean Sea Level
ATSU	Air Traffic Service Unit

AFIS	Air Flight Information Service
AoA	Angle-of-Attack
SVFR	Special Visual Flight Rules
ELOS	Equivalent Level of Safety
ADS-B	Automatic Dependent Surveillance - Broadcast
TCAS	Traffic Collision Avoidance System
GPS	Global Positioning System
SSR	Secondary Surveillance Radar
COTS	Commercial of-the-shelf
IMU	Inertial Measurement Unit
PS	Pitot-static
ISA	International Standards Atmosphere
KF	Kalman Filter
EKF	Extended Kalman Filter
UT	Unscented Transform
UKF	Unscented Kalman Filter
IMM	Interacting Multiple Model
MMAE	Multiple Model Adaptive Estimation
MH	Metropolis-Hastings
DMC	Direct Monte Carlo
MCMC	Markov Chain Monte Carlo
SS	Subset Simulation
CCDF	Complementary Cumulative Distribution Function
SMC	Sequential Monte Carlo

Acknowledgements

I would like to thank my supervisors Prof. Jason Ralph and Prof. Simon Maskell for their guidance, advice and support throughout the duration of my PhD study. I would like to thank Prof. Siu-Kui Au for his suggestions and insights during the research. I would like to thank Flávio DeMelo and Matteo Fasiolo for their discussions that has helped me understand some of the fundamental concepts which allowed me to progress in methods discussed in this thesis. Special thanks to the post-doctoral researchers, Dr. Elias Griffiths and Dr. Austin Brockmeier for their help with MATLAB programming during the early stages of my PhD study. I would like to thank my viva examiners, Prof. Apostol Vourdas and Dr. Jeyarajan Thiyagalingam for there comments and suggestions that were very helpful in submitting this thesis.

I would like to thank my friends Sebastian Karam and Aaron Dickenson for their continuous encouragement throughout my PhD study. I would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for funding the PhD which has made this thesis possible.

Last but by no means least, I would like to thank my family for their constant support and encouragement over the years.

Chapter 1

Introduction

The operation of Unmanned Aircraft Systems (UAS) is restricted to segregated airspace unless an equivalent level of safety to manned aviation can be demonstrated [8]. Segregated airspace are blocks of airspace that are designated for UAS to operate within [9]. The restriction of UAS to operate within segregated airspace limits potential applications of UAS. Over the last decade numerous Aviation Authorities from around the world have expressed an interest in integrating UAS operations into non-segregated airspace [10]. However, issues related to safety, security and regulations prevent this from occurring [11, 12]. Regulatory bodies have expressed that the safety standards of UAS should be no less demanding than current standards that are applied to manned aircraft. An approach is to satisfy 'equivalence' to existing safety standards that apply to manned aviation. However, conducting manned aviation consists of interacting with various entities, for example, operating in different classes of airspace and communicating with air traffic controllers. At the same time, the operation of UAS should be transparent to other airspace users and Air traffic controllers. To clarify, the UAS operation should not impose any extra service requirements than ones already imposed in manned aviation [8].

The restriction of UAS to operate within segregated airspace has motivated research into and the development of automated safety systems to ensure that UAS can operate in non-segregated airspace safely. Such systems are reliant on algorithms that are computationally demanding to ensure that adequate levels of safety are achieved. This computational demand requires fast hardware that is more expensive and leads to an increase in size, weight and cost of Unmanned Aircraft Systems.

Collision Avoidance has been identified as the most important issue that needs to be addressed since it concerns the safety of the aircraft and other airspace users [11,13]. This is to prevent collision between aircraft and ensure survivability of the platform. The ability to ensure that collisions between the Unmanned Aircraft (UA) and other airspace users are avoided, is an essential safety requirement. Although the lack of a pilot or flight crew on-board the aircraft means that the aircraft is somewhat expendable, collision with other airspace users is unacceptable. In the event of a collision, the secondary effect of falling debris might also lead to ground casualties or (worse) fatalities [14]. Therefore, a collision must be avoided at all cost. Perhaps, the quality for any Unmanned Aircraft is to demonstrate the ability to follow the Rules of the Air mandated by the Aviation Authority. To be specific, all airspace users are expected to follow the 'Right of Way' rules. More importantly, safe separation between aircraft must be maintained even if the Right of Way rules are not strictly followed. The lack of a pilot on-board the aircraft creates the requirement for a 'Sense-And-Avoid' capability. The Sense-And-Avoid (SAA) capability of the UAS consists of two functions - Conflict Detection (CD) and Resolution (R). The conflict detection stage involves determining if a conflict exists using data obtained from sensors. This requires calculating the probability of conflict from the probability distribution of the state of traffic inferred from sensor data. Often the magnitude of this probability is very low and requires significant computational load for this to be estimated. Despite the low magnitude of the probability, it is important that it is estimated accurately since the resolution stage is reliant on this and given the catastrophic outcome of a possible collision – the ability to estimate the probability is essential, as it affects the safe operations of the UAS.

Before capabilities, such as Sense-And-Avoid can be considered, an essential prerequisite for any aircraft (manned or unmanned) is the ability to maintain accurate information defining the current heading, speed and attitude of the aircraft. This is an estimate of the aircraft's state and is inferred from the measurements obtained from sensors on-board the aircraft. Maintaining accurate state information is essential since the aircraft's capabilities such as Navigation, Manoeuvring and Sense-And-Avoid are reliant on this information. False sensor data obtained from faulty sensors can lead to inaccurate state information that causes incorrect actions to be executed by the aircraft and result in a failure to accomplish the mission, or worse - loss of the aircraft and fatalities. For manned aircraft, operating in good visibility under Visual Flight Rules, the problem of faulty sensors might be a mere inconvenience. In such an event, the pilot could land the aircraft at the nearest airfield by looking out of the window. However, in the case of Unmanned Aircraft, the lack of a pilot or flight crew on-board the aircraft means that the operator is reliant on state-estimates deduced from data obtained from sensors. Fault Detection and Diagnosis (FDD) methods are essential to detect faulty data obtained from faulty sensors to preserve the accuracy of the state-estimate. Such methods involve processing data obtained from multiple sensors to determine whether they are faulty or normal. Processing data from multiple sensors increases the computational load associated with maintaining accurate state information in a time-sensitive situation.

This thesis seeks to reduce the computational overheads associated with automated safety systems such as Sense and Avoid and Fault Detection systems by providing efficient algorithms which will lead to reductions in Size, Weight, and Power – and reduced Cost (SWAP-C) of Unmanned Aircraft Systems [15].

1.1 Thesis Structure

The thesis examines the problem of large computational requirements by developing efficient algorithms that address the collision avoidance and multiple sensor fault detection problems. Chapter 2 familiarises the reader with background information related to this area of research and describes the associated problems. Chapter 3 and 4 develop a Six-Degrees-of-Freedom (SixDoF) flight model for an aircraft that represents a generic Unmanned Aircraft. This is used as a test-bed to generate sensor data during common flight modes. Chapter 5 introduces simulations for multiple sensor faults that could occur on-board an aircraft or UAS. An efficient sensor fault detection method is demonstrated during common flight scenarios. Chapter 6 and 7 explore rare-event simulation with applications to collision avoidance. Chapter 6 describes the theory of Subset Simulation. Chapter 7 applies Subset Simulation to estimate the probability of conflict for conflicting and potentially conflicting scenarios based on the Right of Way Rules (defined in Annex 2 - Rules of the Air issued by the International Civil Aviation Organization [16]). Chapter 8 concludes the thesis and suggests possible future work.

1.2 Novel Contributions of the Thesis

- The development of an autopilot for a Six-Degrees-of-Freedom (SixDoF) flight model that can be used as a test-bed during common flight scenarios.
- The development of sensor simulation models on-board the aircraft to provide observations for deducing the state of the aircraft. For example, sensor data from Pitot-static, Angle-of-Attack and Inertial Measurement Unit sensors during faulty and normal sensor operation are generated.
- A new Multiple Sensor Fault detection method using Generalized Pseudo Bayes - 1 (GPB-1) is developed. This method detects faulty sensors by evaluating the hypotheses for each sensor sequentially, which results in a faster and more accurate update method than evaluating the hypotheses in parallel.
- The accuracy and computational load required for estimating low probabilities of conflict between air traffic in non-cooperative scenarios is improved by the application of Subset Simulation. Estimating low probabilities of conflict accurately using Direct Monte Carlo method requires a large number of samples that gener-

ates large computational load. The Subset Simulation method uses a fraction of the samples required by Direct Monte Carlo to achieve the same level of accuracy.

- A benchmarking method based on the Right of Way rules is defined that can be used for future comparison of efficiency and accuracy in estimating the probability of conflict between air traffic.

1.3 Publications

- *Doing the Right Thing: Collision Avoidance for Autonomous Air Vehicles*, C. Mishra, M. Mehta, E. J. Griffith and J. F. Ralph, 2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester 2013

- *Efficient Estimation of probability of conflict between air traffic using Subset Simulation*, Chinmaya Mishra, Simon Maskell, Siu-Kui Au, Jason F.Ralph. Submitted to IEEE Transactions on Aerospace and Electronic Systems.

  http://arxiv.org/pdf/1604.07363.pdf

Chapter 2

Background

Currently, Unmanned Aircraft Systems (UAS) are restricted to operate within segregated airspace unless an adequate level of safety is satisfied. Segregated airspace are temporary blocks of airspace that UAS operate within [8]. This is to protect other aircraft that might be operating within the vicinity from hazards, such as the risk of collision with the UAS. However, restricting UAS operations to segregated airspace limits users in benefiting from the potential applications of UAS such as assisting emergency services, establishing communication between remote locations and monitoring environmental changes [17,18]. It also inconveniences existing manned flight operations. For example, manned flights must modify there usual flight plans and navigate around to ensure that segregated airspace is not breached. These alterations increase the expenditure of flight operations [12]. Manned flights are notified of the regions where and when UAS operations are occurring by NOTAMs (NOtice To AirMen) that are issued by the Aviation Authority governing the country's airspace [19]. Within the United Kingdom, UAS operate under the provision of Civil Aviation Publication (CAP) 722 issued by the Civil Aviation Authority (CAA) [8]. The Air Navigation Order issued by the CAA is followed by manned aviation and takes precedence over any operation [20].

This chapter outlines the issues arising from increased computational load generated by satisfying safety related requirements such as collision avoidance and sensor fault detection for Unmanned Aircraft Systems. This chapter begins by familiarising the reader with the existing Rules and Regulations followed by manned aviation in section 2.1. The problem of satisfying UAS safety 'equivalence' to existing manned aviation is discussed in section 2.2. Collision Avoidance and Sensor Fault detection are essential safety requirements for UAS and are discussed in sections 2.3 and 2.4 respectively.

2.1 Rules and Regulations

The Rules and Regulations for civilian flight date back to the 1920s that began with a series of conventions such as the 1919 Paris Convention [21, 22], the 1926 Iibero-American Convention [23] and the 1926 Havana Convention [24]. The most significant convention is the 1944 Chicago Convention which had 52 signatory states [25]. The convention established the foundation for aviation safety, procedures, rules and regulation with International cooperation that is currently enforced. Since then, the number of signatory states has grown to 191 and is now known as the International Civil Aviation Organization (ICAO) [25]. The Rules and Regulations issued by each state's Aviation Authority, such as the Civil Aviation Authority (CAA) in the United Kingdom (UK) and the Federal Aviation Administration (FAA) in the United States (US), are based on the Rules and Regulations published by the ICAO, which consist of 18 Annexes [26,27]. This section sets out some of the relevant Rules and Regulations that govern General Aviation.

2.1.1 Flight Rules

According to Annex 2 to the Convention on International Civil Aviation - Rules of the Air, a flight is conducted under one of three different flight rules [6,16]. These are:

- Visual Flight Rules (VFR): flights conducted under VFR are reliant on the flight crew's ability to perform tasks such as navigation; and maintain safe separation between traffic and terrain by visual reference. Therefore, VFR flight can only be conducted when the Visual Meteorological Conditions (VMC) minima are met and the weather conditions are suitable for the flight crew to maintain sufficient situational awareness while relying on visual cues only. The VMC minima vary depending on the flight level of the aircraft. For example, while flying at or above 3050m Above Mean Sea Level (AMSL), visibility of 8km must be maintained, below this but above 900m (AMSL) or 300m which ever is higher, flight visibility of 3km needs to be maintained [16].

- Instrument Flight Rules (IFR): flights are conducted when the weather conditions are below the VMC minima – otherwise known as Instrument Meteorological Conditions (IMC) and a VFR flight cannot be conducted. Such weather conditions are known as Instrument Meteorological Conditions (IMC) and the pilot or flight crew are reliant on instruments to navigate – a combination of Collision Avoidance systems and advisories from Air traffic Service Units (ATSU) are used to maintain safe separation [16].

- Special Visual Flight Rules (SVFR): is a provision for VFR flight that needs to access restricted airspace where VFR is prohibited. This is useful for specific scenarios; for example, when an aircraft is operating under VFR and needs access to an airfield that requires transiting through restricted airspace [16].

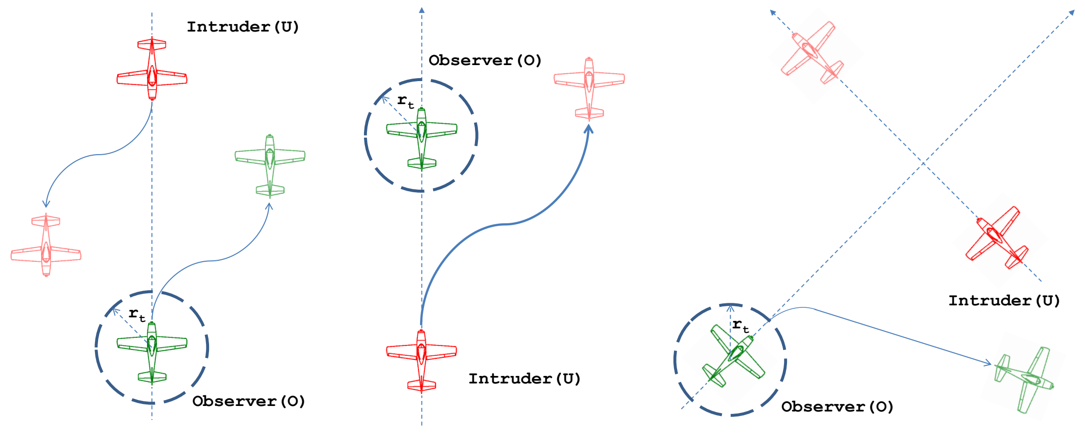
2.1.2 Right of Way

The Right of Way rules stated in Annex 2 are followed by traffic to resolve conflict(s) that might be encountered between aircraft during flight. A conflict exists in a scenario where a collision or loss of minimum separation between aircraft is imminent and no action is taken. The element(s) of traffic that are involved in the conflict need to execute manoeuvres to ensure the conflict is resolved. The type of manoeuvre and the element of traffic that needs to execute it, depends on the geometry of the encounter. The geometry of three different types of conflicts are shown in Figure 2.1 and are defined as:

- **Head-on**: A Head-on conflict exists when an aircraft is approaching another aircraft on a Head-on (or approximately Head-on) collision course. In such a case, each aircraft must alter course to the right to avoid collision as shown in Figure 2.1(a).
- **Overtaking**: An Overtaking conflict occurs when an aircraft is approaching another aircraft from behind as shown in Figure 2.1(b). To be specific, an overtaking condition exists when the overtaking aircraft is approaching the rear of the aircraft to be overtaken within an angle less than 70 degrees from the extended centreline. In such a scenario, the aircraft being overtaken has the Right of Way and should maintain speed and heading as shown in Figure 2.1(b). The overtaking aircraft must alter course right and keep clear of the aircraft to be overtaken.
- **Converging**: A converging conflict occurs when two aircraft are converging and a conflict is imminent as shown in Figure 2.1(c) – the aircraft on the right has the right of way and should maintain speed and heading as shown in the figure. The aircraft on the left must alter its course right to resolve the collision.

Annex 2 also states that regardless of the Right of Way rules above, it is essential that a collision is avoided. The Pilot-in-Command of the respective aircraft is responsible to ensure that a collision does not occur.

A minimum threshold has not been defined explicitly by the ICAO [28–30]. However, Aviation Authorities and Regulatory Bodies implicitly understand that a minimum threshold of 500 ft must be maintained in all directions. For example the ‘Low Flying Rule’ issued by the CAA states that no aircraft should be flown closer than 500 ft to any person, vessel, vehicle or structure except with written permission from the CAA [20]. The FAA defines a separation of 500 ft as ‘well clear’ [31].



(a) Head-on (b) Overtaking (c) Converging

Figure 2.1: These figures illustrate the geometric configuration of the different conflicts that might be encountered within a block of airspace. This includes different manoeuvres required to be executed by the respective parties to resolve the conflict.

2.1.3 Air Traffic Service Unit

Air Traffic Service Units (ATSU) are ground based Units that monitor and communicate with flights with the primary goal of ensuring safe operations. There are different types of Air Traffic Control, each type has different callsigns and responsibilities [6]:

- Air Traffic Control Unit types: Tower, Radar, Approach, Ground, Director, Control - These units can be only operated by licensed air traffic controllers. The call sign describes the area of responsibility. These types of units are authorised to issue instructions or advise pilots.
- Aerodrome Flight Information Service (AFIS): Information - This unit provides flight information service regarding traffic in the vicinity.
- Aerodrome Air/Ground Communication Service: Radio - This type of unit assists pilots by providing information only.

2.1.4 Airspace

The airspace is divided into regions and further subdivided into classes. Annex 11 to the convention on International Civil Aviation – Air Traffic Services defines the different classes of airspace designated A to G in order of most restrictive to least restrictive, where airspaces A to E are controlled airspace and F to G are uncontrolled airspace [32]. The restrictions and limitations of the different classes of airspace are defined in Table 2.1.

Airspace Class	Clearance	IFR/VFR	Separation	Traffic Information
A	required	IFR and SVFR only	between all aircraft	For all aircraft
B	required	IFR/VFR	between all aircraft	For all aircraft
C	required	IFR/VFR	IFR/IFR and IFR/VFR	VFR/VFR
D	required	IFR/VFR	IFR/IFR	IFR - VFR
E	required for IFR only	IFR/IFR	IFR - IFR/SVFR	All aircraft if possible
F	not required	IFR/VFR	not provided	Available for all aircraft if possible
G	not required	IFR/VFR	not provided	Available for all aircraft if possible

Table 2.1: The different classes of airspace and restrictions [6]

It is apparent from Table 2.1 that the different restrictions, limitations and services a flight is subject to, vary according to the class of airspace it is operating within and the flight rules that it operates under. In order for a UAS to be allowed to operate within controlled airspace alongside its manned equivalent, it must satisfy the same safety standards as its manned equivalent. However, applying a single standard of safety to all Unmanned Aircraft is not practical and might be unnecessary [33]. For example, an Unmanned Aircraft used for communications – operating above a densely populated city must satisfy a higher level of safety than an Unmanned Aircraft used for surveying caves in a remote location [11]. There are different types of UAS with varying levels of autonomy, depending on the type of mission, some are more suited than others. A mission might require access to controlled airspace that has a higher level of safety than uncontrolled airspace [34]. Additionally, accessing controlled airspace requires communicating with ATSU. For manned flight, it is the flight crew's responsibility to ensure communication with the appropriate ATSU. However, the absence of a flight crew for an Unmanned Aircraft means it is the remote operator's responsibility to ensure communications during the mission [8]. A sensible approach to satisfy the safety level of a UAS is to follow a ‘Concept of Operation’ (CONops) on a ‘mission-by-mission’ basis where safety analysis can be conducted that considers factors such as the type of aircraft, the class of airspace that needs to be accessed to accomplish the mission, and the responsibilities of the operator [13, 35, 36].

2.2 Equivalent Level of Safety

The operation of UAS flights needs to be treated with the same level of importance as current manned flight [28]. The Aviation Authorities state that interaction with a UAS should be no less demanding than its manned equivalent. A proposed method is to derive an ‘Equivalent Level Of Safety’ (ELOS) to existing manned standards [10] [28]. Dalamagidkis et al. approach the problem of deriving system requirements to be able to meet the ELOS for Unmanned Aircraft Systems by proposing safety risk models [10]. A safety risk model is used for estimating fatalities that could occur due to accidents involving Unmanned Aircraft Systems. This is approached by evaluating the current level of safety of manned aviation using statistics of fatalities that have occurred due to manned air accidents between 1983 and 2006, obtained from the National Transport Safety Board (NTSB) [12, 37]. A risk model for estimating the fatalities after accidents due Ground Impact and mid-air Collisions are derived based on fatalities that have occurred in the past due to manned aviation [38]. The model considers factors such as the population density of area affected by the accident, the size, mass, orientation of the aircraft and kinetic energy dissipated on impact is also considered. In some cases, the estimated fatality rate is mentioned as over conservative for UAS since it is derived using statistics of casualties and fatalities of mid-air collision between manned aircraft.

Zeitlin and Larcher have expressed the problem of demonstrating ELOS for UAS Collision Avoidance [13]. The problem exists in establishing a baseline performance of Collision Avoidance for manned flight that is required to demonstrate ELOS. The performance is dependent on the pilot or flight crew’s ability to visually acquire the hazard (traffic or obstacle) and react in time to resolve the conflict by selecting the appropriate manoeuvre. Although it is possible to estimate the human ability to visually acquire hazards such as traffic – estimating other factors such as reaction time, the ability to select the appropriate resolution manoeuvre and the ability to adapt to the changing situation is difficult [13]. Furthermore, the collision risk in manned aviation varies depending on the class of airspace it is operating within. For example, operating in controlled airspace has low risk of collision than operating in uncontrolled airspace [13].

2.3 Sense and Avoid

Collision Avoidance has been identified as a critical requirement to enable the integration of UAS operations in non-segregated airspace [13, 39]. Aviation in controlled airspace involves following a combination of procedures and communicating with Air Traffic Control as outlined by respective Aviation Authorities to ensure safety [40]. For manned aviation, it is the pilot's responsibility to maintain safe separation and avoid collision with traffic [16]. The same applies in the case of Unmanned Aircraft – it is the responsibility of the remote operator to maintain situational awareness of the aircraft and the vicinity it is operating within to avoid any collision [41]. The lack of pilot or flight crew on-board the Unmanned Aircraft requires the operator to rely on sensors to be aware of its own aircraft and the immediate environment, which could contain hazards such as birds, terrain, traffic and other obstacles. For such capability the term 'Sense-And-Avoid' (SAA) is more appropriate than 'See-And-Avoid'. The SAA capability for UAS is composed of two main functions [33]. Firstly, 'Separation Assurance' – this is a pre-emptive measure to reduce probability of collision by following procedures such as maintaining minimum separation as instructed by ATSU. Usually, at this stage the aircraft will make small corrections to its course that result in gentle manoeuvres to ensure that the required separation minima are maintained and the aircraft remains 'well clear' of traffic and hazards [20]. Secondly, 'Collision Avoidance' – this is the last level of protection where aggressive manoeuvres might be required to ensure that a collision is prevented. The Collision Avoidance stage consists of two functions – Conflict Detection (CD) and Resolution (R). The Conflict Detection phase consists of detecting traffic, terrain or obstacles. In the case of detected traffic, it needs to be tracked accurately. This means determining that the detection is valid according to current state-space (position and trajectory). The projected trajectory of the tracked object must be evaluated to determine if it conflicts with the Unmanned Aircraft [33]. If a conflict exists, the Resolution (R) stage involves executing the appropriate manoeuvre to resolve the conflict. The appropriate manoeuvre is one that the Unmanned Aircraft is capable of executing and is compliant with the Right of Way Rules according to the geometry of the encounter. More importantly, the resolution manoeuvre must prevent a collision [33].

Initial work on CD&R in the early 1980's can be found in robotics where the collision avoidance problem has been treated as a path planning task [42] and an early approach to the problem involved using artificial potential fields [43]. Such methods are suitable for scenarios where movement of the vehicles may be relatively slow, restricted in space or in scope. However, over the following decades, the increased use of UAS has created demand for autonomous CD&R solutions which are more suitable for the dynamic aerospace environment. A large number of CD&R methods have been proposed during this period and comprehensive surveys have been conducted by Kuchar and Yang [44], Krozel et al. [45], Warren [46] and Zeghal [47]. Kuchar and Yang have proposed a taxonomy of methods useful in identifying gaps and directing future efforts within the SAA community [44]. More recently, Albaker and Rahim have presented an up to date survey of CD&R methods for UAS [48]. The CD&R methods can be categorised into two main groups known as Cooperative and Non-Cooperative CD&R. The following sub-sections give a brief overview of Cooperative and Non-cooperative CD&R.

2.3.1 Cooperative CD&R

Cooperative CD&R methods require coordination between traffic to resolve the conflict. For example, following the Right of Way rules is a form of cooperative resolution. In order to ensure successful conflict resolution, the resolution manoeuvre needs to be coordinated with the conflicting traffic. This requires sharing information such as the current altitude, speed and heading of the aircraft by communication via radio or by using cooperative equipment such as Automatic Dependent Surveillance - Broadcast (ADS-B), Transponders and Traffic Collision Avoidance System (TCAS). This information is known as the state of the aircraft. The intended altitude, speed and heading is also shared with traffic. The traffic is expected to share similar information to ensure coordination.

Automatic Dependent Surveillance - Broadcast

This is a cooperative surveillance technology that consists of equipment on-board an aircraft that broadcasts the current state of its own aircraft (position, velocity and heading). The state information is obtained from a Navigation system on-board the aircraft, such as GPS [49].

Transponders

Some aircraft carry transponders that respond to interrogation from ground based units such as ATSU and Secondary Surveillance Radar (SSR). Airborne equipment is reliant on transponders carried by traffic to be aware of the traffic operating in the vicinity and resolve potential conflicts. An example of airborne equipment is Traffic Collision Avoidance System (TCAS).

Traffic Collision Avoidance System

Traffic Collision Avoidance System (TCAS) is a Collision Avoidance system that was first deployed on-board commercial aircraft in the US during the 1990s [50–52]. Currently, the system is mandated by the ICAO to be carried by all aircraft that exceed a maximum take-off weight of 5700 kg or carrying more than 19 passengers. The TCAS system issues Traffic Advisories (TA) when traffic is detected and is projected to breach the aircraft's protected zone. Resolution Advisories (RA) are issued to the pilot or flight crew by TCAS to resolve the conflicts.

The TCAS solution is infeasible for UAS for number of reasons:

- TCAS operates under the assumption that traffic encountered is capable of achieving climb rates of 1500 ft/min to 2500 ft/min. However, not all Unmanned Aircraft platforms can achieve this sort of performance [53].
- TCAS is reliant on all aircraft carrying transponders to be detected. Not all traffic are equipped with transponders.

In a situation where a manned flight operating in uncontrolled airspace (F and G) in VFR without traffic alerting service and without TCAS, is reliant on the pilot looking out of the window to See-And-Avoid any hazards or possible conflicts. In the case of an Unmanned Aircraft, the lack of pilot or flight crew means that the operator is reliant on sensors on-board the aircraft to satisfy the See-And-Avoid requirement. For such cases, non-cooperative methods need to be used to satisfy CD&R.

2.3.2 Non-Cooperative CD&R

Non-Cooperative CD&R assumes that no information related to the current state or future intent has been shared between aircraft (i.e. there is no flight plan exchange or radio/data link). This is a far more challenging problem since information related to traffic state and intentions must be measured or inferred from the behaviour of non-cooperative aircraft. Normally, this will be due to the lack of appropriate technology on-board the aircraft: for example, a lightweight commercial of-the-shelf (COTS) UAS, obtained by the general public and used for recreational purposes. Problems occur when these aircraft are operated within non-segregated airspace. This type of airspace contains aircraft (manned or unmanned) that adhere to the Rules of the Air and expect traffic to do so as well. The lack of cooperative technology on-board a lightweight UAS prevents awareness of traffic and increases the risk of a mid-air collision. This problem needs to be addressed due to the increased number of near miss incidents involving such UAS operating within non-segregated airspace [54]. The problem of the lack of information is addressed by using on-board sensors. Information related to state of traffic is obtained from observations using sensors such as Radar, Lidar and/or cameras. For example, Mcfadyen et al. have considered using visual predictive control with a spherical camera model to create a collision avoidance controller [55]. Recently, Huh et al. have proposed a vision based Sense-and-Avoid framework that utilizes a camera to detect and avoid approaching airborne intruders [56]. A collision avoidance system that uses a combination of Radar and electro-optical sensors have been prototyped and tested by Accardo et al [57]. Measurement data obtained from sensors are inherently noisy. This gives rise to uncertainties in the observed state and predicted motion of the non-cooperative aircraft. In an environment where future trajectories are uncertain, the likelihood of a conflict is an essential metric. Obtaining an accurate estimate for the Probability of Conflict (P_c), given the sensor data, is a key parameter required to resolve traffic conflicts.

Probabilistic methods for conflict resolution that require the calculation of metrics like the probability of conflict (P_c) have been discussed in [44]. Nordlund and Gustafsson [58] noted the huge number of simulations required to get sufficient reliability for small risks, and suggested an approach that reduced the three dimensional problem to a one dimensional integral along piecewise straight paths [59,60]. More recently, Jilkov et al. have extended a method developed by Blom and Bakker [61] and estimated P_c using multiple models for aircraft trajectory prediction [62]. Many probabilistic CD&R methods use Monte Carlo methods where uncertainties exist [52,62–68]. Unfortunately, for scenarios where the expected P_c is low, a Monte Carlo method will require a very large number of samples to estimate P_c with any accuracy. The large number of samples generate an increased computational load. Despite the low probability, the catastrophic outcome of a collision makes it essential to estimate the probability of conflict accurately. The computational demand requires the computation capacity to be large, which results in increased size, weight and power requirements. This might not be feasible due to limited resources and size, weight and power restrictions. Chapters 6 and 7 focus on the problem of estimating low probability of conflict between Air traffic efficiently and accurately.

2.4 Fault Detection

As mentioned previously, accurate awareness of the state of the aircraft is an important requirement for safe operation of any aircraft. Systems on-board the aircraft that handle crucial functions such as Collision Avoidance, Navigation and Flight Controls are reliant on accurate state estimation. The state of the aircraft is estimated by fusing data obtained from multiple sensors on-board the aircraft. The accuracy of the data is dependent on the integrity of the sensors. If false data are obtained from faulty sensor then the state-estimate is corrupted and that misleads the aircraft and the flight crew.

Fatalities due to faulty sensor readings have occurred in the recent past on commercial flights such as the Air France (AF447) and XL Airways Germany Flight 888T accidents. On the 1st of June 2009, Air France Flight AF447 departed from Rio de Janeiro, Brazil and was en route to Paris, France. The aircraft crashed into the Atlantic Ocean when it was unable to recover from an aerodynamic stall. The investigation of the flight AF447 accident by Bureau d'Enquêtes et d'Analyses pour la sécurité de láviation civile (BEA) concluded that inconsistent airspeed and altitude readings due to the suspected formation of ice crystals in the pitot tubes caused the autopilot to disconnect and reconfigure to an alternate law [69]. This faulty data obtained from the sensors contributed to misleading the flight crew and caused incorrect reactions [4]. This is an example of an instance where false state awareness has misled the flight crew and caused an accident. On 27th of November 2008, XL Airways Germany Flight 888T departed from Perpignan-Rivesaltes Airport in France, on an acceptance flight test, prior to the aircraft being handed back to Air New Zealand [70]. Two hours later, the aircraft crashed into the Mediterranean sea. The final report by BEA concluded that one of the factors contributing to the accident was improper maintenance procedures, which caused water to enter the Angle-of-Attack (AoA) sensors [70]. The water froze during the flight and caused the Angle-of-Attack sensor to output incorrect AoA data. The accident occurred when the flight crew improvised a demonstration of the Angle-of-Attack protections. However, the frozen AoA sensor made it impossible to trigger the protection and this led to improper handling by the flight crew and resulted in a crash.

A sensor fault detection capability is required for all aircraft to maintain state awareness. This capability is essential in the case of Unmanned Aircraft, where there is no pilot or flight crew on-board the aircraft. The operator is reliant on sensors to determine the state of the aircraft. The typical output of a fault detection system is a binary indication – either the sensor has failed or it is operating normally [71]. Numerous fault detection methods have been proposed with various different modelling methods [72, 73]. The two main groups of Fault Detection and Isolation (FDI) methods can be categorised as Hardware Redundancy and Analytical Redundancy [72, 73]. Hardware Redundancy is where multiple sensors measure and compare the same type of signal to deduce if a fault exists. The difference between the measurements is known as a residual [74]. For sensors operating normally, the magnitude of the residual is very small. The system is deigned so that a fault will generate a significant residual. A residual above a defined threshold is detected as a fault. Hardware Redundancy is a common approach for fault detection and is currently in practice within industry [73]. For instance, Boeing uses a voting scheme on its aircraft, where multiple measurements obtained from the same type of sensor are compared and erroneous ones are omitted [75]. This sort of voting scheme operates under the assumption that the majority of signals that are similar represent the truth and any signal that is significantly different to the majority is the result of a failure and must be discarded [76]. However, this is not a robust approach and catastrophic failures have occurred [75]. For example, in the event that all sensors exhibit similar faults, the residual will still remain as zero despite a fault having occurred – a false negative. Also, multiple sensors lead to an increase in weight and cost, this might not be feasible due to payload restrictions [72, 77]. For such cases, Analytical Redundancy is required where the residual is generated by comparing the sensor measurement against the estimated sensor measurement obtained from a mathematical model of the system [78]. The mathematical model is typically derived from first principles, such as Newton's Laws of motion for mechanical systems [72].

A popular approach to fault detection using analytical redundancy involves running a bank of models, where the different models match particular behaviour of the system. In other words, each model is a hypothesis that represents the behaviour of the system in normal or faulty mode. For example, a system with one sensor has a bank with 2 models – one representing the hypothesis that the sensor is operating normally and the other representing the hypothesis that the sensor is faulty. The bank of models are run in parallel and the estimated measurement generated from each model is compared with the actual measurement obtained from the sensor to generate a residual [79]. This residual is used to determine the probability that the hypotheses are true. The probability of the hypotheses is used to deduce the state of the system (normal or faulty). This approach is an adaptation of Multiple Models, an approach that is popular in the Target Tracking community [79].

A combination of both Hardware and Analytical redundancy methods needs to be used for aircraft. This is because an aircraft requires many different types of sensors, because each type of sensor provides a different measurement needed to deduce an element of the state information for the aircraft. For example, the pressure measurements obtained from a Pitot-static system are used to deduce the barometric altitude and speed of the aircraft [80] – but this sensor cannot be used to deduce the orientation of the aircraft. Instead, the rotational rate measurements obtained from the Inertial Measurement System (IMU) are used to deduce the orientation of the aircraft. The different types of sensor commonly found on aircraft and the measurements they provide are discussed later in Chapter 4. Hardware redundancy is often required to meet safety requirements outlined by Aviation Authority to ensure that the level of acceptable probability of failure is met. For example, the safety analysis of the Electronic Flight Instrument System (EFIS) conducted in [80], consists of a fault tree analysis of the EFIS that shows a failure rate 10^{-12} per flight hour by including multiple hardware redundancies for instruments. This meets the acceptable probability of failure rate stated by the FAA of 10^{-9} per flight hour [81]. However, this increase in the number of sensors leads to an increase in the hypotheses for normal and faulty sensors. The increase in hypotheses results in an increased number of models that need to be evaluated, which causes an increase in the computational load.

2.5 Summary

This chapter has familiarised the reader with some of the existing rules and regulations followed by manned aircraft. Such rules and regulations are expected to be observed by Unmanned Aircraft in order to share the same airspace as manned flight. Collision Avoidance and Sensor Fault Detection capabilities are essential safety requirements for the safe operation of Unmanned Aircraft Systems alongside manned flight. Non-cooperative Collision Avoidance requires calculating the probability of a conflict using Monte Carlo methods. For scenarios where the probability of conflict is low, a large number of samples are required to estimate the probability of conflict accurately. Sensor Fault Detection is an essential requirement since core UAS functionalities are reliant on sensor data. Multiple sensors on-board the aircraft result in high computational load for sensor fault detection.

Chapter 3

Navion Flight Model

A Six-Degrees-of-Freedom flight model for an aircraft is required to represent an Unmanned Aircraft that can be used to generate flight data for testing collision avoidance and fault detection algorithms. A flight model for a Navion aircraft defined in [7] is suitable since it is a standard flight model and can be used to compare future algorithms for benchmarking purposes. This flight model has been used in the past for various research related applications such as estimating the state of the aircraft by visual cues [82] and testing aircraft flight control systems [83].

The flight data generated by simulation allows the capabilities of the aircraft such as turn rates and climb rates during common flight modes to be considered. The most common flight modes that aircraft spend time operating within are in level flight, climbing, descending and turning. A basic autopilot system will allow the operator to specify a heading and altitude to be maintained.

This chapter presents a standard flight model for a Navion aircraft with an autopilot that represents the Unmanned Aircraft to be studied in later chapters. A series of test scenarios are simulated using the autopilot.

3.1 Axes Systems

Before a flight model is derived, the axes systems needs to be established to define the state of the aircraft. The three types of right handed axes systems commonly used are Earth-axes, Body-axes and Wind-axes [84].

- Earth-axes: Figure 3.1 shows the Earth-axes as the fixed frame of reference where O_e is the origin of the axis, X_e points North and is orthogonal to Y_e which points East. The Z_e axis points Down towards the centre of the earth and is aligned with the gravity vector [85]. The Z_e axis is orthogonal to the X_eY_e plane. The X_eY_e horizontal plane is tangential to the surface of the earth, assuming a *flat Earth* non-rotating approximation [86]. This approximation is suitable for defining the aircraft's trajectory that is operating locally within distances of 20 km [85,87]. In the case of vehicles operating over longer distances, more complex axes systems, such as spherical Earth-Axes with the rotating Earth assumption, need to be considered. For example, the trajectory of a long rage missile or the orbital path of a satellite are cases where the curvature and rotation of the earth needs to be considered [88]. However, the flat earth approximation is suitable for simulating low speed aircraft, such as the Navion aircraft. All simulations considered within this thesis assume non-rotating flat Earth-axes since the Navion aircraft typically flies at 150 knots (77.2ms^{-1}) and the period of the simulation does not exceed 120 seconds. This results in a maximum distance of approximately 9km, which is suitable for non-rotating flat Earth-axes.

- Body-axes: The Forces and Moments experienced by the aircraft are calculated in Body-axes. Figure 3.1 show the Body-axes fixed to the body of the aircraft where the origin O_b is located at the aircraft's centre of gravity, the X_b axis extends forward along the aircraft centre line and is known as the longitudinal axis. This is orthogonal to the Y_b axis, which extends along the right wing and is known as the lateral axis [88]. The Z_b axis extends Down and is orthogonal to the X_bY_b plane.

- Wind-axes: The aerodynamic forces and moments exerted on the aircraft are defined in the Wind-axes. This system of axes is included to allow for motion in a non-stationary air mass. The Wind-axes are oriented with respect to the aircraft Body-axes. The X_w axis points in the direction of the aircraft true speed V_a. The Y_w is orthogonal to the X_w axis. The Z_w is orthogonal to the X_wY_w plane.

Figure 3.1 illustrates the aircraft's Body-axes system and the Earth-axes system. The Body-axes are used to define the Forces and Moments experienced by the aircraft. The origin of the aircraft axis system defines the orientation and position of the aircraft with respect to the fixed earth co-ordinates.

3.2 Equations of Motion

The equations of motion for a Six-Degrees-of-Freedom (SixDoF) aircraft are derived from Newton's Second law of motion and stated as: *The summation of all external forces acting on a body is equal to the time rate of change of the momentum of the body; and the summation of all external moments acting on the body is equal to the time rate of change of the moment of momentum (angular momentum)* [7]. The summation of

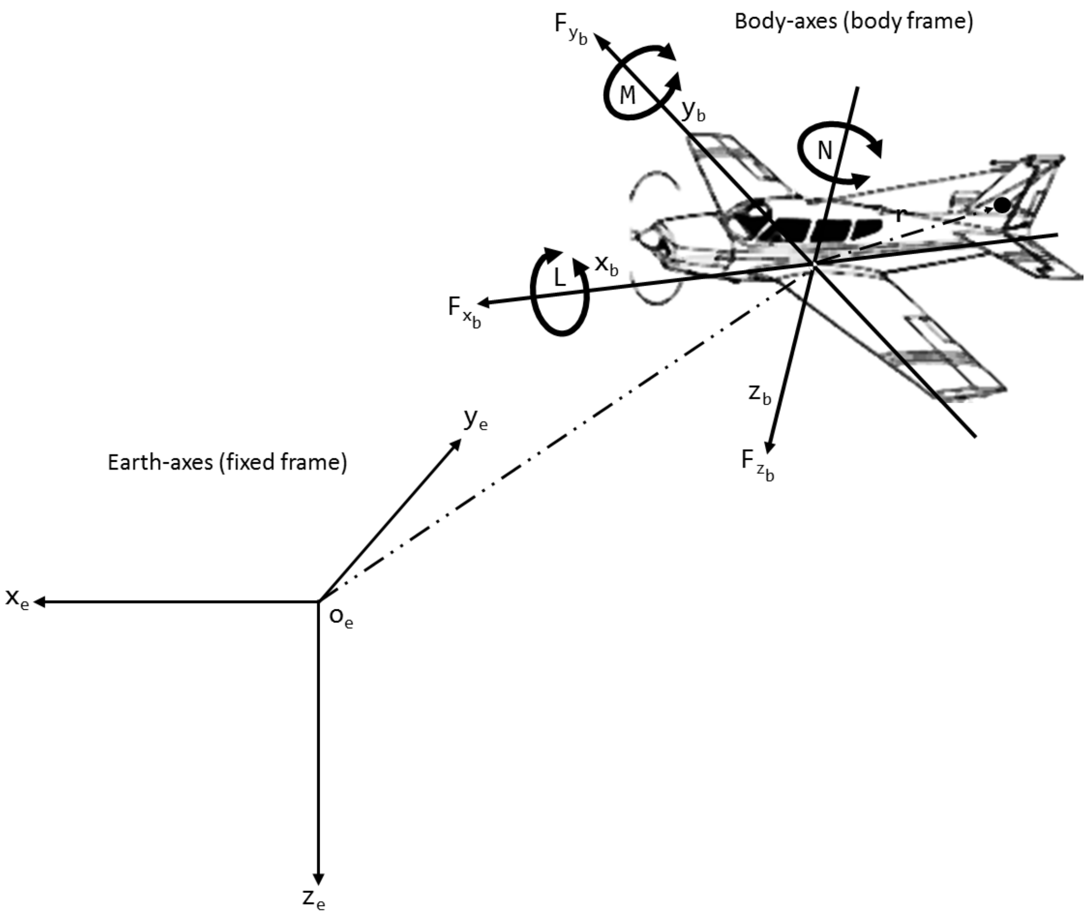


Figure 3.1: The Earth-axes and aircraft Body-axes

forces $\mathbf{F}$ acting on a body with mass m due to the change in momentum is expressed in vector form as

$$\mathbf{F} = \frac{d}{dt}(m\mathbf{V}) \tag{3.1}$$

where $\mathbf{V}$ is the velocity of the body with respect to the Body-axes. The force and velocity are vector quantities and are expressed as $\mathbf{F} = [F_{X_b}, F_{Y_b}, F_{Z_b}]$ and $\mathbf{V} = [u, v, w]$ respectively in the Body-axes, where $F_{X_b}, F_{Y_b}, F_{Z_b}$ and u, v, w are forces and speeds acting along the X_b, Y_b, Z_b directions respectively. Consider an element of mass δm that has a position vector $\mathbf{r}$ from the centre of mass in the aircraft and $\mathbf{V}$ is the velocity of δm relative to the inertial frame of reference. Then, the force acting on the element of mass is $\delta\mathbf{F}$. This is expressed as

$$\delta\mathbf{F} = \delta m\frac{d\mathbf{V}}{dt} \tag{3.2}$$

The velocity of the element of mass is

$$\mathbf{V} = \mathbf{V}_c + \frac{d\mathbf{r}}{dt} = \mathbf{V}_c + \boldsymbol{\omega} \times \mathbf{r} \tag{3.3}$$

where $\mathbf{V}_c$ is the velocity of the centre of mass, $\frac{d\mathbf{r}}{dt}$ is the velocity of the element of mass relative to the centre of mass and $\boldsymbol{\omega}$ is the angular velocity about the centre of mass. The position vector $\mathbf{r}$ and angular velocity $\boldsymbol{\omega}$ are expressed as

$$\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{3.4}$$

$$\boldsymbol{\omega} = p\mathbf{i} + q\mathbf{j} + r\mathbf{k} \tag{3.5}$$

The summation of the forces acting on the total mass is total force experienced by the body. This is expressed as

$$\mathbf{F} = \sum \delta\mathbf{F} = \sum \frac{d}{dt}\left(\mathbf{V}_c + \frac{d\mathbf{r}}{dt}\right)\delta m \tag{3.6}$$

$$\mathbf{F} = \frac{d}{dt}\sum \mathbf{V}_c \delta m + \frac{d\mathbf{r}}{dt}\sum \frac{d}{dt}\delta m \tag{3.7}$$

assuming the total mass remains constant the above can be rewritten as

$$\mathbf{F} = \frac{d\mathbf{V}_c}{dt}m + \frac{d^2}{dt^2}\sum \mathbf{r}\ \delta m \tag{3.8}$$

Since $\mathbf{r}$ is measured from the centre of mass, the summation $\sum \mathbf{r}\ \delta m = 0$. This simplifies to

$$\mathbf{F} = \frac{d\mathbf{V}_c}{dt}m \tag{3.9}$$

The derivative of an arbitrary vector $\mathbf{A}$ referred to a rotating body frame having an angular velocity $\boldsymbol{\omega}$ can be represented by the following vector identity [7]:

$$\left.\frac{d\mathbf{A}}{dt}\right|_I = \left.\frac{d\mathbf{A}}{dt}\right|_B + \boldsymbol{\omega} \times \mathbf{A} \tag{3.10}$$

This identity is applied to equation (3.1)

$$\mathbf{F} = m\left(\left.\frac{d\mathbf{V}_c}{dt}\right|_B + \boldsymbol{\omega} \times \mathbf{V}_c\right) \tag{3.11}$$

where $\boldsymbol{\omega} \times \mathbf{V}_c$ is given by

$$\boldsymbol{\omega} \times \mathbf{V}_c = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p & q & r \\ u & v & w \end{vmatrix} = \mathbf{i}(qw - rv) + \mathbf{j}(ru - pw) + \mathbf{k}(pv - qu) \tag{3.12}$$

$$\mathbf{F} = m\Big[\mathbf{i}(\dot{u} + qw - rv) + \mathbf{j}(\dot{v} + ru - pw) + \mathbf{k}(\dot{w} + pv - qu)\Big] \tag{3.13}$$

The equations for the individual components are

$$F_{x_b} = m(\dot{u} + qw - rv) \tag{3.14}$$

$$F_{y_b} = m(\dot{v} + ru - pw) \tag{3.15}$$

$$F_{z_b} = m(\dot{w} + pv - qu) \tag{3.16}$$

Similarly, the total moment of the body $\mathbf{M}$ is the time rate of change of the moment of momentum $\mathbf{H}$. This is expressed as

$$\mathbf{M} = \frac{d}{dt}\mathbf{H} \tag{3.17}$$

where $\mathbf{M}$ is the total moment of the body and $\mathbf{H}$ is the moment of momentum. Both are vector quantities and are expressed in the Body-axes; $\mathbf{M} = [L, M, N]^{\mathrm{T}}$ where L is the rolling moment about the X_b axis, M is the pitching moment about the Y_b axis and N is the yaw moment about the Z_b axis in the Body-axes. The moment of momentum $\mathbf{H} = [H_X, H_Y, H_Z]^{\mathrm{T}}$ where H_X, H_Y, H_Z are components of $\mathbf{H}$ about the X_b, Y_b, Z_b axis respectively.

The equations for the individual components are

$$L = \frac{d}{dt}H_X \tag{3.18}$$
$$M = \frac{d}{dt}H_Y \tag{3.19}$$
$$N = \frac{d}{dt}H_Z \tag{3.20}$$

The moment $\delta\mathbf{M}$ of an element of mass δm is expressed as

$$\delta\mathbf{M} = \frac{d}{dt}\delta\mathbf{H} = \frac{d}{dt}(\mathbf{r} \times \mathbf{V})\delta m \tag{3.21}$$

The velocity of the mass element can be expressed in terms of the velocity of the centre of mass and the relative velocity of the mass element to the centre of mass

$$\mathbf{V} = \mathbf{V}_c + \boldsymbol{\omega} \times \mathbf{r} \tag{3.22}$$

where $\boldsymbol{\omega}$ is the angular velocity of the vehicle and $\mathbf{r}$ is the position of the element of the mass relative to the centre of mass. The total moment of momentum can be written as

$$\mathbf{H} = \sum \delta\mathbf{H} = \sum(\mathbf{r} \times \mathbf{V}_c)\delta m + \sum[\mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r})]\delta m \tag{3.23}$$

The velocity is constant with respect to the summation and can be taken outside

$$\mathbf{H} = \mathbf{V}_c \sum \mathbf{r}\delta m + \sum [\mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r})]\delta m \tag{3.24}$$

Since $\sum \mathbf{r}\delta m = 0$ the expression simplifies to

$$\mathbf{H} = \sum [\mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r})]\delta m \tag{3.25}$$

$$\boldsymbol{\omega} \times \mathbf{r} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p & q & r \\ x & y & z \end{vmatrix} = \mathbf{i}(qz - ry) + \mathbf{j}(rx - pz) + \mathbf{k}(py - qx) \tag{3.26}$$

$$\mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r}) = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x & y & z \\ (qz - ry) & (rx - pz) & (py - qx) \end{vmatrix} \tag{3.27}$$

$$= \mathbf{i}[y(py - qx) - z(rx - pz)] \tag{3.28}$$

$$+ \mathbf{j}[z(qz - ry) - x(py - qx)] \tag{3.29}$$

$$+ \mathbf{k}[x(rx - pz) - y(qz - ry)] \tag{3.30}$$

$$\mathbf{H} = \sum \mathbf{i}[y(py-qx)-z(rx-pz)]+\mathbf{j}[z(qz-ry)-x(py-qx)]+\mathbf{k}[x(rx-pz)-y(qz-ry)]\delta m \tag{3.31}$$

The equations for the individual components expressed in the Body-axes are

$$\begin{aligned} H_x &= \sum [y(py - qx) - z(rx - pz)]\delta m \\ &= \sum (py^2 - qxy - zrx + pz^2)\delta m \\ &= \sum [p(y^2 + z^2) - xyq - xzr]\delta m \end{aligned} \tag{3.32}$$

$$\begin{aligned} H_y &= \sum [z(qz - ry) - x(py - qx)]\delta m \\ &= \sum [qz^2 - ryz - pyx + qx^2)]\delta m \\ &= \sum [q(x^2 + z^2) - yzr - yxp]\delta m \end{aligned} \tag{3.33}$$

$$\begin{aligned} H_z =& \sum [x(rx - pz) - y(qz - ry)]\delta m \\ =& \sum [rx^2 - pzx - yqz + ry^2]\delta m \\ =& \sum [r(x^2 + y^2) - zxp - zyq]\delta m \end{aligned} \tag{3.34}$$

The summations in the above equations are the mass moments and products of inertia of the aeroplane that are defined by the following expressions [7].

$$I_x = \int\int\int (y^2 + z^2)\ \delta m \tag{3.35}$$

$$I_y = \int\int\int (x^2 + z^2)\ \delta m \tag{3.36}$$

$$I_z = \int\int\int (x^2 + y^2)\ \delta m \tag{3.37}$$

$$I_{xy} = \int\int\int xy\ \delta m \tag{3.38}$$

$$I_{xz} = \int\int\int xz\ \delta m \tag{3.39}$$

$$I_{yz} = \int\int\int yz\ \delta m \tag{3.40}$$

The expressions are applied to equations (3.33), (3.34) and (3.35).

$$H_x = p\sum (y^2 + z^2)\ \delta m - q\sum xy\ \delta m - r\sum xz\ \delta m = pI_x - qI_{xy} - rI_{xz} \tag{3.41}$$

$$H_y = q\sum (x^2 + z^2)\ \delta m - r\sum yz\ \delta m - p\sum xy\ \delta m = qI_y - rI_{yz} - pI_{xy} \tag{3.42}$$

$$H_z = r\sum (x^2 + y^2)\ \delta m - p\sum xz\ \delta m - q\sum yz\ \delta m = rI_z - pI_{xz} - qI_{yz} \tag{3.43}$$

Expression (3.10) is applied to equation (3.21) the angular moments

$$\mathbf{M} = \left.\frac{d\mathbf{H}}{dt}\right|_B + \boldsymbol{\omega} \times \mathbf{H} \tag{3.44}$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} + \boldsymbol{\omega} \times \mathbf{H} = \begin{bmatrix} \dot{H}_x \\ \dot{H}_y \\ \dot{H}_z \end{bmatrix} + \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p & q & r \\ H_x & H_y & H_z \end{vmatrix} \tag{3.45}$$

where $\boldsymbol{\omega} \times \mathbf{H}$ is given by

$$\begin{aligned}\boldsymbol{\omega} \times \mathbf{H} &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p & q & r \\ H_x & H_y & H_z \end{vmatrix} \\ &= \mathbf{i}qH_z - rH_y \\ &+ \mathbf{j}rH_x - pH_z \\ &+ \mathbf{k}pH_y - qH_x \end{aligned}$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} \dot{H}_x \\ \dot{H}_y \\ \dot{H}_z \end{bmatrix} + \begin{bmatrix} qH_z - rH_y \\ rH_x - pH_z \\ pH_y - qH_x \end{bmatrix} \tag{3.46}$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} \dot{p}I_x - \dot{q}I_{xy} - \dot{r}I_{xz} \\ \dot{q}I_y - \dot{r}I_{yz} - \dot{p}I_{xy} \\ \dot{r}I_z - \dot{p}I_{xz} - \dot{q}I_{yz} \end{bmatrix} + \begin{bmatrix} q(rI_z - pI_{xz} - qI_{yz}) - r(qI_y - rI_{yz} - pI_{xy}) \\ r(pI_x - qI_{xy} - rI_{xz}) - p(rI_z - pI_{xz} - qI_{yz}) \\ p(qI_y - rI_{yz} - pI_{xy}) - q(pI_x - qI_{xy} - rI_{xz}) \end{bmatrix} \tag{3.47}$$

Assuming the aircraft is symmetrical in the X_bZ_b plane, the moments of Inertia on both sides are equal, therefore $I_{yz} = I_{xy} = 0$. The moment equations reduces to

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} \dot{p}I_x - \dot{r}I_{xz} \\ \dot{q}I_y \\ \dot{r}I_z - \dot{p}I_{xz} \end{bmatrix} + \begin{bmatrix} q(rI_z - pI_{xz}) - rqI_y \\ r(pI_x - rI_{xz}) - p(rI_z - pI_{xz}) \\ pqI_y - q(pI_x - rI_{xz}) \end{bmatrix} \tag{3.48}$$

$$L = \dot{p}I_x - \dot{r}I_{xz} + qr(I_z - I_y) - qpI_{xz} \tag{3.49}$$

$$M = \dot{q}I_y + rp(I_x - I_z) + I_{xz}(p^2 - r^2) \tag{3.50}$$

$$N = \dot{r}I_z - \dot{p}I_{xz} + pq(I_y - I_x) + qrI_{xz} \tag{3.51}$$

Another assumption is that the aircraft is symmetrical in X_bY_b plane, in practice this might not be the case but the difference in inertia on both side of the plane is small enough to be negligible and is approximated as $I_{xz} = 0$

$$L = \dot{p}I_x + qr(I_z - I_y) \tag{3.52}$$

$$M = \dot{q}I_y + rp(I_x - I_z) \tag{3.53}$$

$$N = \dot{r}I_z + pq(I_y - I_x) \tag{3.54}$$

3.2.1 Aircraft Orientation

The location and orientation of the aircraft cannot be defined in the aircraft's Body-axes and needs to be expressed in the Earth-axes. This requires transforming the values expressing the Body-axes with respect to Earth-axes by applying a series of rotations known as Euler angles. The order that the rotations are applied in is important [7,89].

1. The first rotation is the roll ϕ about the X_e axis.
2. The second rotation is the pitch θ about the Y_e axis.
3. The third rotation is the heading ψ about the Z_e axis.

3.2.2 Body-axes to Earth-axes transformation

$$T_B^e(\psi,\theta,\phi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{3.55}$$

3.2.3 Earth-axes to Body-axes transformation

$$T_e^B(\psi,\theta,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.56}$$

3.2.4 Gravitational and Thrust Forces

The forces $\mathbf{F}$ exerted on the aircraft are composed of the aerodynamic forces of the aircraft, earth's gravity and the thrust produced by the aircraft's engine. This is expressed as

$$\mathbf{F} = \mathbf{F}_{a_w} + \mathbf{F}_{g_e} + \mathbf{F}_p \tag{3.57}$$

where $\mathbf{F}$ is the total force exerted on the aircraft. The aerodynamic force contributions are defined in the Wind-axes $\mathbf{F}_{a_w} = [-D, Y, -L_{\text{lift}}]^{\text{T}}$ where D is the drag force, Y is the side force and L_{lift} is the lift force. The gravitational force contributions are defined in Earth-axes as $\mathbf{F}_{g_e} = [0, 0, mg]^{\text{T}}$. The thrust force $\mathbf{F}_p = [T, 0, 0]^{\text{T}}$ is aligned along the X_b axis of the aircraft's Body-axes. This can be different for other types of aircraft since it is dependent on the type and geometric position of the power-plant on the aircraft. Some aircraft have power-plants that are offset from the X_b axis and have a Y_b and Z_b component resulting in offset thrust that contribute to the moment of the aircraft. This is common on large passenger aircraft that have multiple power-plants such as Boeing 747 [90] and the Airbus A320. More sophisticated power-plants have Thrust

Vectoring capabilities where the direction of thrust can be changed during flight [85]. This is a common feature for Fast Jets used for combat such as the Lockheed Martin F-22 and the Sukhoi Su-37. For all examples considered in this thesis, it is assumed that the aircraft's thrust is aligned along the X_b axis in the aircraft's Body-axes.

The force components $\mathbf{F}_{a_w}$ and $\mathbf{F}_{g_e}$ need to be expressed in the Body-axes. The aerodynamic forces are rotated about the z axis by side-slip angle β to the stability axis followed by a rotation about the y axis by the Angle-of-Attack α to the Wind-axes. The Angle-of-Attack α is the angle between the mean chord line of the wing and the relative airflow [84]. The side-slip angle β is the angle between the horizontal component of the velocity vector v and the total velocity V_a [7].

$$\begin{bmatrix} F_{a_x} \\ F_{a_y} \\ F_{a_z} \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -D \\ Y \\ -L_{\text{lift}} \end{bmatrix} \tag{3.58}$$

$$\begin{bmatrix} F_{a_x} \\ F_{a_y} \\ F_{a_z} \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha \end{bmatrix} \begin{bmatrix} -D \\ Y \\ -L_{\text{lift}} \end{bmatrix} \tag{3.59}$$

$$F_{a_x} = -D\cos\alpha\cos\beta - Y\cos\alpha\sin\beta + L_{\text{lift}}\sin\alpha \tag{3.60}$$

$$F_{a_y} = -D\sin\beta + Y\cos\beta \tag{3.61}$$

$$F_{a_z} = -D\sin\alpha\cos\beta - Y\sin\alpha\sin\beta - L_{\text{lift}}\cos\alpha \tag{3.62}$$

where the Angle-of-Attack and the angle of side-slip are computed as

$$\alpha = \tan^{-1}\left(\frac{u - u_g}{w}\right) \tag{3.63}$$

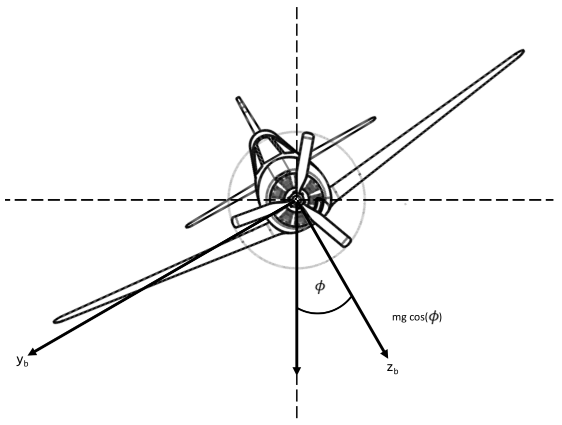
$$\beta = \sin^{-1}\left(\frac{v - v_g}{V_a}\right) \tag{3.64}$$

where V_a is defined as

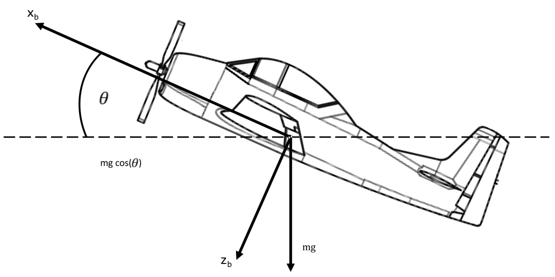
$$V_a = \sqrt{(u - u_g)^2 + (v - v_g)^2 + (w - w_g)^2} \tag{3.65}$$

Gravitational Force

The gravitational force is expressed in earth frame of axis. This is converted to body frame of axis by using the Euler angle rotations.



(a)



(b)

Figure 3.2: Aircraft bank 3.2(a) and pitch 3.2(a) orientations

$$\begin{bmatrix} F_{g_{x_b}} \\ F_{g_{y_b}} \\ F_{g_{z_b}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{3.66}$$

$$\begin{bmatrix} F_{g_{x_b}} \\ F_{g_{y_b}} \\ F_{g_{z_b}} \end{bmatrix} = \begin{bmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\cos\phi \end{bmatrix} \tag{3.67}$$

Equations (3.14), (3.15), (3.16) are substituted into the total force equation (3.57)

$$\begin{bmatrix} m(\dot{u}+qw-rv) \\ m(\dot{v}+ru-pw) \\ m(\dot{w}+pv-qu) \end{bmatrix} = \begin{bmatrix} -D\cos\alpha\cos\beta - Y\cos\alpha\sin\beta + L_{\text{lift}}\sin\alpha \\ -D\sin\beta + Y\cos\beta \\ -D\sin\alpha\cos\beta - Y\sin\alpha\sin\beta - L_{\text{lift}}\cos\alpha \end{bmatrix} + \begin{bmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\cos\phi \end{bmatrix} + \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} \tag{3.68}$$

The Drag D, Lift L_{lift} and side forces Y are a function of dynamic pressure Q, wing surface area of the aircraft S and various other factors such as the attitude rates and control surface state. The forces can be defined as

$$D = QSC_D \tag{3.69}$$

$$Y = QSC_Y \tag{3.70}$$

$$L_{\text{lift}} = QSC_L \tag{3.71}$$

The Drag, Lift and Side force coefficient are C_D, C_L and C_Y respectively. These coefficients are dimensionless and a function of the aircraft's geometric features such as wingspan b, wing chord length c and wing surface area S. The coefficients are determined experimentally by using a scale model of the aircraft in a wind tunnel and varying parameters such as α and measuring effect this has on the Force and Moments exerted on the aircraft [7]. The aerodynamic coefficients are composed of stability derivatives. The stability derivatives are determined by taking Taylor series expansion about the trim condition of the aircraft, where all the forces and moments acting on the aircraft are zero. The change in forces and moments acting on the aircraft due to change in variables such as control surface deflections are represented by the stability derivatives.

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_\beta}\beta + \frac{c}{2V_a}C_{L_q}q + C_{L_{\delta_e}}\delta_e \tag{3.72}$$

$$C_Y = C_{Y_\alpha}\alpha + C_{Y_\beta}\beta + \frac{b}{2V_a}(C_{Y_p}p + C_{Y_r}r) + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_e}}\delta_e + C_{Y_{\delta_r}}\delta_r \tag{3.73}$$

$$C_D = C_{D_0} + C_k C_L^2 \tag{3.74}$$

where δ_a, δ_e and δ_r are the control surface deflections of the aircraft's ailerons, elevator and rudder respectively. The translation equations of motion are defined as:

$$\dot{u} = -Q\frac{S}{m}(C_D\cos\alpha\cos\beta + C_Y\cos\alpha\sin\beta - C_L\sin\alpha) - g\sin\theta + \frac{T}{m} - qw + rv \tag{3.75}$$

$$\dot{v} = -Q\frac{S}{m}(C_D\sin\beta - C_Y\cos\beta) + g\cos\theta\sin\phi - ru + pw \tag{3.76}$$

$$\dot{w} = -Q\frac{S}{m}(C_D\sin\alpha\cos\beta + C_Y\sin\alpha\sin\beta + C_L\cos\alpha) + g\cos\theta\cos\phi - pv + qu \tag{3.77}$$

The moment equations are

$$L = QSbC_l \tag{3.78}$$

$$M = QScC_m \tag{3.79}$$

$$N = QSbC_n \tag{3.80}$$

where C_l, C_m and C_n are the rolling, pitching and yawing moment coefficients.

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_\beta}\beta + \frac{c}{2V_a}(C_{m_{\dot\alpha}}\dot\alpha + C_{m_q}q) + C_{m_{\delta_e}}\delta_e \tag{3.81}$$

$$C_l = C_{l_\alpha}\alpha + C_{l_\beta}\beta + \frac{b}{2V_a}(C_{l_p}p + C_{l_r}r) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_e}}\delta_e + C_{l_{\delta_r}}\delta_r \tag{3.82}$$

$$C_n = C_{n_\alpha}\alpha + C_{n_\beta}\beta + \frac{b}{2V_a}(C_{n_p}p + C_{l_q}q) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_e}}\delta_e + C_{l_{\delta_r}}\delta_r \tag{3.83}$$

The dynamic pressure Q is a function of the air density ρ and the speed V_a of the aircraft. This is defined as

$$Q = \frac{1}{2}\rho V_a^2 \tag{3.84}$$

$$\dot{p} = -\frac{1}{I_x}[qr(I_z - I_y) - QSbC_l] \tag{3.85}$$

$$\dot{q} = -\frac{1}{I_y}[rp(I_x - I_z) - QScC_m] \tag{3.86}$$

$$\dot{r} = -\frac{1}{I_z}[pq(I_y - I_x) - QSbC_n] \tag{3.87}$$

$$\tag{3.88}$$

3.2.5 Navion Aerodynamic coefficients

Table 3.1: Navion aerodynamic coefficients [7]

C_L	C_Y	C_l	C_m	C_n
$C_{L_o} = 0$	$C_{Y_0} = 0$	$C_{l_o} = 0$	$C_{m_o} = 0$	$C_{n_o} = 0$
$C_{L_\alpha} = 4.44$	$C_{Y_\alpha} = 0$	$C_{l_\alpha} = 0$	$C_{m_\alpha} = -0.683$	$C_{n_\alpha} = 0$
$C_{L_\beta} = 0$	$C_{Y_\beta} = -0.564$	$C_{l_\beta} = -0.074$	$C_{m_\beta} = 0$	$C_{n_\beta} = 0.071$
$C_{L_{\dot\alpha}} = 0$	$C_{Y_{\dot\alpha}} = 0$	$C_{l_{\dot\alpha}} = 0$	$C_{m_{\dot\alpha}} = -4.36$	$C_{n_{\dot\alpha}} = 0$
$C_{L_p} = 0$	$C_{Y_p} = 0$	$C_{l_p} = -0.41$	$C_{m_p} = 0$	$C_{n_p} = -0.0575$
$C_{L_q} = 3.8$	$C_{Y_q} = 0$	$C_{l_q} = 0$	$C_{m_q} = -9.96$	$C_{n_q} = 0$
$C_{L_r} = 0$	$C_{Y_r} = 0$	$C_{l_r} = 0.107$	$C_{m_r} = 0$	$C_{n_r} = -0.125$
$C_{L_{\delta_a}} = 0$	$C_{Y_{\delta_a}} = 0$	$C_{l_{\delta_a}} = -0.134$	$C_{m_{\delta_a}} = 0$	$C_{n_{\delta_a}} = -0.0035$
$C_{L_{\delta_e}} = 0.355$	$C_{Y_{\delta_e}} = 0$	$C_{l_{\delta_e}} = 0$	$C_{m_{\delta_e}} = -0.923$	$C_{n_{\delta_e}} = 0$
$C_{L_{\delta_r}} = 0$	$C_{Y_{\delta_r}} = 0.157$	$C_{l_{\delta_r}} = 0.107$	$C_{m_{\delta_r}} = 0$	$C_{n_{\delta_r}} = -0.072$

3.3 Flight Simulation

3.3.1 Level flight conditions

A reference operating point is trimmed level flight where the flight path angle ($\gamma = 0$). The flight path angle is the angle between the aircraft velocity vector in the earth axis and local horizon (X_e, Y_e plane in the earth-axis). In this flight mode the aircraft is at constant speed ($\dot u = \dot v = \dot w = 0$) and maintains a constant attitude ($\dot p = \dot q = \dot r = p =$

$q = r = 0$) and wings are level ($\phi = 0$) and side-slip angle ($\beta = 0$). The elevator angle δ_e and engine thrust T needs to be determined to maintain this flight condition at a given altitude z at a constant airspeed V_a. The Angle-of-Attack is constant ($\dot{\alpha} = 0$). Since flight path angle ($\gamma = \theta - \alpha = 0$) then ($\theta = \alpha$). The lift L_{lift} and weight of the aircraft are balanced

$$L_{\text{lift}} = mg = QSC_L \tag{3.89}$$

$$C_L = \frac{mg}{QS} \tag{3.90}$$

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e \tag{3.91}$$

equation (3.91) is rearranged for Angle-of-Attack α to be the subject as

$$\alpha = \frac{C_L}{C_{L_\alpha}} - \frac{C_{L_0}}{C_{L_\alpha}} - \frac{C_{L_{\delta_e}}\delta_e}{C_{L_\alpha}} \tag{3.92}$$

The pitching moment coefficient $C_m = 0$ equation (3.81) reduces to

$$0 = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}}\delta_e \tag{3.93}$$

which is rearranged for the elevator δ_e to be the subject

$$\delta_e = -\frac{C_{m_0}}{C_{m_{\delta_e}}} - \frac{C_{m_\alpha}}{C_{m_{\delta_e}}}\alpha \tag{3.94}$$

Equation (3.92) is substituted into equation (3.94) to form

$$\delta_e = -\frac{C_{m_0}}{C_{m_{\delta_e}}} - \frac{C_{m_\alpha}}{C_{m_{\delta_e}}}\left(\frac{C_L}{C_{L_\alpha}} - \frac{C_{L_0}}{C_{L_\alpha}} - \frac{C_{L_{\delta_e}}\delta_e}{C_{L_\alpha}}\right) \tag{3.95}$$

$$\delta_e = \frac{-C_{m_0}C_{L_\alpha} - C_{m_\alpha}C_L + C_{m_\alpha}C_{L_0}}{C_{m_{\delta_e}}C_{L_\alpha} - C_{m_\alpha}C_{L_{\delta_e}}} \tag{3.96}$$

The required elevator angle to maintain level flight is determined and applied to equation (3.92) to determine the Angle-of-Attack for the aircraft. The C_L is used to compute C_D using equation (3.74). The thrust is determined by equation (3.75) by

$$T = QSC_D \cos\alpha \tag{3.97}$$

3.3.2 Aircraft State Vector

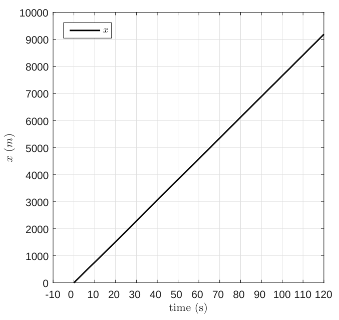
The state of the aircraft is defined by the state vector

$$\mathbf{X} = [x, u, a_x, z, w, a_z, \theta, q, \dot{q}, y, v, a_y, \psi, p, \dot{p}, \phi, r, \dot{r}]^{\mathrm{T}} \tag{3.98}$$

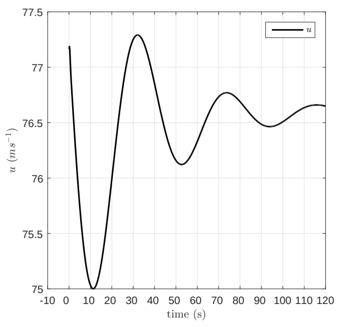
that defines the state of the aircraft mathematically. The elements x, u, $\dot{u}$, z, w, $\dot{w}$, θ, q, $\dot{q}$ are longitudinal states of the aircraft and y, v, a_y, ψ, p, $\dot{p}$, ϕ, r, $\dot{r}$ are lateral states of the aircraft defined by convention [85]. The state elements x, y and z are the displacements of the aircraft in the X_e, Y_e and Z_e directions respectively in the Earth-axes. The state elements u, v and w are the speeds of the aircraft in the X_b, Y_b and Z_b directions respectively in the aircraft Body-axes. The state elements a_x, a_y and a_z are the accelerations of the aircraft in the X_b, Y_b and Z_b directions respectively in the Body-axes. The state elements ψ, θ and ϕ are relative attitudes of the aircraft body about the Z_e, Y_e and X_e axes respectively of the Earth-axes. The state elements p, q and r are the angular rates of the aircraft body about the X_b, Y_b and Z_b axes respectively in the aircraft Body-axes. The state elements $\dot{p}, \dot{q}$ and $\dot{r}$ are the angular rates of the aircraft body about the X_b, Y_b and Z_b axes respectively in Body-axes.

3.3.3 Level Flight Results

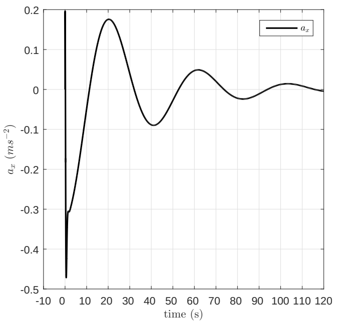
An example of the level fight simulation of the SixDoF model is presented below. The simulation begins in the trimmed condition at an altitude of 5000 ft (1524m) and at a speed of 150 knots (77.2ms^{-1}).



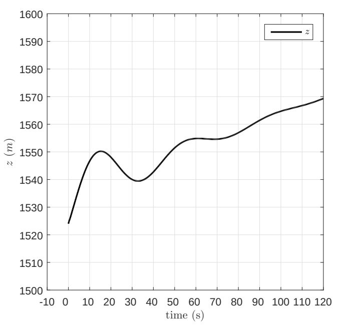
(a) Displacement in the X_e direction



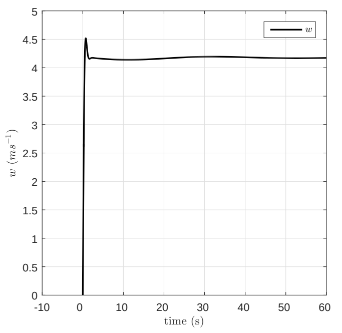
(b) Speed u in the X_e direction



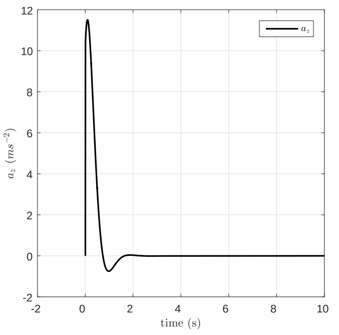
(c) Acceleration a_x in the X_e direction



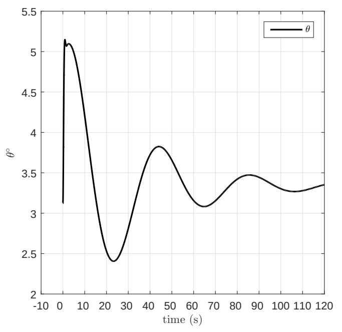
(d) Displacement in the Z_e direction



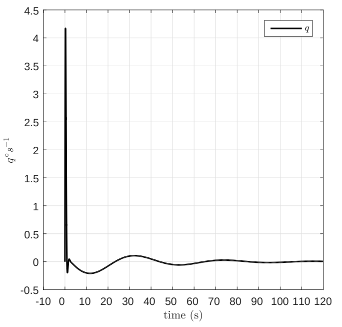
(e) Speed v in the Z_e direction



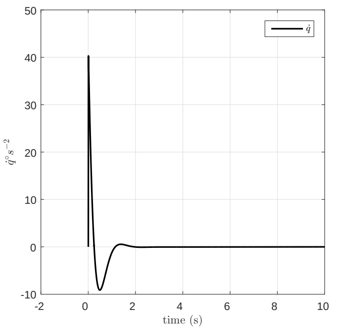
(f) Acceleration a_z in the Z_e direction



(g) Pitch attitude θ about the X_b axis



(h) Pitch rate q about the X_b axis



(i) Pitch acceleration $\dot{q}$ about the X_b axis

Figure 3.3: Longitudinal states for Navion during level flight at 5000 ft (1524m).

Level flight is simulated for 120 seconds. Figure 3.3 show the longitudinal states x, u, a_x, z, w, a_z, θ, q and $\dot{q}$ of the aircraft during the simulation. The lateral states y, v, a_y, ψ, p, $\dot{p}$, ϕ, r and $\dot{r}$ remain zero since level flight does not induce side force, rolling and yaw moments. The longitudinal states a_x, a_y and $\dot{q}$ show an oscillations at the start of the simulation that gradually dissipates during the period of the simulation. The duration of the oscillation is large and causes dependent states to fluctuate. A control system is required to dampen the oscillation and achieve the target flight condition. The acceleration a_x oscillates for the period of the simulation. This causes the speed u to oscillate for the period of the simulation. The flight model needs to be dampened along the longitudinal axis to reach the desired flight condition quickly. This requirement is addressed in the next section.

3.4 Autopilot

The objective of the autopilot is to simulate common aircraft flight modes such as level, climbing, descending and turning flight. This allows test scenarios to be specified that will generate flight data for testing fault detection algorithms. The autopilot inputs are the desired heading ψ_c and altitude z_c to be maintained. The change in altitude and heading is achieved by using a displacement autopilot shown in [7].

To maintain the desired altitude z_c, the aircraft needs to be dampened along the longitudinal axis. Figure 3.3(d) shows an increase in altitude Δz during the simulation. This occurs due to the change in displacement accelerations (a_x, a_z) and pitch acceleration $\dot{q}$. These changes cause oscillation in the speeds u, w and rotational rates q that result in change in displacements Δx, Δz and $\Delta\theta$. During level flight the Angle-of-Attack α needs to be constant to maintain the desired altitude. The α is proportional to the elevator angle δ_e to maintain level flight as defined by equation 3.94. However, the change in pitch rate causes a change in α.

$$\alpha_\epsilon = \alpha - \alpha_k \tag{3.99}$$

where α_ϵ is the error, α is the desired Angle-of-Attack and α_k is current Angle-of-Attack. The error in altitude is proportional to the error in the Angle-of-Attack

$$\Delta z \propto \tan(\alpha_\epsilon) \tag{3.100}$$

the small angle approximation $\tan a \approx a$ reduces the equation to

$$\Delta z \propto \alpha_\epsilon \tag{3.101}$$

$$\Delta z = K_1 \alpha_\epsilon \tag{3.102}$$

where K_1 is a gain.

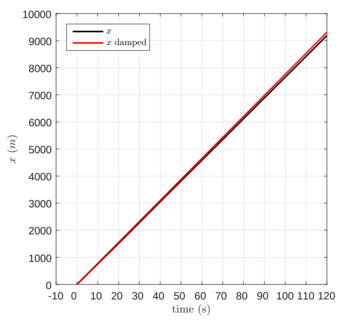
$$\alpha_\epsilon = \frac{1}{K_1}\Delta z \tag{3.103}$$

The elevator angle δ_e required to minimize α_ϵ is determined

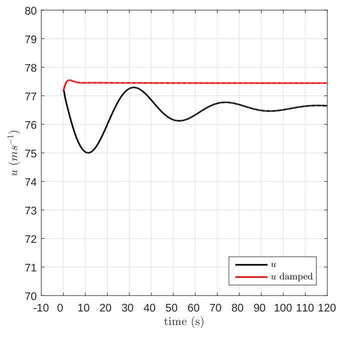
$$\delta_{e_{k+1}} = \delta_{e_k} + \text{sign}(\alpha_\epsilon)\frac{\min(K_{\delta_e}|\alpha_\epsilon|, q_{\max})}{q_{\delta_e}} \tag{3.104}$$

$$\text{where} \quad \text{sign}(\alpha_\epsilon) = \left\{ \begin{array}{ll} -1 & \text{if } \alpha_\epsilon < 0 \\ 0 & \text{if } \alpha_\epsilon = 0 \\ +1 & \text{if } \alpha_\epsilon > 0 \end{array} \right\}$$

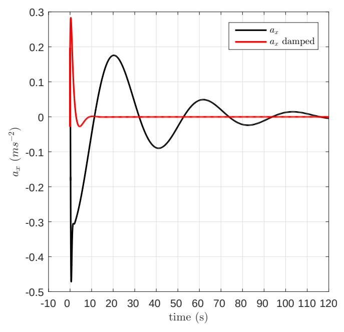
where $\frac{q}{\delta_e} = q_{\delta_e}$ is the pitch rate per degree of the elevator angle, $\delta_{e_{k+1}}$ is the required elevator angle, δ_{e_k} is the current angle of the elevator for α_k. The elevator gain is K_{δ_e}. The pitch rate is limited to q_{max}.



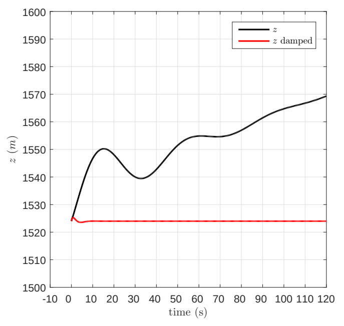
(a) Displacement in the X_e direction



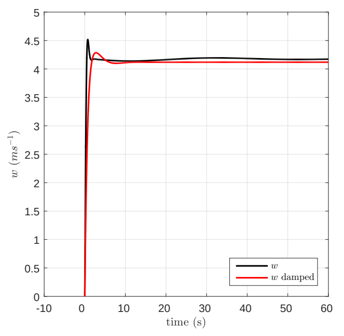
(b) Speed u in the X_e direction



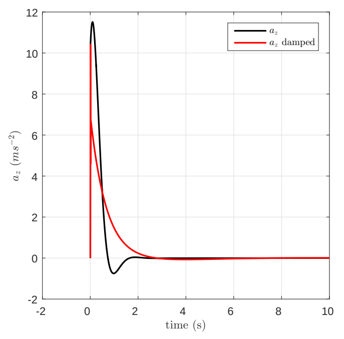
(c) Acceleration a_x in the X_e direction



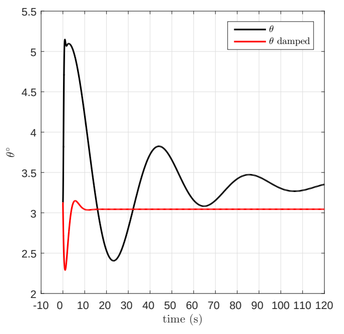
(d) Displacement in the Z_e direction



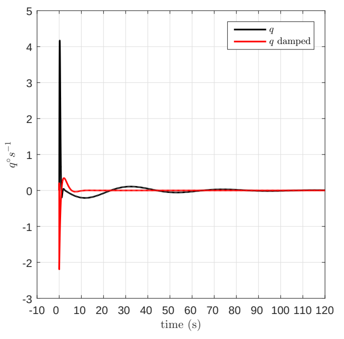
(e) Speed v in the Z_e direction



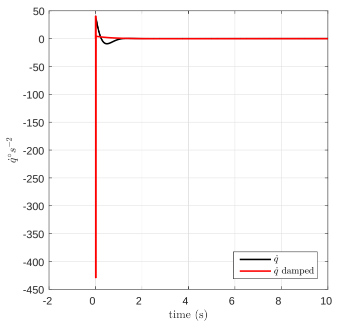
(f) Acceleration a_z in the Z_e direction



(g) Pitch attitude θ about the X_b axis



(h) Pitch rate q about the X_b axis



(i) Pitch acceleration $\dot{q}$ about the X_b axis

Figure 3.4: Longitudinal states for Navion in level flight at 5000ft (1524m) with a dampened system.

Figure 3.4 shows the damped longitudinal states during level flight.

The heading of the aircraft is controlled by inducing a roll rate p using ailerons. Inducing a roll causes a bank angle ϕ that changes the heading of the aircraft. The difference between the current heading ψ and the desired heading ψ_{c} is $\Delta\psi$. This is the angle between the unit vector of the velocity vector of the aircraft $\hat{\mathbf{V}}_{\text{earth}}$ and the desired heading vector $\mathbf{u}_c$ defined

$$\hat{\mathbf{V}}_{\mathrm{earth}} = T_B^e(\psi, \theta, \phi)\mathbf{V}_{\mathrm{body}} \tag{3.105}$$

where $\mathbf{V}_{\mathrm{body}} = [u, v, w]^{\mathrm{T}}$ is the velocity of the aircraft in the body axis.

$$\mathbf{u}_c = T_B^e(\psi_c, 0, 0)\mathbf{u} \tag{3.106}$$

where $\mathbf{u} = [1, 0, 0]^{\mathrm{T}}$ is a unit vector aligned along the X axis of the aircraft body axis that is rotated by ψ_c with respect the earth axis. The difference in heading is proportional to the required bank angle ϕ_d

$$\phi_d = \mathrm{sign}(\Delta\psi)\mathrm{min}(K_\phi|\Delta\psi|, \phi_{\mathrm{max}}) \tag{3.107}$$

$$\text{where} \quad \mathrm{sign}(\Delta\psi) = \left\{ \begin{array}{ll} -1 & \text{if } \Delta\psi < 0 \\ 0 & \text{if } \Delta\psi = 0 \\ +1 & \text{if } \Delta\psi > 0 \end{array} \right\}$$

K_ϕ is a gradient and ϕ_{max} is the maximum bank angle limit for the aircraft. This is determined according to the maximum load factor n_{max} that the aircraft is capable of sustaining $n = \frac{1}{\cos\phi}$ [91]. For aircraft such as Navion the maximum load factor is expected to remain between 1 and 2. Assuming $n_{max} = 1.5$, a bank limit of $\phi_{max} = \pm 45^\circ$ is appropriate. The aileron control surface deflection $\delta_{a_{k+1}}$ is determined by

$$\delta_{a_{k+1}} = \delta_{a_k} + \mathrm{sign}(\Delta\phi)\frac{\mathrm{min}(K_{\delta_a}|\phi_d|, p_{\mathrm{max}})}{p_{\delta_a}} \tag{3.108}$$

$$\text{where} \quad \mathrm{sign}(\Delta\phi) = \left\{ \begin{array}{ll} -1 & \text{if } \Delta\phi < 0 \\ 0 & \text{if } \Delta\phi = 0 \\ +1 & \text{if } \Delta\phi > 0 \end{array} \right\}$$

δ_{a_k} is the current aileron angle of deflection, K_{δ_a} is the aileron gain, the roll rate is limited to p_{max} and $\frac{p}{\delta_a} = p_{\delta_a}$ is the roll rate per degree of the aileron angle.

The aileron deflection also induces a yaw rate that contributes to the side-slip angle β of the aircraft [85]. The rudder control surface δ_r is used to minimize the side-slip angle. The rudder control deflection $\delta_{r_{k+1}}$ is determined

$$\delta_{r_{k+1}} = \delta_{r_k} + \mathrm{sign}(\beta_k)\frac{\mathrm{min}(K_{\delta_r}|\beta_k|, r_{\mathrm{max}})}{r_{\delta_r}} \tag{3.109}$$

$$\text{where} \quad \mathrm{sign}(\beta_k) = \left\{ \begin{array}{ll} -1 & \text{if } \beta_k < 0 \\ 0 & \text{if } \beta_k = 0 \\ +1 & \text{if } \beta_k > 0 \end{array} \right\}$$

δ_{r_k} is the current rudder angle of deflection, K_{δ_r} is the rudder gain, $r_{\max}$ is the maximum yaw rate limit and $\frac{r}{\delta_r} = r_{\delta_r}$ is the yaw rate per degree of the aileron angle.

The parameters for the autopilot are:

- $K_1 = 200$
- $K_{\delta_e} = 0.9$
- $q_{\max} = 10°s^{-1}$
- $K_\phi = 5$
- $\phi_{\max} = 45°$
- $K_{\delta_a} = 1.75$
- $p_{\max} = 10°s^{-1}$
- $K_{\delta_r} = 10$
- $r_{\max} = 5°s^{-1}$

3.4.1 Climbing flight

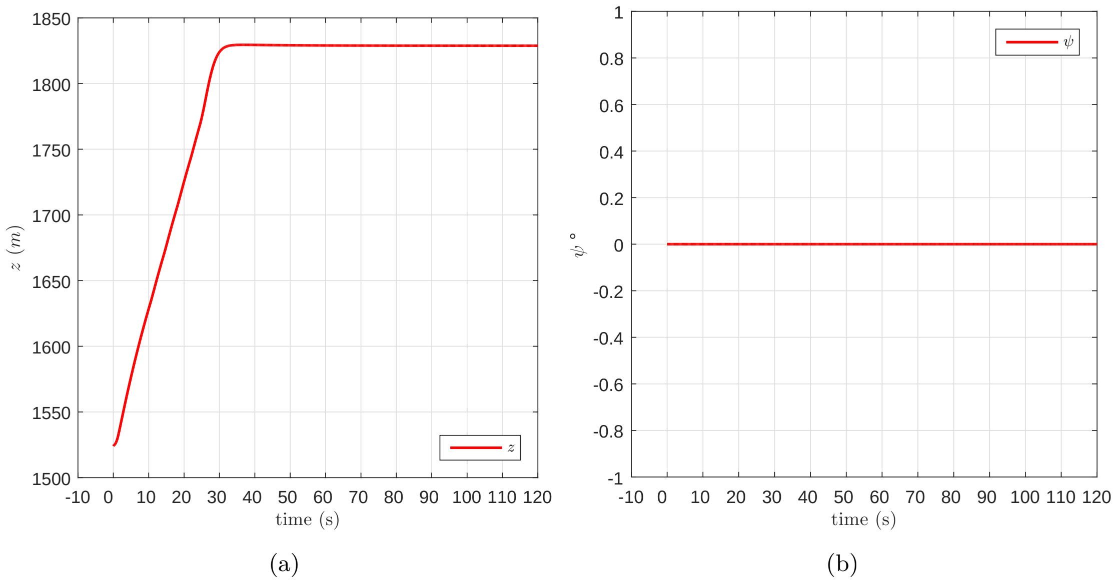
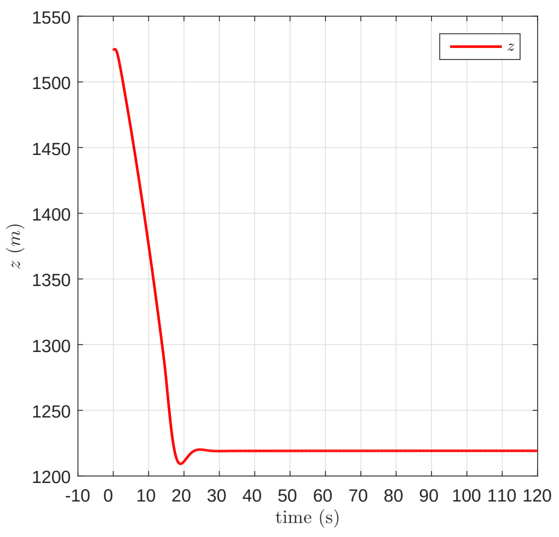
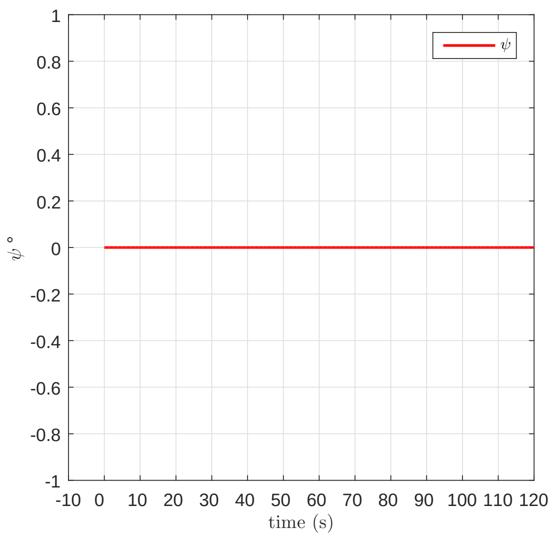


Figure 3.5: The aircraft climbs and maintains altitude of 1828.8m (6000ft) and heading $0°$.

3.4.2 Descending flight



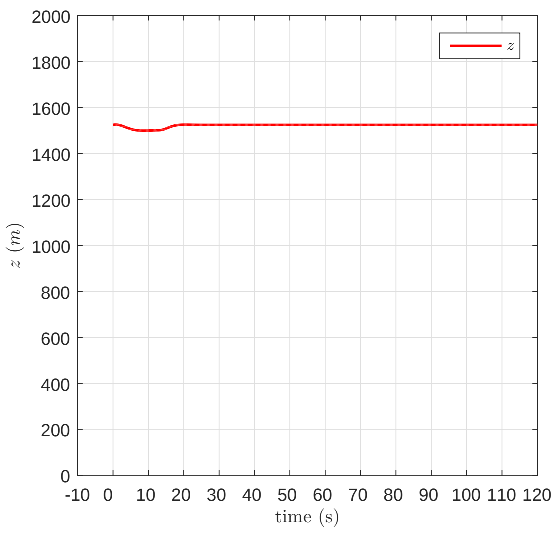
(a)



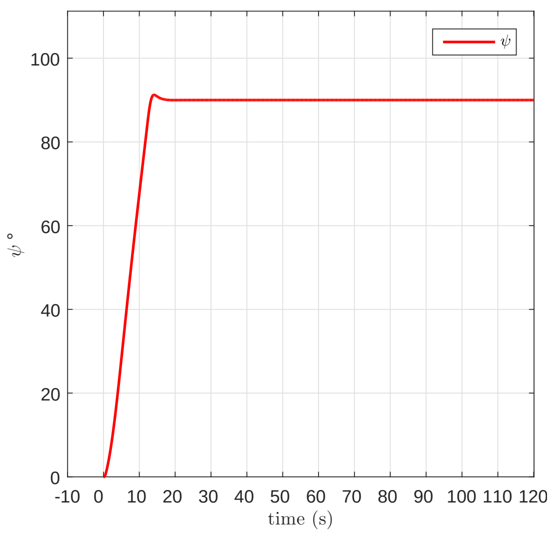
(b)

Figure 3.6: The aircraft descends and maintains altitude of 1219.2m (4000ft) and heading $0°$.

3.4.3 Turning flight



(a)



(b)

Figure 3.7: The aircraft turns and maintains a heading of $90°$ *and altitude of 1219.2m (4000ft).*

3.5 Summary

This chapter has demonstrated the development of a Navion flight model with an autopilot that represents an Unmanned Aircraft. This will be used as a test platform in simulation of common flight scenarios. The autopilot is useful for initialising different flight phases such as climbing, descending, turning and level flight. The model can be used as a test platform to generate data for fault detection algorithms.

Chapter 4

Sensors and State Estimation

Aircraft systems are reliant on data obtained from on-board sensors to deduce their state and the state of the environment they are operating within. The measurements obtained from sensors are inherently noisy and they limit the accuracy of the state estimate of the aircraft. State estimation methods are used to filter the noise from the measurements and maintain an accurate estimate of the aircraft's state, such as the current altitude, speed and heading of the aircraft. There are different types of sensors, each type of sensor is used to provide specific information related to the state of the aircraft [80].

Sensors can be categorised as dependent and independent. Dependent sensors are reliant on distributed systems. For example, the Global Positioning System (GPS) receivers on-board an aircraft are reliant on the GPS satellite constellation to determine the aircraft position [92]. This type system is vulnerable, to attacks such as spoofing and jamming. In contrast, independent sensors are not dependent on external systems to provide measurements. For example Angle-of-Attack (AoA) sensors are on-board the aircraft and provide the measurement of the angle between the mean chord of the wing and the relative airflow. This angle is denoted by α.

The three different types of independent sensors that are commonly found on-board aircraft and are considered for this flight model are an Inertial Measurement Unit, an Angle-of-Attack sensor and a Pitot-static sensor.

- Inertial Measurement Unit (IMU): Measures the accelerations (a_x, a_y and a_z) and rotation rates (p, q and r) of the aircraft. These are integrated with respect to time to produce the displacement, velocity, attitude and rotation rate of the aircraft.

- Angle-of-Attack (AoA) sensor: Measures the angle α between the mean chord of the aerofoil wing and the relative airflow.

- Pitot-static (PS) sensor: Measures the static pressure P_s and total pressure P_T

encountered during the flight. The static pressure is used determine the barometric altitude. The total pressure is a sum of the static pressure and the Dynamic pressure Q. This is used infer the local airspeed aircraft.

This chapter begins by defining the measurement models for the above sensors based on [80]. Noisy sensor data are simulated by adding zero-mean Gaussian white noise to the expected measurement determined using the Navion flight model defined in the previous chapter. White noise is a process where the autocovariance for two different times is zero, ie. the noise is uncorrelated in time [93]. Gaussian noise is used because of the Central Limit Theorem, where the sum of many noise sources always (or nearly always) approximates to a Gaussian distribution. Therefore, a Gaussian distribution is a good model for sensors. However, in practice sensors have bias due to factors such as imperfect manufacturing processes or coupling effects due to interference between instruments [94]. The examples in this thesis do not consider such errors. This approach of using additive white noise has been used in reference [75] and is common in simulation and modelling. Section 4.2 considers popular state estimation methods such as Kalman Filter, Extended Kalman Filter and Unscented Kalman Filter to filter the noise and maintain an accurate estimate of the aircraft state.

4.1 Sensor simulation

4.1.1 Inertial Measurement Unit

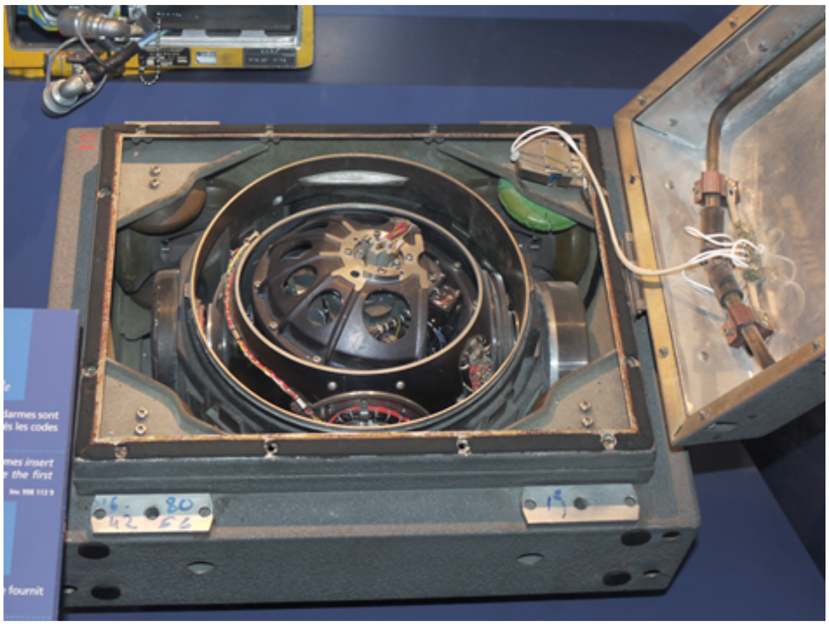


Figure 4.1: Inertial Measurement Unit [1]

The Inertial Measurement Unit (IMU) is located inside the aircraft commonly at or near the aircraft's centre of gravity. The IMU is composed of accelerometers and gyroscope sensors that measure the accelerations and rotational rates of the aircraft respectively. The position, velocity and orientation of the aircraft are deduced by integrating the measurements obtained from the IMU over time. This sensor's measurements are simulated by adding white noise to the expected output of the sensor. The measurements obtained at time-step k from the Inertial Measurement Unit (IMU) are simulated as

$$\mathbf{Y}_k^{(\mathrm{IMU})} = h^{(\mathrm{IMU})}([a_x, a_y, a_z, p, q, r]^{\mathrm{T}}, \mathbf{V}_{\mathrm{IMU}}) = [a_x, a_y, a_z, p, q, r]^{\mathrm{T}} + \mathbf{V}_{\mathrm{IMU}} \tag{4.1}$$

where $\mathbf{V}_{\mathrm{IMU}} \sim \mathcal{N}(0, \sigma^2_{\mathrm{IMU}})$ is the measurement noise and

$$\sigma^2_{\mathrm{IMU}} = \begin{bmatrix} \sigma^2_{a_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{a_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{a_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_p & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_q & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_r \end{bmatrix} \tag{4.2}$$

The variance of the measured accelerations a_x,a_y and a_z are $\sigma^2_{a_x}$, $\sigma^2_{a_y}$ and $\sigma^2_{a_z}$ respectively. The variance of rotation rates p, q and r are σ^2_p, σ^2_q and σ^2_r. The covariance have been assumed negligible. In practice, the magnitude of cross-couplings are non-zero but the magnitude of these terms are significantly smaller than the diagonal terms and can be assumed negligible for the purposes of this thesis.

4.1.2 Angle-of-Attack Sensor

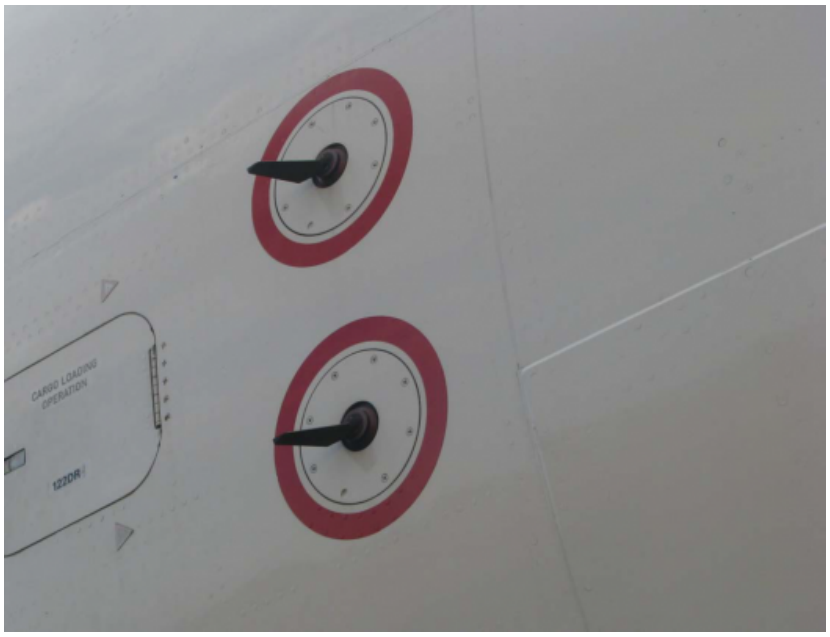


Figure 4.2: Two Angle-of-Attack Sensors on Airbus A330 [2]

This is an external sensor that is mounted on the side of side of the aircraft. It consists of a vane that rotates as the aircraft pitches. The Angle-of-Attack (AoA) α is the angle between the mean chord of the aerofoil and the relative airflow. The data from this sensor are simulated by determining the expected value for α based on the equations of motion and adding noise terms.

$$Y_k^{(\alpha)} = h^{(\mathrm{AoA})}(u_k, w_k, V_\alpha) = \tan^{-1}\left(\frac{u_k}{w_k}\right) + V_\alpha \tag{4.3}$$

where $V_\alpha \sim \mathcal{N}(0, \sigma_\alpha^2)$ and σ_α^2 is the variance of the angle of attack, u_k and w_k are the speed aircraft along the X-axis and Z-axis in the aircraft body-axis respectively.

4.1.3 Pitot-static Sensor

The Pitot-static (PS) sensor is an external sensor mounted on the fuselage of the aircraft. This sensor measures the total pressure P_T and static pressure P_s encountered during flight. These measurements are used to determine the current speed and barometric altitude of the aircraft. The dynamic pressure Q is the difference between the total pressure P_T and the static pressure P_s; ($Q = P_T - P_s$). The airspeed V_a is proportional to the dynamic pressure and the air density ρ at the current altitude.

The International Standard Atmosphere (ISA) model is used to approximate the air density ρ and static pressure P_s encountered at a specific altitude [84]. This is a standard model that has been commonly used for applications such as simulating missiles.

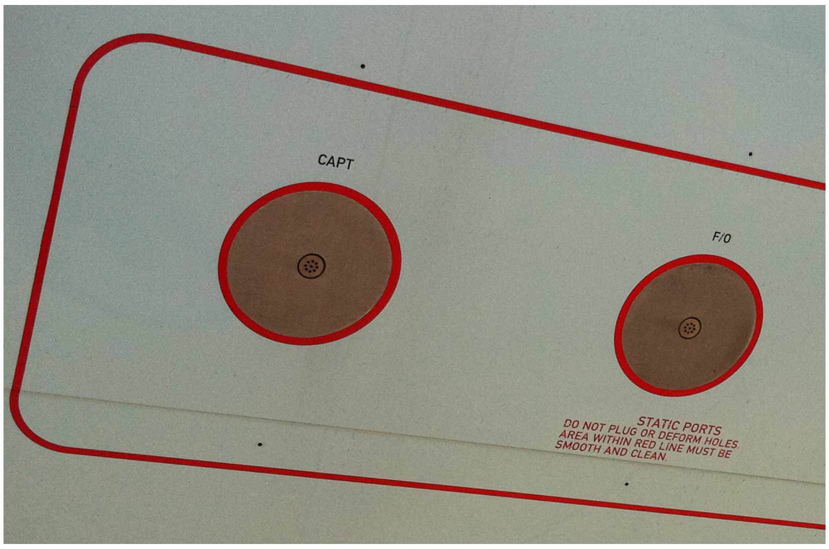


Figure 4.3: Two Static ports on Airbus A330 [3]

The ISA atmosphere model is constructed by averaging the atmospheric conditions over the globe and over the seasons, in practice meteorological variations in temperatures can be modelled by using up to date weather data [84]. Such variations in temperature are not considered in examples in this thesis.

The air density ρ at a given altitude z is determined

$$\rho = \rho_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR} - 1\right)} \tag{4.4}$$

where the air density at sea level is $\rho_0 = 1.225$ kgm^{-3}. The lapse rate $l = 6.49$ Kkm^{-1}. The temperature at sea level $T_0 = 288.15$ K. The acceleration due to gravity $g = 9.81$ ms^{-2}. The specific gas constant is $R = 287$ Jkg^{-1} K^{-1}. The static pressure P_s at the current altitude z is simulated

$$P_s = P_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR}\right)} \tag{4.5}$$

where the static pressure at sea level is $P_0 = 101.325$ kPa. The dynamic pressure Q is calculated using

$$Q = \frac{1}{2}\rho V_a^2 = \frac{1}{2}\rho_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR} - 1\right)} V_a^2 \tag{4.6}$$

and the total pressure encountered during flight is given by

$$P_T = \frac{1}{2}\rho_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR} - 1\right)} V_a^2 + P_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR}\right)} \tag{4.7}$$

The static pressure $Y_k^{(P_s)}$ and the total pressure $Y_k^{(P_T)}$ of the sensor are simulated by adding noise to the expected total pressure P_T and static pressure for the current speed V_a at altitude z.

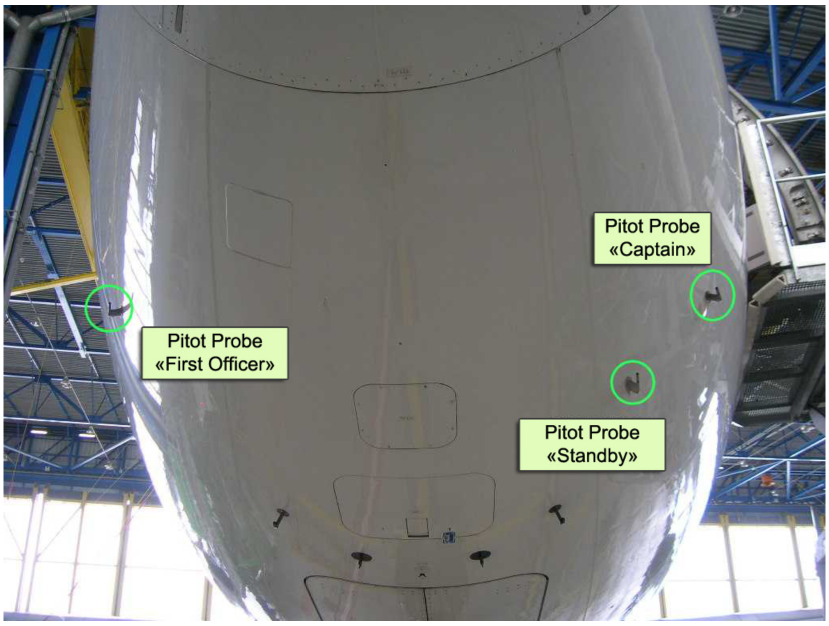


Figure 4.4: Pitot tubes on A330 [4]

$$Y_k^{(P_s)} = h^{(P_s)}(z, V_{P_s}) = P_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR}\right)} + V_{P_s} \tag{4.8}$$

$$\begin{aligned} Y_k^{(P_T)} &= h^{(P_T)}(u, v, w, z, V_{P_T}) \\ &= P_T + V_{P_T} \\ &= \frac{1}{2}\rho_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR}-1\right)} (u^2 + v^2 + w^2) + P_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR}\right)} + V_{P_T} \end{aligned} \tag{4.9}$$

where $V_{P_s} \sim \mathcal{N}(0, \sigma^2_{P_s})$ and $V_{P_T} \sim \mathcal{N}(0, \sigma^2_{P_T})$

4.2 State Estimation

The noisy sensor outputs must be filtered to obtain an accurate estimate of the aircraft state. The accuracy such information must be preserved since it affects the actions of the autopilot.

4.2.1 Kalman Filter

The Kalman Filter (KF) is a popular state estimation method that was proposed by Rudolf Kalman in 1960 [95]. This method has been used for various applications such as Navigation [96], Finance [97] and Medicine [98]. The Kalman filter is a recursive filter that provides an optimal minimum mean-squared error (MMSE) estimate of the current state when estimating the state of linear dynamic systems with Gaussian White noise [93]. A description of the Kalman filter algorithm for a linear dynamic system is given below [93].

A discrete-time linear system is defined in state-space form

$$\mathbf{X}_{k+1|k} = F\mathbf{X}_k + \tilde{\mathbf{W}} + \mathbf{B}\eta_k \tag{4.10}$$

where $\mathbf{X}_k$ is the state vector that defines the state of the system at time-step k, F is the transition matrix that is known and defines the linear system process. The process noise is $\tilde{\mathbf{W}}$. The $\mathbf{B}$ matrix is the control gain and η_k is the control vector. The measurement equation is

$$\mathbf{Y}_{k+1} = H\mathbf{X}_k + \mathbf{V} \tag{4.11}$$

where $\mathbf{Y}_{k+1}$ is the measurement, H is the measurement matrix and $\mathbf{V}$ is the measurement noise. The Kalman filter is initialised with an initial estimate $\hat{\mathbf{X}}_0$ and initial error covariance S_0 obtained from a series of preliminary measurements at $k = 0$. The estimated state of the system at time-step k is $\hat{\mathbf{X}}_k$ and is time-updated by

$$\hat{\mathbf{X}}_{k+1|k} = F\hat{\mathbf{X}}_k + \mathbf{B}\eta_k \tag{4.12}$$

where $\hat{\mathbf{X}}_{k+1|k}$ is the estimated state. The estimated error covariance at time-step k is S_k and is time-updated by

$$S_{k+1|k} = FS_kF^T + \tilde{Q} \tag{4.13}$$

where S_{k+1} is the predicted error covariance and $\tilde{Q}$ is the process noise covariance. The Kalman gain G is determined by

$$G = S_{k+1|k}H^T([HS_{k+1|k}H^T] + R)^{-1} \tag{4.14}$$

where R is the measurement noise covariance. This is followed by updating the state-estimate $\hat{\mathbf{X}}_{k+1}$ and error covariance S_{k+1} respectively.

$$\hat{\mathbf{X}}_{k+1|k+1} = \hat{\mathbf{X}}_{k+1|k} + G\epsilon \tag{4.15}$$

where ϵ is the innovation

$$\epsilon = Y_{k+1} - [H\hat{\mathbf{X}}_{k+1|k}] \tag{4.16}$$

$$S_{k+1|k+1} = [I - GH]S_{k+1|k} \tag{4.17}$$

The Kalman Filter is optimal for estimating the state of linear systems with linear measurements and Gaussian noise. However, many practical problems require estimating the state of nonlinear systems and processes. For example, tracking the trajectory of a ballistic object re-entering the Earth's atmosphere is very nonlinear [99].

4.2.2 Extended Kalman filter

In the case of nonlinear systems, an extended version of the Kalman Filter known as the Extended Kalman Filter (EKF) [100] is often used. This is where the nonlinear function is linearised around an estimated point using a Taylor series expansion and the linearised terms are substituted into the Kalman Filter. The algorithm for a nonlinear discrete EKF using first-order Taylor series expansion is shown below [93, 101, 102].

A discrete-time nonlinear system that is defined by

$$\mathbf{X}_{k+1|k} = f(\mathbf{X}_k) + \tilde{\mathbf{W}} + \mathbf{B}\eta_k \tag{4.18}$$

where $f(...)$ is a known nonlinear function that propagates the current state $\mathbf{X}_k$ to the next state $\mathbf{X}_{k+1}$. The measurement equation is

$$\mathbf{Y}_{k+1} = h(\mathbf{X}_k) + \mathbf{V} \tag{4.19}$$

where $h(...)$ is the nonlinear measurement function. To estimate the predicted state $\hat{\mathbf{X}}_{k+1|k}$ the nonlinear function $f(...)$ is expanded around the most recent estimate $\hat{\mathbf{X}}_k$ using vector Taylor series expansion up to the first-order terms as shown in [101] to yield

$$f(\mathbf{X}) + \tilde{W} \approx f(\hat{\mathbf{X}}_k) + \nabla f(\hat{\mathbf{X}}_k)(\mathbf{X} - \hat{\mathbf{X}}_k) + \tilde{W} + \mathbf{B}\eta_k \tag{4.20}$$

where ∇f is the gradient of the function $f(\mathbf{X})$

$$\nabla f(\hat{\mathbf{X}}_k) = \begin{bmatrix} \frac{\partial f}{\partial X_1} \\ \vdots \\ \frac{\partial f}{\partial X_n} \end{bmatrix} \tag{4.21}$$

the Jacobian matrix is

$$F_k = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial X_1} & \cdots & \frac{\partial f_m}{\partial X_n} \end{bmatrix} \tag{4.22}$$

The predicted state $\hat{\mathbf{X}}_{k+1|k}$ is determined by

$$\hat{\mathbf{X}}_{k+1|k} = f(\hat{\mathbf{X}}_k) + \mathbf{B}\eta_k \tag{4.23}$$

The predicted error covariance $S_{k+1|k}$ is determined by equation

$$S_{k+1|k} = F_k S_k F_k^T + \tilde{Q} \tag{4.24}$$

The measurement function is expanded around $\mathbf{X} = \hat{\mathbf{X}}_{k+1|k}$

$$h(\mathbf{X}) + V \approx h(\hat{\mathbf{X}}_{k+1|k}) + \nabla h(\hat{\mathbf{X}}_{k+1|k})(\mathbf{X} - \hat{\mathbf{X}}_{k+1|k}) + V \tag{4.25}$$

where ∇h is the gradient of the function $h(\hat{\mathbf{X}}_{k+1|k})$

$$\nabla h(\hat{\mathbf{X}}_{k+1|k}) = \begin{bmatrix} \frac{\partial h}{\partial X_1} \\ \vdots \\ \frac{\partial h}{\partial X_n} \end{bmatrix} \tag{4.26}$$

the Jacobian measurement matrix is given by

$$H_k = \begin{bmatrix} \frac{\partial h_1}{\partial X_1} & \cdots & \frac{\partial h_1}{\partial X_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial X_1} & \cdots & \frac{\partial h_m}{\partial X_n} \end{bmatrix} \tag{4.27}$$

The measurement matrix is used to calculate the Kalman gain

$$G = S_{k+1|k} H_k^T ([H_k S_{k+1|k} H_k^T] + R)^{-1} \tag{4.28}$$

where R is the measurement noise covariance. This is followed by updating the time-updated estimate $\hat{\mathbf{X}}_{k+1|k}$ and error covariance $\hat{S}_{k+1}$ respectively.

$$\hat{\mathbf{X}}_{k+1|k+1} = \hat{\mathbf{X}}_{k+1|k} + G\{\mathbf{Y}_{k+1} - [h(\hat{\mathbf{X}}_{k+1|k})]\} \tag{4.29}$$

where $\mathbf{Y}_{k+1}$ is the measurement vector.

$$S_{k+1|k+1} = [I - GH_k] S_{k+1|k} \tag{4.30}$$

The main difference between the Kalman Filter and Extended Kalman Filter is the evaluation of the Jacobian matrices that are used within the Kalman Filter. In the case of second-order or a 'Higher-order' EKF – the second or higher order terms of the Taylor series expansion are used which require computing the Hessian matrices that are the derivatives of the Jacobian matrices. However, including higher order terms does not guarantee accuracy of the estimate since the linearisation is conducted around the latest estimate because the actual state is not available, therefore errors in the estimated state are propagated to the linearised approximation of the system [93]. Unless the EKF is initialised with accurate measurements and the error propagation is well approximated, the error increases and the estimate departs from the true state of the system [93]. This is a well known problem within the target tracking community that has been encountered for applications such as estimating the ballistic parameters of missiles [103, 104].

4.2.3 Unscented Transformation

The Unscented Transformation (UT) proposed by Julier et al. addresses the linearisation issues faced by the Extended Kalman Filter (EKF) [105]. *The UT is founded on the intuition that it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation* [5]. The UT method draws a set of sigma points with mean $\hat{\mathbf{X}}_k$ and covariance S_k. A nonlinear function is applied to each of the sigma points. The mean and covariance of the transformed sigma points represent the mean and covariance of the nonlinear transformation.

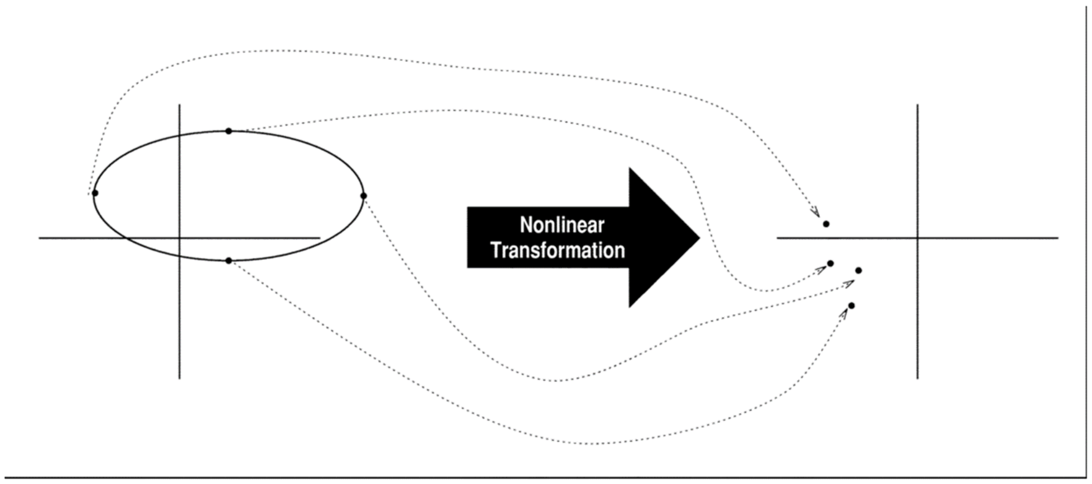


Figure 4.5: The Unscented Transform – a nonlinear function is applied to a set of sigma points [5]

Consider the nonlinear measurement equation (4.19). The measurement $\mathbf{Y}_{k+1}$ with dimension $n_y \times 1$ is observed for the current state of the system $\mathbf{X}_{k+1}$ with dimension $n_x \times 1$. The sigma points are determined by

$$\mathbf{X}^{(0)} = X_k \tag{4.31}$$

$$\mathbf{X}^{(i)} = X_k + \left(\sqrt{n_x S}\right)_i \quad i = 1, ..., n_x \tag{4.32}$$

$$\mathbf{X}^{(i+n_x)} = X_k - \left(\sqrt{n_x S}\right)_i \quad i = 1, ..., n_x \tag{4.33}$$

where $\sqrt{n_x S}$ is the square root of the matrix $n_x S$ and S is the error covariance. The subscript i in $\left(\sqrt{n_x S}\right)_i$ within equations 4.32 and 4.33 refers to the i^{th} column in the matrix $n_x S$. The sigma points are propagated using the non-linear measurement function $h(...)$.

$$Y^{(i)} = h(X_k^{(i)}) \quad i = 0, ..., 2n_x \tag{4.34}$$

The mean of the transformed sigma points are calculated

$$\hat{\mathbf{Y}} = \sum_{i=0}^{n_Y} W^{(i)} Y^{(i)} \tag{4.35}$$

where $W^{(i)} = \frac{1}{2n_x}$ are the weights of the sigma points. The weight (gain) matrix is chosen to minimise the trace of the updated covariance [5]. The covariance of the sigma points, S, is calculated as

$$S = \sum_{i=0}^{n_Y} \mathbf{W}^{(i)} (Y^{(i)} - \hat{\mathbf{Y}})(Y^{(i)} - \hat{\mathbf{Y}})^{\mathrm{T}} \quad i = 0, ..., 2n_y \tag{4.36}$$

This approach captures the second order linearisation information of the nonlinear transformation without the need for a second order Taylor series expansion. The Unscented Transformation can be generalised in the Unscented Kalman Filter [101]. The Unscented Kalman Filter provides an improved performance in state estimation when compared to Extended Kalman Filter for nonlinear systems [101]. The EKF requires the computation of the Jacobian and in some cases – Hessian matrices, the derivative of Jacobian matrices. It is therefore suited to analytical process and measurement equations [101]. However, in practice, some systems cannot be defined analytically and it is numerically difficult to compute Jacobian matrices [101]. For example, the Navion flight model defined in the previous chapter consists of nonlinear dynamics. Such nonlinear problems have been known to cause increased complexity when computing Jacobian or Hessian Matrices [5].

The general form of the Unscented Kalman filter assumes that process and measurement noise is additive and that the covariance is constant. However, estimating the state of nonlinear systems where the noise covariance changes and/or is within the nonlinear function (not additive) requires the simulation of the noise within the nonlinear function.

4.2.4 Augmented Unscented Kalman Filter Algorithm

The augmented form of the UKF [5, 106] is suitable for the estimation of the state of nonlinear systems where the noise is within the nonlinear function and the covariance of the noise changes. It provides a method to estimate the properties of the noise alongside the state estimates.

The augmented state vector $\mathbf{X}_a$ is defined as

$$\mathbf{X}_{a_k} = [\hat{\mathbf{X}}_k \; \tilde{\mathbf{W}} \; \mathbf{V}]^{\mathrm{T}} \tag{4.37}$$

where $\hat{\mathbf{X}}_k = [\hat{x}, \hat{u}, \hat{a}_x, \hat{z}, \hat{w}, \hat{a}_z, \hat{\theta}, \hat{q}, \hat{\dot{q}}, \hat{y}, \hat{v}, \hat{a}_y, \hat{\psi}, \hat{p}, \hat{\dot{p}}, \hat{\phi}, \hat{r}, \hat{\dot{r}}]^{\mathrm{T}}$ is the estimated state of the aircraft and the state elements are as mentioned in the previous chapter section 3.3.2. The process noise is $\tilde{\mathbf{W}}$ and the measurement noise is $\mathbf{V}$. The total dimension of the augmented state vector is $n_a = n_x + n_w + n_v$ where n_x, n_w and n_v are the dimensions of the state-estimate, process noise and measurement noise vectors respectively. The augmented covariance matrix is defined as:

$$\mathbf{S}_{a_k} = \begin{bmatrix} \mathbf{S}_k & 0 & 0 \\ 0 & \tilde{\mathbf{Q}} & 0 \\ 0 & 0 & \mathbf{R} \end{bmatrix} \tag{4.38}$$

where $\mathbf{S}_k$, $\tilde{\mathbf{Q}}$ and $\mathbf{R}$ are the error, process noise and measurement noise covariances respectively.

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \tilde{\mathbf{Q}}_\sigma \frac{\sigma_{a_x}}{\Delta T} & 0 & 0 & 0 & 0 & 0 \\ 0 & \tilde{\mathbf{Q}}_\sigma \frac{\sigma_{a_y}}{\Delta T} & 0 & 0 & 0 & 0 \\ 0 & 0 & \tilde{\mathbf{Q}}_\sigma \frac{\sigma_{a_z}}{\Delta T} & 0 & 0 & 0 \\ 0 & 0 & 0 & \tilde{\mathbf{Q}}_\sigma \frac{\sigma_{\dot{p}}}{\Delta T} & 0 & 0 \\ 0 & 0 & 0 & 0 & \tilde{\mathbf{Q}}_\sigma \frac{\sigma_{\dot{q}}}{\Delta T} & 0 \\ 0 & 0 & 0 & 0 & 0 & \tilde{\mathbf{Q}}_\sigma \frac{\sigma_{\dot{r}}}{\Delta T} \end{bmatrix} \tag{4.39}$$

where $\tilde{\mathbf{Q}}_\sigma$ is

$$\tilde{\mathbf{Q}}_\sigma = \begin{bmatrix} \frac{\Delta T^5}{20} & \frac{\Delta T^4}{8} & \frac{\Delta T^3}{6} \\ \frac{\Delta T^4}{8} & \frac{\Delta T^3}{3} & \frac{\Delta T^2}{2} \\ \frac{\Delta T^3}{6} & \frac{\Delta T^2}{2} & \Delta T \end{bmatrix} \tag{4.40}$$

$$\mathbf{R} = \begin{bmatrix} \sigma^2_{\mathrm{IMU}} & 0 & 0 & 0 \\ 0 & \sigma^2_\alpha & 0 & 0 \\ 0 & 0 & \sigma^2_{P_s} & 0 \\ 0 & 0 & 0 & \sigma^2_{P_T} \end{bmatrix} \tag{4.41}$$

The $2n_a$ sigma vectors $\mathbf{X}_{a_k}$ are composed as

$$\mathbf{X}^{(0)}_{a_k} = [\hat{\mathbf{X}}^{\mathrm{T}}_k \; \mathbf{0}_{1\times n_w} \; \mathbf{0}_{1\times n_v}]^{\mathrm{T}} \tag{4.42}$$

$$\mathbf{X}^{(i)}_{a_k} = \mathbf{X}^{(0)}_{a_k} + \sqrt{(n_a+\lambda)\mathbf{S}_{a_k}} \quad i = 1, ..., n_a \tag{4.43}$$

$$\mathbf{X}^{(i)}_{a_k} = \mathbf{X}^{(0)}_{a_k} - \sqrt{(n_a+\lambda)\mathbf{S}_{a_k}} \quad i = (n_a+1), ..., 2n_a \tag{4.44}$$

where the scaling parameter λ is

$$\lambda = (n_a + \kappa)\zeta^2 - n_a \tag{4.45}$$

where ζ defined the spread of the sigma points from $\mathbf{X}_{a_k}^{(0)}$ and κ. To clarify, the structure of the i^{th} sigma point is $\mathbf{X}_{a_k}^{(i)} = [\mathbf{X}_{a_k^{\hat{X}}}^{\mathrm{T}}, \mathbf{X}_{a^W}^{\mathrm{T}}, \mathbf{X}_{a^V}^{\mathrm{T}}]^{\mathrm{T}}$ where $\mathbf{X}_{a_k^{\hat{X}}}^{\mathrm{T}}$ is the current state-estimate of the aircraft, $\mathbf{X}_{a^W}^{\mathrm{T}}$ is the process noise component of the sigma point and $\mathbf{X}_{a^V}^{\mathrm{T}}$ is the sensor noise component of the sigma point.

The Weightings W_m and W_c are composed as:

$$W_{m_0} = \frac{\lambda}{n_a + \lambda} \tag{4.46}$$

$$W_{c_0} = (1 - \zeta^2 + \xi) + W_{m_0} \tag{4.47}$$

$$W_{m_i} = W_{c_i} = \frac{1}{2(n_a + \lambda)} \quad i = 1, ..., 2n_a \tag{4.48}$$

Time update

The sigma points are propagated using the Navion dynamics function $f_{\text{navion}}(...)$ (represented by algorithm 3 in Appendix A)

$$\mathbf{X}_{a_{k+1|k}^{\hat{X}}}^{(i)} = f_{\text{navion}}(\mathbf{X}_{a_k^{\hat{X}}}^{(i)}, \mathbf{X}_{a_k^W}^{(i)}, \eta_k) \quad i = 1, ..., 2n_a \tag{4.49}$$

where $\mathbf{X}_{a_k^{\hat{X}}}^{(i)}$ is the current state-estimate of the aircraft, $\mathbf{X}_{a_k^W}^{(i)}$ is the process noise and $\eta_k = [\delta_a, \delta_e, \delta_r, T]^{\mathrm{T}}$ is the control vector. The aircraft state estimate is given by

$$\hat{\mathbf{X}}_{a_{k+1|k}^{\hat{X}}} = \sum_{i=0}^{2n_a} W_{m_i} \mathbf{X}_{a_{k+1|k}^{\hat{X}}}^{(i)} \tag{4.50}$$

where $\hat{\mathbf{X}}_{a_{k+1|k}^{\hat{X}}}$ is the current state-estimate of the aircraft. The error covariance of the estimate is given by

$$S_{xx_{k+1|k}} = \sum_{i=0}^{2n_a} W_{c_i} [\hat{\mathbf{X}}_{a_{k+1|k}^{\hat{X}}} - \mathbf{X}_{a_{k+1|k}^{\hat{X}}}^{(i)}][\hat{\mathbf{X}}_{a_{k+1|k}^{\hat{X}}} - \mathbf{X}_{a_{k+1|k}^{\hat{X}}}^{(i)}]^T \tag{4.51}$$

Measurement update

The update stage begins by determining the expected measurements of the time updated sigma points $\mathbf{X}_{a_{k+1|k}^{\hat{X}}}^{(i)}$. The sigma points are input into the sensor measurement functions defined in section 4.1.

The expected measurement for the IMU is determined by equation (4.1)

$$\mathbf{Y}_k^{(\text{IMU},i)} = h^{(\text{IMU})}([a_z^{(i)}, a_y^{(i)}, a_z^{(i)}, p^{(i)}, q^{(i)}, r^{(i)}]^{\text{T}}, \mathbf{X}^{(i)}_{a^{\mathbf{v}_{\text{IMU}}}_{k+1|k}}) \quad i = 0, ..., 2n_a \tag{4.52}$$

where $\mathbf{Y}_k^{(\text{IMU},i)}$ is the expected measurement of the IMU sensor for the i^{th} sigma point, the elements $a_x^{(i)}$, $a_y^{(i)}$, $a_z^{(i)}$ $p^{(i)}$ $q^{(i)}$ and $r^{(i)}$ are obtained from the $\mathbf{X}^{(i)}_{a^{\mathbf{x}}_{k+1|k}}$ vector and $\mathbf{X}^{(i)}_{a^{\mathbf{v}_{\text{IMU}}}_{k+1|k}}$ are the elements of the sigma point representing the IMU measurement noise. The expected measurement for the AoA is determined by equation (4.3)

$$Y_k^{(\text{AoA},i)} = h^{(\text{AoA})}(u^{(i)}, w^{(i)}, \mathbf{X}^{(i)}_{a^{V_{\text{AoA}}}_{k+1|k}}) \quad i = 0, ..., 2n_a \tag{4.53}$$

where $Y_k^{(\text{AoA},i)}$ is the expected measurement of the AoA sensor for the i^{th} sigma point, the elements $u^{(i)}$ and $w^{(i)}$ are the elements obtained from the $\mathbf{X}^{(i)}_{a^{\mathbf{x}}_{k+1|k}}$ vector and $\mathbf{X}^{(i)}_{a^{V_{\text{AoA}}}_{k+1|k}}$ are the elements of the sigma point representing the AoA measurement noise.

$$Y_k^{(P_s,i)} = h^{(P_s)}(z^{(i)}, \mathbf{X}^{(i)}_{a^{V_{P_s}}_{k+1|k}}) \quad i = 0, ..., 2n_a \tag{4.54}$$

where $Y_k^{(P_s,i)}$ is the expected measurement of the static pressure from the pitot static sensor for the i^{th} sigma point, the element $z^{(i)}$ is obtained from the $\mathbf{X}^{(i)}_{a^{\mathbf{x}}_{k+1|k}}$ vector and $\mathbf{X}^{(i)}_{a^{V_{P_s}}_{k+1|k}}$ are the elements of the propagated sigma point representing the static pressure measurement noise.

$$Y_k^{(P_T,i)} = h^{(P_T)}(u^{(i)}, v^{(i)}, w^{(i)}, z^{(i)}, \mathbf{X}^{(i)}_{a^{V_{P_T}}_{k+1|k}}) \quad i = 0, ..., 2n_a \tag{4.55}$$

where $Y_k^{(P_T,i)}$ is the expected measurement of the total pressure from the pitot static sensor for the i^{th} sigma point, the elements $u^{(i)}$, $v^{(i)}$, $w^{(i)}$ and $z^{(i)}$ are obtained from the $\mathbf{X}^{(i)}_{a^{\mathbf{x}}_{k+1|k}}$ vector and $\mathbf{X}^{(i)}_{a^{V_{P_T}}_{k+1|k}}$ are the elements of the propagated sigma point representing the total pressure measurement noise.

The expected measurement $\hat{\mathbf{Y}}_{k+1}$ is composed of the expected measurements obtained from each sensor. This is

$$\hat{\mathbf{Y}}_{k+1}^{(i)} = [\mathbf{Y}_k^{(\text{IMU},i)^{\text{T}}}, Y_k^{(\text{AoA},i)}, Y_k^{(P_s,i)}, Y_k^{(P_T,i)}]^{\text{T}} \quad i = 0, ..., 2n_a \tag{4.56}$$

$$\hat{\mathbf{Y}}_{k+1} = \sum_{i=0}^{2n_a} \mathbf{W}_{m_i} \mathbf{Y}_{k+1}^{(i)} \tag{4.57}$$

where $\hat{\mathbf{Y}}_{k+1}$ is the estimated measurement of the sensors. The measurement covariance is calculated by

$$S_{yy_{k+1}} = \sum_{i=0}^{2n_a} \mathbf{W}_{c_i} [\hat{\mathbf{Y}}_{k+1} - \mathbf{Y}_{k+1}^{(i)}][\hat{\mathbf{Y}}_{k+1} - \mathbf{Y}_{k+1}^{(i)}]^T \tag{4.58}$$

The cross covariance between the estimated state and measurement estimate is calculated

$$S_{xy_{k+1}} = \sum_{i=0}^{2n_a} \mathbf{W}_{c_i} [\hat{\mathbf{X}}_{k+1|k} - \mathbf{X}^{(i)}_{a^{\hat{X}}_{k+1|k}}][\hat{\mathbf{Y}}_{k+1} - \mathbf{Y}^{(i)}_{k+1}]^T \tag{4.59}$$

where $S_{xy_{k+1}}$ and $S_{yy_{k+1}}$ are the measured state and innovation covariances respectively. The Kalman Gain K is calculated as

$$K = S_{xy_{k+1}} S^{-1}_{yy_{k+1}} \tag{4.60}$$

The state-estimate $\hat{\mathbf{X}}_{k+1|k+1}$ and error covariance $S_{xx_{k+1|k+1}}$ are updated using Kalman gain K. The measurement innovation is

$$\epsilon = \mathbf{Y}_k - \hat{\mathbf{Y}}_{k+1} \tag{4.61}$$

where Y_k is the measurement obtained from the sensor. The state-estimate is updated

$$\hat{\mathbf{X}}_{k+1|k+1,y_k} = \hat{\mathbf{X}}_{k+1|k} + K\epsilon \tag{4.62}$$

The error covariance is updated

$$S_{xx_{k+1|k+1,y_k}} = S_{xx_{k+1}} + K S_{yy_{k+1}} K^T \tag{4.63}$$

The updated state-estimate $\hat{\mathbf{X}}_{k+1|k+1}$ obtained from the UKF algorithm is provided to the autopilot.

4.3 Simulation

The sensors are added to the flight model and tested by simulating a series of common flight scenarios as presented in the previous chapter. The Navion SixDoF flight model is assumed to have 3 Pitot-static, 2 Angle-of-attack and an Inertial Measurement Unit. Noisy sensor data is generated by simulation. The Augmented Unscented Kalman Filter is used filter the noise and provide a state-estimate to the autopilot. Figure 4.6 illustrates simulation flow diagram where $\hat{X}_0$ is the initial state estimate of the aircraft, $\mathbf{X}_0$ is the initial state of the aircraft, $\mathbf{C} = [\psi_c, z_c]^{\mathrm{T}}$ is the control input to the autopilot and η_{k+1} are the controls from the autopilot to the aircraft.

The four common flight scenarios are simulated with aircraft's initial conditions of an altitude of 5000 ft (1524m), heading 000° and speed of 150 knots(77.2ms^{-1}). The simulation is initialised with the following parameters:

- True state of aircraft $\mathbf{X}_0 = [0, 77.2, 0, 1524, \mathbf{0}_{1\times 14}]^{\mathrm{T}}$
- Aircraft controls at trimmed state $\boldsymbol{\eta} = [0,\ -2.3°,\ 0,\ 3600\text{N}\ ]^{\mathrm{T}}$

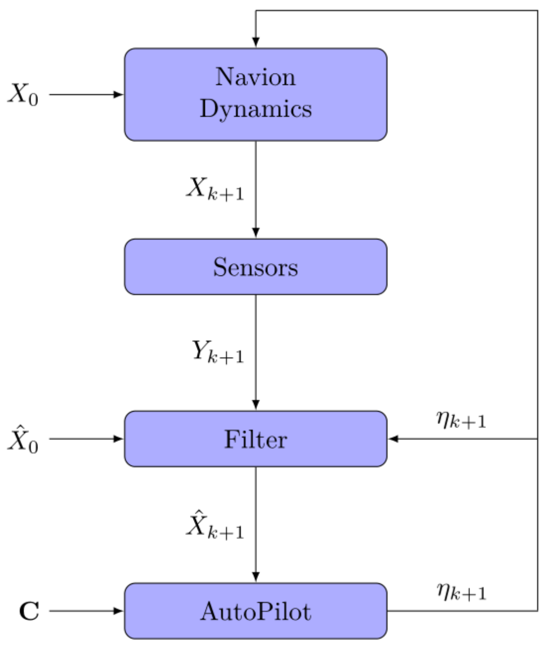


Figure 4.6: Simulation Flow diagram

- Initial aircraft estimate $\hat{\mathbf{X}}_0$: initialised by three sets of measurements obtained from the Pitot-static, Angle-of-Attack and Inertial Measurement Unit sensors.
- Inertial Measurement Unit [107]

  $\mathbf{V}_{\text{IMU}} \sim \mathcal{N}(0, \sigma^2_{\text{IMU}})$

$$\sigma^2_{\text{IMU}} = \begin{bmatrix} 0.2^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01^2 \end{bmatrix} \tag{4.64}$$

- Angle of attack sensor [108]

  $V_\alpha \sim \mathcal{N}(0, 0.25°)$

- Pitot-static sensor [75]

  $V_{P_s} \sim \mathcal{N}(0, 0.1)$

  $V_{P_T} \sim \mathcal{N}(0, 0.1)$

  $b = 0.01$

- Process noise parameters

$$\sigma_{ax} = 0.1686$$

$$\sigma_{ay} = 0.175$$

$$\sigma_{\dot{q}} = 0.0321$$

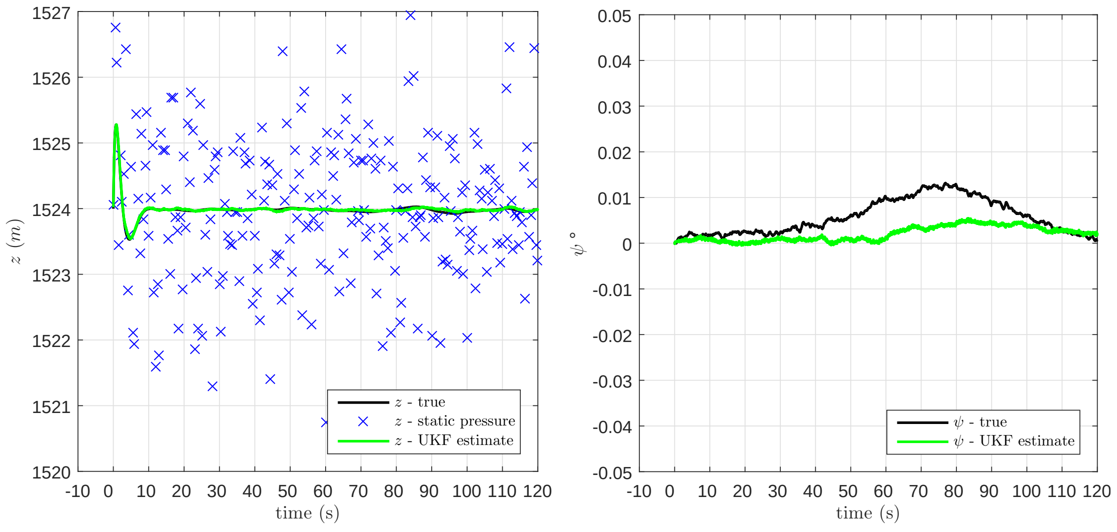
$$\sigma_{az} = 0.1686$$

$$\sigma_{\dot{p}} = 0.0683$$

$$\sigma_{\dot{r}} = 0.0026$$

4.3.1 Level flight

Figure 4.7 shows the altitude and heading of the aircraft during the level flight test. The aircraft maintains an altitude of 5000ft (1524m) (as shown in Figure 4.7a) and a heading of 0° as expected (as shown in Figure 4.7b).

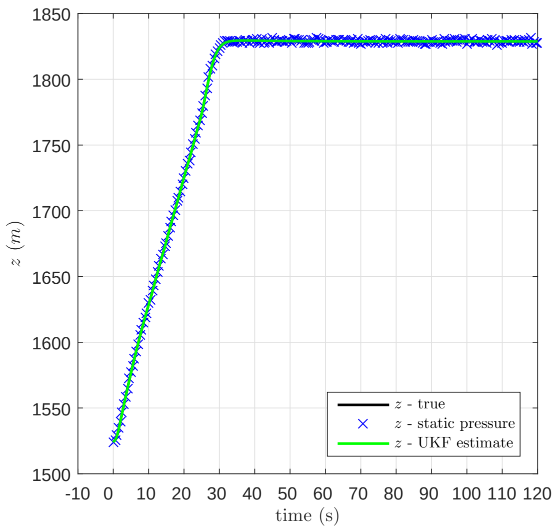


(a) Altitude Z_e (b) Heading ψ

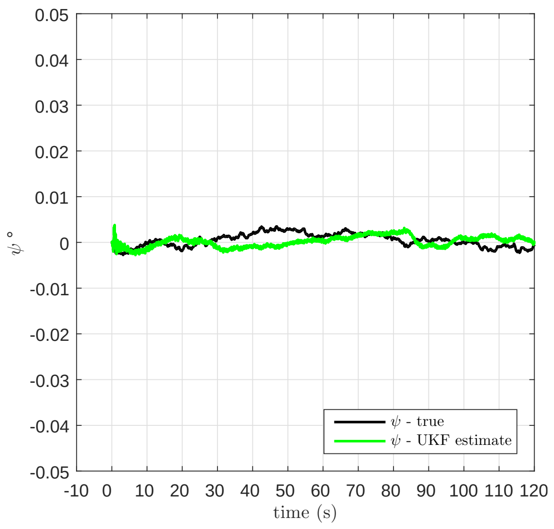
Figure 4.7: Altitude and Heading during level flight

4.3.2 Climbing flight

Figure 4.8 shows the altitude and heading of the aircraft during the climbing flight test. The aircraft climbs to and maintains an altitude of 6000ft (1828.8m) as expected (as shown in Figure 4.8a). The aircraft maintains a heading of 0° as expected (as shown in Figure 4.8b).



(a) Altitude Z_e

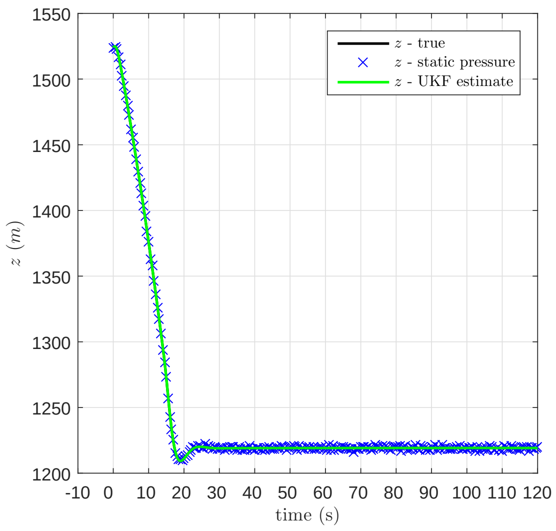


(b) Heading ψ

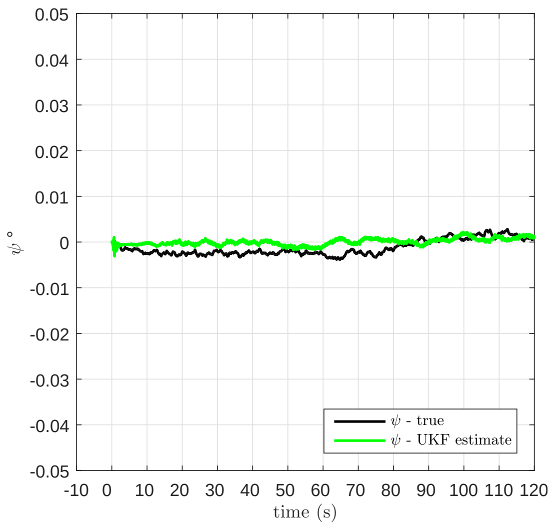
Figure 4.8: Altitude and Heading during climbing flight

4.3.3 Descending flight

Figure 4.9 shows the altitude and heading of the aircraft during the descending flight test. The aircraft descends to and maintains an altitude of 4000ft (1219.2m) as expected (as shown in Figure 4.9a). The aircraft maintains a heading of 0° as expected (as shown in Figure 4.9b).



(a) Altitude Z_e

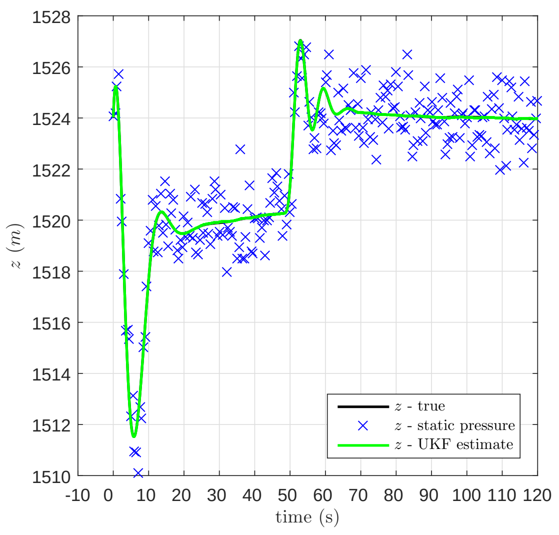


(b) Heading ψ

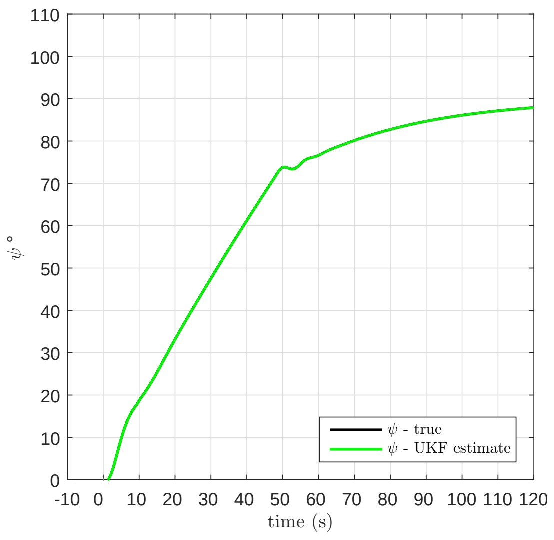
Figure 4.9: Altitude and Heading during Descending flight

4.3.4 Turning flight

Figure 4.10 shows the altitude and heading of the aircraft during the turning flight test. The aircraft maintains an altitude of 5000ft (1524m) (as shown in Figure 4.10a) and turns towards a heading of 090° as expected (as shown in Figure 4.10b).



(a) Altitude Z_e



(b) Heading ψ

Figure 4.10: Altitude and Heading during turning flight

During the four test scenarios the aircraft is able to successfully reach and maintain the desired flight conditions as specified by the autopilot.

4.4 Summary

This chapter has defined sensor models that simulate noisy measurements based on the Navion flight model developed in the previous chapter. The noisy measurements are filtered using the Augmented Unscented Kalman Filter to maintain an accurate state-estimate of the aircraft. The state-estimate of the aircraft is input into the autopilot and a series of common flight scenarios are demonstrated successfully. A Six-Degrees-of-Freedom flight model for a Navion aircraft with sensors and an Autopilot has been developed to be used as a test bed.

Chapter 5

Sensor Fault Detection

The ability to maintain accurate awareness of the current state of the aircraft and its immediate environment is a crucial requirement for Unmanned Aircraft Systems (UAS). This awareness is dependent on data obtained from sensors on-board the UAS. Faulty sensor data leads to a false sense of awareness of the state of the aircraft and this causes incorrect actions to be commanded. This leads to a failure in accomplishing the mission, or worse results in the loss of the aircraft which could cause collateral damage. The absence of flight crew on-board UAS requires a robust fault detection system, since there can be no manual override on-board the aircraft.

A popular fault detection method is the use of stochastic state estimation methods where the innovation is used as residual for fault detection [73]. Initial fault detection and isolation using the innovation from the Kalman Filter was proposed by Mehra and Peschon [109], where faults were detected by statistical testing of the mean and covariance. For example, in normal (fault-free) conditions, the innovation of the Kalman Filter is expected to be white noise with zero-mean and known covariance [73, 109].

The Multiple Model Adaptive Estimation (MMAE) is an extension of this approach where a bank of Kalman filters are run in parallel and each filter represents a hypothesis for the behaviour of the system being estimated [72]. Each filter is based on the expected behaviour of the system when a fault has occurred or has not occurred. The measurement obtained from a sensor is used to determine the innovation for each filter. The innovations of each of the filters are used to determine the probabilities that each model matches the current state of the system. This is modelled on a particular type of fault and is used to deduce whether a particular fault has occurred and the confidence for that fault is the probability determined for the model. The overall state-estimate of the system is the weighted sum of the estimates output by each filter in the bank. The weights are proportional to the filter probability [79]. Each filter is initialised and re-initialised using its own estimate and each filter is running independently and there is no interaction between filters. This approach has been found to be suitable for fault detection problems with unknown structure or parameter [79]. But this is not suitable for FDD scenarios where the system structure or parameter changes. The lack of interaction between the models causes faults to be miss-identified and/or false alarms due to degradation of the state-estimate [79]. This problem is addressed by allowing interaction between the models. In its simplest form, each model is re-initialised at every time-step by the overall state-estimate obtained from the previous iteration [79]. This is known as the Generalized Pseudo Bayes-1 [110]. A more complex form of this is the Interacting Multiple Model (IMM) that 'switches' between models in a probabilistic manner [79]. The 'switching' feature preserves the accuracy of the estimate since the filter switches to the model that best represents the current state of the system due to the change in system structure and/or parameter changes.

For nonlinear problems the Kalman Filter is known to be non-optimal and the EKF and UKF have been used instead. For example, Cork and Walker have implemented a fault detection method for detecting IMU faults on-board an aircraft using an IMM [111]. The nonlinear dynamics of the aircraft were estimated using a bank of UKFs. The filters represent the behaviour of different combinations of IMU failures. To clarify, an IMU consists of six sensors (three accelerometers and three gyroscopes as described in chapter 4). There are 2^6 different combinations of sensor faults. This is modelled by a combination of 64 parallel UKF filters that represent the hypotheses of the combination of different failures. The increased number of sensors incurs increased computational load to model all possible failures [111]. If additional sensors are included, such as the Angle-of-Attack and Pitot-static sensors considered in the previous chapter, this will result in increased computational load due to the increased number of filters ($2^8 = 256$).

Another approach is to evaluate the hypothesis of each sensor in sequence. This reduces the number of models required for detecting multiple sensor faults. For situations where the hypotheses for a large number of sensors is required; particle filters are suitable to evaluate the sensor hypotheses in parallel. Cheng Qi *et al.* present a distributed fault detection method using particle filters based on state estimation theory for multiple sensor platforms [112]. Each detector for each sensor makes a local decision for the individual sensor. Each sensor on the platform evaluates a hypothesis based on its observations to determine the likelihood that it is operating in normal or faulty mode. The local decision obtained from each detector attached to the sensor is passed to the global decision centre, where all decisions are fused to carry out a global decision.

Liu *et al.* have proposed multiple sensor fault detection for surveillance in Air Traffic Control using hybrid estimation algorithms [113]. The two types of sensors are considered in this work are Automatic Dependent Surveillance - Broadcast (ADS-B) [49] and Multilateration [114,115]. Multilateration is a form of navigation that uses radio beacons to determine the position of the aircraft based on the time difference of arrival (TDOA). Successful implementation of Multilateration are LORAN [116] and OMEGA [113]. Both ADS-B and Multilateration sensors in reference [113] have an estimation algorithm running in parallel, the estimate obtained from each sensor is fused together using a hybrid algorithm inspired by the IMM algorithm [93]. The proposed algorithm was tested with a linearised aircraft model in common flight modes. These modes are: coordinated turn, constant velocity and constant descent. This method's feature of running estimation algorithms in parallel might not be feasible for an increased number of sensors because this will increase the amount of time required to update the state-estimate.

Here, an approach is adopted where each sensor's data is considered in sequence using a Generalized Pseudo Bayes-1 (GPB-1) method which consider the hypotheses for each sensor in sequence. This approach is likely to reduce the accuracy relative to an IMM, but the use of a GPB-1 offers a compromise between performance and computational expense [93].

This chapter begins by defining the fault simulation model for each sensor on-board the Navion aircraft model mentioned in chapter 4. A multiple sensor fault detection method using Generalized Pseudo Bayes-1 (GPB-1) is proposed which updates the state-estimate of the aircraft with measurements obtained from each sensor sequentially in section 5.2. The fault detection method is tested by simulating a series of common flight modes in section 5.3. The accuracy and time taken to update the state-estimate is compared with a parallel implementation of the GPB-1 multiple sensor fault detection method in section 5.4.

5.1 Sensor Fault simulation

5.1.1 Inertial Measurement Unit

The fault mode of this sensor is simulated by zeroing the acceleration and rotational rates of the sensor and adding extra noise as shown in [111]. This is a standard form of fault simulation.

$$Y_k^{(\mathrm{IMU}*)} = V_{\mathrm{IMU}*} \tag{5.1}$$

This represents the IMU measuring $a_x = a_y = a_z = p = q = r = 0$ and $V_{\mathrm{IMU}*} >> V_{\mathrm{IMU}}$. The '*' symbol indicates the faulty sensor measurements.

5.1.2 Angle-of-Attack

This sensor is susceptible to faults such as freezing due to adverse weather conditions. One fault mode is where the sensor is frozen. In this mode, the vane will be in a stuck position despite a change in the Angle-of-Attack of the aircraft. This type of fault is simulated as

$$Y_k^{(\alpha *)} = \alpha_F + V_{\alpha *} \tag{5.2}$$

where α_F is the expected value of the Angle-of-Attack α (chapter 4) when the fault event occurs and $V_{\alpha *} >> V_\alpha$. Another fault mode is when the sensor is partially frozen. This affects the angular rate of rotation of the vane by a delay. This type of fault can be simulated using

$$Y_k^{(\alpha *)} = \alpha_{k-1} + \frac{\Delta t}{c}(\alpha_k - \alpha_{k-1}) + V_{\alpha *} \tag{5.3}$$

where c is a delay constant and Δt is the period of the time step between consecutive states. This fault replicates a delay effect due to partial freezing.

5.1.3 Pitot-static Sensor

This sensor is susceptible to blockages caused by debris and ice during operations in adverse weather conditions. Such blockages causes the sensor to output faulty measurements. The data output from a system with a blocked pitot inlet is an abrupt increase or decrease in pressure due to the blockage.

$$Y_k^{(P_T *)} = P_T + V_{P_T *} + b \tag{5.4}$$

$$Y_k^{(P_s *)} = P_s + V_{P_s *} + b \tag{5.5}$$

where b is the bias [75]. A partially blocked pitot inlet and a static source will show gradual variations in speed, it is simulated using the equation

$$Y_k^{(P_T \sim)} = P_T + V_{P_T *} + b\sin(2\pi f t) \tag{5.6}$$

$$Y_k^{(P_s \sim)} = P_s + V_{P_s *} + b\sin(2\pi f t) \tag{5.7}$$

where f is the frequency of the time-varying effect in the partial fault mode. This replicates the effect of water and ice crystals partially blocking the pitot tube within the sensors and is based on reference [75]. $V_{P_T *} >> V_{P_T}$ and $V_{P_s *} >> V_{P_s}$

Table 5.1: Simulated Sensor Faults

Event ID i	Sensor index j	Sensor type	Fault type e	Fault equation(s)	Start Time (**s**) t_s
1	1	Pitot	Partial Block	5.6,5.7	1
2	4	AoA	Partial Block	5.3	1.5
3	2	Pitot	Partial Block	5.6,5.7	1.5
4	5	AoA	Partial Block	5.3	2.5
5	3	Pitot	Partial Block	5.6,5.7	3
6	1	Pitot	Block	5.4,5.5	2
7	4	AoA	Block	5.3	3
8	2	Pitot	Block	5.4,5.5	2.5
9	5	AoA	Block	5.3	3.5
10	3	Pitot	Block	5.4,5.5	4
11	6	IMU	fault	5.1	12

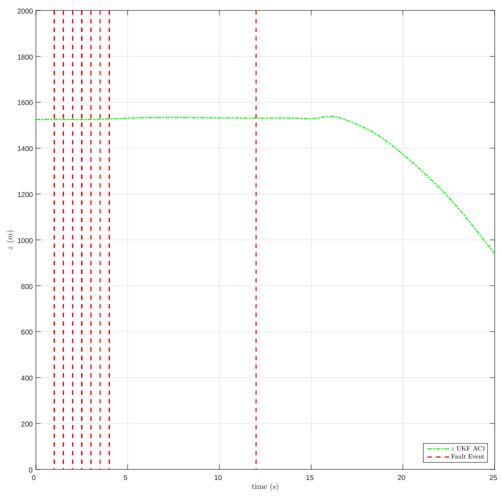
5.1.4 Faulty sensor test scenarios

The problem of faulty sensor data misleading the aircraft's autopilot is demonstrated by simulating the sensor faults shown in Table 5.1. The faults are simulated during the four common (Level, Climbing, Descending and Turning) flight modes demonstrated in the previous chapters using the Navion SixDoF simulation.

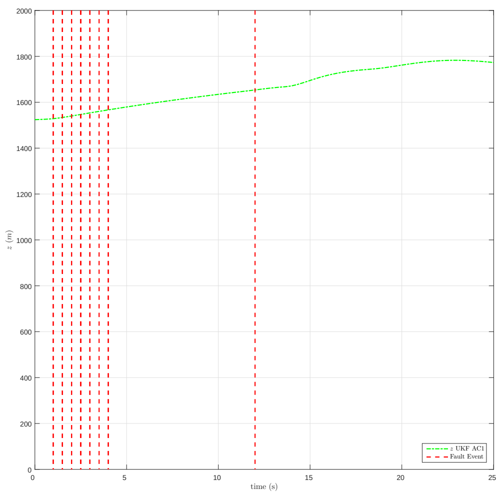
Figure 5.1 shows the altitude of the aircraft (AC1) during the four common flight scenarios. Figure 5.1(e) shows the heading during the turning flight scenario. The vertical red lines represents the instant that a sensor fault occurs and corresponds to the faults in Table 5.1. In all four scenarios the aircraft is unable to achieve the autopilot's target state. This is because faulty sensor data corrupt the state-estimate of the aircraft.

5.2 Multiple sensor fault detection using sequential UKF Generalized Pseudo Bayes-1

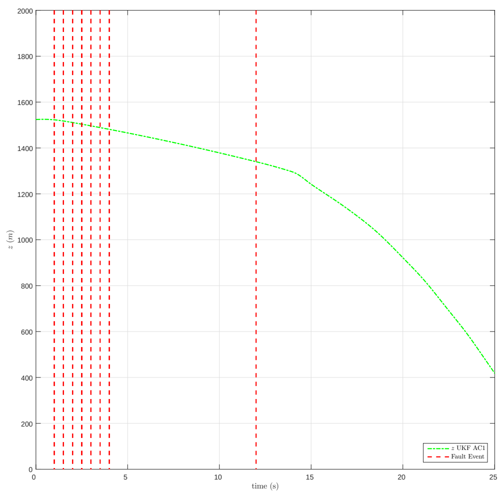
To prevent the state-estimate from being corrupted, the hypotheses that the sensor is operating normally or faulty must be verified from a series of measurements obtained from the sensor. The Generalized Pseudo Bayes-1 approach to detect a faulty sensor is to calculate the likelihood that the measurement is normal (fault-free) L and the likelihood that it is faulty $L^{(*)}$ and update the state-estimate according to the proportion



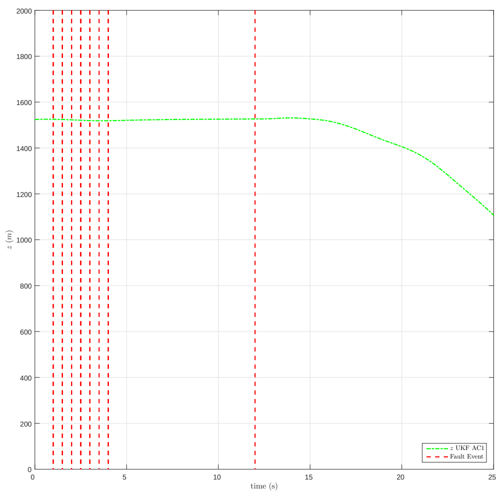
(a) Level Flight



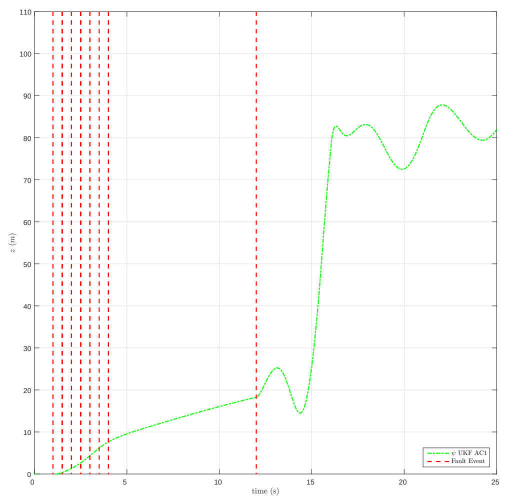
(b) Ascending Flight



(c) Descending Flight



(d) Turning Flight



(e) Turning Flight heading

Figure 5.1: The effect of sensor faults on the altitude of the aircraft during the four common flight modes: Level flight, Ascending flight, Descending flight and Turning flight.

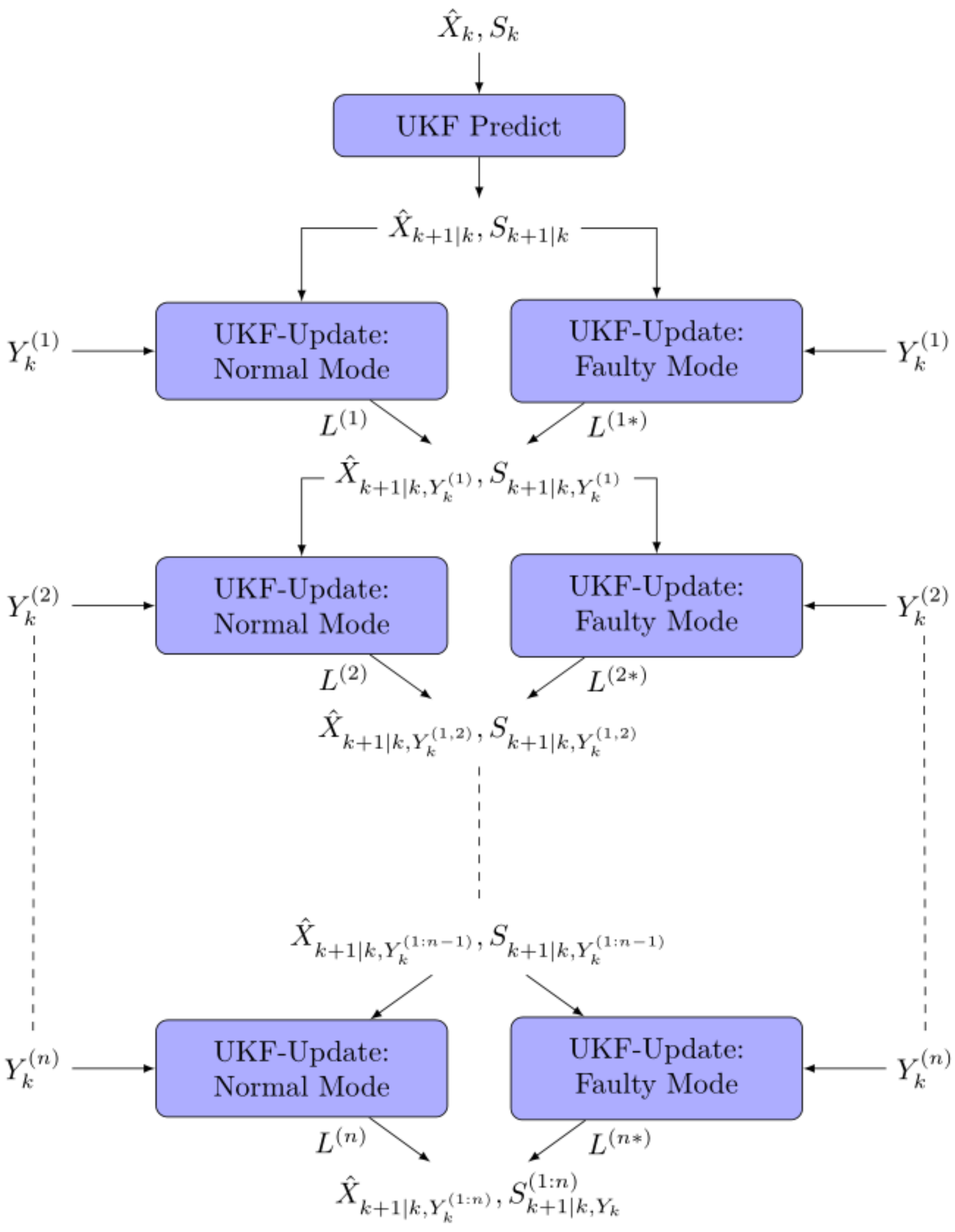


Figure 5.2: Flow diagram for UKF-GPB-1 multiple sensor fault detection evaluating hypotheses sequentially

of the likelihoods [117].

Figure 5.2 shows a flow diagram for the UKF-GPB-1 sequential multiple sensor fault detection process that is used to preserve the accuracy of state-estimate of the aircraft. The process begins with the 'UKF Predict' stage, where the current state-estimate $\hat{\mathbf{X}}_{k+1}$ and error covariance S_{k+1} are time updated to $\hat{X}_{k+1|k}$ and $S_{xx_{k+1|k}}$ respectively as defined by equation (4.49) in the previous chapter. The measurement update stage consists of two update models – one represents the hypothesis that the sensor is operating normally and is shown as 'UKF-Update: Normal Mode' the other represents the hypothesis that it is faulty and is shown as 'UKF-Update: Faulty Mode'. Both models update the state-estimate and error covariance with the measurement obtained from the sensor. The state-estimate and error covariance output from the normal model is $\hat{\mathbf{X}}^{(j)}_{k+1|k+1}$, and $S^{(j)}_{xx_{k+1}}$ respectively. The state-estimate and error covariance output from the faulty model is $\hat{\mathbf{X}}^{(j*)}_{k+1|k+1}$ and $S^{(j*)}_{xx_{k+1}}$ respectively. The estimates and covariances from both models are fused proportional to the likelihood that the sensor is operating normally or faulty. The likelihood is determined by evaluating the probability density of the sensor measurement within a Gaussian approximated probability distribution defined by the estimated measurement and measurement covariance determined during the update stage. The likelihood $L^{(j)}$ for 'UKF-Update: Normal Mode' is determined by

$$L^{(j)} = p(\mathbf{Y}^{(j)}_{k} | \hat{\mathbf{Y}}^{(j)}_{k+1}, S^{(j)}_{yy_{k+1}}) \tag{5.8}$$

where $\mathbf{Y}^{(j)}_{k}$ is the measurement obtained from the j^{th} sensor. The estimated measurement is $\hat{\mathbf{Y}}^{(j)}_{k+1}$ and the measurement covariance $S^{(j)}_{yy_{k+1}}$ is calculated during the normal sensor update model by using equations (4.57) and (4.58) respectively. The likelihood $L^{(j*)}$ for 'UKF-Update: Faulty Mode' is determined

$$L^{(j*)} = p(\mathbf{Y}^{(j)}_{k} | \hat{\mathbf{Y}}^{(j*)}_{k+1}, S^{(j*)}_{yy_{k+1}}) \tag{5.9}$$

where $\hat{\mathbf{Y}}^{(j*)}_{k+1}$ and $S^{(j*)}_{yy_{k+1}}$ are the estimated measurement and measurement covariance calculated during the faulty sensor update by equations (4.57) and (4.58) respectively. The likelihoods are normalised

$$\hat{L}^{(j)} = \frac{L^{(j)}}{L^{(j)} + L^{(j*)}} \tag{5.10}$$

$$\hat{L}^{(j*)} = \frac{L^{(j*)}}{L^{(j)} + L^{(j*)}} \tag{5.11}$$

The normalised likelihoods are used to indicate whether the sensor that provided the measurement $\mathbf{Y}^{(j)}$ is normal or faulty. When $\hat{L}^{(j)} > \hat{L}^{(j*)}$ the sensor is operating normally and when $\hat{L}^{(j*)} > \hat{L}^{(j)}$ the sensor is faulty [73]. In practice, when $\hat{L}^{(j*)} > 0$

this indicates a faulty sensor. The state-estimate and error covariance obtained from both models are fused

$$\hat{\mathbf{X}}_{k+1|k+1} = a\hat{L}^{(j)}\hat{\mathbf{X}}^{(j)}_{k+1|k+1} + (1-a)\hat{L}^{(j*)}\hat{\mathbf{X}}^{(j*)}_{k+1|k+1} \tag{5.12}$$

$$D_1 = [\hat{\mathbf{X}}^{(j)}_{k+1|k+1} - \hat{\mathbf{X}}_{k+1|k+1}][\hat{\mathbf{X}}^{(j)}_{k+1|k+1} - \hat{\mathbf{X}}_{k+1|k+1}]^T \tag{5.13}$$

$$D_2 = [\hat{\mathbf{X}}^{(j*)}_{k+1|k+1} - \hat{\mathbf{X}}_{k+1|k+1}][\hat{\mathbf{X}}^{(j*)}_{k+1|k+1} - \hat{\mathbf{X}}_{k+1|k+1}]^T \tag{5.14}$$

$$S_{xx_{k+1}} = \hat{L}^{(j)}S^{(j)}_{xx_{k+1}}D_1 + \hat{L}^{(j*)}S^{(j*)}_{xx_{k+1}}D_2 \tag{5.15}$$

where a is the probability that the sensor is functioning normally – this has to be estimated empirically by the sensor manufacturer through rigorous testing [80].

The difference between the normal and faulty update models is the measurement covariance R. The faulty sensor UKF model is assumed to have a significantly larger measurement covariance R^* than the measurement covariance used in the normal UKF ($R^* >> R$). This is to detect the erratic and inconsistent nature of faulty sensor measurements that have a large variance. This can be observed in the black box data recovered from the AF477 crash – that shows large abrupt variations in airspeed prior to the crash [4]. This update process is repeated for each sensor until the n^{th} sensor has been reached as shown in Figure 5.2.

5.3 Results

The UKF-GPB-1 fault detection method is tested for the same flight scenarios with the same faults as the previous section. The aircraft without fault detection is designated AC1 and the aircraft with the GPB-1 sensor fault detector is designated AC2.

5.3.1 Level Flight

Figure 5.3 shows the altitude of both aircraft during the level flight scenario. AC2 compared to AC1 loses altitude at a lower rate after all the sensors have failed. The GPB-1 fault detection method is able to detect the likelihood of faults and preserve the accuracy of the state-estimate.

Figure 5.4 are fault likelihood plots for each sensor on the aircraft during the level flight scenario. The fault likelihood plots are used to detect the occurrence of a fault within a sensor. The sensor's normalised likelihoods for normal mode $\hat{L}^{(j)}$ and faulty mode $\hat{L}^{(j*)}$ are plotted as blue and red lines respectively. This is used to indicate if a fault has been detected in the sensor. A fault is detected when the red line increases

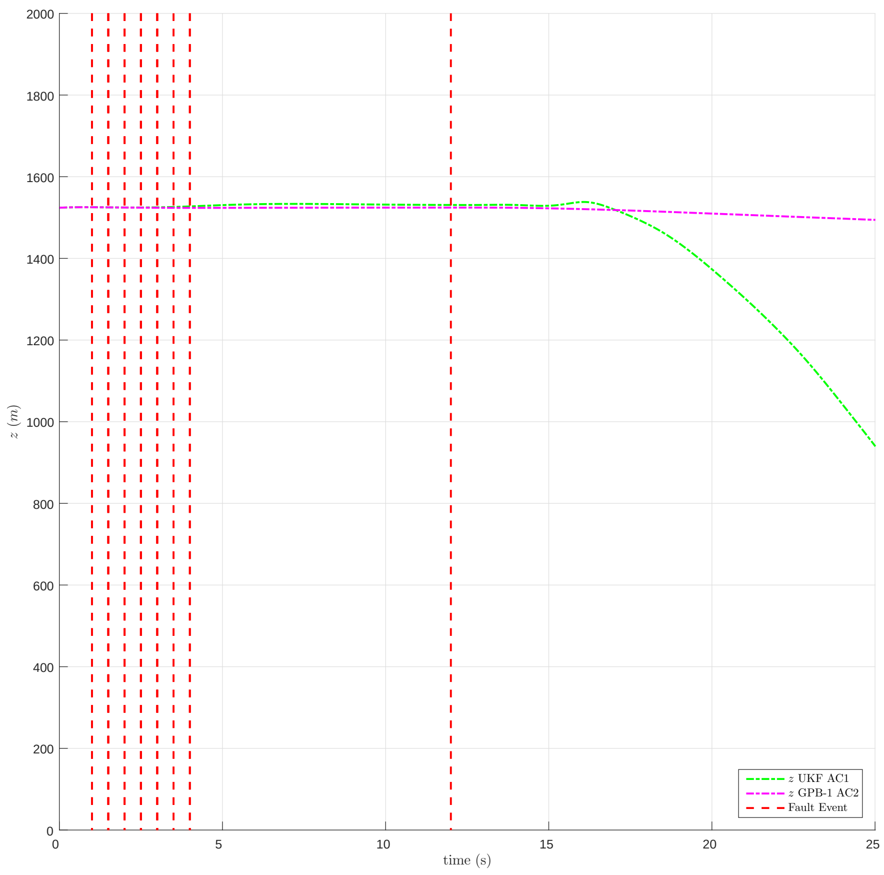
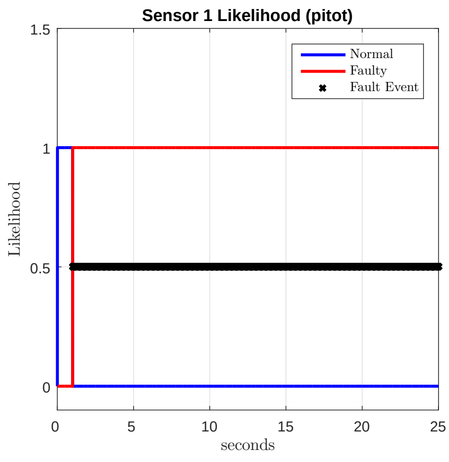
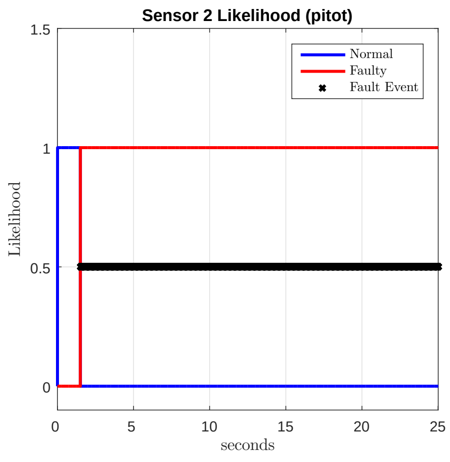


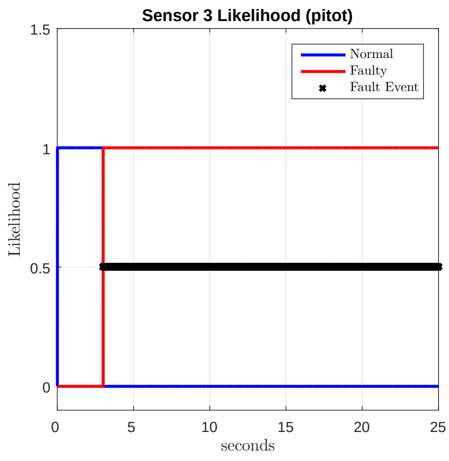
Figure 5.3: Altitude of aircraft using UKF-GPB-1 sensor fault detection during level flight scenario.



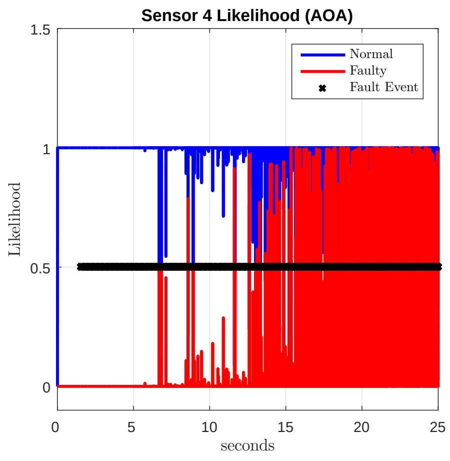
(a)



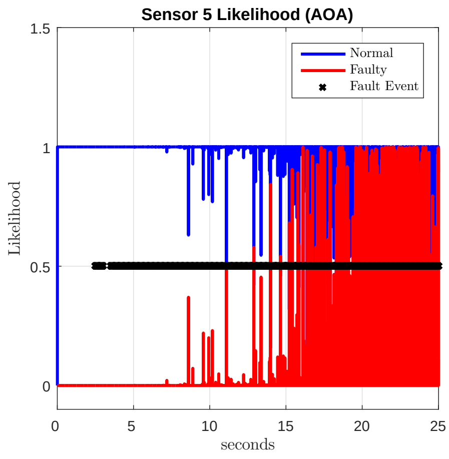
(b)



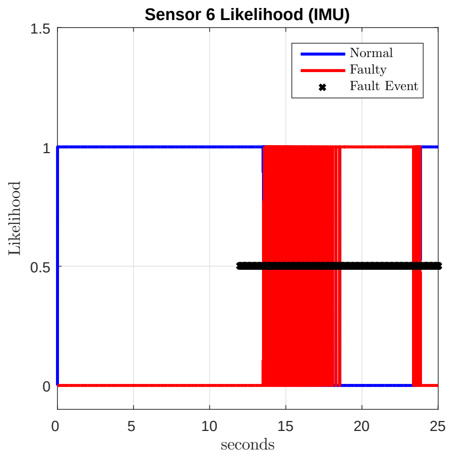
(c)



(d)



(e)



(f)

Figure 5.4: Fault likelihood for sensors during Level Flight

and the blue line decreases. A series of black markers at 0.5 have been plotted to indicate the period when the sensor is producing faulty measurements. The vertical red lines mark the instant a sensor fault event begins as mentioned in Table 5.1.

The fault likelihood plots for the pitot sensors shown in Figures 5.4(a), 5.4(b) and 5.4(c) indicate that the faults are detected promptly. This is because of the nature of the fault encountered – a partially blocked or blocked pitot-static shows an abrupt offset from the expected measurement of a sensor operating normally. This is detected as an outlier of the innovation distribution.

In contrast the fault likelihood plots for the AoA sensors shown in Figures 5.4(d) and 5.4(e) indicate that the AoA faults take time to be detected. This is because the expected AoA measurement is near constant and doesn't change very much during the course of the flight. This is similar to the output of a frozen or partially frozen sensor. It takes time for the faulty sensor measurements to accumulate for it to be detected as an outlier of the innovation distribution. The abrupt fluctuations between faulty and normal are due to the measurements being close to the faulty and normal estimated sensor measurement distribution, where as time increases some measurements are close to both distributions. Similarly, the fault likelihood for the IMU sensor shown in Figure 5.4(f) takes time to be detected since the expected rotational rates (p, q, r) and accelerations (a_x, a_y, a_z) during level flight are close to zero. This coincides with the nature of the fault encountered.

5.3.2 Climbing flight

The climbing flight scenario shows the altitude of both aircraft, climbing at approximately the same rate. AC1 has a brief increase in the climb rate due to the increased error in the state-estimate after the IMU fault event. The final trajectory of AC1 shows the beginning of a descent. In comparison, AC2 shows a slight reduction in the climb rate.

The sensor fault likelihood plots in figure 5.6 shows that all faults are detected for AC2 within the duration that each sensor is faulty. The fault likelihood plots for pitot sensors 2 and 3 shown in Figures 5.6(b) and 5.6(c) respectively, indicate a false alarm at $t = 1$ which coincides with true detection for sensor 1. This is because the aircraft is in transient mode as the aircraft enters the climb. The abrupt change in speed caused an abrupt change in total pressure, resulting in a false alarm.

The fault likelihood plots for AoA Sensors 4 and 5 (AoA) shown in Figures 5.6(d) and Figures 5.6(e) respectively takes time to indicate a fault, this is due to similar reasons as noted in level flight scenario where the AoA is near constant. The fault likelihood plots for the IMU sensor are detected instantaneously since the expected accelerations and rotation rates are near zero. The measurements obtained from the IMU are close to normal and faulty mode distributions where some of the fault ob-

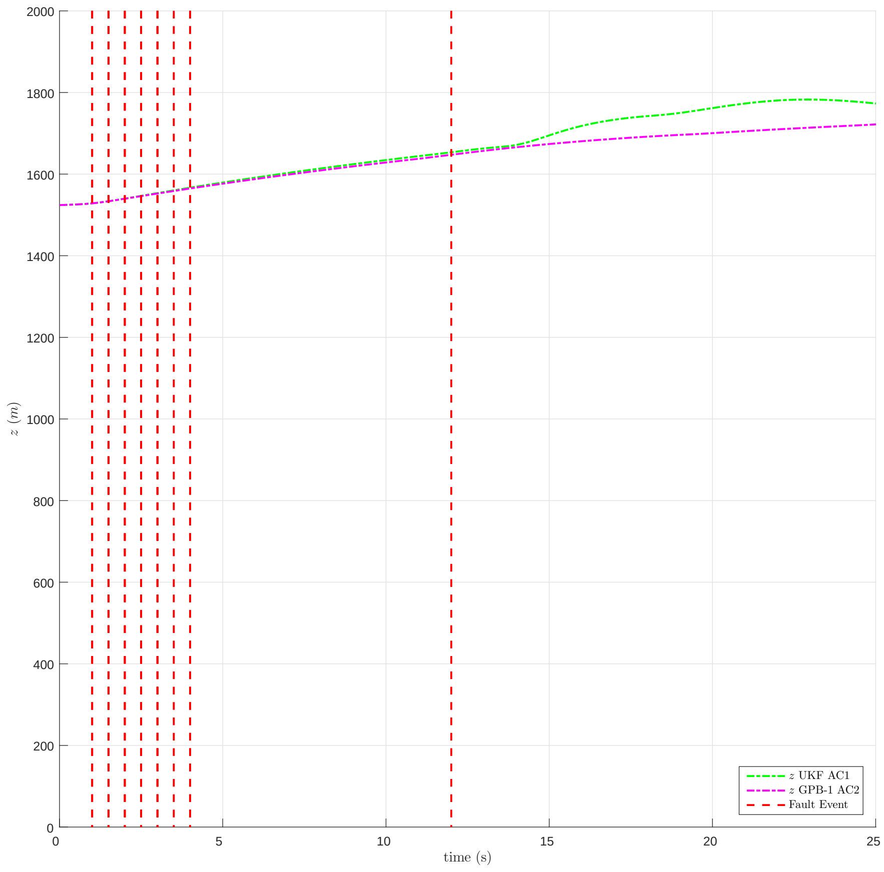
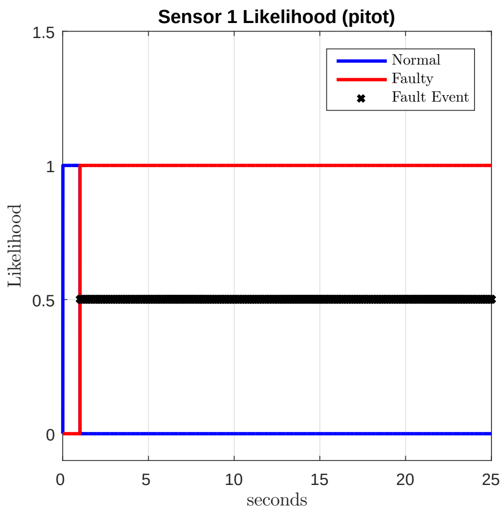
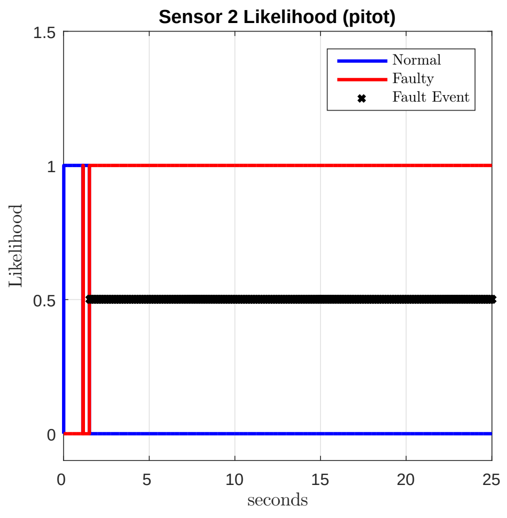


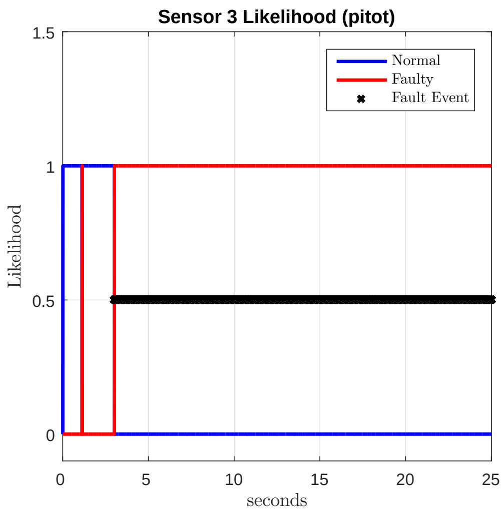
Figure 5.5: Altitude of aircraft using UKF-GPB-1 sensor fault detection during climbing flight scenario.



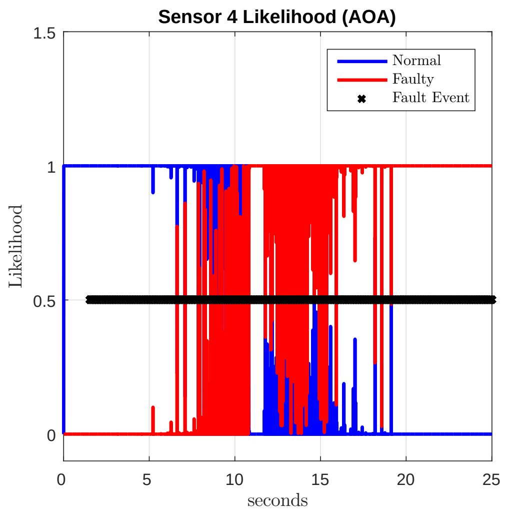
(a)



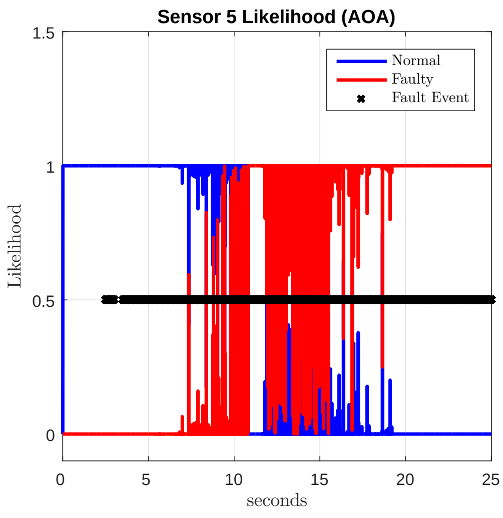
(b)



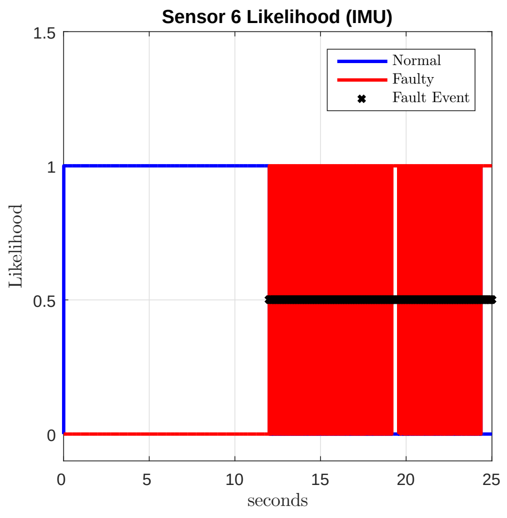
(c)



(d)



(e)



(f)

Figure 5.6: Fault likelihood for sensors during Climbing Flight

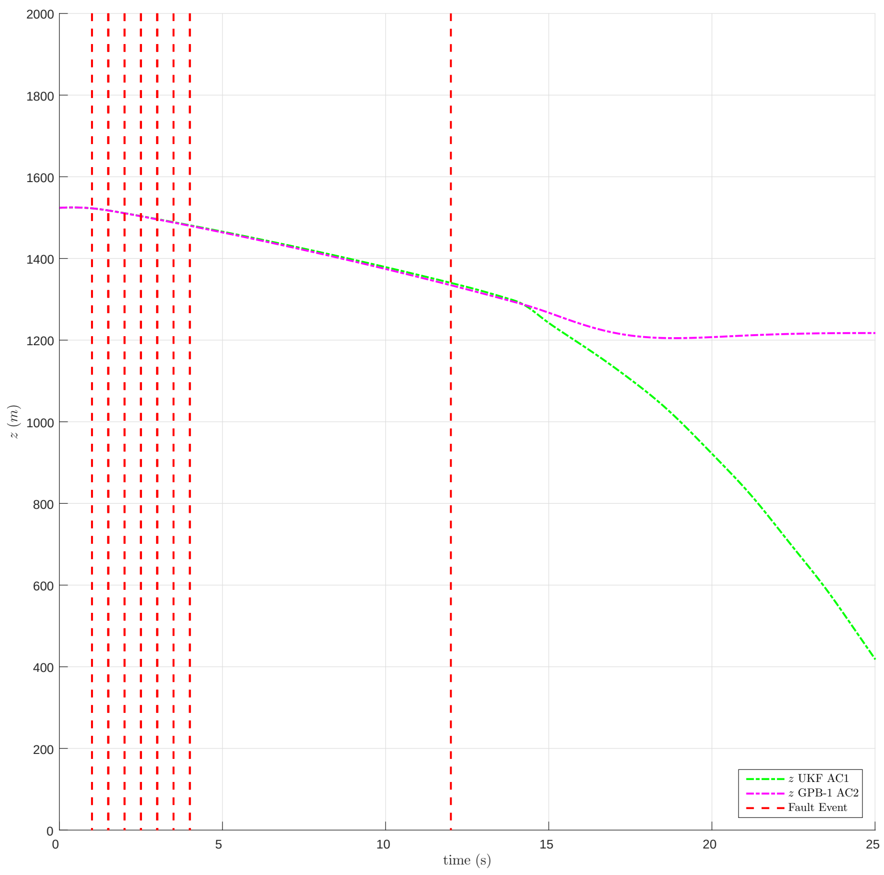


Figure 5.7: Altitude of aircraft using UKF-GPB-1 sensor fault detection during descending flight scenario.

servations coincide with the expected normal sensor observations these cause abrupt changes between normal and faulty detections.

5.3.3 Descending flight

The descending flight scenario shows both AC1 and AC2 descending to the target altitude as shown in Figure 5.7. AC1 continues to descend below the target altitude, whereas AC2 is able to descend and maintain the target altitude. The sensor fault detection shown in Figure 5.8 is similar to previous scenarios, where the pitot and IMU sensor fault detection is instantaneous and the AoA sensor takes time to detect the fault. This is to be expected, for the same reasons as the previous scenario.

5.3.4 Turning flight

The turning flight scenario shows AC1 is unable to maintain altitude during the right turn, whereas AC2 maintains level flight during the turn. Furthermore, Figure 5.9 shows AC1 making an abrupt change in heading which is not the same as the target

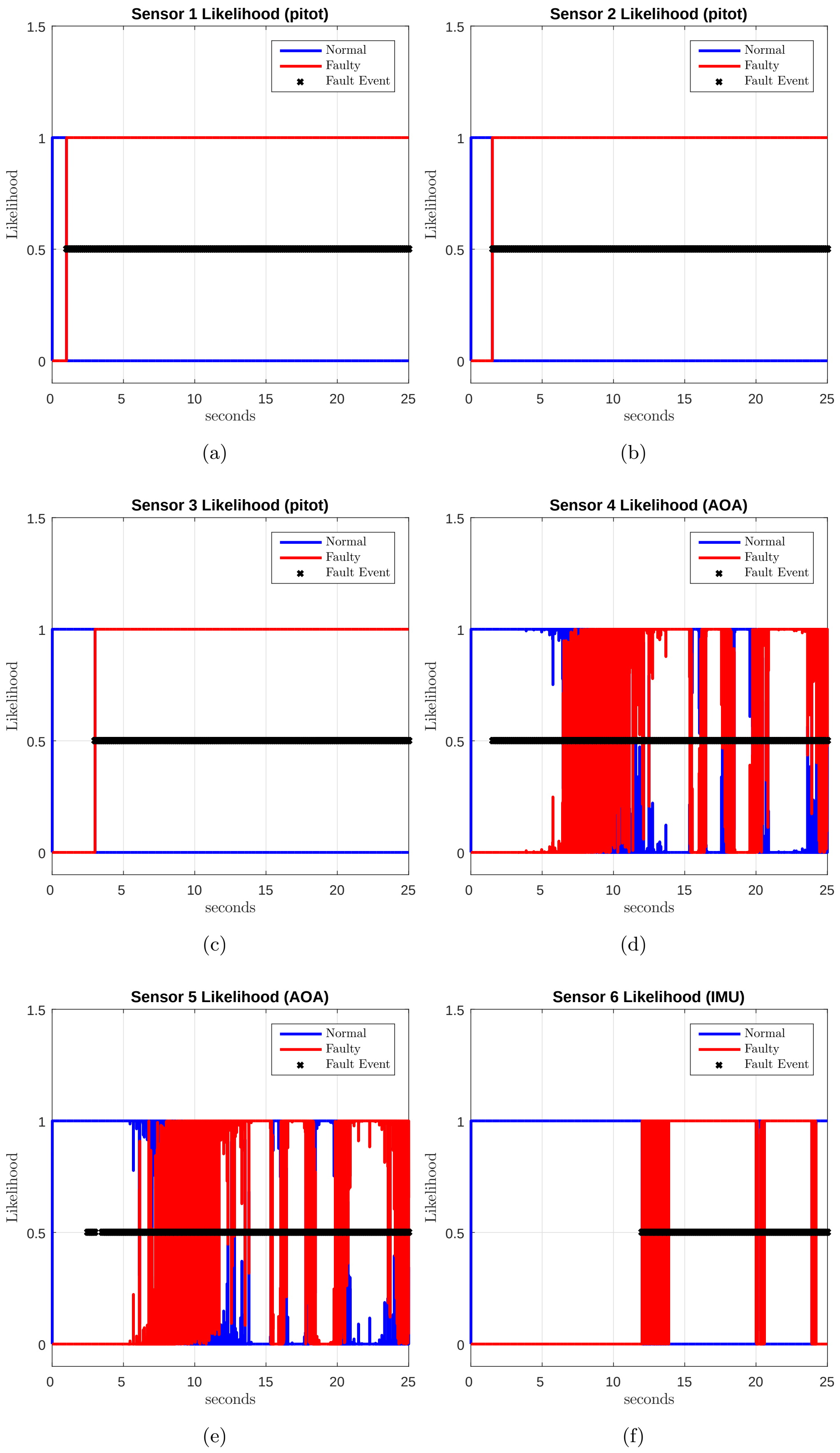
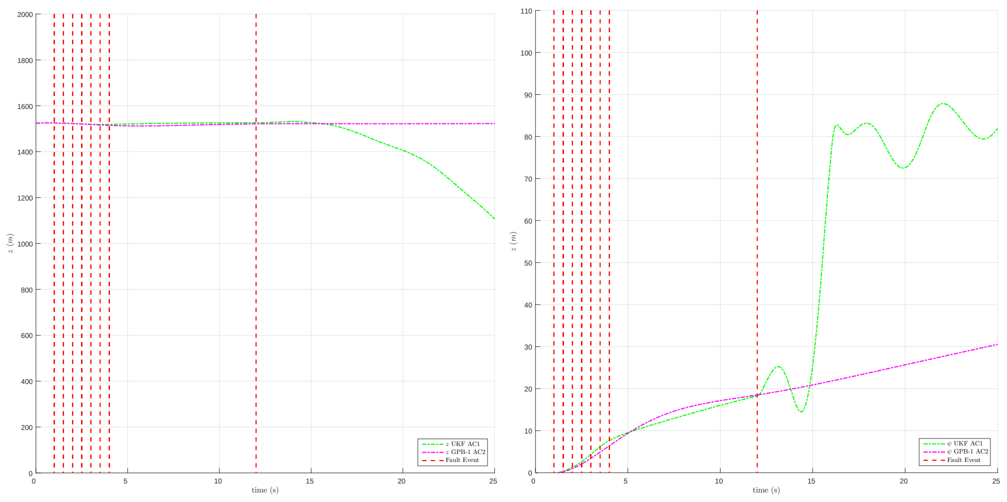


Figure 5.8: Fault likelihood for sensors during descending Flight



(a) (b)

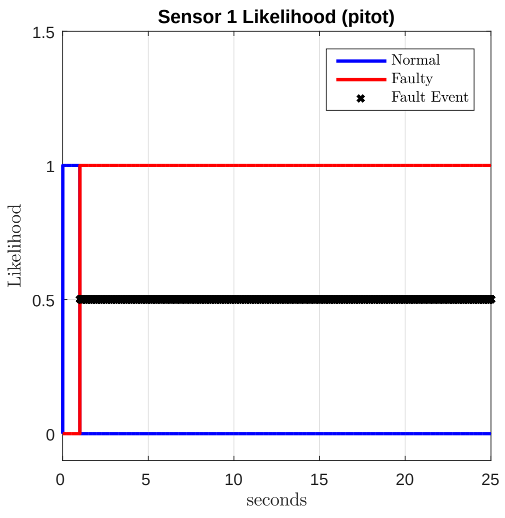
Figure 5.9: Altitude of aircraft using UKF-GPB-1 sensor fault detection during turning flight scenario.

heading. AC2's change in heading is constant and on course to the target heading.

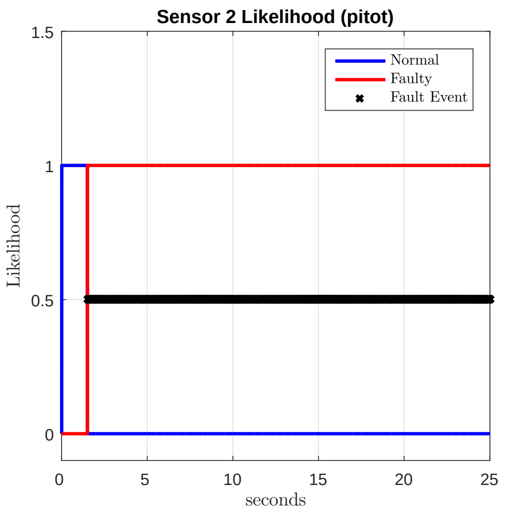
The sensor fault detection during this scenario is shown in Figure 5.10. The pitot sensor fault detection is instantaneous for the same reasons as explained in the level flight scenario. The AoA sensor fault takes time to be detected, as seen in the previous scenario. However, the AoA has fewer fluctuations than in the previous scenarios. This is because the AoA is changing during the turn due to the change in speed. The faulty observations obtained as time increases is closer to the faulty sensor distribution and further from the normal expected sensor distribution. As time increases the number of fluctuations reduce. The IMU sensor faults are detected instantaneously and has fewer fluctuations than the previous scenarios this is expected since the expected accelerations and rotations during a turn is non-zero. The faulty sensor observations are zero mean and are closer to the fault distribution and further from the expected distribution for normal sensor. This increases the fault likelihood ($\hat{L}^{(IMU*)} >> \hat{L}^{(IMU)}$) indicating a fault.

5.4 Multiple sensor fault detection using parallel UKF Generalized Pseudo Bayes-1

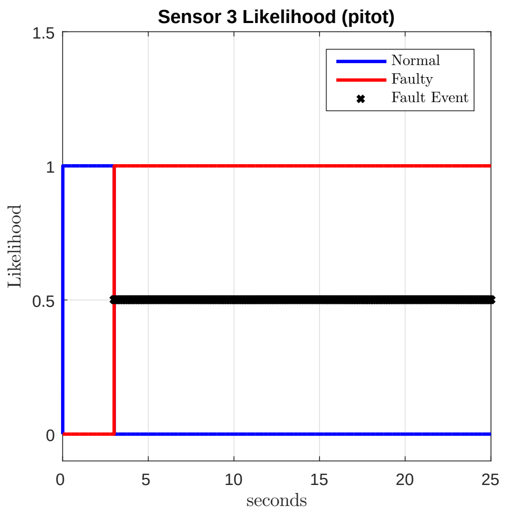
The previous section implemented a UKF-GPB-1 fault detection method by evaluating the hypotheses for each sensor sequentially. This section compares the accuracy of the state-estimate and time taken to update the state-estimate with sensor measurements when the hypotheses are evaluated in parallel as shown in Figure 5.11. Each hypothesis



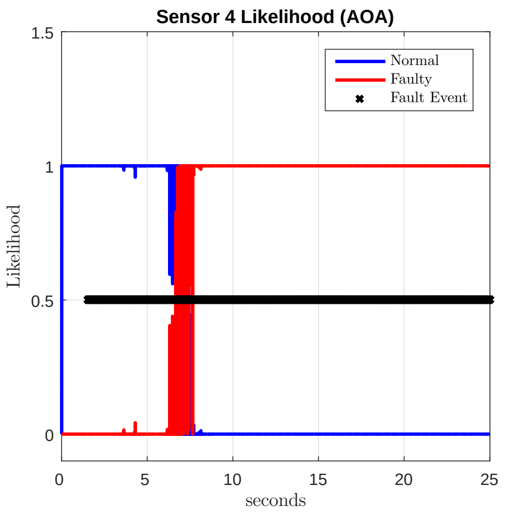
(a)



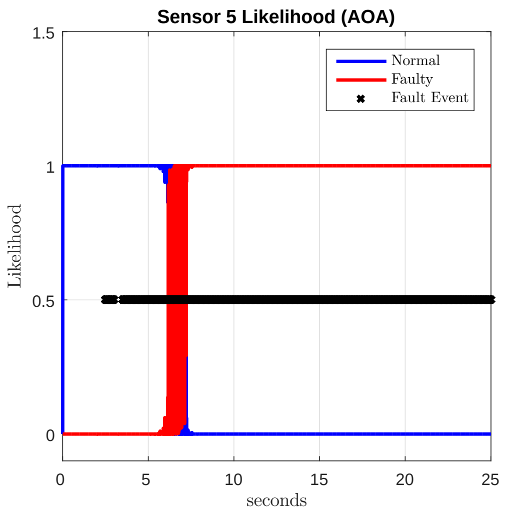
(b)



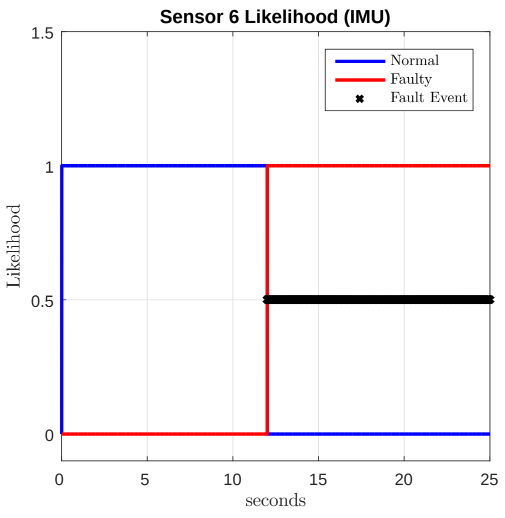
(c)



(d) Turning Flight



(e)



(f)

Figure 5.10: Fault likelihood for sensors during turning flight

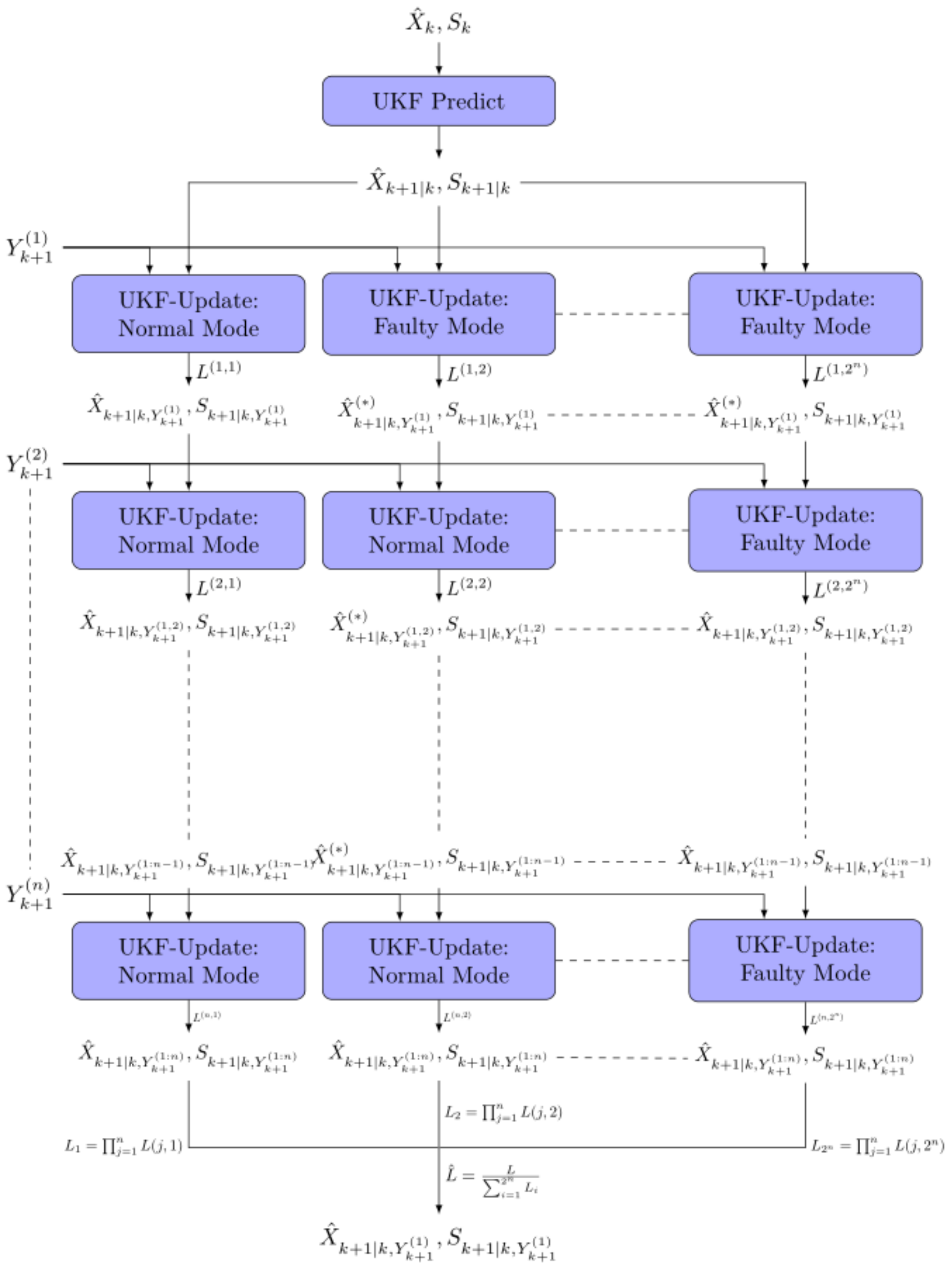


Figure 5.11: Flow diagram for UKF-GPB-1 multiple sensor fault detection using parallel hypotheses

Table 5.2: Parallel GPB-1 fault detection hypothesis for n sensors

	$H^{(1)}$	$H^{(2)}$	$\dots$	$H^{(2^n)}$
$S^{(1)}$	0	1	$\dots$	1
$S^{(2)}$	0	0	$\dots$	1
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
$S^{(n)}$	0	0	$\dots$	1

Table 5.3: Filter comparison

Filter-1 (Input to Autopilot)	**Filter-2**
Parallel	Parallel
Parallel	Sequential
Sequential	Sequential
Sequential	Parallel

consists of a combination of multiple sensors in normal or faulty mode. Given n sensors there are 2^n combinations of hypotheses to be evaluated. Table 5.2 shows the general form of the permutations for example hypothesis $H^{(2)}$ assumes sensor $S^{(1)}$ is the only sensor that is faulty.

The simulation is modified to compare the accuracy and time taken to update the state-estimate using both sequential and parallel implementations. Both implementations are executed within the same loop as shown in Figure 5.12. The outputs of Filter-1 and Filter-2 are $\hat{X}^{(1)}_{k+1}$ and $\hat{X}^{(2)}_{k+1}$ respectively. The output of Filter-1 is the state-estimate used for the autopilot. The time taken to update the UKF state-estimate using the parallel and sequential implementation at each time-step throughout the duration of the simulation is shown in Figure 5.13. The sequential implementation is approximately 37 times faster than the parallel implementation. This is as expected since the sequential implementation executes $2n$ hypotheses where as the parallel implementation executes 2^n hypotheses; a significantly larger number of hypotheses than the sequential implementation.

The accuracy of a state-estimator is determined by evaluating Root-Mean-Square-Error (RMSE) metric between the true state of the aircraft and the state-estimate

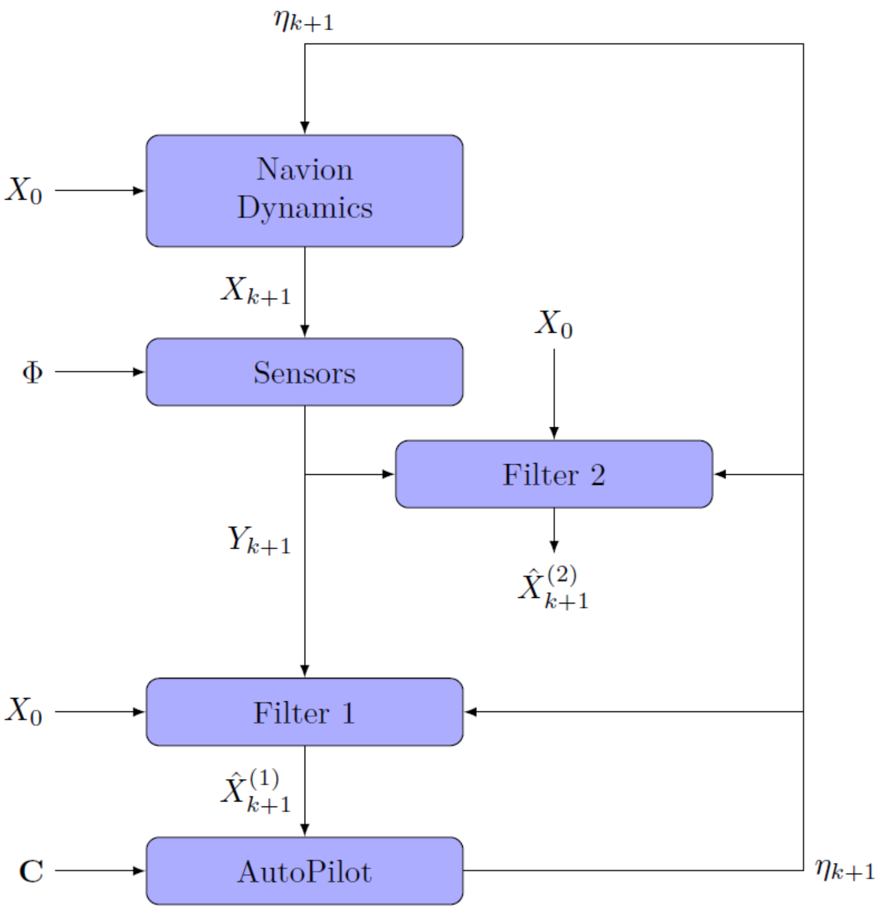


Figure 5.12: Simulation flow diagram for sequential and parallel GPB-1

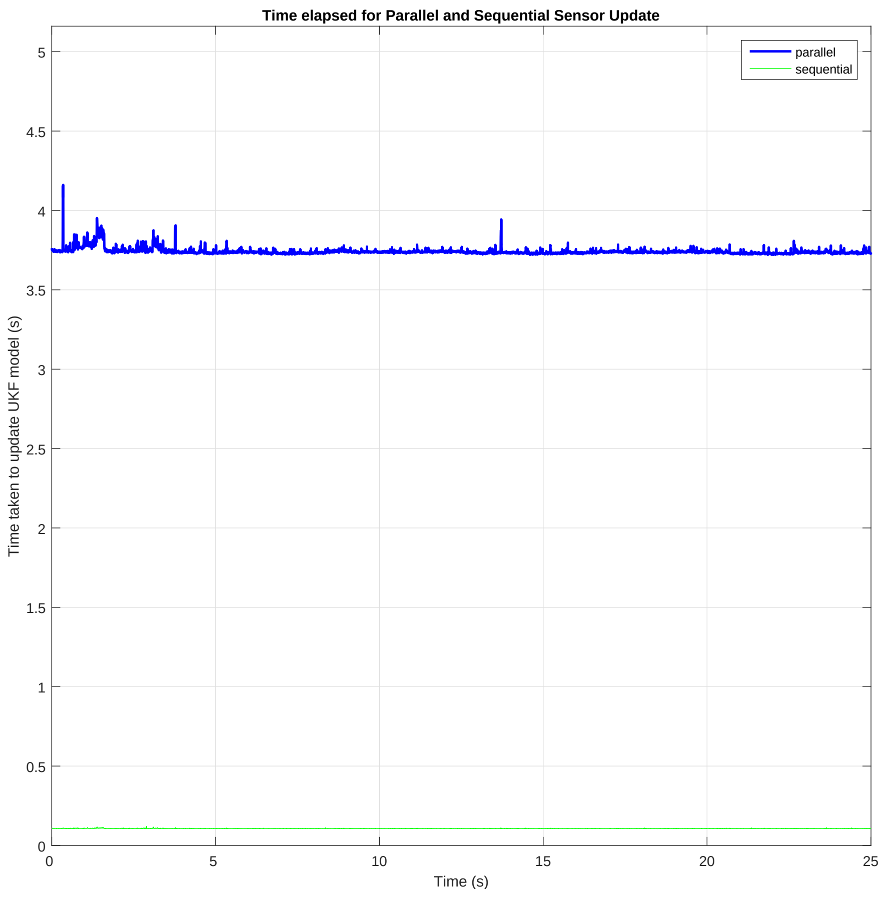


Figure 5.13: Time taken for measurement update of Parallel vs Sequential implementation of UKF-GPB-1 Fault detector

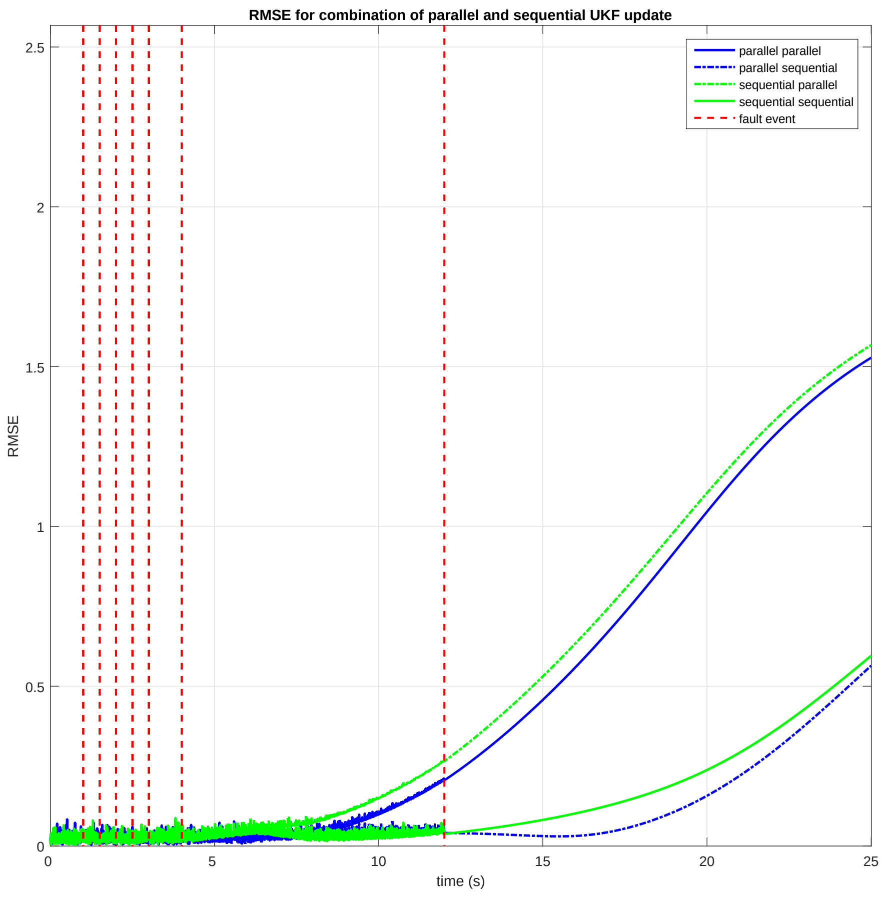


Figure 5.14: RMSE of parallel and sequential implementation of GPB-1 sensor fault detection

obtained from the estimator. In the above scenarios, the true state of the aircraft is influenced by the estimator since the state-estimate is input to the autopilot and the autopilot controls the actions of the aircraft that affects the true state of the aircraft. A comparison of accuracy for both methods requires a common true state. This is done by executing the combination shown in Table 5.3. Row 1 is compared with row 2 and row 3 is compared with row 4 since they share the aircraft's true state information.

Figure 5.14 shows the accuracy of using the parallel UKF-GPB-1 and sequential UKF-GPB-1 by determining the RMSE between the true state of the aircraft and the state-estimates generated by the parallel UKF-GPB-1 and the sequential UKF-GPB-1.

$$RMSE = \sqrt{\frac{1}{k}\left(\sum \mathbf{X} - \hat{\mathbf{X}}\right)^2} \tag{5.16}$$

Prior to conducting this research, the parallel UKF-GPB-1 was expected to outperform the sequential UKF-GPB-1 in terms of accuracy. It was anticipated that one of the combinations of models used in the parallel UKF-GPB-1 would match the true state and there would be less corruption of the estimates than the sequential UKF-GPB-1.

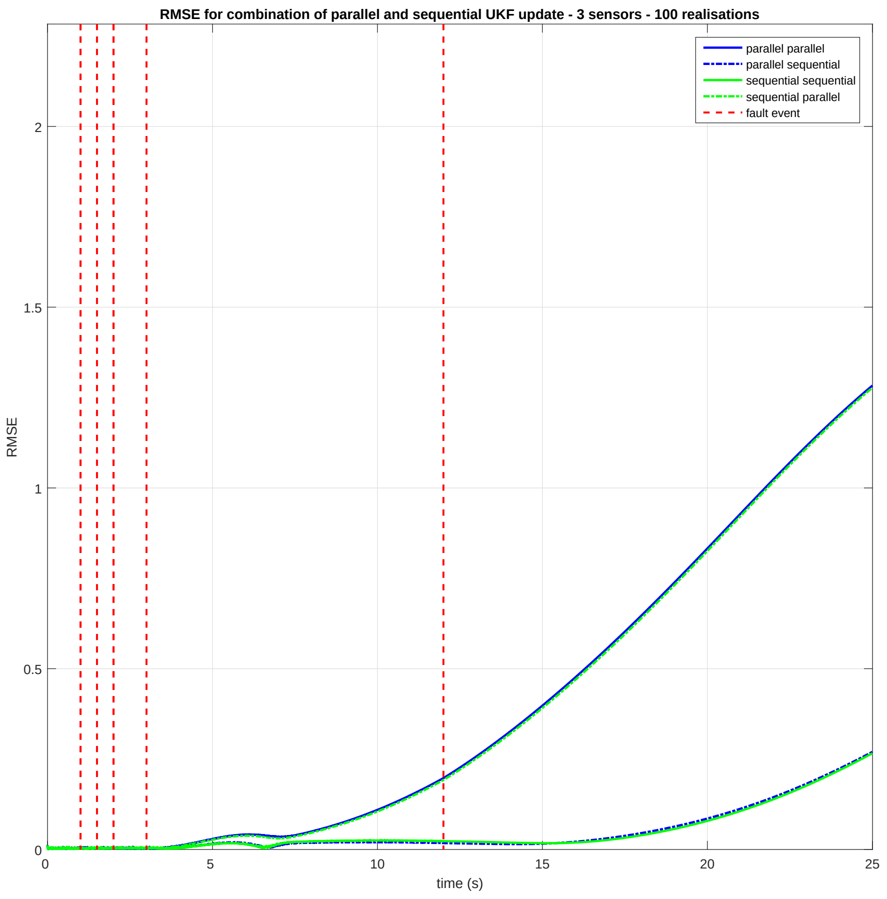


Figure 5.15: RMSE of Parallel and Sequential UKF-GPB-1 sensor fault detection for 100 realisations

However, Figure 5.14 shows that this is not the case and the sequential implementation is more accurate (as well as faster – see Figure 5.13). To understand why this comes about, it is helpful to note that the error-statistics are deliberately mis-modelled. The errors associated with a fault (as described in section 5.1) are, in reality and in the simulation, correlated over time but the correlation is relatively small. The fault mode in the UKF-GPB-1 filters assumes that the errors associated with a fault are uncorrelated and independent over time but relatively large. Theoretically, a model could be used to represent every type of faults for estimating the biases present within every sensor, but this is not practical to implement since it would require defining every single fault that could occur to within a sensor which is not feasible. This intentional mis-modelling has a detrimental effect on the parallel UKF-GPB-1's accuracy relative to that of the sequential UKF-GPB-1. This comes about because the sequential UKF-GPB-1 is better able to interpolate between the faulty and normal models. In contrast, the parallel UKF-GPB-1 makes a stronger implicit assumption that one of the combinations of sensors is faulty and that those faulty sensors are generating errors that are independent (over time) and large. This is not the case, since at some instances during the simulation some of the sensors in the combination is faulty not all of them.

5.5 Summary

This chapter has developed and demonstrated a Multiple Sensor Fault detector using Unscented Kalman Filter Generalized Pseudo Bayes-1 (GPB-1) that evaluates each sensors hypotheses sequentially. The filter successfully detects multiple sensor faults during a series of common flight scenarios. The sequential implementation of GPB-1 updates the state-estimate faster and maintains a more accurate state-estimate than the parallel implementation of GPB-1.

Chapter 6

Rare Event simulation

Collision Avoidance is an essential safety feature required for Unmanned Aircraft Systems to operate alongside manned aircraft within non-segregated airspace. The Collision Avoidance capability consists of two functions known as Conflict Detection and Resolution (CD&R). The Cooperative CD&R scenario assumes that the current state and future intent is shared between traffic and any conflict encountered is resolved cooperatively. In contrast, non-cooperative scenarios assume that no information is shared between traffic. Instead, such information is deduced or inferred from measurements of the non-cooperative traffic obtained from the sensors on-board the UAS. However, the measurements obtained from sensors are inherently noisy, which give rise to uncertainties in the observed state and predicted motion of non-cooperative traffic. In an environment where future trajectories are uncertain, the probability of conflict (P_c) is an essential metric since it serves as an indicator for Conflict Detection in a non-cooperative environment. The accuracy of the metric is important since the performance of the Resolution stage of the Collision Avoidance system is dependent on the accuracy of the Conflict Detection stage.

Fortunately, conflicts between air traffic are typically low. However, despite the low probability, the catastrophic outcome of a collision makes it essential to estimate the probability of conflict accurately. Estimating low probabilities of conflict accurately using Direct Monte Carlo method requires a large number of samples which leads to large computational load. A rare event simulation method called Subset Simulation is explored to be applied to the problem of estimating low probabilities.

Subset Simulation is a Rare Event simulation method that was proposed by Au and Beck in 2001 [118]. The method calculates the probability of rare events occurring as the product of the probabilities of less-rare events. Such an approach is less computationally expensive than 'brute force' Direct Monte Carlo. The Subset Simulation method has mostly been used for applications in Civil Engineering discipline. It was initially used to estimate small probabilities of failure P_F in structures due to seismic risk [119].

Specifically, Reliability sensitivity analysis is a requirement within Civil Engineering that relies on Monte Carlo methods to address the problem of estimating probabilities where uncertainties exist. Subset Simulation has been applied to the problem of estimating low failure probability as product of larger probabilities [120, 121]. Composite system reliability has been estimated using Subset Simulation in reference [122]. Bourinet et al. use the Subset Simulation method from the perspective of Support Vector Machine Classification to estimate the probability associated with structural reliability [123].

A general outline of the Subset Simulation method is presented in this chapter and the interested reader is referred to [124] for more details. The Subset Simulation method uses a combination of the Direct Monte Carlo and the Metropolis Hastings methods, a description of these methods are presented in sections 6.1 and 6.2 respectively. Section 6.3 describes the Subset Simulation method. The algorithms mentioned in this chapter are presented in Appendix A.

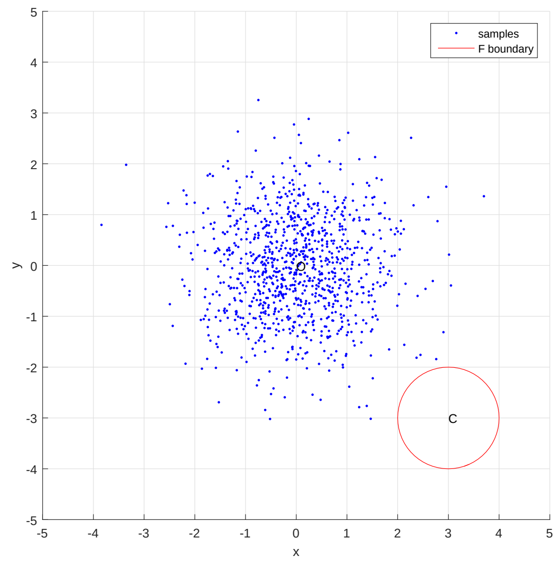
6.1 Direct Monte Carlo

The Direct Monte Carlo (DMC) method is a sampling method that can be used to characterise a distribution of interest [125]. The objective of this section is to estimate the probability of a type of event to occur. Therefore, the DMC method is used as a 'statistical averaging' tool, where the probability of failure P_F is estimated as the ratio of failure responses to the total number of trials [124].

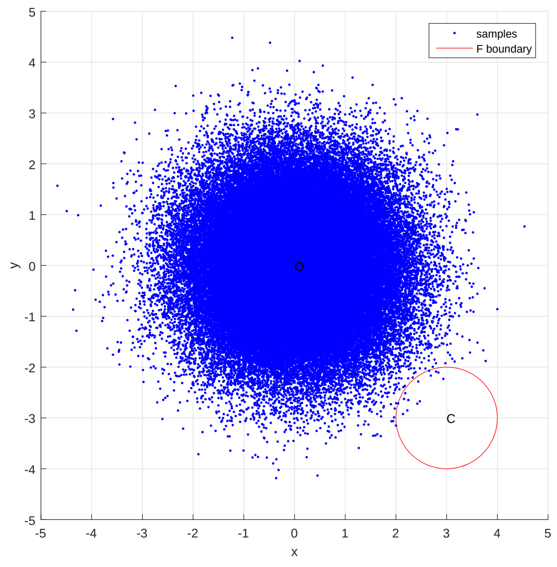
A set of N independent identically distributed (i.i.d) inputs $\{X_n : n = 1, ..., N\}$ are drawn from the proposal distribution $q(X|\mu, \sigma^2)$ of the input parameter space. The proposal distribution can be any known distribution that can be used to generate samples. A Normal distribution that is centred at the mean μ and has a variance of σ^2. A set of system responses are observed $\{Y_n = h_r(X_n) : n = 1, ..., N\}$, where $h_r(...)$ is the system process. The occurrence of a failure event F is indicated when a scalar quantity b_F (threshold) is exceeded. The number of samples that exceed the threshold is Y_F. Therefore the probability of failure is estimated as $P_F = P(Y \geq b_F) = \frac{Y_F}{N}$. Such an approach is suitable for large probabilities (such as $P > 0.1$) where a small number of samples can be used to estimate the probability. However for small probabilities (such as the tail region of the pdf, where $P \leq 10^{-3}$) a large number of samples must be drawn to estimate the probability accurately. This is illustrated by the following example.

6.1.1 Estimating probability of drawing samples from region F

Figure 6.1 shows a 10×10 square centred at $O = [0, 0]^T$. The region F is a circle with radius $r_c = 1$, centred at $C = [3, -3]^T$ within this square. The objective is to estimate



(a) Direct Monte Carlo with 100 samples



(b) Direct Monte Carlo with 10^5 samples

Figure 6.1: The probability of drawing samples from the region F is estimated using Direct Monte Carlo. Figure 6.1(a) estimates the $P_F = 0$ with 100 samples. Figure 6.1(b) estimates the $P_F = 1.5 \times 10^{-4}$ with 10^5 samples.

the probability of drawing samples from this region. The probability distribution of the overall area is represented by a Gaussian distribution centred at $O = [0, 0]^T$. A set of N samples $\{X_n : n = 1, ..., N\}$ are drawn where each sample is a vector; $X_n = [x_n, y_n]^T$. The x and y values of each sample are the x-coordinate and y-coordinates of the position respectively. To clarify, $X_1 = [x_1, x_2]^T$ where $x_1 \sim \mathcal{N}(0, 1)$ and $y_1 \sim \mathcal{N}(0, 1)$. The distance between the position of each sample and centre of circle C is $\{R_n = H(X_n, C) : n = 1, ...N\}$ as defined by Algorithm 5. To clarify, the distance between sample X_1 and C is $R_1 = H(X_1, C)$. Algorithm 6 is used to estimate the probability of drawing samples from the region F.

Figure 6.1(a) shows 100 samples drawn from the distribution. Note no samples are drawn from the area F. The probability is estimated $P_F = 0$. The number of samples are increased to $N = 10^5$. Figure 6.1(b) shows some samples are drawn from the region F and the probability is estimated $P_F = 1.5 \times 10^{-4}$ This illustrates that Direct Monte Carlo requires a significantly large number of samples to estimate the probability of drawing samples from the region F.

This method estimates P_F by attempting to realise the entire pdf centred at O that includes the area F. As the area F reduces the number of samples required to estimate P_F increases making such an approach computationally demanding. A different algorithm is needed.

6.2 Metropolis Hastings

Metropolis-Hastings (MH) is a Markov Chain Monte Carlo (MCMC) method used to characterise a distribution of interest by sampling from a known distribution. This distribution of interest is known as the target distribution. The MH algorithm originates from the Metropolis algorithm first used in statistical physics by Metropolis and co-workers (Metropolis et al, 1953) [126]. Hastings proposed a generalised form of this algorithm leading to the Metropolis Hastings (MH) algorithm [127].

The MH method generates samples from the proposal distribution $q(X|x_0, \sigma^2)$ by starting from a seed value x_0. A chain of n samples is then generated, starting with x_0. The sample x_{k+1} is generated from the current sample x_k using the following steps [124]:

1. Generate a candidate sample $x^* \sim\ q(x^*|x_k, \sigma^2)$.
2. Calculate an acceptance ratio: $\alpha = \frac{q(x_k|x^*,\sigma^2)f(x^*)}{q(x^*|x_k,\sigma^2)f(x_k)}$
3. Draw a sample e from a uniform distribution [0,1]
4. Set $x_{k+1} = \begin{cases} x^* & \text{if } e < \alpha \\ x_k & \text{otherwise} \end{cases}$
5. Repeat steps 1 to 4 until n samples have been generated.

The function $f(...)$ defines the target density for the input sample. While, $n \rightarrow \infty$, this process is guaranteed to accept samples from q that leads to the realisation of the target distribution [128]. To help ensure that all regions of the target density are explored, multiple seeds can be used to generate multiple chains of samples in parallel [124].

6.2.1 Drawing samples from the region F

The Metropolis Hastings method is defined in algorithm 7 and it is applied to the example of estimating the probability of drawing samples from region F as shown in the previous section. The covariance of the proposal σ^2 is a 2×2 identity matrix $I_{2\times2}$ and the covariance of the distribution of interest $\sigma_{r_c}^2 = r_c^2 \times I_{2\times2}$ where r_c is the radius of the region F. For this example $r_c = 1$, therefore $\sigma_{r_c}^2 = I_{2\times2}$.

Figure 6.2 illustrates the chains of samples generated by the Metropolis Hastings algorithm. This Figure shows 10 samples drawn from the proposal distribution using the DMC method. These samples are seeds $s = \{X_1, ..., X_{10}\}$. The MH algorithm is applied using the seeds s. Each seed generates a chain of 10 samples. Note that many sample chains do not reach the region F. It is clear that it might be more efficient to generate more samples for chains with seeds that are closer to the region F since they have higher likelihood of generating samples that are within the region F or closer to the region F. Subset Simulation achieves this and is described in the next section.

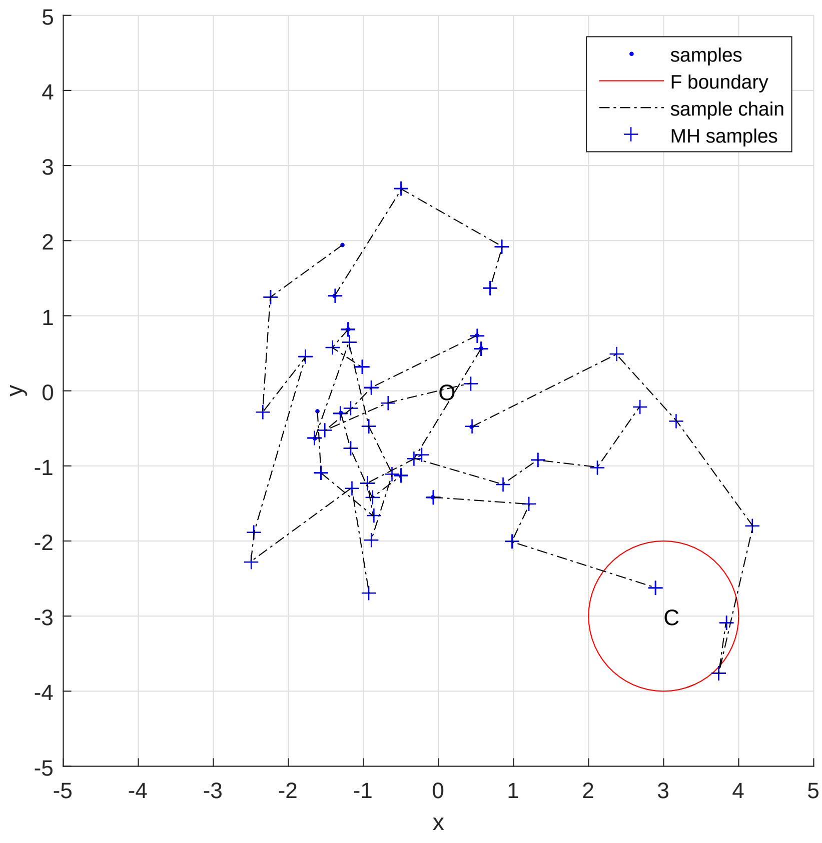


Figure 6.2: Drawing samples from the region F using Metropolis Hastings algorithm to generate chains of conditional samples. The initial samples used as seeds are drawn using Direct Monte Carlo.

6.3 Subset Simulation

Subset Simulation generates a Complimentary Cumulative Distribution Function (CCDF) of the response quantity of interest Y. The probability of failure P_F can be directly estimated from the CCDF. This CCDF is constructed by generating samples that satisfy a series of intermediate thresholds $b_1 > b_2 > b_3 > ... > b_{m-1}$ that divide the space into m nested regions. These thresholds are defined adaptively as the simulation progresses. This is described later on in this section. The threshold b_{m-1} is the required failure threshold b_F ($b_{m-1} = b_F$). The intermediate thresholds allow the probability of failure to be estimated using a classical conditional structure given by

$$P_F = P(Y < b_{m-1}|Y < b_{m-2})P(Y < b_{m-2}) \tag{6.1}$$

Samples are generated to satisfy the threshold for each level. The total number of levels m is dependent on the magnitude of the target probability P_F. Subset Simulation uses 'level probability' $p_0 \in (0, 1)$ to control how quickly the simulation reaches the target event of interest [124]. The target probability is used to approximate the number

	Level 0			Level i				Level $m-1$	
$\mathbf{P}_n^{(0)}$	$\mathbf{B}_n^{(0)}$	$\tilde{\mathbf{X}}_n^{(0)}$	$\mathbf{P}_n^{(i)}$	$\mathbf{B}_n^{(i)}$	$\tilde{\mathbf{X}}_n^{(i)}$	.....	$\mathbf{P}_n^{(m-1)}$	$\mathbf{B}_n^{(m-1)}$	$\tilde{\mathbf{X}}_n^{(m-1)}$
$P_1^{(0)}$	$B_1^{(0)}$	$\tilde{X}_1^{(0)}$							
$\vdots$	$\vdots$	$\vdots$							
$P_{N-N_c}^{(0)}$	$B_{N-N_c}^{(0)}$	$\tilde{X}_{N-N_c}^{(0)}$							
$P_{N-N_c+1}^{(0)}$	$B_{N-N_c+1}^{(0)}$	$\tilde{X}_{N-N_c+1}^{(0)}$	$P_1^{(i)}$	$B_1^{(i)}$	$\tilde{X}_1^{(i)}$				
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$				
$P_N^{(0)}$	$B_N^{(0)}$	$\tilde{X}_N^{(0)}$	$P_{N-N_c}^{(i)}$	$B_{N-N_c}^{(i)}$	$\tilde{X}_{N-N_c}^{(i)}$				
			$P_{N-N_c+1}^{(i)}$	$B_{N-N_c+1}^{(i)}$	$\tilde{X}_{N-N_c+1}^{(i)}$	.....	$P_1^{(m-1)}$	$B_1^{(m-1)}$	$\tilde{X}_1^{(m-1)}$
			$\vdots$	$\vdots$	$\vdots$	.....	$\vdots$	$\vdots$	$\vdots$
			$P_N^{(i)}$	$B_N^{(i)}$	$\tilde{X}_N^{(i)}$	.....	$P_{N-N_c}^{(m-1)}$	$B_{N-N_c}^{(m-1)}$	$\tilde{X}_{N-N_c}^{(m-1)}$
						.....	$P_{N-N_c+1}^{(m-1)}$	$B_{N-N_c+1}^{(m-1)}$	$\tilde{X}_{N-N_c+1}^{(m-1)}$
							$\vdots$	$\vdots$	$\vdots$
							$P_N^{(m-1)}$	$B_N^{(m-1)}$	$\tilde{X}_N^{(m-1)}$

Table 6.1: A table of samples, response values and probability intervals generated at various levels during Subset Simulation.

of levels m required by evaluating $P_F = (p_0)^m$. To clarify, if the target probability is $P_F = 10^{-5}$ and $p_0 = 0.1$ then the total number of levels required will be $m = 5$.

6.3.1 Level 0

Subset Simulation begins at level $i = 0$ with Direct Monte Carlo (DMC) sampling from the entire region of interest. A set of N samples $\{X_n^{(0)} : n = 1, ..., N\}$ are drawn from a proposal distribution $q(X_n^{(0)}|\mu, \sigma^2)$ (as described in section 6.1). The set of output responses $Y_n^{(0)}$ are evaluated $\{Y_n^{(0)} = h_r(X_n^{(0)}) : n = 1, ..., N\}$. The function $h_r(...)$ defines the system response to the input sample. In the context of SS, the responses $Y_n^{(0)}$ are also known as the quantity of interest. The set $Y_n^{(0)}$ is sorted in descending order to create the set $\{B_n^{(0)} : n = 1, ..., N\}$. The input samples $X_n^{(0)}$ are reordered $\tilde{X}_n^{(0)}$ and correspond to the sorted quantity of interest $B_n^{(0)}$. To clarify, $\tilde{X}_1^{(0)}$ is the input sample that generates the largest output $B_1^{(0)}$. A CCDF is generated by plotting $B_n^{(0)}$ against the probability intervals $P_n^{(0)}$. The probability intervals $P_n^{(0)}$ are generated using the following equation:

$$P_n^{(i)} = p_0^i \frac{N-n}{N} \quad n = 1......N \tag{6.2}$$

The vector of probability intervals $P_n^{(0)}$ is concatenated with the sorted quantity of interest $B_n^{(0)}$ and their respective samples $\tilde{X}_n^{(0)}$ as illustrated in Table 6.1 by the column titled ‘Level 0’.

The set of probability intervals $P_n^{(0)}$ are plotted against $B_n^{(0)}$ to generate the CCDF. Level 0 makes it possible to accurately approximate CCDF values from $1 - N^{-1}$ to

$\mathbf{P}_n$	$\mathbf{B}_n$	$\tilde{\mathbf{X}}_n$	
$P_1^{(0)}$	$B_1^{(0)}$	$\tilde{X}_1^{(0)}$	
$\vdots$	$\vdots$	$\vdots$	*Level 0*
$P_{N-N_c}^{(0)}$	$B_{N-N_c}^{(0)}$	$\tilde{X}_{N-N_c}^{(0)}$	*samples retained*
$P_1^{(i)}$	$B_1^{(i)}$	$\tilde{X}_1^{(i)}$	
$\vdots$	$\vdots$	$\vdots$	*Level i*
$P_{N-N_c}^{(i)}$	$B_{N-N_c}^{(i)}$	$\tilde{X}_{N-N_c}^{(i)}$	*samples retained*
$\vdots$	$\vdots$	$\vdots$	
$P_1^{(m-1)}$	$B_1^{(m-1)}$	$\tilde{X}_1^{(m-1)}$	
$\vdots$	$\vdots$	$\vdots$	
$P_{N-N_c}^{(m-1)}$	$B_{N-N_c}^{(m-1)}$	$\tilde{X}_{N-N_c}^{(m-1)}$	
$P_{N-N_c+1}^{(m-1)}$	$B_{N-N_c+1}^{(m-1)}$	$\tilde{X}_{N-N_c+1}^{(m-1)}$	
$\vdots$	$\vdots$	$\vdots$	*Level* $m-1$
$P_N^{(m-1)}$	$B_N^{(m-1)}$	$\tilde{X}_N^{(m-1)}$	*samples retained*

Table 6.2: The probabilities and response values of samples generated at various levels of subset simulation are concatenated. Note that the seeds used generate samples at each level are discarded and replaced with there respective conditional samples.

p_0. Typically the region of interest within the pdf is outside this range (since SS is typically used to realise rare events). To explore probabilities below p_0, further levels of simulation must be conducted.

6.3.2 Level $i > 0$

The subsequent levels of SS where, $i > 0$ explore the rarer regions of the probability distribution. This is achieved by generating multiple chains of conditional samples using the MH method as discussed in the previous section. The number of chains and number of samples per chain are N_c and N_s respectively. They are determined as

$$N_c = p_0 N \tag{6.3}$$

$$N_s = p_0^{-1} \tag{6.4}$$

Each level of subset simulation maintains N samples ($N = N_c N_s$). The response values of conditional samples generated for the current level i must not exceed the intermediate threshold b_i for this level. This threshold is determined by

$$b_i = B^{(i-1)}_{N-N_c} \quad i \text{ is the current subset level} \tag{6.5}$$

The intermediate threshold for level $i = 1$ is $b_1 = B^{(0)}_{N-N_c}$. To clarify the intermediate threshold is the $(N - N_c)^{\text{th}}$ element of the sorted set of response values $B^{(0)}_n$. The set of seeds $s^{(i)}_j$ are used to generate samples for the current level i are samples generated from the previous level $(i - 1)$ are defined by

$$s^{(i)}_j = \tilde{X}^{(i-1)}_n \tag{6.6}$$

where $1 \leq j \leq N_c$, $(N - N_c + 1) \leq n \leq N$ and $i > 0$.

The set of seeds used to generate conditional samples for level $i = 1$ is $s^{(1)} = \{\tilde{X}^{(0)}_{N-N_c+1}, ..., \tilde{X}^{(0)}_N\}$. The N conditional samples $X^{(1)}_n$ are generated using the MH method. The quantities of interest for $X^{(1)}_n$ are determined $\{Y^{(1)}_n = h_r(X^{(1)}_n) : n = 1, ..., N\}$ and are sorted in the same manner as the previous level $B^{(1)}_n$. The set $B^{(1)}_n$ and respective samples $\tilde{X}^{(1)}_n$ are concatenated with the probability intervals $P^{(1)}_n$ as illustrated in Table 6.1 by the column titled 'Level i'. Note the samples $\{\tilde{X}^{(0)}_{N-N_c+1}, ..., \tilde{X}^{(0)}_N\}$ shown in the column titled 'Level 0' are used as seeds to generate the conditional samples $\{\tilde{X}^{(i)}_1, ..., \tilde{X}^{(i)}_N\}$ in column titled 'Level i'.

This process is continued until the target level of probability $(p_0)^m$ is reached at level $i = m - 1$; as shown by the column titled 'Level $m - 1$'. The samples used as seeds to generate samples for the consecutive level are discarded and replaced with the generated samples. This is illustrated in Table 6.2. The column of probability intervals $\mathbf{P}_n$ are plotted against the respective quantities of interest $\mathbf{B}_n$ to generate a CCDF.

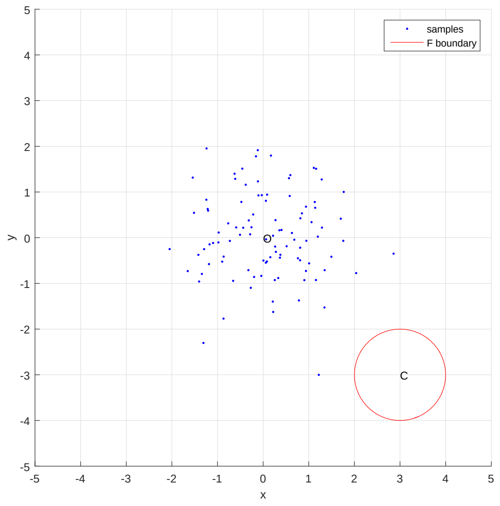
This method is continued until the target level of probability $P_F = (p_0)^m$ is reached. By generating and evaluating conditional samples, the output samples tend towards the target distribution with significantly less trials than are needed when using the DMC method. The progressive nature of the algorithm can be demonstrated in the example problem of estimating the probability of drawing samples from the region F.

6.3.3 Estimating Probability of drawing samples from region F

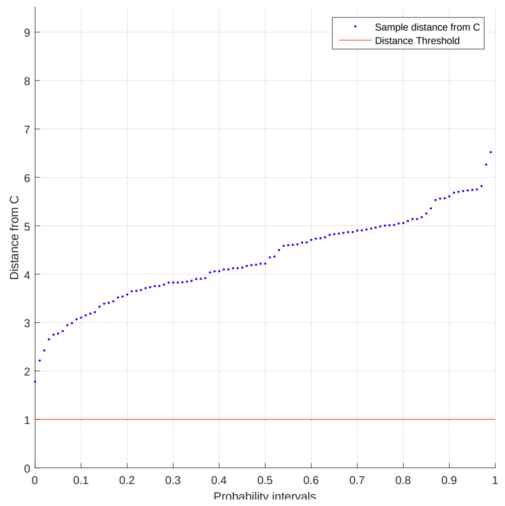
The example of estimating the probability of drawing samples from the region F shown in the previous sections is used to illustrate the Subset Simulation method (using algorithm 9). The radius of the circle bounding the region F is $r_c = 1$. The SS parameters used for this example are: $p_0 = 0.1$, $N = 100$, $N_s = 10$, $N_c = 10$, $m = 2$. Subset Simulation is typically used to realise rare events (for $P_F \leq 10^{-3}$ therefore $m > 3$). However for the purpose of this example the number of levels is kept low ($m = 2$).

The simulation begins with level 0 Direct Monte Carlo where a set of $N = 100$ samples $\{X_n^{(0)} : n = 1, ..., 100\}$ are drawn from a Gaussian distribution centred at $O = [0, 0]^T$ as shown in Figure 6.3(a). The quantity of interest $\{R_n^{(0)} = H(X_n^{(0)}, C) : n = 1, ..., 100\}$ is the distance between each sample $X_n^{(0)}$ and the centre of the circle $C = [3, -3]^T$ (this is the equivalent of $Y_n^{(0)}$ used previously). This is determined by process $H(...)$ as defined by algorithm 5. If the condition $R_n^{(0)} \leq r_c^{(0)}$ is satisfied then the nth sample $X_n^{(0)}$ is within the region F. This condition is used to determine if a sample is within the region F. The quantity of interest $R_n^{(0)}$ is sorted in descending order $\{B_n^{(0)} : n = 1, ..., 100\}$. This is because the samples with the lowest distances will be closest to the region F and have a higher likelihood of generating conditional samples closer to or within the region F than other samples as the simulation progresses to higher levels ($i > 0$). The input samples $X_n^{(0)}$ are reordered $\tilde{X}_n^{(0)}$ and correspond to the sorted quantity of interest $B_n^{(0)}$; to clarify, the distance between the sample $\tilde{X}_1^{(0)}$ and C is $B_1^{(0)}$. The probability intervals $P_n^{(0)}$ are determined by equation (6.2). The sorted quantity of interest $B_n^{(0)}$ and respective samples $\tilde{X}_n^{(0)}$ are concatenated with the probability intervals $P_n^{(0)}$ as shown in the column titled 'Level 0' in Table 6.3a. The CCDF shown in Figure 6.3(b) is generated by plotting the probability intervals $P_n^{(0)}$ against $B_n^{(0)}$. This CCDF shows that no samples have a distance less than the radius r_c therefore no samples have been drawn from the region F.

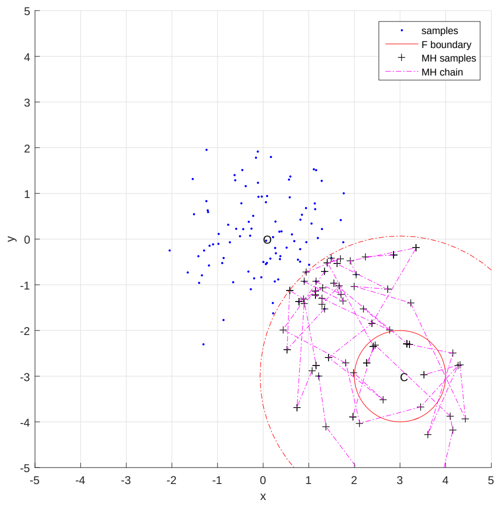
The SS method continues to the next level ($i = 1$) and generates N conditional samples using the MH method. The conditional samples $\{X_n^{(1)} : n = 1, ..., 100\}$ are generated from a set of seeds $s_j^{(1)} = \{\tilde{X}_{91}^{(0)}, ..., \tilde{X}_{100}^{(0)}\}$ that correspond to the sorted distances $\{B_n^{(0)} : n = 91, ..., 100\}$ from the previous level 0. The intermediate threshold $b_1 = B_{90}^{(0)}$ determined by equation (6.5) is used to ensure the conditional samples $X_n^{(1)}$ generated by each seed satisfies the condition $R_n^{(1)} \leq b_1$. The respective sample distances $R_n^{(1)}$ from C are less than or equal to the level 1 threshold b_1. This is to enable



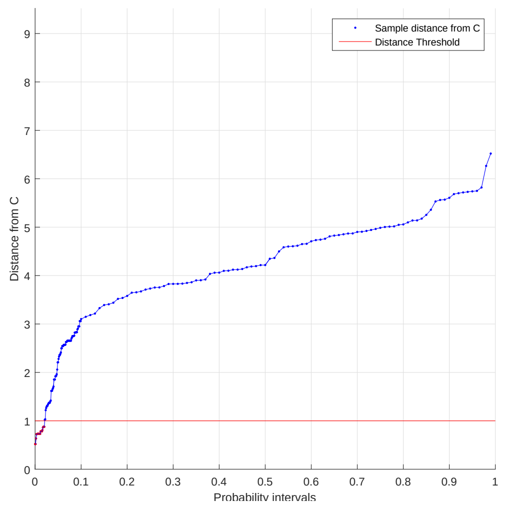
(a) Subset Simulation Level 0



(b) Subset Simulation Level 0 CCDF



(c) Subset Simulation Level 1



(d) Subset Simulation Level 1 CCDF

Figure 6.3: Subset Simulation is applied to the problem of estimating the probability of drawing samples from the region F. Subset Simulation begins with level 0 by drawing $N = 100$ samples from a Gaussian distribution centred at $O = [0, 0]$ using the DMC method as shown in Figure 6.3(a). The quantity of interest is the distance between each sample and C. These are plotted against probability intervals to generate a CCDF as shown in Figure 6.3(b). No samples are within the region F. The SS method proceeds to level 1 and conditional samples are generated using the MH method. The N_c level 0 samples are used to generate the conditional samples shown in Figure 6.3(c). These conditional samples are drawn progressively closer to the region F until some samples are drawn from the region F. This is achieved by drawing samples from intermediate thresholds closer to the boundary of F. The quantity of interest for the samples are determined and plotted against the probability intervals for the current level. This CCDF is appended to the previous CCDF by replacing the samples used as seeds from the previous level as shown in Figure 6.3(d).

Level 0			Level 1		
$\mathbf{P}_n^{(0)}$	$\mathbf{B}_n^{(0)}$	$\tilde{\mathbf{X}}_n^{(0)}$	$\mathbf{P}_n^{(1)}$	$\mathbf{B}_n^{(1)}$	$\tilde{\mathbf{X}}_n^{(1)}$
$P_1^{(0)}$	$B_1^{(0)}$	$\tilde{X}_1^{(0)}$			
$\vdots$	$\vdots$	$\vdots$			
$P_{90}^{(0)}$	$B_{90}^{(0)}$	$\tilde{X}_{90}^{(0)}$			
$P_{91}^{(0)}$	$B_{91}^{(0)}$	$\tilde{X}_{91}^{(0)}$	$P_1^{(1)}$	$B_1^{(1)}$	$\tilde{X}_1^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$P_{100}^{(0)}$	$B_{100}^{(0)}$	$\tilde{X}_{100}^{(0)}$	$P_{90}^{(1)}$	$B_{90}^{(1)}$	$\tilde{X}_{90}^{(1)}$
			$P_{91}^{(1)}$	$B_{91}^{(1)}$	$\tilde{X}_{91}^{(1)}$
			$\vdots$	$\vdots$	$\vdots$
			$P_{100}^{(1)}$	$B_{100}^{(1)}$	$\tilde{X}_{100}^{(1)}$

(a)

$\mathbf{P}_n$	$\mathbf{B}_n$	$\tilde{\mathbf{X}}_n$
$P_1^{(0)}$	$B_1^{(0)}$	$\tilde{X}_1^{(0)}$
$\vdots$	$\vdots$	$\vdots$
$P_{90}^{(0)}$	$B_{90}^{(0)}$	$\tilde{X}_{90}^{(0)}$
$P_1^{(1)}$	$B_1^{(1)}$	$\tilde{X}_1^{(1)}$
$\vdots$	$\vdots$	$\vdots$
$P_{90}^{(1)}$	$B_{90}^{(1)}$	$\tilde{X}_{90}^{(1)}$
$P_{91}^{(1)}$	$B_{91}^{(1)}$	$\tilde{X}_{91}^{(1)}$
$\vdots$	$\vdots$	$\vdots$
$P_{100}^{(1)}$	$B_{100}^{(1)}$	$\tilde{X}_{100}^{(1)}$

(b)

Table 6.3: Samples generated during both levels of Subset Simulation.

a progressive nature of drawing samples that are closer to the region F. The conditional samples are generated using algorithm 8. This will eventually lead to samples being drawn from the region F as SS proceeds to higher number of levels in the future. The level 1 threshold is marked by the dotted arc in Figure 6.3(c). The figure shows chains of samples that lead to the region F. The distances $R_n^{(1)}$ of samples $X_n^{(1)}$ generated in level 1 are sorted in descending order $\{B_n^{(1)} : n = 1, ..., 100\}$. The input samples $X_n^{(1)}$ are reordered $\tilde{X}_n^{(1)}$ and correspond to the sorted distances $B_n^{(1)}$. The probability intervals $P_n^{(1)}$ are generated using equation (6.2) and concatenated with the sorted distances $B_n^{(1)}$ and their corresponding samples $\tilde{X}_n^{(1)}$. Table 6.3a illustrates the conditional samples generated in level 1 using samples from level 0. The seeds used to generate samples in level 1 are discarded and replaced with the generated level 1 samples as illustrated in Table 6.3b. Note the probability intervals $\{P_n^{(0)} : n = 91, ..., 100\}$, sorted distances $\{B_n^{(1)} : n = 91, ..., 100\}$ and the corresponding input samples $\{\tilde{X}_n^{(1)} : n = 91, ..., 100\}$ from level 0 that were used as seeds to generate the samples for level 1 are discarded and replaced with level 1 samples $\tilde{X}_n^{(1)}$ and their respective distances $B_n^{(1)}$ and probability intervals $P_n^{(1)}$. This process is repeated until the maximum number of levels m is reached. This is when $i = m - 1$. Figure 6.3(d) shows the overall CCDF at level 1. The overall CCDF is used to estimate the probability of drawing samples from the region F as approximately $P_F = 0.02$.

This example demonstrates the progressive nature of Subset Simulation when used to generate conditional samples to realise the rare 'tail' region of the pdf. This feature of SS results in the empirical observation that SS requires significantly fewer samples when compared to naive DMC to obtain estimates with the same accuracy. Subset Simulation is useful for generating samples that progress to the distribution of interest.

6.4 Summary

This chapter has familiarised the reader with the Subset Simulation method and its utility of estimating low probabilities as a product of larger probabilities. The Direct Monte Carlo and Metropolis Hastings methods that the Subset Simulation method is based on, has been explained.

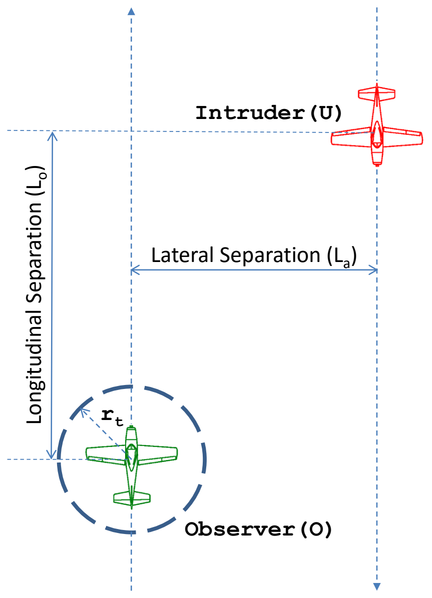
Chapter 7

Application of Subset Simulation for Airborne Conflict detection

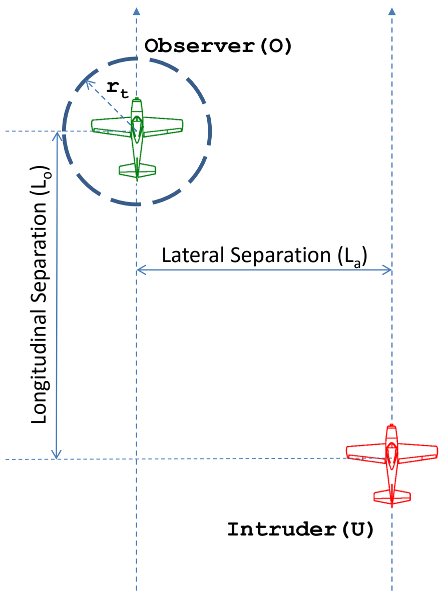
The estimation of the probability of conflict P_c between air traffic is a useful metric for Conflict Detection & Resolution (CD&R) methods. Such methods can be used in piloted aircraft but are useful for UAS where an automated method for CD&R will be required as part of a Sense-and-Avoid system [44].

The safe operation of Unmanned Aircraft System alongside manned aircraft within non-segregated airspace requires the ability to maintain safe separation between traffic. Although a minimum threshold has not been defined by the International Civil Aviation Organization (ICAO), Aviation Authorities and Regulatory Bodies implicitly understand that a minimum threshold of 500 ft must be maintained in all directions. A conflict event occurs when two or more aircraft collide or if there is a loss of this separation between them within a block of airspace. The conflict type depends on the geometry of the encounter between traffic, as defined in reference [20] (mentioned in chapter 2). If a conflict is detected, the conflict type needs to be identified so that the appropriate resolution manoeuvre can be executed by the CD&R system to resolve the conflict. This chapter addresses a key component of a detection of a conflict by estimating the probability of conflict P_c using Subset Simulation.

This chapter begins by defining the test scenarios used to estimate the P_c between air traffic using the Subset Simulation method in section 7.1. Section 7.2 defines the application of Subset Simulation for estimating the probability of conflict between traffic. Section 7.3 estimate the probability of conflict between traffic during a series of conflicting and potentially conflicting scenarios using Subset Simulation and Direct Monte Carlo methods. Section 7.4 compares the accuracy and efficiency of estimating low probabilities of conflict P_c using Subset Simulation and Direct Monte Carlo method.



(a) Head-on pass



(b) Intruder overtaking Observer

Figure 7.1: The potentially conflicting scenarios based on the different conflicts shown in Figure 2.1

7.1 Test scenarios

A non-cooperative scenario is assumed, where the traffic does not share information. This is a challenging situation since the information related to the state and intentions of the traffic might be unknown or incorrect. The only information available regarding the state of traffic is from measurements or inference using sensors. In such a scenario, CD&R system must allow for the possibility that the non-cooperative traffic may take inappropriate actions or may not adhere to the Rules of the Air. This type of situation requires a UAS to react and take appropriate action to ensure safe separation. To achieve this the P_c needs to be continuously evaluated against the behaviour of the observed traffic so that the likelihood of the traffic causing a conflict can be calculated. Figure 7.1 illustrates some potentially conflicting scenarios based on Figure 2.1. During some phases of the scenario, the expected P_c can be very low; such as a magnitude of 10^{-8} (this is demonstrated later in this section). The previous sections have demonstrated that estimating low probabilities using the Direct Monte Carlo method is inefficient and this motivates the use of Subset Simulation (SS). Assessing the full pdf may not be feasible and may not be required. Subset Simulation provides an efficient method of determining the probability associated with all predicted conflicts thereby estimating P_c. In applying SS to this problem, P_c plays the role of the threshold of failure P_F.

The Subset Simulation method is used to estimate the probability of conflict P_c during the simulation of the potentially conflicting scenarios of the Observer and Intruder aircraft in the Head-on and Overtaking situations as shown in figures 7.1(a) and 7.1(b) respectively. Both scenarios show the Observer and Intruder in a non-conflicting a state, where the Intruder is not within the Observer's protected zone. The Observer's protected zone is marked as a circle around the Observer with radius $r_t = 152.4$m (500ft). Although the current state is non-conflicting there is a potential for future conflict. For example from the Observer's perspective the Intruder could continue on its course or turn right or turn left. The latter could cause a loss of separation or worse – a collision between the Observer and the Intruder. Also in the situation when the lateral separation L_a between the Observer and Intruder is lower than or equal to the radius of the Observer's protected zone r_t; $(r_t \leq L_a)$ a conflict occurs due to loss of separation or collision between the Observer and the Intruder. Therefore, the likelihood of such conflict needs to be realised by estimating P_c.

The Subset Simulation method is used by the Observer to determine the probability of conflict P_c between itself and the approaching Intruder for the potentially conflicting scenarios shown in Figure 7.1. However, since some parameters are not available this requires the method to be adapted. The order of magnitude for the target probability (conflict) region $(p_0)^m$ is unknown. The solution to this problem is addressed later in this section. Therefore the number of subset levels m required to reach the target probability level with a fixed p_0 is unknown. The Intruder and Observer are simulated as *nearly constant acceleration* point models [129]. This is a simple model that is used to illustrate the use of Subset Simulation. It can be augmented by more complex dynamic models such as Six-Degrees-of-Freedom (SixDoF) aircraft model presented in chapter 3. This would not affect the use of Subset Simulation and the computational advantages that it provides. The dynamics of the Intruder and Observer are modelled in state space form as $U(K+1) = AU(K)$ and $O(K+1) = AO(K)$ respectively in two-dimensional Cartesian space, where K is the time–step index. The Intruder and Observer statevectors are $U(K) = [x, u, a_x, y, v, a_y]^T$ and $O(K) = [x, u, a_x, y, v, a_y]^T$ respectively. The displacement, velocity and acceleration in the x-direction are represented by x, u and a_x respectively. The displacement, velocity and acceleration in the y direction are represented by y, v and a_y respectively. The state transition matrix A is defined

$$A = \begin{bmatrix} 1 & \Delta T & \frac{1}{2}\Delta T^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta T & \frac{1}{2}\Delta T^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.1}$$

where ΔT is the period of discretized time-step. The sampling frequency $f = \frac{1}{\Delta T}$. The Observer estimates the state of the Intruder $\hat{U}(K)$ using a Kalman Filter [93]. The periodic measurements of the Intruder's position $Z = [x, y]$ is defined by the measurement equation as

$$Z = HU(K) + [w_x, w_y]' \tag{7.2}$$

where H is the measurement matrix.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{7.3}$$

$$w_x \sim \mathcal{N}(0, \sigma_x) \tag{7.4}$$

$$w_y \sim \mathcal{N}(0, \sigma_y) \tag{7.5}$$

The periodic position measurements are simulated by adding noise as w_x and w_y to the x and y directions respectively. The standard deviation of the of the measurement error in the x and y directions are σ_x and σ_y respectively. For the sake of simplicity the measurement noise is uncorrelated. The instantaneous state-estimate of the Intruder is determined using a Kalman Filter. The Intruder's state-estimate $\hat{U}(K+1)$ and covariance $\hat{S}(K+1)$ is predicted using equations

$$\hat{U}(K+1) = A\hat{U}(K) \tag{7.6}$$

$$\hat{S}(K+1) = A\hat{S}(K)A^T + Q \tag{7.7}$$

The process noise covariance is Q. This is the *white-noise jerk* version of the *Wiener-Process Acceleration* model [129].

$$Q = \begin{bmatrix} Q_\sigma \frac{\sigma_{a_x}^2}{\Delta T} & 0 \\ 0 & Q_\sigma \frac{\sigma_{a_y}^2}{\Delta T} \end{bmatrix} \tag{7.8}$$

$$Q_\sigma = \begin{bmatrix} \frac{1}{20}\Delta T^5 & \frac{1}{8}\Delta T^4 & \frac{1}{6}\Delta T^3 \\ \frac{1}{8}\Delta T^4 & \frac{1}{3}\Delta T^3 & \frac{1}{2}\Delta T^2 \\ \frac{1}{6}\Delta T^3 & \frac{1}{2}\Delta T^2 & \Delta T \end{bmatrix} \tag{7.9}$$

The parameters $\sigma_{a_x}^2$ and $\sigma_{a_y}^2$ are the variance of acceleration parameters in the x and y directions respectively. The Kalman gain G is evaluated during the update stage:

$$G = \hat{S}(K+1)H^T([H\hat{S}(K+1)H^T] + R)^{-1} \tag{7.10}$$

where R is the measurement covariance.

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{7.11}$$

This is followed by updating the Intruder estimate $\hat{U}(K+1)$ and error covariance $\hat{S}(K+1)$ respectively.

$$\hat{U}(K+1) = \hat{U}(K+1) + G\{Z(K) - [H\hat{U}(K+1)]\} \tag{7.12}$$

$$\hat{S}(K+1) = [I - GH]\hat{S}(K+1) \tag{7.13}$$

7.2 Estimating Probability of Conflict using Subset Simulation

The Subset Simulation method is applied to the Head-on pass scenario with lateral separation $L_a = 1000$m and longitudinal separation $L_o = 2000$m. The duration of the simulation $t = 20$s with sampling frequency $f = 20$Hz and the measurement frequency $f_M = 2$Hz. The initial conditions of the Intruder and Observer are $O(0) = [0, 77.2\text{ms}^{-1}, 0, 0, 0, 0]^T$ and $U(0) = [2000\text{m}, -77.2\text{ms}^{-1}, 0, 1000\text{m}, 0, 0]^T$. The Observer's protected zone radius $r_t = 152.4$m.

Kalman Filter parameters:

- $\sigma_x = 0.1$m
- $\sigma_y = 0.1$m
- $\sigma_{a_x}^2 = 0.01\text{m}^2\text{s}^{-4}$
- $\sigma_{a_y}^2 = 0.01\text{m}^2\text{s}^{-4}$

Subset Simulation parameters:

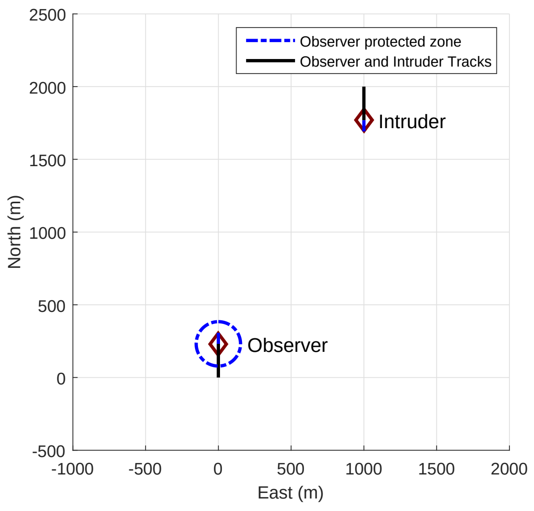


Figure 7.2: Head-on pass scenario with 1000m Lateral Separation

- $N = 100$
- $p_0 = 0.1$
- $N_c = 10$
- $N_s = 10$
- $m = 7$

Ideally the SS method should continue to higher levels of simulation until conflicting samples are encountered and P_c can be estimated using the CCDF. This is assuming infinite simulation resources are available. This is impractical for implementation since simulation capacity is limited due to limited resources available. Therefore the SS method implemented requires a limited number of levels[1] to be defined m.

Subset Simulation estimates $P_c(K+1)$ where $K+1$ is the time-step of an instance during the simulation as shown in Figure 7.2. Subset Simulation begins with level 0 Direct Monte Carlo sampling. A set of 100 samples $\{U_n^{(0)} : n = 1, ..., 100\}$ representing

[1]An alternative implementation: During the process of SS estimating the P_c; the SS method continues to higher levels until conflicting samples are found. If new information is received (such as a new Intruder measurement that updates the Intruder state-estimate) and the SS method has not found conflicting samples, then the calculation for the current time-step should be abandoned and restarted with the new information. Restarting is necessary since the information used to calculate P_c becomes obsolete once more recent information is obtained. This approach would be useful for situations where real-time computation is enforced. Note that this chapter does not enforce constraints associated with real-time computation.

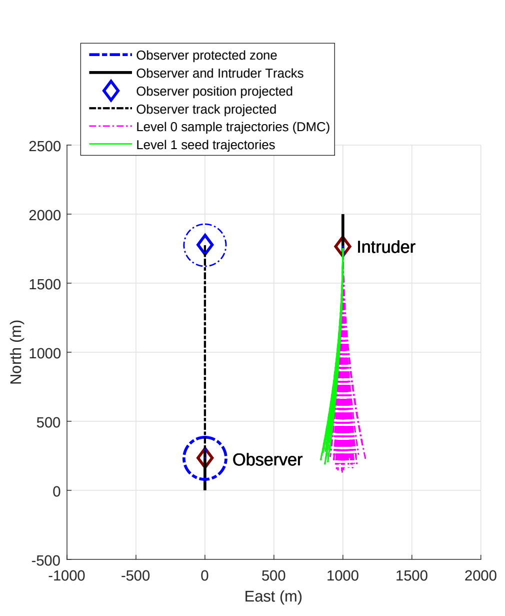
the Intruder's pdf are drawn from the distribution that is centred at the Intruder's mean $\hat{U}(K+1)$ and covariance $\hat{S}(K+1)$. The mean and covariance are obtained from the Kalman filter defined in algorithm 13.

The set of samples $U_n^{(0)}$ and the intended vector of the Observer $O(K)$ are propagated to generate trajectories $J_{U_n}^{(0)}$ and J_O respectively. A trajectory J is a set of consecutive state vectors indexed by the time-step k where $k = 1, ..., tf = 1, ..., 400$ and $f = 20\text{Hz}$ is the sampling frequency (as defined in algorithm 10). For example the Observer trajectory $J_O = [O(1), ..., O(tf)] = [O(1), ..., O(400)]$, where $O(1)$ is the state vector of the Observer at time-step $k = 1$. The propagation time $t = 20\text{s}$. This is also the period of the simulation. Figure 7.3(a) shows the Intruder samples and the respective trajectories generated with the projected position of the Observer during level 0 for a Head-on pass scenario with lateral separation $L_a = 1000\text{m}$. No conflicting samples have been encountered yet. A conflicting sample is an Intruder sample $U_n^{(i)}$ generated in level i with a trajectory $J_n^{(i)}$ that has a miss-distance $r_n^{(i)}$ between the Observer trajectory J_O and satisfies the conflict condition $r_n^{(i)} \leq r_t$. The number of conflicting samples encountered in a level is D.

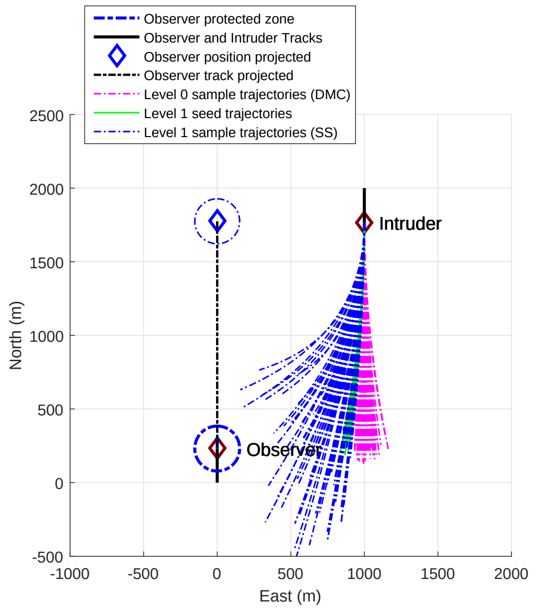
The quantities of interest are the miss-distances $\{r_n^{(0)} : n = 1, ..., 100\}$. These are the minimum distances between the Intruder samples' trajectories $\{J_{U_n}^{(0)} : n = 1, ..., 100\}$ and the Observer trajectory J_O. Algorithm 11 defines the procedure to determine the miss-distances between the Observer and Intruder trajectories. A conflict is projected to occur when there is a loss of minimum separation between any sample in set J_{U_n} and the Observer trajectory J_O at any instance. The set of miss-distances $r_n^{(0)}$ are sorted in descending order $\{B_n^{(0)} : n = 1, ..., 100\}$. The input samples $U_n^{(0)}$ are reordered $\tilde{U}_n^{(0)}$ to correspond to the sorted miss-distances $B_n^{(0)}$. To clarify, the sample $\tilde{U}_1^{(0)}$ produces a trajectory $J_{\tilde{U}_1}$ that has the largest miss-distance $B_1^{(0)}$ between itself the trajectory produced by the Observer J_O. The samples with lower miss-distances in the current level have a higher likelihood of generating conditional samples that satisfy the conflict condition than other samples in the current level. The vector of probability intervals $P_n^{(0)}$ are generated by

$$P_{n+1}^{(i)} = p_0^i \frac{N-n}{N} \quad n = 0, ..., (N-1) \tag{7.14}$$

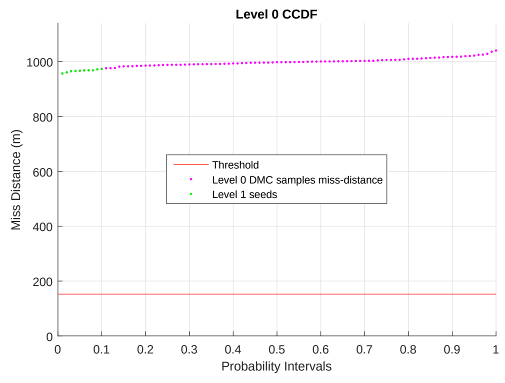
Note that the range of n in this equation is different to equation 6.2. This is due to the maximum number of levels limit m. In the event that SS reaches the maximum number of levels without encountering conflicting samples the probability of conflict will be estimated $P_c = P_N^{(m-1)} = P_{100}^{(m-1)} = 0$ (the last probability interval in the $P_n^{(m-1)}$ vector that is generated by equation 6.2) and this does not reflect the low magnitude of the probability. In contrast, the probability interval generated by equation 7.14 allows the probability of conflict to be estimated $P_c < P_{100}^{(m-1)}$; $P_{100}^{(m-1)} = 1 \times 10^{-8}$. This information means that although no conflicting samples have been encountered despite exhausting all levels of SS the expected P_c is estimated to be lower than $(p_0)^m$, the lowest probability level realisable due to the maximum number of levels limit reached by SS. Such information is more useful than the estimate $P_c = 0$ evaluated by equation 6.2. The level 0 CCDF is constructed by plotting the probabilities $P_n^{(0)}$ against $B_n^{(0)}$ as shown in Figure 7.3(c). No conflicting samples have been drawn in level 0 since no miss-distances satisfy the conflict condition. If the number of conflicting samples $D > N_c$ then the probability of conflict is estimated $P_c = P_{(N-D+1)}^{(i)}$. This also applies for the situation where the maximum number of levels has been reached $(i = m - 1)$ and some conflicts have been encountered where the number of conflicts encountered is less than or equal to N_c; $(N_c \geq D > 0)$. The DMC method estimates the probability of conflict $P_c = \frac{D}{N}$ as defined by algorithm 12.



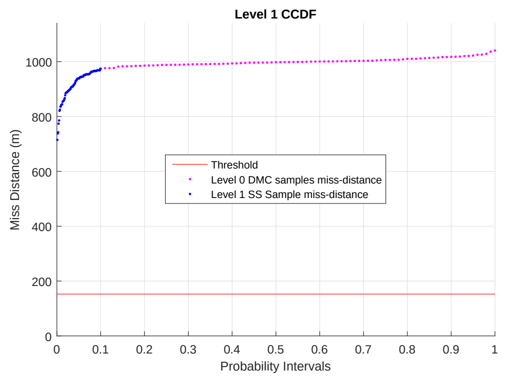
(a) Level 0 DMC sample trajectories with highlighted trajectories of samples used as seeds



(b) Trajectories for Level 1 samples generated using N_c samples from level 0

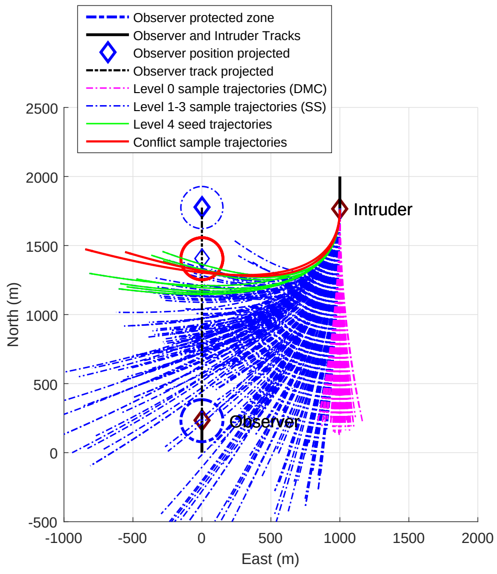


(c) Level 0 CCDF with miss-distance of Level 1 seeds highlighted

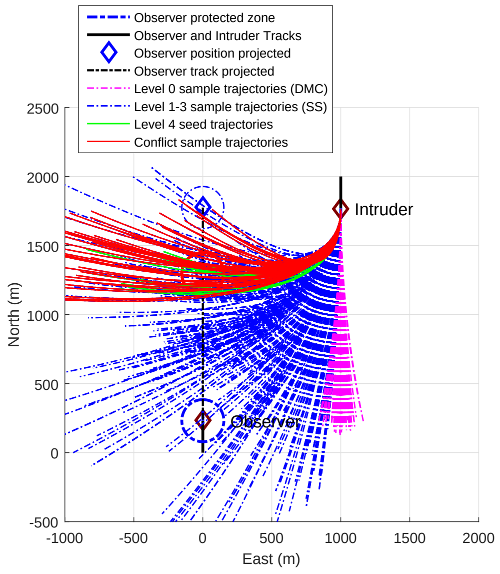


(d) Level 1 CCDF

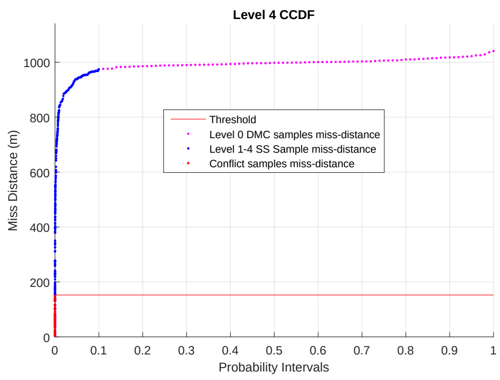
Figure 7.3: These figures illustrate the application of SS to estimate the $P_c(K+1)$ *at the time-step* $K+1$ *during a Head-on pass between an Observer* $O(K)$ *and Intruder* $U(K)$ *with lateral separation of 1000m. SS begins with level 0 (DMC) where* $N = 100$ *samples are drawn from a distribution centred at the Intruder's state-estimate* $\hat{U}(K+1)$ *with a covariance of* $\hat{S}(K+1)$ *obtained from the Kalman Filter. Figure 7.3(a) shows trajectories generated by level 0 samples, no conflicting samples have been encountered. The simulation proceeds to level 1 where conditional samples are generated using* N_c *samples from level 0 as seeds. The trajectories of the level 0 samples used as seeds are highlighted in Figure 7.3(a). The MH method is applied to generate conditional samples from the seeds. The trajectories of generated samples for level 1 are shown in Figure 7.3(b). This process is continued to generates more trajectories as the number of levels increase. The method continues until conflicting samples are encountered at higher levels as shown in Figure 7.4.*



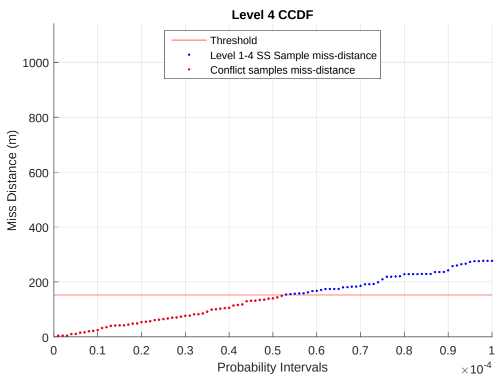
(a) Sample Trajectories for levels 0 to 3



(b) Samples Trajectories for levels 0 to 4



(c) Level 4 CCDF



(d) Level 4 CCDF

Figure 7.4: The above figures show trajectories of conditional samples generated as the simulation continues to higher levels. Subset Simulation continues until the number of conflicting samples D found in a level is greater than N_c within a level as shown in Figure 7.4(b). The probability of conflict is estimated as $P_c(K+1) = 0.52 \times 10^{-4}$ as shown in Figure 7.4(d).

However if the condition $D > N_c$ is not satisfied and $i < m - 1$; SS proceeds to the next level ($i > 0$) and continues until the condition is satisfied or if the maximum number of levels is reached. This is because the conflict region of the pdf is not represented accurately enough due to the lack of sufficient samples representing the conflict region in the current level. Therefore it is necessary generate more conditional samples at higher levels of SS to progress towards representing the conflict region of the pdf more accurately.

The following subset levels ($i > 0$) generate N conditional Intruder samples $\tilde{U}_n^{(i)}$ using the Metropolis Hastings method as defined in algorithm 14. The set of seeds $s_j^{(i)}$

required to generate the samples are selected from samples in the previous level using

$$s_j^{(i)} = \tilde{U}_n^{(i-1)} \tag{7.15}$$

where $1 \leq j \leq N_c$, $(N - N_c + 1) \leq n \leq N$ and $i > 0$.

Figure 7.3(a) highlights the trajectories of level 0 samples selected as seeds to generate level 1 conditional samples. Figure 7.3(b) shows the trajectories of the conditional samples generated in level 1. The set $s_j^{(i)}$ contains N_c seeds; one for each chain. Each chain generates N_s samples. This maintains the total number of samples as N for each level. The MH method uses an indicator d (as shown in algorithm 14) to ensure the miss-distance $r^{(i)*}$ between the Observer's trajectory J_O and Intruder trajectory $J^{(i)*}$ of the proposed sample $U^{(i)*}$ is less than the intermediate threshold b_i set by equation 6.5. If $r^{(i)*} > b_i$ then the proposed sample is rejected and the current sample of the Intruder is maintained.

The miss-distances $\{r_n^{(1)} : n = 1, ..., 100\}$ of the conditional samples $U_n^{(1)}$ generated in level 1 are determined and sorted in descending order $B_n^{(1)}$ using the same method as level 0. The input samples $U_n^{(1)}$ are reordered $\tilde{U}_n^{(1)}$ to correspond to the sorted miss-distances $B_n^{(1)}$. The probability intervals $P_n^{(1)}$ for the current level are generated and plotted against $B_n^{(1)}$ to construct a CCDF. Figure 7.3(d) shows the CCDF generated up to level 1. Note the miss-distances of the samples used as seeds from the previous level 0 (that are highlighted in Figure 7.3(c)) are discarded and replaced with the miss-distances of the conditional samples generated in level 1. This illustrates that the samples used as seeds are discarded and replaced with the conditional samples generated in the current level. This process is repeated as SS progresses to higher levels until the condition $D > N_c$ is satisfied or the maximum number of levels is reached as defined in algorithm 15. Figure 7.4(a) shows the trajectories of the conflicting samples encountered in level 3. However the condition $D > N_c$ had not been satisfied. This required SS to proceed to level 4 and generate conditional samples that satisfy the condition $D > N_c$ as shown in Figure 7.4(b). The CCDF generated up to level 4 is shown in Figure 7.4(c). The CCDF is used to estimate the $P_c(K + 1) = 0.52 \times 10^{-4}$ as shown in Figure 7.4(d). This process is repeated through out the duration of the simulation to determine the probability of conflict for each time-step using samples from the prediction of the Intruder's estimate $\hat{U}(K + 1)$ and covariance $\hat{S}(K + 1)$.

7.3 Results

The Subset Simulation method has been tested and compared with the Direct Monte Carlo (DMC) method to estimate the probability of conflict P_c between the Observer and Intruder by simulating the scenarios shown in Figure 7.1. The Observer and Intruder were modelled as points with nearly constant velocity in a geometric configuration based on the three different types of conflict shown in Figure 2.1. The P_c metric was estimated as an average of 50 Monte Carlo simulation during the Head-on and Overtaking conflicts as shown in figures 7.1(a) and 7.1(b) respectively. The tests were repeated with varying lateral separations $L_a = \{0, 100, 152, 500, 1000, 1100\}$m.

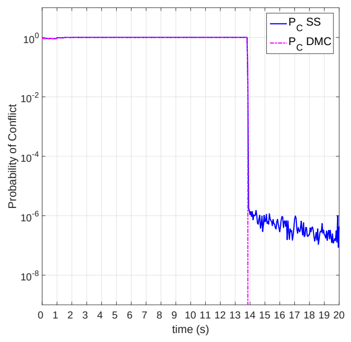
The following Subset Simulation parameters were used for all scenarios: $N = 100$; Level probability: $p_0 = 0.1$; $N_c = p_0 N = 10$; $N_s = \frac{1}{p_0} = 10$; $m = 7$; Observer minimum separation threshold $r_t = 500\text{ft} = 152.4\text{m}$. Algorithm 16 defines the simulation conducted.

The number of samples used for each level of SS remain constant. However the number of levels required at a given time-step vary depending on the magnitude of P_c. Therefore the total number of samples N_T required to realise a conflict at a given time-step varies as a function of time-step. In the interest of a fair comparison of the computational effort between the two methods, an equal number of samples are evaluated for both methods. The estimation using DMC is conducted with N_T samples, where N_T is the number of samples that are used in the SS method at the same time-step. To clarify, if the SS method reaches level $i = 4$ to satisfy the conflict condition for estimating the $P_c^{(\text{SS})}(K)$ at time-step K, then $N_T = 100 \times 5 = 500$ samples have been used by the SS method. Therefore DMC estimates the $P_c^{(\text{DMC})}(K)$ for the same time-step with 500 samples only.

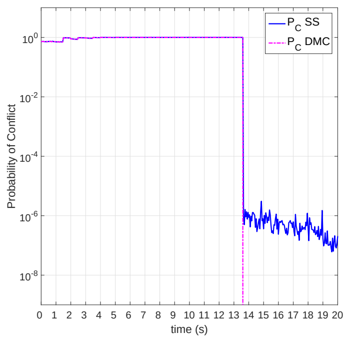
7.3.1 Estimation of P_c for Head-on Pass scenario

The Intruder and Observer parameters used for the Head-on pass scenario are as follows: The Intruder and Observer maintain a constant speed of 150 knots (77.17ms^{-1}). The Observer maintains a constant heading of $0°$; the Intruder maintains a constant heading of $180°$. The Observer's minimum separation threshold is $r_t = 500\text{ft} = 152.4\text{m}$. The Longitudinal separation is $L_o = 2000\text{m}$

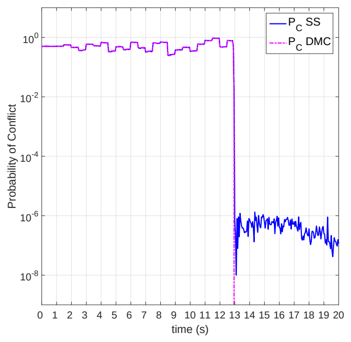
Figures 7.5(a), 7.5(b) and 7.5(c) show the estimation of P_c for the Head-on pass scenario using SS and DMC methods with lateral separations of 0m, 100m and 152m respectively. The scenarios are conflicting because the geometric configuration and initial conditions of both the Observer and Intruder are conflicting and remain as such throughout the duration of the simulation. When $t \leq 12$s the Intruder and Observer are approaching each other the estimated P_c increases. This is as expected because a conflict is imminent. Both estimation methods show approximately the same P_c as expected, since the first level of the SS method is DMC sampling. At this stage the conflict region of the pdf is large and the probability of drawing a sample which leads to a conflict is high. The conflict occurs at $t \approx 12.5$s due to the loss of separation between the Observer and Intruder. Figure 7.5(c) shows the estimation of P_c with lateral separation $L_a = 152.4m = r_t$. This is a conflicting scenario since the Intruder skims Observer's protected boundary at $t \approx 12.5$s as the Observer and Intruder pass



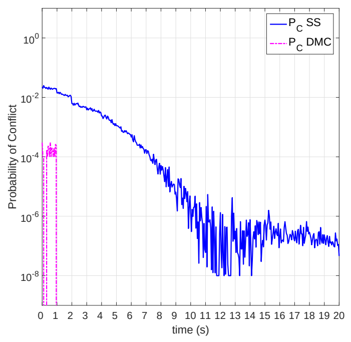
(a) P_c during Head-on conflict with 0 m Lateral separation



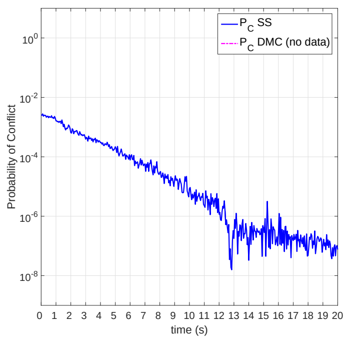
(b) P_c during Head-on conflict with 100m Lateral separation



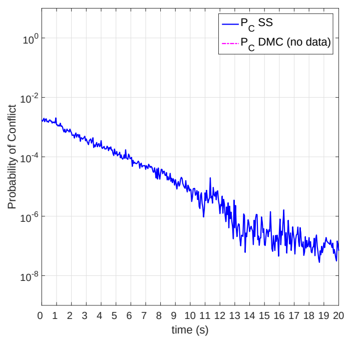
(c) P_c during Head-on conflict with 152.4m Lateral separation



(d) P_c during Head-on pass with 500m Lateral separation

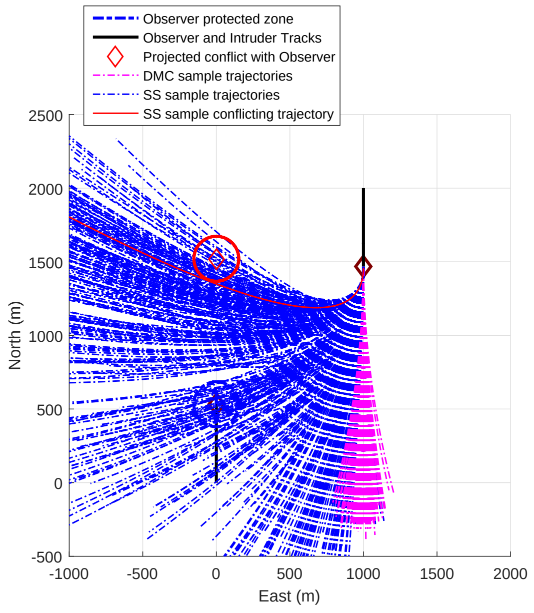


(e) P_c during Head-on pass with 1000m Lateral separation

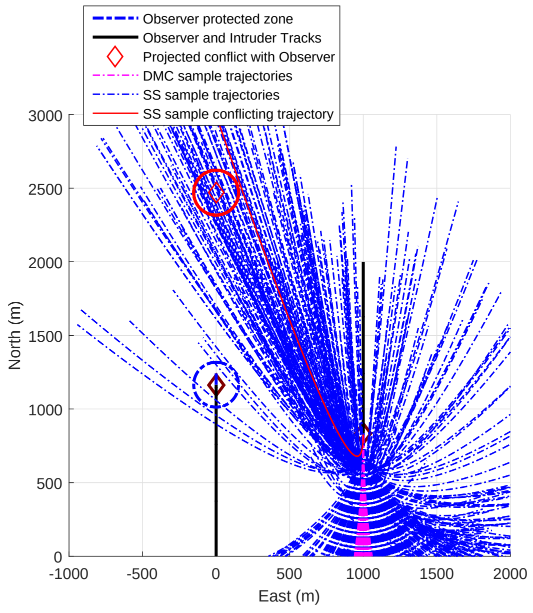


(f) P_c during Head-on pass with 1100m Lateral separation

Figure 7.5: The estimated P_c using the Subset Simulation and Direct Monte Carlo methods during the Head-on pass as shown in Figure 7.1a with varying lateral separation $L_a = \{0, 100, 152, 500, 1000, 1100\}m$.



(a) Head-on conflict scenario with 1000m Lateral separation before head-on pass.



(b) Head-on conflict scenario with 1000m Lateral separation after pass.

Figure 7.6: SS and DMC trajectories for Head-on pass with lateral separation 1000m

each other. The oscillations during $t \leq 12s$ are due to $L_a = r_t$. This is a borderline situation.

The Intruder and Observer pass each other at $t \approx 13$s. The P_c estimated by both methods is still 1 until $t > 14$s where the Intruder has exited the Observer's protected zone. At this stage the Observer and Intruder have receding relative velocities and are moving away from each other. P_c is expected to reduce at this stage as shown in the log-y plot. The conflict region of the pdf reduces since both Intruder and Observer are moving away from each other. The SS method estimates the P_c as being close to zero at an order of magnitude of 10^{-7}. The lowest probability which can be realised is $P_c = 10^{-8}$. This is due to a maximum level restriction imposed in the simulation. In such instances the probability of conflict can be considered to be less than the order of 10^{-8}. At this stage the DMC method draws the same number of samples as SS but is unable to find conflicting samples and estimates $P_c = 0$. This is because the region of conflict within the pdf has reduced and the probability of drawing a conflicting sample is rare. This requires the DMC method to draw and evaluate a larger number of samples at this stage before a conflicting sample is drawn from the rare region of conflict within the pdf. The SS method is able to obtain the conflicting samples from the rare region of the pdf by generating samples conditionally in such a way that the samples satisfy the intermediate thresholds leading to the rare region using the MH method. Each subset level corresponds to an intermediate threshold. This progressive feature of the SS method allows a more efficient approach to reach the rare ‘tail’ region of the pdf.

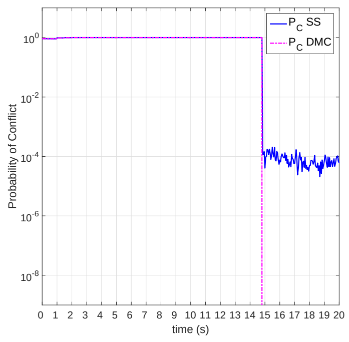
As the lateral separation of the scenario is increased, the expected P_c decreases. The scenario is simulated with a lateral separation of 500m, 1000m and 1100m as shown in figures 7.5(d), 7.5(e) and 7.5(f) respectively. These are non-conflicting scenarios. The figures show abrupt variations in P_c. These are caused by the Monte Carlo nature of our algorithm. Note that, since the sampling frequency is high relative to the thickness of the line in the figure, the variations in P_c are particularly readily perceived. The conflict region of the pdf is smaller than the previous scenarios. The SS estimation method is able to estimate low P_c throughout the duration of the simulation, whereas with an equivalent number of samples the DMC method is unable to find conflicting or near conflicting samples of the Intruder in most instances. Figure 7.5(d) shows abrupt variations in the P_c estimated by the DMC method when $t < 1$s where the estimate tends to zero. These are instances where the DMC method is unable for find any conflicting samples and estimates $P_c = 0$.

Figures 7.6(a) and 7.6(b) show the trajectories of the samples evaluated by SS and DMC methods at an instance before and after the Intruder and Observer pass each other respectively. The progressive nature of the SS method can be observed as a concentration of trajectories leading to the conflict trajectory. In contrast the DMC method has drawn the same number of samples (most are overlapping) without realising any conflicts.

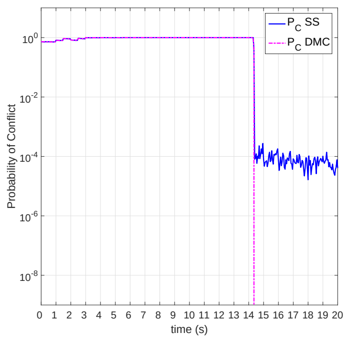
7.3.2 Estimation of P_c for Intruder Overtaking Observer

The scenario parameters used are as follows: The Intruder speed is 300knots = 154.3ms^{-1} and the Observer speed is 150knots = 77.17ms^{-1}. Both Intruder and Observer maintain a constant heading of a constant heading of 180°. The longitudinal distance L_o between the Intruder and Observer is $L_o = 1000$m.

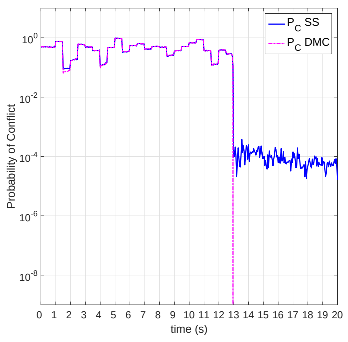
Both SS and DMC methods have been applied to the Overtaking scenario as shown in Figure 7.1(b). Similar to the previous scenario, the SS method is able to obtain samples from the rare conflicting region of the pdf consistently throughout the duration of the simulation for this scenario. As the lateral separation increases, the P_c decreases (as expected). Figures 7.7(e) and 7.7(f) show the P_c when the lateral separation is 1000m and 1100m respectively. The change in P_c is less abrupt compared to the 100m lateral separation after the Intruder as passed the Observer when $t > 13$s. The P_c is approximately the same throughout the duration of the simulation. This is because the increased lateral separation includes samples with low turn rates in the conflict category and these are common enough to be drawn by the DMC method and SS method. With low lateral separation the conflicting samples will need high turn rates. These are rare and are realised by using SS method. In contrast the DMC method is unable to realise



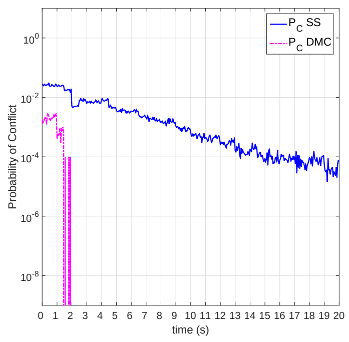
(a) P_c during Intruder overtaking Observer conflict with 0m Lateral separation



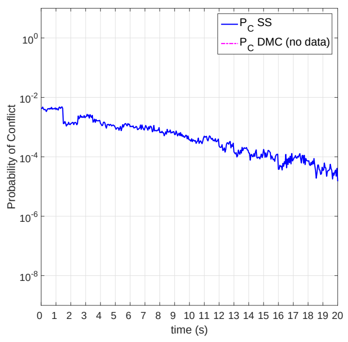
(b) P_c during Intruder overtaking Observer conflict with 100m Lateral separation



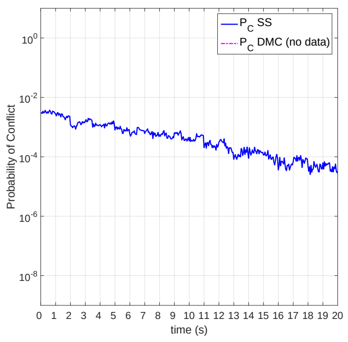
(c) P_c during Intruder overtaking Observer conflict with 152m Lateral separation



(d) P_c during Intruder overtaking Observer with 500m Lateral separation



(e) P_c during Intruder overtaking Observer with 1000m Lateral separation



(f) P_c during Intruder overtaking Observer with 1100m Lateral separation

Figure 7.7: The P_c is estimated using the SS and DMC methods during the Intruder Overtaking the Observer scenario as shown in Figure 7.1(b) with varying lateral separation $L_a = \{0, 100, 152, 500, 1000, 1100\}m$.

them. Also throughout the simulation, the relative change in angle of the Intruder from the Observer's perspective reduces as the lateral separation is increased. The conflicting samples can have lower turn rates despite the Intruder having passed the Observer. Such samples are common and can be realised by both methods.

7.4 Accuracy and Efficiency of Subset Simulation

A range of magnitudes of probabilities have been evaluated within the simulated scenarios shown in the previous section. This section analyses the accuracy and efficiency of using the Subset Simulation and Direct Monte Carlo methods to estimate probabilities at each of a number of orders of magnitude. In order for a fair comparison to be conducted – a common phase within a simulation scenario must be found where both methods are able to realise conflicting samples and estimate the probability of conflict.

The first order of magnitude considered for comparison is $P_{c_1} \approx 10^{-1}$. A suitable phase to conduct the comparison is at $t = 1$s during the Head-on scenario with lateral separation $L_a = 152.4$ and longitudinal separation $L_o = 2000$m where a conflict is inevitable. At this phase $p_0 \leq P_{c_1} < 1$ and both methods estimate a similar probability of conflict. This is as expected since the probability is large enough to generate sufficient conflicting samples in the first level of Subset Simulation and it does not progress to higher levels of Subset Simulation. The first level of Subset Simulation is Direct Monte Carlo so the performance is the same.

The second order of magnitude considered is P_{c_2}. This probability needs to be lower than P_{c_1} where $P_{c_2} < p_0$. Such phases occur frequently in the Head-on pass and Overtaking scenarios, typically when $t > 14$s as shown in figures 7.5 and 7.7 respectively. Note, during such phases the Subset Simulation method is able to obtain conflicting samples and provide a good estimate for P_c. However, the Direct Monte Carlo method fails to find conflicting samples and is unable to estimate the probability of conflict accurately (other than in a trivial case, $P_c = 0$ that is inaccurate). For example the Head-on pass scenarios in Figure 7.5 shows abrupt changes in P_c in some cases from a magnitude of 10^{-1} to 10^{-8} at approximately 13s as the Observer and Intruder pass each other. This change in magnitude of probability is very large and abrupt (steep). The magnitude 10^{-8} is very rare. For such probabilities the Subset Simulation method is able to obtain conflicting samples and estimate the P_c but Direct Monte Carlo method fails to obtain conflicting samples and results in estimating $P_c = 0$. The Direct Monte Carlo method requires a large number of samples to estimate probabilities of such magnitude (10^{-8}). This might not be practical due to limited simulation resources. Therefore, this order of magnitude of probability is impractical for comparison since although the Subset Simulation method is able to find conflicting samples and estimate the P_c, the Direct Monte Carlo method is unable to find conflicting samples and fails

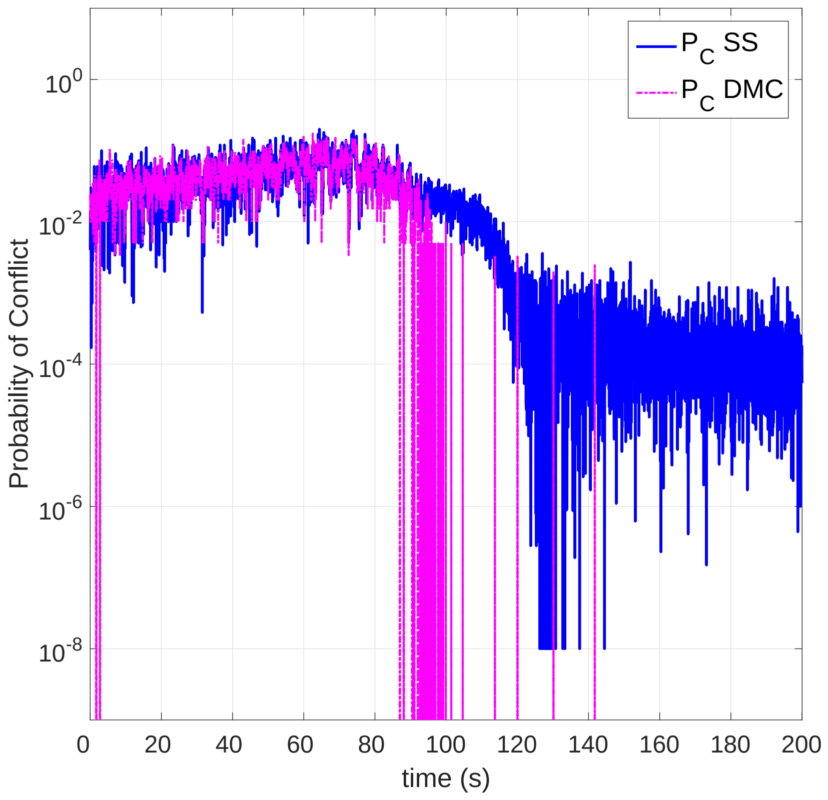
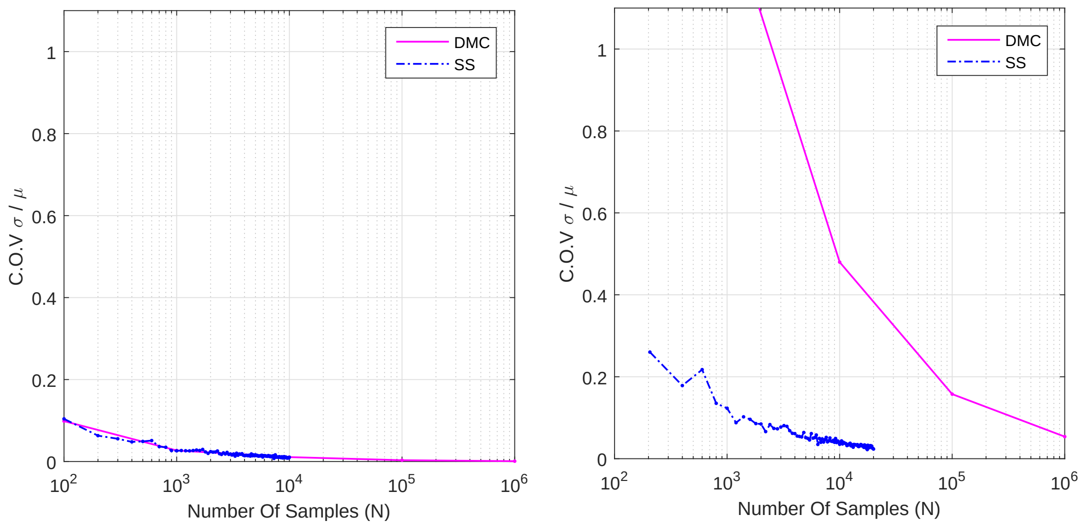


Figure 7.8: Head-on pass scenario with 1000m lateral separation and 20km longitudinal separation



(a) Coefficient of variance for varying number of samples using SS and DMC methods for estimating P_{c_1}

(b) Coefficient of variance for varying number of samples using SS and DMC methods for estimating P_{c_2}

Figure 7.9: A comparison of accuracy and efficiency between DMC and SS methods for estimating the P_c during the Head-on pass scenario.

to estimate the P_c.

In order to find a phase where P_{c_2} can be evaluated by both methods the simulation of the Head-on pass scenario with lateral separation of 1000m was repeated once with increased longitudinal separation $L_o = 20000$m for an increased period of $t = 200$s. This allowed the change in P_c to occur less abruptly. Figure 7.8 shows P_c estimated by Subset Simulation and Direct Monte Carlo methods during this scenario. Note, during the period $80\text{s} < t < 120\text{s}$, there are frequent abrupt variations in the P_c estimated by the Direct Monte Carlo method as zero. These are phases where the method was unable to find a conflicting sample and estimated the probability of conflict as zero. A suitable phase for P_{c_2} is at $t = 100$s where the probability of conflict estimated by Subset Simulation has reduced to approximately 10^{-2}; ($P_{c_2} \approx 10^{-2}$). This satisfies the $p_0 > P_{c_2}$ criteria. Also, it is the last phase after which the frequency of the Direct Monte Carlo method finding conflicting samples to estimate the P_c diminishes. In other words, it is the last phase where both methods are able to generate conflicting samples to estimate the probability of conflict for a comparison to be conducted.

The accuracy and efficiency are compared by calculating the coefficient of variance (c.o.v.) $\delta = \frac{\sigma}{\mu}$ for estimating the probabilities of conflict P_{c_1} and P_{c_2} using both Subset Simulation and Direct Monte Carlo methods for varying samples sizes N. The mean μ and standard deviation σ is calculated over 50 Monte Carlo runs. The sample intervals for Direct Monte Carlo are $N_{\text{dmc}} = \{10^2, 10^3, 10^4, 10^5, 10^6\}$ and the sample intervals for Subset Simulation are $N_{\text{SS}} = \{100n : n = 1, ..., 100\}$. Note, that N_{SS} is the number of samples at each level of Subset Simulation. The total number of levels can vary for each Monte Carlo run of Subset Simulation. This causes a total number of samples to vary for each Monte Carlo run. To allow a fair comparison an average of the total number of samples for each Monte Carlo run of Subset Simulation is used.

The c.o.v. for estimating P_{c_1} using Subset Simulation and Direct Monte Carlo methods at varying sample sizes N is shown by Figure 7.9(a). Note, both methods have similar c.o.v. as the average sample size increases. This is expected since the probability is large enough to be realised in level 0 of Subset Simulation that is Direct Monte Carlo. In Figure 7.9(b) the c.o.v. of Subset Simulation for the lower probability of conflict P_{c_2} becomes significantly lower than the c.o.v. of DMC as the average number of samples is increased. A point of comparison between both methods can be made where the number of samples $N = 10^4$. Note that the c.o.v for Direct Monte Carlo is approximately 0.48 and the c.o.v for Subset Simulation is approximately 0.04. Also note that in order for the DMC method to achieve similar c.o.v as the Subset Simulation method it must use $N = 10^6$ samples. Therefore the Subset Simulation estimates probabilities of magnitude 10^{-2} approximately ten times more accurately than the Direct Monte Carlo method while using a fraction of the samples (approximately $\frac{1}{100}$) that are required by the Direct Monte Carlo method to achieve similar levels of accuracy.

7.5 Summary

This chapter has demonstrated the utility of the Subset Simulation method to estimate the probability of conflict (P_c) between air traffic within a block of airspace during conflicting and potentially conflicting scenarios based on the Rules of the Air defined by the International Civil Aviation Organization. These scenarios can be used to conduct benchmarks for comparing future algorithms. The Subset Simulation method has demonstrated the ability to seek samples from the rare conflict region of interest in an effort to estimate the probability of conflict with lower computational effort than Direct Monte Carlo method. For the equivalent number of samples, the Direct Monte Carlo method fails to consistently obtain samples from the region of interest within the probability distribution function. The ability of Subset Simulation to estimate low probability of conflict (of magnitude 10^{-2}) approximately ten times more accurately than the Direct Monte Carlo method while using approximately $\frac{1}{100}$ of the total samples used by the Direct Monte Carlo method to achieve the same level of accuracy as Subset Simulation. This has been demonstrated at a phase during a potentially conflicting scenario based on the Rules of the Air. This example situation has demonstrated that the Subset Simulation method is able to estimate low probabilities more accurately than Direct Monte Carlo method while using less samples than the Direct Monte Carlo method. The Subset Simulation method is more accurate and efficient than the Direct Monte Carlo method for estimating low probability of conflict between air traffic.

Chapter 8

Conclusion and Future Work

This thesis has examined the problem of large computational requirements for safety related algorithms on-board Unmanned Aircraft Systems. Multiple sensor fault detection and Sense-and-Avoid are such essential safety requirements that generate large computational load that have been explored in this thesis.

For multiple sensor fault detection, a Generalized Pseudo Bayes-1 (GPB-1) method, consisting of Unscented Kalman Filters, that represent normal and faulty hypotheses of aircraft sensors, are used to detect faulty sensors and update the aircraft's state-estimate. The hypotheses for each sensor are evaluated sequentially using measurements obtained from the aircraft's sensors. This method is referred to as the sequential UKF-GPB-1 and has been tested successfully by simulation. A Six-degrees-of-Freedom flight model for a Navion aircraft with multiple sensors (three Pitot-static, two Angle-of-Attack and an Inertial Measurement Unit) and an autopilot has been developed to simulate common flight scenarios. This is a standard flight model that can be used for benchmarking purposes to compare future algorithms. The fault detection method has been tested by simulating multiple sensor faults during common flight scenarios such as level, climbing, descending and turning flight. The fault detection method successfully detected multiple sensor faults during the different flight scenarios. The sequential UKF-GPB-1 is compared against the parallel UKF-GPB-1 where each hypotheses are a combination of sensor failures. The sequential UKF-GPB-1 updates the aircraft state-estimate faster and maintains a more accurate state-estimate of the aircraft than the parallel UKF-GPB-1.

For Sense-And-Avoid, the Subset Simulation method has been used to estimate the probability of conflict, P_c, between air traffic in non-cooperative scenarios. This method has been tested by simulating a series of conflicting and potentially conflicting scenarios based on the Rules of the Air defined by the International Civil Aviation Organization. These scenarios can be used to conduct benchmarks for comparing future algorithms. For scenarios with low probabilities of conflict (of magnitude 10^{-2}) the Subset Simulation method estimates the P_c to be approximately ten times more accurate than the Direct Monte Carlo method. The Subset Simulation method achieves this while using approximately $\frac{1}{100}$ of the total samples used by the Direct Monte Carlo method to achieve the same level of accuracy as Subset Simulation.

The efficient and accurate methods for multiple sensor fault detection and estimating low probabilities of conflict can benefit Sensor Fault Detection and Sense-And-Avoid systems on-board the UAS respectively. The computational capacity required for such systems on-board the UAS can be reduced using the algorithms proposed in this thesis – leading to reduced Size, Weight, Power – and reduced Cost (SWAP-C) of Unmanned Aircraft Systems.

8.1 Future Work

The Subset Simulation method is scalable to involve multiple Intruders where the P_c is estimated for each intruder. This is useful for the resolution stage, where Intruders can be prioritised based on the respective P_c and an optimised resolution to manoeuvre is determined to minimise the new P_c after the resolution manoeuvre. A more efficient method of estimating the P_c would be to modify the Subset Simulation method to use Sequential Monte Carlo Samplers instead of the Markov Chain Monte Carlo method [130]. This will allow the implementation to be parallelised in the seed selection stage and will give rise to improved statistical efficiency.

Appendix A

Pseudo Code

A.1 Navion

Algorithm 1 Level Flight Condition

```
 1: function NavionLevelFlight($\mathbf{X}$,$V_c$,$h_c$)
 2:     $\rho = \rho_0 \left(1 - \frac{lh_c}{T_0}\right)^{\left(\frac{g}{lR} - 1\right)}$ ▷ Compute Air density
 3:     $Q = \frac{1}{2}\rho V_c^2$
 4:     $C_L = \frac{mg}{QS}$
 5:     $\delta_e = \frac{-C_{m_0}C_{L_\alpha} - C_{m_\alpha}C_L + C_{m_\alpha}C_{L_0}}{C_{m_{\delta_e}}C_{L_\alpha} - C_{m_\alpha}C_{L_{\delta_e}}}$
 6:     $\alpha = \frac{C_L}{C_{L_\alpha}} - \frac{C_{L_0}}{C_{L_\alpha}} - \frac{C_{L_{\delta_e}}\delta_e}{C_{L_\alpha}}$
 7:     $C_D = C_{D_0} + C_k C_L^2$
 8:     $T = QSC_D \cos\alpha$
 9:     return T, $\delta_e$
10: end function
```

Algorithm 2 Navion Aircraft simulation

```
 1: function NavionDynamics($\Delta t$, $\mathbf{K}$, $\mathbf{I}_{(x,y,z)}$, $X_k$, $\eta_k$)
    ▷Extract current state elements and constants
 2:     $[I_x, I_y, I_z] = \mathbf{I}_{(x,y,z)}$
 3:     $[b, c, S, C_{L_0}, C_{L_\alpha}, C_{L_\beta}, C_{L_q}, C_{L_{\delta_e}}, C_{Y_\alpha}, C_{Y_\beta}, C_{Y_p}, C_{Y_r} C_{Y_{\delta_a}}, C_{Y_{\delta_e}}, C_{Y_{\delta_r}}, C_{D_0}, C_k,$
        $C_{l_\alpha}, C_{l_\beta}, C_{l_p}, C_{l_r}, C_{l_r}, C_{l_{\delta_a}}, C_{l_{\delta_e}}, C_{l_{\delta_r}}, C_{m_0}, C_{m_\alpha}, C_{m_\beta}, C_{m_{\dot\alpha}}, C_{m_q}, C_{m_{\delta_e}}$
        $C_{n_\alpha}, C_{n_\beta}, C_{n_p}, C_{l_q}q, C_{l_{\delta_a}}, C_{l_{\delta_e}}, C_{l_{\delta_r}}] = \mathbf{K}$
 4:     $[x, u, a_x, z, w, a_z, \theta, q, \dot{q}, y, v, a_y, \psi, p, \dot{p}, \phi, r, \dot{r}]^{\mathrm{T}} = X_k$ ▷ Current state elements
 5:     $[\delta_a, \delta_e, \delta_r, T]^{\mathrm{T}} = \eta_k$ ▷ Current control elements
 6:     $V_a = \sqrt{u^2 + v^2 + w^2}$ ▷ Compute aircraft speed
 7:     $\alpha = \tan^{-1}(\frac{w}{u})$ ▷ compute angle-of-attack
 8:     $\beta = \sin^{-1}(\frac{v}{V_a})$ ▷ compute sideslip angle
 9:     $\rho = \rho_0 \left(1 - \frac{lz}{T_0}\right)^{\left(\frac{g}{lR} - 1\right)}$ ▷ Compute Air density
```

```
10:     $Q = \frac{1}{2}\rho V_a^2$ ▷ *Compute Dynamic pressure*
11:     $\dot{\alpha} = \frac{u\dot{w} - w\dot{u}}{u^2 + w^2}$
        ▷*Compute Force coefficients*
12:     $C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_\beta}\beta + \frac{c}{2V_a}C_{L_q}q + C_{L_{\delta_e}}\delta_e$
13:     $C_Y = C_{Y_\alpha}\alpha + C_{Y_\beta}\beta + \frac{b}{2V_a}(C_{Y_p}p + C_{Y_r}r) + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_e}}\delta_e + C_{Y_{\delta_r}}\delta_r$
14:     $C_D = C_{D_0} + C_k C_L^2$
        ▷*Compute Accelerations*
15:     $\dot{u} = -Q\left(\frac{S}{m}\right)(C_D\cos\alpha\cos\beta + C_Y\cos\alpha\sin\beta - C_L\sin\alpha) + \frac{T}{m} - g\sin\theta - qw + rv$
16:     $\dot{v} = -Q\left(\frac{S}{m}\right)(C_D\sin\beta - C_Y\cos\beta) + g\sin\phi\cos\theta - ru + pw$
17:     $\dot{w} = -Q\left(\frac{S}{m}\right)(C_D\sin\alpha\cos\beta + C_Y\sin\alpha\sin\beta + C_L\cos\alpha) + g\cos\theta\cos\phi - pv + qu$
        ▷*Compute Moment coefficients*
18:     $C_l = C_{l_\alpha}\alpha + C_{l_\beta}\beta + \frac{b}{2V_a}(C_{l_p}p + C_{l_r}r) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_e}}\delta_e + C_{l_{\delta_r}}\delta_r$
19:     $C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_\beta}\beta + \frac{c}{2V_a}(C_{m_{\dot{\alpha}}}\dot{\alpha} + C_{m_q}q) + C_{m_{\delta_e}}\delta_e$
20:     $C_n = C_{n_\alpha}\alpha + C_{n_\beta}\beta + \frac{b}{2V_a}(C_{n_p}p + C_{l_q}q) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_e}}\delta_e + C_{l_{\delta_r}}\delta_r$
        ▷*Compute angular accelerations*
21:     $\dot{p} = -\frac{1}{I_x}[qr(I_z - I_y) - QSbC_l]$
22:     $\dot{q} = -\frac{1}{I_y}[rp(I_x - I_z) - QScC_m]$
23:     $\dot{r} = -\frac{1}{I_z}[pq(I_y - I_x) - QSbC_n]$
24:     **return** $\dot{u}, \dot{v}, \dot{w}, \dot{p}, \dot{q}, \dot{r}$
25: **end function**
```

Algorithm 3 The SixDoF function

```
1: **function** NavionSixDoF($\Delta t$, $\mathbf{K}$, $\mathbf{I}_{(x,y,z)}$, $\mathbf{X}_k$, $\eta_k$)
2:     $[\dot{u}_{k+1}, \dot{v}_{k+1}, \dot{w}_{k+1}, \dot{p}_{k+1}, \dot{q}_{k+1}, \dot{r}_{k+1}]^{\mathrm{T}}$ = NavionDynamics($\Delta t$, $\mathbf{K}$, $\mathbf{I}_{(x,y,z)}$, $\mathbf{X}_k$, $\eta_k$)
3:     $[\Delta u, \Delta v, \Delta w]^{\mathrm{T}} = [\dot{u}_{k+1}, \dot{v}_{k+1}, \dot{w}_{k+1}]^{\mathrm{T}}\Delta t$ ▷ *Integrate acceleration to determine change in speed*
4:     $[u_{k+1}, v_{k+1}, w_{k+1}]^{\mathrm{T}} = [u_k, v_k, w_k]^{\mathrm{T}} + [\Delta u, \Delta v, \Delta w]^{\mathrm{T}}$
       ▷*Rotate to earth oriented axis*
5:     $\begin{bmatrix}\dot{x}_{k+1}\\ \dot{y}_{k+1}\\ \dot{z}_{k+1}\end{bmatrix} = \begin{bmatrix}\cos\psi_k & -\sin\psi_k & 0\\ \sin\psi_k & \cos\psi_k & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta_k & 0 & \sin\theta_k\\ 0 & 1 & 0\\ -\sin\theta_k & 0 & \cos\theta_k\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi_k & -\sin\phi_k\\ 0 & \sin\phi_k & \cos\phi_k\end{bmatrix}\begin{bmatrix}u_{k+1}\\ v_{k+1}\\ w_{k+1}\end{bmatrix}$
6:     $[\Delta x, \Delta y, \Delta z]^{\mathrm{T}} = [\dot{x}_{k+1}, \dot{y}_{k+1}, \dot{z}_{k+1}]^{\mathrm{T}}\Delta t$
7:     $[x_{k+1}, y_{k+1}, z_{k+1}]^{\mathrm{T}} = [x_k, y_k, z_k]^{\mathrm{T}} + [\Delta x, \Delta y, \Delta z]^{\mathrm{T}}$
8:     $\begin{bmatrix}\dot{\psi}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}1 & \sin\phi_k\tan\theta_k & \cos\phi_k\tan\theta_k\\ 0 & \cos\phi_k & -\sin\phi_k\\ 0 & \sin\phi_k\sec\theta_k & \cos\phi_k\sec\theta_k\end{bmatrix}\begin{bmatrix}p_{k+1}\\ q_{k+1}\\ r_{k+1}\end{bmatrix}$
9:     $[\Delta\psi, \Delta\theta, \Delta\phi]^{\mathrm{T}} = [\dot{\psi}, \dot{\theta}, \dot{\phi}]^{\mathrm{T}}\Delta t$
```

```
10:     $[\psi_{k+1}, \theta_{k+1}, \phi_{k+1}]^{\mathrm{T}} = [\psi_k, \theta_k, \phi_k]^{\mathrm{T}} + [\Delta\psi, \Delta\theta, \Delta\phi]^{\mathrm{T}}$ ▷ *Determine aircraft attitude*
11:     $[\Delta p, \Delta q, \Delta r] = [\dot{p}, \dot{q}, \dot{r}]^{\mathrm{T}} \Delta t$ ▷ *Integrate angular acceleration to determine change in angular speed*
12:     $[p_{k+1}, q_{k+1}, r_{k+1}]^{\mathrm{T}} = [p_k, q_k, r_k]^{\mathrm{T}} + [\Delta p, \Delta q, \Delta r]^{\mathrm{T}}$ ▷ *Aircraft angular rates*
13:     $\mathbf{X}_{k+1} = [x_{k+1}, u_{k+1}, \dot{u}_{k+1}, z_{k+1}, w_{k+1}, \dot{w}_{k+1}, \theta_{k+1}, q_{k+1}, \dot{q}_{k+1},$
        $y_{k+1}, v_{k+1}, \dot{v}_{k+1}, \psi_{k+1}, p_{k+1}, \dot{p}_{k+1}, \phi_{k+1}, r_{k+1}, \dot{r}_{k+1}]^{\mathrm{T}}$
14:     return $\mathbf{X}_{k+1}$
15: **end function**
```

Algorithm 4 Aircraft Autopilot

```
1:  **function** AUTOPILOT($\mathbf{X}$, $\psi_C$, $z_C$)
2:      $[x, u, a_x, z, w, a_z, \theta, q, \dot{q}, y, v, a_y, \psi, p, \dot{p}, \phi, r, \dot{r}]^{\mathrm{T}} = X_k$ ▷ *Current state elements*
3:      $\Delta z = z - z_c$
4:      $\alpha_\epsilon = \frac{1}{K_1} \Delta z$
5:      $\delta_{e_{k+1}} = \delta_{e_k} + \mathrm{sign}(\alpha_\epsilon) \frac{\min(K_{\delta_e} |\alpha_\epsilon|, q_{\max})}{q_{\delta_e}}$
6:      $\mathbf{V}_{\mathrm{body}} = [u, v, w]^{\mathrm{T}}$
7:      $\hat{\mathbf{V}}_{\mathrm{earth}} = T_B^e(\psi, \theta, \phi) \mathbf{V}_{\mathrm{body}}$
8:      $\mathbf{u}_c = T_B^e(\psi_c, 0, 0) \mathbf{u}$
9:      $\phi_d = \mathrm{sign}(\Delta\psi) \min(K_\phi |\Delta\psi|, \phi_{\max})$
10:     $\delta_{a_{k+1}} = \delta_{a_k} + \mathrm{sign}(\Delta\phi) \frac{\min(K_{\delta_a} |\phi_d|, p_{\max})}{p_{\delta_a}}$
11:     $\delta_{r_{k+1}} = \delta_{r_k} + \mathrm{sign}(\beta_k) \frac{\min(K_{\delta_r} |\beta_k|, r_{\max})}{r_{\delta_r}}$
12:     **return** $[\delta_{a_{k+1}}, \delta_{e_{k+1}}, \delta_{r_{k+1}}]$
13: **end function**
```

A.2 Direct Monte Carlo

Algorithm 5 Determine distance between samples X and C

```
1:  **function** H($X$,$C$)
2:      $V = X - C$
3:      $R = \sqrt{V_x^2 + V_y^2}$
4:      **return** $R$
5:  **end function**
```

Algorithm 6 Direct Monte Carlo

```
1:  **function** DMC($N$, $C$, $r_c$)
2:      $D = 0$
3:      **for** $n = 1 : N$ **do**
4:          $x \sim \mathcal{N}(0, 1)$
5:          $y \sim \mathcal{N}(0, 1)$
6:          $X_n = [x, y]^T$
7:          $R_n = \mathrm{H}(X_n, C)$
```

```
 8:         if $R_n \leq r_c$ then
 9:             $D = D + 1$
10:         end if
11:     end for
12:     $P_F = \frac{D}{N}$
13:     return $P_F$
14: end function
```

A.3 Metropolis Hastings

Algorithm 7 Generate conditional chains of samples using Metropolis Hastings algorithm

```
 1: function MH($s$, $n$, $C$, $r_c$)
 2:     $\sigma^2_{r_c} = r_c^2 I_{2\times 2}$
 3:     for $j = 1 : |s|$ do ▷ For each seed
 4:         $X_0 = s_j$ ▷Select seed sample
 5:         for $k = 0 : n - 1$ do
            ▷Generate Candidate sample $X^*$
 6:             $g \sim \mathcal{N}(0, 1)$
 7:             $X^* = X_k + g$
                ▷Calculate acceptance ratio
 8:             $\beta = \frac{q(X^*|X_k,\sigma^2)}{q(X_k|X^*,\sigma^2)} \frac{p(X^*|C,\sigma^2_{r_c})}{p(X_k|C,\sigma^2_{r_c})}$
 9:             $\alpha = min\{1, \beta\}$
10:             $e \sim [0, 1]$
11:             $X^{(j)}_{k+1} = \begin{cases} X^* & \text{if } e < \alpha \\ X_k & \text{if } e \geq \alpha \end{cases}$
12:         end for
13:     end for
14:     return $X^{(j)}$
15: end function
```

A.4 Subset Simulation

Algorithm 8 Generate conditional chains of samples of Subset Simulation using Metropolis Hastings algorithm

```
 1: function MH_I($s$, $n$, $C$, $r_c$)
 2:     $\sigma^2_{r_c} = r_c^2 I_{2\times 2}$
 3:     for $j = 1 : |s|$ do ▷ For each seed
 4:         $X_0 = s_j$ ▷Select seed sample
 5:         for $k = 0 : n - 1$ do
            ▷Generate Candidate sample $X^*$
```

```
6:            $g \sim \mathcal{N}(0,1)$
7:            $X^* = X_k + g$
              $\triangleright$ *Determine distance between $X^*$ and $C$*
8:            $R^* = \mathrm{H}(X^*, C)$
              $\triangleright$ *Determine distance between $X_k$ and $C$*
9:            $R_k = \mathrm{H}(X_k, C)$
              $\triangleright$ *Indicator function for range*
10:           $d = \begin{cases} 1 & \text{if } R^* \leq r_c \\ 0 & \text{if } R^* > r_c \end{cases}$
              $\triangleright$ *Calculate acceptance ratio*
11:           $\beta = \frac{q(X^*|X_k,\sigma^2)}{q(X_k|X^*,\sigma^2)} \frac{p(X^*|C,\sigma^2_{r_c})}{p(X_k|C,\sigma^2_{r_c})}$
12:           $\alpha = min\{1, \beta\}$
13:           $e \sim [0,1]$
14:           $X^{(j)}_{k+1} = \begin{cases} X^* & \text{if } e < \alpha \\ X_k & \text{if } e \geq \alpha \end{cases}$
15:           $R^{(j)}_{k+1} = \begin{cases} R^* & \text{if } e < \alpha \\ R_k & \text{if } e \geq \alpha \end{cases}$
16:       end for
17:   end for
18:   return $X^{(j)}, R^{(j)}$
19: end function
```

Algorithm 9 Subset Simulation

```
1: function SS($C$, $N$, $p_0$, $m$)
2:     $N_c = p_0 N$
3:     $N_s = p_0^{-1}$
4:     $i = 0$  *Set current level*
       $\triangleright$ *Direct Monte Carlo: Draw N samples and determine quantity of interest*
5:     for $n = 1 : N$ do
6:         $X^{(i)}_n \sim \mathcal{N}(0,1)$
           $\triangleright$ *Quantity of interest: Determine distance between samples $X^{(i)}_n$ and $C$*
7:         $R^{(i)}_n = \mathrm{H}(X^{(i)}_n, C)$
8:     end for
9:     $B^{(i)}_n \leftarrow R^{(i)}_n$ *Sort distances in descending order*
10:    $\tilde{X}^{(i)}_n \leftarrow X^{(i)}_n$ *Reorder the input samples to correspond to the sorted quantity of interest $B^{(i)}_n$*
       $\triangleright$ *Generate probability intervals; equation 6.2*
11:    for $n = 1 : N$ do
```

```
12:         $P_n^{(i)} = p_0^i \frac{N-n}{N}$
13:     end for
        ▷CCDF: Concatenate vectors $P_n^{(i)}$, $B_n^{(i)}$ and sample $\tilde{X}_n^{(i)}$
14:     $E_n = [P_n^{(i)}, B_n^{(i)}, \tilde{X}_n^{(i)}]$
        ▷Begin lower levels of subset simulation
15:     for $i = 1 : m - 1$ do
            ▷Set threshold
16:         $b_i = B_{N-N_c}^{(i-1)}$
            ▷Set seeds using equation 6.6
17:         for $j = 1 : N_c$ do
18:             $n = N - N_c + j$
19:             $s_j^{(i)} = \tilde{X}_n^{(i-1)}$
20:         end for
            ▷Generate conditional samples using Metropolis Hastings algorithm
21:         $[X_n^{(i)}, R_n^{(i)}] = \mathrm{MH\_I}(s_j^{(i)},\ N_s,\ C,\ b_i)$
22:         $B_n^{(i)} \leftarrow R_n^{(i)}$ Sort distances in descending order
23:         $\tilde{X}_n^{(i)} \leftarrow X_n^{(i)}$ Reorder the input samples to correspond to the sorted quantity
    of interest $B_n^{(i)}$
            ▷Generate probability intervals; equation 6.2
24:         for $n = 1 : N$ do
25:             $P_n^{(i)} = p_0^i \frac{N-n}{N}$
26:         end for
            ▷CCDF: Discard all rows after $E_{i(N-N_c)}$
            ▷Concatenate $P_n^{(i)}$, $B_n^{(i)}$, $\tilde{X}_n^{(i)}$ and append to $E$
27:         for $n = 1 : N$ do
28:             $E_{i(N-N_c+n)} = [P_n^{(i)}, B_n^{(i)}, \tilde{X}_n^{(i)}]$
29:         end for
30:     end for
31:     return $E$
32: end function
```

A.5 Estimating Probability of Conflict between Air traffic

Algorithm 10 Propagate State to generate trajectory

```
1: function SampleTrajectory($U_0,\ f, t, A$)
2:     $J_0 = U_0$
3:     for $k = 0 : tf$ do
4:         $U(k+1) = AU(k)$
5:         $J(k+1) = U(k+1)$
```

```
6:     end for
7:     return $J$
8: end function
```

Algorithm 11 Determine miss-distance r and minimum points $\hat{U}_{xy}, O_{xy}$ between observer trajectory J_O and Intruder trajectory $J_{\hat{U}}$

```
1: function MinDistance($J_O, J_U$)
   ▷Difference between Observer and Intruder trajectory
2:     $J_{OU} = J_U - J_O$
       ▷Distance between each point on trajectories
3:     $r_{OU} = \sqrt{J_{OU_x}^2 + J_{OU_y}^2}$
       ▷Minimum distance
4:     $r_{OU_{\min}} = \min(r_{OU})$
       ▷Index of minimum distance
5:     $k = \{r_{OU_n} | n = r_{OU_{\min}}\}$
6:     $J_{O_{\min}} = J_{O_{xy}}(k)$
7:     $J_{U_{\min}} = J_{U_{xy}}(k)$
8:     return $r_{O\hat{U}_{\min}}, J_{O_{\min}}, J_{\hat{U}_{\min}}$
9: end function
```

Algorithm 12 Estimating Probability of Conflict using Direct Monte Carlo

```
1: function PC_DMC($f, t, A, O, \hat{U}, \hat{S}, N, r_t$)
2:     $D = 0$
       ▷Propagate Observer for t seconds
3:     $J_O$ = SampleTrajectory($O,\ f,\ t,\ A$)
4:     for $n = 1 : N$ do
       ▷Draw sample
5:         $U_n \sim \mathcal{N}(\hat{U}, \hat{S})$
           ▷Propagate Intruder Samples for t seconds
6:         $J_n$ = SampleTrajectory($U_n,\ f, t, A$)
           ▷Determine miss-distance between Observer and Sample Trajectories
7:         $r_n$ = MinDistance($J_O, J_n$)
8:         if $r_n \leq r_s$ then
9:             $D = D + 1$
10:        end if
11:    end for
12:    $P_c = \frac{D}{N}$
13:    return $P_c, D, U_n, J_O, J_n, r_n$
14: end function
```

Algorithm 13 Kalman Filter

```
1: function KF($\hat{U}(K)$, $\hat{S}(K)$, $Z$, $H$, $Q$, $R$, $M_Z$)
   ▷Predict
2:     $\hat{U}(K+1) = A\hat{U}(K)$
3:     $\hat{S}(K+1) = A\hat{S}(K)A^T + Q$
       ▷Update if new measurement is available
4:     if $M_Z$ = true then
5:         $G = \hat{S}(K+1)H^T\{[H\hat{S}(K+1)H^T] + R\}^{-1}$
6:         $\hat{U}(K+1) = \hat{U}(K+1) + G\{Z - [H\hat{U}(K+1)]\}$
7:         $\hat{S}(K+1) = [I - GH]\hat{S}(K+1)$
8:     end if
9:     return $\hat{U}(K+1), \hat{S}(K+1)$
10: end function
```

Algorithm 14 Generate conditional samples using Metropolis Hastings

```
1: function MH_ConflictSamples($f$, $t$, $A$, $O$, $\hat{U}$, $\hat{S}$, $s_j$, $N_s$, $r_t$)
2:     $\sigma^2_{r_t} = r_t^2 I_{2\times 2}$
3:     $J_O$ = SampleTrajectory($O$, $f, t, A$)
4:     for $j = 1 : N_c$ do
5:         $U_0 = s_j$ ▷Select seed sample
           ▷For each seed generate $N_s$ samples
6:         for $k = 0 : N_s - 1$ do
               ▷Draw acceleration sample from mean
7:             $a^*_x \sim \mathcal{N}(0,1)$
8:             $a^*_y \sim \mathcal{N}(0,1)$
9:             $g = [0, 0, a^*_x, 0, 0, a^*_y]^T$
               ▷Generate Candidate sample $U^*$
10:            $U^* = U_k + g$
               ▷Propagate Samples for $t$ seconds
11:            $J^*_U$ = SampleTrajectory($U^*$, $f, t, A$)
12:            $J_{U_k}$ = SampleTrajectory($U_k$, $f, t, A$)
               ▷Determine minimum miss-distance and $(x, y)$ coordinates of minimum
               points between Observer and Sample Trajectories
13:            $[r_k, J_{O_{\min}}, J_{U_{k_{\min}}}]$ = MinDistance($J_O, J_{U_k}$)
14:            $[r^*, J^*_{O_{\min}}, J^*_{U_{\min}}]$ = MinDistance($J_O, J^*_U$)
               ▷Indicator function for miss-distance
15:            $d = \begin{cases} 1 & \text{if } r^* < r_t \\ 0 & \text{if } r^* \geq r_t \end{cases}$
```

```
                ▷$Calculate\ acceptance\ ratio$
16:             $\beta = \frac{p(J^*_{\hat{U}_{\min}} | J^*_{O_{\min}}, \sigma^2_{r_t}) q(U^* | \hat{U}, \hat{S})}{p(J_{U_{k_{\min}}} | J_{O_{\min}}, \sigma^2_{r_t}) q(U_k | \hat{U}, \hat{S})} d$
17:             $\alpha = \min\{1, \beta\}$
18:             $e \sim\ [0,1]$
                ▷$Accept\ candidate\ sample,\ trajectory\ and\ miss\text{-}distance\ if\ e < a$
19:             $U^{(j)}_{k+1} = \begin{cases} U^* & \text{if } e < \alpha \\ U_k & \text{if } e \geq \alpha \end{cases}$
20:             $J^{(j)}_{k+1} = \begin{cases} J^* & \text{if } e < \alpha \\ J_k & \text{if } e \geq \alpha \end{cases}$
21:             $r^{(j)}_{k+1} = \begin{cases} r^* & \text{if } e < \alpha \\ r_k & \text{if } e \geq \alpha \end{cases}$
22:         end for
23:     end for
24:     return $U^{(j)}$, $J^{(j)}$, $r^{(j)}$
25: end function
```

Algorithm 15 Estimate Probability of Conflict Using Subset Simulation

```
 1: function PC_SS($f$, $t$, $A$, $O$, $\hat{U}$, $\hat{S}$, $N$, $r_t$, $p_0$, $m$)
 2:     $N_c = p_0 N$
 3:     $N_s = p_0^{-1}$
 4:     $i = 0$ ▷$Set\ current\ level$
        ▷$Direct\ Monte\ Carlo$
 5:     $[D, U^{(i)}_n, r^{(i)}_n] = \text{PC\_DMC}(f, t, A, O, \hat{U}, \hat{S}, N, r_t)$
 6:     $B^{(i)}_n \leftarrow r^{(i)}_n$ Sort distances in descending order
 7:     $\tilde{U}^{(i)}_n \leftarrow U^{(i)}_n$ Reorder the input samples to correspond to the sorted quantity of
    interest $B^{(i)}_n$
        ▷Generate probability intervals; equation 7.14
 8:     for $n = 0 : N - 1$ do
 9:         $P^{(i)}_{n+1} = p_0^i \frac{N-n}{N}$
10:     end for
        ▷CCDF: Concatenate vectors $P^{(i)}_n$, $B^{(i)}_n$ and samples $\tilde{U}^{(i)}_n$
11:     $E_n = [P^{(i)}_n, B^{(i)}_n, \tilde{U}^{(i)}_n]$
12:     while $D < N_c$ and $i < m$ do
13:         $i = i + 1$
14:         $b_i = B^{(i-1)}_{N-N_c}$ ▷Set threshold
            ▷Set seeds using equation 6.6
15:         for $j = 1 : N_c$ do
```

```
16:             $n = N - N_c + j$
17:             $s_j^{(i)} = \tilde{U}_n^{(i-1)}$
18:         end for
            ▷Metropolis Hastings to obtain conflicting samples
19:         $[U_n^{(i)}, r_n^{(i)}] = $ MH_ConflictSamples($f$, $t$, $A$, $O$, $\hat{U}$, $\hat{S}$, $s_j$, $N_s$, $b_i$)
20:         $B_n^{(i)} \leftarrow r_n^{(i)}$ Sort distances in descending order
21:         $\tilde{U}_n^{(i)} \leftarrow U_n^{(i)}$ Reorder the input samples to correspond to the sorted quantity
    of interest $B_n^{(i)}$
            ▷Generate probability intervals; equation 7.14
22:         for $n = 0 : N - 1$ do
23:             $P_{n+1}^{(i)} = p_0^i \frac{N-n}{N}$
24:         end for
            ▷CCDF: Discard all rows after $E_{i(N-N_c)}$
            ▷Concatenate $P_n^{(i)}$, $B_n^{(i)}$, $\tilde{U}_n^{(i)}$ and append to $E$
25:         for $n = 1 : N$ do
26:             $E_{i(N-N_c+n)} = [P_n^{(i)}, B_n^{(i)}, \tilde{U}_n^{(i)}]$
27:         end for
28:         $D = |B_n^{(i)} \leq r_t|$ ▷Number of conflicts $D$
29:     end while
30:     if $D > 0$ then
31:         $P_c = P_{(N-D+1)}^{(i)}$
32:     else
33:         $P_c = P_N^{(i)}$ ▷No conflicting samples were found select lowest probability inter-
    val
34:     end if
35:     return $P_c$, $E$
36: end function
```

Algorithm 16 Determine Probability of Conflict using SS and DMC

```
1: $O(0)$ ▷Initialise Observer
2: $U(0)$ ▷Initialise Intruder
3: $\hat{U}(0)$ ▷Initialise Intruder Estimate
4: $\hat{S}(0)$ ▷Initialise Intruder Covariance
5: $M_c = 0$ ▷Measurement counter
6: for $K = 0 : tf$ do
7:     $O(K+1) = AO(K)$ ▷Propagate Observer
8:     $U(K+1) = AU(K)$ ▷Propagate Intruder
9:     $M_Z = false$ ▷Flag to indicate new measurement
10:    if $M_c = \frac{f}{f_M}$ then ▷Conduct Intruder position measurement
11:        $Z = HU(K+1) + [w_x, w_y]^T$
```

```
12:         $M_Z$ = true ▷*Set flag to indicate that new measurement is available for Kalman filter Update*
13:         $M_c = 0$ ▷ *Reset measurement counter*
14:     **end if**
15:     $M_c = M_c + 1$ ▷*Increment measurement counter*
        ▷*Predict/Update estimate of Intruder with Kalman filter*
16:     $[\hat{U}(K+1), \hat{S}(K+1)] = \mathrm{KF}(\hat{U}(K), \hat{S}(K), Z, H, Q, R, M_Z)$
        ▷*Estimate Probability of Conflict using Subset Simulation*
17:     $P_c^{(\mathrm{SS})}(K+1) = \mathrm{PC\_SS}(f, t, A, O, \hat{U}(K+1), \hat{S}(K+1), N, r_t, p_0, m)$
        ▷*Estimate Probability of Conflict using Direct Monte Carlo*
18:     $P_c^{(\mathrm{DMC})}(K+1) = \mathrm{PC\_DMC}(f, t, A, O, \hat{U}(K+1), \hat{S}(K+1), N, r_t)$
19: **end for**
```

Bibliography

[1] lbl-engineer.livejournal.com. Inertial Measurement Unit on-board French S3 missile. [Online]. Available: http://lbl-engineer.livejournal.com/ [Accessed: 2016, May]

[2] Aviation Safety Network. AoA sensor. [Online]. Available: http://news.aviation-safety.net/2014/12/10/ [Accessed: 2016, May]

[3] Static Pressure Ports. [Online]. Available: http://www.upcscavenger.com/pitot-static_system/#page=ps [Accessed: 2016, May]

[4] Bureau d'Enquétes et d'Analyses pour la sécurité de l'aviation civile, "Final report on the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro–Paris," *Paris: BEA*, 2012. [Online]. Available: https://www.bea.aero/docspa/2009/f-cp090601.en/pdf/f-cp090601.en.pdf [Accessed: June, 2015]

[5] S. Julier and J. Uhlmann, "Unscented Filtering and Non Linear Estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[6] J. M. Pratt, *Air Law Operational Procedures Communications*, Third Edition ed. AFE, 2009.

[7] R. C. Nelson, *Flight Stability and Automatic Control.* WCB/McGraw Hill, 1998, vol. 2.

[8] Civil Aviation Authority, "CAP 722 Unmanned Aircraft System Operations in UK Airspace Guidance," *Published by TSO (the Stationery Office) on behalf of the UK Civil Aviation Authority*, 2015.

[9] Ministry Of Defence, "Joint Doctrine Note 2 / 11 the UK Approach To Unmanned Aircraft," 2011.

[10] K. Dalamagkidis, K. Valavanis, and L. Piegl, "On unmanned aircraft systems issues, challenges and operational restrictions preventing integration into the National Airspace System," *Progress in Aerospace Sciences*, vol. 44, no. 7, pp. 503–519, 2008.

[11] M. T. Degarmo, "Issues Concerning Integration of Unmanned Aerial Vehicles in Civil Airspace," *MITRE*, no. 04-1232, 2004.

[12] K. Dalamagkidis, K. Valavanis, and L. Piegl, *On Integrating Unmanned Aircraft Systems into the National Airspace System.* Springer, 2012.

[13] A. Zeitlin, A. Lacher, J. Kuchar, and A. Drumm, "Collision Avoidance for Unmanned Aircraft: Proving the Safety Case," *The MITRE Corporation and MIT Lincoln Laboratory*, pp. 1–12, 2006.

[14] R. A. Clothier and R. A. Walker, "Safety Risk Management of Unmanned Aircraft Systems," in *Handbook of Unmanned Aerial Vehicles.* Springer, 2015, pp. 2229–2275.

[15] W. C. Barott, E. Coyle, T. Dabrowski, C. Hockley, and R. S. Stansbury, "Passive Multispectral Sensor Architecture for Radar-EOIR Sensor Fusion for Low SWAP UAS Sense and Avoid," *Record - IEEE PLANS, Position Location and Navigation Symposium*, vol. c, pp. 1188–1196, 2014.

[16] International Civil Aviation Organization, "Annex 2 to the Convention on International Civil Aviation - Rules of the Air," *Tenth Edition, International Civil Aviation Organization*, 2005.

[17] G. Limnaios, N. Tsourveloudis, and K. P. Valavanis, "Introduction, in Sense and Avoid in UAS: Research and Applications," *Sense and Avoid in UAS: Research and Applications*, vol. 61, p. 87, 2012.

[18] Association For Unmanned Vehicle Systems International. The Benefits of Unmanned Aircraft Systems. [Online]. Available: https://epic.org/events/UAS-Uses-Saving-Time-Saving-Money-Saving-Lives.pdf [Accessed: 2016, July]

[19] International Civil Aviation Organization, "Annex 15 to the Convention on International Civil Aviation - Aeronautical Information Service," *Fourteenth Edition, International Civil Aviation Organization*, 2013.

[20] Civil Aviation Authority, "CAP 393 Air Navigation: The Order and the Regulations," *Published by TSO (the Stationery Office) on behalf of the UK Civil Aviation Authority*, 2015.

[21] Paris Convention. Convention Relating to the Regulation of Aerial Navigation Signed at Paris, October. [Online]. Available: http://library.arcticportal.org/1580/1/1919_Paris_conevention.pdf [Accessed: 2016, July]

[22] Paris Convention. The 1919 Paris Convention- The starting point for the regulation of air navigation. [Online]. Available: http://www.icao.int/secretariat/PostalHistory/1919_the_paris_convention.htm [Accessed: 2016, July]

[23] IIBERO-American Convention. 1926: The Ibero-American Convention. [Online]. Available: http://www.icao.int/secretariat/PostalHistory/1926_the_iibero_american_convention.htm [Accessed: 2016, July]

[24] The Havana Convention. 1928: The Havana Convention. [Online]. Available: http://www.icao.int/secretariat/PostalHistory/1928_the_havana_convention.htm [Accessed: 2016, July]

[25] International Civil Aviation Authority. (1944) Chicago convention. [Online]. Available: http://www.icao.int/publications/pages/doc7300.aspx [Accessed: 2016, May]

[26] International Civil Aviation Organization, "The Convention on International Civil Aviation - Annexes 1 to 18," *Convention on International Civil Aviation*, pp. 29–31, 2012.

[27] International Civil Aviation Organization (ICAO), "ICAO Member States," 2013. [Online]. Available: http://www.icao.int/MemberStates/Member%20States.English.pdf [Accessed: 2016, July]

[28] X. Prats, L. Delgado, J. Ramírez, P. Royo, and E. Pastor, "Requirements, Issues, and Challenges for Sense and Avoid in Unmanned Aircraft Systems," *Journal of Aircraft*, vol. 49, no. 3, pp. 677–687, 2012.

[29] J. Verstraeten, M. Stuip, and T. van Birgelen, "Assessment of Detect and Avoid Solutions for Use of Unmanned Aircraft Systems in Nonsegregated Airspace," in *Handbook of Unmanned Aerial Vehicles.* Springer, 2015, pp. 1955–1979.

[30] Eurocontrol. (2006) Eurocontrol specifications for the use of military unmanned aerial vehicles as operational air traffic outside segregated airspace. [Online]. Available: https://www.eurocontrol.int/sites/default/files/content/documents/single-sky/cm/civil-mil-coordination/cmac-rpa-specifications-v-2-0-20120201.pdf [Accessed: 2016, July]

[31] Federal Aviation Administration. FAA Order 8700, Change 3. [Online]. Available: https://www.faa.gov/documentLibrary/media/Order/8900.1.pdf [Accessed: 2016, May]

[32] International Civil Aviation Organization, "Annex 11 to the Convention on International Civil Aviation - Air Traffic Services," *Thirteenth Edition, International Civil Aviation Organization*, no. July, 2001.

[33] A. D. Zeitlin, "Performance Tradeoffs and the Development of Standards," *Sense and Avoid in UAS: Research and Applications*, pp. 35–54, 2012.

[34] X. Prats, J. Ramírez, L. Delgado, and P. Royo, "Regulations and Requirements," *Sense and Avoid in UAS: Research and Applications*, p. 87, 2012.

[35] J. K. Kuchar, J. Andrews, A. Drumm, T. Hall, V. Heinz, S. Thompson, and J. Welch, "A Safety Analysis Process for the Traffic Alert and Collision Avoidance System (TCAS) and See-and-Avoid Systems on Remotely Piloted Vehicles," *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, no. September, 2004.

[36] M. Gigarjill, "Integration of Unmanned Aircraft Systems into the National Airspace System Concept of Operations," *Federal Aviation Administration*, pp. 1–120, 2012.

[37] National Transportation Safety Board (NTSB). Aviation Accident Statistics. [Online]. Available: http://www.ntsb.gov/aviation/Stats.htm [Accessed: 2016, July]

[38] K. Dalamagkidis, *Hazard and Safety Risk Modeling.* Springer Science & Business Media, 2015.

[39] P. Angelov, *Sense and Avoid in UAS: Research and Applications.* John Wiley & Sons, 2012.

[40] B. Korn, S. Tittel, and C. Edinger, "Stepwise Integration of UAS in non-segregated Airspace - The potential of Tailored UAS ATM Procedures," *ICNS 2012: Bridging CNS and ATM - Conference Proceedings*, 2012.

[41] International Civil Aviation Organization-Cir 328 AN/190. Unmanned Aircraft Systems (UAS). [Online]. Available: http://www.icao.int/Meetings/UAS/Documents/Circular%20328_en.pdf [Accessed: 2016, May]

[42] H. P. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover." DTIC Document, Tech. Rep., 1980.

[43] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[44] J. K. Kuchar and L. C. Yang, "A Review of Conflict Detection and Resolution Modeling Methods," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 1, no. 4, pp. 179–189, 2000.

[45] J. Krozel, M. E. Peters, and G. Hunter, "Conflict Detection and Resolution for Future Air Transportation Management," *NASA CR-97-205 944*, 1997.

[46] A. Warren, “Medium term Conflict Detection for Free Routing: Operational Concepts and Requirements Analysis,” in *Digital Avionics Systems Conference, 1997. 16th DASC., AIAA/IEEE*, vol. 2. IEEE, 1997, pp. 9–3.

[47] K. Zeghal, “A Review of Different Approaches Based on Force Fields for Airborne Conflict Resolution,” in *AIAA Guidance, navigation and control conference*, 1998, pp. 818–827.

[48] B. Albaker and N. Rahim, “A Survey of Collision Avoidance Approaches for Unmanned Aerial Vehicles,” in *technical postgraduates (TECHPOS), 2009 international conference for.* IEEE, 2009, pp. 1–7.

[49] E. A. Lester, “Benefits and Incentives for ADS-B equipage in the National Airspace System,” Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[50] J. Kuchar and A. C. Drumm, “The traffic alert and collision avoidance system,” *Lincoln Laboratory Journal*, vol. 16, no. 2, p. 277, 2007.

[51] T. B. Wolf, “DSpace@MIT : Aircraft Collision Avoidance using Monte Carlo Real-Time Belief Space Search,” no. 2007, 2009. [Online]. Available: http://dspace.mit.edu/handle/1721.1/54226 [Accessed: 2016, May]

[52] T. B. Wolf and M. J. Kochenderfer, “Aircraft Collision Avoidance Using Monte Carlo Real-Time Belief Space Search,” *Journal of Intelligent & Robotic Systems*, vol. 64, no. 2, pp. 277–298, 2011.

[53] M. J. Kochenderfer and H. J. D. Reynolds, *Decision Making Under Uncertainty: Theory and Application.* MIT press, 2015.

[54] C. Whitlock, “FAA Records detail hundreds of close calls between Airplanes and Drones,” *Washington Post*, 2015, [Online; accessed 8 April 2016].

[55] A. Mcfadyen, L. Mejias, P. Corke, and C. Pradalier, “Aircraft Collision Avoidance using Spherical Visual Predictive Control and Single Point Features,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on.* IEEE, 2013, pp. 50–56.

[56] S. Huh, S. Cho, Y. Jung, and D. H. Shim, “Vision-Based Sense-and-Avoid Framework for Unmanned Aerial Vehicles,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 4, pp. 3427–3439, Oct 2015.

[57] D. Accardo, G. Fasano, L. Forlenza, A. Moccia, and A. Rispoli, “Flight Test of a Radar-Based Tracking System for UAS Sense and Avoid,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 2, pp. 1139–1160, APRIL 2013.

[58] P. J. Nordlund and F. Gustafsson, “Probabilistic Noncooperative Near Mid-Air Collision Avoidance,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 2, pp. 1265–1276, 2011.

[59] P.-J. Nordlund and F. Gustafsson, “Probabilistic Conflict Detection for Piecewise Straight Paths,” *Linköping University Electronic Press, Tech. Rep. 2871*, 2008.

[60] F. Lindsten, P.-J. Nordlund, and F. Gustafsson, “Conflict Detection Metrics for Aircraft Sense and Avoid Systems,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2009, pp. 65–70.

[61] H. A. Blom and G. Bakker, “Conflict Probability and Incrossing Probability in Air Traffic Management,” in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 3. IEEE, 2002, pp. 2421–2426.

[62] V. P. Jilkov, X. R. Li, and J. H. Ledet, “Improved Estimation of Conflict Probability for Aircraft Collision Avoidance,” in *Information Fusion (FUSION), 2014 17th International Conference on.* IEEE, 2014, pp. 1–7.

[63] J. P. Chryssanthacopoulos and M. J. Kochenderfer, “Accounting for State Uncertainty in Collision Avoidance,” *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 951–960, 2011.

[64] F. Belkhouche, “Modeling and Calculating the Collision Risk for Air Vehicles,” *IEEE transactions on vehicular technology*, vol. 62, no. 5, pp. 2031–2041, 2013.

[65] H. A. Blom, J. Krystul, G. Bakker, M. B. Klompstra, and B. K. Obbink, “Free Flight Collision Risk Estimation by Sequential Monte Carlo Simulation,” *Stochastic hybrid systems*, pp. 249–281, 2006.

[66] O. Watkins and J. Lygeros, “Stochastic Reachability for Discrete Time Systems: An Application to Aircraft Collision Avoidance,” in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 5, Dec 2003, pp. 5314–5319 Vol.5.

[67] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, “A Probabilistic Approach to Aircraft Conflict Detection,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 1, no. 4, pp. 199–220, 2000.

[68] J. Krozel and M. Peters, “Strategic Conflict Detection and Resolution for Free Flight,” in *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, vol. 2, Dec 1997, pp. 1822–1828 vol.2.

[69] P. Goupil, “AIRBUS state of the art and practices on FDI and FTC in flight control system,” *Control Engineering Practice*, vol. 19, no. 6, pp. 524–539, 2011.

[70] B. d. et d'Analyses, "Accident on 27 November 2008 off the coast of Canet-Plage (66) to the Airbus A320-232 registered D-AXLA operated by XL Airways Germany," *Paris: BEA*, 2010.

[71] S. Ding, *Model-based fault diagnosis techniques: design schemes, algorithms, and tools.* Springer Science & Business Media, 2008.

[72] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A Survey of Fault Detection, Isolation, and Reconfiguration Methods," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 636–653, 2010.

[73] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 226, no. 10, pp. 1329–1360, 2012.

[74] S. Hansen and M. Blanke, "Diagnosis of Airspeed Measurement Faults for Unmanned Aerial Vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 1, pp. 224–239, January 2014.

[75] P. Freeman, P. Seiler, and G. J. Balas, "Air data system fault modeling and detection," *Control Engineering Practice*, vol. 21, no. 10, pp. 1290–1301, Oct. 2013.

[76] M. Oosterom, R. Babuska, and H. B. Verbruggen, "Soft Computing Applications in Aircraft Sensor Management and Flight Control Law Reconfiguration," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 32, no. 2, pp. 125–139, 2002.

[77] P. Goupil, "Oscillatory failure case detection in the A380 electrical flight control system by analytical redundancy," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 17, no. PART 1, pp. 681–686, 2007.

[78] R. Patton, "Fault detection and diagnosis in aerospace systems using analytical redundancy," *Computing & Control Engineering Journal*, vol. 2, no. 3, p. 127, 1991.

[79] Y. Zhang and X. R. Li, "Detection and Diagnosis of Sensor and Actuator Failures Using IMM Estimator," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 4, pp. 1293–1313, Oct 1998.

[80] I. Moir, A. Seabridge, and M. Jukes, *Civil Avionics Systems.* John Wiley & Sons, 2013.

[81] Federal Aviation Authority, “FAR/FAR 25, Airworthiness Standards: Transport Category Airplane.”

[82] P. Gurfil and H. Rotstein, “Partial Aircraft State Estimation from Visual Motion Using the Subspace Constraints Approach,” *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 5, pp. 1016–1028, 2001.

[83] V. G. Nair, M. Dileep, and V. George, “Aircraft Yaw Control System using LQR and Fuzzy Logic Controller,” *International Journal of Computer Applications*, vol. 45, no. 9, pp. 25–30, 2012.

[84] T. A. Talay, “Introduction to the Aerodynamics of Flight,” *Scientific and Technical Information Office National Aeronautics and Space Administration*, 1975.

[85] M. V. Cook, *Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability and Control.* Butterworth-Heinemann, 2012.

[86] P. H. Zipfel, *Modelling and Simulation of Aerospace Vehicle Dynamics, Second Edition.* American Institute of Aeronautics and Astronautics, 2007.

[87] a. De Marco, E. L. Duke, and J. S. Berndt, “A General Solution to the Aircraft Trim Problem,” *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit*, no. AUGUST 2007, pp. 1–40, 2007.

[88] G. M. Siouris, *Missile Guidance and Control Systems.* Springer Science & Business Media, 2004.

[89] J. Strickland, *Missile Flight Simulation.* Lulu Inc, 2012.

[90] S. Ganguli, A. Marcos, and G. Balas, “Reconfigurable LPV Control Design for Boeing 747-100/200 Longitudinal Axis,” in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 5. IEEE, 2002, pp. 3612–3617.

[91] M. E. Eshelby, “Aircraft Performance Theory and Practice,” *Aircraft Engineering and Aerospace Technology*, vol. 73, no. 2, 2000.

[92] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice.* Springer Science & Business Media, 2012.

[93] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications To Tracking and Navigation: Theory Algorithms and Software.* John Wiley & Sons, 2004.

[94] J. M. Pratt, *The Private Pilot’s Licence Cource: Navigation Meteorology.* Airplan Flight Equipment, 2003.

[95] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[96] J. Bijker and W. Steyn, “Kalman Filter configurations for a low-cost loosely integrated inertial navigation system on an airship,” *Control Engineering Practice*, vol. 16, no. 12, pp. 1509–1518, 2008.

[97] C. Wells, *The Kalman Filter in Finance.* Springer Science & Business Media, 2013, vol. 32.

[98] O. Sayadi and M. B. Shamsollahi, “ECG Denoising and Compression Using a Modified Extended Kalman Filter Structure,” *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 9, pp. 2240–2248, 2008.

[99] A. Farina, B. Ristic, and D. Benvenuti, “Tracking a Ballistic Target: Comparison of Several Nonlinear Filters,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 854–867, 2002.

[100] A. H. Jazwinski, *Stochastic Processes and Filtering Theory.* Courier Corporation, 2007.

[101] D. Simon, *Optimal State Estimation: Kalman, H infinity, and Nonlinear Approaches.* John Wiley & Sons, 2006.

[102] E. K. Chong and S. H. Zak, “An Introduction to Optimization,” 2013.

[103] D. Lerro and Y. Bar-Shalom, “Tracking With Debiased Consistent Converted Measurements Versus EKF,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 3, pp. 1015–1022, 1993.

[104] P. J. Costa, “Adaptive Model Architecture and Extended Kalman-Bucy Filters,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 2, pp. 525–533, 1994.

[105] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, “A New Approach for Filtering Nonlinear Systems,” in *American Control Conference, Proceedings of the 1995*, vol. 3. IEEE, 1995, pp. 1628–1632.

[106] E. A. Wan, R. V. D. Menve, and N. W. W. Rd, “The Unscented Kalman Filter for Nonlinear Estimation,” in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000.* IEEE, 2000, pp. 153–158.

[107] O. J. Woodman, “An Introduction to Inertial Navigation,” University of Cambridge, Tech. Rep., 2007.

[108] Federal Aviation Administration William J. Hughes. Low Cost Accurate Angle of Attack System. [Online]. Available: http://nar-associates.com/technical-flying/angle_of_attack/DFRDAS_AccurateLowCostAoA.pdf [Accessed: 2016, May]

[109] R. K. R. Mehra and J. Peschon, "An Innovations Approach to Fault Detection and Diagnosis in Dynamic Systems," *Automatica*, vol. 7, no. 5, pp. 637–640, 1971.

[110] K. Watanabe and S. G. Tzafestas, "Generalized pseudo-bayes estimation and detection for abruptly changing systems," *Journal of Intelligent and Robotic Systems*, vol. 7, no. 1, pp. 95–112, 1993.

[111] L. Cork and R. Walker, "Sensor Fault Detection for UAVs using a Nonlinear Dynamic Model and the IMM-UKF Algorithm," in *2007 Information, Decision and Control.* IEEE, Feb. 2007, pp. 230–235.

[112] Q. C. Q. Cheng, P. Varshney, and C. Belcastro, "Fault Detection in Dynamic Systems via Decision Fusion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 1, 2008.

[113] W. Liu, J. Wei, M. Liang, Y. Cao, and I. Hwang, "Multi-Sensor Fusion and Fault Detection using Hybrid Estimation for Air Traffic Surveillance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, pp. 2323–2339, 2013.

[114] J. Gertz, "Multisensor Surveillance for Improved Aircraft Tracking," *Lincoln laboratory journal*, vol. 2, no. 3, pp. 381–396, 1989.

[115] Y. C. Lee, "Analysis of Range and Position comparison methods as a means to provide GPS Integrity in the User Receiver," in *Proceedings of the Annual Meeting of the Institute of Navigation*, 1986, pp. 1–4.

[116] F. Goodrich, "Multilateration techniques for positive aircraft location and identification," in *Digital Avionics Systems Conference, 1995., 14th DASC.* IEEE, 1995, pp. 71–76.

[117] H. Blom, "An Efficient Filter for Abruptly Changing Systems," in *The 23rd IEEE Conference on Decision and Control*, no. 23, 1984, pp. 656–658.

[118] S. K. Au and J. L. Beck, "Estimation of small failure probabilities in high dimensions by subset simulation," *Probabilistic Engineering Mechanics*, vol. 16, no. 4, pp. 263–277, 2001.

[119] S. Au and J. Beck, "Subset Simulation and its Application to Seismic Risk Based on Dynamic Analysis," *Journal of Engineering Mechanics*, vol. 129, no. 8, pp. 901–917, 2003.

[120] S. Song, Z. Lu, and H. Qiao, "Subset simulation for structural reliability sensitivity analysis," *Reliability Engineering & System Safety*, vol. 94, no. 2, pp. 658–665, 2009.

[121] F. Miao and M. Ghosn, "Modified subset simulation method for reliability analysis of structural systems," *Structural Safety*, vol. 33, no. 4-5, pp. 251–260, 2011.

[122] B. Hua, Z. Bie, S. K. Au, W. Li, and X. Wang, "Extracting Rare Failure Events in Composite System Reliability Evaluation Via Subset Simulation," *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 753–762, 2015.

[123] J. M. Bourinet, F. Deheeger, and M. Lemaire, "Assessing small failure probabilities by combined subset simulation and Support Vector Machines," *Structural Safety*, vol. 33, no. 6, pp. 343–353, 2011.

[124] S.-K. Au and Y. Wang, *Engineering Risk Assessment with Subset Simulation.* John Wiley & Sons, 2014.

[125] E. Zio, *The Monte Carlo Simulation Method for System Reliability and Risk Analysis.* Springer, 2013.

[126] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[127] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[128] C. P. Robert and G. Casella, "The Metropolis–Hastings Algorithm," in *Monte Carlo Statistical Methods.* Springer, 2004, pp. 267–320.

[129] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, Oct 2003.

[130] P. Del Moral, A. Doucet, and A. Jasra, "Sequential Monte Carlo Samplers," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 3, pp. 411–436, 2006.